c-entron.NET - Dokumentation der Modul-Use-Cases

> **Generiert**: 2025-11-04
> **Version**: 2025 1.0.0.0
> **Zweck**: Umfassende Dokumentation aller Use Cases der c-entron Module

Inhaltsverzeichnis

1. Abrechnung (Billing)

1.1 Vertragsabrechnung (Contract Billing)

Module Path: `src/centron/Centron.WPF.UI/Modules/Finances/AutomatedBilling/`
Controller: AutomatedBillingAppModuleController.cs
ViewModel: AutomatedBillingViewModel.cs
Interface: IAutomatedBillingLogic.cs
Category: Abrechnung (Billing)
Module ID: `{ED303F19-86F3-4BE3-9F97-44CBD64D39FA}`
Description: Automatische Abrechnung von wiederkehrenden Verträgen und Abonnements
License: `LicenseGuids.ContractBilling` OR `LicenseGuids.Centron`
Rights: `UserRightsConst.Sales.AUTOMATED_BILLING`

Modul-Architektur

Dieses Modul nutzt einen **Wizard-basierten Ablauf** mit den folgenden Schritten:

1. **Schritt 1: Verträge auswählen**
   - Abrechnung-Datum Auswahl
   - Vertragsauswahl
2. **Schritt 2: Einstellungen**
   - Versandeinstellungen (Druck, E-Mail, etc.)
3. **Schritt 3: Verträge abrechnen**
   - Vorschau und Ausführung
4. **Abrechungshistorie**

Wizard-Seiten

1. `SeperatorWizardPageViewModel` - "Schritt 1: Verträge auswählen"
2. `BillingDateWizardPageViewModel` - Datums- und Filterauswahl
3. `ContractSelectionWizardPageViewModel` - Vertragsauswahl
4. `SeperatorWizardPageViewModel` - "Schritt 2: Einstellungen"
5. `SendSettingsWizardPageViewModel` - Versandmethoden-Einstellungen
6. `SeperatorWizardPageViewModel` - "Schritt 3: Verträge abrechnen"
7. `OverviewWizardPageViewModel` - Vorschau und Ausführung
8. `SeperatorWizardPageViewModel` - "Abrechungshistorie"
9. `BillingHistoryPageViewModel` - Bisherige Abrechnungsläufe anzeigen

Vollständige Use Cases

1. Vertrag-Auswahl und Filterung

1.1 Filter Contracts by Date

Purpose: Select contracts due for billing up to a specific date
Property: `ContractDateFilter` (DateTime)
Method: `LoadBasicContractsAsync(bool onlyDateFilter)`
Filter Field: `SearchBillingContractsFilter.DateTo`
Default: Today's date
Code Reference: AutomatedBillingViewModel.cs:194-198

```
public DateTime ContractDateFilter
{
    get => this._contractDateFilter;
    set => this.SetProperty(ref this._contractDateFilter, value, nameof(this.ContractDateFilter));
}
```

1.2 Filter Contracts by Customer

Purpose: Bill only specific customers
Property: `SelectedCustomers` (ObservableCollection)
UI: Multi-select list with contract count per customer
Logic: Filters contracts where `CustomerI3D` matches selected customers
Code Reference: AutomatedBillingViewModel.cs:594-601

```
if (SelectedCustomers.Count > 0)
{
    if (this.SelectedCustomers.Count != this.Customers.Count)
        result = result.Where(f => SelectedCustomers.Select(s => s.HolderID).Contains(f.CustomerI3D)).ToList();
}
else { return result.Where(f => f.I3D < 0).ToList(); }
```

1.3 Filter Contracts by Contract Type (Vertragsart)

Purpose: Bill only specific contract types
Property: `SelectedContractKinds` (ObservableCollection)
UI: Multi-select list with contract count per type
Logic: Filters by `ContractKindI3D`
Code Reference: AutomatedBillingViewModel.cs:603-610

1.4 Filter Contracts by Branch (Filiale)

Purpose: Bill contracts for specific branches
Property: `SelectedBranches` (ObservableCollection)
UI: Multi-select list with contract count per branch
Logic: Filters by `BranchI3D`
Code Reference: AutomatedBillingViewModel.cs:612-618

1.5 Filter by Calculation Type

Purpose: Select automatic, manual, or on-demand billing contracts
Types:

- **Auto**: Automatically billed contracts (`ContractCalculationKind.Auto`)
- **Manual**: Manually triggered billing (`ContractCalculationKind.Manual`)
- **Need**: On-demand billing (`ContractCalculationKind.Need`)
  - Click counters (`ContractNeedCalcKind.Click`)
  - Contingent limits (`ContractNeedCalcKind.ContingentLimit`)
  - Dynamic billing (`ContractNeedCalcKind.Dynamic`)

Implementation: Enum flags in `SearchBillingContractsFilter.CalculationKind`
Code Reference: AutomatedBillingViewModel.cs:427-430

```
if ((_contractsFilter.CalculationKind & ContractCalculationKind.Manual) != ContractCalculationKind.Manual)
    flt.CalculationKind = ContractCalculationKind.Auto | ContractCalculationKind.Need;
else flt.CalculationKind = ContractCalculationKind.Auto | ContractCalculationKind.Need | ContractCalculationKind.Manual;
```

1.6 Filter by Contract Extra Kind

Purpose: Select specific contract categories
Types:

- **Easy Contracts**: Standard recurring contracts (`IsEasyContractSelected`)
- **Click Counter Contracts**: Usage-based billing (`IsClickContractSelected`)
- **Contingent Contracts**: Volume-based contracts (`IsContingentContractSelected`)

Implementation: Boolean flags in filter
Code Reference: AutomatedBillingViewModel.cs:585-592

```
if (!filter.IsEasyContractSelected || !filter.IsClickContractSelected || !filter.IsContingentContractSelected)
{
    result = result.Where(f => (filter.IsEasyContractSelected && f.ExtraKind == ContractExtraKind.Easy) ||
                               (filter.IsClickContractSelected && ((int)f.ExtraKind & (int)ContractExtraKind.ClickDevice) == (int)ContractExtraKin
                               (filter.IsContingentContractSelected && ((int)f.ExtraKind & (int)ContractExtraKind.Contingent) == (int)ContractExtr
}
```

1.7 Filter by Billing Interval

Purpose: Select contracts by billing frequency
Properties:

- `IsBillingIntervalActive` (bool)
- `IntervalKind` (nullable)
- `IntervalDuration` (nullable)

Options: Monthly, quarterly, yearly, etc.
Code Reference: AutomatedBillingViewModel.cs:620-624

```
if (filter.IsBillingIntervalActive && filter.IntervalKind.HasValue && filter.IntervalDuration.HasValue)
{
    result = result.Where(f => f.BillingIntervalDuration == filter.IntervalDuration && f.BillingIntervalKind == (BillingIntervalKinds)filter.Inte
}
```

1.8 Filter by Advance/Arrear Billing

Purpose: Distinguish between prepaid and postpaid billing
Filter Field: `AdvanceBilling` (nullable bool)
Types:

- `BillingKinds.Billingadvance` : Prepaid billing (Vorauskasse)
- `BillingKinds.Billingarrear` : Postpaid billing (Nachkasse)

Code Reference: AutomatedBillingViewModel.cs:644-648

```
if (filter.CalculationKind.HasFlag(ContractCalculationKind.Auto) && filter.AdvanceBilling != null)
{
    if ((bool)filter.AdvanceBilling) result = result.Where(f => f.BillingKind == BillingKinds.Billingadvance && f.CalculationKind == ContractCalc
    else result = result.Where(f => f.BillingKind == BillingKinds.Billingarrear && f.CalculationKind == ContractCalculationKind.Auto).ToList();
}
```

1.9 Filter Collect Invoices (Sammelrechnungen)

Purpose: Handle grouped invoices
Filter Field: `CollectOnly` (nullable bool)
Logic: Filters by `CollectInvoice` value
Code Reference: AutomatedBillingViewModel.cs:650-654

```
if (filter.CollectOnly != null)
{
    if ((bool)filter.CollectOnly) result = result.Where(f => f.CollectInvoice.GetValueOrDefault() > 0).ToList();
    else result = result.Where(f => f.CollectInvoice.GetValueOrDefault() == 0).ToList();
}
```

2. Click Counter Management (Usage-Based Billing)

2.1 Load Current Counter States

Purpose: Retrieve current meter readings for click counter devices
Method: `LoadCounterAsync(List<int> contractsI3D)`
Returns: Sets `CounterToContracts` property
DTO: `AutomaticFacturaCounterToContractDTO`
Use Cases: Copy machines, printers, industrial equipment with usage meters
Code Reference: AutomatedBillingViewModel.cs:459-477

```
public async Task LoadCounterAsync(List<int> contractsI3D)
{
    try
    {
        var counterToContractsResult = await ClassContainer.Instance.WithInstance((IAutomatedBillingLogic automatedBillingLogic) => automatedBill
        if (counterToContractsResult.Status == ResultStatus.Error)
        {
            return;
        }
        this.CounterToContracts = (List<AutomaticFacturaCounterToContractDTO>)counterToContractsResult.Data;
        this.CounterHasChanged = false;
    }
    finally
    {
        IsProcessing = false;
    }
}
```

Interface Method: IAutomatedBillingLogic.cs:27

```
Task<Result<IList<AutomaticFacturaCounterToContractDTO>>> GetCurrentCounterState(List<int> contractsI3D);
```

2.2 Get Counter History

Purpose: View historical counter readings
Method: `IAutomatedBillingLogic.GetCounterHistory(List<string> lstParam)`
Returns: `AutomaticFacturaCounterHistoryDTO` list
Use Case: Track counter reading trends, detect anomalies
Interface Reference: IAutomatedBillingLogic.cs:35

```
Task<Result<IList<AutomaticFacturaCounterHistoryDTO>>> GetCounterHistory(List<string> lstParam);
```

2.3 Update Counter Values

Purpose: Manually enter or import new meter readings
Method: `IAutomatedBillingLogic.StoreCounterState(List<AutomaticFacturaCounterToContractDTO> newCountersState)`
Validation: `CounterHasChanged` flag prevents billing until counters are saved
Property: `CounterHasChanged` (bool)
Code Reference: AutomatedBillingViewModel.cs:230-238

```
public bool CounterHasChanged
{
    get => this._counterHasChanged;
    set
    {
        this._counterHasChanged = value;
        this.IsBilligAllowed = !value;  // Prevent billing if counters changed
    }
}
```

Interface Reference: IAutomatedBillingLogic.cs:38

```
Task<Result<bool>> StoreCounterState(List<AutomaticFacturaCounterToContractDTO> newCountersState);
```

2.4 Get Counter by Barcode

Purpose: Scan device barcode to retrieve counter info
Method: `IAutomatedBillingLogic.GetCounterToBarcode(IList<string> barcodes)`
Returns: `AutomaticFacturaCounterToContractDTO` list
Use Case: Quick counter lookup via barcode scanner
Interface Reference: IAutomatedBillingLogic.cs:30

```
Task<Result<IList<AutomaticFacturaCounterToContractDTO>>> GetCounterToBarcode(IList<string> barcodes);
```

2.5 Import Counter Values (Rivebird Integration)

Purpose: Automatically import counter readings from external systems
Method: `IAutomatedBillingLogic.RivebirdImportValues()`
Returns: `AutomaticFacturaCounterImportDTO` list
Integration: Rivebird IoT platform for automatic meter reading
Interface Reference: IAutomatedBillingLogic.cs:55

```
Task<Result<IList<AutomaticFacturaCounterImportDTO>>> RivebirdImportValues();
```

2.6 Manage Counter Types

Purpose: Define counter categories (black/white, color, scan, etc.)
Command: `ShowCounterTypesCommand`
Rights Check: `UserRightsConst.Masterdata.Contracts.COUNTER_TYPES`
Dialog: Opens `CounterTypesViewModel` modal
Code Reference: AutomatedBillingViewModel.cs:517-524

```
private bool CanShowCounterTypes()
{
    return CentronCache.Instance.CurrentUserAppRights.Any(f => f.I3D == UserRightsConst.Masterdata.Contracts.COUNTER_TYPES);
}
private async Task ShowCounterTypes()
{
    await CentronApplication.Instance.DialogManager.ShowDialogWindow(new CounterTypesViewModel(), DialogKind.Modal);
}
```

2.7 Get Counter Kinds

Purpose: Retrieve available counter type definitions
Method: `IAutomatedBillingLogic.GetCounterKinds()`
Returns: `CounterKindDTO` list
Interface Reference: IAutomatedBillingLogic.cs:39

```
Task<Result<IList<CounterKindDTO>>> GetCounterKinds();
```

2.8 Get Counter Score Reasons

Purpose: Retrieve reasons for counter discrepancies
Method: `IAutomatedBillingLogic.GetCounterScoreReasons()`
Returns: List of string reasons
Use Case: Explain why counter reading differs from expected value
Interface Reference: IAutomatedBillingLogic.cs:36

```
Task<Result<IList<string>>> GetCounterScoreReasons();
```

2.9 Get Input Counter State with Filter

Purpose: Query counter states with advanced filtering
Method: `IAutomatedBillingLogic.GetInputCounterState(SearchCounterStateFilter filter)`
Returns: `AutomaticFacturaCounterToContractDTO` list
Use Case: Search counters by date range, device type, etc.
Interface Reference: IAutomatedBillingLogic.cs:28

```
Task<Result<IList<AutomaticFacturaCounterToContractDTO>>> GetInputCounterState(SearchCounterStateFilter filter);
```

2.10 Create Master Data from Import

Purpose: Generate device master data from counter import
Method: `IAutomatedBillingLogic.CreateMasterDataListFromImport(List<AutomaticFacturaCounterImportDTO> forDeviceDTo)`
Returns: `DeviceClickCounterDTO` list
Use Case: Auto-create device records from first import
Interface Reference: IAutomatedBillingLogic.cs:33

```
Task<Result<IList<DeviceClickCounterDTO>>> CreateMasterDataListFromImport(List<AutomaticFacturaCounterImportDTO> forDeviceDTo);
```

3. Contract Position Management

3.1 Get Contract Items

Purpose: View all line items for a contract
Method: `IAutomatedBillingLogic.GetContractItems(int contractI3D)`
Returns: `ReceiptContractPosDTO` list
Use Case: Display contract positions for review
Interface Reference: IAutomatedBillingLogic.cs:24

```
Task<Result<IList<ReceiptContractPosDTO>>> GetContractItems(int contractI3D);
```

3.2 Check Positions Without Quantity

Purpose: Identify contract items missing quantity values
Property: `PositionsWithoutQuantity` (List)
Use Case: Validate contracts before billing
Code Reference: AutomatedBillingViewModel.cs:170-174

```
public List<EmptyCountContractPosition> PositionsWithoutQuantity
{
    get => this._positionsWithoutQuantity;
    set => this.SetProperty(ref this._positionsWithoutQuantity, value, nameof(this.PositionsWithoutQuantity));
}
```

3.3 Get Position Deficit by Barcode

Purpose: Find missing barcodes in contract positions
Method: `IAutomatedBillingLogic.GetPositionDeficitBarcode(List<int> contractsI3D, bool isBarcodeDeficit)`
Returns: `ContractPositions` list
Use Case: Identify positions needing barcode assignment
Interface Reference: IAutomatedBillingLogic.cs:46

```
Task<Result<IList<ContractPositions>>> GetPositionDeficitBarcode(List<int> contractsI3D, bool isBarcodeDeficit);
```

3.4 Get Partible Article Positions

Purpose: Handle partially billable items
Method: `IAutomatedBillingLogic.GetContractPartibleArticlePositionen(List<int> contractsI3D)`
Returns: `ContractPartibleArticlePositionen` list
Use Case: Bill only consumed portion of items
Interface Reference: IAutomatedBillingLogic.cs:47

```
Task<Result<IList<ContractPartibleArticlePositionen>>> GetContractPartibleArticlePositionen(List<int> contractsI3D);
```

4. Billing Execution

4.1 Create Invoices from Contracts

Purpose: Generate invoices for selected contracts
Method:
`IAutomatedBillingLogic.CreateInvoiceToContractComplete(IList<ContractToInvoiceParam> billingParams, ReceiptMailTemplateDTO mailTemplate, bool bPrev`
Parameters:

- `billingParams` : Billing parameters per contract
- `mailTemplate` : Email template (if sending via email)
- `bPreview` : Preview mode flag

Returns: `InvoiceToContractResult` with success/error details
Interface Reference: IAutomatedBillingLogic.cs:26

```
Task<Result<InvoiceToContractResult>> CreateInvoiceToContractComplete(IList<ContractToInvoiceParam> billingParams, ReceiptMailTemplateDTO mailTem
```

4.2 Set Receipt Date

Purpose: Define the invoice date
Property: `ReceiptDate` (DateTime)
Default: Today's date
Code Reference: AutomatedBillingViewModel.cs:188-192

```
public DateTime ReceiptDate
{
    get => this._receiptDate;
    set => this.SetProperty(ref this._receiptDate, value, nameof(this.ReceiptDate));
}
```

Initialized to: `this.ReceiptDate = DateTime.Today;` (Line 358)

4.3 Preview Invoices Before Creation

Purpose: Review invoices before final generation
Property: `WithPreview` (bool)
Page: `ContractPreviewViewModel` in wizard

Default: `true`
Code Reference: AutomatedBillingViewModel.cs:264-268

```
public bool WithPreview
{
    get => this._withPreview;
    set => this.SetProperty(ref this._withPreview, value, nameof(this.WithPreview));
}
```

Initialized to: `this.WithPreview = true;` (Line 375)

4.4 Lock/Unlock Billing

Property: `IsLocked` (bool)
Purpose: Prevent accidental billing execution
Code Reference: AutomatedBillingViewModel.cs:252-256

```
public bool IsLocked
{
    get => this._isLocked;
    set => this.SetProperty(ref this._isLocked, value, nameof(this.IsLocked));
}
```

4.5 Billing Allowed Check

Property: `IsBilligAllowed` (bool)
Purpose: Enable/disable billing based on validation
Logic: Disabled when `CounterHasChanged` is true
Code Reference: AutomatedBillingViewModel.cs:212-216

```
public bool IsBilligAllowed
{
    get => this._isBilligAllowed;
    set => this.SetProperty(ref this._isBilligAllowed, value, nameof(this.IsBilligAllowed));
}
```

5. Invoice Delivery Settings

5.1 Select Send Type

Purpose: Choose how invoices are delivered
Property: `SelectedSendType` (AssetSendType enum)
Options:

- **Print**: Print to local/network printer
- **Email**: Send via email
- **Mail**: Physical mail
- **Portal**: Customer portal access
- **Other custom types**

Default: `AssetSendType.Print`
Code Reference: AutomatedBillingViewModel.cs:182-186

```
public AssetSendType SelectedSendType
{
    get => this._selectedSendType;
    set => this.SetProperty(ref _selectedSendType, value, nameof(SelectedSendType));
}
```

Initialized to: `this.SelectedSendType = AssetSendType.Print;` (Line 360)

5.2 Override Send Type per Contract

Purpose: Force specific delivery method regardless of customer preference
Property: `OverwriteSendType` (bool)
Use Case: Special billing runs requiring specific delivery
Default: `false`
Code Reference: AutomatedBillingViewModel.cs:258-262

```
public bool OverwriteSendType
{
    get => this._overwriteSendType;
    set => this.SetProperty(ref this._overwriteSendType, value, nameof(this.OverwriteSendType));
}
```

Initialized to: `this.OverwriteSendType = false;` (Line 361)

5.3 Select Printer

Purpose: Choose target printer for print delivery
Property: `SelectedPrinter` (string)
Code Reference: AutomatedBillingViewModel.cs:200-204

```
public string SelectedPrinter
{
    get => this._selectedPrinter;
    set => this.SetProperty(ref this._selectedPrinter, value, nameof(this.SelectedPrinter));
}
```

5.4 Direct Mail Sending

Purpose: Send emails immediately without user confirmation
Property: `DirectMail` (bool)
Default: `true`
Code Reference: AutomatedBillingViewModel.cs:224-228

```
public bool DirectMail
{
    get => this._directMail;
    set => this.SetProperty(ref this._directMail, value, nameof(this.DirectMail));
}
```

Initialized to: `this.DirectMail = true;` (Line 376)

6. Email Composition

6.1 Set Email Recipients

Properties:

- `To` (ObservableCollection): Primary recipients
- `CC` (ObservableCollection): Carbon copy
- `BCC` (ObservableCollection): Blind carbon copy

Use Case: Define email recipients for invoice delivery
Code Reference: AutomatedBillingViewModel.cs:270-286

```
public ObservableCollection<string> To
{
    get => this._to ?? (this._to = new ObservableCollection<string>());
    set => this.SetProperty(ref this._to, value, nameof(this.To));
}

public ObservableCollection<string> CC
{
    get => this._cc ?? (this._cc = new ObservableCollection<string>());
    set => this.SetProperty(ref this._cc, value, nameof(this.CC));
}

public ObservableCollection<string> BCC
{
    get => this._bcc ?? (this._bcc = new ObservableCollection<string>());
    set => this.SetProperty(ref this._bcc, value, nameof(this.BCC));
}
```

6.2 Set Email Subject

Property: `MailSubject` (string)
Use Case: Customize email subject line
Code Reference: AutomatedBillingViewModel.cs:288-292

```
public string MailSubject
{
    get => this._mailSubject;
    set => this.SetProperty(ref this._mailSubject, value, nameof(this.MailSubject));
}
```

6.3 Set Email Body

Property: `Body` (string)
Use Case: Add custom message text
Code Reference: AutomatedBillingViewModel.cs:294-298

```
public string Body
{
    get => this._body;
    set => this.SetProperty(ref this._body, value, nameof(this.Body));
}
```

6.4 Add Email Attachments

Property: `Attachments` (IList)
Use Case: Attach additional documents (terms, contracts, etc.)
Code Reference: AutomatedBillingViewModel.cs:300-304

```
public IList<DocumentDTO> Attachments
{
    get => this._attachments;
    set => this.SetProperty(ref this._attachments, value, nameof(this.Attachments));
}
```

Initialized to: `private IList<DocumentDTO> _attachments = new List<DocumentDTO>();` (Line 100)

7. Special Article Import (Dynamic Data)

7.1 Create Special Articles from Import

Purpose: Import dynamic contract items (e.g., usage data, licenses)
Method: `IAutomatedBillingLogic.CreateSpecialArticleToContract(IList<SpecialArticleToContractHeadImport> headImportList)`
Returns: `SpecialArticleToContractHeadImportResultDTO`
Use Case: Add usage-based or variable contract items
Interface Reference: IAutomatedBillingLogic.cs:41

```
Task<Result<SpecialArticleToContractHeadImportResultDTO>> CreateSpecialArticleToContract(IList<SpecialArticleToContractHeadImport> headImportList
```

7.2 Save/Update Special Article Heads

Purpose: Manage special article configurations
Method: `IAutomatedBillingLogic.SaveOrUpdateSpecialArticleToContractHead(SpecialArticleToContractHeadDTO headDTO)`
Returns: `SpecialArticleToContractHeadDTO`
Interface Reference: IAutomatedBillingLogic.cs:42

```
Task<Result<SpecialArticleToContractHeadDTO>> SaveOrUpdateSpecialArticleToContractHead(SpecialArticleToContractHeadDTO headDTO);
```

7.3 Delete Special Article Heads

Purpose: Remove special article configurations
Method: `IAutomatedBillingLogic.DeleteSpecialArticleToContractHead(List<int> headI3Ds)`
Interface Reference: IAutomatedBillingLogic.cs:43

```
Task<Result> DeleteSpecialArticleToContractHead(List<int> headI3Ds);
```

7.4 Search Special Articles

Purpose: Find special article configurations by filter
Method: `IAutomatedBillingLogic.SearchSpecialArticleToContractHead(SearchSpecialArticleToContractHeadFilter headFilter)`
Returns: `SpecialArticleToContractHeadDTO` list
Interface Reference: IAutomatedBillingLogic.cs:44

```
Task<Result<IList<SpecialArticleToContractHeadDTO>>> SearchSpecialArticleToContractHead(SearchSpecialArticleToContractHeadFilter headFilter);
```

7.5 Import from MSP Evaluation

Purpose: Create special articles from MSP (Managed Service Provider) data
Method: `IAutomatedBillingLogic.CreateSpecialArticleToContractFromMspEvaluation(MspEvaluationCompensationItemDTO compensationItemDto)`
Returns: `SpecialArticleToContractHeadImportResultDTO`
Use Case: Integrate MSP monitoring data into billing
Interface Reference: IAutomatedBillingLogic.cs:65

```
Task<Result<SpecialArticleToContractHeadImportResultDTO>> CreateSpecialArticleToContractFromMspEvaluation(MspEvaluationCompensationItemDTO compen
```

8. Billing History & Results

8.1 View Billing Results

Purpose: Review created invoices and their status
Property: `BillingResults` (ObservableCollection)
Page: `BillingHistoryPageViewModel`
Code Reference: AutomatedBillingViewModel.cs:176-180

```
public ObservableCollection<InvoiceToContractResult> BillingResults
{
    get => this._billingResults;
    set => SetProperty(ref this._billingResults, value, nameof(this.BillingResults));
}
```

8.2 Save Billing Results

Purpose: Export billing results
Command: `cmdSaveResult`
Method: `SaveBillingResults()`
Code Reference: AutomatedBillingViewModel.cs:919-922

```
public void SaveBillingResults()
{
    MessageBox.Show(BillingResults.Count > 0 ? BillingResults.Count.ToString() : "0");
}
```

Command initialization: `this.cmdSaveResult = new DelegateCommand(SaveBillingResults);` (Line 401)

8.3 Store Billing Log

Purpose: Log invoice creation events
Method: `IAutomatedBillingLogic.StoreLog(int invoiceI3D, string reportName, string description, AssetSendType sendType)`
Use Case: Audit trail for billing operations
Interface Reference: IAutomatedBillingLogic.cs:49

```
Task<Result> StoreLog(int invoiceI3D, string reportName, string descriptiom, AssetSendType sendType);
```

8.4 Store Billing Result Status

Purpose: Save success/error status per contract
Method: `IAutomatedBillingLogic.StoreBillingResult(int contractID, int invoiceID, int status, string result, string comment)`
Interface Reference: IAutomatedBillingLogic.cs:57

```
Task<Result> StoreBillingResult(int contractID, int invoiceID, int status, string result, string comment);
```

8.5 Load Historical Billing Results

Purpose: Query past billing runs
Method: `IAutomatedBillingLogic.LoadBillinResult(DateTime dtFrom, DateTime dtTo, List<int> contractI3Ds)`
Parameters:

- `dtFrom` : Start date
- `dtTo` : End date
- `contractI3Ds` : Optional contract filter

Returns: `ContractBillingResultDTO` list
Interface Reference: IAutomatedBillingLogic.cs:59

```
Task<Result<IList<ContractBillingResultDTO>>> LoadBillinResult(DateTime dtFrom, DateTime dtTo, List<int> contractI3Ds);
```

9. Report Management

9.1 Get Reports by Send Type

Purpose: Retrieve available invoice templates per delivery method
Method: `IAutomatedBillingLogic.GetReportToSendType(string reportGroupGuid)`

Returns: `ReportToSendTypeToGroup` list
Use Case: Select appropriate template for print/email/portal
Interface Reference: IAutomatedBillingLogic.cs:51

```
Task<Result<IList<ReportToSendTypeToGroup>>> GetReportToSendType(string reportGroupGuid);
```

9.2 Get Report Parameters

Purpose: Retrieve report template parameters
Method: `IAutomatedBillingLogic.GetReportParams(string reportGroupGuid)`
Returns: `ReportGroupParameterDTO` list
Use Case: Provide input parameters to report templates
Interface Reference: IAutomatedBillingLogic.cs:52

```
Task<Result<IList<ReportGroupParameterDTO>>> GetReportParams(string reportGroupGuid);
```

10. Contract Type & Settings Management

10.1 Show Contract Types

Purpose: Open contract type management
Command: `ShowContractTypesCommand`
Navigation: Opens `ContractTypeAppModuleController`
Rights Check: `UserRightsConst.Masterdata.Contracts.CONTRACT_TYPES`
Code Reference: AutomatedBillingViewModel.cs:508-515

```
private bool CanShowContractTypes()
{
    return CentronCache.Instance.CurrentUserAppRights.Any(f => f.I3D == UserRightsConst.Masterdata.Contracts.CONTRACT_TYPES);
}
private void ShowContractTypes()
{
    CentronApplication.Instance.Modules.OpenModule(new ContractTypeAppModuleController());
}
```

Command initialization: `this.ShowContractTypesCommand = new DelegateCommand(this.ShowContractTypes, this.CanShowContractTypes);` (Line 398)

10.2 Show Counter Types

Purpose: Manage click counter categories
Command: `ShowCounterTypesCommand`
Rights Check: `UserRightsConst.Masterdata.Contracts.COUNTER_TYPES`
Dialog: Opens `CounterTypesViewModel` modal
Code Reference: AutomatedBillingViewModel.cs:517-524

```
private bool CanShowCounterTypes()
{
    return CentronCache.Instance.CurrentUserAppRights.Any(f => f.I3D == UserRightsConst.Masterdata.Contracts.COUNTER_TYPES);
}
private async Task ShowCounterTypes()
{
    await CentronApplication.Instance.DialogManager.ShowDialogWindow(new CounterTypesViewModel(), DialogKind.Modal);
}
```

Command initialization: `this.ShowCounterTypesCommand = new AsyncCommand(this.ShowCounterTypes, this.CanShowCounterTypes);` (Line 399)

10.3 Show Termination Types

Purpose: Manage contract termination reasons
Command: `ShowTerminationTypesCommand`
Rights Check: `UserRightsConst.Masterdata.Contracts.TERMINATION_TYPES`
Dialog: Opens `TerminationTypesViewModel` modal
Code Reference: AutomatedBillingViewModel.cs:526-533

```
private bool CanShowTerminationTypes()
{
    return CentronCache.Instance.CurrentUserAppRights.Any(f => f.I3D == UserRightsConst.Masterdata.Contracts.TERMINATION_TYPES);
}
private async Task ShowTerminationTypes()
{
    await CentronApplication.Instance.DialogManager.ShowDialogWindow(new TerminationTypesViewModel(), DialogKind.Modal);
}
```

Command initialization: `this.ShowTerminationTypesCommand = new AsyncCommand(this.ShowTerminationTypes, this.CanShowTerminationTypes);` (Line 400)

11. Customer-Specific Features

11.1 Show Customer Invoice Info

Purpose: Display customer-specific billing notes/warnings
Method: `GetInvoiceInfo(int customerI3D)`
Implementation: Loads RTF-formatted invoice info from customer record
UI: Shows confirmation dialog with invoice notes
Property: `CustomerInvoiceInfo` (List)
Code Reference: AutomatedBillingViewModel.cs:958-986

```
public async Task GetInvoiceInfo(int customerI3D)
{
    if (CustomerInvoiceInfo == null ) CustomerInvoiceInfo = new List<IntStringList>();
    var cashInfo = CustomerInvoiceInfo.Where(f => f.ID == customerI3D).FirstOrDefault();
    string message = string.Empty;
    if (cashInfo != null)
    {
        if (string.IsNullOrWhiteSpace(cashInfo.Value)) return;
        message = cashInfo.Value;
    }
    if (cashInfo == null)
    {
        var customer = await ClassContainer.Instance.WithInstance(async (ICustomerLogic logic) => await logic.GetCustomerByI3DAsync(customerI3D))
        IntStringList customerInfo = new IntStringList() {ID = customer.I3D, Value = customer.InvoiceInfo };
        CustomerInvoiceInfo.Add(customerInfo);
        message = customerInfo.Value;
    }

    if (!string.IsNullOrWhiteSpace(message))
    {
        var richEditDocumentServer = new RichEditDocumentServer();
        richEditDocumentServer.Document.RtfText = message;

        if (!string.IsNullOrWhiteSpace(richEditDocumentServer.Document.Text))
        {
            await CentronApplication.Instance.DialogManager.ShowConfirmationDialog(richEditDocumentServer.Document.Text, "Rechnungsinfo");
        }
    }
}
```

11.2 Handle Corporation Billing

Purpose: Support multi-company billing
Property: `HasCorporation` (bool)
Logic: Detects contracts with corporate structure
Code Reference: AutomatedBillingViewModel.cs:946-956

```
public Boolean HasCorporation
{
    get
    {
        return this._hasCorporation;
    }
    set
    {
        this.SetProperty(ref this._hasCorporation, value, () => this.HasCorporation);
    }
}
```

Set when loading contracts: `HasCorporation = this.Contracts.Any(f => f.WithCorporation);` (Line 445)

11.3 Show Invoice Info Flag

Property: `WithoutInvoiceinfo` (bool)
Purpose: Toggle customer invoice info display
Code Reference: AutomatedBillingViewModel.cs:322

```
public bool WithoutInvoiceinfo { get; set; }
```

12. Maintenance & Admin Functions

12.1 Update Contract Font Family

Purpose: Change font for contract documents
Method: `IAutomatedBillingLogic.UpdateContractFontFamily(string newFontFamily, int? contractNumber)`
Returns: `int` (affected records count)
Use Case: Standardize document formatting
Interface Reference: IAutomatedBillingLogic.cs:63

```
Task<Result<int>> UpdateContractFontFamily(string newFontFamily, int? contractNumber);
```

12.2 Repair Missing Richtext in Contract Items

Purpose: Fix corrupted rich text fields
Method: `IAutomatedBillingLogic.RepairContractItemsMissingRichtext()`
Returns: `int` (repaired records count)
Use Case: Data cleanup and migration
Interface Reference: IAutomatedBillingLogic.cs:64

```
Task<Result<int>> RepairContractItemsMissingRichtext();
```

12.3 Get Counter Import Sources

Purpose: List available counter import integrations
Method: `IAutomatedBillingLogic.GetCounterImports()`
Returns: `IntStringList` of import sources
Use Case: Select import provider (Rivebird, manual, etc.)
Interface Reference: IAutomatedBillingLogic.cs:61

```
Task<Result<IList<IntStringList>>> GetCounterImports();
```

12.4 Get Article I3D by Codes

Purpose: Resolve article identifiers from codes
Method: `IAutomatedBillingLogic.GetArticleI3D(IList<string> codes)`
Returns: `IntStringList` mapping codes to I3Ds
Use Case: Import validation and lookup
Interface Reference: IAutomatedBillingLogic.cs:31

```
Task<Result<IList<IntStringList>>> GetArticleI3D(IList<string> codes);
```

13. Wizard Navigation & Flow Control

13.1 Navigate Between Wizard Pages

Implementation: `WizardHelperViewModel<AutomatedBillingViewModel>`
Property: `Wizard` (WizardHelperViewModel)
Pages:

1. Separator: "Schritt 1: Verträge auswählen"
2. `BillingDateWizardPageViewModel` - Date and filter selection
3. `ContractSelectionWizardPageViewModel` - Contract list
4. Separator: "Schritt 2: Einstellungen"
5. `SendSettingsWizardPageViewModel` - Delivery settings
6. Separator: "Schritt 3: Verträge abrechnen"
7. `OverviewWizardPageViewModel` - Preview and execute
8. Separator: "Abrechungshistorie"
9. `BillingHistoryPageViewModel` - Historical results

Code Reference: AutomatedBillingViewModel.cs:377-393

```
this.Wizard = new WizardHelperViewModel<AutomatedBillingViewModel>(this);
this.Wizard.PropertyChanged += WizardOnPropertyChanged;

this.Wizard.Pages.Add(new SeperatorWizardPageViewModel("Schritt 1: Verträge auswählen"));
this.Wizard.Pages.Add(new BillingDateWizardPageViewModel());
this.Wizard.Pages.Add(new ContractSelectionWizardPageViewModel());

this.Wizard.Pages.Add(new SeperatorWizardPageViewModel("Schritt 2: Einstellungen"));
this.Wizard.Pages.Add(new SendSettingsWizardPageViewModel(this.SelectedSendType, this.OverwriteSendType));
this.Wizard.Pages.Add(new SeperatorWizardPageViewModel("Schritt 3: Verträge abrechnen"));
this.Wizard.Pages.Add(new OverviewWizardPageViewModel());
this.Wizard.Pages.Add(new SeperatorWizardPageViewModel("Abrechungshistorie"));
this.Wizard.Pages.Add(new BillingHistoryPageViewModel());

//navigating to the first page that is not disabled
this.Wizard.CurrentPage = this.Wizard.Pages.First(f => f.IsEnabled);
```

13.2 Auto-Start Mode

Property: `IsAutoStart` (bool)
Purpose: Automatically load contracts on page entry
Behavior: Loads basic contracts with date filter only
Default: `true`
Code Reference: AutomatedBillingViewModel.cs:122-131

```
public bool IsAutoStart
{
    get { return this._isAutoStart; }
    set
    {
        this.SetProperty(ref this._isAutoStart, value, () => this.IsAutoStart);
        if (value && (BasicContracts?.Count ?? 0) == 0)
            _ = LoadBasicContractsAsync(onlyDateFilter: true);
    }
}
```

Initialized to: `IsAutoStart = true;` (Line 353)

13.3 Filter Change Detection

Purpose: Detect when filter changes require data reload
Method: `CheckNeedRefresh(bool manuellOnly)`
Property: `FilterHasChanged` (bool)
Logic: Compares current filter with stored filter
Code Reference: AutomatedBillingViewModel.cs:826-884

```
public bool CheckNeedRefresh(bool manuellOnly)
{
    if (manuellOnly && (_filterForBasisContract.CalculationKind & ContractCalculationKind.Manual) != ContractCalculationKind.Manual) return true;

    if (IsAutoStart) return false;

    if (_filterForBasisContract.ContractKinds.Count > 0)
    {
        if (_filterForBasisContract.ContractKinds.Count < SelectedContractKinds.Count) return true;

        foreach (var id in SelectedContractKinds.Select(f=>f.HolderID).ToList())
            if (_filterForBasisContract.ContractKinds.IndexOf(id) == -1) return true;
    }

    // ... extensive filter comparison logic ...

    return false;
}
```

13.4 Lock/Unlock Billing

Property: `IsBilligAllowed` (bool)
Logic: Disallowed when `CounterHasChanged` is true
Purpose: Prevent billing with unsaved counter changes
Code Reference: AutomatedBillingViewModel.cs:230-238

```
public bool CounterHasChanged
{
    get => this._counterHasChanged;
    set
    {
        this._counterHasChanged = value;
        this.IsBilligAllowed = !value;  // Billing disabled when counters changed
    }
}
```

13.5 Wizard Page Property Changed Handler

Purpose: React to wizard page changes
Method: `WizardOnPropertyChanged(object sender, PropertyChangedEventArgs propertyChangedEventArgs)`

Behavior: Hides/shows navigation buttons based on current page
Code Reference: AutomatedBillingViewModel.cs:479-506

```
private void WizardOnPropertyChanged(object sender, PropertyChangedEventArgs propertyChangedEventArgs)
{
    if (propertyChangedEventArgs.PropertyName != nameof(Wizard.CurrentPage))
    {
        return;
    }

    if (!(Wizard.CurrentPage is BillingDateWizardPageViewModel))
        foreach (var page in Wizard.Pages)
        {
            if (page is BillingDateWizardPageViewModel && ((BillingDateWizardPageViewModel)page).NeedRefresh)
            {
                break;
            }
        }

    if (Wizard.CurrentPage is OverviewWizardPageViewModel || Wizard.CurrentPage is ContractPreviewViewModel)
    {
        IsWizardNavigationButtonGroupVisible = false;
    }
    else
    {
        IsWizardNavigationButtonGroupVisible = true;
    }
}
```

14. Start Module with Specific Contracts

14.1 Open Module with Pre-Selected Contracts

Purpose: Start billing for specific contracts directly
Method: `StartWithContracts(IList<int> contractI3Ds)`
Use Case: Called from other modules (e.g., contract management) to bill selected contracts
Implementation: Sets filter to specific contract IDs and navigates to selection page
Code Reference: AutomatedBillingViewModel.cs:924-934

```
public void StartWithContracts(IList<int> contractI3Ds)
{
    var wizPage = (ContractSelectionWizardPageViewModel)this.Wizard.Pages.First(f => f is ContractSelectionWizardPageViewModel);

    wizPage.SharedState.ChangeFilter((filter) =>
    {
        filter.ContractI3Ds = contractI3Ds.ToList();
    });
    _selectionPageActiv = true;
    //this.Wizard.CurrentPage = wizPage;
}
```

Module Controller Integration: AutomatedBillingAppModuleController.cs:27-35

```
if (param?.Count() > 1)
{
    object[] originalParameters = (object[])param[1];
    var parameters = originalParameters.FirstOrDefault() as AutomatedBillingAppModuleControllerParameters;
    if (parameters != null)
    {
        viewModel.StartWithContracts(parameters.ContractI3Ds);
    }
}
```

Data Entities

Key DTOs Used:

Contract Data

- **ReceiptContractHeadDTO**: Contract header data with billing info
- **ReceiptContractPosDTO**: Contract line items
- **ContractHolderList**: Contract holder information (customer/branch/type)
- **FoundContractViewModel**: UI wrapper for contract display
- **ContractToInvoiceParam**: Billing parameters per contract

Counter Data

- **AutomaticFacturaCounterToContractDTO**: Click counter current state
- **AutomaticFacturaCounterHistoryDTO**: Historical counter readings
- **AutomaticFacturaCounterImportDTO**: Counter import data
- **DeviceClickCounterDTO**: Click counter device master data
- **CounterKindDTO**: Counter type definition

Billing Results

- **InvoiceToContractResult**: Billing operation result
- **ContractBillingResultDTO**: Historical billing result
- **BillingResultWizardpageViewModel**: UI billing result page

Special Articles

- **SpecialArticleToContractHeadDTO**: Dynamic article configuration
- **SpecialArticleToContractHeadImport**: Import data for special articles
- **SpecialArticleToContractHeadImportResultDTO**: Import operation result

Email & Delivery

- **ReceiptMailTemplateDTO**: Email template data
- **DocumentDTO**: Attachment document
- **AssetSendType**: Delivery method enum

Filters

- **SearchBillingContractsFilter**: Contract search criteria
- **SearchCounterStateFilter**: Counter search criteria
- **SearchSpecialArticleToContractHeadFilter**: Special article search

Reports

- **ReportToSendTypeToGroup**: Report template mapping
- **ReportGroupParameterDTO**: Report parameters

General

- **IntStringList**: Generic ID/value pair
- **ContractPositions**: Position data
- **ContractPartibleArticlePositionen**: Partible article data
- **EmptyCountContractPosition**: Validation data

User Rights

Module Access: `UserRightsConst.Sales.AUTOMATED_BILLING`

Additional Rights:

- **Contract Types Management**: `UserRightsConst.Masterdata.Contracts.CONTRACT_TYPES`
- **Counter Types Management**: `UserRightsConst.Masterdata.Contracts.COUNTER_TYPES`
- **Termination Types Management**: `UserRightsConst.Masterdata.Contracts.TERMINATION_TYPES`

License Requirements

Primary License: `LicenseGuids.ContractBilling` **OR** `LicenseGuids.Centron`

Connection Types

Supported:

- `CentronConnectionType.SqlServer` (Direct database)
- `CentronConnectionType.CentronWebServices` (REST API)

Summary Statistics

- **Total Use Cases**: 60+
- **Functional Areas**: 14
- **Wizard Pages**: 9
- **Interface Methods**: 25+
- **DTOs**: 25+
- **Commands**: 4
- **Properties**: 40+

Module State Management

Loading State

- **IsLoaded** (bool): Module initialization complete
- **IsProcessing** (bool): Background operation in progress
- **IsStarted** (bool): Billing execution started

Filter State

- **FilterHasChanged** (bool): Current filter differs from stored

- **NeedCustomerRefresh** (bool): Customer list needs reload
- **CounterHasChanged** (bool): Counter values modified

UI State

- **IsWizardNavigationButtonGroupVisible** (bool): Show/hide wizard navigation
- **IsContractKindsSelected** (bool): Contract types partially selected
- **IsCustomersSelected** (bool): Customers partially selected
- **IsBranchesSelected** (bool): Branches partially selected

Integration Points

Called From

- Contract Management Module
- Customer Management
- MSP Collector Module

Calls To

- **ICustomerLogic**: Customer data
- **IAutomatedBillingLogic**: All billing operations
- **CentronRestService**: Web service calls
- **ReportEngine**: Invoice generation
- **FileManagement**: Document handling

External Systems

- **Rivebird**: Counter value import
- **MSP Systems**: Service monitoring data
- **Email Service**: Invoice delivery
- **Print Service**: Document printing

1.2 Pauschalabrechnung (Flat Rate Billing)

Module Path: `c:\DEV\C-entron.net\c-entron.NET\src\centron\Centron.WPF.UI\Modules\Finances\FlatrateBilling\`
Controller: FlatRateProjectAppModuleController.cs:15
ViewModel: FlatRateProjectViewModel.cs:35
Category: Abrechnung (Billing)
Module ID: {A0850A14-93A7-441E-B202-64B80C2488E5}
Description: Verwaltung und Erstellung von Pauschalabrechnungen (Management and creation of flat rate billings)
Rights: No specific rights defined
Connection Types: SqlServer only

Module Architecture

The Pauschalabrechnung module implements a sophisticated system for managing flat rate billing on orders with helpdesk timer assignment, balance tracking, and financial analysis. Key workflows include order loading, timer assignment to flat rate positions, balance item management, and comprehensive financial tracking.

Key Components:

- **Part List Structure**: Flat rate positions expand to show individual items and auto-calculated balance
- **Balance Calculation**: `FlatRateAmount - SumOfPartListItems = RemainingBalance`
- **Order Locking**: Prevents concurrent modifications via `AssetLockBL<OrderLock>`

- **Version Control**: Creates editable versions via `OrderBL.CreateNewVersion()`

Business Logic: `OrderBalanceBL` handles timer assignments, balance calculations, and extra item management. Uses `AssetArticleBL` for part list operations.

1.3 Provisionsauswertung (Commission Evaluation)

Module Path: `c:\DEV\C-entron.net\c-entron.NET\src\centron\Centron.WPF.UI\Modules\Finances\Receipts\Provision\Evaluation\`
Controller: ProvisionEvaluationAppModuleController.cs:12
ViewModel: ProvisionEvaluationViewModel.cs:28
Category: Abrechnung (Billing)
Module ID: C3EEF97C-C1CD-4979-8694-6D4B76D45A67
Description: Auswertung der Schema-basierten Provisionierung (Evaluation of schema-based commission)
Rights: UserRightsConst.Sales.Provision.PROVISION_EVALUATION_MODULE (20800088)
Connection Types: SqlServer, CentronWebServices

Module Architecture

Provides comprehensive analysis and reporting of schema-based commission calculations for employees. Three-panel layout: Employee List (left) with totals, Goals Panel (top-right) showing monthly targets vs. actuals, and Receipts Panel (bottom-right) with detailed commission breakdowns.

Data Flow: User Filters → `IReceiptLogic.GetReceiptProvisionEvaluation()` → DTO grouping by EmployeeI3D → Goal loading → Display with computed totals

Key Features: Monthly goal tracking with cumulative summaries, Excel export with configurable columns, receipts without commission warning, integration with schema management and customer assignments.

1.4 Provisionsschemas verwalten (Provision Schema Management)

Module Path: `src/centron/Centron.WPF.UI/Modules/Finances/Receipts/Provision/Schemas/`
Controller: ProvisionSchemaManagementAppModuleController.cs:15
ViewModel: ProvisionSchemaManagementViewModel.cs:24
Category: Abrechnung (Billing)
Module ID: 3C78D0DC-D7A8-44E2-976A-A010C5D494B9
Description: Verwaltung von Provisionsschemas (Management of commission schemas)
Rights: UserRightsConst.Sales.Provision.PROVISION_SCHEMA_MANAGEMENT
Connection Types: SqlServer, CentronWebServices

Module Architecture

Sophisticated system for defining commission schemas with time-based expiration and automatic succession via NextSchema chains. Implements cycle detection (Floyd's algorithm) and topological sorting (Kahn's algorithm) for dependency-safe persistence.

Core Features:

- **NextSchema Chain**: Automatic succession when current schema expires
- **Cycle Detection**: Prevents infinite loops before save (Floyd's algorithm)
- **Topological Sorting**: Saves schemas in dependency order to satisfy FK constraints
- **Employee-Based Rules**: 11 receiver types (FixedEmployee, CustomerAdviser1-6, ReceiptEditor, ReceiptAdviser1-2, ServiceArticleEmployee)
- **Provision Calculation**: SharePercentage, ProvisionPercentage, Source (All/ServiceOnly/ProductsOnly/MaterialGroups/OwnServiceArticlesOnly), Value (Auto/Earnings/Sales)

Business Logic: `ReceiptProvisionSchemaBL` handles priority evaluation, bulk application to receipts (batches of 200), and contract provision inheritance.

1.5 Provisionsschema Kundenzuordnung (Commission Schema Assignment)

Module Path: `c:\DEV\C-entron.net\c-entron.NET\src\centron\Centron.WPF.UI\Modules\Finances\Receipts\Provision\SchemaCustomerAssignments\`
Controller: AssignmentsAppModuleController.cs:15
ViewModel: AssignmentsManagementViewModel.cs:23
Category: Abrechnung (Billing)
Module ID: E839B44E-865F-460E-88CD-2460040483DA
Description: Verwaltung welche Provisionsschemas welchen Kunden zugeordnet sind (Management of commission schema assignments)
Rights: UserRightsConst.Sales.Provision.PROVISION_SCHEMA_CUSTOMER_ASSIGNMENT
Connection Types: SqlServer, CentronWebServices

Module Architecture

Manages commission schema assignment using **3-tier priority system**:

Priority Hierarchy (evaluated in order):

1. **Branch + Customer Assignment** (Highest): `ReceiptProvisionSchemaCustomerAssignment` table
2. **Customer Assignment**: `AccountCustomer.ProvisionSchemaI3D` column
3. **Global Schema** (Fallback): `ApplicationSettings.GlobalReceiptProvisionSchemaI3D`

Data Entities:

- `ReceiptProvisionSchemaCustomerAssignment` - Branch-specific (CustomerI3D, BranchI3D, SchemaI3D)
- `AccountCustomer.ProvisionSchemaI3D` - Customer-level assignment
- `ApplicationSettings.GlobalReceiptProvisionSchemaI3D` - System-wide default

Key Features: Dynamic branch column generation, multi-row edit support, bulk "Apply to Receipts" operation (batches of 200), session-level caching per customer+branch combination.

Schema Resolution (runtime): `GetCurrentProvisionSchemaForCustomer()` evaluates priority, caches result with key `"ProvisionSchemaForCustomer_{customerI3D}_{branchI3D}"`.

1.6 Vereinfachte Ticketabrechnung (Simplified Ticket Billing)

Module Path: `src/centron/Centron.WPF.UI/Modules/Finances/TimerBilling/`
Controller: TimerBillingAppModuleController.cs:14
ViewModel: TimerBillingViewModel.cs:23
Category: Abrechnung (Billing)
Module ID: {5B7AB256-86E4-4B33-B739-C1FB79399C68}
Description: Abrechnung von Tickets und einzelnen Zeiten (Billing of tickets and individual timers)
Rights: None
Connection Types: SqlServer, CentronWebServices

Module Architecture

Implements **7-step wizard workflow** for simplified ticket billing:

1. **Timer Selection** - 15+ filter criteria with inline editing
2. **AI Text Rating (Beta)** - 30-second timeout per timer, quality assessment with threshold-based selection
3. **Settings** - Text insertion, grouping, sorting, ticket closure options
4. **Order Item Mapping** - Map timers from orders to specific positions
5. **Summary** - Customer/ticket/timer counts, send type distribution
6. **Invoice Creation** - Multi-channel delivery (print, mail, PDF) with preview
7. **Results** - Display results with quick receipt access

Key Features:

- **AI Integration**: `AITextRating.AITextRatingTask()` with 30-second timeout, improved text generation

- **Timer Splitting**: Auto-split timers spanning multiple hourly surcharge rate periods with proportional break distribution
- **Inline Editing**: Full edit mode (all properties) vs. text-only mode with change tracking
- **Order Item Mapping**: Group by order, map to specific positions, price source control
- **Multi-Channel Delivery**: Customer-specific report assignments, duplicate report chain support, preview options
- **Batch Processing**: Progress tracking, memory management with `GC.Collect()`, cancellation handling (current vs. entire batch)

Wizard Navigation: Uses `WizardHelperViewModel<TimerBillingViewModel>` with conditional page enabling based on data availability.

2. Administration

2.1 Aufschläge Stundensätze (Hourly Rate Surcharges)

Module Path: `src/centron/Centron.WPF.UI/Modules/Administration/HourlySurchargeRates`
Controller: `HourlySurchargeRatesAppModuleController`
ViewModel: `HourlySurchargeRatesViewModel`
Category: Administration
Description: Verwalten von Aufschlagssätzen zu Mitarbeiterstunden für Abrechnungen und Projektkalkulationen

Use Cases

2.1.1 Stundenzuschläge für Mitarbeitertypen definieren

Zweck: Definition von prozentualen oder festen Zuschlägen auf Stundensätze für verschiedene Mitarbeitertypen und Situationen

Ablauf:

1. Benutzer öffnet Stundenzuschlag-Verwaltung
2. Erstellt neuen Zuschlag (z.B. "Überstunden", "Wochenende", "Nacht")
3. Definiert Zuschlagsart (prozentual % oder fix in €)
4. Setzt gültigen Zeitraum (von/bis Datum)
5. Optional: Verknüpft mit bestimmten Mitarbeitern oder Mitarbeitertypen
6. Speichert Konfiguration
7. System wendet Zuschlag bei Stundenzettel-Abrechnung an

Betroffene Felder: SurchargeType, SurchargeRate, SurchargeAmount, ValidFrom, ValidTo, EmployeeType

Auswirkungen:

- Automatische und korrekte Berechnung von Zuschlägen
- Konsistente Anwendung über alle Stundenzetteleingaben
- Flexible Konfiguration für verschiedene Zuschlag-Szenarien
- Verhindert Fehler bei manueller Zuschlagberechnung

2.1.2 Zuschlagsregeln pro Vertrag oder Projekt

Zweck: Spezifische Zuschlagsregeln auf Vertrags- oder Projektebene definieren (abweichend von Standard)

Ablauf:

1. Benutzer öffnet Vertrags- oder Projekt-Details
2. Navigiert zu Zuschlag-Einstellungen
3. Überschreibt globale Zuschlagssätze mit projektspezifischen Werten
4. Setzt Gültigkeitszeitraum für diese Regeln
5. Optional: Definiert Max-Zuschlag-Grenze für Kostencontrolling
6. Speichert Vertrags-/Projekt-spezifische Konfiguration
7. System nutzt diese Regeln bei Abrechnung für das Projekt

Betroffene Felder: ContractI3D, ProjectI3D, OverrideSurcharge, MaxSurchargeLimit, SurchargeRuleValidFrom

Auswirkungen:

- Flexible Abrechnung je Kunde/Projekt
- Ermöglicht unterschiedliche Gebührenstrukturen
- Bessere Preiskalkulationen für spezielle Aufträge
- Verhindert unvorhergesehene Kostenüberschreitungen

2.1.3 Urlaubszuschläge und Krankheitszuschläge konfigurieren

Zweck: Definition von Zuschlägen für Urlaubs- und Krankheitstage bei Stundenabrechnung

Ablauf:

1. Benutzer definiert Urlaubszuschlag (z.B. "Urlaubsaufschlag +50%")
2. Erfasst Krankheitszuschlag (z.B. "Krankenheit 0% - kostenlos für Unternehmen")
3. Stellt Anwendungslogik ein (ab wieviel Tagen Zuschlag?)
4. Optional: Differenziert nach Mitarbeiter-Klassifikation
5. Speichert Regeln
6. System wendet automatisch an, wenn Urlaub/Krankheit in Stundenzettel eingetragen

Betroffene Felder: AbsenceType, SurchargePercentage, ApplicableFrom, EmployeeClass

Auswirkungen:

- Automatische und faire Abrechnung von Absenzzeiten
- Keine manuellen Fehler bei Urlaubs-/Krankheits-Berechnung
- Transparente Kostenerfassung für Projekte
- Compliance mit Tarifverträgen

2.1.4 Zeitbasierte Zuschlagsmodelle (Spät-, Nacht-, Wochenend-Zuschläge)

Zweck: Definition und Verwaltung von Zuschlägen basierend auf Tageszeit oder Wochentag

Ablauf:

1. Benutzer erstellt Spät-Zuschlag (z.B. nach 18 Uhr +20%)
2. Erstellt Nacht-Zuschlag (z.B. 22-6 Uhr +50%)
3. Erstellt Wochenend-Zuschlag (Samstag +30%, Sonntag +50%)
4. Definiert Übergangsbereiche (z.B. ab 30 Min. Spät-Tätigkeit Zuschlag)
5. Optional: Stellt maximale Arbeitszeit pro Zuschlag-Kategorie
6. Speichert Zeitmodelle
7. System prüft Stundenzettel-Einträge gegen diese Zeiten und wendet Zuschläge an

Betroffene Felder: TimeSlotType, StartTime, EndTime, DayOfWeek, SurchargePercentage, MinutesThreshold

Auswirkungen:

- Korrekte Abrechnung von Schichtarbeit
- Automatische Einhaltung von Tarifverträgen
- Dokumentation für Arbeitszeit-Audits
- Fair für Mitarbeiter, transparent für Kunden

2.1.5 Zuschlag-Verlauf und Änderungsverfolgung

Zweck: Dokumentation aller Zuschlag-Änderungen für Compliance und Audit

Ablauf:

1. Benutzer öffnet Zuschlag-Details
2. Sieht Änderungsverlauf (wer, wann, was wurde geändert)
3. Kann alte Version einsehen und ggfs. wiederherstellen
4. Export von Zuschlag-Konditionen für Dokumentation
5. Benutzer generiert Report über alle aktuellen Zuschl äge
6. Speichert Report für Vertrags-Archiv
7. System archiviert automatisch alte Zuschlag-Versionen

Betroffene Felder: ChangeDate, ChangedByID, OldValue, NewValue, ChangeReason, ArchivedVersion

Auswirkungen:

- Vollständige Nachverfolgbarkeit von Änderungen
- Audit-Sicherheit bei Streitigkeiten
- Einfache Fehlerkorrektur möglich
- Compliance mit Aufbewahrungsrichtlinien

2.1.6 Zuschlag-Simulation und Vorschau

Zweck: Vorschau der Auswirkungen von Zuschlag-Änderungen auf Stundenzettel und Abrechnungen

Ablauf:

1. Benutzer plant Zuschlag-Änderung (z.B. Überstunden-Zuschlag erhöhen)
2. Aktiviert Simulations-Modus
3. System berechnet Impact auf aktive Stundenzettels (z.B. "Mehrkosten: +€5.000 monatlich")
4. Zeigt betroffene Mitarbeiter und Projekte
5. Benutzer kann Änderung mit den Auswirkungen vergleichen
6. Bestätigt oder verwirft Änderung
7. Bei Bestätigung: System wendet ab sofort an

Betroffene Felder: SimulationMode, AffectedRecords, ImpactAnalysis, CostDifference, ApprovedChange

Auswirkungen:

- Informierte Entscheidungen bei Zuschlag-Änderungen
- Verhindert unerwartete Kostensprünge
- Bessere Finanzplanung
- Transparenz für Geschäftsleitung

2.2 c-entron DSGVO (GDPR Compliance)

Module Path: `src/centron/Centron.WPF.UI/Modules/Administration/DSGVO`
Controller: `CentronDataSecurityAppModuleController`
ViewModel: `CentronDataSecurityViewModel`
Category: Administration
Description: Verwaltung von DSGVO-Compliance-Einstellungen, Datenschutzvorgaben und Datenverarbeitungsverträgen

Use Cases

2.2.1 Datenverarbeitungsvorgaben und -verträge verwalten

Zweck: Dokumentation und Verwaltung von Auftragsverarbeitungsverträgen (AVVs) und Datenschutzvorgaben

Ablauf:

1. Benutzer öffnet DSGVO-Modul

2. Navigiert zu Datenverarbeitungsvorgaben
3. Erstellt neuen AV-Vertrag mit Kategorien (z.B. "Kundendaten-Verarbeitung")
4. Dokumentiert Datentypen, Verarbeitungszweck, Speicherdauer
5. Lädt AV-Vertrag-Dokument hoch oder verlinkt
6. Definiert Sicherheitsmaßnahmen und Kontrollpflichten
7. Speichert Vertrag mit Gültigkeitsdatum und Unterschriftsfeldern
8. System generiert Compliance-Report für Audits

Betroffene Felder: ContractType, DataCategories, ProcessingPurpose, RetentionPeriod, SecurityMeasures, ContractDocument

Auswirkungen:

- Rechtssicherheit durch dokumentierte Datenverarbeitung
- Compliance mit DSGVO Art. 28
- Zentrale Verwaltung aller Verträge
- Audit-Trail für Datenschutzbeauftragte

2.2.2 Datenlösch-Regeln und -Fristen konfigurieren

Zweck: Automatisierte Datenlöschung nach Fristen gemäß DSGVO und Aufbewahrungsrichtlinien

Ablauf:

1. Benutzer konfiguriert automatische Löschregeln
2. Definiert Datentyp (z.B. "Inaktive Kundendaten")
3. Setzt Aufbewahrungsfrist (z.B. "3 Jahre nach letzter Aktivität")
4. Wählt Löschmethode (weich-löschen, physisches Löschen, Anonymisierung)
5. Optional: Stellt Genehmigung vor Löschung ein
6. Aktiviert Regel
7. System führt automatisch Löschung durch und protokolliert

Betroffene Felder: DataType, RetentionDays, DeletionMethod, ApprovalRequired, AutomationActive, DeletionLog

Auswirkungen:

- Automatische Compliance mit Aufbewahrungsfristen
- Sicherheit: Keine unnötigen Daten mehr gespeichert
- Audit-Protokolle dokumentieren Löschungen
- DSGVO-Artikel 17 (Recht auf Vergessenwerden) unterstützt

2.2.3 Datenschutz-Einwilligungen verwalten

Zweck: Verwaltung und Nachverfolgung von Einwilligungen zur Datenverarbeitung

Ablauf:

1. Benutzer erstellt Einwilligungs-Template (z.B. "Marketing-Mails", "Newsletter", "Analytik")
2. Definiert Zweck, Umfang und Gültigkeitsdauer
3. Erstellt oder importiert Datenschutzerklärung
4. Verknüpft mit Kundenkonten
5. System erfasst Einwilligungen bei Kundenregistrierung/Vertragsabschluss
6. Dokumentiert Datum, IP-Adresse und Kanal (Online, Unterschrift, etc.)
7. Bietet Kunden Möglichkeit, Einwilligung zu widerrufen
8. Generiert Nachweise für Audits

Betroffene Felder: ConsentType, ConsentPurpose, ConsentDate, IpAddress, ConsentChannel, RevocationDate

Auswirkungen:

- Nachweisbarkeit der Einwilligungen
- Automatische Compliance mit DSGVO Art. 7
- Dokumentation für Behörden
- Schutz vor rechtlichen Ansprüchen

2.2.4 Datensubjekt-Anfragen (Auskunft, Berichtigung, Löschung)

Zweck: Verwaltung von Anfragen betroffener Personen zu ihren Daten

Ablauf:

1. Benutzer empfängt Auskunftsanfrage (per Formular, Mail, Post)
2. Erstellt neuen Datensubjekt-Anfrage-Ticket
3. Ordnet Anfragtyp zu (Auskunft, Berichtigung, Löschung, Datenübertragung)
4. System sammelt alle Daten der betroffenen Person
5. Benutzer generiert Auskunftsbericht (z.B. PDF mit allen Daten)
6. Sendet Antwort innerhalb der gesetzlichen Frist (30 Tage)
7. Archiviert Anfrage mit Antwort-Dokument
8. System verfolgt Frist-Einhaltung

Betroffene Felder: RequestType, SubjectIdentification, DataCollected, ResponseGenerated, ResponseDate, DeadlineDate

Auswirkungen:

- Compliance mit DSGVO Art. 12-15 (Auskunftspflicht)
- Automatische Fristüberwachung
- Zentrale Dokumentation aller Anfragen
- Transparenz für betroffene Personen

2.2.5 Datenschutz-Folgeabschätzung (DPIA)

Zweck: Dokumentation und Durchführung von Datenschutz-Folgeabschätzungen für riskante Verarbeitungen

Ablauf:

1. Benutzer identifiziert riskante Verarbeitung (z.B. "Automatische Entscheidungsfindung")
2. Startet DPIA-Prozess
3. Dokumentiert Verarbeitungszweck, beteiligte Datentypen, Risiken
4. Bewertet Risiko-Schweregrad (hoch/mittel/niedrig)
5. Definiert Gegenmaßnahmen zur Risiko-Minimierung
6. Konsultiert ggfs. Datenschutzbeauftragte
7. Archiviert DPIA-Dokument
8. System erinnert an Überprüfung nach Frist

Betroffene Felder: ProcessingDescription, DataTypes, IdentifiedRisks, RiskLevel, Countermeasures, DpiaApprovedDate

Auswirkungen:

- DSGVO Art. 35: Obligatorisch für risikoreiche Verarbeitungen
- Dokumentation für Aufsichtsbehörden
- Systematische Risiko-Analyse
- Defensiv bei Inspektionen

2.2.6 Datenschutzverletzungen (Sicherheitsvorfälle) melden

Zweck: Registrierung und Meldung von Datenschutzverletzungen (Breaches) an Behörden und betroffene Personen

Ablauf:

1. Sicherheitsvorfall wird erkannt (z.B. Datenabfluss, Ransomware, Hacking)
2. Benutzer erstellt Sicherheitsvorfalls-Ticket
3. Dokumentiert Art, Umfang, Betroffen-Zeitpunkt
4. Beurteilt Schweregrad (z.B. PII, Bankdaten, Gesundheitsdaten)
5. Leitet Meldung an Datenschutzbeauftragte ein
6. System prüft, ob Meldung an Behörde erforderlich ist
7. Bereitet Benachrichtigung an betroffene Personen vor
8. Archiviert Vorfalls-Dokumentation und Handlung

Betroffene Felder: BreachType, DataAffected, AffectedPersons, BreachDate, ReportingDate, RemediationSteps

Auswirkungen:

- Rechtliche Anforderung: Meldepflicht gemäß DSGVO Art. 33
- Transparenz mit betroffenen Personen
- Audit-Trail für Behörden
- Prävention durch Dokumentation von Vorfällen

2.3 Einstellungen (Settings)

Modulpfad: `src/centron/Centron.WPF.UI/Modules/Administration/Settings/`
Controller: `SettingsAppModuleController`
ViewModel: `SettingsContainerViewModel`
Schnittstelle: `ISettingsLogic`
Kategorie: Administration
Beschreibung: Zentrale Verwaltung aller Systemeinstellungen, Konfigurationen und Globalen Optionen
Lizenz: `LicenseGuids.Centron`
Rechte: `UserRightsConst.Administration.SETTINGS`

Modul-Architektur

Das Einstellungsmodul ist ein **Multi-Tab-Interface** zur Verwaltung verschiedener Einstellungsbereiche:

- **Allgemeine Einstellungen**: Systemverhalten, Sprache, Zahlenformate, Datumsformate
- **UI-Einstellungen**: Themes, Fensterposition, Startoption, Favoriten
- **Logging & Monitoring**: Log-Level, Log-Verzeichnis, Debug-Optionen
- **E-Mail & Kommunikation**: SMTP-Konfiguration, Mail-Server, Benachrichtigungen
- **Datenbank & Verbindungen**: Connection Strings, Backup-Pfade, Synchronisierung
- **Lizenzen & Aktivierung**: Lizenzschlüssel, Modulaktivierungen, Feature-Flags
- **Benutzerverwaltung**: Single Sign-On, Azure AD Integration, Passwortrichtlinien
- **Berichterstellung**: Report-Server, PDF-Export-Optionen

Vollständige Use Cases

2.3.1 Globale Systemparameter konfigurieren

Zweck: Grundlegende Systemeinstellungen wie Sprache, Zahlenformat, Datumsformat einstellen
Betroffene Felder:

- `CultureInfo` (de-DE, en-US, etc.)
- `NumberFormat` (Dezimaltrennzeichen)
- `DateFormat` (DD.MM.YYYY vs MM/DD/YYYY)
- `TimeZone` (Zeitzone für Server und Client)
  Auswirkung: Globale Formatierung in der ganzen Anwendung
  Speicherort: `ApplicationSettings` Tabelle in der Datenbank

2.3.2 UI-Theme und Darstellung anpassen

Zweck: Benutzer-spezifisches Aussehen und Verhalten der Oberfläche konfigurieren
Optionen:

- **Theme Selection** (Light, Dark, Office, Windows 11)
- **Font Size** (90%, 100%, 110%, 120%)
- **Ribbon Position** (Top, Bottom)
- **Startbildschirm** (Dashboard, Letzte Modul, Benutzerdefiniert)
- **Fenster-Status** (Maximiert, Fullscreen, Standard)
- **Favoriten-Module** (Drag & Drop Anordnung)
  Speicherort: Benutzer-spezifische Settings in `ApplicationSettings`

2.3.3 Logging und Debugging aktivieren

Zweck: Diagnose und Fehlerbehandlung für Entwicklung und Support
Einstellungen:

- **Log-Level** (Debug, Info, Warning, Error, Fatal)
- **Log-Ziel** (File, Console, EventLog)
- **Log-Verzeichnis** (z.B. `C:\Logs\centron\`)
- **Rotation-Policy** (Täglich, Nach Größe, Keine)
- **Max Log-Größe** (MB, GB)
- **Debug-Modus** (Konsolen-Fenster anzeigen)
- **Performance-Tracing** (Query-Zeiten, UI-Rendering messen)
  Code-Referenz: [CentronLogAppModuleController]

2.3.4 E-Mail und Benachrichtigungen konfigurieren

Zweck: SMTP-Server und Versandoptionen einstellen
Felder:

- **SMTP-Server**: Hostname (z.B. mail.example.com)
- **Port**: Standard 587 (TLS) oder 465 (SSL)
- **Authentifizierung**: Benutzername, Passwort
- **Absender-Adresse**: system@example.com
- **Standard-CC**: Optional Kopien an bestimmte Adressen
- **SSL/TLS**: Verschlüsselungsart
- **Test-E-Mail**: Button zum Versand einer Test-Mail
- **Benachrichtigungs-Filter**: Welche Events benachrichtigen per E-Mail

2.3.5 Datenbank-Verbindung konfigurieren

Zweck: Verbindungstrings und Datenbankeinstellungen verwalten
Konfigurationen:

- **Server**: SQL-Server Hostname/IP
- **Datenbank**: Datenbankname (z.B. "Centron")
- **Authentifizierung**: SQL-Authentifizierung oder Windows-Auth
- **Connection-String Encryption**: Verschlüsseln gespeicherter Verbindungen
- **Connection Pool Size**: (Min 5, Max 500)
- **Connection Timeout**: (Sekunden)
- **Backup-Pfad**: Automatische Backups
- **Backup-Frequenz**: Täglich, Wöchentlich, etc.

2.3.6 Lizenzen verwalten und Feature-Flags setzen

Zweck: Modullizenzen aktivieren/deaktivieren und Feature-Gating
Funktionen:

- **Lizenzschlüssel eingeben**: Neue Lizenz registrieren
- **Aktive Lizenzen anzeigen**: Welche Module sind lizenziert

- **Lizenz-Verfall warnen**: Tage vorher benachrichtigen
- **Feature-Flags**: Experimentelle Features aktivieren/deaktivieren
- **Trial-Modus**: Testfunktionen für 30 Tage aktivieren
- **Lizenztyp-Information**: Perpetual vs Subscription
- **Benutzer-Limit**: Maximal parallel arbeitende Benutzer (wenn lizenziert)

2.3.7 Benutzerverwaltung und Authentifizierung

Zweck: Single Sign-On und Authentifizierungsmethoden konfigurieren
Optionen:

- **Azure Active Directory (AAD)**: Aktivieren/Deaktivieren
- **AAD Tenant ID**: Azure Tenant-ID eintragen
- **Automatische Benutzer-Erstellung**: Neue AAD-Benutzer automatisch in c-entron anlegen
- **Passwort-Komplexität**: Anforderungen definieren
- **Session-Timeout**: Inaktivität-Grenze (Minuten)
- **Zwei-Faktor-Authentifizierung**: Optional aktivierbar
- **LDAP/Active Directory**: Alternative Authentifizierung
- **API-Keys Verwaltung**: Für REST-API-Zugriffe

2.3.8 Berichterstellung und Export konfigurieren

Zweck: Report Server und PDF-Export Einstellungen
Konfigurationen:

- **Report-Server URL**: SSRS-Server Adresse
- **Report-Verzeichnis**: Netzwerk-Pfad für Report-Templates
- **PDF-Export Standard**: Papierformat, Auflösung, Kompression
- **Export-Verzeichnis**: Default-Speicherort für Exports
- **Automatische Report-Generierung**: Zeitgesteuert
- **E-Mail-Versand von Reports**: Nach Generierung automatisch mailen
- **Report-Archivierung**: Alte Reports automatisch löschen nach X Tagen
- **Watermark**: Vertraulichkeit-Kennzeichnung auf Exports

2.3.9 Sprach- und Lokalisierungseinstellungen

Zweck: Mehrsprachigkeit und regionale Anpassungen
Optionen:

- **Sprache**: Deutsch, English, Français (je nach Lizenz)
- **Zahlen-Lokalität**: 1.000,00 (German) vs 1,000.00 (English)
- **Währung**: EUR, USD, etc.
- **Land/Region**: Für Steuern, Rechtsvorschriften
- **Feiertags-Kalender**: Land-spezifische Feiertage

2.3.10 Sicherheits- und Compliance-Einstellungen

Zweck: Datenschutz, Auditing, DSGVO-Compliance
Felder:

- **Audit-Logging**: Alle Änderungen protokollieren
- **Encryption-at-Rest**: Datenbank-Verschlüsselung
- **Passwort-Historie**: Wie viele alte Passwörter blockieren
- **DSGVO-Daten-Löschung**: Automatische Anonymisierung nach X Jahren
- **Datenzugriff-Logging**: Wer hat welche sensiblen Daten zugegriffen
- **IP-Whitelist**: Nur bestimmte IPs dürfen zugreifen
- **Compliance-Reports**: Automatische Berichte für Audits

2.4 Kontenrahmen (Chart of Accounts)

Module Path: `src/centron/Centron.WPF.UI/Modules/Administration/Settings` (Accounting Sub-Section)
Controller: Settings Module
ViewModel: Settings Module
Category: Administration
Description: Verwaltung von Buchhaltungskontenrahmen

Use Cases

2.4.1 Standard-Kontenrahmen importieren

Zweck: SKR03 oder SKR04 Kontenrahmen aus vordefinierter Vorlage importieren
Ablauf:

1. Administrator öffnet Einstellungen → Buchhaltung → Kontenrahmen
2. Wählt Kontenrahmen-Typ (SKR03, SKR04, IKR) aus Dropdown
3. System importiert alle Konten mit Nummern, Bezeichnungen und Steuerklassen
4. Bestätigung zeigt Anzahl importierter Konten
   Betroffene Felder: KontenrahmenTyp, KontoNummer, Bezeichnung, Steuerklasse, KontoArt
   Auswirkungen:

- Buchhaltungskonten stehen für Buchungen zur Verfügung
- DATEV-Export verwendet korrekte Kontenzuordnungen
- Mandant ist bereit für Finanzbuchhaltung

2.4.2 Benutzerdefinierte Konten hinzufügen

Zweck: Zusätzliche Konten für spezielle Geschäftsvorfälle anlegen
Ablauf:

1. Administrator klickt auf "Neues Konto" in Kontenrahmen-Übersicht
2. Gibt Kontonummer (muss im gültigen Bereich liegen), Bezeichnung und Kontoart ein
3. Ordnet Steuerschlüssel und übergeordnetes Konto zu
4. Speichert neues Konto
   Betroffene Felder: KontoNummer, Bezeichnung, KontoArt, SteuerschlüsselI3D, ÜbergeordnetesKontoI3D
   Auswirkungen:

- Erweitert Standard-Kontenrahmen um firmenspezifische Konten
- Konten stehen in Buchungsmasken zur Verfügung
- Werden in Buchhaltungsexporte einbezogen

2.4.3 Konten-Mapping für verschiedene Mandanten

Zweck: Unterschiedliche Kontenrahmen pro Mandant verwenden
Ablauf:

1. Administrator wechselt Mandanten-Kontext in Einstellungen
2. Öffnet Kontenrahmen-Konfiguration für ausgewählten Mandanten
3. Kann anderen Kontenrahmen-Typ wählen oder individuelle Anpassungen vornehmen
4. Speichert mandantenspezifische Konfiguration
   Betroffene Felder: MandantI3D, KontenrahmenTyp, IndividuelleKonten
   Auswirkungen:

- Multi-Mandanten-Fähigkeit mit unterschiedlichen Buchhaltungssystemen
- Jeder Mandant kann eigenen Kontenrahmen pflegen
- Zentrale Verwaltung bleibt gewährleistet

2.4.4 Konten-Status verwalten (Aktiv/Inaktiv)

Zweck: Nicht mehr benötigte Konten deaktivieren ohne Datenverlust
Ablauf:

1. Administrator öffnet Kontenrahmen-Liste
2. Markiert Konto und setzt Status auf "Inaktiv"
3. System prüft, ob offene Buchungen auf diesem Konto existieren
4. Bei Erfolg wird Konto in Auswahllisten ausgeblendet
   Betroffene Felder: KontoI3D, IsActive, IsDeleted, LetzteBuchungDatum
   Auswirkungen:

- Verhindert neue Buchungen auf veralteten Konten
- Historische Buchungen bleiben erhalten und auswertbar
- Reduziert Unübersichtlichkeit in Kontenauswahlen

2.5 Leasing/Service (Leasing/Service Management)

Module Path: `src/centron/Centron.WPF.UI/Modules/Administration/ServiceAndLeasing`
Controller: `ServiceLeasingAppModuleController`
ViewModel: `ServiceLeasingViewModel`
Category: Administration
Description: Verwaltung von Leasing und Service Sätzen

Use Cases

2.5.1 Leasing-Rate konfigurieren

Zweck: Standardisierte Leasing-Konditionen für Hardware und Software definieren
Ablauf:

1. Administrator öffnet Leasing/Service Modul
2. Erstellt neuen Leasing-Satz mit Laufzeit (z.B. 36 Monate), Ratenintervall (monatlich), Faktor
3. Ordnet Produktkategorien zu (Server, Arbeitsplätze, Drucker)
4. Definiert Restwertberechnung und Verlängerungsoptionen
   Betroffene Felder: LeasingSatzName, LaufzeitMonate, RatenIntervall, LeasingFaktor, Restwert, ProduktkategorieI3D
   Auswirkungen:

- Schnelle Angebotserstellung für Leasing-Hardware
- Standardisierte Kalkulation über alle Angebote hinweg
- Automatische Vertragserstellung bei Auftragsumwandlung

2.5.2 Wartungsvertrag-Vorlage anlegen

Zweck: Service-Level-Agreements (SLA) als wiederverwendbare Vorlagen erstellen
Ablauf:

1. Administrator klickt auf "Neuer Service-Satz"
2. Definiert Service-Parameter: Reaktionszeit, Entstörungszeit, Verfügbarkeit
3. Legt monatlichen Pauschalpreis oder Stundensatz fest
4. Verknüpft mit Ticketprozess-Vorlagen für Priorisierung
   Betroffene Felder: ServiceSatzName, ReaktionszeitStunden, EntstörungszeitStunden, Verfügbarkeit, MonatsPreis, StundensatzI3D
   Auswirkungen:

- SLA-konforme Ticketbearbeitung
- Vertragliche Reaktionszeiten werden überwacht
- Automatische Eskalation bei Fristüberschreitungen

2.5.3 Gemischte Verträge (Leasing + Service)

Zweck: Kombinierte Hardware-Leasing und Wartungsverträge als Paket anbieten
Ablauf:

1. Administrator erstellt kombinierten Satz

2. Wählt Leasing-Komponente (z.B. Server 36 Monate) aus
3. Fügt Service-Komponente hinzu (z.B. 24/7-Support)
4. System berechnet Gesamtrate automatisch
   Betroffene Felder: LeasingSatzI3D, ServiceSatzI3D, KombinationsRabatt, GesamtMonatsrate
   Auswirkungen:

- Attraktive Paketangebote für Kunden
- Vereinfachte Vertragsverwaltung
- Umsatzplanung durch wiederkehrende Einnahmen

2.5.4 Preisaktualisierung für bestehende Sätze

Zweck: Inflationsanpassung oder Marktpreisänderungen in Sätzen nachpflegen
Ablauf:

1. Administrator wählt Leasing- oder Service-Satz aus
2. Ändert Preise, Faktoren oder Konditionen
3. System zeigt betroffene aktive Verträge an
4. Administrator entscheidet über Übernahme in laufende Verträge (optional)
   Betroffene Felder: LeasingSatzI3D, ServiceSatzI3D, GültigAb, AlterPreis, NeuerPreis, VertragsAnpassung
   Auswirkungen:

- Aktuelle Marktpreise in neuen Angeboten
- Optional: Vertragsverlängerungen mit neuen Konditionen
- Historische Preise bleiben für alte Verträge erhalten

2.6 Mailvorlagen (Email Templates)

Module Path: `src/centron/Centron.WPF.UI/Modules/Administration/MailTemplates`
Controller: `MailTemplatesAppModuleController`
ViewModel: `MailTemplatesViewModel`
Category: Administration
Description: Alle Mailvorlagen

Use Cases

2.6.1 Standard-Mailvorlage erstellen

Zweck: Wiederverwendbare E-Mail-Vorlagen für häufige Geschäftsvorgänge anlegen
Ablauf:

1. Administrator öffnet Mailvorlagen-Modul und klickt auf "Neue Vorlage"
2. Gibt Vorlagenname, Betreff und HTML-formatierten Text ein
3. Fügt Platzhalter ein (z.B. {KundenName}, {RechnungsNummer}, {Betrag})
4. Definiert Verwendungszweck (Rechnung, Angebot, Mahnung, Ticketbenachrichtigung)
   Betroffene Felder: VorlagenName, Betreff, MailText, Verwendungszweck, Platzhalter, IsHTML
   Auswirkungen:

- Konsistente Kundenkommunikation
- Zeitersparnis durch vorgefertigte Texte
- Corporate Identity wird gewahrt

2.6.2 Mailvorlage mit Anhang-Logik

Zweck: Automatisches Anfügen von Dokumenten beim Versand konfigurieren
Ablauf:

1. Administrator öffnet bestehende Vorlage (z.B. "Rechnung versenden")
2. Aktiviert Anhang-Option und wählt Dokumenttyp (PDF-Rechnung, AGB, Datenschutz)

3. Definiert Bedingungen für Anhänge (z.B. nur bei Erstrechnung AGB anfügen)
4. Testet Vorlage mit Beispieldaten
   Betroffene Felder: VorlagenI3D, AnhangAktiv, DokumentTypen, AnhangBedingungen
   Auswirkungen:

- Automatischer Versand von Rechnungs-PDFs
- Compliance durch automatische AGB-Zustellung
- Reduzierte Fehlerquote bei manuellen Anhängen

2.6.3 Mehrsprachige Mailvorlagen

Zweck: E-Mail-Vorlagen in Deutsch und Englisch für internationale Kunden bereitstellen
Ablauf:

1. Administrator erstellt deutsche Vorlage als Basis
2. Klickt auf "Übersetzung hinzufügen" und wählt Englisch
3. Gibt englische Betreffzeile und Text ein mit denselben Platzhaltern
4. System wählt automatisch passende Sprache basierend auf Kundenstammdaten
   Betroffene Felder: VorlagenI3D, SpracheCode, BetreffDE, BetreffEN, MailTextDE, MailTextEN
   Auswirkungen:

- Professionelle Ansprache internationaler Kunden
- Automatische Sprachauswahl basierend auf Kundensprache
- Einheitliches Erscheinungsbild in allen Sprachen

2.6.4 Ereignisbasierte Mailvorlagen

Zweck: Automatische E-Mail-Benachrichtigungen bei Systemereignissen konfigurieren
Ablauf:

1. Administrator öffnet Mailvorlagen und wählt "Ereignis-Vorlage"
2. Wählt Auslöser (Ticket erstellt, Rechnung fällig, Vertrag läuft aus)
3. Definiert Empfängerlogik (Kunde, zuständiger Mitarbeiter, Abteilung)
4. Aktiviert Vorlage und setzt Versandzeitpunkt (sofort, täglich, wöchentlich)
   Betroffene Felder: EreignisTyp, EmpfängerLogik, VersandZeitpunkt, IsAktiv, BedingungsFilter
   Auswirkungen:

- Proaktive Kundenkommunikation ohne manuellen Aufwand
- Rechtzeitige Erinnerungen an Fälligkeiten
- Verbesserte Service-Qualität durch zeitnahe Benachrichtigungen

2.7 Mandanten (Tenants/Multi-Tenancy)

Module Path: `src/centron/Centron.WPF.UI/Modules/Administration/MandatorManagement`
Controller: `MandatorManagementAppModuleController`
ViewModel: `MandatorManagementViewModel`
Category: Administration
Description: Mandanten Verwaltung

Use Cases

2.7.1 Neuen Mandanten anlegen

Zweck: Separate Geschäftseinheit oder Tochtergesellschaft im System abbilden
Ablauf:

1. Administrator öffnet Mandanten-Verwaltung und klickt auf "Neuer Mandant"
2. Gibt Firmenname, Anschrift, Steuernummer, Handelsregisternummer ein
3. Wählt Kontenrahmen (SKR03/SKR04), Währung und Sprache

4. System erstellt isolierte Datenbereiche für Kunden, Artikel, Rechnungen
   Betroffene Felder: MandantName, Firmenanschrift, Steuernummer, HRNummer, KontenrahmenTyp, WährungCode, SpracheCode
   Auswirkungen:

- Vollständige Datentrennung zwischen Mandanten
- Separate Buchhaltung und Nummernkreise
- Multi-Mandanten-Fähigkeit für Unternehmensgruppen

2.7.2 Mandanten-Wechsel für Benutzer

Zweck: Mitarbeiter können zwischen verschiedenen Mandanten umschalten
Ablauf:

1. Benutzer klickt auf Mandanten-Auswahl in Hauptmenü
2. System zeigt Liste der zugewiesenen Mandanten
3. Benutzer wählt gewünschten Mandanten aus
4. Anwendung lädt Daten und Einstellungen des neuen Mandanten
   Betroffene Felder: BenutzerI3D, MandantI3D, MandantRechte, LetzterMandant
   Auswirkungen:

- Flexible Arbeit in mehreren Geschäftseinheiten
- Kontextabhängige Datendarstellung
- Verhindert versehentliche Datenvermischung

2.7.3 Mandanten-spezifische Einstellungen

Zweck: Individuelle Konfiguration pro Mandant (Logo, Geschäftszeiten, Zahlungsbedingungen)
Ablauf:

1. Administrator wählt Mandanten aus und öffnet Einstellungen
2. Lädt Firmenlogo hoch, definiert Standard-Zahlungsziele
3. Konfiguriert E-Mail-Server, Briefpapier-Layout und Dokumentvorlagen
4. Legt Geschäftszeiten und Feiertage fest
   Betroffene Felder: MandantI3D, LogoDatei, StandardZahlungsziel, EMailServer, Briefkopf, Geschäftszeiten
   Auswirkungen:

- Individuelles Erscheinungsbild pro Mandant
- Korrekte Fälligkeitsberechnung unter Berücksichtigung von Feiertagen
- Mandantenspezifische Dokumentengestaltung

2.7.4 Mandanten-übergreifende Auswertungen

Zweck: Konsolidierte Berichte über alle Mandanten hinweg für Geschäftsführung
Ablauf:

1. Administrator mit Spezialrecht öffnet Auswertungsmodul
2. Wählt "Mandantenübergreifende Auswertung"
3. System aggregiert Umsätze, Kosten, Gewinne aller Mandanten
4. Erstellt Vergleichsreport mit Drill-Down-Möglichkeit pro Mandant
   Betroffene Felder: MandantI3D, Umsatz, Kosten, Gewinn, Periode, Konsolidierungsregel
   Auswirkungen:

- Gesamtüberblick über Unternehmensgruppe
- Vergleichbarkeit der Geschäftseinheiten
- Fundierte Entscheidungsgrundlage für Management

2.8 Mitarbeiter (Employees)

Module Path: `src/centron/Centron.WPF.UI/Modules/Administration/EmployeeManagement`
Controller: `EmployeeManagementAppModuleController`

ViewModel: `EmployeeManagementViewModel`
Category: Administration
Description: Mitarbeiter Verwaltung

Use Cases

2.8.1 Mitarbeiter-Stammdaten anlegen

Zweck: Neuen Mitarbeiter mit allen relevanten Daten im System erfassen
Ablauf:

1. Personalabteilung öffnet Mitarbeiter-Modul und klickt auf "Neuer Mitarbeiter"
2. Erfasst Personalien (Name, Geburtsdatum, Anschrift, Kontaktdaten)
3. Hinterlegt Vertragsdetails (Eintritt, Position, Abteilung, Gehalt)
4. Erstellt optional zugehörigen Systembenutzer mit Rechten
   Betroffene Felder: Vorname, Nachname, Geburtsdatum, Anschrift, Telefon, EMail, Eintrittsdatum, PositionI3D, AbteilungI3D
   Auswirkungen:

- Mitarbeiter steht für Arbeitszeiterfassung zur Verfügung
- Kann Tickets zugewiesen bekommen
- Erscheint in Personallisten und Organigrammen

2.8.2 Arbeitszeiten und Abwesenheiten pflegen

Zweck: Urlaubstage, Krankheit und Arbeitszeitmodelle verwalten
Ablauf:

1. Vorgesetzter oder HR öffnet Mitarbeiterdetails
2. Navigiert zu Abwesenheiten-Tab und erfasst Urlaub mit Datum Von/Bis
3. System berechnet Urlaubsanspruch und prüft Verfügbarkeit
4. Genehmigung wird in Workflow zur Bestätigung geschickt
   Betroffene Felder: MitarbeiterI3D, AbwesenheitsTyp, DatumVon, DatumBis, Urlaubstage, Genehmigt, GenehmigtVonI3D
   Auswirkungen:

- Personalplanung berücksichtigt Abwesenheiten
- Ticket-Routing vermeidet abwesende Mitarbeiter
- Zeiterfassung zeigt Soll-Arbeitszeit korrekt

2.8.3 Stundensätze und Qualifikationen hinterlegen

Zweck: Verrechnungssätze für Projekte und Service-Einsätze definieren
Ablauf:

1. Administrator öffnet Mitarbeiter und wechselt zu "Stundensätze"
2. Legt Verkaufspreis (Kundensatz) und Kostensatz (intern) fest
3. Definiert Qualifikationen und Zertifizierungen (IT-Security, Netzwerk, Programmierung)
4. System verwendet Sätze automatisch bei Ticket-Abrechnung und Projektkalkulation
   Betroffene Felder: MitarbeiterI3D, StundensatzVerkauf, StundensatzKosten, Qualifikationen, Zertifizierungen
   Auswirkungen:

- Automatische Kostenberechnung bei Service-Tickets
- Projektkalkulationen nutzen korrekte Personalsätze
- Ressourcenplanung basierend auf Qualifikationen

2.8.4 Mitarbeiter-Austritte und Archivierung

Zweck: Ausgeschiedene Mitarbeiter korrekt aus dem System entfernen
Ablauf:

1. HR öffnet Mitarbeiter-Stammdaten und setzt Austrittsdatum
2. System deaktiviert zugehörigen Systembenutzer automatisch
3. Offene Tickets werden auf neuen Zuständigen umverteilt

4. Mitarbeiter wird in Reports als "Inaktiv" markiert, Daten bleiben erhalten
   Betroffene Felder: MitarbeiterI3D, Austrittsdatum, IsActive, SystemBenutzerI3D, TicketNeuzuordnung
   Auswirkungen:

- Keine neuen Zuweisungen an ausgeschiedene Mitarbeiter
- Historische Daten bleiben für Auswertungen verfügbar
- DSGVO-konforme Archivierung ohne Datenverlust

2.9 Rechteverwaltung (Rights Management)

Module Path: `src/centron/Centron.WPF.UI/Modules/Administration/RightsManagement`
Controller: `RightsManagamentAppModuleController`
ViewModel: `RightsManagmentViewModel`
Category: Administration
Description: Verwaltung von Rechten und Rechtegruppen

Use Cases

2.9.1 Rechtegruppe erstellen

Zweck: Vordefinierte Rechte-Sets für typische Benutzerrollen anlegen
Ablauf:

1. Administrator öffnet Rechteverwaltung und klickt auf "Neue Rechtegruppe"
2. Gibt Gruppenname ein (z.B. "Service-Techniker", "Buchhaltung", "Vertrieb")
3. Wählt aus Rechte-Baum alle benötigten Berechtigungen aus
4. Speichert Gruppe für Zuweisung an Benutzer
   Betroffene Felder: RechteGruppeName, Beschreibung, RechteI3D (Array), IsActive
   Auswirkungen:

- Schnelle Benutzer-Einrichtung durch Gruppen-Zuweisung
- Konsistente Berechtigungen für gleiche Rollen
- Zentrale Änderung wirkt auf alle Gruppenmitglieder

2.9.2 Benutzer zu Rechtegruppe zuweisen

Zweck: Mitarbeitern ihre Berechtigungen durch Gruppenzugehörigkeit erteilen
Ablauf:

1. Administrator öffnet Benutzerverwaltung und wählt Mitarbeiter aus
2. Navigiert zu Tab "Rechte" und klickt auf "Gruppe hinzufügen"
3. Wählt eine oder mehrere Rechtegruppen aus (z.B. "Service-Techniker" + "CRM-Lesezugriff")
4. System addiert alle Rechte der ausgewählten Gruppen
   Betroffene Felder: BenutzerI3D, RechteGruppeI3D, GültigVon, GültigBis
   Auswirkungen:

- Benutzer erhält Zugriff auf zugewiesene Module
- Mehrfache Gruppenzugehörigkeit kombiniert Berechtigungen
- Zeitgesteuerte Rechtevergabe möglich

2.9.3 Einzelrechte gezielt erteilen oder entziehen

Zweck: Feinabstimmung von Berechtigungen unabhängig von Gruppenrechten
Ablauf:

1. Administrator öffnet Benutzer-Rechte und aktiviert "Erweiterte Ansicht"
2. Erteilt zusätzliche Einzelrechte (z.B. "Rechnungen stornieren") explizit
3. Entzieht spezifische Rechte (z.B. "Preise ändern") durch Negativ-Eintrag

4. Einzelrechte überschreiben immer Gruppenrechte
   Betroffene Felder: BenutzerId3D, RechtId3D, Erlaubnis (Granted/Denied), Priorität
   Auswirkungen:

- Flexible Ausnahmen von Standard-Rollen
- Temporäre Sonderrechte für Projekte
- Explizites Verbot von Rechten trotz Gruppenzugehörigkeit

2.9.4 Rechte-Audit und Compliance-Bericht

Zweck: Überprüfen, welche Benutzer auf sensible Funktionen zugreifen können
Ablauf:

1. Compliance-Beauftragter öffnet Rechteverwaltung → Auswertungen
2. Wählt kritisches Recht aus (z.B. "Belege löschen", "Stammdaten exportieren")
3. System zeigt alle Benutzer mit diesem Recht (direkt oder über Gruppen)
4. Export als Excel für Dokumentation bei Audits
   Betroffene Felder: RechtId3D, BenutzerId3D, RechteGruppeId3D, ZuweisungsGrund, LetzteÄnderung
   Auswirkungen:

- Transparenz über Berechtigungsstruktur
- Erfüllung von Compliance-Anforderungen (GoBD, DSGVO)
- Identifikation übermäßiger Rechtevergabe

2.10 Textbaustein Verwaltung (Text Module Management)

Module Path: `src/centron/Centron.WPF.UI/Modules/Administration/TextBlockManagement`
Controller: `TextBlockManagementAppModuleController`
ViewModel: `TextBlockManagementViewModel`
Category: Administration
Description: Durch dieses Modul können die Textbausteine der c-entron verwaltet werden

Use Cases

2.10.1 Standard-Textbaustein erstellen

Zweck: Häufig verwendete Texte als wiederverwendbare Bausteine definieren
Ablauf:

1. Administrator öffnet Textbaustein-Verwaltung und klickt auf "Neuer Baustein"
2. Wählt Kategorie (Angebot, Rechnung, E-Mail, Ticket-Antwort)
3. Gibt Kürzel (z.B. "WILLKOMMEN"), Titel und formatierten Text ein
4. Fügt Platzhalter ein wie {Firmenname}, {Ansprechpartner}, {Datum}
   Betroffene Felder: TextbausteinKürzel, Titel, TextInhalt, KategorieId3D, Platzhalter, IsHTML
   Auswirkungen:

- Schnelle Texteingabe durch Kürzel-Eingabe
- Konsistente Formulierungen in Dokumenten
- Zeitersparnis bei wiederkehrenden Texten

2.10.2 Kategorisierung und Verschlagwortung

Zweck: Textbausteine thematisch ordnen für schnelles Auffinden
Ablauf:

1. Administrator wählt Textbaustein aus und öffnet Details
2. Ordnet eine oder mehrere Kategorien zu (z.B. "AGB", "Garantiebedingungen", "Technische Hinweise")
3. Vergibt Schlagwörter (Tags) wie "Hardware", "Software", "Zahlungsbedingungen"

4. System bietet bei Eingabe in Dokumenten passende Bausteine basierend auf Kontext
   Betroffene Felder: TextbausteinI3D, KategorieI3D, Tags, Verwendungszweck
   Auswirkungen:

- Kontextabhängige Vorschläge beim Dokumentenschreiben
- Filterung nach Themengebieten
- Bessere Übersicht bei vielen Bausteinen

2.10.3 Mehrsprachige Textbausteine

Zweck: Textbausteine in Deutsch und Englisch für internationale Kunden bereitstellen
Ablauf:

1. Administrator öffnet deutschen Textbaustein
2. Klickt auf "Übersetzung hinzufügen" und wählt Englisch
3. Gibt englische Version mit denselben Platzhaltern ein
4. System wählt automatisch passende Sprache basierend auf Kundenstammdaten
   Betroffene Felder: TextbausteinI3D, SpracheCode, TextInhaltDE, TextInhaltEN
   Auswirkungen:

- Professionelle Ansprache internationaler Kunden
- Gleiche Textqualität in allen Sprachen
- Automatische Sprachwahl in Dokumenten

2.10.4 Textbausteine in Vorlagen einbetten

Zweck: Vordefinierte Dokumente mit automatisch eingefügten Textbausteinen erstellen
Ablauf:

1. Administrator erstellt Word- oder PDF-Vorlage für Angebote
2. Fügt Platzhalter für Textbausteine ein: {{TEXTBAUSTEIN:AGB}}
3. System ersetzt beim Dokumentengenerieren Platzhalter durch aktuelle Bausteine
4. Änderungen an Bausteinen wirken automatisch in allen Vorlagen
   Betroffene Felder: VorlagenI3D, TextbausteinKürzel, PlatzhalterPosition
   Auswirkungen:

- Zentrale Pflege von Standardtexten
- Konsistenz über alle Dokumente hinweg
- Automatische Aktualisierung bei Textänderungen

2.11 Ticketprozess Vorlagen (Ticket Process Templates)

Module Path: `src/centron/Centron.WPF.UI/Modules/Administration/Settings` (Ticketing Sub-Section)
Controller: Settings Module
ViewModel: Settings Module
Category: Administration
Description: In diesem Modul können Vorlagen für Ticketprozesse verwaltet werden

Use Cases

2.11.1 Standard-Ticketprozess definieren

Zweck: Vordefinierte Workflow-Schritte für häufige Service-Anfragen erstellen
Ablauf:

1. Administrator öffnet Einstellungen → Ticketing → Prozessvorlagen
2. Erstellt neue Vorlage "Hardware-Austausch" mit Schritten: Anfrage → Diagnose → Bestellung → Einbau → Test → Abschluss
3. Definiert für jeden Schritt: Verantwortliche Rolle, geschätzte Dauer, Pflichtfelder

4. Legt Übergangsbedingungen fest (z.B. Diagnose abgeschlossen → Automatisch "Bestellung" erstellen)
   Betroffene Felder: VorlagenName, ProzessSchritte, RolleI3D, DauerMinuten, Pflichtfelder, Übergangsbedingungen
   Auswirkungen:

- Standardisierte Ticket-Abwicklung
- Automatische Schrittfolge reduziert Fehler
- Transparente Prozesse für Kunden und Mitarbeiter

2.11.2 SLA-Zeiten in Prozessvorlage

Zweck: Service-Level-Agreements mit Reaktions- und Lösungszeiten hinterlegen
Ablauf:

1. Administrator öffnet Prozessvorlage (z.B. "Kritischer Störfall")
2. Definiert SLA-Parameter: Reaktionszeit 2 Stunden, Lösungszeit 8 Stunden
3. Legt Eskalationsregeln fest (nach 1h Ticket-Priorität erhöhen, nach 6h Manager informieren)
4. System überwacht Zeiten automatisch und löst Aktionen aus
   Betroffene Felder: VorlagenI3D, ReaktionszeitMinuten, LösungszeitMinuten, EskalationsStufen, BenachrichtigungsEmpfänger
   Auswirkungen:

- Vertragskonforme Ticket-Bearbeitung
- Automatische Eskalation verhindert SLA-Verletzungen
- Transparente Zeitmessung für Reporting

2.11.3 Checklisten in Prozessschritten

Zweck: Prüfpunkte in Ticket-Prozessen verankern für Qualitätssicherung
Ablauf:

1. Administrator wählt Prozessschritt "Abschluss" aus
2. Fügt Checkliste hinzu: "Kunde kontaktiert", "Dokumentation aktualisiert", "Hardware inventarisiert"
3. Macht Checklisten-Vollständigkeit zur Bedingung für Schritt-Abschluss
4. System zeigt Checkboxen im Ticket für verantwortlichen Techniker
   Betroffene Felder: ProzessSchrittI3D, ChecklistenItems, IstPflichtfeld, AbschlussValidierung
   Auswirkungen:

- Verhindert vergessene Tätigkeiten
- Qualitätssicherung durch strukturierte Prüfung
- Nachvollziehbarkeit durch dokumentierte Checks

2.11.4 Automatische Ticket-Klassifizierung

Zweck: Neue Tickets automatisch passender Prozessvorlage zuordnen
Ablauf:

1. Administrator öffnet Prozessvorlage und konfiguriert "Auto-Zuweisung"
2. Definiert Regeln: Betreff enthält "Server" UND Priorität = "Hoch" → Vorlage "Kritischer Störfall"
3. Legt Standardwerte für automatisch erstellte Tickets fest (Zuständigkeit, Kategorie)
4. System wendet beim Ticket-Eingang Regeln an und startet Workflow
   Betroffene Felder: VorlagenI3D, ZuweisungsRegeln, BetreffKeywords, PrioritätI3D, AutoStart
   Auswirkungen:

- Konsistente Ticket-Behandlung ohne manuelle Klassifizierung
- Sofortiger Prozessstart spart Zeit
- Fehlerreduktion durch regelbasierte Zuweisung

2.12 Vertragsarten (Contract Types)

Module Path: `src/centron/Centron.WPF.UI/Modules/Administration/Settings` (Contract Configuration Sub-Section)
Controller: Settings Module

ViewModel: Settings Module
Category: Administration
Description: Vertragsarten verwalten

Use Cases

2.12.1 Vertragsart für Wartungsverträge definieren

Zweck: Standardisierte Vertragstypen für Service-Vereinbarungen anlegen
Ablauf:

1. Administrator öffnet Einstellungen → Verträge → Vertragsarten
2. Erstellt neue Vertragsart "Hardware-Wartung" mit Standard-Laufzeit 12 Monate
3. Definiert Verlängerungsoptionen (automatisch, manuell, Kündigungsfrist)
4. Legt Abrechnungsintervall (monatlich, vierteljährlich, jährlich) fest
   Betroffene Felder: VertragsartName, StandardLaufzeitMonate, Verlängerungstyp, Kündigungsfrist, Abrechnungsintervall
   Auswirkungen:

- Schnelle Vertragserstellung durch Vorlagen
- Einheitliche Vertragsbedingungen
- Automatische Vertragsverlängerungen reduzieren Administrationsaufwand

2.12.2 Leasing-Vertragsart mit Rückgabebedingungen

Zweck: Spezielle Vertragsarten für Hardware-Leasing mit Rücknahmeregelungen
Ablauf:

1. Administrator erstellt Vertragsart "Hardware-Leasing 36 Monate"
2. Aktiviert Option "Rückgabe erforderlich" und definiert Restwert-Berechnung
3. Hinterlegt Bedingungen für Vertragsverlängerung oder Kauf nach Laufzeit
4. Verknüpft mit Leasing-Sätzen aus Modul 2.5
   Betroffene Felder: VertragsartI3D, RückgabePflicht, RestwertBerechnung, VerlängerungsOptionen, LeasingSatzI3D
   Auswirkungen:

- Automatische Erinnerung an Vertragsende für Rücknahme
- Korrekte Restwert-Abrechnung bei vorzeitiger Beendigung
- Integration mit Lagerverwaltung für Rücknahmen

2.12.3 SLA-Vertragsart mit Reaktionszeiten

Zweck: Service-Level-Verträge mit garantierten Antwort- und Lösungszeiten
Ablauf:

1. Administrator erstellt Vertragsart "Premium-Support 24/7"
2. Definiert garantierte Reaktionszeit (2h), Lösungszeit (8h), Verfügbarkeit (99,5%)
3. Verknüpft mit Ticketprozess-Vorlagen für automatische Priorisierung
4. Hinterlegt Strafzahlungen bei SLA-Verletzung
   Betroffene Felder: VertragsartI3D, ReaktionszeitStunden, LösungszeitStunden, Verfügbarkeit, TicketProzessVorlageI3D, PenaltyRegelung
   Auswirkungen:

- Tickets von Premium-Kunden werden automatisch priorisiert
- SLA-Überwachung erfolgt systemgestützt
- Automatische Eskalation bei Fristverzug

2.12.4 Vertragsarten-Genehmigungsworkflow

Zweck: Mehrere Genehmigungsstufen für hochwertige oder komplexe Verträge
Ablauf:

1. Administrator öffnet Vertragsart und aktiviert "Genehmigungspflichtig"
2. Definiert Schwellenwerte (z.B. Verträge über 10.000 EUR/Jahr benötigen Geschäftsführungs-Genehmigung)
3. Legt Genehmigungskette fest: Vertrieb → Abteilungsleiter → Geschäftsführung

4. System blockiert Vertragsaktivierung bis alle Stufen genehmigt haben
   Betroffene Felder: VertragsartI3D, GenehmigungErforderlich, Schwellenwerte, GenehmigungsStufen, Genehmiger
   Auswirkungen:

- Risikokontrolle bei großen Verträgen
- Transparenter Genehmigungsstatus für alle Beteiligten
- Audit-Trail für Compliance-Nachweise

3. Adressen/CRM (Addresses/CRM)

3.1 Adressstamm (Address Master Data)

Module Path: `src/centron/Centron.WPF.UI/Modules/Finances/Crm`
Controller: `CrmAppModuleController`
ViewModel: `CrmMainViewModel`
Category: Adressen/CRM
Description: Zentrale Verwaltung aller Geschäftspartner-Adressen (Kunden, Lieferanten, Interessenten) mit Kontaktpersonen, Aktivitätshistorie und CRM-Notizen

Use Cases

3.1.1 Neue Kundenadresse erfassen

Zweck: Neue Kundenadressen in das System aufnehmen

Ablauf:

1. Benutzer klickt auf "Neue Adresse" im Menü
2. CRM-Formular wird geöffnet mit Feldern: Firma, Adresse, Stadt, PLZ, Land, Telefon, E-Mail
3. Benutzer wählt Adresstyp (Kunde, Lieferant, Interessent) aus Dropdown
4. Erforderliche Felder werden mit Asterisk gekennzeichnet
5. Nach Validierung wird die Adresse in Datenbank gespeichert
6. System generiert automatisch Kunden-ID
7. Bestätigung mit Meldung "Adresse erfolgreich erstellt"

Betroffene Felder: Account (Name, Street, PostalCode, City, Country), AccountType, CreatedDate, CreatedByI3D

Auswirkungen: Neue Adresse ist sofort in allen Modul verfügbar (Verkauf, Einkauf, Rechnungen). Automatische Zuweisung zu Verkaufsgebiet basierend auf PLZ. Verfügbar für Kampagnen und Mailing.

3.1.2 Kontaktperson zu Adresse hinzufügen

Zweck: Verwaltung von Ansprechpartnern bei Kundenadressen

Ablauf:

1. Benutzer öffnet Kundenadresse im CRM
2. Tab "Kontaktpersonen" wird ausgewählt
3. Button "Neue Kontaktperson" wird geklickt
4. Eingabedialog mit Feldern: Vorname, Name, Titel, Telefon, E-Mail, Abteilung
5. Benutzer markiert Kontaktperson als "Primärkontakt" für Rechnungen/Versand
6. Kontaktperson wird der Adresse zugeordnet und gespeichert
7. Kontaktperson ist nun in Versand-Modulen selektierbar

Betroffene Felder: ContactPerson (FirstName, LastName, PhoneNumber, Email, Department), Account (PrimaryContactPersonI3D)

Auswirkungen: Automatische Vorauswahl bei Lieferantenadressen und Rechnungserstellung. Wenn Primärkontakt gelöscht wird, wird Warnung angezeigt. Mail-Merge Vorlagen können auf Kontaktdaten zugreifen.

3.1.3 Adresse aktualisieren

Zweck: Änderungen an Kundenadressdaten durchführen

Ablauf:

1. Benutzer sucht Adresse in CRM über Suchleiste
2. Adresse wird geöffnet im Bearbeitungsmodus
3. Benutzer ändert relevante Felder (z.B. Telefonnummer, E-Mail, Adresse)
4. System zeigt "Änderungen ausstehend" Badge
5. Benutzer klickt "Speichern"
6. System überprüft auf Duplikate
7. Änderungen werden gespeichert mit Audit-Trail (Wer, Wann, Was)
8. System aktualisiert automatisch verknüpfte Datensätze

Betroffene Felder: Account (Street, PostalCode, City, Country, PhoneNumber, Email), ChangedDate, ChangedByI3D

Auswirkungen: Änderungen werden in Rechnungen und Versandetiketten sofort wirksam. Alter Wert wird in Audit-Tabelle archived. E-Mail-Benachrichtigungen an verknüpfte Mitarbeiter (wenn konfiguriert).

3.1.4 Adresse mit Aktivitätshistorie anzeigen

Zweck: Übersicht aller Interaktionen mit einer Adresse/Geschäftspartner

Ablauf:

1. Benutzer öffnet Kundenadresse
2. Tab "Aktivitätshistorie" wird angewählt
3. System zeigt zeitlich sortierte Liste: Anrufe, E-Mails, Treffen, Verkaufs-Chancen, erstellte Rechnungen
4. Jeder Eintrag zeigt Datum, Typ, Benutzer, Beschreibung
5. Benutzer kann Details-Popup öffnen für jeden Eintrag
6. Filter nach Datum-Bereich oder Aktivitätstyp möglich
7. Export zu Excel für Bericht möglich

Betroffene Felder: Activity (AccountI3D, Type, CreatedDate, CreatedByI3D, Description), ActivityHistory (archivierte Activities)

Auswirkungen: Unterstützt Vertriebsteam-Handoff. Dokumentation von Kundenbeziehungen für Compliance. Automatische Speicherung von Email-Kopien (wenn konfiguriert).

3.1.5 Duplicate-Check und Adress-Merge

Zweck: Vermeidung von doppelten Einträgen und Bereinigung von Adressdatensätzen

Ablauf:

1. Bei Eingabe neuer Adresse führt System automatisch Duplikat-Prüfung durch
2. Wenn ähnliche Adresse gefunden wird, zeigt System Warnung mit Vorschlag
3. Benutzer kann "Nicht duplizieren" bestätigen oder "Merge durchführen" wählen
4. Im Merge-Dialog: System schlägt Datensätze zur Zusammenführung vor
5. Benutzer wählt Felder aus "Haupt-Adresse" und "Zu-Mergender-Adresse"
6. Alle verknüpften Datensätze (Rechnungen, Verträge) werden auf Haupt-Adresse umgeleitet
7. Alte Adresse wird als "merged" markiert (nicht gelöscht)

Betroffene Felder: Account (IsDeleted, MergedWithI3D), verknüpfte Fremdschlüssel

Auswirkungen: Konsistente Datenbasis. Vermeidung von Reporting-Fehlern durch Duplikate. Verkaufshistorie wird konsolidiert. Adress-Qualität verbessert sich.

3.1.6 Adresse löschen oder archivieren

Zweck: Entfernen von inaktiven oder fehlerhaften Adressdatensätzen

Ablauf:

1. Benutzer klickt auf Adresse und wählt "Archivieren" oder "Löschen"
2. System prüft auf verknüpfte Datensätze (offene Rechnungen, Verträge, Tickets)
3. Wenn verknüpfte Datensätze existieren: Warnung wird angezeigt mit Details
4. Bei "Archivieren": Adresse wird als inaktiv markiert (soft delete), bleibt aber sichtbar im Audit
5. Bei "Löschen": Benutzer muss Grund angeben, System protokolliert in Audit-Log
6. Benutzer muss Bestätigung geben
7. Adresse wird aus aktiven Listen entfernt

Betroffene Felder: Account (IsDeleted, DeletedDate, DeletedByI3D), Audit-Log

Auswirkungen: Archivierte Adressen bleiben für Reporting und Audit-Trail erhalten. Rechnungen zeigen Kundennamen auch wenn Adresse gelöscht. Soft-Delete ermöglicht Wiederherstellung. Compliance-relevant für DSGVO.

3.2 Audit

Module Path: `src/centron/Centron.WPF.UI/Modules/Finances/Crm` (Tab)
Controller: `AuditTabViewModel`
ViewModel: `AuditTabViewModel`
Category: Adressen/CRM
Description: Durchführung und Verwaltung von Kundenaudits mit Checklisten, Audit-Berichten und Abweichungsmanagement

Use Cases

3.2.1 Neues Audit erstellen

Zweck: Durchführung von Kundenaudits mit standardisierten Checklisten

Ablauf:

1. Benutzer öffnet Kundenadresse und wählt Tab "Audit"
2. Button "Neues Audit" wird geklickt
3. Dialog mit Optionen: Audit-Template wählen, Audit-Datum setzen, Auditor zuweisen
4. Benutzer wählt Template aus vordefinierten Audit-Checklisten (z.B. "Qualitätsaudit", "Datenschutz-Audit")
5. System erstellt neues Audit-Objekt und zeigt Checklisten-Fragen
6. Audit wird mit Status "In Bearbeitung" initialisiert
7. Audit ist nun für Feldarbeit verfügbar

Betroffene Felder: Activity (Type = "Audit"), ActivityTemplate, AuditChecklist, AuditI3D, AssignedUserI3D

Auswirkungen: Audit wird in Kundenzeitlinie dokumentiert. Automatische Benachrichtigung an zugewiesenen Auditor. Audit-Ergebnisse beeinflussen Kundenbewertung und werden in Compliance-Reports angezeigt.

3.2.2 Audit durchführen und Fragen beantworten

Zweck: Erfassung von Audit-Ergebnissen vor Ort oder remote

Ablauf:

1. Auditor öffnet zugewiesenes Audit
2. System zeigt Checklisten-Fragen einzeln oder als Liste
3. Für jede Frage: Auditor wählt Antwort (Ja/Nein/N.A./Mit Mängel)
4. Bei "Mit Mängel": Freitextfeld für Beschreibung und Foto-Upload möglich
5. Auditor kann Notizen und Datum für jede Antwort hinzufügen
6. System speichert Antworten lokal und sync mit Server
7. Nach Fertigstellung: Benutzer klickt "Audit Abschließen"

Betroffene Felder: AuditChecklistAnswer (Question, Answer, Notes, PhotoPath), AuditI3D, CompletedDate, CompletedByI3D

Auswirkungen: Abgeschlossene Audits können nicht mehr bearbeitet werden (Audit-Trail). Abweichungen (Mängel) generieren automatisch Verbesserungsmaßnahmen. Trends werden in Kundenbewertung berechnet.

3.2.3 Audit-Report erstellen

Zweck: Generierung von professionellen Audit-Berichten mit Findings und Recommendations

Ablauf:

1. Nach Abschluss des Audits kann Benutzer Report generieren
2. System wählt Report-Template (Standard, Management-Summary, Detailliert)
3. Report enthält: Datum, Auditor, Abweichungen, Fotos, Trend-Analyse
4. Report wird als PDF oder Word-Dokument generiert
5. Benutzer kann Report vor Export bearbeiten und Kommentare hinzufügen
6. Report wird signiert und mit Audit-Datensatz archiviert
7. Customer erhält automatisch Kopie per Email (wenn konfiguriert)

Betroffene Felder: AuditReport (Content, Template, GeneratedDate, GeneratedByI3D), AuditI3D

Auswirkungen: Audit-Reports sind rechtsgültig und für Compliance erforderlich. Kann als Nachweis für Kundenaudits dienen. Report-History bleibt für DSGVO-Anfragen erhalten.

3.2.4 Abweichungen managen

Zweck: Verfolgung und Behebung von bei Audits identifizierten Abweichungen

Ablauf:

1. System identifiziert automatisch alle "Mit Mängel" Antworten als Abweichungen
2. Abweichungen werden mit Schweregrad (Kritisch, Major, Minor) bewertet
3. Benutzer kann Abweichung editieren: Beschreibung anpassen, Frist für Behebung setzen
4. Automatische Erstellung von Ticket für Kundenteam (wenn aktiviert)
5. Status verfolgbar: Offen → Geplant → In Bearbeitung → Geschlossen → Verifiziert
6. Benutzer kann Verify-Audit durchführen nach Abweichwege-Beseitigung
7. Abgeschlossene Abweichungen werden dokumentiert

Betroffene Felder: AuditDeviation (SeverityLevel, Description, DueDate, Status), TicketI3D, AssignedUserI3D

Auswirkungen: Kritische Abweichungen eskalieren automatisch an Management. Abweichungsquoten beeinflussen Kundenbewertung. Reports zeigen Abweichungs-Trends.

3.2.5 Audit-History und Trend-Analyse

Zweck: Verfolgung von wiederholten Audits und Identifikation von Verbesserungstrends

Ablauf:

1. Benutzer öffnet Kundenadresse und sucht nach bisherigen Audits
2. System zeigt Liste aller durchgeführten Audits (Datum, Auditor, Status)
3. Benutzer kann Audit vergleichen: Alte vs. neue Ergebnisse
4. System generiert Trend-Chart: Bestehungsquote über Zeit
5. Automatische Berechnung von Verbesserungsrate (z.B. "von 85% auf 92% bestanden")
6. Benutzer kann Export als Bericht für Kundengespräche nutzen

Betroffene Felder: Activity (Type="Audit", DateRange), AuditChecklistAnswer (Historical), Trend-Daten

Auswirkungen: Demonstriert Kundenfortschritt in Audits. Identifiziert systematische Probleme. Daten für Kundenbeziehungs-Management nützlich.

3.2.6 Audit-Templates verwalten

Zweck: Verwaltung und Anpassung von standardisierten Audit-Checklisten

Ablauf:

1. Admin öffnet Audit-Template-Verwaltung
2. Benutzer kann vordefinierte Templates ansehen oder neue erstellen
3. Template besteht aus: Name, Beschreibung, Kategorie, Fragen-Liste
4. Jede Frage hat: Text, Antworttyp (Ja/Nein, Multiple Choice), Punkt-Gewichtung
5. Admin kann Template als "Aktiv", "Draft", oder "Archiviert" markieren
6. Bei Änderungen können alte Audits mit Template aktualisiert werden (optional)
7. Audit-Manager kann Template versionieren

Betroffene Felder: AuditTemplate (Name, Description, Questions, IsActive, Version, ChangedDate, ChangedByI3D)

Auswirkungen: Standardisierte Audits gewährleisten Vergleichbarkeit. New Templates sind sofort für neue Audits verfügbar. Alte Templates können für Audit-Revisionen verwendet werden.

3.3 CRM-Projekte (CRM Projects)

Module Path: `src/centron/Centron.WPF.UI/Modules/Finances/Crm/Settings/CRMProject`
Controller: `CrmProjectSettingsController`
ViewModel: `CrmProjectSettingsViewModel`
Category: Adressen/CRM
Description: Verwaltung von Kundenprojekten zur Strukturierung von Verkaufs-, Service- und Supportaktivitäten pro Geschäftspartner

Use Cases

3.3.1 Neues CRM-Projekt erstellen

Zweck: Erstellung von Projektstrukturen für Kundenbeziehungen und Sachbearbeitungskontexte

Ablauf:

1. Benutzer öffnet Kundenadresse und navigiert zu "Projekte"
2. Button "Neues Projekt" wird geklickt
3. Dialog mit Feldern: Projektname, Projektleiter, Startdatum, Zieldatum, Budget
4. Benutzer wählt Projekttyp (Implementierung, Support, Weiterentwicklung, etc.)
5. Optional: Projektbeschreibung, Technologie-Stack, Zielgruppe eingeben
6. Projekt wird mit Status "Geplant" erstellt
7. Projekt erhält automatische ID für Referenzierung

Betroffene Felder: CrmProject (Name, ProjectManagerI3D, StartDate, EndDate, Budget, Type, Description, AccountI3D)

Auswirkungen: Projekt wird in Kundenzeitlinie sichtbar. Automatisch verlinkt mit Adresse. Alle zugehörigen Tickets/Aktivitäten können diesem Projekt zugeordnet werden. Ermöglicht Projekt-Budgetierung und -Controlling.

3.3.2 Projektteam zusammenstellen

Zweck: Zuweisung von Mitarbeitern und Rollen zu CRM-Projekten

Ablauf:

1. Benutzer öffnet CRM-Projekt im Bearbeitungsmodus
2. Tab "Team" wird angewählt
3. Button "Mitarbeiter hinzufügen" wird geklickt
4. Dialog mit Liste der verfügbaren Mitarbeiter
5. Benutzer wählt Mitarbeiter und weist Rolle zu (Projektleiter, Entwickler, Supporter, etc.)
6. Optional: Auslastung in % für dieses Projekt angeben
7. Zeitraum der Zugehörigkeit definieren (von/bis Datum)
8. Mitarbeiter wird zu Projektteam hinzugefügt

Betroffene Felder: CrmProjectTeam (CrmProjectI3D, EmployeeI3D, Role, Percentage, StartDate, EndDate)

Auswirkungen: Mitarbeiterlisten aktualisieren sich. Mitarbeiter sehen Projektliste in ihrem Dashboard. Automatische Benachrichtigung an Projektteam. Auslastungs-Reports berücksichtigen Projektauslastung.

3.3.3 Projekt-Phasen definieren

Zweck: Strukturierung von Projekten in logische Arbeitsphasen

Ablauf:

1. Benutzer öffnet CRM-Projekt
2. Tab "Phasen" wird angewählt
3. Button "Neue Phase" wird geklickt
4. Benutzer definiert: Phasenname, Startdatum, Enddatum, Verantwortlicher
5. Optional: Budget für Phase und Meilensteine hinzufügen
6. System erlaubt verschachtelte Phasen (Phase → Unterphasen)
7. Phasen können als "Geplant", "In Bearbeitung", "Fertig", "On Hold" markiert werden
8. Benutzer kann Abhängigkeiten zwischen Phasen definieren (Phase B startet nach Phase A)

Betroffene Felder: CrmProjectPhase (CrmProjectI3D, Name, StartDate, EndDate, Status, BudgetI3D, ParentPhaseI3D)

Auswirkungen: Phasen-basierte Kontrolle ermöglicht besseres Projekt-Management. Meilensteine triggern automatische Benachrichtigungen. Phase-Abschluss kann Rechnungsprozess auslösen (wenn konfiguriert).

3.3.4 Budget und Kosten-Tracking

Zweck: Überwachung von Projektbudgets und Dokumentation von Projektkosten

Ablauf:

1. Benutzer öffnet CRM-Projekt
2. Tab "Budget & Kosten" wird angewählt
3. System zeigt Gesamtbudget und Ausgegeben-Summe
4. Benutzer kann Kostenstellen erfassen: Materialien, Personalstunden, Subunternehmer
5. Zeiterfassungen von Mitarbeitern werden automatisch aggregiert
6. System warnt bei Budget-Überschreitung (80%, 95%, 100%)
7. Benutzer kann Prognose für Restbudget anpassen
8. Report zeigt Kosten nach Kategorie und zeitlichen Verlauf

Betroffene Felder: CrmProject (Budget, SpentAmount), CrmProjectCost (Amount, Category, Date, EmployeeI3D), TimeEntry (CrmProjectI3D, Hours)

Auswirkungen: Genaue Projektrentabilität berechnen. Budget-Überschreitungen eskalieren zu Management. Ermöglicht profitable Preisgestaltung für zukünftige Projekte. Basisinformation für Projekt-Abrechnung.

3.3.5 Projekt-Status und Fortschritt

Zweck: Verfolgung des Projekt-Fortschritts und Status-Kommunikation

Ablauf:

1. Benutzer öffnet CRM-Projekt
2. System zeigt Projekt-Übersicht mit Fortschrittsbalken (% abgeschlossen)
3. Fortschritt wird basierend auf abgeschlossenen Phasen/Aufgaben berechnet
4. Benutzer kann manuell Fortschritt aktualisieren oder Notizen hinzufügen
5. "Status-Report" kann mit Knopfdruck generiert und per Email versendet werden
6. Projekt-Status kann sein: Geplant → In Bearbeitung → Im Plan → Verzögert → Abgeschlossen
7. Status-Änderungen triggern automatische Benachrichtigungen an Stakeholder

Betroffene Felder: CrmProject (Status, ProgressPercentage, LastUpdatedDate), CrmProjectStatusHistory (Status, ChangedDate, Notes)

Auswirkungen: Automatische Status-Updates an Kundenportal (wenn aktiviert). Verzögerungen identifizieren sich automatisch. Unterstützt Eskalations-Prozesse bei Problemen.

3.3.6 Projekt-Abschluss und Archivierung

Zweck: Formaler Abschluss und Archivierung abgeschlossener Projekte

Ablauf:

1. Benutzer öffnet abgeschlossenes Projekt
2. Button "Projekt Abschließen" wird geklickt
3. System prüft: Alle Phasen beendet? Budget finalisiert? Dokumentation vollständig?
4. Abschließungs-Checkliste wird angezeigt
5. Nach Bestätigung: Projekt wird als "Abgeschlossen" markiert
6. Abschluß-Report wird generiert: Zeitaufwand, Budget-Verbrauch, Ergebnisse
7. Projekt wird archiviert, bleibt aber für Reporting und historische Auswertungen sichtbar
8. Kunde erhält formale Abschluss-Notifikation (optional)

Betroffene Felder: CrmProject (Status="Completed", ClosedDate, ClosedByI3D), CrmProjectClosureReport

Auswirkungen: Abgeschlossene Projekte können nicht mehr bearbeitet werden. Ergebnisse für Lessons-Learned verfügbar. Daten für Benchmarking zukünftiger Projekte. Kunden-Zufriedenheits-Umfragen können ausgelöst werden.

3.4 Kampagnen/Mailing (Campaigns/Mailing)

Module Path: `src/centron/Centron.WPF.UI/Modules/Finances/Campaigns`
Controller: `CampaignAppModuleController`
ViewModel: `CampaignMainViewModel`
Category: Adressen/CRM
Description: Verwaltung von Marketing-Kampagnen mit Zielgruppen-Definition, Mailing-Erstellung und Erfolgs-Tracking

Use Cases

3.4.1 Neue Kampagne erstellen

Zweck: Initialisierung von Marketing-Kampagnen zur gezielten Kundenakquisition und -pflege

Ablauf:

1. Benutzer navigiert zu "Neue Kampagne"
2. Dialog mit Feldern: Kampagnenname, Beschreibung, Startdatum, Enddatum, Budget
3. Benutzer wählt Kampagnentyp (Email-Kampagne, Newsletter, Event-Einladung, Produktankündigung, etc.)
4. Zielgruppe wird definiert (Kundengruppe, Region, Branche, etc.)
5. Responsibles werden zugewiesen (Projektmanager, Autor)
6. Kampagne wird mit Status "Draft" erstellt
7. Benutzer kann Kampagne als Template speichern für zukünftige Nutzung

Betroffene Felder: Campaign (Name, Description, Type, StartDate, EndDate, Budget, Status, CreatedByI3D)

Auswirkungen: Kampagne wird in CRM-Übersicht sichtbar. Automatische Zielgruppen-Berechnung. Notifikation an Projektteam. ROI-Tracking wird vorbereitet.

3.4.2 Zielgruppe definieren

Zweck: Auswahl und Segmentierung von Empfänger für Kampagnen

Ablauf:

1. Benutzer öffnet Kampagne
2. Tab "Zielgruppe" wird angewählt
3. Benutzer kann Filter setzen: Kundentyp, Branche, Region, Größe, Kaufhistorie
4. System zeigt Anzahl der gefilterten Adressen in Echtzeit
5. Benutzer kann vordefinierte Segmente (z.B. "VIP-Kunden", "Inaktive Kunden") verwenden
6. Optional: Ausschluss-Listen (z.B. "bereits kontaktiert in letzen 3 Monaten")
7. Finale Zielgruppe wird angezeigt mit demografischen Daten
8. System warnt bei zu kleinen oder zu großen Zielgruppen

Betroffene Felder: Campaign (TargetSegmentI3D), CampaignTarget (AccountI3D, CampaignI3D)

Auswirkungen: Zielgruppen-Größe beeinflusst Budget-Berechnung. Segment-Daten verfügbar für Analyse. Ausschluss-Listen verhindern unerwünschte Kontakte.

3.4.3 Email-Kampagne / Newsletter erstellen

Zweck: Erstellung und Gestaltung von Email-Inhalten für Kampagnen

Ablauf:

1. Benutzer öffnet Kampagne und wählt Tab "Email-Vorlage"
2. Editor wird geöffnet (WYSIWYG oder HTML-Editor)
3. Benutzer kann Vorlagen-Blöcke verwenden: Header, Footer, Buttons, Texte, Bilder
4. Personalisierung möglich: {{Kundennname}}, {{Anrede}}, {{Branche}}
5. System validiert Links und Bilder
6. Vorschau zeigt Email in verschiedenen Clients (Outlook, Gmail, Mobile)
7. A/B-Test-Varianten können erstellt werden (Betreff-Zeile, CTA)
8. Email wird als Draft gespeichert oder geplant

Betroffene Felder: Campaign (Subject, BodyTemplate, FromAddress, ReplyToAddress), CampaignPhase (EmailTemplate)

Auswirkungen: Email-Vorlage bleibt für zukünftige Kampagnen verfügbar. Personalisierungsfelder erhöhen Click-Through-Rate. A/B-Tests ermöglichen Optimierung.

3.4.4 Kampagne planen und starten

Zweck: Zeitgesteuerte Aktivierung von Kampagnen und Versand-Automatisierung

Ablauf:

1. Benutzer öffnet fertige Kampagne im Status "Draft"
2. Tab "Versand-Planung" wird angewählt
3. Benutzer setzt Startdatum und Versandzeit
4. Optional: Staggered Release (z.B. 500 Emails pro Stunde, um Server nicht zu überlasten)
5. Bounce/Complaint Handling wird konfiguriert
6. Unsubscribe-Landingpage wird automatisch verlinkt
7. Nach Bestätigung: Kampagne wird auf Status "Geplant" gesetzt
8. System triggert automatischen Versand zu gesetztem Zeitpunkt

Betroffene Felder: Campaign (Status, ScheduledStartDate, ScheduledEndDate), CampaignSchedule (StartTime, EndTime, RateLimit)

Auswirkungen: Automatischer Versand entlastet Team. Staggered Release verhindert Spam-Flaggung. Bounce-Handling aktualisiert Adress-Daten. Unsubscribe-Rate wird automatisch gemessen.

3.4.5 Kampagnen-Performance tracken

Zweck: Erfassung und Analyse von Kampagnen-Ergebnissen (Öffnungsrate, Klicks, Conversions)

Ablauf:

1. Nach Kampagnen-Start: System zeigt Live-Dashboard mit Metriken
2. Verfügbare KPIs: Versand-Anzahl, Zustellungsrate, Öffnungsrate, Click-Through-Rate, Conversion-Rate
3. System zeigt Zeitverlauf: Öffnungen pro Stunde, Klicks pro Tag, etc.
4. Benutzer kann Geographische Daten anzeigen (Öffnungen nach Land/Stadt)
5. Link-Performance wird angezeigt: Welche Links wurden am meisten geklickt?
6. Kampagnen können live optimiert werden (z.B. Betreff-Zeile ändern bei schlechter Öffnungsrate)
7. Report wird automatisch täglich generiert

Betroffene Felder: CampaignTracking (SendCount, OpenCount, ClickCount, ConversionCount), CampaignLinkClick (LinkI3D, ClickCount)

Auswirkungen: Live-Daten ermöglichen schnelle Anpassungen. A/B-Test-Ergebnisse informieren zukünftige Kampagnen. ROI wird automatisch berechnet basierend auf Conversions.

3.4.6 Mailing-Kampagne abschließen und Report

Zweck: Archivierung von Kampagnen und Generierung von Abschluss-Berichten

Ablauf:

1. Nach Ende der Kampagne oder auf Anforderung: Button "Kampagne Abschließen" wird geklickt
2. System validiert: Alle Emails versendet? Tracking-Daten erfasst?
3. Finale Metriken werden berechnet: Gesamt-ROI, Kosteneffizienz pro Kontakt, Gesamt-Conversions
4. Report wird generiert mit Elementen: Zielgruppe, Versand-Daten, Performance-Charts, Lessons Learned
5. Benutzer kann Kampagne als "Best Practice" markieren
6. Kampagne wird archiviert und aus aktiver Liste entfernt
7. Opt-Out-Adressen werden verwaltet/archiviert

Betroffene Felder: Campaign (Status="Completed", EndDate, FinalROI), CampaignReport (Content, GeneratedDate)

Auswirkungen: Abgeschlossene Kampagnen bleibt im System für historische Vergleiche. Best-Practice-Kampagnen können Basis für neue Kampagnen sein. Opt-Out-Liste wird für DSGVO-Compliance-Reports verwendet.

3.5 Lieferanten-Verträge (Supplier Contracts)

Module Path: `src/centron/Centron.WPF.UI/Modules/Finances/Crm/AccountContracts`
Controller: `AccountContractsAppModuleController`
ViewModel: `AccountContractsManagementViewModel`
Category: Adressen/CRM
Description: Verwaltung von Lieferantenverträgen mit Vertragskonditionen, Leistungsvereinbarungen und Vertragserneuerungen

Use Cases

3.5.1 Neuen Lieferantenvertrag erstellen

Zweck: Erfassung von Lieferantenverträgen mit Konditionen und Zahlungsbedingungen

Ablauf:

1. Benutzer öffnet Lieferantenadrasse und navigiert zu "Verträge"
2. Button "Neuer Vertrag" wird geklickt
3. Dialog mit Feldern: Vertragsbezeichnung, Lieferant, Startdatum, Enddatum, Vertragswert
4. Benutzer wählt Vertragstyp (Leistungsvertrag, Rahmenvertrag, Wartungsvertrag, etc.)
5. Zahlungsbedingungen eingeben: Zahlungsfrist, Rabatt, Mahngebühren
6. Vertragstext wird hochgeladen oder in Editor eingegeben
7. Verantwortlicher und Freigeber werden zugewiesen
8. Vertrag wird mit Status "Draft" erstellt

Betroffene Felder: Contract (Name, SupplierI3D, StartDate, EndDate, ContractValue, Type, Terms, CreatedByI3D)

Auswirkungen: Vertrag wird mit Lieferant verlinkt. Automatische Benachrichtigung bei Ablauf. Zahlungsbedingungen beeinflussen Rechnungsverarbeitung. Frühwarnung bei Erneuerungsbedarf.

3.5.2 Vertrag genehmigen und aktivieren

Zweck: Freigabeprozess für Lieferantenverträge mit Autorisierungen

Ablauf:

1. Benutzer öffnet Vertrag im Status "Draft"
2. Tab "Genehmigungen" wird angewählt
3. System zeigt Genehmigungs-Workflow: Ersteller → Abteilungsleiter → Manager
4. Freigeber kann Vertrag anschauen und kommentieren
5. Freigeber klickt "Genehmigt" oder "Abgelehnt"
6. Bei Ablehnung: Kommentar erforderlich, Vertrag geht zurück an Ersteller
7. Nach finaler Genehmigung: Vertrag wird aktiviert (Status "Active")
8. Automatische Benachrichtigung an Lieferant (optional)

Betroffene Felder: Contract (Status, ApprovedByI3D, ApprovedDate), ContractApprovalHistory (ApprovalStep, ApprovedByI3D, ApprovedDate, Comments)

Auswirkungen: Nur genehmigte Verträge sind rechtsgültig. Audit-Trail dokumentiert Freigabeprozess. Compliance-relevant für Governance.

3.5.3 Vertragskonditionen und SLA definieren

Zweck: Dokumentation von Service-Level-Agreements und Leistungs-KPIs

Ablauf:

1. Benutzer öffnet aktiven Vertrag

2. Tab "SLA & Konditionen" wird angewählt
3. Benutzer definiert Service-Level-Metriken: Verfügbarkeit (z.B. 99.5%), Response-Time, Resolution-Time
4. Benutzer definiert Eskalations-Pfade bei SLA-Verletzung
5. Strafklauseln können definiert werden (z.B. Gutschrift bei SLA-Verletzung)
6. Prioritäts-Matrix für Ticket-Handling wird konfiguriert
7. Support-Zeiten/Schichten werden dokumentiert
8. Metriken werden in aktive Monitoring-Systeme integriert

Betroffene Felder: Contract (SLATemplate), ContractSLA (MetricName, TargetValue, Penalty), TicketSLAPolicy

Auswirkungen: Ticket-Bearbeitung wird automatisch basierend auf Vertrags-SLA priorisiert. SLA-Verletzungen triggern automatische Eskalationen. Reports zeigen Compliance mit SLAs.

3.5.4 Vertrag überwachen und Performance tracking

Zweck: Laufende Überwachung der Vertragserfüllung und Performance-Metriken

Ablauf:

1. Benutzer öffnet Vertrag
2. Tab "Performance" wird angewählt
3. System zeigt Dashboard mit Metriken: Verfügbarkeit, Response-Time, Kundenfreude-Score
4. Live-Daten werden angezeigt (aktualisiert stündlich oder täglich)
5. Benutzer kann SLA-Verletzungen in Rot gekennzeichnet sehen
6. Benutzer kann Notizen hinzufügen zu Performance-Events
7. Berichte können exportiert werden für Lieferanten-Gespräche
8. System warnt bei Trends (z.B. sinkende Verfügbarkeit)

Betroffene Felder: ContractPerformanceMetric (MetricValue, MeasurementDate), Ticket (SLAStatus, ResolutionTime)

Auswirkungen: Frühzeitige Erkennung von Qualitätsproblemen. Datengrundlage für Verhandlungen. Dokumentation für Lieferanten-Management.

3.5.5 Vertrag erneuern oder beenden

Zweck: Verwaltung von Vertragserneuerungen und ordentlichen Beendigungen

Ablauf:

1. System identifiziert Verträge, die bald ablaufen (konfigurierbare Frist, z.B. 90 Tage)
2. Benutzer erhält automatische Benachrichtigung: "Vertrag endet in X Tagen"
3. Benutzer öffnet Vertrag und klickt "Erneuerung"
4. System erstellt neuen Vertrag mit aktuellen Daten, erlaubt Anpassungen
5. Alternativ: Benutzer klickt "Beenden" und muss Kündigungsdatum und Grund eingeben
6. System prüft auf offene Verpflichtungen (ausstehende Lagerbestände, Support-Fälle)
7. Nach Bestätigung: Alter Vertrag wird archiviert, neuer wird aktiviert
8. Lieferant erhält Benachrichtigung (optional)

Betroffene Felder: Contract (EndDate, Status="Ended"), ContractRenewal (PreviousContractI3D, NewContractI3D), ContractArchive

Auswirkungen: Automatische Vertragsmanagement-Effizienz. Keine unbeabsichtigten Vertragsverlängerungen. Audit-Pfad für Vertragszyklus-Management.

3.5.6 Vertragsrisiken und Compliance-Checks

Zweck: Identifikation von Compliance-Problemen und Vertragsrisiken

Ablauf:

1. Benutzer öffnet Vertrag
2. Tab "Compliance & Risiken" wird angewählt
3. System führt automatische Checks aus: DSGVO-Datenschutzklauseln, Versicherungsanforderungen, Audit-Rechte
4. Benutzer kann Risiken manuell hinzufügen (z.B. "Lieferant ist in instabilen Markt tätig")
5. Risikolevel wird berechnet: Grün (niedrig), Gelb (mittel), Rot (hoch)
6. Hohe Risiken generieren automatisch Review-Aufgaben
7. Compliance-Audit-Reports können generiert werden
8. Versicherungsanforderungen können direkt an Versicherungs-Team übermittelt werden

Betroffene Felder: Contract (ComplianceStatus, RiskLevel), ContractRiskAssessment (RiskType, RiskLevel, MitigationPlan)

Auswirkungen: Proaktive Risiko-Management. DSGVO-Compliance dokumentiert. Versicherungsanforderungen automatisch verwaltet. Regulatorische Anforderungen erfüllt.

3.6 PLM (Product Lifecycle Management)

Module Path: `src/centron/Centron.WPF.UI/Modules/PLM`
Controller: `PlmAppModuleController`
ViewModel: `PlmViewModel`
Category: Adressen/CRM
Description: Verwaltung von Produktlebenszyklen von der Konzeption bis zur Ausmusterung mit Revisions-Tracking und Konfigurationsmanagement

Use Cases

3.6.1 Neues Produkt im PLM erstellen

Zweck: Initialisierung von neuen Produkten im Produktlebenszyklus-Management-System

Ablauf:

1. Benutzer navigiert zu "Neues Produkt"
2. Dialog mit Feldern: Produktname, Beschreibung, Produktkategorie, Hersteller
3. Benutzer wählt Lebenszyklus-Template (Standard, Express, Custom)
4. Produktnummer wird automatisch generiert
5. Status wird auf "Konzept" gesetzt
6. Erste Version (v0.1) wird erstellt
7. Rollen zugewiesen: Produktmanager, Entwickler, Qualitätsprüfer
8. Produkt wird mit Kanban-Board und Status-Tracking verlinkt

Betroffene Felder: Product (Name, ProductNumber, Category, LifecycleTemplate), ProductVersion (Version, Status, CreatedByI3D, CreatedDate)

Auswirkungen: Produkt wird in Katalog sichtbar. Automatische Benachrichtigung an Produktteam. Entwicklungs-Roadmap wird aktualisiert.

3.6.2 Produktversionen und Revisionen managen

Zweck: Verwaltung von Produktversionen und deren Revisions-Historie

Ablauf:

1. Benutzer öffnet Produkt
2. Tab "Versionen" wird angewählt
3. System zeigt alle Versionen: v0.1, v0.2, v1.0, v1.1, etc.
4. Benutzer kann neue Version erstellen durch Klick "Neue Version"
5. Version erbt Eigenschaften von vorheriger Version
6. Benutzer kann Änderungen dokumentieren: Neue Features, Bug Fixes, Performance-Verbesserungen
7. Version wird dem Entwicklungsteam zugewiesen

8. Status kann sein: Draft → In Review → QA → Released → Deprecated

Betroffene Felder: ProductVersion (ProductI3D, Version, ChangeLog, Status, BasedOnVersionI3D, ReleaseDate)

Auswirkungen: Revisions-Kontrolle ermöglicht Rückwärts-Kompatibilität-Checks. Automatische Generierung von Release Notes. Support-Team kann schnell ältere Versionen identifizieren.

3.6.3 Konfigurationen und Varianten definieren

Zweck: Verwaltung von Produktkonfigurationen und Varianten

Ablauf:

1. Benutzer öffnet Produkt
2. Tab "Konfigurationen" wird angewählt
3. Benutzer kann Konfigurations-Optionen definieren (z.B. für Software: Farbe, Sprache, Lizenz-Typ)
4. Jede Option hat zulässige Werte: z.B. Farbe = [Rot, Blau, Grün]
5. Regelwerk für Abhängigkeiten kann definiert werden (z.B. "Farbe Rot nur wenn Premium-Lizenz")
6. Varianten-Kombinationen werden berechnet und angezeigt
7. Jede Variante bekommt eigene SKU
8. Kompatibilität mit anderen Produkten kann dokumentiert werden

Betroffene Felder: Product (ConfigurationType), ProductConfiguration (OptionName, AllowedValues), ProductVariant (SKU, ConfigurationI3D, ProductI3D)

Auswirkungen: Vertrieb kann schnell korrekte Produkt-Variante identifizieren. Lagerbestands-Management wird optimiert. Bill-of-Materials wird automatisch generiert.

3.6.4 Technische Dokumentation und Spezifikationen

Zweck: Verwaltung von Produktdokumentation, Spezifikationen und technischen Zeichnungen

Ablauf:

1. Benutzer öffnet Produkt
2. Tab "Dokumentation" wird angewählt
3. Benutzer kann Dateien hochladen: Datenblätter, Benutzerhandbücher, Schaltpläne, CAD-Zeichnungen
4. Jede Datei wird einer Version zugeordnet
5. Dateien können als "Entwurf", "Gültig", "Veraltet" markiert werden
6. Versionskontrolle für Dokumente: System verfolgt Änderungen
7. Dokumente können mit Revisionen gekennzeichnet werden (Rev. A, Rev. B, etc.)
8. Zugriffs-Rechte können restriktiv gesetzt werden (z.B. nur für Techniker)

Betroffene Felder: ProductDocumentation (FileType, VersionI3D, Status, AccessLevel, UploadedByI3D)

Auswirkungen: Zentralisierte technische Dokumentation. Automatische Verfügbarkeit im Kundenportal (wenn konfiguriert). Support-Team hat sofortigen Zugriff. Compliance-relevant für technische Richtlinien.

3.6.5 Qualitätskontrolle und Zertifizierungen

Zweck: Verwaltung von QA-Prozessen, Tests und Produktzertifizierungen

Ablauf:

1. Benutzer öffnet Produkt
2. Tab "Qualität & Zertifizierungen" wird angewählt
3. QA-Test-Plan wird definiert: Tests, Test-Kriterien, Verantwortlicher
4. Benutzer kann Test-Ergebnisse dokumentieren: Bestanden/Nicht bestanden

5. System warnt bei fehlgeschlagenen Tests
6. Zertifizierungen können dokumentiert werden (z.B. ISO 9001, CE-Kennzeichnung)
7. Zertifikat-Gültigkeitsdauer wird überwacht
8. Reports zeigen Test-Coverage und Qualitäts-Metriken

Betroffene Felder: ProductQATest (TestName, CriticalityLevel, Status, TestResultI3D), ProductCertification (CertificationType, ValidFrom, ValidTo, CertificationBodyI3D)

Auswirkungen: Qualitätssicherung ist dokumentiert. Zertifikats-Ablauf wird automatisch überwacht. Abweichungen triggern Eskalationen.

3.6.6 End-of-Life Planung und Ausmusterung

Zweck: Verwaltung von Produkteinstellungen und Migration auf Nachfolgeprodukte

Ablauf:

1. Benutzer öffnet älteres Produkt
2. Tab "End-of-Life" wird angewählt
3. Benutzer definiert: Produkteinstellung-Datum, Support-Enddatum
4. Nachfolgeprodukt wird optional zugeordnet
5. Migration-Plan für Kunden wird erstellt
6. System generiert automatisch: Kundenbenachrichtigungen, Support-Ankündigungen
7. Alte Produkte werden aus aktiven Listen entfernt (bleibt aber für Historisches)
8. Lagerbestands-Abverkauf wird geplant

Betroffene Felder: Product (Status="EndOfLife", EndOfLifeDate, SuccessorProductI3D, SupportEndDate), ProductEOLNotification

Auswirkungen: Kundenbeziehungen bleiben während Migration erhalten. Support-Kosten sinken. Compliance für Produkthaftung bleibt erhalten.

3.7 Stammblätter (Master Sheets)

Module Path: `src/centron/Centron.WPF.UI/Modules/Finances/MasterDataLists/OverView`
Controller: `MasterDataListOverviewAppModuleController`
ViewModel: `MasterDataListOverviewViewModel`
Category: Adressen/CRM
Description: Zentrale Verwaltung und Übersicht aller Stammdatenlisten (Branchen, Länder, Geschäftstypen, etc.)

Use Cases

3.7.1 Stammdatenlisten anzeigen und filtern

Zweck: Übersicht und Navigation über alle verfügbaren Stammdatenlisten

Ablauf:

1. Benutzer öffnet Modul "Stammblätter"
2. System zeigt Liste aller verfügbaren Stammdaten-Typen: Branchen, Länder, Regionen, Geschäftstypen, etc.
3. Jede Liste zeigt: Name, Anzahl der Einträge, Letzte Änderung, Status (Aktiv/Archiviert)
4. Benutzer kann nach Liste-Typ filtern (z.B. nur geografische Listen)
5. Suchleiste ermöglicht schnelle Suche nach spezifischer Liste
6. Doppelklick auf Liste öffnet diese zur Bearbeitung
7. Sortierung nach Name, Größe, Änderungsdatum möglich

Betroffene Felder: MasterDataListMetadata (ListName, EntryCount, LastModifiedDate, IsActive)

Auswirkungen: Zentralisierte Stammdaten-Verwaltung. Benutzer finden schnell benötigte Listen. Audit-Trail für Änderungen verfügbar.

3.7.2 Neue Stammdatenliste erstellen

Zweck: Erstellung von neuen Kategorien für Stammdaten-Management

Ablauf:

1. Benutzer klickt "Neue Liste"
2. Dialog mit Feldern: Listenname, Beschreibung, Datentyp (Text, Zahl, Datum, etc.)
3. Benutzer definiert Listenstruktur: Spalten und deren Datentypen
4. Optional: Standard-Einträge werden eingegeben
5. Benutzer kann Template-Option wählen (z.B. "Länder-Template" mit Pre-filled-Daten)
6. Zugriffs-Rechte werden definiert (Wer darf lesen/bearbeiten/löschen)
7. Liste wird erstellt und ist sofort verfügbar

Betroffene Felder: MasterDataList (Name, Description, DataType, StructureDefinition, IsActive, CreatedByI3D)

Auswirkungen: Neue Listen sind sofort in Drop-downs verfügbar. System kann automatisch auf neue Stammdaten hinweisen. Konfigurierbarkeit verbessert sich.

3.7.3 Einträge zu Stammdatenlisten hinzufügen/bearbeiten

Zweck: Verwaltung einzelner Einträge in Stammdatenlisten

Ablauf:

1. Benutzer öffnet Stammdatenliste (z.B. "Branchen")
2. System zeigt Gitterview mit allen Einträgen
3. Benutzer kann neuen Eintrag hinzufügen durch Klick "Neuer Eintrag"
4. Eingabe-Dialog mit Feldern der Liste wird geöffnet
5. Benutzer gibt Daten ein (z.B. Branchennummer, Branchenbeschreibung)
6. Validierung prüft auf Duplikate und Datentyp-Korrektheit
7. Benutzer kann Eintrag bearbeiten oder löschen (mit Bestätigung)
8. Änderungen werden sofort gespeichert und audited

Betroffene Felder: MasterDataEntry (MasterDataListI3D, Value, Description, IsActive, ChangedDate, ChangedByI3D)

Auswirkungen: Stammdaten bleiben aktuell. Änderungen wirken sich sofort auf alle Drop-downs aus. Audit-Trail dokumentiert alle Änderungen.

3.7.4 Stammdaten-Import aus Dateien

Zweck: Bulk-Import von Stammdaten aus Excel oder CSV-Dateien

Ablauf:

1. Benutzer klickt "Import" auf Stammdatenliste
2. Dialog zeigt Datei-Browser
3. Benutzer wählt Excel-Datei oder CSV-Datei
4. System versucht automatisch Spalten zuzuordnen (Spalte 1 → ListeName, etc.)
5. Benutzer kann Spalten-Zuordnung manuell korrigieren
6. Duplikat-Prüfung wird durchgeführt
7. Import-Vorschau zeigt, welche Zeilen importiert/aktualisiert werden
8. Nach Bestätigung: Daten werden importiert mit Audit-Trail

Betroffene Felder: MasterDataEntry (ImportedDate, ImportedByI3D), MasterDataImportLog (FileName, ImportDate, SuccessCount, ErrorCount)

Auswirkungen: Massenimport spart Zeit. Keine manuelle Dateneingabe nötig. Fehler werden dokumentiert für Nachverfolgung.

3.7.5 Stammdaten-Export und Reporting

Zweck: Export von Stammdaten für externe Nutzung oder Backup-Zwecke

Ablauf:

1. Benutzer öffnet Stammdatenliste
2. Benutzer klickt "Export"
3. Dialog mit Optionen: Format (Excel, CSV, JSON), Spalten-Auswahl, Filter
4. Benutzer kann nur aktive Einträge exportieren oder alle
5. Export-Format wird gewählt
6. System generiert Datei
7. Benutzer kann Report-Template für regelmäßige Exports konfigurieren
8. Datei wird heruntergeladen oder per Email versendet

Betroffene Felder: MasterDataExport (ExportedDate, ExportedByI3D, Format, FilterCriteria)

Auswirkungen: Stammdaten verfügbar für externe Systeme. Backups erstellt. Compliance-Reports können automatisch generiert werden.

3.7.6 Stammdaten-Versionierung und Audit-Trail

Zweck: Verfolgung von Änderungen an Stammdaten für Compliance und Debugging

Ablauf:

1. Benutzer öffnet Stammdatenliste
2. Button "Änderungshistorie" zeigt alle bisherigen Änderungen
3. System zeigt Timeline: Datum, Benutzer, Aktion (Hinzugefügt/Geändert/Gelöscht), Alte vs. Neue Werte
4. Benutzer kann alte Versionen anschauen im Detail-Dialog
5. Benutzer kann optional Änderung rückgängig machen ("Revert to Version X")
6. Rückgängig-Machung wird ebenfalls audited
7. Benutzer kann Report mit vollständiger Change-History exportieren

Betroffene Felder: MasterDataAuditLog (ListI3D, EntryI3D, Action, OldValue, NewValue, ChangedByI3D, ChangedDate), MasterDataVersion (VersionNumber, CreatedDate, CreatedByI3D)

Auswirkungen: Compliance-Reports zeigen wer wann was geändert hat. Debugging von Problemen wird erleichtert. Regulatory-Anforderungen (z.B. DSGVO) erfüllt. Revert-Möglichkeit verhindert Datenverlust.

4. Automatisierung (Automation)

4.1 Erwartete Events (Expected Events)

Module Path: `src/centron/Centron.WPF.UI/Modules/Helpdesk/ExpectedEvents`
Controller: `ExpectedEventsAppModuleController`
ViewModel: `ExpectedEventsViewModel`
Category: Automatisierung
Description: Konfiguration von erwarteten Ereignissen und automatischen Aktionen bei deren Eintritt

Use Cases

4.1.1 Erwartetes Event definieren

Zweck: Erstellung von Regeln für automatische Aktionen bei bestimmten Ereignissen
Ablauf: 1. Benutzer öffnet Expected Events 2. Wählt Ereignistyp (z.B. Ticket nicht bearbeitet) 3. Definiert Bedingungen 4. Wählt Aktion 5. Speichert und

aktiviert Event
Betroffene Felder: Type, Condition, Frequency, Action, IsActive
Auswirkungen: Automatisierte Prozesse. SLA-Einhaltung überwacht. Keine manuellen Maßnahmen erforderlich.

4.1.2 Event-Bedingungen mit Logik kombinieren

Zweck: Komplexe Bedingungen mit AND/OR Logik
Ablauf: 1. Bedingungseditor öffnen 2. Mehrere Bedingungen kombinieren 3. Logik-Operatoren setzen (AND/OR) 4. Testen 5. Speichern
Betroffene Felder: EventCondition, Property, Operator, Value, LogicOperator
Auswirkungen: Flexible Regeln. Komplexe Szenarien abbildbar. Proaktive Maßnahmen.

4.1.3 Automatische Aktionen konfigurieren

Zweck: Maßnahmen die beim Event-Eintritt ausgeführt werden
Ablauf: 1. Aktions-Typ wählen (Status, Benachrichtigung, Zuweisung) 2. Parameter eingeben 3. Mehrere Aktionen verknüpfen 4. Reihenfolge definieren 5. Aktivieren
Betroffene Felder: ActionType, Parameter, Sequence, IsEnabled
Auswirkungen: Einheitliche Behandlung. Keine zeitliche Verzögerung. Automatisierte Eskalation.

4.1.4 Event-Test durchführen und aktivieren

Zweck: Simulation und sichere Aktivierung von Events
Ablauf: 1. Test-Lauf starten 2. Betroffene Tickets prüfen 3. Auswirkungen anschauen 4. Bedingungen ggfs. anpassen 5. Event aktivieren
Betroffene Felder: TestResult, MatchingRecords, AffectedTickets, EstimatedActions
Auswirkungen: Fehler werden vor Aktivierung erkannt. Vorhersehbarer Effekt. Sicherer Rollout.

4.2 Erwartete Events Auswertung (Expected Events Evaluation)

Module Path: `src/centron/Centron.WPF.UI/Modules/Helpdesk/ExpectedEventsReporting`
Controller: `ExpectedEventsReportingAppModuleController`
ViewModel: `ExpectedEventsReportingViewModel`
Category: Automatisierung
Description: Auswertung und Analyse von erwarteten Ereignissen und deren automatischen Auslösungen

Use Cases

4.2.1 Event-Auslösungshistorie anzeigen

Zweck: Übersicht über alle durchgeführten automatischen Aktionen durch Events
Ablauf: 1. Report öffnen 2. Event auswählen oder alle anzeigen 3. Zeitraum definieren 4. Auslösungen werden tabellarisch angezeigt
Betroffene Felder: EventI3D, TriggerDate, AffectedTickets, ActionTaken, Status
Auswirkungen: Vollständige Nachverfolgbarkeit. Vertrauen in Automatisierung. Audit-Trail verfügbar.

4.2.2 Effektivität von Events messen

Zweck: Analyse wie effektiv Events ihre Ziele erreichen
Ablauf: 1. Metriken definieren (z.B. Tickets mit SLA-Verstoß vor/nach Event) 2. Zeitperioden vergleichen 3. Reports generieren
Betroffene Felder: EventI3D, MetricType, BeforeValue, AfterValue, EffectivenessPercentage
Auswirkungen: Optimierungspotenziale erkannt. ROI von Automatisierungen messbar. Entscheidungshilfe für Prozessverbesserung.

4.2.3 Event-Fehler und fehlgeschlagene Aktionen

Zweck: Erkennung und Behebung von Events die nicht wie erwartet funktionieren
Ablauf: 1. Fehlerreport aufrufen 2. Fehlgeschlagene Aktionen filtern 3. Details anschauen 4. Event ggfs. anpassen
Betroffene Felder: EventI3D, ErrorType, ErrorMessage, FailedTickets, Timestamp
Auswirkungen: Fehlerhafte Events werden schnell erkannt. Systemzuverlässigkeit erhöht. Kosten für manuelle Nacharbeit sinken.

4.2.4 Event-Performance und Auslastung

Zweck: Monitoring der Last und Performance der Event-Verarbeitung
Ablauf: 1. Dashboard öffnen 2. Metriken anschauen: Event-Läufe pro Stunde, Durchschnittliche Verarbeitungszeit 3. Engpässe erkennen
Betroffene Felder: EventExecutions, AverageProcessingTime, PeakTimes, SystemLoad
Auswirkungen: Performance-Bottlenecks identifiziert. Skalierungsbedarf erkannt. Systemstabilität gewährleistet.

4.3 Reportserver (Report Server)

Module Path: `src/centron/Centron.WPF.UI/Modules/Administration/ReportServer`
Controller: `ReportServerAppModuleController`
ViewModel: `ReportServerViewModel`
Category: Automatisierung
Description: Automatische Generierung und Versand von Reports an Kunden via E-Mail mit zeitlicher Planung

Use Cases

4.3.1 Report-Template definieren

Zweck: Erstellung von Report-Vorlagen mit Parametern und Formatierung
Ablauf: 1. Template öffnen 2. Report-Art wählen (Umsatz, OPOS, Ticket-Statistik) 3. Filter konfigurieren 4. Layout anpassen 5. Speichern
Betroffene Felder: TemplateI3D, ReportType, Parameters, Format, CreatedDate
Auswirkungen: Wiederverwendbare Vorlagen. Konsistente Reportstruktur. Zeit für Report-Erstellung spart.

4.3.2 Report-Versand zeitlich planen

Zweck: Automatische Generierung und Versand von Reports nach Zeitplan
Ablauf: 1. Report-Profil erstellen 2. Empfänger hinzufügen 3. Zeitplan definieren (täglich, wöchentlich) 4. Versandkanal wählen (E-Mail) 5. Aktivieren
Betroffene Felder: ScheduleType, Frequency, Recipients, DeliveryTime, IsActive
Auswirkungen: Vollautomatischer Reportversand. Keine manuellen Eingriffe. Kunden erhalten regelmäßige Updates.

4.3.3 Report-Filter und Dimensionen anpassen

Zweck: Kundenspezifische Filterung und Dimensionierung von Reports
Ablauf: 1. Report öffnen 2. Filter setzen (Kunde, Region, Produkt, Zeitraum) 3. Verdichtung wählen (täglich, wöchentlich, monatlich) 4. Spalten auswählen 5. Vorschau 6. Speichern
Betroffene Felder: FilterCriteria, Dimensions, Granularity, SelectedColumns
Auswirkungen: Personalisierte Reports pro Kunde. Relevante Daten prominent. Informationsüberfluss vermieden.

4.3.4 Report-Versand-Historie und Fehlerbehandlung

Zweck: Nachverfolgung aller versendeten Reports und Behandlung von Versandfehlern
Ablauf: 1. Versand-Historie öffnen 2. Filter anwenden (erfolgreich/fehlgeschlagen) 3. Details prüfen 4. Fehler analysieren 5. Manuellen Versand triggern wenn nötig
Betroffene Felder: ReportI3D, SentDate, Recipient, Status, ErrorMessage, RetryCount
Auswirkungen: Vollständige Nachverfolgbarkeit. Versandfehler werden erkannt. Compliance-Audit möglich.

5. Buchhaltung/Finanzen (Accounting/Finances)

5.1 Buchhaltungsexport/-import (Accounting Export/Import)

Modulpfad: `src/centron/Centron.WPF.UI/Modules/Finances/AccountingExport/`
Controller: `AccountingExportAppModuleController`
ViewModel: `AccountingExportViewModel`
Schnittstelle: `IAccountingExportLogic`

Kategorie: Buchhaltung/Finanzen
Beschreibung: Export von Finanztransaktionen in verschiedene Buchhaltungssysteme und Import von Kontoauszügen
Lizenz: `LicenseGuids.Centron`
Rechte: `UserRightsConst.Finances.ACCOUNTING_EXPORT`

Modul-Architektur

Das Buchhaltungsexport-Modul verbindet c-entron mit verschiedenen **Buchhaltungssystemen** (DATEV, Abacus, SAP, Sage, Lexware, Navision, etc.) über standardisierte Schnittstellendefinitionen und Format-Konvertierung:

- **Bidirektional**: Export von Rechnungen/Zahlungen + Import von Kontoauszügen
- **Multi-Format-Support**: CSV, XML, OFX, SEPA-XML
- **Fehlerverarbeitung**: Validierung, Konflikt-Erkennung, Retry-Logik

Vollständige Use Cases

5.1.1 Rechnungen und Gutschriften ins Buchhaltungssystem exportieren

Zweck: Alle erstellten Belege automatisch zur Buchung in Fibukas exportieren
Ablauf:

1. Benutzer wählt Datumsbereich (z.B. letzte Woche)
2. System ruft alle Rechnungen/Gutschriften ab
3. Für jede Rechnung wird generiert: Belegnummer, Datum, Betrag, Steuersatz, Kunde, Kontierung
4. Export-Format wählen (DATEV, Abacus, etc.)
5. Button "Exportieren" → Datei zum Download oder direkter Upload
   Felder: ReceiptNumber, ReceiptDate, NetAmount, TaxAmount, CustomerName, GLAccount
   Auswirkung: Datei wird auf Nachfragesystem hochgeladen/bereitgestellt

5.1.2 Zahlungseingänge in Buchhaltung verbuchen

Zweck: Eingegangene Zahlungen automatisch als Kontobuchungen exportieren
Prozess:

1. Zahlungseingang erfasst (Bank-Import oder manuell)
2. System matched Zahlung zu Rechnung
3. Buchungszeile generiert: Datum, Betrag, Konten (Debitor ↔ Bank), Referenz (Rechnungsnummer)
4. Export zu Buchhaltung als Zahlungsdiarium
   Daten: PaymentDate, PaymentAmount, InvoiceReference, CustomerAccount, BankAccount

5.1.3 Kontoauszüge importieren und abstimmen

Zweck: Bankauszüge automatisch einlesen und gegen offene Posten abgleichen
Ablauf:

1. Kontoauszug-Datei hochladen (CSV/OFX/MT940)
2. System parst Datei: Datum, Betrag, Referenz, Konterpartei
3. Automatischer Abgleich gegen offene Rechnungen
4. Manuelle Zuordnung für nicht gematchte Buchungen
5. Speichern → Zahlungseingang wird generiert (wenn noch nicht vorhanden)
   Import-Format: Bank-Auszug mit Kontonummer, Datum, Betrag, Referenztext

5.1.4 Lagerbestandsveränderungen exportieren

Zweck: Lagermutationen (Einkauf, Verkauf, Umbuchung) in Buchhaltung buchen
Export-Daten:

- Artikel-Zugänge (Einkauf): Datum, Menge, Lagerkonto, Wert
- Artikel-Abgänge (Verkauf): Datum, Menge, Lagerkonto, FIFO/LIFO-Wert
- Bestandskorrektionen: Inventur-Differenzen
  Auswirkung: Automatische Lagerkonten-Buchungen in Fibuka

5.1.5 Provisionsabrechnungen und Kostenverteilungen exportieren

Zweck: Vertriebsprovisionen und interne Kostenverteilungen buchen
Export enthält:

- Verkäufer-Provisionen: Name, Provision (EUR), Provision %-Satz
- Kostenträger-Verteilung: Projekt-ID, Kostenstelle, Betrag
- Zeitaufschläge: Stunden × Stundensatz
  Zielkonto: Provisionskonten, Kostenträgerkonten

5.1.6 Debitorenrechnungen im Debitorenstamm registrieren

Zweck: Neue Debitorenkonten automatisch in Buchhaltung anlegen
Daten:

- Neue Kunden aus c-entron
- Kundennummer, Name, Adresse, Steuernummer
- Standard-Zahlungsbedingungen
- Kreditlimit (wenn konfiguriert)
  Auswirkung: Debitor-Stamm wird in DATEV/Abacus aktualisiert

5.1.7 Fehlerhafte Exporte korrigieren und neu exportieren

Zweck: Validierungsfehler beheben und Neuversand durchführen
Fehlertypen:

- Fehlende Kontierungen
- Ungültige Kontonummern
- Beträge mit zu vielen Dezimalstellen
- Ungültige Zeichenzeichen
  Prozess: Fehler anzeigen → Korrigieren → "Erneut exportieren"

5.1.8 Export-Status und Audit-Trail prüfen

Zweck: Nachverfolgung welche Belege bereits exportiert wurden
Anzeige:

- Export-Datum und -Zeit
- Export-Format
- Empfänger-System (DATEV, Abacus, etc.)
- Status: "Erfolgreich", "Mit Warnung", "Fehler"
- Fehlermeldungen (wenn vorhanden)
  Funktion: "Export-Historie anzeigen" → Detaillierte Logs

5.1.9 Batch-Export mit Zeitplan automatisieren

Zweck: Tägliche/wöchentliche Exporte automatisch durchführen
Konfiguration:

- Zeitplan: Täglich 18:00 Uhr
- Automatischer Filter: Nur Belege der letzten 24 Stunden
- Ziel-Format und -Pfad (FTP, lokal, Cloud)
- Benachrichtigung nach Export
  Auswirkung: Scheduling-Task wird registriert

5.1.10 Import-Validierung und Konflikt-Auflösung

Zweck: Importierte Daten prüfen und Duplikate/Konflikte handhaben
Validierungen:

- Duplikat-Check: Existiert Beleg bereits?
- Betragsabweichung: Import-Betrag ≠ c-entron Betrag?
- Währungs-Konvertierung (wenn unterschiedliche Währung)

- Datum-Validierung (nicht in der Zukunft, nicht älter als 2 Jahre)
  Konflikt-Auflösung: Benutzer wählt: "Überschreiben", "Behalten", "Überspringen"

5.2 DATEV Belegtransfer (DATEV Document Transfer)

Modulpfad: `src/centron/Centron.WPF.UI/Modules/Finances/DatevTransfer/`
Controller: `DatevTransferAppModuleController`
ViewModel: `DatevTransferViewModel`
Schnittstelle: `IDatevTransferLogic`
Kategorie: Buchhaltung/Finanzen
Beschreibung: Direkte Anbindung zu DATEV für Belegverwaltung und digitale Archivierung
Lizenz: `LicenseGuids.Centron`
Rechte: `UserRightsConst.Finances.DATEV_TRANSFER`

Modul-Architektur

Das DATEV Belegtransfer-Modul ist eine spezialisierte **Schnittstelle zur DATEV-Plattform** für den Austausch von Originalbelegen, Scans und Dokumente:

- **Upload**: Rechnungs-PDFs, Bestellungen, Lieferscheine zu DATEV
- **Download**: DATEV Rechnungen und Belege importieren
- **Mapping**: Automatische Zuordnung zu c-entron Belegen
- **Archivierung**: Digitale Belegverwaltung mit Volltext-Suche

Vollständige Use Cases

5.2.1 Rechnung als PDF zu DATEV hochladen

Zweck: Erstellte Rechnungen digital an DATEV Aufbewahrungssystem übertragen
Prozess:

1. Rechnung in c-entron erstellen
2. Button "An DATEV übertragen"
3. Rechnung wird als PDF exportiert
4. DATEV Connect-Upload-Verbindung nutzen
5. DATEV empfängt Beleg + Metadata (Belegnummer, Datum, Betrag, Lieferant)
   Daten: InvoiceNumber, OriginalFileName, DocumentHash, MetadataXML
   Auswirkung: Beleg in DATEV archiviert

5.2.2 Eingangsrechnungen von DATEV importieren

Zweck: Rechnungen die DATEV von Lieferanten erhalten hat in c-entron übernehmen
Ablauf:

1. c-entron lädt Eingangsrechnungs-Liste von DATEV
2. System zeigt neue Rechnungen (noch nicht in c-entron)
3. Benutzer wählt zu importierende Rechnungen
4. Automatische Zuordnung zu c-entron Lieferant (via Steuernummer/Name)
5. Buchungszeile wird vorausgefüllt
6. Benutzer bestätigt → Eingangsrechnung wird angelegt

5.2.3 Belegscans und Anhänge speichern

Zweck: Gescannte Originalbelege zu c-entron Dokumenten hinzufügen
Prozess: Scanner-Software lädt PDF zu DATEV → c-entron matched Scan zu Beleg via OCR → Scan-PDF wird zu Attachments hinzugefügt

5.2.4 Belegverwaltung und Langzeitarchivierung (10 Jahre)

Zweck: Belege revisionssicher für 10 Jahre archivieren
Features: Automatische Archivierung, unveränderliches Archiv, Zugriff nur autorisiert, Compliance-Audit-Trail

5.2.5 Volltext-Suche in archivierten Belegen

Zweck: Schnelle Suche nach archivierten Rechnungen/Belegen
Suchfelder: Belegnummer, Lieferant/Kunde, Betreffzeile, OCR-Text, Zeitraum, Bereich

5.2.6 Belegverwerfung und Wiederherstellung

Zweck: Fehlerhafte Belege löschen/restaurieren
Prozess: Beleg verwerfen → logische Löschung (30 Tage Recovery möglich) → physisches Löschen nach Aufbewahrungsfrist

5.2.7 Belegfreigabe und Genehmigungsprozesse

Zweck: Belege nur nach Freigabe in Buchhaltung verbuchen
Workflow: Hochgeladen → Prüfung → Freigabe oder Ablehnung → Verbuchen

5.2.8 Belegsperrung und Compliance-Hold

Zweck: Belege für Audit/Rechtsstreit sperren und unveränderbar machen

5.2.9 Batch-Belegverarbeitung

Zweck: Mehrere Belege gleichzeitig hochladen/archivieren/genehmigen

5.2.10 GDPdU Compliance und Audit-Trail

Zweck: Compliance mit deutschen Archivierungs-Richtlinien (10 Jahre, Unveränderbarkeit, Zugriffskontrolle)

5.3 Kalkulation pro Filiale (Branch-based Calculation)

Modulpfad: `src/centron/Centron.WPF.UI/Modules/Finances/BranchCalculation/`
Controller: `BranchCalculationAppModuleController`
ViewModel: `BranchCalculationViewModel`
Schnittstelle: `IBranchCalculationLogic`
Kategorie: Buchhaltung/Finanzen
Beschreibung: Getrennte Kostenrechnung und Rentabilitätsanalyse nach Filialen
Lizenz: `LicenseGuids.Centron`
Rechte: `UserRightsConst.Finances.BRANCH_CALCULATION`

Modul-Architektur

Das Kalkulations-Modul ermöglicht **dezentralisierte Finanzberichte** pro Filiale/Standort mit Kosten-Tracking und Profitability-Analysen:

- **Kosten-Zuordnung**: Direkte vs. Gemeinkostenverteilung
- **Ergebnis-Rechnung**: Umsatz - Kosten = Ergebnis pro Filiale
- **Vergleich**: Soll vs. Ist, Trends, Benchmarking

Vollständige Use Cases

5.3.1 Filial-Kostenrechnung erstellen und anzeigen

Zweck: Finanzielle Performance jeder Filiale einzeln nachvollziehen
Ablauf: Benutzer wählt Filiale → Zeitraum → System berechnet: Umsatz, Kosten, EBIT → Ergebnis

5.3.2 Gemeinkostenverteilung nach Schlüssel

Zweck: Zentrale Kosten auf Filialen fair verteilen
Verfahren: Nach Headcount, Umsatz, Fläche, oder benutzerdefinierten Schlüsseln

5.3.3 Deckungsbeitrag pro Filiale berechnen

Zweck: Welche Filiale trägt am meisten zu Unternehmensgewinn bei?
Formel: Umsatz - Variable Kosten = Deckungsbeitrag

5.3.4 Vergleichsanalyse zwischen Filialen

Zweck: Welche Filiale läuft gut, welche schlecht?
Vergleich: Kennzahlen wie Rentabilität %, Break-Even-Point, Kostenquote

5.3.5 Budgets pro Filiale definieren und kontrollieren

Zweck: Sollen-Ist-Vergleiche durchführen
Prozess: Budget setzen → laufend gegen Ist vergleichen → Abweichung Analyse

5.3.6 Investitionsrentabilität pro Filiale

Zweck: Wie schnell zahlt sich eine Investition in Filiale X aus?
Berechnung: ROI = (Gewinn / Investition) × 100

5.3.7 Filialen-Ranking und Performance-Dashboard

Zweck: Schneller Überblick welche Filiale Top/Bottom Performer ist
Anzeige: Ranking nach Umsatz, Rentabilität, Wachstum

5.3.8 Kostenmodelle für Szenarien-Analysen

Zweck: "Was passiert wenn..." - Szenarien durchspielen
Beispiel: "Wenn Filiale X Miete um 10% reduziert → Rentabilität +XY%"

5.4 Mahnung (Dunning/Collection)

Modulpfad: `src/centron/Centron.WPF.UI/Modules/Finances/Dunning/`
Controller: `DunningOverviewAppModuleController`
ViewModel: `DunningOverviewViewModel`
Schnittstelle: `IDunningLogic`
Kategorie: Buchhaltung/Finanzen
Beschreibung: Automatisierte Forderungsverwaltung mit Mahnstufen und Inkasso-Eskalation
Lizenz: `LicenseGuids.Dunning` OR `LicenseGuids.Centron`
Rechte: `UserRightsConst.Finances.DUNNING`

Modul-Architektur

Das Mahnungsmodul automatisiert den kompletten **Inkasso-Prozess** mit gestaffelten Mahnstufen:

- **Stufe 1 (Zahlungserinnerung)**: Nach 14 Tagen Überfälligkeit, automatische Benachrichtigung
- **Stufe 2 (Erste Mahnung)**: Nach 21 Tagen, mit Mahngebühren
- **Stufe 3 (Zweite Mahnung)**: Nach 28 Tagen, erhöhte Gebühren
- **Stufe 4 (Letzte Mahnung)**: Nach 35 Tagen, Verweis auf Inkasso
- **Inkasso-Eskalation**: Übergabe an externe Inkasso-Firma oder juristische Schritte

Vollständige Use Cases

5.4.1 Überfällige Rechnungen identifizieren und Mahnprozess starten

Zweck: Automatisch alle überfälligen Rechnungen erfassen und Mahnlauf initiieren
Ablauf aus Benutzersicht:

1. Benutzer öffnet Mahnungsmodul
2. System ruft alle offenen Rechnungen mit Zahlungsziel <heute ab (Standard 14 Tage)
3. Benutzer kann Filter setzen: Kunde, Filiale, Mindestbetrag
4. "Mahnlauf starten" Button wird geklickt
5. System generiert Mahnauszüge pro Mahnstufe
   Betroffene Daten:

- Rechnungsdatum (`ReceiptDate`)
- Zahlungsziel (`PaymentTerms` in Tagen)
- Zahlungseingänge (`PaymentRecords`)
- Aktuelle Mahnstufe (`DunningLevel`)
- Letzte Mahnverwarnzeit (`LastDunningDate`)

5.4.2 Mahnstufen-Regeln konfigurieren

Zweck: Definieren wann welche Mahnstufe automatisch ausgelöst wird
Konfigurierbare Felder:

- **Tage bis Stufe 1**: Nach X Tagen Überfälligkeit (Standard: 14 Tage)
- **Tage bis Stufe 2**: (Standard: 21 Tage)
- **Tage bis Stufe 3**: (Standard: 28 Tage)
- **Tage bis Stufe 4**: (Standard: 35 Tage)
- **Tage bis Inkasso-Übergabe**: (Standard: 42 Tage)
- **Mahngebühren pro Stufe**: EUR pro Mahnung oder Prozentsatz
- **Zinsen/Verzugszinsen**: Automatisch berechnet nach BGB (gesetzlich 5%)
- **Exemption für kleine Beträge**: Rechnungen unter EUR 50 nicht mahnen

5.4.3 Personalisierte Mahnbriefe generieren

Zweck: Mahnschreiben basierend auf Vorlagen automatisch erstellen und versenden
Felder pro Mahnbrief:

- **Empfänger-Daten**: Kundennamen, Adresse, Ansprechpartner
- **Rechnungs-Details**: Rechnungsnummer, Betrag, Fälligkeitsdatum
- **Mahnstufen-Info**: "Dies ist die 1. Mahnung" mit Rechtshinweis
- **Forderungssumme**: Ursprüngliche Rechnung + Mahngebühren + Zinsen
- **Zahlungsaufforderung**: Zahlungskonditionen (5 Tage Zahlungsziel ab Mahnbrief)
- **Zahlungsverbindung**: Bankverbindung, IBAN, BIC
- **Optionaler Text**: Kundenspezifische Noten im Brief
- **Unterschrift-Feld**: Personalname oder Firmenname

5.4.4 Zahlungserinnnerungen per E-Mail versenden

Zweck: Elektronische Benachrichtigungen statt Briefe verschicken
Prozess:

1. Automatische E-Mail-Template-Auswahl pro Mahnstufe
2. Platzhalter ersetzen: {{KundenName}}, {{Rechnungsnummer}}, {{Betrag}}, {{Fälligkeitsdatum}}
3. Optionaler PDF-Anhang (Mahnbrief)
4. E-Mail-Tracking aktivieren (Öffnen, Clicks verfolgen)
5. Versand protokolliert in `DunningEmailLog` Tabelle
   E-Mail-Zeilen:

- An: Kundenemails (alle E-Mails des Kontakts)
- CC: Accounting-Team (wenn konfiguriert)

- BCC: Audit-Log
- Subject: "Zahlungserinnerung zu Rechnung XYZ"

5.4.5 Mahnungen manuell anpassen und neu versenden

Zweck: Benutzer kann fehlerhafte Mahnungen korrigieren und erneut versenden
Mögliche Anpassungen:

- Zahlungsbetrag korrigieren (wenn Teilzahlung erfolgt)
- Mahnstufe herabstufen (bei Kulanzregelung)
- Verzugszinsen neu berechnen
- Empfänger-Adresse ändern
- Text-Ergänzung hinzufügen (z.B. "Zahlungsaufschub bis...")
- Versand-Kanal wechseln (Brief ↔ E-Mail ↔ Fax)
- Erneute Versendung mit neuem Datum
  Button: "Mahnung erneut versenden" oder "Mahnbrief aktualisieren"

5.4.6 Zahlungseingänge matching und Mahnprozess stoppen

Zweck: Wenn Zahlung eingeht, Mahnprozess automatisch beenden
Ablauf:

1. Zahlungseingang erfasst (automatisch via Bank-Import oder manuell)
2. System matched Zahlung zu offener Rechnung
3. Wenn Mahnung noch aktiv: Mahnprozess stoppen
4. Bestätigung-E-Mail an Debitor (optional)
5. Mahnhistorie bleibt erhalten für Audit-Trail
   Auswirkung auf Felder:

- `IsResolved` : true
- `ResolutionDate` : Heute
- `DunningLevel` : 0 (Zurücksetzen)

5.4.7 Zinsberechnung und Verzugszinsen

Zweck: Automatisch Verzugszinsen berechnen gemäß Gesetz
Konfiguration:

- **Zinssatz**: Standard 5% + Leitzins (§ 288 BGB)
- **Berechnung ab**: 1. Tag nach Zahlungsziel
- **Rounding**: Auf volle EUR/CT runden
- **Max Zinsen**: Optional Höchstlimit definieren
  Beispiel:
- Rechnung EUR 1.000 fällig 2025-11-15
- Heute 2025-11-30 (15 Tage überfällig)
- Zinsen = 1.000 × 5% × (15/360) = EUR 2,08

5.4.8 Inkasso-Übergabe vorbereiten und exportieren

Zweck: Mahnungen die nicht bezahlt werden an Inkasso-Firma übergeben
Prozess:

1. System identifiziert Mahnungen auf Stufe 4 älter als X Tage
2. Exportformat für externe Inkasso wählen (CSV, XLS, PDF)
3. Daten exportieren:
   - Debitor-Informationen
   - Rechnungs-Details
   - Gesamtforderung (Rechnung + Gebühren + Zinsen)
   - Mahnhistorie
4. Optionale Nachricht an Debitor "Wir übergeben Ihren Fall an Inkasso"
5. Datei an Inkasso-Firma versenden
   Export-Felder: Name, Adresse, Debitor-ID, Rechnungsnummer, Betrag, Zahlungsziel

5.4.9 Mahnhistorie und Audit-Trail anzeigen

Zweck: Dokumentation aller Mahnaktionen für Rechtssicherheit
Anzeige:

- Versand-Datum jeder Mahnung
- Versand-Kanal (Brief, E-Mail, Fax)
- Empfänger-Adresse
- Mahntext (archiviert)
- Gebühren pro Mahnung
- Zahlungen, die eingegangen sind
- Hinweise/Notizen (z.B. "Debitor angerufen - zahlt bis...")
- Benutzer, der Mahnung versendet hat
  Funktion: "Mahnverlauf anzeigen" → Popup mit vollständiger Chronologie

5.4.10 Mahnungen stornieren oder Prozess abbrechen

Zweck: Mahnungen zurückziehen, wenn Kunde zahlt oder Kulanzregelung
Möglichkeiten:

- **Einzelne Mahnung stornieren**: Diese Mahnung rückgängig machen, Gebühren erstatten
- **Kompletten Mahnprozess beenden**: Alle ausstehenden Mahnungen rückgängig machen
- **Culpa-Erlass**: Grund dokumentieren (z.B. "Kundenbeschwerde zu Recht")
- **Gutschrift ausstellen**: Mahngebühren dem Debitor gutschreiben
- **Notiz hinzufügen**: "Einspruch berechtigt - Fehler in Rechnung"
  Folge: Mahnhistorie bleibt sichtbar, aber Status auf "Abgebrochen" gesetzt

5.5 OPOS (Outstanding Items / Offene Posten)

Modulpfad: `src/centron/Centron.WPF.UI/Modules/Finances/Opos/`
Controller: `OposOverviewAppModuleController`
ViewModel: `OposOverviewViewModel`
Schnittstelle: `IOposLogic`
Kategorie: Buchhaltung/Finanzen
Beschreibung: Verwaltung aller offenen Forderungen und Verbindlichkeiten mit automatischem Bank-Import
Lizenz: `LicenseGuids.Centron`
Rechte: `UserRightsConst.Finances.OPOS`

Modul-Architektur

OPOS zeigt die **Debitorische/Kreditorische Schuldenliste** mit automatischem Import von Bank-/DATEV-Daten und Zahlungs-Matching:

- **Debitorische OPOS**: Offene Rechnungen der Kunden
- **Kreditorische OPOS**: Offene Rechnungen an Lieferanten
- **Auto-Import**: DATEV-Schnittstelle, Bank-Export
- **Matching**: Zahlungen automatisch zuordnen

Vollständige Use Cases

5.5.1 Offene Posten anzeigen und filtern

Zweck: Schnell sehen welche Rechnungen noch offen sind
Filter: Nach Kunde, Lieferant, Zeitraum, Überblättigkeit (>30/60/90 Tage), Betrag
Anzeige: Rechnungsnummer, Datum, Fälligkeitsdatum, Betrag, Tage überfällig

5.5.2 Zahlungseingänge matching zu offenen Posten

Zweck: Eingehende Zahlung automatisch zuordnen welche Rechnung ausgeglichen wird
Prozess: Zahlung erfasst → System sucht offene Rechnung mit passendem Betrag → Match → Ausgeglichen

5.5.3 Bank-Kontoauszüge als OPOS importieren

Zweck: Kontoauszüge automatisch einlesen und als offene Items registrieren
Format: CSV, OFX, MT940, SEPA-XML
Ablauf: Datei hochladen → System parst → Neue OPOS Einträge erstellen

5.5.4 Teilzahlungen und Storno handhaben

Zweck: Wenn Kunde weniger zahlt als Rechnung (Teilzahlung) oder Rechnung storniert
Logik: Teild zahlung aktualisiert Restbetrag → Neue OPOS mit Differenzbetrag
Storno: Rechnung wird auf Betrag 0 gesetzt

5.5.5 OPOS-Report für Buchhaltung/Management

Zweck: Detaillierten Bericht alle offenen Posten exportieren
Format: Excel, PDF
Inhalt: Kunde/Lieferant, Rechnungsnummer, Betrag, Überblättigkeit, Mahnstatus

5.5.6 OPOS-Abstimmung und Abweichungs-Analyse

Zweck: OPOS Bestand prüfen (Soll-Ist-Abgleich mit Buchhaltung)
Prozess: c-entron OPOS vs. DATEV/Abacus OPOS → Differenzen anzeigen → Ursachen finden

5.5.7 Automatische OPOS-Bereinigung (aged debt)

Zweck: Sehr alte Posten (>2 Jahre) als "Delkonti" abschreiben
Prozess: Automatisches Flag "Delkonti-reif" → Abschreibungs-Vorschlag → Buchung

5.5.8 Export für Buchhaltung und Schnittstellen

Zweck: OPOS-Bestand in andere Systeme exportieren
Ziele: DATEV, Abacus, SAP, Lexware
Format: CSV, XML, OFX

5.6 SEPA (SEPA Payments / SEPA-Lastschriften)

Modulpfad: `src/centron/Centron.WPF.UI/Modules/Finances/Sepa/`
Controller: `SepaAppModuleController`
ViewModel: `SepaViewModel`
Schnittstelle: `ISepaLogic`
Kategorie: Buchhaltung/Finanzen
Beschreibung: Management von SEPA-Lastschriften (SDD - SEPA Direct Debit) und Zahlungsaufträgen
Lizenz: `LicenseGuids.Centron`
Rechte: `UserRightsConst.Finances.SEPA`

Modul-Architektur

SEPA-Modul integriert europäischen Zahlungsverkehr (Single Euro Payments Area) für automatisierte Lastschrift-Verfahren:

- **SEPA-Mandate**: Ermächtigungen von Kunden zum Abbuchen ihrer Konten
- **SEPA Direct Debit**: Automatische Abbuchung von Kundenkonten
- **SEPA Credit Transfer**: Überweisung an Lieferanten
- **Compliance**: Einhaltung SEPA-Regelwerk und Datenschutz

Vollständige Use Cases

5.6.1 SEPA-Mandate von Kunden verwalten

Zweck: Nachweise über Kundengenehmigung zum automatischen Abbuchen
Daten: Kundennummer, Bankverbindung, IBAN, BIC, Mandate-Datum, Signatur

Ablauf: Kunde unterzeichnet Mandat → Scannen → In c-entron erfassen → Archivieren

5.6.2 SEPA-Lastschrift-Datei (XML) generieren

Zweck: Alle fälligen Rechnungen in einer Lastschrift-Datei zusammenfassen und zur Bank schicken
Format: pain.008.003.02 (SEPA XML Standard)
Ablauf: Offene Posten mit Mandate → XML generieren → Bank hochladen

5.6.3 Bank-Rückmeldungen zu Lastschriften verarbeiten

Zweck: Mit Fehler bei Lastschriften (IBAN ungültig, Konto gekündigt, etc.) umgehen
Fehler-Typen: "IBAN Format wrong", "Account closed", "Overdraft protection"
Aktion: Kunden kontaktieren, Bankverbindung korrigieren, Wiederholung

5.6.4 SEPA Credit Transfer zu Lieferanten-Zahlungen

Zweck: Zahlungen an Lieferanten über SEPA-Überweisung durchführen
Prozess: Lieferanten-Rechnungen fällig → Zahlung genehmigen → SEPA-XML → Bank hochladen

5.6.5 SEPA-Gebühren und Kosten verwalten

Zweck: Tracking der Bank-Gebühren für SEPA-Transaktionen
Gebühren: Pro Lastschrift, Rückbuchung, Mandat-Verwaltung
Buchung: Gebühren zu OPOS oder Konto-Abstimmung hinzufügen

5.6.6 SEPA-Status und Audit-Trail

Zweck: Nachverfolgung welche Lastschriften erfolgreich verarbeitet wurden
Anzeige: Datei-Name, Generierungs-Datum, Anzahl Transaktionen, Gesamtbetrag, Status

5.6.7 SEPA-Validierung und Fehlerprüfung

Zweck: Vor Versand an Bank prüfen ob Datei valide ist
Checks: IBAN-Format, Mandate vorhanden, Betrag positiv, Vor-Fälligkeit korrekt

5.7 Zahlungseingang (Payment Receipt / Payments)

Modulpfad: `src/centron/Centron.WPF.UI/Modules/Finances/Payments/`
Controller: `PaymentsAppModuleController`
ViewModel: `PaymentsViewModel`
Schnittstelle: `IPaymentLogic`
Kategorie: Buchhaltung/Finanzen
Beschreibung: Verwaltung von Zahlungseingängen, Bank-Import und automatisches Matching
Lizenz: `LicenseGuids.Centron`
Rechte: `UserRightsConst.Finances.PAYMENTS`

Modul-Architektur

Zahlungseingangs-Modul ist das **Kernstück der Liquiditätsverwaltung** mit automatischem Bank-Import, Matching-Algorithmen und Compliance:

- **Bank-Import**: CSV, MT940, SEPA-XML, OFX, Fintech-APIs (FinAPI)
- **Matching**: Automatische Zuordnung zu offenen Rechnungen
- **Rekonciliation**: Bank-Abstimmung und Korrektur
- **Compliance**: Audit-Trail, Doppelbuchungs-Prävention

Vollständige Use Cases

5.7.1 Bankauszüge hochladen und importieren

Zweck: Täglich neue Zahlungen von Bank abholen und verarbeiten
Format: CSV, MT940, SEPA-XML, OFX, oder direkte FinAPI-Anbindung
Ablauf: Datei Upload → Validierung → Transactions parsen → Matching starten

5.7.2 Automatisches Matching zu Rechnungen

Zweck: System findet automatisch welche Zahlung zu welcher Rechnung passt
Algorithmus: Betrag-Vergleich, Referenztexterkennung (OCR), Fuzzy-Matching, Zeitraum-Prüfung
Genauigkeit: Über 90% automatische Matches, Rest manuell lösen

5.7.3 Manuelle Zahlungs-Zuordnung bei Nicht-Matches

Zweck: Zahlungen die automatisches Matching nicht schaffte manuell zuordnen
Prozess: Zahlung + offene Rechnungen zeigen → Benutzer klickt Zuordnung → Speichern

5.7.4 Teilzahlungen und Zahlungsrückgaben handhaben

Zweck: Wenn Zahlung < Rechnungsbetrag oder Zahlung rückgängig gemacht wird
Teilzahlung: Neue OPOS mit Restbetrag erstellen
Rückgabe: Transaktion mit Referenz stornieren

5.7.5 Bank-Abstimmung und Diskrepanz-Analyse

Zweck: Monatlich prüfen ob Bank-Bestand in c-entron = echtes Bank-Konto
Prozess: Bank-Bestand vs. c-entron Bestand → Differenzen finden → Ursachen → Korrektionen

5.7.6 Zahlungs-Status und Reporting

Zweck: Überblick wie viel wurde eingezahlt, ausstehend, überfällig
Reports: Tägliche/wöchentliche Zahlungseingänge, Ausstände nach Kunde, Trend

5.7.7 FinAPI-Integration für Automatisierung

Zweck: Zahlungen 100% automatisch von Bank über FinAPI abholen
Prozess: FinAPI-Credentials konfigurieren → täglich auto-sync → Zahlungen importiert

5.7.8 Zahlungseingang verbuchen in Buchhaltung

Zweck: Matched Zahlungen als Buchungszeile exportieren
Konten: Debitor-Konto ↔ Bank-Konto
Export: Zu DATEV, Abacus, etc.

6. Controlling/Analytics (Controlling/Analytics)

6.1 Analytics

Module Path: `src/centron/Centron.WPF.UI/Modules/Statistics/SaleStatistics`
Controller: `SaleStatisticsAppModuleController`
ViewModel: `SaleStatisticsViewModel`
Category: Controlling/Analytics
Description: Erstellung, Bearbeitung und Export verschiedenster Auswertungen (Verkauf, Einkauf, Mitarbeiter, Tickets)
License: LicenseGuids.Centron oder LicenseGuids.Analytics

Use Cases

6.1.1 Verkaufsstatistik nach Artikel erstellen

Zweck: Verkaufsleistung einzelner Artikel über einen Zeitraum analysieren

Ablauf:

1. Benutzer öffnet Analytics-Modul
2. Wählt "Verkaufsstatistik" als Auswertungstyp
3. Definiert Zeitraum (von/bis) und Artikel-Filter
4. System lädt Verkaufsdaten aus Sales-Tabelle
5. Aggregiert nach Artikel: Menge, Umsatz, Gewinn
6. Zeigt Top-Artikel nach Umsatz/Menge in Tabelle und Chart
7. Benutzer kann Auswertung exportieren (Excel, PDF)

Betroffene Felder: Articles (Name, ArticleNumber), Sales (Quantity, Amount, Cost, Date), Customer, SalesOrder

Auswirkungen:

- Bestandsplanung basierend auf Verkaufsvolumen
- Lagerverwaltung anpassen
- Marketingbudget auf Top-Seller konzentrieren

6.1.2 Kundenstatistik mit Jahresvergleich

Zweck: Kundenumsatz entwicklung und Trend-Analyse über mehrere Jahre

Ablauf:

1. Benutzer wählt "Kundenstatistik" aus
2. Definiert Vergleichsjahre und Kundensegment
3. System aggregiert Verkaufsdaten pro Kunde pro Jahr
4. Zeigt Spalten: Kundenname, Jahr 1, Jahr 2, Jahr 3, Trend, Delta %
5. Identifiziert steigende/fallende Trends
6. Erlaubt Drill-down zu Einzelverkäufen
7. Export mit Trend-Analyse und Warnsignale (rückläufig >20%)

Betroffene Felder: Customer (Name, CustomerNumber, Segment), Sales (Amount, Date, Quantity)

Auswirkungen:

- Kundenbeziehungen gezielt pflegen (Rückgang erkennen)
- Retention-Kampagnen planen
- Umsatzprognosen verfeinern

6.1.3 Mitarbeiterleistung nach Verkauf vergleichen

Zweck: Verkaufsleistung einzelner Mitarbeiter/Außendienstler bewerten

Ablauf:

1. Benutzer öffnet Mitarbeiter-Auswertung
2. Definiert Zeitraum und optional Mitarbeiter/Team-Filter
3. System berechnet pro Mitarbeiter: Verkaufsmenge, Umsatz, Gewinn, Quote
4. Zeigt Ranking-Tabelle mit Leistungsindikatoren
5. Vergleicht gegen Durchschnitt und Plan-Sollwert
6. Chart zeigt Best-Performer vs. Underperformer
7. Ermöglicht PDF-Export als Leistungsbericht

Betroffene Felder: Employee (Name, Department), Sales (Amount, Quantity, CreatedByI3D), Plan/Target-Tabelle

Auswirkungen:

- Provisionsabrechnung basierend auf Daten
- Schulungsbedarf identifizieren
- Team-Incentives justieren

6.1.4 Ticket-Statistik (Support-Leistung)

Zweck: Support-Team-Leistung und Ticket-Bearbeitungsqualität analysieren

Ablauf:

1. Benutzer wählt "Ticket-Statistik" als Auswertungstyp
2. Definiert Zeitraum, Priorität, Status-Filter
3. System aggregiert pro Mitarbeiter: Ticket-Anzahl, Ø Bearbeitungszeit, Eskalationen, Kundenzufriedenheit
4. Zeigt SLA-Einhaltung in % (Target vs. Actual)
5. Identifiziert häufigste Ticket-Kategorien
6. Erstellt Heatmap: wann anfallen meiste Tickets
7. Exportiert als Qualitäts- und Kapazitätsbericht

Betroffene Felder: Ticket (Title, Priority, Status, CreatedByI3D, CreatedDate, ClosedDate), TicketHistory, Customer

Auswirkungen:

- Support-Personal planen (Spitzenlast erkennen)
- Training-Schwerpunkte definieren
- SLA-Compliance überwachen

6.1.5 Einkaufsstatistik und Lieferanten-Leistung

Zweck: Lieferanten-Leistung und Einkaufsausgaben analysieren

Ablauf:

1. Benutzer öffnet Einkaufsauswertung
2. Wählt Zeitraum und Lieferanten-Filter
3. System zeigt pro Lieferant: Bestellvolumen, Gesamtausgaben, Liefertreue %, Rückmeldequote
4. Ranking nach Spend und Zuverlässigkeit
5. Vergleicht Lieferanten-Leistung visuell
6. Ermöglicht Drill-down zu einzelnen Bestellungen
7. Export für Lieferanten-Gespräche und Negotiations

Betroffene Felder: Supplier, PurchaseOrder (Amount, OrderDate, DeliveryDate), ExternalServiceData

Auswirkungen:

- Lieferanten-Verträge neu verhandeln
- Single-Source-Risks identifizieren
- Einkaufsmengen optimieren

6.1.6 Gewinn-Analyse nach Materialgruppe

Zweck: Profitabilität verschiedener Produktkategorien vergleichen

Ablauf:

1. Benutzer wählt "Gewinnauswertung nach Materialgruppe"

2. Definiert Zeitraum und Materialgruppen-Filter (optional)
3. System berechnet pro Materialgruppe: Verkaufsmenge, Umsatz, Gesamtkosten, Gewinn, Gewinnmarge %
4. Zeigt Tabelle mit allen Materialgruppen sortiert nach Rentabilität
5. Identifiziert Gewinner (>25% Marge) und Problemgruppen (<5% Marge)
6. Visualisiert Trend über Monatsvergleich
7. Export mit Empfehlungen (Preisstrategie, Lagerreduzierung)

Betroffene Felder: MaterialGroup (Name), Article (MaterialGroupI3D), Sales (Quantity, Amount, Cost), Stock

Auswirkungen:

- Preisanpassungen für unrentable Gruppen
- Sortimentsbereinigung durchführen
- Strategische Fokussierung auf High-Margin-Produkte

6.1.7 Filialvergleich und Standort-Performance

Zweck: Performance verschiedener Vertriebsstandorte/Filialen vergleichen

Ablauf:

1. Benutzer selektiert "Filialvergleich"
2. Wählt Zeitraum und Filialen-Filter
3. System berechnet pro Filiale: Ø-Tagesverkauf, Gesamtumsatz, Personalkostenquote, Gewinn, Rendite
4. Ranking nach Umsatz und Profitabilität
5. Zeigt Abweichungen vs. Budget
6. Ermöglicht Drill-down zu Abteilungen/Teams pro Filiale
7. Export für Filialleiter-Meetings

Betroffene Felder: Branch (Name, Location, BranchI3D), Sales (BranchI3D), Employee (BranchI3D), Budget-Planung

Auswirkungen:

- Ressourcen zu Top-Filialen verschieben
- Underperformer analysieren und optimieren
- Best-Practice zwischen Filialen austauschen

6.1.8 Lagerbestands-Entwicklung und Umschlag

Zweck: Bestandsbewegung und Lagerumschlag analysieren

Ablauf:

1. Benutzer wählt "Lagerauswertung"
2. Definiert Zeitraum und ggf. Lagerstandorte/Materialgruppen
3. System zeigt: Anfangsbestand, Zu/Abgänge, Endbestand pro Artikel
4. Berechnet Lagertage (wie lange im Lager) und Umschlaggeschwindigkeit
5. Identifiziert Slow-Movers (>180 Tage ohne Verkauf) und Fast-Movers
6. Zeigt Lagerwertentwicklung im Zeitverlauf (Trend)
7. Export mit Optimierungsempfehlungen (Abverkauf, Neubestellung)

Betroffene Felder: Stock (Quantity, StockValue, ArticleI3D), Stock-History, Sales (CreatedDate)

Auswirkungen:

- Lagerverwaltung optimieren
- Überbestände abbauen
- Liquidität verbessern (Kapital aus Lager freisetzen)

6.2 Leistungsnachweise (Performance Reports)

Module Path: `src/centron/Centron.WPF.UI/Modules/Statistics/EmployeeAnalytics`
Controller: `EmployeeAnalyticsAppModuleController`
ViewModel: `EmployeeAnalyticsViewModel`
Category: Controlling/Analytics
Description: Darstellung der Mitarbeiterauslastung und Leistungskennzahlen
License: LicenseGuids.Centron oder LicenseGuids.Analytics

Use Cases

6.2.1 Wochenauslastung pro Mitarbeiter anzeigen

Zweck: Tagesaktuelle Auslastung und verfügbare Kapazität pro Mitarbeiter sehen

Ablauf:

1. Benutzer (Projektleiter/Personalleiter) öffnet Leistungsnachweise-Modul
2. System lädt aktuelle Woche standardmäßig
3. Zeigt Tabelle: Mitarbeiter, Mo-Fr Auslastung in %, Wochengesamt, verfügbare Stunden
4. Farben kennzeichnen: Grün <70%, Gelb 70-90%, Rot >90%
5. Benutzer kann Woche vor/zurück navigieren
6. Klick auf Mitarbeiter zeigt Details: gebuchte Projekte/Aufgaben mit Stundenanteil
7. "Freie Kapazität" erkennbar für neue Projekteinplanung

Betroffene Felder: Employee (Name, Department, Capacity-Stunden), Workload (Date, Hours, ProjectI3D), Ticket (Assigned Employee, EstimatedHours)

Auswirkungen:

- Projektressourcenplanung optimieren
- Überlastung vermeiden (Burnout-Prävention)
- Freie Kapazität für neue Projekte nutzen
- Vertretungsregelungen frühzeitig erkennen

6.2.2 Monatsauslastung und Abwesenheiten berücksichtigen

Zweck: Monatliche Auslastungsplanung unter Berücksichtigung von Urlaub/Krankenstand

Ablauf:

1. Benutzer wählt Monat aus oder navigiert zu Monat
2. System berechnet für jeden Mitarbeiter: Sollstunden (Arbeitstage × 8h - Abwesenheitszeiten)
3. Summiert tatsächlich gebuchte Stunden aus Projekten
4. Zeigt: Name, Sollstunden, Ist-Stunden, Auslastung %, Urlaub-Tage, Abweichung
5. Identifiziert unter/überausgelastete Mitarbeiter
6. Chart zeigt Trend über letzte 12 Monate
7. Export als PDF-Report für Personalgespräche

Betroffene Felder: Employee, Absence (Type: Urlaub/Krank/Studium, Duration), Workload (Date, Hours), Department-Sollstunden

Auswirkungen:

- Personalkosten kontrollieren (zu viel/wenig Auslastung kostet)
- Fehlzeitenquoten analysieren
- Mitarbeitergespräche datengestützt führen
- Urlaubsplanung optimieren

6.2.3 Jahresauslastung und Leistungstrends

Zweck: Leistungsentwicklung einzelner Mitarbeiter über Jahresvergleich bewerten

Ablauf:

1. Benutzer wählt Mitarbeiter und Vergleichsjahre
2. System aggregiert monatliche Auslastung pro Jahr
3. Zeigt Tabelle: Monat, Jahr 1, Jahr 2, Jahr 3, Trend, Durchschnitt
4. Farblich kodiert: 100% = Grün, unter 80% oder über 120% = Rot
5. Chart zeigt Kurven-Verlauf aller Jahre überlagert
6. Ermittelt Ø-Auslastung und Standardabweichung
7. Erlaubt Vergleich mit Team-Durchschnitt

Betroffene Felder: Employee, Workload (Monthly aggregation), Absence (Annual summary)

Auswirkungen:

- Leistungstrends erkennen (steigend/fallend)
- Mitarbeiter-Entwicklung nachverfolgen
- Gehaltserhöhungen/Beförderungen begründen
- Probleme frühzeitig adressieren

6.2.4 Projektauslastung und Team-Kapazität

Zweck: Pro Projekt die Ressourcenauslastung und Kapazitätsengpässe erkennen

Ablauf:

1. Benutzer wählt Projekt-Filter
2. System zeigt alle dem Projekt zugeordneten Mitarbeiter
3. Tabelle: Mitarbeiter, Geplante Stunden (Plan), Gebuchte Stunden (Ist), Abweichung, Zeit auf Projekt %
4. Gesamtauslastung des Projekts: Ø über alle Team-Mitglieder
5. Warnt vor Engpässen: wenn einzelne Mitarbeiter >100%
6. Zeigt Trend: Auslastung über Projektdauer (Ramping-up/down erkennbar)
7. Erlaubt Kapazitätsverschiebungen (andere Mitarbeiter hinzufügen)

Betroffene Felder: Project, Employee (ProjectI3D), Workload (ProjectI3D), Task (EstimatedHours, ActualHours)

Auswirkungen:

- Projektpläne realistisch gestalten
- Ressourcen-Konflikte vermeiden
- Projekterfolgsquoten erhöhen
- Risiken frühzeitig erkennen

6.2.5 Abteilungs-Kapazitätsverlauf

Zweck: Abteilungsüberblick: Gesamtauslastung und Kapazitätsplanung auf Abteilungsebene

Ablauf:

1. Benutzer wählt Abteilung (z.B. Entwicklung, Support, Sales)
2. System aggregiert alle Mitarbeiter der Abteilung
3. Zeigt: Ø-Auslastung Abteilung, Minimal/Maximal-Auslastung im Team, Kapazitätsreserve in Stunden
4. Heatmap: Pro Mitarbeiter Auslastungs-Balken (wie ausgelastet)
5. Trend-Chart: Auslastung über letzte 6 Monate
6. Warnt vor kritischer Überauslastung (>95% durchschnittlich)
7. Planungs-Werkzeuge für nächste Projekte

Betroffene Felder: Department (Name), Employee (DepartmentId3D), Workload (aggregiert nach Abteilung)

Auswirkungen:

- Neueinstellungen begründen (zu wenig Kapazität)
- Outsourcing-Bedarf erkennen
- Abteilungsbudget planen
- Leistungsanforderungen transparent machen

6.2.6 Expertisen und Spezialisierungen visualisieren

Zweck: Wer hat welche Fähigkeiten und wie ausgelastet sind Spezialisten?

Ablauf:

1. Benutzer wählt Expertise-Filter (z.B. "C#", "Datenbankdesign", "UI/UX")
2. System zeigt alle Mitarbeiter mit dieser Expertise
3. Tabelle: Name, Expertise-Level (Junior/Senior/Lead), aktuelle Auslastung, verfügbare Kapazität
4. Warnt vor Bottlenecks: wenn Lead-Spezialisten >95% ausgelastet
5. Zeigt Successors: Junior-Mitarbeiter die in Expertise trainiert werden
6. Erlaubt Skill-Profile zu aktualisieren
7. Export für Trainings- und Succession-Planung

Betroffene Felder: Employee (Skills, ExpertiseLevelId3D), Workload (SkillRequiredId3D), Project (RequiredSkills)

Auswirkungen:

- Kritische Kompetenzen identifizieren
- Wissenstransfer gezielt planen
- Single-Point-of-Failure vermeiden
- Fachkarrieren entwickeln

6.2.7 Überauslastung und Burnout-Risiko erkennen

Zweck: Präventiv identifizieren, welche Mitarbeiter Burnout-Risiko haben

Ablauf:

1. System berechnet pro Mitarbeiter Risiko-Score basierend auf:
   - Durchschnittliche Auslastung >90% über 3 Monate
   - Überstunden pro Woche >10h
   - Ohne längere Pausen/Urlaub >12 Monate
   - Fehlzeitenquote steigend (Frühe Krankheitswarnung)
2. Risiko-Kategorien: Grün (OK), Gelb (Warnung), Rot (Kritisch)
3. Benutzer kann geflaggte Mitarbeiter anklicken → Detailanalyse
4. Empfehlungen: Maßnahmen zur Entlastung
5. Generiert Bericht für Personalleiter und Betriebsrat
6. Tracking: Welche Maßnahmen waren erfolgreich?

Betroffene Felder: Employee (CreatedDate, LastAbsenceDate), Workload (Hours, Date), Absence (StartDate, Duration), Overtime

Auswirkungen:

- Mitarbeitergespräche früher führen
- Ressourcen umverteilen vor Burnout
- Krankheitsrate/Fluktuation senken
- Unternehmenskultur verbessern

6.2.8 Wissenstransfer und Mentoring-Effektivität

Zweck: Nachverfolgung, ob Mentoring und Wissenstransfer-Initiativen erfolgreich sind

Ablauf:

1. Benutzer definiert Mentoring-Pairs (Senior-Junior)
2. System zeigt Wissenstransfer-Indikatoren über Zeit:
   - Junior-Auslastung steigend? (zeigt steigende Eigenständigkeit)
   - Senior-Auslastung sinkend? (zeigt erfolgreiche Delegation)
   - Fehlerquote Junior sinkend?
   - Junior-Gehalt-Anpassungen angebracht?
3. Dashboard: Welche Mentorships funktionieren, welche nicht?
4. Empfehlungen: Mentoring anpassen oder beenden
5. Export für HR-Entwicklungsplanung

Betroffene Felder: Employee (MentorI3D, StartDate), Workload (Mentee-Progress), Quality-Metrics, Salary-Range

Auswirkungen:

- Nachwuchsentwicklung optimieren
- Erfolgreiche Mentoren identifizieren
- Mentoring-Programme verbessern
- Langzeitkarrieren planen

6.3 Management Info (Management Information)

Module Path: `src/centron/Centron.WPF.UI/Modules/Statistics/ManagementInfo`
Controller: `ManagementInfoAppModuleController`
ViewModel: `ManagementInfoViewModel`
Category: Controlling/Analytics
Description: Anzeige der aktuellen Firmenkennzahlen mit interaktiven Charts und Filialen-Filter
License: LicenseGuids.Centron oder LicenseGuids.ManagementInfo

Use Cases

6.3.1 Dashboard mit Echtzeit-Geschäftszahlen öffnen

Zweck: Geschäftsführung erhält auf einen Blick die wichtigsten KPIs (Tagesstand)

Ablauf:

1. Geschäftsführer/Manager öffnet Management Info Dashboard
2. System zeigt standardmäßig heutige Auswertung
3. Zentrale Kennzahlen als große Karten angezeigt:
   - Tagesverkauf (€ und Menge)
   - Tagesgewinn (€ und %)
   - Aktive Aufträge (Anzahl)
   - Ausstehende Zahlungen (€)
   - Lagerbestand (€ Wert)
   - Personalkosten heute (€)
4. Vergleich mit Vortag/Vorjahr in % (Trend-Pfeile)
5. Ø-Werte der letzten 7/30 Tage zum Vergleich
6. Alle Zahlen in Echtzeit aktualisiert (Auto-Refresh alle 5 Min)

Betroffene Felder: Sales (CreatedDate, Amount, Cost, Quantity), Stock (StockValue), Employee (Salary), Order (Status, Amount)

Auswirkungen:

- Tagesgeschäft schnell überblicken
- Anomalien sofort erkennen
- Schnelle Entscheidungen ermöglichen
- Tagesrapporte automatisieren

6.3.2 Filialvergleich mit Heatmap-Visualisierung

Zweck: Performance alle Filialen gleichzeitig bewerten und vergleichen

Ablauf:

1. Dashboard zeigt Filialauswahl (Dropdown oder Schaltflächen)
2. Benutzer kann einzelne Filiale fokussieren oder alle sehen
3. Heatmap wird angezeigt: Spalten=Filialen, Zeilen=KPIs
4. Farbcodierung: Grün >Plan, Gelb ca. Plan, Rot <Plan
5. KPIs pro Filiale: Tagesverkauf, Gewinn, Besucher, Conversion %, Durchschnittseinkauf
6. Klick auf Filiale → Drill-down zu Departments dieser Filiale
7. Export als PNG/PDF für Filial-Manager-Meetings

Betroffene Felder: Branch (Name, BranchI3D, TargetAmount), Sales (BranchI3D, Amount, Date), Visitor-Counter, Budget

Auswirkungen:

- Filial-Performance transparent
- Best-Practices identifizieren
- Benchmarking zwischen Filialen
- Ressourcenallokation optimieren

6.3.3 Materialgruppen-Rentabilität visualisieren

Zweck: Schnell sehen, welche Produktgruppen gerade rentabel/unrentabel sind

Ablauf:

1. Benutzer wählt Zeitraum (heute, diese Woche, dieser Monat)
2. System zeigt Materialgruppen-Analyse:
   - Tabelle: Materialgruppe, Verkaufte Menge, Umsatz, Kosten, Gewinn, Marge %
   - Balken-Chart: Marge % pro Gruppe (sortiert absteigend)
   - Top 3 (grün) und Bottom 3 (rot) Groups hervorgehoben
3. Trend-Pfeile: Marge vs. Vormonat (steigend/fallend)
4. Klick auf Gruppe → zeigt Best-Selling Articles darin
5. Warnung wenn negative Gruppen oder unter 10% Marge

Betroffene Felder: MaterialGroup (Name), Article (MaterialGroupI3D), Sales (Quantity, Amount, Cost, Date), Stock

Auswirkungen:

- Preisschwäche schnell erkennen
- Sortiment optimieren
- Strategische Entscheidungen (Delisten/Fokussierung)
- Kostenreduktion priorisieren

6.3.4 Umsatzziele und Abweichungen anzeigen

Zweck: Ist Umsatz auf Kurs oder müssen Maßnahmen eingeleitet werden?

Ablauf:

1. System zeigt aktuellen Umsatzplan (Budget) vs. Ist
2. Auf mehreren Ebenen: Gesamt, pro Filiale, pro Team, pro Kategorie
3. Für jeden Bereich: Soll-Umsatz, Ist-Umsatz, Abweichung €/%, Trend-Forecast bis Monatsende
4. Farbcodierung: Grün ≥100% Plan, Gelb 95-99%, Rot <95%
5. Wenn rot: Warnung und Empfehlung (z.B. "Gewinnmarge reduzieren" oder "Rabatte prüfen")
6. Chart zeigt täglichen Fortschritt: Soll-Linie vs. Ist-Linie (Catch-up möglich?)
7. Prognose: "Bei aktuellem Trend erreichen Sie X% des Plans"

Betroffene Felder: Sales (Amount, Date), Budget/Plan-Tabelle (TargetAmount, TargetDate), Forecast-Model

Auswirkungen:

- Umsatzziele kontrollieren
- Gegensteuern rechtzeitig einleiten
- Mitarbeitermotivation (Fortschritt sichtbar)
- Prognosegenauigkeit verbessern

6.3.5 Gewinn-Vergleich: Aktuell vs. Plan vs. Vorjahr

Zweck: Rentabilität bewerten auf Soll-Ist-Vergleich

Ablauf:

1. Dashboard-Bereich "Profitabilität" zeigt 3 Spalten-Vergleich:
   - Spalte 1: Dieser Monat bislang (Ist)
   - Spalte 2: Plan für diesen Monat (Budget-Target)
   - Spalte 3: Vorjahr gleiche Periode
2. Für jede Spalte: Gewinn €, Gewinnmarge %, Prognose Gesamtmonat
3. Abweichungs-Analyse: Worin unterscheiden sich die Perioden?
   - Mehr/weniger Volumen?
   - Bessere/schlechtere Marge?
   - Höhere/niedrigere Kosten?
4. Trend-Charts zeigen Entwicklung über letzte 12 Monate
5. Export für Geschäftsbesprechungen

Betroffene Felder: Sales (Amount, Cost, Date), Budget, Costs-Tabelle, Forecast

Auswirkungen:

- Profitabilität steuern
- Kosteneinsparungen priorisieren
- Preis-/Volumen-Mix analysieren
- Strategische Anpassungen einleiten

6.3.6 Lagerbestands-Wertentwicklung und Verfallsrisiken

Zweck: Lagervermögen überwachen und Risiken erkennen

Ablauf:

1. Dashboard zeigt Lagerbestands-Sektion:
   - Gesamtlagerwert (€) heute vs. gestern vs. 30-Tage-Ø
   - Top 20 Artikel nach Lagerwert (um Fokus auf kritische Bestände)
   - Slow-Mover (>180 Tage ohne Verkauf) mit Lagerwert & Verfallsrisiko
   - Verfallsdatum-Kalender: Welche Artikel verfallen in den nächsten 30/90 Tagen?
2. Warnungen: Lagerwert steigt ohne Verkaufssteigerung = Risiko
3. Empfehlungen: Abverkauf initiieren, Rückvergütungen von Lieferanten prüfen
4. Vergleich mit Budgetiertem Lagerwert

Betroffene Felder: Stock (Quantity, StockValue, ArticleI3D), Article (ExpirationDate), Sales (CreatedDate, Quantity), Budget-Lagerwert

Auswirkungen:

- Vermögensverschwendung vermeiden
- Liquidität verbessern
- Lagerkostenquote reduzieren
- Abschreibungsrisiken erkennen

6.3.7 Personalkosten-Analyse (Quotenvergleich)

Zweck: Sind Personalkosten im Plan? Sind sie wirtschaftlich?

Ablauf:

1. Sektion zeigt Personalkosten-Quote: Personalkosten / Umsatz (€ und %)
2. Vergleich mit Plan, Branchendurchschnitt, Vorkwartale
3. Aufschlüsselung nach: Gehalt, Sozialabgaben, Schulung, Sozialleistungen
4. Pro Filiale/Abteilung die Quote anzeigen (Vergleich)
5. Wenn Quote zu hoch: Warnung und Optimierungsvorschläge
6. Trend über 12 Monate: steigt oder sinkt die Quote?
7. Forecast: Wie wird Quote am Jahresende sein?

Betroffene Felder: Employee (Salary, StartDate), Sales (Amount), Absence (Duration), Training-Kosten

Auswirkungen:

- Personalbudget kontrollieren
- Überbesetzung erkennen
- Produktivität pro Mitarbeiter berechnen
- Gehaltsanpassungen validieren

6.3.8 Echtzeit-Warnungen und Anomalien-Erkennung

Zweck: Automatisch über Probleme benachrichtigt werden, nicht manuell suchen

Ablauf:

1. System definiert Schwellwerte basierend auf Branchen-Standards:
   - Tagesverkauf sinkt >20% vs. Durchschnitt → WARNUNG
   - Retourn-Quote steigt >5% → WARNUNG
   - Lagerwert wächst ohne Verkaufswachstum → WARNUNG
   - Ausstehende Zahlungen >Plan-Prognose → WARNUNG
   - Fehlerquote im Ticket-System >10% → WARNUNG
2. Warnungen werden im Dashboard angezeigt (Ampel-System: Rot/Gelb/Grün)
3. Klick auf Warnung → Drill-down zur Ursachenanalyse
4. Optional E-Mail/SMS-Benachrichtigungen an Geschäftsführer
5. Historisches Warnung-Log: Welche Probleme gab es?

Betroffene Felder: Alle Business-Entities (Sales, Stock, Order, Ticket, etc.)

Auswirkungen:

- Proaktiv statt reaktiv handeln
- Geschäftsführer entlastet
- Risiken früh erkennen
- Schwellwerte kontinuierlich anpassen/lernen

6.4 Mitarbeiterauslastung (Employee Utilization)

Module Path: `src/centron/Centron.WPF.UI/Modules/MyCentron/MyDay/EmployeeOverview`
Controller: `MyDayEmployeeOverviewAppModuleController`
ViewModel: `MyDayEmployeeOverviewWrapperViewModel`
Category: Controlling/Analytics
Description: Übersicht über die Arbeitstage von Mitarbeitern und Abteilungen mit Kapazitätsanzeige
License: LicenseGuids.Centron oder LicenseGuids.MyCentron

Use Cases

6.4.1 Tagesübersicht Mitarbeiterkapazität anzeigen

Zweck: Auf einen Blick sehen, welche Mitarbeiter heute verfügbar sind

Ablauf:

1. Benutzer (Disponent/Projektleiter) öffnet Mitarbeiterauslastung-Modul
2. System zeigt heutiges Datum und Liste aller Mitarbeiter
3. Für jeden Mitarbeiter: Name, Abteilung, geplante Stunden, bereits gebuchte Stunden, verfügbar noch
4. Farbe-Kodierung: Grün (viel frei), Gelb (teilweise ausgelastet), Rot (voll ausgelastet/über)
5. Icons kennzeichnen Status: im Urlaub, krank, auswärts, verfügbar
6. Klick auf Mitarbeiter → Details mit Tagesplan zeigen
7. Schnelle Kapazität anpassen per Drag&Drop (Aufgaben zuordnen)

Betroffene Felder: Employee (Name, Department), Absence (Date, Type), Workload (Date, Hours, EmployeeI3D), Task (AssignedI3D)

Auswirkungen:

- Tagesplanung optimieren
- Verfügbare Ressourcen nutzen
- Übergangslösungen bei Ausfällen finden
- Arbeitskräfte-Flexibilität erhöhen

6.4.2 Wochenplanungsmodus mit Auslastungs-Wärmebild

Zweck: Ganze Woche im Überblick, wo sind Engpässe?

Ablauf:

1. Benutzer wählt Wochenansicht
2. System zeigt: Spalten=Tage (Mo-Fr), Zeilen=Mitarbeiter/Teams
3. Jede Zelle zeigt Auslastung %, farbcodiert (Wärmebild): Blau <50%, Grün 50-80%, Gelb 80-100%, Rot >100%
4. Klick auf Zelle → Details zeigen (welche Projekte/Aufgaben)
5. Automatische Vorschläge: "Verschieben Sie Mitarbeiter X von Mittwoch auf Dienstag"
6. Export als Planungsbericht

Betroffene Felder: Employee, Workload (Date, Hours), Task (ScheduledDate), Absence

Auswirkungen:

- Wochenplanung proaktiv optimieren
- Engpässe früher erkennen
- Arbeitsverteilung ausgleichen
- Überstunden vermeiden

6.4.3 Abteilungs-Kapazitätsplanung und Prognose

Zweck: Genug Personal in der Abteilung für geplante Arbeit?

Ablauf:

1. Benutzer wählt Abteilung (z.B. Entwicklung, Support)
2. System zeigt: Sollkapazität (Mitarbeiterzahl × 8h), verfügbare Kapazität heute/diese Woche/dieser Monat
3. Geplante Arbeit (Workload) vs. Kapazität vergleichen
4. Wenn Workload > Kapazität: Warnung mit Empfehlung (Zusatzzeit, Outsourcing, Priorisierung)
5. Trend über letzte 12 Monate: War die Abteilung chronisch überbelastet?
6. Forecast: Basierend auf geplanten Projekten, wie sieht es in 3/6 Monaten aus?
7. Export für Budgetplanung (Neueinstellungen rechtfertigen)

Betroffene Felder: Department (Name), Employee (DepartmentI3D), Workload (aggregiert), Project (StartDate, EndDate, Effort-Estimation), Budget

Auswirkungen:

- Personalbedarf begründen
- Über/Unterauslastung erkennen
- Projektprioritäten setzen
- Recruiting-Bedarf planen

6.4.4 Urlaubs- und Abwesenheitsplanung

Zweck: Ausfallzeiten koordinieren, Vertretungen rechtzeitig planen

Ablauf:

1. Benutzer sieht Kalenderansicht mit allen Abwesenheitsarten (Urlaub, Krank, Schulung, Sabbatical)
2. Pro Mitarbeiter: genehmigte Urlaubstage vs. verfügbare (restliche) anzeigen
3. Automatische Warnung: "Mitarbeiter X hat kein Resturlaub mehr bis Jahresende"
4. Warnung vor kritischen Ausfällen: "2 von 3 C#-Entwicklern sind die nächste Woche im Urlaub"
5. Suggestionsmodus: "Verschieben Sie Urlaub von Mitarbeiter Y, um Engpass zu vermeiden?"
6. Automatische Abwesenheits-Meldung an Anrufer
7. Urlaubsverteilungsbericht (ist Urlaubsverteilung fair?)

Betroffene Felder: Employee, Absence (Type, StartDate, EndDate, DayCount, ApprovedByI3D), Arbeitsvertrag (Urlaubstage pro Jahr)

Auswirkungen:

- Urlaubskonflikte vermeiden
- Fair Urlaubsverteilung sicherstellen
- Personalvertretung proaktiv lösen
- Compliance mit Arbeitsrecht

6.4.5 Übersteunden und Mehrarbeit erfassen und tracken

Zweck: Wer macht viel Überstunden? Ist das nachhaltig?

Ablauf:

1. System zeigt pro Mitarbeiter monatliche Überstunden (über 8h/Tag hinaus)
2. Trend über 6/12 Monate anzeigen
3. Gesamtüberblick: welche Mitarbeiter machen viele Überstunden?
4. Warnung bei Mitarbeitern mit >10h Überstunden/Woche (Burnout-Risiko)
5. Auslöst Benachrichtigung an Vorgesetzten: "Mitarbeiter X macht zu viele Überstunden"
6. Abrechnung: Überstunden kompensieren (Freizeit, Bezahlung) tracken
7. Report: Überstundenquote vs. Budget

Betroffene Felder: Workload (Hours, OvertimeHours, Date), Employee (Salary, OvertimeRate), Absence (CompensationDate, CompensationHours)

Auswirkungen:

- Mitarbeiter-Burnout vermeiden
- Realistische Personalplanung
- Kostenkalkulationen verfeinern
- Work-Life-Balance verbessern

6.4.6 Spezialist-Kapazität und kritische Fähigkeiten

Zweck: Wer sind die kritischen Spezialisten? Wie ausgelastet sind sie?

Ablauf:

1. Benutzer kann nach Fähigkeiten/Skills filtern
2. System zeigt Experten dieser Fähigkeit: Name, Expertise-Level, aktuelle Auslastung, verfügbare Stunden
3. Warnung bei: nur 1 Experte vorhanden (Risiko!) oder Experte >95% ausgelastet
4. Succession-Plan: Welche Juniors werden in dieser Fähigkeit trainiert?
5. Empfehlung: "Trainieren Sie Mitarbeiter Y zu Backup für kritische Skill Z"
6. Prognose: Wenn dieser Experte kündigt, wie lange bis Ersatz?

Betroffene Felder: Employee, SkillProfile/ExpertiseTable, Workload, Mentoring-Beziehungen

Auswirkungen:

- Single-Point-of-Failure vermeiden
- Succession-Planung aktiv gestalten
- Wissenstransfer gezielt fördern
- Unternehmensstabilität verbessern

6.4.7 Abteilungs-Vergleich und Benchmarking

Zweck: Welche Abteilung ist effizienter ausgelastet?

Ablauf:

1. System zeigt Vergleich mehrerer Abteilungen Side-by-Side
2. KPIs pro Abteilung: Ø Auslastung, Überstundenquote, Fehlerquote, Produktivität (Output/Person)
3. Ranking nach Effizienz (wer macht am meisten bei normaler Arbeitszeit?)
4. Identifiziert Best Practices (welche Abteilung macht es besser?)
5. Gibt Empfehlungen: "Abteilung A sollte von Abteilung B das Planungs-System übernehmen"
6. Export für Abteilungsleiter-Meetings

Betroffene Felder: Department (Name), Employee (DepartmentI3D), Workload, Quality-Metrics, Output-Metrics

Auswirkungen:

- Best-Practice teilen
- Weniger effiziente Abteilungen verbessern
- Ressourcen optimal verteilen
- Leistungskultur entwickeln

6.4.8 Szenarien-Planung: Was-wenn-Analysen

Zweck: Wie würde sich Änderung auf Kapazität auswirken?

Ablauf:

1. Benutzer kann "Was-wenn" Szenarien simulieren:
   - "Wenn Mitarbeiter X kündigt, wer übernimmt seine Projekte?"
   - "Wenn wir 2 neue Projekte starten, reicht die Kapazität?"
   - "Wenn Mitarbeiter Y für 3 Monate in Training geht?"
2. System kalkuliert automatisch Auswirkungen
3. Zeigt: Neue Auslastung, Überstundenprognose, welche Projekte gefährdet, Lösungsvorschläge
4. Kann mehrere Szenarien parallel vergleichen
5. Export der Analyse für Entscheidungsfindung

Betroffene Felder: Employee, Workload, Project, Budget, Resource-Planning

Auswirkungen:

- Personalentscheidungen datengestützt treffen
- Risiken frühzeitig erkennen
- Strategische Planung verbessern
- Kosten senken durch bessere Planung

6.5 MSP-Auswertung (MSP Evaluation)

Module Path: `src/centron/Centron.WPF.UI/Modules/Global/MSPLicensesCompare`
Controller: `MSPComparerAppModuleController`
ViewModel: `MSPComparerViewModel`
Category: Controlling/Analytics
Description: Auswertung und Vergleich von MSP-Collector Imports mit Lizenz-Analysen
License: LicenseGuids.Centron oder LicenseGuids.MSP

Use Cases

6.5.1 MSP-Lizenz-Datenimport und Vergleich

Zweck: Lizenzdaten von MSP-Plattform (Microsoft, cloud-Provider) importieren und Abweichungen prüfen

Ablauf:

1. Benutzer öffnet MSP-Auswertung-Modul
2. Zeigt verfügbare MSP-Datenquellen (z.B. Microsoft CSP, AWS, Azure)
3. Import durchführen (CSV, XML, oder direkte API-Integration)
4. System vergleicht: Lizenz-Bestand in MSP vs. c-entron-Datenbank
5. Identifiziert Diskrepanzen: "In MSP 50 Lizenzen, aber nur 40 in c-entron erfasst"
6. Zeigt Lizenz-Details: Typ, Laufzeit, Kosten, Nutzer-Zuordnung
7. Empfehlungen: Lizenzen aktualisieren/abgleichen

Betroffene Felder: MSPLicenseData (LicenseType, Quantity, Cost, ValidFrom/To), Contract, ExternalServiceData

Auswirkungen:

- Lizenz-Kosten kontrollieren
- Über/Unter-Lizenzierung erkennen
- Compliance-Risiken minimieren
- Abrechnungsgenauigkeit verbessern

6.5.2 Lizenz-Nutzungsvergleich: Soll vs. Ist

Zweck: Zahlen wir für Lizenzen die wir nicht nutzen?

Ablauf:

1. System zeigt pro Lizenz-Typ: Gekaufte Lizenzen vs. tatsächlich genutzte
2. Nutzer-Zuordnung: Wer verwendet welche Lizenz
3. Detektiert: Idle-Lizenzen (gekauft aber nicht verwendet) seit >30 Tagen
4. Warnung bei Übernutzung (mehr Nutzer als Lizenzen)
5. Empfehlung: "Freigeben Sie 10 Office-Lizenzen, diese werden nicht genutzt" (monatliche Ersparnis)
6. Langzeit-Trend: Ist die Lizenznutzung steigend/fallend?
7. ROI-Berechnung pro Lizenz-Typ

Betroffene Felder: MSPLicenseData, UserLicenseAssignment, Workload (für Nutzungs-Tracking)

Auswirkungen:

- Lizenz-Kosten durch Optimierung senken
- Budget effizienter gestalten
- Compliance überwachen
- Software-Audit vorbereiten

6.5.3 Kostenvergleich und Budget-Planung

Zweck: Sind Lizenz-Ausgaben in den Grenzen?

Ablauf:

1. Dashboard zeigt: Monatliches Lizenz-Budget vs. aktuelle Ausgaben
2. Detaillierung: Kosten pro Lizenz-Kategorie (Office, ERP, Cloud, etc.)
3. Vergleich mit Vormonat/Vorkwartalen
4. Trend-Chart: Kostenentwicklung über 12 Monate
5. Forecast: Basierend auf Trend, wie hoch sind Gesamtkosten am Jahresende?
6. Warnung: "Bei aktuellem Trend überschreiten Sie Budget um 15%"
7. Maßnahmen-Empfehlung: Anzahl reduzieren, zu günstigerem Anbieter wechseln, Bundling-Rabatte

Betroffene Felder: MSPLicenseData (Cost, ValidFrom/To), Budget-Planung, Contract (Price, Discount)

Auswirkungen:

- Kostenkontrolle verschärfen
- Budget-Überraschungen vermeiden
- Vertragsverhandlungen unterstützen
- CFO-Reporting automatisieren

6.5.4 Service-Level und Performance-Metriken

Zweck: Hält der MSP-Provider die vereinbarten Service-Level ein?

Ablauf:

1. System importiert Verfügbarkeitsdaten vom MSP-Provider
2. Zeigt: Uptime %, SLA-Ziele (z.B. 99.9%), tatsächliche Verfügbarkeit
3. Ausfallzeiten analysieren: Wann, wie lange, Auswirkung
4. Provider-Leistung bewerten: Grün (>99%), Gelb (95-99%), Rot (<95%)
5. Incident-Historie: Welche Probleme gab es, wie wurden sie gelöst?
6. Berechnet "Credits" die dem Unternehmen zustehen für SLA-Verletzungen
7. Report für Vertragsverhandlungen

Betroffene Felder: MSPLicenseData, ServiceLevelAgreement, IncidentLog, Uptime-Metrics

Auswirkungen:

- Dienstqualität kontrollieren
- Provider-Leistung bewerten
- Finanzielle Ansprüche durchsetzen
- Verträge auf Basis von Leistung bewerten

6.5.5 Multi-Provider-Vergleich

Zweck: Vergleiche Kosten und Leistung zwischen mehreren MSP-Anbietern

Ablauf:

1. Benutzer zeigt mehrere Provider-Quellen an (z.B. Microsoft, AWS, Google Cloud)
2. System normalisiert Daten und macht vergleichbar
3. Vergleichstabelle: Lizenztyp, Kosten, Verfügbarkeit, Support, Vertragsbedingungen
4. Gesamtkostenvergleich (TCO Total Cost of Ownership)
5. Ranking der Provider nach Kosteneffizienz
6. Szenarien: "Was kostet Migration zu Provider X?"
7. Export für Einkaufs-Entscheidungen

Betroffene Felder: Multiple MSPLicenseData-Quellen, Cost-Models, SLA-Data

Auswirkungen:

- Kostenoptimierungen durch Provider-Wechsel
- Verhandlungsposition stärken
- Beste-Klasse-Benchmarks etablieren
- Finanzielle Ziele erreichen

6.5.6 Compliance und Lizenz-Audit

Zweck: Sind wir lizenzkonform? Risikofreie Nutzung?

Ablauf:

1. System prüft: Gekaufte Lizenzen vs. tatsächliche Nutzer/Installationen
2. Warnung bei Übernutzung (Compliance-Risiko)
3. Warnung bei verfallenen/ungültigen Lizenzen
4. Dokumentiert: Wer hat welche Lizenz, seit wann, bis wann gültig
5. Audit-Trail: Änderungen an Lizenz-Zuordnungen tracken
6. Generiert Compliance-Bericht für externe Audits
7. Empfehlungen zur Risikominderung

Betroffene Felder: UserLicenseAssignment (StartDate, EndDate), Software-Installation, License-Entitlement

Auswirkungen:

- Bußgelder durch Lizenzverstoß vermeiden
- Audit-Prozesse vereinfachen
- Nachweise dokumentieren
- Rechtsicherheit gewährleisten

6.6 MSP-Collector (MSP Collector)

Module Path: `src/centron/Centron.WPF.UI/Modules/Statistics/MspCollectors`
Controller: `MspCollectorAppModuleController`
ViewModel: `MspCollectorAppViewModel` (core: `MspCollectorViewModel`)

Category: Controlling/Analytics
Description: Oberfläche für den MSP-Collector mit Reports und Daten-Mapping
License: LicenseGuids.Centron oder LicenseGuids.MSP

Use Cases

6.6.1 MSP-Daten-Import aus verschiedenen Quellen

Zweck: Lizenzdaten von verschiedenen MSP-Plattformen importieren und normalisieren

Ablauf:

1. Benutzer öffnet MSP-Collector-Modul
2. Wählt Datenquelle: Microsoft CSP, AWS, Google Cloud, oder manuelles Upload (CSV/XML)
3. Definiert Mapping: Welche Spalten entsprechen welchen Feldern in c-entron?
4. Import durchführen (eine oder mehrere Dateien)
5. System validiert Daten: Duplikate, ungültige Formate, fehlende Felder
6. Zeigt Import-Zusammenfassung: X neue Lizenzen, Y aktualisiert, Z Fehler
7. Import bestätigen oder zurückweisen

Betroffene Felder: MSPCollectorData (SourceSystem, ImportDate, RawData), Contract, ExternalServiceData

Auswirkungen:

- Automatisierter Daten-Abgleich
- Fehlerquoten reduzieren
- Import-Prozess beschleunigen
- Manuelle Dateneingabe eliminieren

6.6.2 Artikel-Referenzen und Verknüpfungen

Zweck: Welche c-entron Artikel correspond zu den MSP-Lizenzen?

Ablauf:

1. Nach Import zeigt System Artikel-Zuordnungs-View
2. Pro MSP-Lizenz: Zeige ähnliche c-entron-Artikel
3. Benutzer ordnet manuell zu (oder Algorithmus schlägt vor)
4. Speichert Zuordnung für zukünftige Importe
5. Duplicate-Detection: Zwei verschiedene Lizenzen auf gleichen Artikel?
6. Reports: "Welche c-entron-Artikel sind noch nicht zu MSP-Lizenzen gemappt?"
7. Export: Zuordnungs-Mapping für andere Systeme

Betroffene Felder: Article (ArticleNumber, Description), MSPCollectorData, ArticleMapping

Auswirkungen:

- Inventar-Genauigkeit verbessern
- Abrechnung korrekt kalkulieren
- Cross-System-Verlinkung etablieren
- Reporting konsistent machen

6.6.3 Kundliche Verknüpfungen und Organisationen

Zweck: Welcher Kunde gehört zu welcher MSP-Lizenz?

Ablauf:

1. Nach Artikel-Mapping zeigt System Kunden-Zuordnungs-View
2. Pro MSP-Datensatz: Zeige entsprechende Kunden in c-entron

3. Benutzer ordnet zu (oder Algorithmus schlägt vor)
4. Speichert Zuordnung
5. Warnung: "Dieser MSP-Datensatz ist noch keinem Kunden zugeordnet"
6. Reports: Customer-Usage, Lizenzierung pro Kunde
7. Export für Kundenbilling

Betroffene Felder: Customer (CustomerNumber, Name), MSPCollectorData, Organization-Mapping

Auswirkungen:

- Kundenzuordnung automatisieren
- Abrechn ungsgenauigkeit verbessern
- Lizenz-Nutzung pro Kunde transparent
- Revenue-Tracking verfeinern

6.6.4 MSP-Reports und Datenansichten

Zweck: Verschiedene Reports auf importierten MSP-Daten anschauen

Ablauf:

1. System bietet vordefinierte Reports:
   - Lizenz-Typ nach Anzahl/Kosten
   - Top-Kunden nach Lizenzvolumen
   - Lizenz-Trend über Monate
   - Ungemappte Datensätze (noch zu bearbeiten)
   - Import-Fehler-Report (was ist schiefgegangen?)
2. Jeder Report hat Filter-Optionen (Datum, Quelle, Kunde, etc.)
3. Drill-down: Pro Report zu Details
4. Export in Excel, PDF, CSV
5. Scheduled Reports: Automatisch täglich/wöchentlich versenden

Betroffene Felder: MSPCollectorData (kompletter History), Mapping-Tabellen

Auswirkungen:

- Transparenz über MSP-Portfolio
- Management-Reporting automatisieren
- Abweichungen schnell erkennen
- Compliance-Dokumentation

6.6.5 Import-Fehlerbehandlung und Validierung

Zweck: Fehler beim Import identifizieren und korrigieren

Ablauf:

1. Während Import führt System Validierungen durch:
   - Pflichtfelder vorhanden?
   - Datentypen korrekt?
   - Duplikate zu existierenden Daten?
   - Werte in zulässigen Bereichen?
2. Fehler werden gesammelt und angezeigt
3. Benutzer kann Fehler korrigieren (inline-Editing) und Import erneut versuchen
4. Fehler-Report: was ist schiefgegangen, wie zu beheben?
5. Validierungsregeln können konfiguriert werden
6. Rollback: Wenn zu viele Fehler, Import abbrechen

Betroffene Felder: Alle MSPCollectorData-Felder, Validation-Rules

Auswirkungen:

- Datenqualität sicherstellen
- Fehlerhafte Prozesse aufdecken
- Automation verbessern (Validierung früh fangen)
- Manuelle Nacharbeiten reduzieren

6.6.6 Datenquelle Konfiguration und Verbindung

Zweck: Verschiedene MSP-Quellen konfigurieren und verbinden

Ablauf:

1. Administrator konfiguriert Datenquellen: Microsoft CSP, AWS, etc.
2. Verbindungsparameter: API-Key, Tenant-ID, Credentials, Endpoint
3. Test-Verbindung: Prüfe ob Verbindung funktioniert
4. Schedule: Automatische Import-Häufigkeit (täglich, wöchentlich)
5. Fehler-Benachrichtigung: Wenn Auto-Import fehlschlägt, Mail an Admin
6. Verbindungs-Historie: Wann war letzte erfolgreiche Abfrage?
7. Pause/Deaktivieren einer Quelle möglich

Betroffene Felder: DataSourceConfiguration, ConnectionCredentials, ScheduleSettings

Auswirkungen:

- Automatisierte Importe möglich
- Manueller Aufwand reduzieren
- Fehler früher erkennen
- Verwaltung zentralisiert

6.7 MSP-Dashboard (MSP Dashboard)

Module Path: `src/centron/Centron.WPF.UI/Modules/Statistics/MspStatistics`
Controller: `MspDashboardAppModuleController`
ViewModel: `MspDashboardViewModel`
Category: Controlling/Analytics
Description: Dashboard für Auswertung von MSP-Leistungsbausteinen mit Drill-Down-Analysen
License: LicenseGuids.Centron oder LicenseGuids.ManagementInfo

Use Cases

6.7.1 MSP-Leistungs-Dashboard öffnen

Zweck: Geschäftsführung erhält Überblick über MSP-Geschäftsbereich

Ablauf:

1. Manager öffnet MSP-Dashboard
2. Zeigt Top-Level KPIs:
   - MSP-Gesamtumsatz diesen Monat (€ und Trend)
   - Anzahl aktive MSP-Lizenzen/Services
   - Durchschnittliche Lizenz-Kosten
   - Customer-Anzahl mit MSP-Services
   - Ø-Marge MSP-Business
3. Vergleich mit Budget und Vormonat
4. Trend-Charts: 12-Monats-Entwicklung

5. Auto-Refresh alle 10 Minuten

Betroffene Felder: MSPStatistics (aggregierte Daten), Contract (MSP-Services), MSPLicenseData

Auswirkungen:

- MSP-Business schnell überblicken
- Leistung regelmäßig überwachen
- Trends erkennen (wachsend/schrumpfend)
- Entscheidungen datengestützt treffen

6.7.2 Drill-Down nach Kunden

Zweck: Details zu einzelnen MSP-Kunden sehen

Ablauf:

1. Benutzer klickt auf "Top-Kunden" Sektion des Dashboards
2. Zeigt Top-20 Kunden nach MSP-Umsatz
3. Pro Kunde: Name, Lizenzanzahl, monatlicher Umsatz, Marge, Trend
4. Klick auf Kunde → Detailansicht:
   - Services/Lizenzen die dieser Kunde nutzt
   - Lizenz-Details: Typ, Quantität, Kosten, Vertragsdauer
   - Zahlungshistorie: Pünktliche Bezahlungen? Ausstände?
   - Service-Level: Verfügbarkeit, Incidents
5. Kontakt zum Kunden: Tel, Email, Ansprechpartner
6. Export als Kundenbericht

Betroffene Felder: Customer, Contract (MSP-Verträge), MSPLicenseData, InvoiceHistory

Auswirkungen:

- Kundenverhältnisse verstehen
- Problem-Kunden schnell identifizieren
- Up-Sell-Chancen erkennen
- Kundenservice verbessern

6.7.3 Service-Performance und Verfügbarkeit

Zweck: Wie gut funktioniert unser MSP-Service?

Ablauf:

1. Dashboard zeigt Service-Performance Sektion:
   - Gesamt-Verfügbarkeit % (Target 99.9%)
   - Incident-Count diese Woche
   - MTTR (Mean Time To Resolve) Durchschnitt
   - Customer-Zufriedenheit (Ø Rating)
2. Trend über 12 Monate: Wird Service besser oder schlechter?
3. Vergleich: Unsere Performance vs. SLA-Vereinbarung
4. Incident-Top-List: Was sind häufigste Probleme?
5. Alarm: Wenn Verfügbarkeit unter 99%, Warnung auslösen
6. Export für Kundenreports

Betroffene Felder: ServiceLevelAgreement, IncidentLog, PerformanceMetrics

Auswirkungen:

- Service-Qualität kontrollieren

- Compliance mit SLA überwachen
- Geldstrafen durch Verfehlung vermeiden
- Kunden-Vertrauen bewahren

6.7.4 Profitabilität nach Service-Typ

Zweck: Welche MSP-Services sind rentabel?

Ablauf:

1. Dashboard zeigt Profitabilität pro Service-Kategorie
2. Spalten: Service-Typ, Umsatz, Kosten, Gewinn, Marge %
3. Farbcodierung: Grün >25% Marge, Gelb 15-25%, Rot <15%
4. Trend-Chart: Rentabilität über 12 Monate
5. Top-3 Services (nach Rentabilität, nicht Umsatz)
6. Bottom-3 Services: Problematische Services identifizieren
7. Empfehlung: "Service X ist unrentabel, Preis anheben oder einstellen?"

Betroffene Felder: Service (Type, Name), Contract (Price), Cost-Calculation, Revenue

Auswirkungen:

- Unrentable Services erkennen
- Preisanpassungen begründen
- Portfolio-Mix optimieren
- Profitabilität steigern

6.7.5 Wachstums-Analyse und Prognose

Zweck: Wächst das MSP-Business? Wie sieht die Zukunft aus?

Ablauf:

1. System zeigt Wachstums-Metriken:
   - Umsatz-Wachstum % vs. Vorjahr
   - Neue Kunden pro Monat (Trend)
   - Churn-Rate: % Kunden die gehen
   - Net-New-ARR (Annual Recurring Revenue Wachstum)
2. Trend-Extrapolation: Wenn Trend anhält, wie wird Business in 6/12 Monaten?
3. Szenarien: Best-Case, Worst-Case, Most-Likely
4. Wachstums-Driver analysieren: Was treibt Wachstum (neue Services, bestehende Kunden upsell)?
5. Benchmark: Wie vergleichen wir mit Branche?
6. Empfehlungen zur Beschleunigung von Wachstum

Betroffene Felder: Revenue (Time-Series), Customer-Anzahl, Contract-Daten, Forecast-Model

Auswirkungen:

- Strategische Planung datengestützt
- Investitionsentscheidungen treffen
- Personalplanung begründen
- Geschäftsziele setzen

6.7.6 Ressourcen-Auslastung im MSP-Team

Zweck: Ist das MSP-Support-Team angemessen ausgelastet?

Ablauf:

1. Dashboard zeigt Team-Auslastung:
   - Tickets pro Support-Mitarbeiter
   - Ø Bearbeitungszeit
   - Anzahl offene Tickets
   - Überstunden-Quote
2. Pro Mitarbeiter: Auslastung % (Grün <80%, Rot >100%)
3. Forecast: Wenn aktuelle Ticket-Rate bleibt, wie viele Mitarbeiter fehlen?
4. Warnung: Wenn MTTR steigt (Zeichen von Überlastung)
5. Trend: War Team immer überbelastet?
6. Empfehlungen: Neue Mitarbeiter, Automatisierung, Outsourcing?

Betroffene Felder: Employee (MSP-Support), Workload, Ticket-History

Auswirkungen:

- Personalentscheidungen treffen (Einstellung/Reduzierung)
- Service-Qualität durch Entlastung verbessern
- Kosten optimieren
- Mitarbeiterzufriedenheit verbessern

6.7.7 Lizenz-Optimierungs-Opportunities

Zweck: Wo können wir Kosten sparen, Margen verbessern?

Ablauf:

1. System scannt nach Optimierungs-Opportunities:
   - Kunden mit hoher Lizenzquote: Sind zu viele Lizenzen aktiv?
   - Underutilized Licenses: Gekauft aber nicht genutzt
   - Alte Verträge: Kann Preis neu verhandelt werden?
   - Bundle-Opportunities: Mehrere Lizenzen zu Paket kombinieren?
2. Pro Opportunity: Einspar-Potenzial € berechnen
3. Priorisierung: Welche bringen am meisten Ertrag?
4. Action-Items: Was konkret müssen wir tun?
5. Tracking: Welche Opportunities wurden bereits umgesetzt?

Betroffene Felder: Contract, MSPLicenseData, UserLicenseAssignment, Cost-History

Auswirkungen:

- Kosteneffizienz verbessern
- Margen erhöhen
- Kunden-Verhandlungen datengestützt
- Profitabilität maximieren

6.7.8 Alerts und Eskalationen

Zweck: Automatisch benachrichtigt über kritische MSP-Ereignisse

Ablauf:

1. System definiert Eskalations-Regeln:
   - Service-Verfügbarkeit <95%? KRITISCH
   - Incident-Response >4h? WARNUNG
   - Customer-Zahlungsausfallrisiko >30 Tage? WARNUNG
   - Umsatz-Abweichung >20% vs. Budget? WARNUNG
   - Employee-Turnover-Risk erkannt? WARNUNG

2. Alerts werden Dashboard-prominent angezeigt
3. Automatische E-Mail/SMS an Manager
4. Escalation: Wenn nicht innerhalb X Stunden adressiert, höhere Ebene benachrichtigen
5. Alert-History: Was waren die Alert-Trends?

Betroffene Felder: Alle MSP-Daten, SLA, Incident, Customer, Employee

Auswirkungen:

- Kritische Probleme früh erkennen
- Proaktives Management möglich
- Kundenprobleme schneller lösen
- Geschäftsrisiken minimieren

6.8 Vertragsauswertung (Contract Evaluation)

Module Path: `src/centron/Centron.WPF.UI/Modules/Finances/ContractEvaluation2`
Controller: `ContractEvaluation2AppModuleController`
ViewModel: `ContractEvaluation2ViewModel`
Category: Controlling/Analytics
Description: Vertragsauswertung und automatisierte Abrechnungsanalyse
License: LicenseGuids.Centron oder LicenseGuids.Abrechnung

Use Cases

6.8.1 Vertrags-Status-Übersicht

Zweck: Alle Verträge im Überblick, Status und Daten

Ablauf:

1. Benutzer öffnet Vertragsauswertung
2. Zeigt Tabelle mit allen Verträgen:
   - Vertragsnummer, Kunde, Service/Produkt, Laufzeit (von-bis), Status (aktiv/expired/pending)
   - Monatliche Gebühren (€), Abrechnung-Rhythmus
   - Nächste Abrechnung Datum, letzte Abrechnung Datum
   - Automatische Abrechnung aktiviert? Ja/Nein
3. Farbe-Codierung: Grün (aktiv, korrekt), Gelb (bald auslaufend), Rot (abgelaufen/Fehler)
4. Filter: nach Status, Kunde, Abrechnungstyp
5. Klick auf Vertrag → Details anzeigen

Betroffene Felder: Contract (ContractNumber, CustomerI3D, StartDate, EndDate, Status), InvoiceSchedule

Auswirkungen:

- Vertrags-Portfolio übersehen
- Ablauf-Termine nicht vergessen
- Fehlerhafte Verträge schnell finden
- Abrechnung prüfen

6.8.2 Automatische Abrechnungs-Kontrolle

Zweck: Wurden alle Verträge korrekt abgerechnet?

Ablauf:

1. System vergleicht: Verträge vs. tatsächliche Rechnungen

2. Pro Vertrag: Wann sollte nächste Rechnung gestellt werden? Wurde sie gestellt?
3. Warnung bei fehlenden Rechnungen: "Vertrag XYZ: Rechnung Mai fehlt!"
4. Vergleicht: Rechnungsbetrag vs. Vertragsbetrag (stimmt es überein?)
5. Warnung bei Abweichungen: "Rechnung zu niedrig!" oder "Rechnung zu hoch!"
6. Zeigt Fehler-Häufigkeit: Welche Verträge werden immer falsch abgerechnet?
7. Empfehlung: Automatische Abrechnung für korrekte Verträge aktivieren

Betroffene Felder: Contract, AutomaticFacturation, Invoice (Amount, InvoiceDate), InvoiceSchedule

Auswirkungen:

- Abrechnungsfehler reduzieren
- Revenue nicht verlieren
- Kunden-Vertrauen bewahren
- Automatisierung verbessern

6.8.3 Vertrags-Abrechnungs-Analyse

Zweck: Einnahmen-Prognose basierend auf Verträgen

Ablauf:

1. System berechnet: Prognostizierte monatliche Einnahmen aus allen Verträgen
2. Basis: Verträge × Laufzeit × Gebühr
3. Berücksichtigt: Bereits auslaufende Verträge (Churn)
4. Zeigt: Prognose für nächste 3/6/12 Monate
5. Vergleicht mit tatsächlichen Einnahmen (Soll vs. Ist)
6. Identifiziert Anomalien: "Warum sind Einnahmen Mai höher als Prognose?"
7. Forecast-Genauigkeit: Wie gut waren bisherige Prognosen?

Betroffene Felder: Contract (Amount, StartDate, EndDate, Status), Invoice (Amount, InvoiceDate), Forecast-Model

Auswirkungen:

- Revenue-Planung datengestützt
- Überraschungen beim Cash-Flow vermeiden
- Finanzielle Planung verbessern
- Geschäftsrisiken identifizieren

6.8.4 Vertrags-Laufzeitverwaltung und Renewals

Zweck: Wann laufen Verträge aus und müssen erneuert werden?

Ablauf:

1. Dashboard zeigt "Renewal Calendar":
   - Verträge sortiert nach Ablaufdatum
   - Warnung ab 90 Tage vor Ablauf: "Vertrag läuft in 3 Monaten aus, jetzt neu verhandeln"
   - Warnung ab 30 Tage: "Vertrag läuft in 1 Monat aus, Renewal dringend!"
2. Pro ablauffähigem Vertrag: Kunde, Wert, Renewal-Wahrscheinlichkeit (Prognose)
3. Action-Items: "Kontaktieren Sie Kunde X für Vertragsverlängerung"
4. Historisch: Welche Verträge wurden nicht erneuert? (Churn-Analyse)
5. Empfehlung: Früh mit Renewal starten (90 Tage vorher)

Betroffene Felder: Contract (EndDate, Status), RenewalHistory, Customer (Status)

Auswirkungen:

- Kundenabgang verhindern (durch rechtzeitige Kontakte)

- Umsatz-Sicherung
- Churn minimieren
- Kundenbeziehungen pflegen

6.8.5 Vertrags-Abweichungs-Analyse

Zweck: Finden von Verträgen die nicht richtig funktionieren

Ablauf:

1. System vergleicht: Vertragsbedingte Leistung vs. tatsächliche Leistung
2. Beispiele von Abweichungen:
   - Verträge: 24/7 Support, aber nur Mo-Fr? ABWEICHUNG
   - Verträge: 99.9% Verfügbarkeit, aber nur 97% erreicht? ABWEICHUNG
   - Verträge: 1000 Stunden p.a., aber 1500 verbraucht? ABWEICHUNG
3. Pro Abweichung: Auswirkung auf Gewinn (Strafzahlungen, Reputationsverlust)
4. Priorität: Welche Abweichungen sind kritisch?
5. Maßnahmen: "Entweder Service verbessern oder Vertrag anpassen"
6. Tracking: Wurden Maßnahmen implementiert?

Betroffene Felder: Contract (Terms, Duration, Commitments), ServiceLevelAgreement, Performance-Metrics

Auswirkungen:

- Service-Lieferung verbessern
- Geldstrafen und Reputationsschäden vermeiden
- Vertragsanpassungen rechtzeitig einleiten
- Kundenvertrauen bewahren

6.8.6 Kundenspezifische Verträge und Multi-Contract-Management

Zweck: Ein Kunde kann mehrere Verträge haben; alle zusammen sehen

Ablauf:

1. Filter nach Kunde
2. Zeigt: Alle Verträge des Kunden, Portfolio-Wert insgesamt
3. Pro Vertrag: Service, Gebühr, Laufzeit, Status
4. Cross-Vertrag-Analysen:
   - Redundanzen: Bezahlt Kunde 2x für gleichen Service?
   - Bundle-Opportunity: Mehrere Verträge zu 1 Paket kombinieren (mit Rabatt)?
   - Upsell-Potential: Welche Services nutzt dieser Kunde noch nicht?
5. Customer-Value: Wie wichtig ist dieser Kunde? (nach Gesamtvolumen)
6. Churn-Risk: Wie wahrscheinlich ist Kundenabgang?
7. Report: Customer-Portfolio für Account-Manager

Betroffene Felder: Customer, Contract (Multiple per Customer), Invoice, ServiceUsage

Auswirkungen:

- Upsell und Cross-Sell Chancen erkennen
- Kunden-Lifetime-Value maximieren
- Kundenrisiken identifizieren
- Account-Management datengestützt

6.8.7 Gewinn-Optimierung und Preis-Analyse

Zweck: Sind Verträge rentabel? Können Preise angepasst werden?

Ablauf:

1. System berechnet pro Vertrag: Gewinn = Einnahmen - Kosten
2. Kostenaufschlüsselung: Direkte Kosten (Server, Support), indirekte Kosten (Overhead)
3. Gewinnmarge % pro Vertrag
4. Ranking: Top-20 profitabelste Verträge, Top-20 unrentabelste
5. Gefährliche Verträge: <5% Marge oder negativ?
6. Empfehlungen: Preiserhöhung bei Renewal, Kosten senken, oder Vertrag kündigen?
7. Szenario: "Wenn wir Preis um 10% erhöhen, wie viele Kunden kündigen?"

Betroffene Felder: Contract (Price, Duration), Cost-Calculation, Invoice, Facturation, Profit-Margin

Auswirkungen:

- Rentabilität steigern
- Unrentable Verträge sanieren
- Preissetzung datengestützt
- Margenoptimierung

6.8.8 Abrechnungs-Automatisierung und Fehlerfreie Fakturierung

Zweck: Verträge komplett automatisiert abrechnen

Ablauf:

1. Administrator konfiguriert Abrechnungs-Regeln pro Vertrag:
   - Abrechnungsrhythmus (monatlich, quarterly, yearly)
   - Betrag und Bedingungen
   - Invoice-Template und Empfänger
2. System führt automatische monatliche Abrechnung durch
3. Prüfung vor Rechnungstellung: Stimmen Daten mit Vertrag überein?
4. Automatische Rechnungs-Generierung und Versand an Kunde
5. Fehler-Handling: Wenn Abrechnung scheitert, Admin benachrichtigen
6. Zahlung-Matching: Automatisch vergleichen, ob Zahlung angekommen
7. Reporting: Abrechnung-Fehlerquote, erfolgreiche Fakturierungen

Betroffene Felder: Contract, AutomaticFacturation-Config, Invoice (Template), Payment-Matching

Auswirkungen:

- Manuelle Abrechnungsarbeit eliminieren
- Abrechnungsfehler reduzieren
- Cash-Flow verbessern (schnellere Rechnungen)
- Verwaltungskosten sparen
- Prozess-Standardisierung

7. Einkauf (Purchasing)

7.1 Belegerfassung (Document Capture)

Module Path: `src/centron/Centron.WPF.UI/Modules/Finances/Receipts`
Controller: `ReceiptAppModuleController`
ViewModel: `SupplierReceiptViewModel`
Category: Einkauf
Description: Erfassung und Verarbeitung von Lieferantenbelegungen (Rechnungen, Lieferscheine, Gutschriften) mit optischer Zeichenerkennung und automatischer Datenextraktion

Use Cases

7.1.1 Lieferantenbeleg scannen und einlesen

Zweck: Digitalisierung von papiergestützten Lieferantendokumenten mit automatischer Datenextraktion

Ablauf:

1. Benutzer navigiert zu "Neuer Beleg"
2. Scanner wird ausgewählt (Multi-Page Scanner möglich)
3. Belege werden gescannt: Rechnungen, Lieferscheine, Begleitzettel
4. System führt optische Zeichenerkennung (OCR) durch
5. Automatische Datenextraktion: Lieferantennummer, Rechnungsnummer, Betrag, Datum
6. Benutzer kann Qualität des Scans überprüfen
7. Extrakte werden vorausgefüllt in Eingabeformular
8. Beleg wird als Attachment gespeichert

Betroffene Felder: SupplierReceipt (ScanImagePath, OCRExtractedData, SupplierI3D, InvoiceNumber, Amount, InvoiceDate)

Auswirkungen: Manuelle Dateneingabe reduziert sich um 80%. Fehlerquote sinkt. Verarbeitungszeit pro Beleg sinkt von Minuten auf Sekunden. Digitale Archive ersetzen Papier.

7.1.2 Automatisch extrahierte Daten validieren und korrigieren

Zweck: Überprüfung und Korrektur von durch OCR extrahierten Daten

Ablauf:

1. System zeigt Scan-Vorschau mit extrahierten Feldern
2. Benutzer validiert automatisch ausgefüllte Felder: Lieferant, Rechnungsnummer, Datum, Betrag
3. Bei Abweichungen: Benutzer kann Wert manuell korrigieren
4. System markiert korrigierte Felder
5. Vertrauens-Score wird angezeigt für jeden Extrakt
6. Bei niedriger Konfidenz: Benutzer wird zur manuellen Eingabe aufgefordert
7. Validierte Daten werden gespeichert
8. Korrekturen werden für ML-Training des OCR-Systems genutzt

Betroffene Felder: SupplierReceipt (ValidatedByI3D, ValidationDate, CorrectedFields, OCRConfidenceScore)

Auswirkungen: Datenqualität steigt. ML-Modell verbessert sich mit Zeit. Benutzer-Korrektionen führen zu besseren zukünftigen Extraktionen.

7.1.3 Beleg mit Bestellung abgleichen

Zweck: Automatische Verknüpfung von Lieferantenbelegen mit bestehenden Bestellungen

Ablauf:

1. Benutzer öffnet gescannten Beleg
2. System sucht automatisch nach passender Bestellung durch Lieferantennummer + Rechnungsnummer
3. Wenn Bestellung gefunden: System zeigt Match mit Bestätigungs-Option
4. Benutzer kann Match bestätigen oder manuell nach Bestellung suchen
5. System vergleicht automatisch: Bestellmenge vs. Rechnungsmenge, Bestellpreis vs. Rechnungspreis
6. Abweichungen werden gekennzeichnet (z.B. Überlieferung, Preisdifferenz)
7. Beleg wird mit Bestellung verlinkt
8. Abweichungen triggern Genehmigungsprozesse

Betroffene Felder: SupplierReceipt (SupplierOrderI3D), PurchaseOrder (LinkedInvoiceI3D), ReceiptVariance (VarianceType, Amount)

Auswirkungen: 3-Way-Match (PO, Rechnung, Eingang) wird automatisiert. Doppelzahlungen werden vermieden. Rechnungsfreigabe beschleunigt sich. Lieferantenrabatte werden automatisch appliziert.

7.1.4 Rechnungen mit automatischer Kategorisierung

Zweck: Automatische Zuordnung von Rechnungspositionen zu Kostenstellen und Konten

Ablauf:

1. Benutzer öffnet gescannten Beleg
2. System zeigt Rechnungspositionen (Artikelnummer, Beschreibung, Menge, Preis)
3. System versucht automatisch Kategorisierung:
   - Artikel-Codes werden mit Lagerkatalog abgeglichen
   - Serviceleistungen werden identifiziert
   - Kostenstellen werden vorgeschlagen
4. Benutzer kann Kategorisierung validieren oder anpassen
5. GL-Accounts werden automatisch zugeordnet (basierend auf Artikel-Klasse)
6. Beleg ist nun bereit für Rechnungsfreigabe
7. Änderungen werden dokumentiert

Betroffene Felder: ReceiptItem (ArticleI3D, CostCenterI3D, GLAccountI3D, CategoryI3D), InvoiceLineItem (ExpenseCategory)

Auswirkungen: Rechnungsverarbeitung wird beschleunigt. Automatische Buchungen möglich. Fehlerquote bei Kategorisierung sinkt. Compliance-Report-Genauigkeit steigt.

7.1.5 Duplikate erkennen und zusammenführen

Zweck: Vermeidung von doppelten Rechnungsbuchungen durch Duplikat-Erkennung

Ablauf:

1. Wenn neuer Beleg gescannt wird: System prüft auf Duplikate
2. Duplikat-Check basiert auf: Lieferant + Rechnungsnummer + Betrag
3. Bei verdächtigem Duplikat: System zeigt Warnung mit Vorschlag
4. Benutzer kann anzeigen: "Ist Duplikat" oder "Neuer Beleg"
5. Bei Duplikat: Älterer Beleg wird als Duplikat markiert (nicht gelöscht)
6. System prüft ob bereits bezahlt:
   - Wenn bezahlt: Warnung an Kreditorenbuchhaltung
   - Zahlung wird ggf. storniert
7. Gültige Beleg wird behalten

Betroffene Felder: SupplierReceipt (IsDuplicate, DuplicateOfI3D, FlaggedByI3D, FlaggedDate)

Auswirkungen: Doppelzahlungen werden verhindert. Audit-Trail dokumentiert Duplikat-Erkennung. Zahlungsgenauigkeit steigt.

7.1.6 Beleg-Workflow und Genehmigung

Zweck: Strukturierte Freigabe von Lieferantenbelegungen mit Genehmigungskette

Ablauf:

1. Benutzer öffnet erfassten Beleg
2. Beleg zeigt Status: Erfasst → Validiert → Abgeglichen → Genehmigt → Gebucht
3. Benutzer kann Beleg zur Genehmigung freigeben
4. System zeigt Genehmigungs-Kette: Sachbearbeiter → Abteilungsleiter → Controller
5. Genehmiger sieht: Beleg-Scan, Extrahierte Daten, Abweichungen
6. Genehmiger kann genehmigen oder mit Kommentar zurückweisen
7. Nach finaler Genehmigung: Beleg wird automatisch in Buchhaltung gebucht
8. Zahlungserinnerung wird generiert (je nach Zahlungsbedingungen)

Betroffene Felder: SupplierReceipt (Status, ApprovedByI3D, ApprovedDate), ReceiptApprovalHistory (ApprovalStep, ApprovedByI3D, ApprovedDate, Comments)

Auswirkungen: Compliance-konforme Genehmigung. Audit-Trail vollständig. Rechnungsfreigabe-Zeit sinkt. Fehler-Risiken sinken durch mehrschichtige Kontrolle.

7.2 Bestellvorschlagsliste (Purchase Order Suggestions)

Module Path: `src/centron/Centron.WPF.UI/Modules/Purchasing/OrderSuggestionList`
Controller: `OrderSuggestionListAppModuleController`
ViewModel: `OrderSuggestionListViewModel`
Category: Einkauf
Description: Automatische Generierung von Bestellvorschlägen basierend auf Lagerbeständen, Verbrauch und Lieferantenkonditionen

Use Cases

7.2.1 Bestellvorschläge automatisch generieren

Zweck: Automatisierte Berechnung von optimalen Bestellmengen basierend auf Lagerkennzahlen

Ablauf:

1. Benutzer navigiert zu "Bestellvorschläge"
2. System zeigt Button "Neue Vorschläge berechnen"
3. Benutzer wählt Parameter: Artikel-Filter, Lieferanten, Lagerort, Berechnung-Methode
4. System berechnet für jede Artikel-Lieferant-Kombination:
   - Aktueller Lagerbestand
   - Durchschnittlicher Verbrauch der letzten Monate
   - Wiederbeschaffungszeit (Lead Time)
   - Sicherheitsbestand
   - Mindestbestellmenge
5. System berechnet optimale Bestellmenge (z.B. mit Economic Order Quantity)
6. Bestellvorschläge werden generiert und angezeigt
7. Benutzer kann Filter anwenden (nur kritische Artikel, etc.)

Betroffene Felder: SuggestionOrder (ArticleI3D, DistributorI3D, SuggestedQuantity, CalculationBasis, LeadTime, SafetyStock)

Auswirkungen: Manuelle Bestellplanung wird überflüssig. Lagerbestände optimieren sich. Kapitalbindung sinkt. Lieferverzögerungen werden verhindert durch automatische Lead-Time-Berechnung.

7.2.2 Bestellvorschläge prüfen und anpassen

Zweck: Überprüfung und manuelle Anpassung der automatisch generierten Vorschläge

Ablauf:

1. Benutzer öffnet generierte Bestellvorschläge in Listenansicht
2. System zeigt Gitterview mit: Artikel, Aktueller Bestand, Verbrauch, Lieferant, Vorschlag-Menge, Betrag
3. Benutzer kann Vorschlag validieren oder anpassen:
   - Menge erhöhen/senken
   - Lieferanten wechseln
   - Artikel aus Liste entfernen
4. System berechnet Auswirkungen: Lagerkosten, Lagerdauer, Finanzierungsbedarf
5. Benutzer kann Rabatte/Skonti manuell berücksichtigen
6. Warnung bei ungewöhnlich großen Abweichungen vom System-Vorschlag
7. Angepasste Vorschläge werden gespeichert

Betroffene Felder: SuggestionOrder (AdjustedQuantity, AdjustedByI3D, AdjustmentReason, Status="Reviewed")

Auswirkungen: Benutzer-Expertise wird genutzt. Besondere Situationen (z.B. geplante Kampagnen) werden berücksichtigt. Bestellgenauigkeit verbessert sich.

7.2.3 Bestellvorschläge in echte Bestellungen umwandeln

Zweck: Konvertierung genehmigter Bestellvorschläge in verbindliche Bestellungen

Ablauf:

1. Benutzer wählt mehrere Bestellvorschläge aus der Liste (oder alle)
2. Button "In Bestellung umwandeln" wird geklickt
3. System validiert:
   - Lieferant verfügbar?
   - Artikel verfügbar?
   - Budget ausreichend?
4. Für jeden Lieferanten wird eine Bestellung erstellt
5. Bestellpositionen werden automatisch mit Lieferant-Konditionen gefüllt
6. Lieferdatum wird berechnet basierend auf Lead-Time
7. Bestellungen werden als Draft erstellt
8. Benutzer kann Bestellungen anschauen vor Freigabe
9. Nach Genehmigung: Bestellungen werden Lieferant zugesendet

Betroffene Felder: SuggestionOrder (Status="Ordered", ConvertedToPOI3D), PurchaseOrder (CreatedFromSuggestionI3D)

Auswirkungen: Bestellungserstellung beschleunigt sich von Stunden auf Minuten. Lieferdaten werden realistisch geplant. Bestellgenauigkeit verbessert sich.

7.2.4 Lieferanten und Preismatrix verwenden

Zweck: Automatische Auswahl optimaler Lieferanten und Preise

Ablauf:

1. System hat Preismatrix für jeden Artikel-Lieferant
2. Preismatrix enthält: Mengen-Rabatte, Lieferzeitabhängige Preise, saisonale Preise
3. Bei Bestellvorschlag-Berechnung wird beste Kombination gewählt:
   - Lieferant mit bester Gesamtkostenposition
   - Lieferant berücksichtigung: Preis, Rabatt, Lieferdauer, Zuverlässigkeit
4. Alternative Lieferanten werden angezeigt
5. Benutzer kann Lieferanten manuell wechseln

6. Vorzugslieferanten können konfiguriert werden
7. Lieferanten-Switching wird dokumentiert

Betroffene Felder: SuggestionDistributor (Price, LeadTime, Rating, DiscountMatrix), PriceMatrix (Quantity, Price, ValidFrom, ValidTo)

Auswirkungen: Einkaufspreise werden optimiert. Lieferanten-Konkurrenz wird genutzt. TCO (Total Cost of Ownership) wird minimiert. Verhandlungsposition mit Lieferanten verbessert sich.

7.2.5 Bestellvorschläge-Report und Analytics

Zweck: Analyse und Reporting der Bestellvorschläge für Management-Entscheidungen

Ablauf:

1. Benutzer öffnet Tab "Analysen & Reports"
2. System zeigt verschiedene Report-Vorlagen:
   - Bestellvolumen nach Lieferant
   - Kostenentwicklung durch Bestellvorschläge
   - Lagerbestandsoptimierung
   - Lieferanten-Diversifizierung
3. Benutzer kann Report-Parameter setzen (Zeitraum, Artikel-Filter, etc.)
4. Reports werden als Diagramme und Tabellen angezeigt
5. Trends werden berechnet: z.B. "Bestellvolumen steigt um 15%"
6. Benutzer kann Reports exportieren (Excel, PDF)
7. Scheduled Reports können automatisch per Email versendet werden

Betroffene Felder: ReportData (ReportType, Parameters, GeneratedDate), SuggestionOrder (Historical Data)

Auswirkungen: Strategische Einkaufs-Entscheidungen werden datengetrieben. Budget-Planung wird genauer. Lieferanten-Management wird optimiert.

7.2.6 Bestellvorschläge-Verwerfung und Archivierung

Zweck: Verwerfung ungültiger Vorschläge und Archivierung älterer Bestellvorschläge

Ablauf:

1. Benutzer markiert Bestellvorschlag als "Verwerfen"
2. Grund kann eingegeben werden: z.B. "Artikel veraltet", "Lieferant nicht verfügbar"
3. Verworfene Vorschläge werden in separate Ansicht verschoben
4. System archiviert abgelaufene Vorschläge automatisch nach 30 Tagen
5. Benutzer kann archivierte Vorschläge noch anschauen (Read-Only)
6. Report zeigt Verwerfungs-Quote und -Gründe
7. Häufige Verwerfungs-Gründe werden analysiert (z.B. Lieferant häufig nicht verfügbar)

Betroffene Felder: SuggestionOrder (Status="Rejected"/"Archived", RejectionReason, RejectedByI3D, RejectedDate)

Auswirkungen: System wird nicht überlastet mit alten Vorschlägen. Verwerfungs-Gründe informieren System-Verbesserungen. Audit-Trail bleibt erhalten.

7.3 EDI Verwaltung (EDI Management)

Module Path: `src/centron/Centron.WPF.UI/Modules/Purchasing/EDIManagement`
Controller: `EDIManagementController`
ViewModel: `EDIManagementViewModel`
Category: Einkauf

Description: Verarbeitung von elektronischen Geschäftsdokumenten (EDI) mit Unterstützung für OpenTrans, ZUGFeRD und Lieferanten-spezifische EDI-Formate (ALSO, Alltron, Komsa, etc.)

Use Cases

7.3.1 EDI-Beleg empfangen und importieren

Zweck: Automatische Verarbeitung von elektronischen Lieferantendokumenten im EDI-Format

Ablauf:

1. EDI-Datei wird von Lieferant empfangen (FTP, Email, Portal)
2. System erkennt EDI-Format automatisch (OpenTrans, ZUGFeRD, proprietär)
3. EDI-Parser validiert Dateistruktur
4. Datei wird in EDI-Datenbank importiert
5. Sender-Authentizität wird validiert (digitale Signatur wenn vorhanden)
6. EDI-Beleg wird mit entsprechender Bestellung abgeglichen
7. System zeigt Import-Status: Erfolgreich, Teilweise, Fehler
8. Import-Log wird dokumentiert für Audit

Betroffene Felder: EDIReceiptHead (FileName, Format, SupplierI3D, ImportDate, ImportedByI3D), EDILog (ImportStatus, ErrorMessages)

Auswirkungen: Manuelles Kopieren entfällt. Fehlerquoten sinken dramatisch. Lieferantendokumente sind sofort verarbeitet. Automatische Prüfung auf Plausibilität.

7.3.2 EDI-Rechnungen verarbeiten und buchen

Zweck: Vollautomatische Verarbeitung von EDI-Rechnungen mit Buchung in Buchhaltung

Ablauf:

1. System zeigt importierte EDI-Rechnungen in Warteschlange
2. System führt 3-Way-Match durch: Bestellung → EDI-Rechnung → Wareneingang
3. EDI-Rechnungsdaten werden extrahiert: Rechnungsnummer, Betrag, Steuern, Zahlungsbedingungen
4. System prüft auf Abweichungen (Menge, Preis, Steuern)
5. Bei Abweichungen < konfiguriertes Limit: Automatische Freigabe
6. Bei größeren Abweichungen: Zur manuellen Überprüfung gekennzeichnet
7. Gültige Rechnungen werden automatisch in Buchhaltung gebucht
8. Zahlung wird terminiert basierend auf Zahlungsbedingungen

Betroffene Felder: EDIInvoice (Amount, TaxAmount, PaymentTerms, Status), EDIReceiptItems (LinkedInvoiceI3D), GLJournal (BookedFromEDI)

Auswirkungen: Rechnungsverarbeitung wird 100% automatisiert. Manuelle Dateneingabe entfällt. Fehlerquote = 0%. Bezahlungsgenauigkeit = 100%. Buchungszyklus verkürzt sich von Tagen auf Stunden.

7.3.3 EDI-Bestellbestätigungen verarbeiten

Zweck: Automatische Verarbeitung von Lieferanten-Bestellbestätigungen (Order Responses)

Ablauf:

1. Lieferant sendet EDI-Bestellbestätigung (Order Response)
2. System empfängt und validiert Bestellbestätigung
3. EDI-Parser extrahiert: Bestellnummer, bestätigte Menge, Lieferdatum, Artikel-Details
4. System gleicht mit gesendeter Bestellung ab
5. Abweichungen werden geprüft:
   - Teilbestätigung (Menge < Bestellung)?
   - Verzögertes Lieferdatum?
   - Geänderte Artikel?

6. Bei Abweichungen: Alert an Einkaufsmanagement
7. Bestellstatus wird aktualisiert: Von "Bestellt" zu "Bestätigt"
8. Lieferdatum wird in Verfügbarkeitsplanung berücksichtigt

Betroffene Felder: EDIOrderResponse (PurchaseOrderI3D, ConfirmedQuantity, ConfirmedDeliveryDate, Deviations), PurchaseOrder (Status, ConfirmedDeliveryDate)

Auswirkungen: Bestätigungsquoten werden erhöht. Lieferplanung wird genauer. Überraschungslieferungen/Verspätungen werden früh erkannt. Automatische Benachrichtigungen an betroffene Abteilungen.

7.3.4 Lieferanten-spezifische EDI-Integrationen verwalten

Zweck: Verwaltung und Konfiguration von Lieferanten-spezifischen EDI-Formaten und Übertragungskanälen

Ablauf:

1. Admin öffnet EDI-Lieferanten-Verwaltung
2. System zeigt konfigurierte Lieferanten: ALSO, Alltron, Komsa, EGIS, etc.
3. Für jeden Lieferanten sind konfiguriert:
   - EDI-Format (OpenTrans, proprietär, etc.)
   - Übertragungskanal (FTP, Email, API)
   - Übertragungsplan (täglich, wöchentlich)
   - Authentifizierungsdaten (Benutzer, Zertifikat)
4. Admin kann neue Lieferanten hinzufügen
5. System testet Verbindung mit EDI-Gateway
6. Fehlerbehandlung wird konfiguriert (Retry, Notification, etc.)
7. EDI-Lieferanten werden automatisch aktualisiert

Betroffene Felder: EDISupplier (SupplierI3D, EDIFormat, ChannelType, TransmissionSchedule, AuthenticationMethod)

Auswirkungen: Multi-Supplier-EDI-Integration wird standardisiert. Neue Lieferanten können schnell integriert werden. Fehlerbehandlung ist konsistent. Compliance ist dokumentiert.

7.3.5 EDI-Fehler und Abweichungen handling

Zweck: Systematische Behandlung von EDI-Import-Fehlern und Datenabweichungen

Ablauf:

1. Bei EDI-Fehler (Parser-Fehler, Formatfehler, etc.):
   - System speichert fehlgeschlagene Datei
   - Fehler wird dokumentiert in EDI-Log
   - Admin wird benachrichtigt
2. System versucht automatisch Fehlerbehebung:
   - Validierung ignorieren und mit Warnings fortfahren
   - Alternative Parser versuchen
   - Mit letzter bekannter guter Version fortfahren
3. Benutzer kann fehlgeschlagene EDI-Dateien manuell prüfen
4. Benutzer kann Fehler dokumentieren und Korrektionen vornehmen
5. Nach Korrektion: Neuversuch des Imports
6. Fehler-Report zeigt Quote fehlgeschlagener Importe pro Lieferant
7. Chronische Fehler triggern Eskalation an Lieferanten

Betroffene Felder: EDILog (Status="Error", ErrorMessage, ErrorType, RetryCount), EDIErrorHandling (Configuration)

Auswirkungen: EDI-Fehlerquote wird minimiert. Automatische Fehlerbehandlung reduziert manuellen Aufwand. Lieferanten werden über Probleme informiert. System wird robuster.

7.3.6 EDI-Bestätigung und Reporting

Zweck: Generierung von EDI-Bestätigungen an Lieferanten und EDI-Performance-Reporting

Ablauf:

1. Nach erfolgreicher EDI-Verarbeitung:
   - System generiert Empfangsbestätigung (APERAK)
   - Bestätigung wird an Lieferant zurück gesendet
   - Format stimmt mit Lieferanten-Anforderungen überein
2. Benutzer kann EDI-Performance-Report anfordern
3. Report zeigt Metriken:
   - Anzahl Belege pro Tag/Lieferant
   - Erfolgsquote (% erfolgreich verarbeitet)
   - Durchschnittliche Verarbeitungszeit
   - Fehlertypen und -häufigkeit
4. Lieferanten-Vergleich (z.B. "ALSO: 99.5% Erfolgsquote vs. Alltron: 98.2%")
5. Report kann per Email versendet oder manuell exportiert werden
6. Trends werden visualisiert (Erfolgsquote verbessert sich über Zeit)

Betroffene Felder: EDIReceiptAcknowledgement (SentToSupplierI3D, SentDate, AcknowledgementFormat), EDIPerformanceMetrics (SupplierI3D, SuccessRate, AverageProcessingTime)

Auswirkungen: Lieferanten-Qualität wird transparent. SLA-Compliance kann überprüft werden. Management hat Datengrundlage für Lieferanten-Optimierung. EDI-ROI wird nachgewiesen.

7.4 Eingang/Kalk (Goods Receipt/Calculation)

Module Path: `src/centron/Centron.WPF.UI/Modules/Finances/Receipts`
Controller: `ReceiptAppModuleController`
ViewModel: `ReceiptViewModel`
Category: Einkauf
Description: Erfassung und Verarbeitung von Wareneingang, Preiskalkulation und Rechnungsbegleichung mit Unterstützung für verschiedene Belegtypen (Angebote, Bestellungen, Lieferscheine, Rechnungen)

Use Cases

7.4.1 Wareneingang erfassen

Zweck: Dokumentation des physischen Wareneingangs und Abgleich mit Bestellung

Ablauf:

1. Benutzer navigiert zu "Neuer Wareneingang"
2. System zeigt Bestellnummer-Eingabefeld
3. Benutzer gibt Bestellnummer ein oder scannt Barcode
4. System lädt bestellte Artikel automatisch
5. Benutzer scannt/erfasst eingegangene Artikel:
   - Barcode oder Artikelnummer
   - Eingegangene Menge
   - Lagerlocation
6. System vergleicht automatisch mit Bestellung:
   - Artikel ok?
   - Menge ok?
   - Qualität ok?
7. Abweichungen werden markiert (Überlieferung, Fehlmengen, Beschädigungen)
8. Wareneingang wird gebucht, Lagerbestand wird aktualisiert

9. Berichte an Bestellsystem werden aktualisiert

Betroffene Felder: ReceiptTable (Type="DeliveryList"), ReceiptItems (ArticleId3D, ReceivedQuantity, StorageLocationId3D), StockMovement (MovementType="Receipt")

Auswirkungen: Lagerbestände sind in Echtzeit aktuell. Abweichungen werden sofort erkannt. Nachbestellungen werden automatisch getriggert bei Unterlieferung. Bestellkette wird beschleunigt.

7.4.2 Preis-Kalkulation und Kostenübernahme

Zweck: Automatische Berechnung von Übernahmekosten und Preisanpassungen

Ablauf:

1. Benutzer öffnet Wareneingang
2. System zeigt Kostenaufschlüsselung:
   - Brutto-Materialkosten
   - Transport/Versand
   - Versicherung
   - Zölle/Importe (bei Import)
3. System kalkuliert automatisch Einstandspreis pro Artikel:
   - Brutto-Materialkosten + anteiliger Nebenkosten / Menge
4. Benutzer kann Abweichungen eingeben (z.B. Rabatt nachträglich gewährt)
5. Benutzer kann unterschiedliche Kalkulationsmethoden wählen:
   - Durchschnitt
   - FIFO (First In, First Out)
   - LIFO (Last In, First Out)
6. Kalkulierte Preise werden gespeichert
7. Lagerbewertung wird aktualisiert basierend auf Kalkulationsmethode
8. Finanzbuchung für Materialkosten wird erstellt

Betroffene Felder: ReceiptItems (GrossMaterialCost, IncidentalCosts, CalculatedUnitCost), StockValuation (ValuationMethod, AverageCost, CalculationDate)

Auswirkungen: Einstandspreise sind präzise. Lagerbewertung ist korrekt. Kostenrechnung wird genauer. Gewinnmargen werden realistisch berechnet.

7.4.3 Annahmeprüfung und Qualitätskontrolle

Zweck: Durchführung von Qualitätsprüfungen beim Wareneingang mit Dokumentation von Mängeln

Ablauf:

1. Nach Wareneingangerfassung wird Annahmeprüfung gestartet
2. Prüfer führt Sichtprüfung durch:
   - Verpackung ok?
   - Artikel sichtbar beschädigt?
   - Verfallsdatum ok?
3. Bei Auffälligkeiten wird Mängelbeleg erstellt:
   - Mangel-Beschreibung
   - Fotos
   - Severity (Kritisch, Major, Minor)
4. System erstellt automatisch:
   - RMA (Return Merchandise Authorization) Falls notwendig
   - Gutschrift-Vorschlag für Lieferant
5. Prüfer kann Artikel akzeptieren oder ablehnen
6. Abgelehnte Waren werden in Quarantäne verschoben
7. Qualitäts-Report wird dokumentiert

Betroffene Felder: ReceiptQualityInspection (Status="Passed"/"Failed", FindingDetails, SeverityLevel), RMA (CreatedFromReceiptI3D), ReceiptItems (QualityStatus)

Auswirkungen: Rücklieferungen werden schnell eingeleitet. Lieferanten-Qualität wird dokumentiert. Mängelquoten werden verfolgbar. Lieferanten können gezielt optimiert werden.

7.4.4 Rechnungsabgleich und Diskrepanzbehandlung

Zweck: Abgleich von Wareneingang mit Lieferantenrechnung und Behandlung von Abweichungen

Ablauf:

1. Lieferantenrechnung wird empfangen (EDI oder eingescannt)
2. System führt 3-Way-Match durch:
   - Bestellung (Soll-Menge, Soll-Preis)
   - Wareneingang (Ist-Menge, Ist-Zeitpunkt)
   - Rechnung (Rechnungs-Menge, Rechnungs-Preis)
3. System prüft auf Diskrepanzen:
   - Mengendifferenz (z.B. 100 bestellt, 95 erhalten, 100 berechnet)
   - Preisdifferenz (z.B. Rabatt nicht berechnet)
   - Steuerdifferenz
4. Diskrepanzen werden klassifiziert:
   - Automatisch auflösbar (z.B. Rundungsfehler)
   - Manuell zu klären
5. Bei Diskrepanz: Benutzer wird benachrichtigt
6. Benutzer kann akzeptieren, ablehnen oder mit Lieferant klären
7. Genehmigte Rechnungen werden zur Bezahlung freigegeben

Betroffene Felder: ReceiptDiscrepancy (DiscrepancyType, Amount, Resolution), EDIInvoice (MatchStatus), PurchaseOrder (InvoiceStatus)

Auswirkungen: Rechnungskorrektheit wird garantiert. Doppelzahlungen werden verhindert. Streitfälle mit Lieferanten werden minimiert. Zahlungsprozess wird beschleunigt.

7.4.5 Varianten und Austausch-Artikel handhaben

Zweck: Verwaltung von Artikel-Varianten und Austausch-Artikel bei Wareneingang

Ablauf:

1. Benutzer erhält Artikel, der nicht exakt der Bestellung entspricht (Variante oder Austausch)
2. System zeigt Abweichungs-Dialog
3. Optionen:
   a) Akzeptieren als bestellter Artikel (mit Preisanpassung falls nötig)
   b) Ablehnen und RMA erstellen
   c) Als Alternative/Variante akzeptieren und buchen
4. Bei Varianten-Akzeptanz:
   - Alternative Artikel wird gebucht
   - Preis wird ggf. angepasst
   - Kundenbenachrichtigung wird generiert (falls Auswirkung auf Verkauf)
5. Bei Austausch-Artikel:
   - Original-Artikel bleibt in Auftrag
   - Neuer Artikel wird als Ersatz gebucht
   - Lieferant wird informiert
6. Varianten-Geschichte wird dokumentiert

Betroffene Felder: ReceiptItems (OrderedArticleI3D, ReceivedArticleI3D, IsVariant, VariantAcceptanceReason), ArticleSubstitution (OriginalArticleI3D, SubstituteArticleI3D)

Auswirkungen: Flexible Lieferanten-Handling. Kleine Abweichungen werden schnell gelöst. RMA-Prozesse werden verkürzt. Lagerbestände sind genauer.

7.4.6 Lieferanten-Leistungskennzahlen und Bonifikation

Zweck: Verfolgung von Lieferanten-Leistung und Berechnung von Boni/Malus

Ablauf:

1. System verfolgt automatisch Lieferanten-Metriken:
   - Liefertreue (% pünktlich, % mit korrekter Menge)
   - Qualität (Mängelquote, RMA-Quote)
   - Preiseinhaltung (% Abweichungen von Vereinbartem)
2. Benutzer kann Bonifikations-Verträge konfigurieren
3. System kalkuliert automatisch Boni/Malus basierend auf Performance
4. Report zeigt Leistungs-Übersicht pro Lieferant
5. Trends werden visualisiert (Verbesserung oder Verschlechterung)
6. Bonifikationen werden automatisch bei Rechnungsfreigabe berücksichtigt
7. Lieferanten-Ranking wird erstellt (Top-Performer vs. Problem-Lieferanten)

Betroffene Felder: SupplierPerformanceMetrics (DeliveryAccuracy, QualityScore, PriceAccuracy), SupplierBonification (BonusPercentage, MalusPercentage), SupplierRanking (PerformanceScore)

Auswirkungen: Lieferanten werden zu besserer Performance motiviert. Verträge reflektieren tatsächliche Performance. Einkauf kann Lieferanten gezielt auswählen basierend auf Daten. Langfristige Lieferanten-Partnerschaften werden gestärkt.

8. Helpdesk (Helpdesk/Support)

8.1 Checklisten (Checklists)

Module Path: `src/centron/Centron.WPF.UI/Modules/Helpdesk/CentronChecklist`
Controller: `CentronChecklistAppModuleController`
ViewModel: `CentronChecklistAppModuleControllerViewModel`
Category: Helpdesk
Description: Erstellung und Verwaltung von Checklisten-Templates und Instanzen
License: LicenseGuids.Centron oder LicenseGuids.Helpdesk

Use Cases

8.1.1 Checklisten-Template erstellen

Zweck: Standardisierte Checklisten-Vorlagen für wiederholte Prozesse definieren

Ablauf:

1. Benutzer (Teamleiter) öffnet Checklisten-Modul
2. Wählt "Neue Template erstellen"
3. Definiert Template-Name und Kategorie (z.B. "Hardware-Setup", "Kundenfreigabe")
4. Fügt Checklistenpunkte hinzu mit:
   - Beschreibung (Was muss getan werden?)
   - Priorität (Kritisch/Normal/Optional)
   - Verantwortlicher (welche Rolle?)
   - Geschätzter Aufwand (Minuten)
5. Kann Sub-Items und Abhängigkeiten zwischen Items definieren

6. Setzt Gültigkeitsdauer für Template (z.B. "Alle 2 Jahre aktualisieren")
7. Speichert Template für wiederverwendung

Betroffene Felder: CentronChecklist (TemplateName, Category, IsTemplate, Items, CreatedByI3D, CreatedDate)

Auswirkungen:

- Prozess-Standardisierung
- Qualität konsistent halten
- Training neuer Mitarbeiter erleichtern
- Audit-Compliance dokumentieren

8.1.2 Checkliste aus Template erstellen und durcharbeiten

Zweck: Standardisierte Checkliste für konkretes Projekt/Ticket durcharbeiten

Ablauf:

1. Support-Techniker erstellt neue Checkliste basierend auf Template
2. Verknüpft mit Ticket/Projekt/Kundenauftrag
3. System zeigt Checkliste als Master-Detail View:
   - Links: Alle Items mit Checkbox
   - Rechts: Details des aktuellen Items, Notizen-Feld, Status
4. Techniker arbeitet Items durch:
   - Liest Beschreibung
   - Führt Aktion durch
   - Checkt Item ab oder markiert "Nicht anwendbar"
   - Fügt Notizen hinzu (bei Fehlern/Abweichungen)
5. System zeigt Fortschritt: X von Y Items abgehakt
6. Abhängigkeiten: Kann Item B erst bearbeitet werden wenn Item A erledigt?
7. Nach Abschluss: Bestätigung mit Zeitstempel und Benutzer

Betroffene Felder: CentronChecklist (ChecklistItems, CheckedByI3D, CheckedDate, Notes, Status, LinkedTicketI3D)

Auswirkungen:

- Nichts vergessen (Qualitätssicherung)
- Prozesse dokumentieren
- Audit-Trail automatisch erzeugt
- Kundenzufriedenheit erhöhen

8.1.3 Checklisten-Vorlagen verwalten und aktualisieren

Zweck: Checklisten-Templates auf dem neuesten Stand halten

Ablauf:

1. Administrator öffnet Template-Management-View
2. Zeigt Liste aller Checklisten-Templates:
   - Vorlagenname, Kategorie, Erstellt am, Letzte Änderung, Verwendungsanzahl
3. Pro Template: Bearbeitungsoptionen:
   - Bearbeiten: Items hinzufügen/löschen/ändern
   - Duplizieren: Schnell neue Variante erstellen
   - Verwendungen: Wieviele Check listen verwendet dieses Template?
   - Löschen (mit Warnung wenn noch in Verwendung)
4. Versionierung: Template-Versionen tracken, alte Versionen archivieren
5. Aktivierung/Deaktivierung: Template inaktiv machen, aber nicht löschen
6. Kommentar: Grund für Änderungen dokumentieren

Betroffene Felder: CentronChecklist (TemplateName, IsActive, VersionNumber, ChangedByI3D, ChangedDate)

Auswirkungen:

- Templates kontinuierlich verbessern
- Best-Practices dokumentieren
- Compliance-Anforderungen anpassen
- Prozessoptimierung systematisch umsetzen

8.1.4 Checklisten-Analysen und Compliance-Reporting

Zweck: Analysieren wie gut Checklisten durchgeführt werden

Ablauf:

1. Manager öffnet Reporting-View
2. Zeigt Checklisten-Statistiken:
   - Insgesamt erstellte Checklisten vs. abgeschlossene
   - Abschlussquote % pro Template
   - Ø-Zeit zum Abschließen einer Checkliste
   - Pro Benutzer: Anzahl durchgeführte Checklisten
3. Warnung bei nicht abgeschlossenen Checklisten (älter als X Tage)
4. Trend: Checklisten-Qualität verbessert sich? (Weniger Abweichungen in Notizen)
5. Erkannte Probleme: Häufige Fehler/Übersehen, wo viel Zeit verschwendet wird
6. Export: Report für Management oder zur Compliance-Dokumentation
7. Filter nach Template, Benutzer, Zeitraum

Betroffene Felder: CentronChecklist (ChecklistItems mit CheckedByI3D, CheckedDate), Audit-Trail

Auswirkungen:

- Prozessqualität transparent
- Schulungsbedarf erkennen
- Prozesse optimieren (schneller machen)
- Compliance-Anforderungen nachweisen

8.1.5 Mobile Checklisten-Erfassung

Zweck: Checklisten auch am Kunden-Vor-Ort durchführen können

Ablauf:

1. Techniker auf Kundenbaustelle öffnet c-entron Mobile App
2. Startet Checkliste für aktuelles Ticket
3. Im Offline-Modus kann Checkliste bearbeitet werden:
   - Items abhaken
   - Fotos machen und anhängen
   - Notizen mit Sprachmemo
   - Fehler fotografieren
4. Sync: Nach Rückkehr ins Büro Synchronisation mit Datenbank
5. System validiert Daten: Alle Felder ausgefüllt? Fotos vorhanden?
6. Checkliste offline verfügbar auch ohne Internetverbindung
7. Status-Icons zeigen Sync-Status (grün=synchron, gelb=ausstehend)

Betroffene Felder: CentronChecklist (alle Felder), Attachments, SyncStatus, LastSyncDatetime

Auswirkungen:

- Papier-Checklisten eliminieren

- Fehlerlose Datenerfassung
- Schneller im Büro verfügbar
- Audit-Trail auch mobil

8.1.6 Verknüpfung mit Tickets und Kundenaufträgen

Zweck: Checklisten-Fortschritt an Ticket-Status binden

Ablauf:

1. Beim Erstellen von Ticket kann Checklisten-Template zugeordnet werden
2. Beispiele: Ticket-Typ "Neuer Kunde" → Template "Kundenfreigabe-Checkliste"
3. Automatische Zuordnung basierend auf Ticket-Kategorie/Typ
4. Ticket zeigt Checklisten-Status: ✓ Abgeschlossen, ⏱ In Arbeit, ✗ Nicht gestartet
5. Checkliste blockiert Ticket-Schließung wenn nicht komplett (Optional: konfigurierbar)
6. Linked-Ansicht: Ticket kann mit Checkliste gescrollt werden (beide sichtbar)
7. Export: Ticket-Abschluss-Report inklusive Checklisten-Bestätigung

Betroffene Felder: Ticket (LinkedChecklistI3D), CentronChecklist (LinkedTicketI3D)

Auswirkungen:

- Kein Ticket vergessen (Prozess-Garantie)
- Audit-Compliance automatisch
- Qualitätssicherung beim Ticket-Abschluss
- Support-Prozess-Transparenz

8.1.7 Checklisten-Konvertierung (Legacy zu Neu)

Zweck: Alte papiergestützte Checklisten in System migrieren

Ablauf:

1. Administrator startet Import-Wizard
2. Importiert alte Checklisten-Scans oder CSV-Dateien
3. System erkennt Items durch OCR oder manuelle Zuordnung
4. Mapping: Alte Item-Namen → neue Item-Strukturen
5. Batch-Verarbeitung: Mehrere Checklisten auf einmal konvertieren
6. Validierung: Sind alle Items erkannt worden?
7. Nach Bestätigung: Checklisten sind digital verfügbar, historische Daten archiviert
8. Reporting: Migrationsstatistiken (wieviele erfolgreich, wieviele Fehler)

Betroffene Felder: CentronChecklist (ImportedFromLegacy, OriginalScannedFilePath, ConversionDatetime)

Auswirkungen:

- Historische Daten nicht verlieren
- Digital suchbar machen
- Compliance-Dokumentation
- Physischen Platz sparen

8.2 Projektverwaltung (Project Management)

Module Path: `src/centron/Centron.WPF.UI/Modules/ProjectManagement`
Controller: `ProjectManagementAppModuleController`

ViewModel: `ProjectManagementViewModel`
Category: Helpdesk
Description: Übersicht über Entwicklungsprojekte mit Ticketing und Ressourcenplanung
License: LicenseGuids.Centron
Hinweis: Primär intern für c-entron software gmbh Entwicklungsprojekte

Use Cases

8.2.1 Entwicklungsprojekte anzeigen und filtern

Zweck: Alle laufenden Entwicklungsprojekte und deren Status überblicken

Ablauf:

1. Benutzer öffnet Projektverwaltung-Modul
2. System zeigt Tabelle aller Projekte:
   - Projektname, Projekttyp (c-entron Development/Service Project), Status, Startdatum, geplantes Enddatum
   - Projektleiter, Team-Größe, Budget, aktuelle Kosten, Gewinn/Verlust
   - SLA/Deadline, Abweichung vom Plan
3. Filter nach: Projekttyp, Status (aktiv/abgeschlossen/storniert), Projektleiter, Kunde
4. Sortierung: Nach Deadline, Budget, Fortschritt
5. Farbcodierung: Rot = überfällig, Gelb = kritisch, Grün = im Plan
6. Klick auf Projekt → Details anzeigen

Betroffene Felder: CrmProject (ProjectName, ProjectKind, Status, StartDate, EndDate, BudgetAmount, ActualCost)

Auswirkungen:

- Projektportfolio transparent
- Prioritäten setzen (welche sind kritisch)
- Ressourcen auf wichtigste Projekte konzentrieren
- Chancen/Risiken schnell erkennen

8.2.2 Tickets pro Projekt anzeigen und verwalten

Zweck: Alle Support-/Entwicklungs-Tickets eines Projekts zusammen sehen

Ablauf:

1. Benutzer wählt Projekt
2. System zeigt verknüpfte Tickets in Detail-Tab:
   - Ticketnummer, Typ (Feature/Bug/Task), Titel, Status, Priorität
   - Zugewiesen an (Entwickler), Geschätzter Aufwand, Tatsächlicher Aufwand
   - Abhängigkeiten zu anderen Tickets
3. Pro Ticket: Drill-Down zu Details, Kommentare, Attachments
4. Batch-Operationen: Mehrere Tickets gleichzeitig Status ändern
5. Gantt-Chart: Timeline der Tickets visualisieren
6. Test-Status pro Ticket: Getestet? Von wem? Bestanden?

Betroffene Felder: Ticket (LinkedProjectI3D, TicketKind, Status, EstimatedHours, ActualHours), TicketDependencies

Auswirkungen:

- Projektfortschritt tracken (Tickets abarbeiten = Projekt voranschreiten)
- Bottlenecks erkennen (welche Tickets sind kritisch)
- Prognosen erstellen (auf Kurs für Deadline?)
- Qualitätssicherung (Tickets durchlaufen Test)

8.2.3 Team-Auslastung und Ressourcenplanung pro Projekt

Zweck: Sind Ressourcen ausreichend? Wie ausgelastet ist das Team?

Ablauf:

1. Projektleiter öffnet "Team-Auslastung"-Tab
2. System zeigt pro Projektmitarbeiter:
   - Name, Rolle, Geplante Stunden gesamt, Gebuchte Stunden, Verfügbar noch, Auslastung %
   - Tickets die diesem Mitarbeiter zugeordnet sind
   - Abhängigkeiten: Blockt dieser Mitarbeiter andere durch Wartezeit?
3. Trend: Entwickelt sich Auslastung nach Plan?
4. Warnung bei Über-/Unterauslastung
5. Scenario-Planung: "Was wenn wir 1 Person hinzufügen?"
6. Resource-Leveling: System schlägt Ressourcen-Umverteilung vor

Betroffene Felder: Employee, Workload (ProjectI3D), Ticket (AssignedI3D), ScheduleData

Auswirkungen:

- Realistische Planung
- Personalentscheidungen treffen (Verstärkung brauchen?)
- Burnout vermeiden
- Projektplan halten

8.2.4 Projektfortschritt und Earned Value Analyse

Zweck: Ist Projekt nach Plan oder gibt es Abweichungen?

Ablauf:

1. Projektleiter öffnet "Fortschritt"-Tab
2. System zeigt Earned Value Metriken:
   - Geplanter Wert (PV): Was sollte bis jetzt erledigt sein?
   - Tatsächlicher Wert (EV): Was ist wirklich erledigt (prozentual der Tickets)?
   - Tatsächliche Kosten (AC): Was hat es tatsächlich gekostet?
3. Abweichungen berechnen:
   - Zeitabweichung (Schedule Variance): Zu früh/spät?
   - Kostenabweichung (Cost Variance): Über/unter Budget?
4. Charts: Geplanter vs. Tatsächlicher Fortschritt über Zeit
5. Prognose: Auf Basis aktuellem Trend, wann wird Projekt fertig? Wie viel kostet es?
6. Risikoliste: Bekannte Risiken die sich realisiert haben?

Betroffene Felder: CrmProject (BudgetAmount, StartDate, EndDate), Ticket (EstimatedHours, ActualHours, Status), Workload (Date, Hours)

Auswirkungen:

- Früh erkennen wenn Projekt aus Ruder läuft
- Gegenmassnahmen einleiten (Ressourcen erhöhen, Scope reduzieren)
- Finanzielle Kontrolle
- Stake-Holder datengestützt informieren

8.2.5 Terminplanung und Abwesenheitsberücksichtigung

Zweck: Berücksichtige Urlaub/Abwesenheit bei Projekt-Planung

Ablauf:

1. Projektleiter sieht Kalenderansicht:
   - Farbliche Markierung: Wer ist wann im Urlaub/Krank/Training?

- Abwesenheit ist blockierte Zeit bei Auslastungsberechnung

2. Auto-Warnung: "Projektleiter X ist in 2 Wochen 1 Woche im Urlaub"
3. Skill-Abbildung: Wer kann X kompensieren wenn Projektleiter X weg ist?
4. Planung: Projekte oder wichtige Milestones nicht in Abwesenheitszeiten planen
5. Was-Wenn: "Wenn dieser Mitarbeiter 1 Woche früher Urlaub nimmt, bekommt Projekt eine Woche Rückschi stand?"
6. Reporting: Abwesenheitsrate pro Team (wird zu viel im Projekt geplant wenn Team weg ist?)

Betroffene Felder: Absence (Date, Type, Duration), Terminplanung (ScheduleDate), Employee, Project

Auswirkungen:

- Realistische Planung (nicht zu ehrgeizig)
- Mitarbeiter-Wohlbefinden (Urlaub respektiert)
- Projektrückschi stand durch Personalmangel vermeiden
- Fair workload distribution

8.2.6 Vertragliche Liefertermine und Verträge

Zweck: Verträge zu Lieferterminen tracken und einhalten

Ablauf:

1. Projektverwaltung zeigt verknüpfte Verträge:
   - Kunde, Lieferdatum, Was zu liefern ist, Rechnungsbetrag
   - Penalty bei Verzug (z.B. 1% pro Woche)
   - Bonus bei frühzeitiger Lieferung (z.B. 2%)
2. Warnung wenn Projekt hinter dem Plan und Lieferdatum in Gefahr
3. Automatische Benachrichtigung 2 Wochen vor Lieferdatum an Projektleiter
4. Prognose: Wenn aktueller Trend: Wann wird Projekt fertig?
5. Eskalation: Wenn Verspätung droht, automatisch an Geschäftsführung
6. Nachverfolgung: Nach Lieferung bestätigen, für Rechnungstellung

Betroffene Felder: Contract (DeliveryDate, PenaltyPercentage, BonusPercentage), CrmProject (StartDate, EndDate), Ticket (Status, EstimatedCompletionDate)

Auswirkungen:

- Terminzusagen erfüllen (Kundenzufriedenheit)
- Finanzielle Penalties vermeiden
- Bonusgewinne realisieren
- Vertrags-Compliance

8.2.7 Technischer Schulden und Refactoring-Tracker

Zweck: Technische Schuld systematisch abblasen

Ablauf:

1. Im Projekt können "Technical Debt" Tickets markiert werden
2. System sammelt diese und zeigt Schulden-Report:
   - Gesamt-Schulden (Schätzung in Stunden)
   - Pro-Area: Welche Code-Module haben am meisten Schulden?
   - Trend: Wird Schuld schneller erzeugt oder abgebaut?
   - Prognose: Wie lange bis Schuld abgebaut?
3. Priorisierung: Welche Schuld bringt am meisten Nutzen wenn abbezahlt?
4. Refactoring-Tickets: Plan für Schuld-Abbau
5. QA-Metriken: Code Coverage, Bugs, Performance verbessern sich?

Betroffene Felder: Ticket (IsDebtTicket, DebtCategory, DebtEstimateHours), CodeQualityMetrics

Auswirkungen:

- Code-Qualität verbessern
- Zukünftige Features schneller realisierbar
- Bug-Raten senken
- Team-Moral verbessern (sauberer Code macht Spaß)

8.2.8 Retrospektiven und Lessons Learned

Zweck: Nach Projekt-Abschluss Best-Practices dokumentieren

Ablauf:

1. Nach Projekt-Completion öffnet Projektleiter "Retrospektive"-View
2. System zeigt Template für Lessons Learned:
   - Was hat gut funktioniert?
   - Was hätte besser gehen können?
   - Geplante vs. Tatsächliche Aufwände (lernen für zukünftige Schätzungen)
   - Team-Rückmeldung: Waren Prozesse klar? Kommunikation OK?
3. Benutzer füllt aus und speichert als Template für ähnliche Projekte
4. Search: Zukünftige Projekte können von ähnlichen alten Projekten lernen
5. Metriken-Sammlung: Durchschnittliche Aufwand pro Feature-Typ für Schätzungen
6. Sharing: Best-Practices mit anderen Teams teilen

Betroffene Felder: CrmProject (LessonsLearned, PostProjectReview), HistoricalProjectData, EstimationDatabase

Auswirkungen:

- Schätzungen zukünftig genauer
- Wiederholte Fehler vermeiden
- Best-Practices systematisch implementieren
- Organisatorisches Lernen

8.3 RMA/Werkstatt (RMA/Workshop)

Module Path: `src/centron/Centron.WPF.UI/Modules/Rma`
Controller: `RmaOverviewAppModulController`
ViewModel: `RmaOverviewViewModel`
Category: Helpdesk
Description: Verwaltung von RMA-Prozessen (Reparatur-Rücksendungen) und Werkstatt-Operationen
License: LicenseGuids.RMAWorkshop

Use Cases

8.3.1 RMA erstellen und registrieren

Zweck: Reparaturfall (RMA) anlegen und in System registrieren

Ablauf:

1. Support-Techniker öffnet "Neue RMA erstellen"
2. Wizard führt durch Schritte:
   - Kundenauswahl oder Vertrag-Zuordnung
   - Artikel/Seriennummern eingeben (was wird repariert?)
   - Fehlerbeschreibung dokumentieren (Benutzer-Beschreibung)
   - Versand-Art: Kunde sendet zu uns oder wir holen ab?

3. System generiert automatisch RMA-Nummer (z.B. RMA-2025-001234)
4. Erzeugt verknüpftes Support-Ticket automatisch
5. Zeigt QR-Code oder Barcode zum Ausdrucken für Versand-Label
6. RMA-Status: "Eingegangen"

Betroffene Felder: RMA (Number, CustomerI3D, ArticleI3D, SerialNumber, Description, Status, CreatedDate)

Auswirkungen:

- Reparaturfall strukturiert dokumentiert
- Audit-Trail von Anfang an
- Kunde hat RMA-Nummer für Tracking
- Werkstatt weiß was zu reparieren ist

8.3.2 RMA-Bearbeitung in der Werkstatt

Zweck: Reparatur durchführen und Fortschritt tracken

Ablauf:

1. Werkstatt-Techniker erhält RMA
2. Scannt RMA-Nummer oder sucht im System
3. System zeigt RMA-Details: Artikel, Fehler, Kundendaten
4. Techniker dokumentiert Reparatur-Schritte:
   - Diagnose: Was war das Problem? (Fehlerkod, Prüfung durchgeführt)
   - Reparatur: Was wurde repariert/ausgetauscht?
   - Teile-Verbrauch: Welche Ersatzteile wurden verwendet? (aus Werkstatt-Lager)
   - Arbeitszeit: Wie lange hat Reparatur gedauert?
5. Quality-Check: Funktioniert es jetzt? Ja/Nein mit Bestätigung
6. RMA-Status: "In Reparatur" → "Repariert" oder "Nicht reparierbar"

Betroffene Felder: RMA (Status, DiagnosisDescription, RepairDescription, LabourHours, Cost), Parts-Verbrauch

Auswirkungen:

- Reparatur-Verlauf dokumentiert
- Kosten tracken (Arbeit + Teile)
- Später für Berechnung/Rechnungstellung verfügbar
- Garantie-Anforderungen erfüllt

8.3.3 RMA zurück an Kunde versenden

Zweck: Repariertes Gerät zurück an Kunde versenden

Ablauf:

1. Nach erfolgreicher Reparatur: "Versand vorbereiten"
2. System zeigt Versand-Optionen: Paket, Express, Standard
3. Adresse: Kundenadresse aus Vertrag/Ticket
4. Packzettel generieren (mit RMA-Nummer, Reparatur-Beschreibung)
5. Versand-Label drucken
6. System erzeugt Versand-Tracking-Nummer (oder Integration mit Logistik-Partner)
7. Kunde erhält Email mit Tracking-Link
8. RMA-Status: "Versendet"
9. Tracking: Kunde kann Lieferung live verfolgen

Betroffene Felder: RMA (Status, ShippingDate, ShippingCarrier, TrackingNumber), Customer (Email, Address)

Auswirkungen:

- Transparente Kommunikation mit Kunde
- Rücksendezeiten tracken
- Versandkosten erfassen
- Logistik-Integration automatisiert

8.3.4 RMA-Lagerverwaltung (RMA-Eigenbestand vs. Kundeneigentum)

Zweck: Unterscheiden zwischen Reparatur-Leihgeräten und Kundeneigentum

Ablauf:

1. System verwaltet zwei Stock-Typen:
   - RmaOwn: Leihgeräte vom Unternehmen (Ersatz während Reparatur)
   - RmaCustomer: Geräte in Reparatur (gehören Kunden)
2. Wenn Gerät nicht sofort reparierbar: Leihgerät aus RmaOwn-Bestand geben
3. Tracking: Welche Leihgeräte sind bei welchen Kunden? Wie lange?
4. Warnung: Leihgerät >30 Tage beim Kunden? Rückruf initiieren
5. Bestandsverwaltung: Leihgeräte müssen gepflegt werden (Wartung, Updates)
6. Nach Reparatur: Original-Gerät zurück, Leihgerät retourniert

Betroffene Felder: Stock (RmaOwn/RmaCustomer), RMA (LoanDeviceI3D, ReturnDate, CustomerDeviceI3D)

Auswirkungen:

- Teuer Leihgeräte nicht verlieren
- Kundenbeziehung durch schnellen Ersatz verbessern
- Vermögens-Kontrolle
- Bestandsplanung (wieviele Leihgeräte braucht es?)

8.3.5 RMA-Kostenberechnung und Rechnungstellung

Zweck: Reparaturkosten berechnen und Kunde rechnen

Ablauf:

1. Nach abgeschlossener Reparatur: "Rechnungsentwurf" generieren
2. System berechnet automatisch:
   - Material-Kosten: Ersatzteile aus Lager × Kosten
   - Arbeits-Kosten: Arbeitsstunden × Stundensatz
   - Versandkosten: Versand hin + zurück
   - Rabatte anwenden (z.B. Garantie, Vertrag)
   - Gesamt-Kosten und Gewinn
3. Benutzer kann Rechnung prüfen und ggf. anpassen
4. Rechnung an Kunden versenden
5. Status: "Abgeschlossen"
6. Zahlung-Tracking: Wurde Rechnung bezahlt?

Betroffene Felder: RMA (LabourCost, MaterialCost, ShippingCost, InvoiceI3D, Status), Invoice

Auswirkungen:

- Reparatur-Einnahmen tracken
- Gewinnmargenanalyse pro RMA-Typ
- Kundenzufriedenheit durch faire Preisstellung
- Revenue-Erkennung für Accounting

8.3.6 RMA-Statistiken und Qualitätsmetriken

Zweck: Analysen zur Reparatur-Qualität und Effizienz

Ablauf:

1. Manager öffnet RMA-Analytics
2. Metriken werden angezeigt:
   - Reparaturquote: % erfolgreich repariert vs. nicht reparierbar
   - Ø-Reparaturzeit: Wie lange dauert Reparatur? (Trend)
   - Kosten pro RMA: Ø-Kosten für Reparatur und Versand
   - Reklamationsquote: % RMA die nochmal zurückkommen (defekt)
   - Lagerumgang-Kosten: Kosten für Leihgeräte
3. Problematische Artikel: Welche Geräte gehen oft kaputt?
4. Prognose: Kapazität der Werkstatt ausreichend?

Betroffene Felder: RMA (Status, RepairOutcome, LabourHours, ReturnRate, Cost), Parts-Usage

Auswirkungen:

- Reparatur-Service-Qualität transparent
- Problematische Lieferanten/Artikel identifizieren
- Werkstatt-Kapazität planen
- Profitabilität Reparaturgeschäft

8.3.7 RMA-Archivierung und Compliance

Zweck: Reparaturdaten für Compliance und Audit dokumentieren

Ablauf:

1. Nach RMA-Abschluss wird Datensatz archiviert
2. Digitale Ablage:
   - RMA-Dokumentation (Diagnose, Reparatur, Qualitäts-Check)
   - Fotos von Schaden/Reparatur (optional)
   - Ersatzteile-Rechnung (Lieferant)
   - Kundenrechnung
3. Aufbewahrungsfristen: Je nach Garantie/Gewährleistung (z.B. 2-5 Jahre)
4. Search: Ältere RMA können für Mustersuche durchsucht werden
5. Export: Für externe Audits (Kunden-Audits, Garantie-Ansprüche)
6. Datenschutz: Kundendata nach GDPR nach Aufbewahrungsfrist löschen

Betroffene Felder: RMA (ArchiveDate, RetentionUntil, DocumentLinks), Audit-Trail

Auswirkungen:

- Garantie-Ansprüche dokumentiert und defensiv
- Compliance mit Datenschutz
- Externe Audits problemlos bestandbar
- Langfristige Analyse historischer Daten

8.4 Taskmanagement (Task Management)

Module Path: `src/centron/Centron.WPF.UI/Modules/Helpdesk/TaskManagement`
Controller: `TaskManagmentAppModuleController`
ViewModel: `TaskManagmentAppModuleControllerViewModel`
Category: Helpdesk

Description: Erstellung und Verwaltung von Tasks mit Prioritäten und Tracking
License: LicenseGuids.TaskManagement

Use Cases

8.4.1 Task erstellen und zuweisen

Zweck: Arbeitspakete als Tasks anlegen und Mitarbeitern zuweisen

Ablauf:

1. Benutzer öffnet "Neue Task" Dialog
2. Definiert:
   - Titel: "Was muss getan werden?"
   - Beschreibung: Detaillierte Anleitung
   - Priorität: Hoch/Normal/Niedrig
   - Zugewiesen an: Mitarbeiter
   - Fällig am: Deadline
   - Kategorie: Typ der Task (Administrativ, Support, Entwicklung, etc.)
   - Parent-Task: Ist das Sub-Task von größerer Task?
3. Optional: Abhängigkeiten zu anderen Tasks
4. Optional: Geschätzter Aufwand (Stunden)
5. Speichert Task und benachrichtigt zugewiesenen Mitarbeiter

Betroffene Felder: Task (Title, Description, Priority, AssignedToI3D, DueDate, Category, Status, CreatedByI3D)

Auswirkungen:

- Alle Arbeit im System transparent
- Keine Aufgaben vergessen
- Priorisierung klar
- Audit-Trail wer hat was aufgetragen

8.4.2 Task-Board und Kanban-Ansicht

Zweck: Visualisiere Tasks im Kanban-Stil nach Status

Ablauf:

1. Benutzer öffnet Task-Board
2. Spalten nach Status: "Neu" → "In Arbeit" → "Review" → "Abgeschlossen"
3. Tasks als Karten auf dem Board angezeigt
4. Drag&Drop: Task von "In Arbeit" auf "Review" ziehen um Status zu ändern
5. Farben: Rot=überfällig, Gelb=demnächst fällig, Grün=OK
6. Klick auf Task: Details, Kommentare, Sub-Tasks anzeigen
7. Filtermöglichkeiten: Nach Zugewiesen, Priorität, Kategorie

Betroffene Felder: Task (Status, DueDate, Priority), Workflow-State

Auswirkungen:

- Visuelle Übersicht des Workflo
- Schnelles Identifizieren von Blockern
- Motivation durch sichtbaren Fortschritt
- Team-Transparenz

8.4.3 Task-Abhängigkeiten und Reihenfolge

Zweck: Beschreiben von Task-Abhängigkeiten (Task B kann erst nach Task A starten)

Ablauf:

1. Bei Task-Erstellung kann "Abhängig von Task X" definiert werden
2. System zeigt Abhängigkeits-Graph: Welche Tasks müssen erst erledigt werden?
3. Automatische Blockierung: Task B kann nicht "fertig" markiert werden solange Task A nicht erledigt
4. Warnung: Wenn Task A überfällig wird, zeigt System dass Task B auch blockiert
5. Gantt-Chart: Zeitliche Darstellung von Task-Sequenzen
6. Prognose: Mit aktueller Geschwindigkeit, wann sind alle Tasks fertig?

Betroffene Felder: Task (DependsOnTaskI3D, BlockedByTasks), ScheduleData

Auswirkungen:

- Richtige Reihenfolge einhalten
- Ungültige Task-Zuordnungen vermeiden (falsche Reihenfolge)
- Prognosen genauer
- Ressourcen-Konflikte erkennen

8.4.4 Task-Kommentare und Zusammenarbeit

Zweck: Team-Kommunikation rund um Task

Ablauf:

1. Zugewiesener Mitarbeiter öffnet Task
2. Kann Kommentare hinzufügen: Status-Updates, Fragen, Probleme
3. @-Mentions: "@Max schau dir das an" → Max bekommt Benachrichtigung
4. Attachments: Dateianhänge (Screenshots, Dokumente)
5. Versionierung: Alte Kommentare bleiben sichtbar (keine Deletion)
6. Notifications: Task-Besitzer und Stakeholder erhalten Updates
7. History: Zeige alle Änderungen an Task (wer hat was wann geändert)

Betroffene Felder: TaskComment (Content, CommentedByI3D, CommentDate, ParentTaskI3D), TaskHistory

Auswirkungen:

- Transparente Zusammenarbeit
- Keine Lost-in-Email-Kommunikation
- Audit-Trail von Entscheidungen
- Asynchrone Kommunikation möglich

8.4.5 Task-Zeittracking und Effort-Vergleich

Zweck: Tracken wie viel Zeit eine Task wirklich dauert

Ablauf:

1. Bei Task-Abschluss: Tatsächlich verbrauchte Stunden eingeben
2. Vergleich: Geschätzte vs. Tatsächliche Stunden
3. System berechnet Accuracy: "Schätzungen meist 20% zu hoch" (Lerneffekt)
4. Report: Ø-Zeiten pro Task-Typ für zukünftige Schätzungen
5. Billing (optional): Zeitaufwand kann für Kunden-Abrechnung genutzt werden
6. Trend: Werden Tasks schneller erledigt? (Prozess-Verbesserung)

Betroffene Felder: Task (EstimatedHours, ActualHours, TimeTrackingEntries), Accuracy-Metrics

Auswirkungen:

- Zukünftige Schätzungen genauer
- Persönliches Productivity-Tracking (wem trauen wir Zeit-Schätzungen?)

- Kundenabrechnung transparenter
- Prozess-Verbesserungen messen

8.4.6 Task-Templates und Wiederkehrende Tasks

Zweck: Standardisierte Task-Sets für wiederholte Arbeiten

Ablauf:

1. Administrator erstellt Task-Template (z.B. "Monats-Abschluss")
2. Template enthält: Liste von Sub-Tasks, Abhängigkeiten, Prioritäten
3. Wenn Monat zu Ende: Template wird als Batch-Task-Set erstellt
4. Alle Sub-Tasks werden automatisch zugewiesen (pro Rolle)
5. Option: "Task jeden Monat automatisch erstellen" (Wiederholung)
6. Benutzer kann Template vor Erstellung anpassen
7. Nach Abschluss: Template für nächsten Monat wieder nutzen

Betroffene Felder: TaskTemplate (Name, SubTasks, RecurrenceRule), Task (TemplateI3D, IsRecurring)

Auswirkungen:

- Wiederholte Arbeit standar disieren
- Kein vergessen von Routineaufgaben
- Konsistente Prozesse
- Training neuer Mitarbeiter

8.4.7 Task-Eskalation und Alerts

Zweck: Automatische Warnungen bei überfälligen/kritischen Tasks

Ablauf:

1. System monitoren Task-Status kontinuierlich
2. Wenn Task überfällig: Auto-Benachrichtigung an Zugewiesene + Manager
3. Eskalationsstufen:
   - 1 Stunde nach Deadline: E-Mail
   - 1 Tag nach Deadline: SMS/Popup
   - 2 Tage nach Deadline: Manager und Abteilungsleiter benachrichtigen
4. Priorisierung: Rote (Hoch-Priorität) überfällig → sofortiger Alarm
5. Blockers: Wenn Task blockiert ist (abhängig von Task die auch überfällig) → "Blockiert, nicht schuld des Zugewiesenen"
6. Status-Bericht: Manager sieht alle überfälligen Tasks

Betroffene Felder: Task (DueDate, Status, Priority), EscalationSettings, NotificationLog

Auswirkungen:

- Überfällige Tasks schneller sichtbar
- Automatische Eskalation entlastet Manager
- Proaktives Management statt reaktiv
- Priorization klar

8.4.8 Task-Performance-Analysen

Zweck: Analysen zur Task-Effizienz und Workflows

Ablauf:

1. Manager öffnet Task-Analytics

2. Metriken anzeigen:
    - Abschlussquote: % der Tasks die pünktlich fertig
    - Ø-Durchlaufzeit: Wie lange dauert Task von Erstellen bis Abschluss?
    - Priorisierungs-Effektivität: Werden Hochprio-Tasks tatsächlich zuerst bearbeitet?
    - Pro-Mitarbeiter: Effizienz, Pünktlichkeit, Accuracy von Schätzungen
    - Pro-Kategorie: Welche Task-Typen sind problematisch?
3. Bottleneck-Analyse: Wo stecken Tasks fest?
4. Trend: Wird Team produktiver? (Tasks schneller abschließen)
5. Prognose: Wie viele Tasks können nächste Woche bearbeitet werden?

Betroffene Felder: Task (CreatedDate, DueDate, CompletedDate, EstimatedHours, ActualHours, Status), Performance-Metrics

Auswirkungen:

- Team-Performance transparent
- Schulungsbedarf erkennen (langsame Task-Bearbeitung)
- Prozesse optimieren
- Capacity-Planung verbessern

8.5 Ticket-Liste (Ticket List)

Modulpfad: `src/centron/Centron.WPF.UI/Modules/Helpdesk/TicketList/`
Controller: `TicketListAppModuleController`
ViewModel: `TicketListViewModel`
Schnittstelle: `ITicketLogic`
Kategorie: Helpdesk
Beschreibung: Zentrale Verwaltung und Übersicht aller Support-Tickets mit erweiterten Filterung und Bulk-Operationen
Lizenz: `LicenseGuids.Helpdesk` OR `LicenseGuids.Centron`
Rechte: `UserRightsConst.Helpdesk.TICKET_LIST`

Modul-Architektur

Das Ticket-Listen-Modul ist die **Haupt-Schnittstelle** für Support-Teams zur Verwaltung aller Tickets:

- **Master-Detail View**: Linke Seite Liste der Tickets, rechte Seite Detail-Vorschau
- **Erweiterte Filter**: Nach Status, Priorität, Typ, Zuweiser, Kunde, etc.
- **Bulk-Operationen**: Mehrere Tickets gleichzeitig bearbeiten/schließen
- **SLA-Übersicht**: Farbliche Anzeige von Überschreitungen
- **Schnelle Aktionen**: Häufig benötigte Operationen per Toolbar
- **Kontextu-Menü**: Rechtsclick für weitere Optionen
- **Favoriten & Saved Searches**: Benutzer-definierte Filter speichern

Vollständige Use Cases

8.5.1 Alle offenen Tickets filtern und anzeigen

Zweck: Übersicht über alle aktiven Support-Tickets für den Tag
Ablauf aus Benutzersicht:

1. Support-Team-Lead öffnet Ticket-Liste morgens
2. System zeigt alle Tickets mit Status "Neu", "In Bearbeitung", "Warte auf Kunde"
3. Tickets sind sortiert nach Priorität (Rot→Gelb→Grün)
4. Spalten zeigen: Ticketnummer, Kunde, Titel, Status, Priorität, Zuweiser, Alter (Tage)
5. Team-Lead sieht auf einen Blick welche Tickets kritisch sind
   Standard-Filter aktiv:

- Status: NOT IN ("Abgeschlossen", "Storniert", "Duplikat")
- Zuweiser: Aktuelle Benutzer oder Team
- Datum: Letzten 30 Tage

8.5.2 Nach Kunde, Typ oder Priorität filtern

Zweck: Fokus auf spezifische Tickets eingrenzen
Filterbar nach:

- **Kunde**: Dropdown mit alle c-entron Kunden, Multi-Select
- **Tickettyp**: Incident, Service Request, Change Request, Problem
- **Priorität**: Kritisch (1), Hoch (2), Mittel (3), Niedrig (4), Trivial (5)
- **Status**: Neu, Zugewiesen, In Bearbeitung, Warte auf Kunde, In Review, Abgeschlossen, Storniert
- **Zuweiser**: Welcher Support-Techniker arbeitet dran
- **Kategorie**: z.B. "Abrechnung", "EDI", "Logistik", "Technical Support"
- **SLA-Status**: "Im Plan", "Warnung", "Überschritten"
- **Erstellungsdatum**: Von/Bis Datumspicker
  UI: Schiebbare Filter-Panel auf linker Seite, Checkboxes

8.5.3 Nach Tickettext suchen (Volltextsuche)

Zweck: Schnell frühere Tickets zu ähnlichen Problemen finden
Suchfelder:

- Ticketdescription
- Tickettitel
- Kundenkommentare
- Support-Team Notizen
- Beigefügte Dateien (wenn OCR aktiviert)
  Such-Operatoren:
- Einfache Worte: `printer` → alle Tickets mit "printer"
- Phrase in Anführungszeichen: `"Windows 10 driver"` → exakte Phrase
- Ausschluss: `-deprecated` → alle außer "deprecated"
- Wildcard: `print*` → print, printer, printing, etc.
  Performance: Live-Suche mit Debounce (0,5 Sekunden), max 1000 Ergebnisse

8.5.4 SLA-Status prüfen und Eskalationen sehen

Zweck: Überwachung von Service Level Agreements
Anzeige pro Ticket:

- **Erstellt vor**: X Tagen/Stunden
- **Zielzeit für Response**: Anfrage muss beantwortet sein bis YYYY-MM-DD HH:MM
- **Zielzeit für Lösung**: Ticket muss geschlossen sein bis YYYY-MM-DD HH:MM
- **Status-Indicator**:
  - Grün: Noch 50% Zeit verbleibend
  - Gelb: Noch 25% Zeit verbleibend (Warnung)
  - Rot: Überschritten oder <5% Zeit verbleibend (Kritisch)
- **Buttons**: "Warten auf Kunde" pausiert SLA temporär
  SLA-Regeln basieren auf: Priorität (Kritisch: 4h Response, 8h Lösung | Niedrig: 24h Response, 5 Tage Lösung)

8.5.5 Mehrere Tickets gleichzeitig (Bulk-Operation) bearbeiten

Zweck: Effiziente Massenbearbeitung, z.B. alle ähnliche Tickets schließen
Mögliche Operationen:

- **Checkbox-Select**: Mehrere Tickets mit Ctrl+Click oder Shift+Click selektieren
- **Select All**: Button "Alle auf Seite wählen" oder "Alle Suchresultate wählen"
- **Bulk-Aktion ausführen**:
  - Priorität ändern (auf 3 für alle setzen)
  - Status ändern (z.B. "In Bearbeitung")

- Zuweiser ändern (Team-Lead weist neue Tickets um)
- Kategorie hinzufügen/ändern
- Tags hinzufügen (z.B. "Follow-up erforderlich")
- Notiz anhängen (z.B. "Beantwortet per Mail")
- Tickets schließen/stornieren
- Massen-Export (CSV für Reporting)

  Bestätigung: "Sind Sie sicher? Diese Aktion betrifft 15 Tickets"

8.5.6 Ticket-Details im Panel oder Pop-up anzeigen

Zweck: Schnelle Vorschau ohne separaten Tab zu öffnen
Detail-Ansicht zeigt:

- Ticketnummer und Erstellungsdatum
- Kundenname und Kontakt (Link zur Kundenkarte)
- Kurzbeschreibung und vollständiger Beschreibungstext
- Historie aller Kommentare/Updates
- Beigefügte Dateien mit Download-Links
- Zugewiesener Techniker
- Priorität und aktueller Status
- SLA-Status mit Enddatum
- Verknüpfte Tickets (verwandte Probleme)

  Buttons: "Edit", "Close", "Assign to me", "Add Note", "Attach File"

8.5.7 Schnelle Aktion: Ticket kommentieren und aktualisieren Status

Zweck: Häufigste Operation - schnelle Kommunikation mit Kunden
Workflow:

1. Ticket anklicken in Liste
2. Text-Feld "Internal Notes" für Team-interne Notizen
3. Text-Feld "Customer Response" für Kundenkommunikation
4. Optional: Dropdown "Mark as" → [Warte auf Kunde, In Bearbeitung, Ready for Review]
5. Button "Send & Update Status" → Kommentar wird gespeichert, E-Mail an Kunde
6. Ticket springt zu neuem Status

   E-Mail-Template: Automatisch "Ihr Ticket XXX wurde aktualisiert. {{YourComment}}"

8.5.8 Ticket schließen mit Lösungs-Dokumentation

Zweck: Formal ein Ticket abschließen und Lösung dokumentieren
Prozess:

1. Ticket-Detail-Panel öffnen
2. Button "Schließen" oder Status zu "Abgeschlossen" ändern
3. Pop-up: "Schließungs-Grund wählen" (Gelöst, Duplikat, Nicht reproduzierbar, Keine Antwort, etc.)
4. Optionales Feld: "Kurze Lösung (für Wissensdatenbank)"
5. Optionales Feld: "Zeitaufwand (Stunden)"
6. E-Mail an Kunde: "Ihr Ticket XXX wurde geschlossen am {{DATE}}"
7. Button "Closed, send to customer"

   Auswirkung: Ticket geht aus aktiver Liste weg, Statistiken updated

8.5.9 Favoriten speichern (gespeicherte Suchanfragen)

Zweck: Benutzer-definierte Filter und Suchkriterien für täglich Nutzung
Funktionen:

- "Als Favorit speichern" Button speichert aktuelle Filter-Kombination
- Favorit-Name eingeben: z.B. "Meine offenen Tickets", "Kritische EDI Tickets", "Warte auf Kunden"
- Favorit erscheint in Sidebar-Menü
- Klick auf Favorit lädt Filter automatisch
- Filter-Kombinationen können geteilt werden (Team-Favoriten)

- **Beispiel-Favoriten**:
  - "My Queue": Status IN (Neu, Zugewiesen) UND Zuweiser = mich
  - "Today's SLA Risk": SLA < 10% UND Status != Abgeschlossen
  - "Customer: Acme Corp": Kunde = Acme + offene Tickets

8.5.10 Ticket-Historie und Änderungsverlauf anzeigen

Zweck: Vollständige Dokumentation aller Änderungen am Ticket
Anzeige:

- **Zeitstrahl**: Chronologische Ansicht aller Events
  - Ticket erstellt: Datum, von wem
  - Status-Änderungen: Wann, wer, von→zu
  - Zuweiser-Änderungen: Wann, wer, von Techniker A zu B
  - Kommentare: Datum, Autor, Text
  - Datei-Uploads: Datum, Dateiname, Größe
  - SLA-Events: Warnung ausgelöst, überschritten, pausiert
  - Verknüpfungen: Andere Tickets verknüpft
- **Filter**: Nur interne Notizen zeigen oder nur Kundenkommunikation
- **Export**: "Ticket-Geschichte als PDF" Button
- **Audit**: Alle Änderungen sind unveränderlich und zeigen Benutzer

9. Hilfe (Help/Utilities)

9.1 Dashboard (Personal Dashboard)

Module Path: `src/centron/Centron.WPF.UI/Modules/MyCentron/Dashboard`
Controller: `CentronDashboardAppModuleController`
ViewModel: `CentronDashboardViewModel`
Category: MyCentron
Description: Persönliches Dashboard mit konfigurierbaren Kacheln für Kennzahlen, Aufgaben, Tickets und Benachrichtigungen

Use Cases

9.1.1 Dashboard-Kacheln konfigurieren

Zweck: Personalisierung der Dashboard-Ansicht mit relevanten Informationen

Ablauf:

1. Benutzer öffnet Dashboard und klickt auf "Kacheln verwalten"
2. Verfügbare Widgets werden angezeigt: Meine Tickets, Offene Aufgaben, OPOS, Umsatz-KPIs
3. Benutzer fügt gewünschte Kacheln hinzu per Drag & Drop
4. Größe und Position der Kacheln werden angepasst
5. Aktualisierungsintervall wird konfiguriert (Echtzeit, 5 Min, 15 Min, Manuell)
6. Layout wird gespeichert und beim nächsten Login wiederhergestellt

Betroffene Felder: DashboardConfiguration (UserId3D, TileType, Position, Size, RefreshInterval)

Auswirkungen: Personalisierte Arbeitsumgebung. Wichtige Informationen auf einen Blick. Effizienter Arbeitsbeginn.

9.1.2 Echtzeit-Benachrichtigungen empfangen

Zweck: Push-Benachrichtigungen für wichtige Ereignisse im Dashboard

Ablauf:

1. System überwacht konfigurierte Trigger: Neues Ticket zugewiesen, OPOS-Erinnerung, Genehmigung erforderlich
2. Bei Ereignis erscheint Benachrichtigungskachel im Dashboard mit Anzahl
3. Benutzer klickt auf Kachel und sieht Details
4. Direktaktion möglich: Ticket öffnen, Rechnung prüfen, Genehmigung erteilen
5. Benachrichtigungen können als gelesen markiert oder verworfen werden

Betroffene Felder: Notification (Type, Title, Message, Priority, IsRead, TargetObjectI3D)

Auswirkungen: Keine wichtigen Ereignisse werden verpasst. Schnelle Reaktionszeiten. Proaktives Arbeiten möglich.

9.1.3 KPI-Kacheln auswerten

Zweck: Visualisierung von Geschäftskennzahlen und Trends im Dashboard

Ablauf:

1. Benutzer fügt KPI-Kachel hinzu: Umsatz, Marge, Ticket-Volumen, OPOS
2. Zeitraum wird gewählt: Heute, Woche, Monat, Quartal, Jahr
3. System berechnet Kennzahl und zeigt Trend an (Sparkline)
4. Vergleich zum Vorjahr oder Zielwert wird visualisiert (grün/gelb/rot)
5. Klick auf Kachel öffnet Detailreport mit Drill-Down-Möglichkeit
6. Export als PDF oder Excel möglich

Betroffene Felder: KPI (Type, Value, TargetValue, ComparisonValue, TrendDirection, Period)

Auswirkungen: Geschäftsentwicklung ist transparent. Abweichungen werden schnell erkannt. Datenbasierte Entscheidungen möglich.

9.1.4 Schnellzugriffe verwenden

Zweck: Direktlinks zu häufig genutzten Modulen und Funktionen

Ablauf:

1. Benutzer fügt Quick-Action-Kachel hinzu
2. Aktionen werden konfiguriert: "Neues Ticket", "Rechnung erfassen", "Kunde anlegen"
3. Kachel zeigt Icons mit Beschreibung an
4. Ein Klick öffnet direkt das Zielmodul mit vorbelegten Werten
5. Häufigkeit der Nutzung wird getrackt
6. Am meisten genutzte Aktionen werden automatisch hervorgehoben

Betroffene Felder: QuickAction (Name, TargetModule, Parameters, UsageCount, LastUsed)

Auswirkungen: Workflows werden beschleunigt. Navigation ist effizienter. Mehrfachklicks werden vermieden.

9.2 Kalender (Calendar)

Module Path: `src/centron/Centron.WPF.UI/Modules/MyCentron/Calendar`
Controller: `CalenderAppController`
ViewModel: `CalenderViewModel`
Category: MyCentron
Description: Terminkalender für Meetings, Kundenbesuche und interne Appointments mit Team-Kalenderansicht

Use Cases

9.2.1 Termin anlegen

Zweck: Erfassung neuer Termine mit Teilnehmern und Ressourcen

Ablauf:

1. Benutzer klickt auf Zeitslot im Kalender oder "Neuer Termin"
2. Termindaten werden erfasst: Titel, Start-/Endzeit, Ort
3. Teilnehmer werden hinzugefügt (interne Mitarbeiter + externe Kontakte)
4. Raum oder Ressource wird gebucht (Besprechungsraum, Fahrzeug)
5. Terminart wird gewählt: Meeting, Kundenbesuch, Telefon, Schulung
6. Optional: Erinnerung, Wiederholung, Verknüpfung zu Ticket/Projekt
7. Termin wird gespeichert, Teilnehmer erhalten Einladung per E-Mail

Betroffene Felder: CalendarEvent (Title, StartDateTime, EndDateTime, Location, Type, Participants, ResourceI3D)

Auswirkungen: Termine sind zentral organisiert. Teilnehmer werden automatisch informiert. Doppelbuchungen werden verhindert.

9.2.2 Team-Kalender einsehen

Zweck: Überblick über Verfügbarkeit von Kollegen für Terminplanung

Ablauf:

1. Benutzer wechselt zu Team-Ansicht im Kalender
2. Mitarbeiter werden ausgewählt (Abteilung, Projekt-Team)
3. System zeigt Kalender aller gewählten Personen nebeneinander
4. Freie Zeitslots werden grün hervorgehoben
5. Benutzer findet passenden Zeitpunkt für gemeinsamen Termin
6. Termin wird direkt aus Team-Ansicht erstellt

Betroffene Felder: TeamCalendarView (SelectedEmployees, DisplayMode, DateRange)

Auswirkungen: Terminabstimmung wird vereinfacht. Verfügbarkeiten sind transparent. Planungsaufwand sinkt.

9.2.3 Wiederkehrende Termine verwalten

Zweck: Automatische Erstellung von regelmäßigen Terminen

Ablauf:

1. Benutzer erstellt Termin und aktiviert "Wiederholen"
2. Wiederholungsregel wird definiert: Täglich, Wöchentlich (Mo/Mi/Fr), Monatlich (1. Montag)
3. Ende der Wiederholung wird festgelegt: Nach X Terminen oder Enddatum
4. Ausnahmen können definiert werden (z.B. Feiertage ausschließen)
5. System generiert Terminserie automatisch
6. Änderungen können auf einzelne Termine oder gesamte Serie angewendet werden

Betroffene Felder: CalendarEvent (RecurrenceRule, RecurrenceEndDate, RecurrenceExceptions)

Auswirkungen: Wiederkehrende Termine müssen nur einmal erfasst werden. Wartungsintervalle werden nicht vergessen. Planung ist langfristig sichtbar.

9.3 c-entron Inspectors (System Health Checks)

Module Path: `src/centron/Centron.WPF.UI/Modules/MyCentron/CentronInspectors`
Controller: `CentronInspectorAppModuleController`
ViewModel: `CentronInspectorViewModel`
Category: MyCentron
Description: System-Health-Checks und Datenintegritätsprüfungen für Administratoren zur Fehlerdiagnose

Use Cases

9.3.1 Datenintegrität prüfen

Zweck: Identifikation von inkonsistenten Datensätzen und Referenzen

Ablauf:

1. Administrator öffnet c-entron Inspectors
2. Prüfung wird ausgewählt: Orphan-Records, Duplikate, Fehlerhafte FKs
3. System durchsucht Datenbank nach konfigurierten Anomalien
4. Ergebnisse werden tabellarisch angezeigt mit Schweregrad (Warnung/Fehler)
5. Details können eingesehen werden: Betroffene Tabellen, Datensatz-IDs
6. Korrekturaktion kann ausgeführt werden: Löschen, Zusammenführen, Referenz korrigieren
7. Prüfbericht wird generiert und archiviert

Betroffene Felder: InspectorCheck (CheckType, ExecutionDate, FoundIssues, ResolvedIssues, Status)

Auswirkungen: Datenqualität wird überwacht. Anomalien werden frühzeitig erkannt. Systemstabilität steigt.

9.3.2 Performance-Checks durchführen

Zweck: Analyse von Systemperformance und Identifikation von Bottlenecks

Ablauf:

1. Admin startet Performance-Inspector
2. System analysiert: Langsame SQL-Queries, Speicherbelegung, API-Response-Zeiten
3. Problematische Queries werden angezeigt mit Laufzeit und Ausführungshäufigkeit
4. Empfohlene Indizes werden vorgeschlagen
5. Admin kann Index-Erstellung direkt anstoßen oder für Wartungsfenster planen
6. Trend-Analyse zeigt Performance-Entwicklung über Zeit

Betroffene Felder: PerformanceCheck (QueryText, ExecutionTime, ExecutionCount, RecommendedIndex)

Auswirkungen: Systemgeschwindigkeit wird optimiert. Benutzererfahrung verbessert sich. Ressourcen werden effizienter genutzt.

9.3.3 Konfigurationsprobleme erkennen

Zweck: Validierung von Systemeinstellungen und Integrationskonfigurationen

Ablauf:

1. Admin wählt Konfigurations-Inspector
2. System prüft: E-Mail-Einstellungen, API-Verbindungen, Lizenzstatus, Buchhaltungsexporte
3. Für jede Konfiguration wird Status angezeigt: OK, Warnung, Fehler
4. Bei Fehlern werden konkrete Hinweise gegeben: "SMTP-Server nicht erreichbar"
5. Test-Funktion steht zur Verfügung: Test-E-Mail versenden, API-Ping
6. Fehler können direkt aus Inspector heraus korrigiert werden

Betroffene Felder: ConfigurationCheck (ConfigurationType, Status, LastChecked, ErrorMessage, TestResult)

Auswirkungen: Fehlkonfigurationen werden schnell erkannt. Ausfallzeiten werden minimiert. Support-Aufwand sinkt.

9.4 MyDay Editor (Time Entry)

Module Path: `src/centron/Centron.WPF.UI/Modules/MyCentron/MyDay/Editor`
Controller: `MyDayEditorAppModuleController`

ViewModel: `MyDayEditorViewModel`
Category: MyCentron
Description: Zeiterfassung für Mitarbeiter zur Dokumentation von Arbeitsstunden auf Projekte, Tickets und Tätigkeiten

Use Cases

9.4.1 Arbeitszeiten erfassen

Zweck: Tägliche Erfassung von Arbeitsstunden pro Projekt und Tätigkeit

Ablauf:

1. Mitarbeiter öffnet MyDay-Editor und wählt Datum
2. Zeiteinträge werden erfasst: Ticket-Nummer, Tätigkeit, Start-/Endzeit oder Dauer
3. Projektzuordnung erfolgt automatisch durch Ticket-Verknüpfung
4. Tätigkeitsart wird gewählt: Programmierung, Support, Meeting, Reise
5. Optionaler Kommentar kann hinzugefügt werden
6. System prüft auf Überschneidungen und warnt bei >10 Stunden/Tag
7. Tag wird abgeschlossen und zur Genehmigung freigegeben

Betroffene Felder: TimeEntry (EmployeeI3D, Date, TicketI3D, ActivityType, StartTime, EndTime, Duration, Comment)

Auswirkungen: Arbeitszeiten sind transparent. Projektkostenrechnung ist präzise. Abrechnung basiert auf realen Zeiten.

9.4.2 Pauschale Tätigkeiten buchen

Zweck: Erfassung von nicht-projektbezogenen Tätigkeiten

Ablauf:

1. Mitarbeiter wählt "Pauschale Tätigkeit" im MyDay-Editor
2. Tätigkeitsart wird ausgewählt: Urlaub, Krankheit, Fortbildung, Intern
3. Ganztags oder Stunden werden erfasst
4. Bei Urlaub/Krankheit: Antrag-Referenz wird verknüpft
5. Bei Fortbildung: Schulungs-ID wird zugeordnet
6. Eintrag wird gespeichert und ist in Berichten sichtbar

Betroffene Felder: TimeEntry (ActivityType=NonBillable, FullDay, AbsenceType, TrainingI3D)

Auswirkungen: Vollständige Zeiterfassung. Personalkosten werden korrekt verteilt. Urlaubsverwaltung ist integriert.

9.4.3 Zeiteinträge genehmigen

Zweck: Freigabe von Arbeitsstunden durch Vorgesetzte

Ablauf:

1. Vorgesetzter öffnet Genehmigungsansicht im MyDay-Editor
2. Offene Zeiteinträge der Mitarbeiter werden angezeigt
3. Einträge werden geprüft: Plausibilität, Projektzuordnung, Dauer
4. Bei Unstimmigkeiten: Rückfrage an Mitarbeiter mit Kommentar
5. Korrekte Einträge werden genehmigt (Bulk-Genehmigung möglich)
6. Genehmigte Stunden sind nun für Abrechnung und Controlling freigegeben

Betroffene Felder: TimeEntry (Status=Approved, ApprovedByI3D, ApprovalDate, ApprovalComment)

Auswirkungen: Vier-Augen-Prinzip bei Zeiterfassung. Fehlerhafte Buchungen werden korrigiert. Abrechnungsqualität steigt.

9.5 Employee Overview (Workload View)

Module Path: `src/centron/Centron.WPF.UI/Modules/MyCentron/MyDay/EmployeeOverview`
Controller: `MyDayEmployeeOverviewAppModuleController`
ViewModel: `MyDayEmployeeOverviewViewModel`
Category: MyCentron
Description: Übersicht über Mitarbeiterauslastung, offene Tasks und Verfügbarkeiten für Ressourcenplanung

Use Cases

9.5.1 Teamauslastung monitoren

Zweck: Überblick über Arbeitsbelastung aller Mitarbeiter

Ablauf:

1. Teamleiter öffnet Employee Overview
2. Mitarbeiter werden angezeigt mit Auslastung in Prozent (diese Woche)
3. Farbcodierung zeigt Status: Grün (<80%), Gelb (80-100%), Rot (>100%)
4. Details zeigen geplante vs. gebuchte Stunden
5. Filter nach Abteilung, Standort oder Skill möglich
6. Drill-Down zeigt konkrete Tickets und Projekte pro Mitarbeiter

Betroffene Felder: EmployeeWorkload (EmployeeI3D, PlannedHours, BookedHours, UtilizationPercentage, Week)

Auswirkungen: Überlastung wird erkannt. Ressourcen können umverteilt werden. Burnout-Prävention.

9.5.2 Verfügbarkeiten prüfen

Zweck: Identifikation freier Kapazitäten für neue Projekte

Ablauf:

1. Projektleiter öffnet Verfügbarkeitsansicht
2. Zeitraum wird gewählt (nächste 2 Wochen)
3. System zeigt Mitarbeiter mit freien Kapazitäten
4. Skills werden berücksichtigt: "Suche Java-Entwickler mit 10h frei"
5. Geplanter Urlaub wird visualisiert
6. Direkte Ticketzuweisung aus Übersicht möglich

Betroffene Felder: EmployeeAvailability (EmployeeI3D, Date, AvailableHours, PlannedAbsence, Skills)

Auswirkungen: Neue Projekte können realistische geplant werden. Ressourcen werden optimal genutzt. Liefertermine sind verlässlich.

9.5.3 Überstunden tracken

Zweck: Erfassung und Ausgleich von Mehrarbeit

Ablauf:

1. Personalabteilung öffnet Überstunden-Report
2. Mitarbeiter mit Überstunden werden angezeigt (Saldo pro Monat)
3. Trend wird visualisiert: Steigend, Konstant, Abnehmend
4. Ausgleichsoptionen werden vorgeschlagen: Freizeitausgleich, Auszahlung
5. Mitarbeiter kann Freizeitausgleich direkt beantragen
6. HR genehmigt und plant Freizeit ein

Betroffene Felder: OvertimeBalance (EmployeeI3D, Month, Hours, CompensationType, CompensationDate)

Auswirkungen: Überstunden werden nicht vergessen. Gesetzliche Anforderungen werden erfüllt. Mitarbeiterzufriedenheit steigt.

9.6 Month Review (Monthly Review)

Module Path: `src/centron/Centron.WPF.UI/Modules/MyCentron/MyDay/MonthReview`
Controller: `MyDayMonthReviewAppModuleController`
ViewModel: `MyDayMonthReviewViewModel`
Category: MyCentron
Description: Monatsabschluss für Zeiterfassung mit Übersicht über gebuchte Stunden, Projekte und Abweichungen

Use Cases

9.6.1 Monatsbericht generieren

Zweck: Zusammenfassung aller Zeitbuchungen eines Monats pro Mitarbeiter

Ablauf:

1. Mitarbeiter öffnet Month Review und wählt Monat
2. System zeigt Zusammenfassung: Gesamt-Stunden, Projekte, Tätigkeitsverteilung
3. Vergleich zu Sollstunden: 160h Soll, 168h Ist = +8h Überstunden
4. Projekte werden aufgeschlüsselt mit Stunden pro Projekt
5. Nicht-projektbezogene Zeiten werden separat ausgewiesen
6. Export als PDF für Personalakte möglich

Betroffene Felder: MonthlyReport (EmployeeI3D, Month, TotalHours, TargetHours, OvertimeHours, Projects)

Auswirkungen: Transparenz über geleistete Arbeit. Basis für Gehaltsabrechnung. Nachweis für Auditoren.

9.6.2 Fehlende Zeiteinträge identifizieren

Zweck: Prüfung auf Vollständigkeit der Zeiterfassung

Ablauf:

1. Mitarbeiter öffnet Month Review
2. System markiert Tage ohne Zeiteinträge (außer Wochenende/Urlaub)
3. Warnung erscheint: "5 Arbeitstage ohne Zeiterfassung"
4. Benutzer kann fehlende Tage direkt nacherfassen
5. Nach Vervollständigung verschwindet Warnung
6. Monat kann erst nach Vollständigkeit abgeschlossen werden

Betroffene Felder: MonthCompleteness (EmployeeI3D, Month, MissingDays, IsComplete)

Auswirkungen: Lückenlose Zeiterfassung. Projektkosten sind vollständig. Compliance-Anforderungen erfüllt.

9.6.3 Monat abschließen

Zweck: Finalisierung des Monats nach Prüfung aller Einträge

Ablauf:

1. Mitarbeiter prüft Month Review auf Vollständigkeit und Korrektheit
2. Button "Monat abschließen" wird geklickt
3. System sperrt alle Zeiteinträge dieses Monats gegen Änderungen
4. Vorgesetzter erhält Benachrichtigung zur finalen Freigabe
5. Nach Freigabe: Daten werden an Lohnbuchhaltung übermittelt

6. Änderungen sind nur noch mit Admin-Rechten möglich

Betroffene Felder: MonthStatus (EmployeeI3D, Month, Status=Closed, ClosedDate, ApprovedByI3D)

Auswirkungen: Monatsdaten sind verbindlich. Nachträgliche Manipulationen werden verhindert. Audit-Trail vorhanden.

9.7 Telephony Call Log (Call History)

Module Path: `src/centron/Centron.WPF.UI/Modules/MyCentron/Telephony/CallLog`
Controller: `TelephonyCallLogAppModuleController`
ViewModel: `TelephonyCallLogViewModel`
Category: MyCentron
Description: Anrufliste mit Integration zu Telefonanlagen für automatische Erfassung von Anrufen und Verknüpfung zu Kunden

Use Cases

9.7.1 Anrufe automatisch erfassen

Zweck: Automatisches Logging aller ein- und ausgehenden Anrufe

Ablauf:

1. Telefonanlage (z.B. Starface, 3CX) ist mit c-entron verbunden
2. Bei Anruf: System erfasst automatisch Rufnummer, Zeitpunkt, Dauer
3. Rufnummer wird abgeglichen mit Kundenstamm (Kontakte)
4. Kunde wird automatisch verknüpft, Name wird angezeigt
5. Bei unbekannten Nummern: "Unbekannter Anrufer" mit Option "Kunde zuordnen"
6. Anrufhistorie ist in Echtzeit verfügbar

Betroffene Felder: CallLog (PhoneNumber, Direction, StartTime, Duration, ContactI3D, EmployeeI3D, Status)

Auswirkungen: Alle Anrufe sind dokumentiert. Rückfragen zu "Wer hat angerufen?" entfallen. Compliance für Datenschutz.

9.7.2 Anrufe zu Tickets verknüpfen

Zweck: Dokumentation von Anrufen als Teil der Ticket-Historie

Ablauf:

1. Mitarbeiter öffnet Call Log nach Telefonat
2. Anruf wird ausgewählt aus Liste
3. "Zu Ticket hinzufügen" wird geklickt
4. Bestehendes Ticket kann gewählt oder neues erstellt werden
5. Optionaler Kommentar wird hinzugefügt: "Kunde meldet Drucker-Problem"
6. Anruf erscheint in Ticket-Historie mit Link zum Call-Log-Eintrag

Betroffene Felder: CallLog (TicketI3D, Comment, LinkedDate), Ticket (CallLogEntries)

Auswirkungen: Vollständige Kommunikationshistorie. Support-Kontext ist nachvollziehbar. Keine Informationsverluste.

9.7.3 Anrufstatistiken auswerten

Zweck: Analyse von Anrufaufkommen und Reaktionszeiten

Ablauf:

1. Manager öffnet Call Log-Statistik
2. Zeitraum wird gewählt (Woche, Monat)

3. KPIs werden angezeigt: Gesamt-Anrufe, Angenommen, Verpasst, Ø Dauer
4. Verteilung nach Mitarbeitern: "Max Müller: 45 Anrufe (23 ein, 22 aus)"
5. Peak-Zeiten werden visualisiert: "Meiste Anrufe 10-12 Uhr"
6. Verpasste Anrufe werden hervorgehoben zur Nachverfolgung

Betroffene Felder: CallStatistics (Period, TotalCalls, AnsweredCalls, MissedCalls, AverageDuration, PeakHour)

Auswirkungen: Personalplanung wird optimiert. Verpasste Anrufe werden reduziert. Service-Level steigt.

9.8 Todo List (Task Management)

Module Path: `src/centron/Centron.WPF.UI/Modules/MyCentron/TodoList`
Controller: `TodoListAppController`
ViewModel: `TodoListViewModel`
Category: MyCentron
Description: Persönliche Aufgabenliste mit Priorisierung, Fälligkeitsdaten und Verknüpfung zu Tickets und Projekten

Use Cases

9.8.1 Aufgaben erstellen und priorisieren

Zweck: Erfassung persönlicher Todos mit Fälligkeitsdatum

Ablauf:

1. Benutzer öffnet Todo-Liste und klickt "Neue Aufgabe"
2. Titel und Beschreibung werden erfasst
3. Fälligkeitsdatum wird gesetzt (optional mit Erinnerung)
4. Priorität wird gewählt: Niedrig, Normal, Hoch, Kritisch
5. Optional: Verknüpfung zu Kunde, Ticket oder Projekt
6. Aufgabe wird gespeichert und erscheint in sortierter Liste

Betroffene Felder: TodoItem (Title, Description, DueDate, Priority, LinkedObjectType, LinkedObjectI3D, IsCompleted)

Auswirkungen: Persönliche Organisation. Aufgaben werden nicht vergessen. Priorisierung ist klar.

9.8.2 Fällige Aufgaben anzeigen

Zweck: Übersicht über heute und überfällige Aufgaben

Ablauf:

1. Todo-Liste öffnet automatisch mit Filter "Fällig heute"
2. Überfällige Aufgaben werden rot markiert
3. Aufgaben für diese Woche werden in separater Sektion angezeigt
4. Benutzer kann Aufgaben als erledigt markieren (Checkbox)
5. Erledigte Aufgaben verschwinden aus Hauptansicht
6. Dashboard-Kachel zeigt Anzahl fälliger Aufgaben

Betroffene Felder: TodoItem (DueDate, IsOverdue, CompletedDate)

Auswirkungen: Fokus auf Wesentliches. Deadlines werden eingehalten. Produktivität steigt.

9.8.3 Wiederkehrende Aufgaben einrichten

Zweck: Automatische Erstellung regelmäßiger Todos

Ablauf:

1. Benutzer erstellt Aufgabe und aktiviert "Wiederkehrend"
2. Wiederholungsintervall wird gewählt: Täglich, Wöchentlich, Monatlich
3. Beispiele: "Backup prüfen" (täglich), "Monatsabschluss" (monatlich am 1.)
4. System erstellt neue Todo-Instanz automatisch nach Erledigung
5. Vorlauf-Zeit kann konfiguriert werden (3 Tage vor Fälligkeit erstellen)
6. Serie kann jederzeit gestoppt oder angepasst werden

Betroffene Felder: TodoItem (IsRecurring, RecurrenceRule, NextCreationDate)

Auswirkungen: Routineaufgaben werden automatisiert. Checklisten sind immer aktuell. Zuverlässigkeit steigt.

10. Logistik (Logistics)

10.1 Artikelimport (Article Import)

Module Path: `src/centron/Centron.WPF.UI/Modules/Warehousing/ArticleImport`
Controller: `ArticleImportAppModuleController`
ViewModel: `ArticleImportAppModuleViewModel`
Category: Logistik
Description: Automatisierte Importierung von Artikelnummern und Produktdaten aus externen Quellen (Distributoren, Lieferanten) mit flexibler Feldmapping und Formatunterstützung

Use Cases

10.1.1 Neue Artikel-Import-Konfiguration erstellen

Zweck: Einrichtung von Importquellen für externe Artikeldaten (FTP, HTTP, Dateien)

Ablauf:

1. Benutzer öffnet Artikelimport-Modul
2. Button "Neue Import-Quelle" wird geklickt
3. Dialog mit Optionen: Importtyp wählen (Single File, TechData, Wortmann, etc.)
4. Quelle konfigurieren: FTP-Server, HTTP-URL, oder lokale Datei
5. Authentifizierung einrichten (Benutzer, Passwort, Zertifikat)
6. Import-Einstellungen setzen: Trennzeichen (;,|), Dezimalzeichen, Kodierung
7. Komprimierungsformat angeben falls nötig (ZIP, GZIP)
8. Konfiguration wird gespeichert

Betroffene Felder: ArticleImport (ImportType, SourceType, Delimiter, DecimalCharacter, Encoding, CompressionType)

Auswirkungen: Externe Datenquellen sind jetzt konfiguriert. Regelmäßige Importe können zeitgesteuert erfolgen. Artikeldaten werden automatisch aktualisiert.

10.1.2 Feldmapping und Datenformate konfigurieren

Zweck: Zuordnung von Import-Feldern zu Centron-Artikelfeldern

Ablauf:

1. Benutzer öffnet bestehende Import-Konfiguration
2. Tab "Feldmapping" wird angewählt
3. System zeigt verfügbare Import-Felder (aus Beispieldatei gelesen)
4. Benutzer ordnet Felder zu:

- Externe "Product Code" → Centron "Article Number"
- Externe "Description" → Centron "Article Name"
- Externe "Price" → Centron "Purchase Price"

5. System zeigt Vorschau der Zuordnung
6. Validierungsregeln können konfiguriert werden (z.B. "Price muss numerisch sein")
7. Mapping wird gespeichert

Betroffene Felder: ArticleImportFieldsAssign (ExternalFieldName, CentronFieldName, DataType, ValidationRule)

Auswirkungen: Flexible Import-Strukturen werden unterstützt. Verschiedene Lieferanten-Formate können verarbeitet werden. Datenqualität wird durch Validierung sichergestellt.

10.1.3 Import-Dateien verarbeiten und Vorschau anzeigen

Zweck: Durchführung von Dateien-Import mit Vorschau vor finalem Import

Ablauf:

1. Benutzer lädt Import-Datei hoch (manuell oder von FTP geholt)
2. System liest erste Zeilen der Datei
3. Dateiformat wird automatisch erkannt
4. System zeigt Vorschau: erste 10-20 Zeilen mit Feldmapping
5. Benutzer validiert Vorschau auf Plausibilität
6. Duplikat-Check wird durchgeführt (existiert Artikel schon?)
7. Import-Statistik wird angezeigt: "X neue Artikel, Y Updates, Z Fehler"
8. Nach Bestätigung wird Import durchgeführt

Betroffene Felder: ArticleImportFileName (FileName, FileSize, ImportDate), ArticleImportLog (Status, RecordsProcessed, RecordsSuccessful, RecordsFailed)

Auswirkungen: Fehler werden vor Import erkannt. Benutzer hat Kontrolle über Datenqualität. Import-Prozess wird transparent und nachvollziehbar.

10.1.4 Artikelmasterdaten aus Distributoren-Katalogen importieren

Zweck: Automatisierte Übernahme von Artikel-Spezifikationen und Katalog-Informationen

Ablauf:

1. Import lädt externe Artikel-Daten (z.B. von TechData-Katalog)
2. System verarbeitet für jeden Artikel:
   - Standardisierte Produktspezifikationen
   - Herkunftsinformationen (Distributor, EAN)
   - Verfügbare Varianten und Optionen
3. System enriches lokale Artikel-Daten mit:
   - Zusätzliche Attribute (Farben, Größen, etc.)
   - Kompatibilitäts-Informationen
   - Cross-Sell/Up-Sell-Empfehlungen
4. Artikel werden als "Mit Distributor verlinkt" markiert
5. Zukünftige Updates nutzen diese Verknüpfung
6. Artikel sind sofort in Vertriebssystemen verfügbar

Betroffene Felder: ExternalArticle (ArticleI3D, DistributorArticleNumber, CatalogData, AttributeMapping), Article (LinkedDistributorArticles)

Auswirkungen: Produktdaten werden angereichert automatisch. Handelsunternehmen haben vollständige Kataloge. Verkaufsuntersützung wird verbessert durch zusätzliche Daten.

10.1.5 Import-Fehlerbehandlung und Fehler-Reports

Zweck: Systematische Behandlung und Dokumentation von Import-Fehlern

Ablauf:

1. Bei Fehler während Import: System dokumentiert Fehlertyp
2. Fehlerbeispiele:
   - Ungültige Dateiformat
   - Fehlende Pflichtfelder
   - Ungültige Datentypen
   - Duplikate
3. Fehler werden klassifiziert: Kritisch, Major, Minor
4. System erzeugt Fehler-Report mit Details
5. Benutzer kann Report anschauen und Filter-Fehler analysieren
6. Option: Fehlerhafte Zeilen ignorieren und mit Rest weitermachen
7. Vollständiger Fehler-Audit wird gespeichert

Betroffene Felder: ArticleImportLog (ErrorType, ErrorSeverity, ErrorMessage, LineNumber, OffendingData), ArticleImportErrorReport

Auswirkungen: Import-Ausfallrate sinkt. Probleme werden schnell identifiziert und behoben. Datenqualität wird überwacht.

10.1.6 Import-Automatisierung und Zeitplanung

Zweck: Regelmäßige automatisierte Importe von Artikel-Updates

Ablauf:

1. Benutzer öffnet Import-Konfiguration
2. Tab "Automatisierung" wird angewählt
3. Zeitplan wird konfiguriert: täglich, wöchentlich, monatlich, bei Bedarf
4. Import-Zeit wird gesetzt (z.B. täglich um 2:00 Uhr nachts)
5. Benachrichtigungen werden konfiguriert:
   - Bei Erfolg: Email an Management?
   - Bei Fehler: Alarm an Admin?
6. Automatische Fehlerbehebung einrichten (z.B. Retry bei Timeout)
7. Import-Historie wird verfolgt (wann lief Import, wie viele Artikel)
8. Nach aktiver Automatisierung läuft Import im Hintergrund

Betroffene Felder: ArticleImport (IsAutomated, SchedulePattern, ScheduledTime, NotificationSettings), ArticleImportLog (ScheduledImportI3D, ExecutionTime)

Auswirkungen: Manuelle Importe entfallen. Artikeldaten sind ständig aktuell. Verwaltungsaufwand sinkt dramatisch. Business-Entscheidungen basieren auf aktuellen Daten.

10.2 Artikelverwaltung (Article Management)

Module Path: `src/centron/Centron.WPF.UI/Modules/Warehousing/ArticleManagement`
Controller: `ArticleManagementAppModuleController`
ViewModel: `ArticleManagementMainViewModel`
Category: Logistik
Description: Zentrale Verwaltung von Artikeln mit Spezifikationen, Barcodes, Preisen, Lagerbeständen und Umrechnungseinheiten

Use Cases

10.2.1 Neuen Artikel erstellen

Zweck: Erfassung neuer Artikel mit Stammdaten und grundlegenden Informationen

Ablauf:

1. Benutzer navigiert zu "Neuer Artikel"
2. Dialog mit Grunddaten wird geöffnet: Artikelnummer, Artikelname, Beschreibung
3. Klassifikation wird zugeordnet: Material-Gruppe, Kategorie, Produkttyp
4. Einheit wird gewählt (Stück, Meter, Liter, etc.)
5. Standard-Einkaufspreis wird eingegeben
6. Barcode kann generiert oder manuell eingegeben werden
7. Optional: Bilder und Attachments hinzufügen
8. Artikel wird als "Draft" erstellt
9. Nach Genehmigung ist Artikel verfügbar

Betroffene Felder: Article (ArticleNumber, ArticleName, Description, MaterialGroupI3D, PurchasePrice, UnitI3D, BarcodeI3D)

Auswirkungen: Artikel ist im System verfügbar. Sofort in Einkauf und Verkauf einsetzbar. Kann in Bestellungen und Angeboten verwendet werden.

10.2.2 Artikel-Spezifikationen und technische Details hinzufügen

Zweck: Dokumentation von technischen Eigenschaften und Produktspezifikationen

Ablauf:

1. Benutzer öffnet Artikel im Bearbeitungsmodus
2. Tab "Spezifikationen" wird angewählt
3. System zeigt vordefinierte Eigenschaften (Größe, Farbe, Gewicht, Material, etc.)
4. Benutzer kann Werte eingeben oder aus Dropdown wählen
5. Benutzer kann benutzerdefinierte Eigenschaften hinzufügen
6. Technische Datenblätter können hochgeladen werden (PDF)
7. Video-Links können hinzugefügt werden
8. Alle Spezifikationen werden gespeichert und sind durchsuchbar

Betroffene Felder: ArticleSpecification (ArticleI3D, PropertyName, PropertyValue, DataType, CustomProperty)

Auswirkungen: Vollständige Produktdokumentation. Kunden finden Artikel schneller. Support-Team hat alle Informationen zur Hand. SEO wird verbessert.

10.2.3 Barcode-Verwaltung und Generierung

Zweck: Verwaltung von Barcodes für Artikel-Verfolgung und Verkauf

Ablauf:

1. Benutzer öffnet Artikel
2. Tab "Barcodes" wird angewählt
3. System zeigt existierende Barcodes
4. Button "Neuer Barcode" wird geklickt
5. Barcode-Typ wird gewählt: EAN-13, EAN-8, CODE-128, QR-Code
6. System kann Barcode automatisch generieren oder manuell eingeben
7. Barcode wird validiert (Prüfsumme, Länge)
8. Varianten-Barcodes können für unterschiedliche Packungsgrößen erstellt werden
9. Barcode wird in Label-Format für Druck exportiert
10. Historische Barcodes werden archiviert (falls Artikel aktualisiert)

Betroffene Felder: ArticleBarcode (ArticleI3D, BarcodeType, BarcodeValue, IsActive, CreatedDate, VariantI3D)

Auswirkungen: Lagerverwaltung wird beschleunigt durch Barcode-Scanning. Bestandsverfolgung wird automatisiert. Bestellgenauigkeit verbessert sich.

10.2.4 Preise und Einkaufskonditionen verwalten

Zweck: Verwaltung von Einkaufs- und Verkaufspreisen sowie Lieferanten-Konditionen

Ablauf:

1. Benutzer öffnet Artikel
2. Tab "Einkauf (EK)" wird angewählt
3. System zeigt Einkaufspreise nach Lieferant
4. Benutzer kann für jeden Lieferant festlegen:
   - Standard-EK-Preis
   - Mengen-Rabatte (z.B. ab 100 Stück -5%)
   - Lieferzeit (Lead Time)
   - Mindestbestellmenge
5. Währung kann festgelegt werden (EUR, CHF, etc.)
6. Preishistorie wird verfolgt (Alte Preise archiviert)
7. Automatische Preisübernahmen aus EDI-Importen konfigurierbar
8. Benutzer kann Verkaufspreise aus Einkaufspreisen berechnen (mit Marge)

Betroffene Felder: ArticlePriceMatrix (ArticleI3D, SupplierI3D, Quantity, Price, DiscountPercentage, LeadTime, MinimumOrderQuantity)

Auswirkungen: Einkaufspreise sind kompetitiv. Kalkulation wird automatisiert. Rabatte werden genutzt. Gewinnmargen werden kontrolliert.

10.2.5 Einheits-Umrechnung und Varianten

Zweck: Verwaltung von Artikel-Varianten und Einheits-Konvertierungen

Ablauf:

1. Benutzer öffnet Artikel
2. Tab "Varianten & Einheiten" wird angewählt
3. Benutzer kann Basis-Einheit definieren (z.B. "1 Palette = 80 Kartons")
4. Alternative Einheiten können hinzugefügt werden
5. Umrechnungsfaktoren werden eingegeben
6. Artikel-Varianten können erstellt werden:
   - Größe (S, M, L, XL)
   - Farbe (Rot, Blau, Grün)
   - Konfiguration (Premium, Standard)
7. Jede Variante erhält eigene Artikelnummer
8. Lagerbestände werden pro Variante verwaltet
9. Preise können pro Variante unterschiedlich sein

Betroffene Felder: ArticleUnitManagement (ArticleI3D, BaseUnit, AlternativeUnit, ConversionFactor), ArticleVariant (VariantDescription, ArticleNumberI3D, SizeI3D, ColorI3D)

Auswirkungen: Komplexe Artikel-Strukturen werden bewältigt. Lageroptimierung wird präziser. Verkauf und Einkauf nutzen unterschiedliche Einheiten problemlos.

10.2.6 Lagerbestand-Management und Schwellenwerte

Zweck: Überwachung und Kontrolle von Lagerbeständen mit automatischen Nachbestellungen

Ablauf:

1. Benutzer öffnet Artikel

2. Tab "Lagerbestand" wird angewählt
3. System zeigt aktuellen Bestand nach Lagerort
4. Benutzer kann Schwellenwerte definieren:
    - Minimumbestand (Meldebestand): z.B. 50 Stück
    - Maximumbestand: z.B. 500 Stück
5. Reorder-Point wird berechnet: Mindestbestand + Sicherheitsbestand
6. Bei Unterschreitung wird automatische Nachbestellung getriggert
7. Lagerbestand wird in Echtzeit aktualisiert bei Verkauf/Einkauf
8. Veralten-Datum kann gesetzt werden für verderbliche Waren
9. Überlagerungswarnung wird gezeigt bei zu alten Beständen

Betroffene Felder: Article (MinimumStock, MaximumStock, SafetyStock), StockMovement (ArticleI3D, Quantity, MovementType, Timestamp)

Auswirkungen: Überlagern wird minimiert. Lieferverzögerungen werden vermieden. Kapitalbindung wird optimiert. Automatische Nachbestellungen sparen Zeit.

10.3 Inventur (Inventory Count)

Module Path: `src/centron/Centron.WPF.UI/Modules/Warehousing/Inventory`
Controller: `InventoryAppModuleController`
ViewModel: `InventoryMainViewmodel`
Category: Logistik
Description: Durchführung und Verwaltung von physischen Bestandsaufnahmen mit Abweichungs-Analyse und automatischer Bestandskorrektur

Use Cases

10.3.1 Inventur planen und initialisieren

Zweck: Vorbereitung und Start einer physischen Bestandsaufnahme

Ablauf:

1. Benutzer navigiert zu "Neue Inventur"
2. Inventur-Typ wird gewählt: Vollständig (alle Artikel) oder Teilweise (selektive Artikel)
3. Lagerorte werden ausgewählt: Ein Lager oder mehrere Lager
4. Inventur-Datum wird gesetzt
5. Gruppen können definiert werden (z.B. "Lager A Bereich 1", "Lager A Bereich 2")
6. System erzeugt automatisch Inventur-Listen nach Gruppen
7. Listen werden als QR-Code oder Barcode-Labels gedruckt
8. Inventur wird mit Status "Offen" initialisiert
9. Benachrichtigungen werden an Lagerpersonal versendet

Betroffene Felder: Inventory (InventoryType, InventoryDate, StorageLocationI3D, Status, CreatedByI3D, CreatedDate)

Auswirkungen: Strukturierte Bestandsaufnahme wird möglich. Lagerbestände werden zu einheitlichem Zeitpunkt aufgenommen. Vergleichbarkeit wird sichergestellt.

10.3.2 Artikel erfassen während Bestandsaufnahme

Zweck: Durchführung der physischen Bestandsaufnahme mit Barcode oder manueller Erfassung

Ablauf:

1. Lagermitarbeiter öffnet Inventur-App (Mobile oder WPF)
2. Benutzer scannt/erfasst Artikel Barcode
3. System zeigt: Artikel-Name, Artikel-Nummer, Lagerlocation

4. Benutzer gibt gezählte Menge ein
5. System speichert Erfassung lokal (offline-Modus möglich)
6. Benutzer kann Bemerkungen hinzufügen (z.B. "Beschädigt", "Falscher Lagerort")
7. Nächster Artikel wird erfasst
8. Nach Abschluss einer Gruppe: Benutzer klickt "Gruppe abgeschlossen"
9. System synchronisiert Daten zum Server

Betroffene Felder: InventoryArticles (InventoryI3D, ArticleI3D, CountedQuantity, Remarks, CountedByI3D, CountedDate)

Auswirkungen: Exakte Bestandszahlen werden erfasst. Abweichungen werden identifiziert. Bestandsdaten sind aktuell nach Inventur.

10.3.3 Abweichungen analysieren und korrigieren

Zweck: Überprüfung und Behandlung von Bestandsabweichungen

Ablauf:

1. Nach Abschluss der Inventur wird Abweichungs-Report generiert
2. System zeigt: Sollbestand vs. Istbestand pro Artikel
3. Abweichungen werden klassifiziert:
   - Kleine Abweichungen (<1%): Akzeptiert oder untersucht
   - Große Abweichungen: Muss geklärt werden
4. Benutzer kann Ursachen erfassen:
   - Fehler in Lagerung
   - Fehler in Systemerfassung
   - Tatsächlicher Schwund
   - Falscher Lagerort
5. Abweichungs-Report wird dokumentiert
6. Nach Genehmigung werden Bestände auf Istbestände korrigiert
7. Bestandsabweichungen werden im Audit protokolliert

Betroffene Felder: InventoryArticles (DiscrepancyAmount, DiscrepancyReason, DiscrepancyAnalysis), InventoryLog (VarianceAmount, VariancePercentage)

Auswirkungen: Ursachen von Bestandsabweichungen werden identifiziert. Schwundraten werden minimiert. Lagerverwaltung wird optimiert.

10.3.4 Inventur abschließen und Bestandskorrektur

Zweck: Formaler Abschluss der Inventur und Aktualisierung der System-Bestände

Ablauf:

1. Benutzer öffnet abgeschlossene Inventur
2. Button "Inventur abschließen" wird geklickt
3. System validiert: Alle Gruppen erfasst? Alle Abweichungen geklärt?
4. Abweichungs-Checkliste wird angezeigt
5. Nach Bestätigung: Bestandskorrektur-Bewegungen werden generiert
6. Bestandsbuchungen aktualisieren Lagerverwaltung
7. Finanzbuchungen für Bestandsabweichungen werden erstellt (falls > Toleranz)
8. Inventur-Report wird generiert
9. System versendet Inventur-Ergebnis per Email

Betroffene Felder: Inventory (Status="Completed", ClosedDate, ClosedByI3D), StockMovement (MovementType="InventoryCorrection", InventoryI3D)

Auswirkungen: Lagerverwaltung wird aktualisiert. Bestandsdaten sind ab sofort aktuell. Finanzbuchhaltung erhält Korrektur-Informationen. Inventur ist abgeschlossen.

10.3.5 Inventur-Ergebnisse und Kennzahlen

Zweck: Analyse und Reporting von Inventur-Ergebnissen

Ablauf:

1. Nach Inventur-Abschluss können Reports angefordert werden
2. Verfügbare Reports:
   - Gesamt-Abweichungsquote (z.B. 2.3%)
   - Abweichungen nach Artikel-Gruppe
   - Abweichungen nach Lagerort
   - Schwund-Analyse
3. System zeigt Trends: Abweichungsquote verbessert sich oder verschlechtert sich?
4. Benutzer kann Reports exportieren (Excel, PDF)
5. Reports können geplant werden (z.B. monatlicher Inventur-Report)
6. Management erhält Kennzahlen zur Bestandsverwaltungs-Qualität

Betroffene Felder: InventoryStatistic (VariancePercentage, TotalCountedQuantity, TotalDiscrepancy), InventoryLog

Auswirkungen: Bestandsverwaltungs-Qualität wird transparent. Trends werden erkannt. Verbesserungspotenziale werden identifiziert.

10.3.6 Seriennummern und Chargenverfolgung in Inventur

Zweck: Erfassung von Seriennummern bei verderblichen oder verfolgungspflichtigen Waren

Ablauf:

1. Bei Erfassung von Artikel mit Seriennummern: System zeigt Seriennummern-Felder
2. Benutzer erfasst Seriennummern der gezählten Artikel
3. System validiert Seriennummern gegen Lagerverwaltung
4. Chargennummern können erfasst werden (für Verfallsdaten)
5. Verfallsdatum-Überprüfung: Abgelaufene Chargen werden markiert
6. Abgelaufene Waren können als "zu vernichten" markiert werden
7. Seriennummern-Historie wird dokumentiert

Betroffene Felder: InventoryArticles (SerialNumber, LotNumber, ExpirationDate), SerialNumberTracking (InventoryI3D, SerialNumber, InventoryStatus)

Auswirkungen: Seriennummern-Verfolgung wird genau. Verfallsdatum-Management wird automatisiert. Compliance-Anforderungen werden erfüllt.

10.4 Kommissionierung (Picking)

Module Path: `src/centron/Centron.WPF.UI/Modules/Warehousing/Commissioning`
Controller: `CommissioningAppModuleController`
ViewModel: `CommissioningViewModel`
Category: Logistik
Description: Verwaltung von Kommissionieraufträgen und Picks mit Unterstützung für Barcode-gesteuerte Lagerpicker und Versand-Vorbereitung

Use Cases

10.4.1 Bestellungen für Kommissionierung auswählen

Zweck: Auswahl und Priorisierung von Bestellungen zur Kommissionierung

Ablauf:

1. Benutzer öffnet Kommissionierungs-Modul
2. System zeigt Liste von Bestellungen: Auftragstyp, Kundenname, Bestelldatum, Status

3. Filter können angewendet werden:
    - Liefertyp: Direktlieferung, Teillieferung, Konsignation
    - Status: Offen, In Bearbeitung, Bereit zum Versand
    - Priorität: Normal, Eilig, VIP
4. Benutzer wählt eine oder mehrere Bestellungen aus
5. System lädt Bestellpositionen
6. Benutzer kann Reihenfolge der Abarbeitung setzen
7. Kommissionieraufträge werden generiert

Betroffene Felder: ArticelHeaderMapping (OrderI3D, CommissioningStatus, Priority), CommissioningPosition (OrderPositionI3D, PickingStatus)

Auswirkungen: Lagerpickprozesse werden strukturiert. Prioritäten werden beachtet. VIP-Aufträge werden bevorzugt. Versand wird beschleunigt.

10.4.2 Artikel kommissionieren und scannen

Zweck: Durchführung des physischen Picks mit Barcode-Unterstützung

Ablauf:

1. Lagermitarbeiter erhält Picking-Liste (gedruckt oder digital)
2. Benutzer navigiert zum ersten Lagerlocation
3. Barcode der Lagerlocation wird gescannt
4. System zeigt: zu pickende Artikel, Menge, Ziellocations
5. Benutzer scannt Artikel-Barcode oder Seriennummern
6. System validiert: Ist das der richtige Artikel? Richtige Menge?
7. Benutzer bestätigt Pick
8. Nächster Artikel wird angezeigt
9. Nach Abschluss aller Artikel: Benutzer bestätigt Kommissionierung abgeschlossen
10. System aktualisiert Bestandsbuchungen

Betroffene Felder: CommissioningPosition (ArticleI3D, PickedQuantity, PickedByI3D, PickedDate, PickingStatus="Completed")

Auswirkungen: Lagerprozesse werden beschleunigt. Fehlerquote sinkt (Barcode-Validierung). Bestandsgenauigkeit wird garantiert. Versand wird vorbereitet.

10.4.3 Seriennummern und Verfallsdatum tracken

Zweck: Erfassung von Seriennummern und Verfallsdatum bei der Kommissionierung

Ablauf:

1. Bei Pick eines Artikels mit Seriennummern wird Seriennummern-Feld angezeigt
2. Benutzer scannt oder erfasst Seriennummer
3. System validiert Seriennummer gegen Lagerverwaltung
4. Verfallsdatum wird angezeigt (falls verfügbar)
5. Warnung bei abgelaufenem oder nahenden Verfallsdatum
6. FIFO-Logik wird angewendet (älteste Chargen zuerst)
7. Seriennummern werden in Versandetiketten dokumentiert
8. Compliance-Anforderungen werden erfüllt

Betroffene Felder: CommissioningPosition (SerialNumber, LotNumber, ExpirationDate), ArticelPositionMapping (UsedSerialNumber, UsedLotNumber)

Auswirkungen: Verfallsdatum-Management wird automatisiert. FIFO-Prozesse werden erzwungen. Compliance wird dokumentiert. Kundenbeziehungen werden geschützt (keine abgelaufenen Waren).

10.4.4 Lagerbestände aktualisieren bei Kommissionierung

Zweck: Automatische Aktualisierung von Lagerbeständen während des Picks

Ablauf:

1. Nach erfolgreicher Kommissionierung wird Bestandsbuchung generiert
2. Lagerbestand wird reduziert um gepickte Menge
3. Bestandsbewegung wird dokumentiert: Von Location A nach Versand-Staging
4. System prüft: Wurden zu viele Artikel aus Location gepickt?
5. Falls zu viele: Warnung wird angezeigt
6. System verfolgt Bestandsbewegung pro Location
7. Echtzeit-Bestandssicht wird aktualisiert
8. Audit-Trail dokumentiert alle Bewegungen

Betroffene Felder: StockMovement (MovementType="Commissioning", ArticleI3D, Quantity, FromLocationI3D, ToLocationI3D, CommissioningPositionI3D)

Auswirkungen: Lagerbestände sind immer aktuell. Überbuchungen werden verhindert. Bestandsgenauigkeit verbessert sich. Finanzbuchhaltung hat genaue Daten.

10.4.5 Unfähige-Picks und Abweichungen behandeln

Zweck: Behandlung von Situationen, in denen ein Artikel nicht gepickt werden kann

Ablauf:

1. Lagermitarbeiter findet Artikel nicht am erwarteten Lagerort
2. Benutzer scannt "Nicht verfügbar" Button
3. System zeigt Alternative:
   - Artikel von anderem Lagerort suchen
   - Kunde informieren, dass Artikel nicht verfügbar
4. Alternative Lagerlocations werden vorgeschlagen
5. Benutzer kann Alternative akzeptieren oder Pickup skippen
6. Bei Skip: Bestellung wird als "Teilweise gepickt" markiert
7. Backorder wird automatisch erstellt
8. Kunde wird informiert über Verfügbarkeit

Betroffene Felder: CommissioningPosition (PickingStatus="PartiallyPicked"/"Unavailable"), BackorderI3D, NotificationSent

Auswirkungen: Unerwartete Lagerbestands-Abweichungen werden dokumentiert. Kundenerwartungen werden gek klärt. Backorder-Prozess wird automatisiert.

10.4.6 Versand vorbereiten und abschließen

Zweck: Vorbereitung von Paketen für Versand nach abgeschlossener Kommissionierung

Ablauf:

1. Nach Abschluss der Kommissionierung werden Artikel zur Versand-Staging gebracht
2. System zeigt: Bestellung, gepickte Artikel, Versand-Adresse
3. Verpackung wird durchgeführt (vom Lagerpersonal oder Versand-Team)
4. Benutzer scannt Paket-Barcode
5. Artikel werden in Paket verpackt und versiegelt
6. Benutzer bestätigt: Paket bereit zum Versand
7. System generiert Versandetikett (mit Tracking-Nummer)
8. Versand wird mit Versand-Dienstleister integriert (GLS, DHL, etc.)
9. Tracking wird mit Bestellung verlinkt
10. Kunde wird automatisch benachrichtigt

Betroffene Felder: CommissioningPosition (Status="ReadyForShipment"), ShipmentI3D, TrackingNumber, ShippingCarrierI3D

Auswirkungen: Versand wird beschleunigt. Fehler werden minimiert. Kundenkommunikation wird automatisiert. Tracking wird verfügbar. Lieferkette wird transparent.

11. MyCentron (MyCentron Portal)

11.1 Dashboard (Web Portal Dashboard)

Module Path: `src/CentronNexus/ServiceBoard/Dashboard`
Component: `Dashboard.razor`
Service: `DashboardService.cs`
Category: MyCentron Portal
Description: Blazor Server-basiertes Web Portal Dashboard mit Live-Daten zu Tickets, MyDay-Items, Zeiterfassung und KPIs

Use Cases

11.1.1 Dashboard-Komponenten konfigurieren (Web Portal)

Zweck: Personalisierung der Web Portal-Ansicht mit relevanten Zusammenfassungen und Echtzeit-Statistiken
Ablauf: 1. Web Portal öffnen 2. Dashboard öffnen 3. Verfügbare Widgets aktivieren: Meine Tickets, Heute's Tasks, Zeiterfassung 4. Größe und Position anpassen 5. Speichern
Betroffene Felder: DashboardConfiguration (UserI3D, WidgetType, IsVisible, Order)
Auswirkungen: Schneller Überblick über Tagesaufgaben. Mobile-freundliche Ansicht. Responsive Design auf allen Geräten.

11.1.2 Live-Ticket-Statistiken im Web Portal anzeigen

Zweck: Echtzeit-Anzeige der Ticket-Metriken und -Status im Web Portal
Ablauf: 1. Dashboard öffnen 2. Ticket-Statistik-Widget laden 3. Verdichtung anzeigen: Offen, In Bearbeitung, Gelöst, Warten 4. Klick auf Kategorie öffnet Ticket-Liste
Betroffene Felder: TicketStatistics (OpenCount, InProgressCount, ResolvedCount, WaitingCount)
Auswirkungen: Übersicht über Ticket-Bestand. Schnelle Navigation zu spezifischen Ticket-Status. Portal-basierte Ticket-Verwaltung.

11.1.3 Zeiterfassung im Web Portal überwachen

Zweck: Tracking von Arbeitsstunden und Pausen direkt im Web Portal
Ablauf: 1. Dashboard öffnen 2. Zeiterfassungs-Widget anzeigen 3. Aktuelle Arbeitszeit sehen 4. Tagesübersicht mit Start/Stop 5. Pausen verwalten
Betroffene Felder: TimeRecordI3D, StartTime, EndTime, BreakDuration, TodayHours
Auswirkungen: Mitarbeiter können von überall aus Zeiten erfassen. Keine Offline-Zeiten verloren. Tagesabschluss jederzeit möglich.

11.1.4 MyDay-Prioritäten im Web Portal synchronisieren

Zweck: Synchronisierung von priorisierten Aufgaben zwischen WPF Desktop und Web Portal
Ablauf: 1. MyDay-Widget laden 2. Priorisierte Aufgaben anzeigen (Top 5) 3. Status aktualisieren 4. Neue Items hinzufügen 5. Mit WPF Desktop synchronisieren
Betroffene Felder: MyDayItem (Priority, Status, DueDate, Assignee)
Auswirkungen: Konsistente Aufgabenverwaltung. Cross-Plattform-Zugriff. Bessere Zusammenarbeit zwischen Desktop und Mobile.

11.2 Mein Tag (My Day - Web Portal)

Module Path: `src/CentronNexus/ServiceBoard/MyDay`
Component: `MyDayPage.razor`
Service: `MyDayService.cs`

Category: MyCentron Portal
Description: Blazor Server-basierte Web Portal Tagesplanungs-Verwaltung mit Echtzeit-Synchronisierung und priorisierten Aufgaben

Use Cases

11.2.1 MyDay-Items im Web Portal verwalten

Zweck: Verwaltung von priorisierten Tagesaufgaben und Arbeitsitems im Web Portal
Ablauf: 1. MyDay öffnen 2. Heute's Items anzeigen 3. Neue Items hinzufügen (per Zettel oder Link) 4. Reihenfolge per Drag&Drop ändern 5. Speichern und mit Desktop synchronisieren
Betroffene Felder: MyDayItem (WorkItemI3D, Priority, Status, DueTime, EmployeeI3D)
Auswirkungen: Mobile-Zugriff auf Tagesplannung. Priorisierte Arbeitsreihenfolge klar. Cross-Device-Synchronisierung.

11.2.2 Arbeitszeiten im MyDay-Kontext tracken

Zweck: Zeiterfassung und Statusverfolgung direkt im MyDay-Item
Ablauf: 1. Item öffnen 2. Start/Stop-Knopf drücken 3. System startet Timer 4. Arbeitsschritte dokumentieren 5. Fertigstellen markieren 6. Speichern
Betroffene Felder: TimeRecord (MyDayItemI3D, StartTime, EndTime, Status, Notes)
Auswirkungen: Genaue Zeiterfassung pro Aufgabe. Automatische Dokumentation der Arbeitsschritte. Bessere Leistungskontrolle.

11.2.3 MyDay-Statistiken und Trendanalyse

Zweck: Anzeige von Leistungsmetriken und Trendanalyse über Web Portal
Ablauf: 1. MyDay-Statistik-Seite öffnen 2. Zeitraum wählen (heute, diese Woche, diesen Monat) 3. Metriken anzeigen: Durchschnittliche Items, Durchschnittliche Aufgabenzeit 4. Trend-Charts visualisieren
Betroffene Felder: MyDayStatistics (TotalItems, CompletedItems, AverageDuration, TrendLine)
Auswirkungen: Selbstreflexion und Leistungstracking. Bessere Zeitschätzung für zukünftige Aufgaben. Motivationsfeedback.

11.2.4 Team-MyDay-Übersicht im Web Portal

Zweck: Abteilungsleiter können MyDay-Übersicht von Team-Mitarbeitern im Portal sehen
Ablauf: 1. Abteilungsleiter öffnet Team-View 2. Team-Mitglieder auswählen 3. Deren heutige MyDay-Items sehen 4. Kapazitäten und Auslastung bewerten 5. Bei Bedarf Items umverteilen
Betroffene Felder: MyDayItem (EmployeeI3D, Priority, Status), Employee (Capacity, CurrentWorkload)
Auswirkungen: Bessere Ressourcenplanung. Kapazitätsausgleich zwischen Mitarbeitern. Bottlenecks früh erkennen.

11.3 Telefonate (Phone Calls - Nicht implementiert)

Module Path: `src/CentronNexus/ServiceBoard/PhoneCalls` (Stub)
Component: `CallsPage.razor` (Placeholder)
Category: MyCentron Portal
Status: ⚠️ **GEPLANT** (Not Yet Implemented)
Description: Telefon- und Kommunikations-Management im Web Portal - zukünftige Funktion

Hinweis

Dieses Modul ist derzeit **nicht implementiert** und existiert nur als Platzhalter. Die vollständige Telefonie-Integration ist für eine zukünftige Version geplant.

Erwartete Funktionen (Roadmap):

- Anruflisten und Anrufhistorie im Portal anzeigen
- Automatische Anrufprotokollierung
- Integration mit CRM-Kundendaten
- Nachrichtentemplate für häufige Gesprächs-Ergebnisse
- Statistiken zu Anrufzeiten und Gesprächsdauern

11.4 Todo-Liste (Todo List - Desktop Only)

Module Path: `src/centron/Centron.WPF.UI/Modules/MyCentron/TodoList` (WPF Desktop nur)
Component: `TodoListView.xaml`
Category: MyCentron Desktop
Status: ⚠️ **NUR IM DESKTOP VERFÜGBAR** (Not in Web Portal)
Description: Verwaltung von persönlichen Aufgaben und Checklisten - nur in WPF Desktop verfügbar

Hinweis

Das Todo-Listen-Modul ist derzeit **nur im WPF Desktop-Client** (Kapitel 9.8) verfügbar. Die Web-Portal-Version wird durch das **MyDay-Modul** (11.2) ersetzt, das eine integrierte Tagesplanungs- und Aufgabenverwaltung bietet.

Für Web Portal-Benutzer:

- Verwenden Sie **Kapitel 11.2 (Mein Tag / MyDay)** für Tagesaufgaben-Management im Web Portal
- MyDay bietet bessere Integration mit Zeiterfassung und Team-Kapazitätsplanung
- Volle Synchronisierung zwischen Desktop Todo-Liste und Portal MyDay geplant für zukünftige Versionen

Desktop-Benutzer:

- Siehe Kapitel 9.8 (TodoList - Desktop-Version) für vollständige Dokumentation und Use Cases

12. Passwort Manager (Password Manager - Deprecated)

12.1 Access Area Management (Zugangsbereicheverwaltung)

Module Path: `src/centron/Centron.WPF.UI/Modules/PasswordManager`
Controller: `AccessAreaManagementAppModuleController`
ViewModel: `AccessAreaManagementViewModel`
Category: Passwort Manager
Description: Verwaltung von logischen Zugriffsbereichen zur Gruppierung und Strukturierung von Zugängen und Passwörtern

Use Cases

12.1.1 Zugangsbereich anlegen

Zweck: Erstellung logischer Bereiche zur Organisation von Zugangsdaten

Ablauf:

1. Administrator öffnet Access Area Management
2. Neuer Bereich wird erstellt: Name, Beschreibung, übergeordneter Bereich
3. Hierarchie wird definiert: "IT-Infrastruktur" → "Server" → "Produktiv-Server"
4. Verantwortlicher wird zugeordnet (Team oder Person)
5. Standard-Richtlinie wird ausgewählt (Passwort-Komplexität)
6. Bereich wird gespeichert und ist sofort für Zuordnung verfügbar

Betroffene Felder: AccessArea (Name, Description, ParentAreaI3D, ResponsibleEmployeeI3D, PolicyI3D)

Auswirkungen: Strukturierte Ablage von Zugängen. Verantwortlichkeiten sind klar. Schnelles Auffinden durch Hierarchie.

12.1.2 Berechtigungen pro Bereich vergeben

Zweck: Steuerung des Zugriffs auf Zugangsdaten nach Bereich

Ablauf:

1. Admin wählt Zugangsbereich und öffnet "Berechtigungen"
2. Benutzer oder Gruppen werden hinzugefügt
3. Berechtigungsstufe wird gewählt: Lesen, Lesen+Passwort anzeigen, Bearbeiten
4. Vererbung von übergeordnetem Bereich kann aktiviert werden
5. Temporäre Berechtigungen möglich: Gültig vom 01.12 bis 31.12
6. Änderungen werden gespeichert und sofort wirksam

Betroffene Felder: AccessAreaPermission (AreaI3D, UserI3D, PermissionLevel, ValidFrom, ValidTo, IsInherited)

Auswirkungen: Zugriffsschutz nach Need-to-Know. Zeitlich begrenzte Zugriffe möglich. Audit-Trail für Compliance.

12.1.3 Bereichsstruktur reorganisieren

Zweck: Anpassung der Hierarchie bei organisatorischen Änderungen

Ablauf:

1. Admin öffnet Baumansicht der Zugriffsbereiche
2. Bereiche können per Drag & Drop verschoben werden
3. System warnt bei Auswirkungen auf Berechtigungen
4. Vererbte Berechtigungen werden automatisch aktualisiert
5. Zugänge im verschobenen Bereich bleiben zugeordnet
6. Änderungsprotokoll wird erstellt

Betroffene Felder: AccessArea (ParentAreaI3D, LastModified, ModifiedByI3D)

Auswirkungen: Flexible Anpassung an Organisationsstruktur. Berechtigungen bleiben konsistent. Historie ist nachvollziehbar.

12.2 Access Management (Zugänge/Passwort-Manager)

Module Path: `src/centron/Centron.WPF.UI/Modules/PasswordManager`
Controller: `AccessManagementAppModuleController`
ViewModel: `AccessManagementViewModel`
Category: Passwort Manager
Description: Zentrale Verwaltung von Zugangsdaten, Passwörtern und Credentials für IT-Systeme und Applikationen

Use Cases

12.2.1 Zugang erfassen

Zweck: Speicherung von Login-Credentials mit verschlüsselter Ablage

Ablauf:

1. Benutzer öffnet Passwort-Manager und klickt "Neuer Zugang"
2. Zugangsdaten werden erfasst: System/URL, Benutzername, Passwort
3. Zugangsbereich wird gewählt (z.B. "Kunden-Server")
4. Zusatzinformationen können hinterlegt werden: IP, Port, Notizen
5. Kunde kann verknüpft werden (für Kunden-spezifische Zugänge)
6. Passwort wird verschlüsselt gespeichert (AES-256)

Betroffene Felder: AccessCredential (SystemName, Username, PasswordEncrypted, AccessAreaI3D, AccountI3D, URL, Notes)

Auswirkungen: Zentrale Passwort-Verwaltung. Verschlüsselter Schutz sensibler Daten. Keine Passwörter in Klartext.

12.2.2 Passwort anzeigen und kopieren

Zweck: Sichere Anzeige von Passwörtern mit Audit-Logging

Ablauf:

1. Benutzer sucht Zugang in Access Management
2. Passwort wird standardmäßig maskiert angezeigt: "**********"
3. "Passwort anzeigen" wird geklickt (Berechtigung erforderlich)
4. System fordert Bestätigung: "Zugriff auf Passwort wird protokolliert"
5. Passwort wird temporär angezeigt (10 Sekunden)
6. Zugriff wird geloggt: Wer, Wann, Welcher Zugang
7. "In Zwischenablage kopieren" verfügbar (Auto-Clear nach 30 Sek)

Betroffene Felder: AccessLog (AccessCredentialId3D, AccessedById3D, AccessDate, ActionType)

Auswirkungen: Nachvollziehbarkeit von Passwort-Zugriffen. Compliance-konform. Missbrauch wird erkennbar.

12.2.3 Passwort generieren und rotieren

Zweck: Automatische Erstellung sicherer Passwörter und regelmäßige Rotation

Ablauf:

1. Benutzer bearbeitet Zugang und klickt "Neues Passwort generieren"
2. Passwort-Generator wird geöffnet mit Optionen: Länge (12-32 Zeichen), Groß-/Kleinbuchstaben, Ziffern, Sonderzeichen
3. Richtlinienvorgaben werden automatisch angewendet
4. Generiertes Passwort wird angezeigt und kann übernommen werden
5. System kennzeichnet Passwort mit Erstellungsdatum
6. Rotation-Reminder wird gesetzt (z.B. alle 90 Tage)

Betroffene Felder: AccessCredential (PasswordEncrypted, PasswordCreatedDate, NextRotationDate, GenerationPolicy)

Auswirkungen: Starke Passwörter werden erzwungen. Regelmäßige Rotation reduziert Risiko. Compliance-Anforderungen erfüllt.

12.2.4 Zugang mit Remote-Tools verknüpfen

Zweck: Direkter Verbindungsaufbau aus Access Management heraus

Ablauf:

1. Benutzer wählt RDP- oder SSH-Zugang aus Liste
2. "Verbinden" wird geklickt
3. System startet entsprechendes Tool: RDP-Client, SSH-Client
4. Credentials werden automatisch übergeben (kein manuelles Eintippen)
5. Verbindung wird hergestellt
6. Bei Trennung: Verbindungsdauer wird protokolliert

Betroffene Felder: AccessCredential (ConnectionType, Port, LastConnectionDate, TotalConnections)

Auswirkungen: Schnellerer Verbindungsaufbau. Tippfehler werden vermieden. Nutzungsstatistiken verfügbar.

12.3 Guideline Management (Richtlinienverwaltung)

Module Path: `src/centron/Centron.WPF.UI/Modules/PasswordManager`
Controller: `GuidelineManagementAppModuleController`
ViewModel: `GuidelineManagementViewModel`

Category: Passwort Manager
Description: Definition und Durchsetzung von Passwort-Richtlinien und Sicherheitsstandards für Zugangsverwaltung

Use Cases

12.3.1 Passwort-Richtlinie definieren

Zweck: Festlegung von Komplexitätsanforderungen für Passwörter

Ablauf:

1. Admin öffnet Guideline Management und klickt "Neue Richtlinie"
2. Richtlinienname wird vergeben: "Server-Passwörter Standard"
3. Anforderungen werden definiert:
   - Mindestlänge: 16 Zeichen
   - Groß-/Kleinbuchstaben, Ziffern, Sonderzeichen erforderlich
   - Keine Wiederholung der letzten 5 Passwörter
   - Gültigkeitsdauer: 90 Tage
4. Richtlinie wird gespeichert und Bereichen zugeordnet

Betroffene Felder: PasswordPolicy (Name, MinLength, RequireUppercase, RequireLowercase, RequireDigits, RequireSpecialChars, ValidityDays, HistoryCount)

Auswirkungen: Einheitliche Sicherheitsstandards. Schwache Passwörter werden verhindert. Compliance-Vorgaben erfüllt.

12.3.2 Richtlinien-Compliance prüfen

Zweck: Identifikation von Zugängen, die Richtlinien nicht erfüllen

Ablauf:

1. Admin öffnet Compliance-Report
2. System prüft alle Zugänge gegen zugeordnete Richtlinien
3. Verstöße werden angezeigt:
   - Passwort zu kurz (12 statt 16 Zeichen)
   - Passwort abgelaufen (seit 120 Tagen)
   - Keine Sonderzeichen enthalten
4. Schweregrad wird farblich markiert: Gelb (Warnung), Rot (Kritisch)
5. Bulk-Aktion möglich: "Alle betroffenen Passwörter neu generieren"

Betroffene Felder: ComplianceCheck (AccessCredentialI3D, PolicyI3D, ViolationType, Severity, CheckDate)

Auswirkungen: Sicherheitslücken werden erkannt. Proaktive Risikovermeidung. Audit-Nachweis für ISO 27001.

12.3.3 Rotation-Erinnerungen konfigurieren

Zweck: Automatische Benachrichtigungen für ablaufende Passwörter

Ablauf:

1. Admin öffnet Richtlinie und aktiviert "Rotation-Erinnerungen"
2. Vorlaufzeit wird definiert: Benachrichtigung 14 Tage vor Ablauf
3. Eskalation wird konfiguriert: Nach Ablauf alle 3 Tage erinnern
4. Empfänger werden festgelegt: Zugangsverantwortlicher + IT-Admin
5. E-Mail-Template wird angepasst
6. System versendet automatisch Erinnerungen

Betroffene Felder: PasswordPolicy (RotationReminderDays, EscalationInterval, NotificationRecipients)

Auswirkungen: Passwörter werden rechtzeitig erneuert. Abgelaufene Zugänge werden minimiert. Compliance-Risiko sinkt.

12.4 RDP Embedded (Remote Desktop eingebettet)

Module Path: `src/centron/Centron.WPF.UI/Modules/PasswordManager`
Controller: `RDPEmbeddedAppModuleController`
ViewModel: `RDPEmbeddedViewModel`
Category: Passwort Manager
Description: Eingebetteter RDP-Client für Remote-Desktop-Verbindungen direkt aus c-entron heraus

Use Cases

12.4.1 RDP-Verbindung aus Access Management starten

Zweck: Direkter Remote-Desktop-Zugriff ohne externe Tools

Ablauf:

1. Benutzer wählt Windows-Server-Zugang aus Access Management
2. "RDP verbinden" wird geklickt
3. System öffnet eingebetteten RDP-Client im c-entron-Fenster
4. Credentials werden automatisch aus Passwort-Manager übergeben
5. Verbindung wird hergestellt ohne manuelle Eingabe
6. Remote-Desktop wird in Tab/Fenster angezeigt

Betroffene Felder: RDPSession (AccessCredentialI3D, StartTime, EndTime, UserI3D, Resolution)

Auswirkungen: Schnellerer Verbindungsaufbau. Keine Passwort-Eingabe erforderlich. Alles in einer Oberfläche.

12.4.2 Multi-Monitor-Unterstützung nutzen

Zweck: Remote-Desktop über mehrere Monitore verteilen

Ablauf:

1. Benutzer startet RDP-Verbindung
2. System erkennt verfügbare Monitore automatisch
3. Option "Vollbild auf allen Monitoren" wird angezeigt
4. Bei Aktivierung: Remote-Desktop nutzt alle verfügbaren Bildschirme
5. Taskleiste und Windows-Funktionen bleiben lokal bedienbar
6. Trennung per Hotkey (Strg+Alt+Pause) möglich

Betroffene Felder: RDPSession (MultiMonitorMode, MonitorCount, Resolution)

Auswirkungen: Effizientes Arbeiten auf Remote-Systemen. Maximale Bildschirmfläche. Produktivitätssteigerung.

12.4.3 Verbindungshistorie einsehen

Zweck: Nachvollziehbarkeit von Remote-Zugriffen für Audit

Ablauf:

1. Admin öffnet RDP-Verbindungshistorie
2. Alle RDP-Sessions werden angezeigt: Benutzer, Server, Start/Ende, Dauer
3. Filter nach Zeitraum, Benutzer oder Server möglich
4. Details zeigen verwendete Credentials und IP-Adresse
5. Export als CSV für externe Auditoren möglich
6. Bei Auffälligkeiten: Screenshot-Protokoll verfügbar (falls aktiviert)

Betroffene Felder: RDPSession (StartTime, EndTime, Duration, AccessCredentialI3D, UserI3D, IPAddress)

Auswirkungen: Vollständige Nachvollziehbarkeit. Compliance-Nachweis. Missbrauch wird erkennbar.

12.5 SSH Embedded (SSH-Client eingebettet)

Module Path: `src/centron/Centron.WPF.UI/Modules/PasswordManager`
Controller: `SSHEmbeddedAppModuleController`
ViewModel: `SSHEmbeddedViewModel`
Category: Passwort Manager
Description: Eingebetteter SSH-Client für Terminal-Verbindungen zu Linux-Servern und Netzwerkgeräten

Use Cases

12.5.1 SSH-Verbindung aufbauen

Zweck: Terminal-Zugriff auf Linux-Server direkt aus c-entron

Ablauf:

1. Benutzer wählt SSH-Zugang aus Access Management
2. "SSH verbinden" wird geklickt
3. System öffnet eingebettetes Terminal-Fenster
4. SSH-Verbindung wird mit hinterlegten Credentials aufgebaut
5. Terminal ist sofort nutzbar (bash, zsh, etc.)
6. Mehrere Sessions können parallel in Tabs geöffnet werden

Betroffene Felder: SSHSession (AccessCredentialI3D, StartTime, EndTime, UserI3D, TerminalType)

Auswirkungen: Kein separater SSH-Client nötig. Passwörter automatisch verfügbar. Zentrale Arbeitsoberfläche.

12.5.2 Befehle und Output protokollieren

Zweck: Audit-Trail für ausgeführte Befehle auf produktiven Systemen

Ablauf:

1. SSH-Session wird mit aktiviertem Logging gestartet
2. Alle eingegebenen Befehle werden mitprotokolliert
3. System-Output wird ebenfalls gespeichert
4. Session-Log ist nach Verbindungstrennung verfügbar
5. Admin kann Logs durchsuchen: "Wer hat 'rm -rf' ausgeführt?"
6. Logs sind unveränderlich und revisionssicher

Betroffene Felder: SSHSessionLog (SessionI3D, Timestamp, Command, Output, IsSuccess)

Auswirkungen: Nachvollziehbarkeit kritischer Aktionen. Troubleshooting wird vereinfacht. Compliance für kritische Systeme.

12.5.3 Gespeicherte Befehlsvorlagen nutzen

Zweck: Schnellzugriff auf häufig verwendete Befehle

Ablauf:

1. Benutzer öffnet SSH-Terminal
2. Rechtsklick zeigt "Befehlsvorlagen"
3. Vordefinierte Befehle werden angezeigt: "Systemstatus prüfen", "Logs anzeigen", "Dienst neu starten"
4. Vorlage wird gewählt und automatisch ins Terminal eingefügt
5. Platzhalter werden abgefragt: "Welcher Dienst? [nginx/apache/mysql]"

6. Befehl wird ausgeführt

Betroffene Felder: CommandTemplate (Name, Command, Parameters, Category, UsageCount)

Auswirkungen: Tippfehler werden vermieden. Standardisierte Abläufe. Effizienzsteigerung für Admins.

13. Produktion (Production)

13.1 Maschinenverwaltung (Machine Management)

Module Path: `src/centron/Centron.WPF.UI/Modules/Production/MachineManagement`
Controller: `MaschineManagementAppModuleController`
ViewModel: `MaschineManagementViewModel`
Category: Produktion
Description: Verwaltung von Produktionsmaschinen, deren Konfiguration, Wartungsintervallen und Auslastung für effiziente Fertigungssteuerung

Use Cases

13.1.1 Maschine anlegen und konfigurieren

Zweck: Erfassung neuer Produktionsmaschinen mit technischen Spezifikationen

Ablauf:

1. Benutzer öffnet Maschinenverwaltung und klickt auf "Neue Maschine"
2. Stammdaten werden erfasst: Name, Hersteller, Modell, Seriennummer
3. Technische Parameter werden konfiguriert: Max. Kapazität, Geschwindigkeit, Energieverbrauch
4. Kostenstellenzuordnung und Standort werden definiert
5. Wartungsintervalle werden festgelegt (Tage/Betriebsstunden)
6. Maschine wird gespeichert und ist sofort für Produktionsplanung verfügbar

Betroffene Felder: Machine (Name, Manufacturer, Model, SerialNumber, Capacity, Location, MaintenanceInterval)

Auswirkungen: Maschinenkapazitäten sind transparent. Produktionsplanung kann Verfügbarkeit berücksichtigen. Wartungstermine werden automatisch vorgeschlagen.

13.1.2 Wartungsprotokoll führen

Zweck: Dokumentation von Wartungsarbeiten und Maschinenzustand

Ablauf:

1. Techniker öffnet Maschinendetails und wählt "Wartung durchführen"
2. System zeigt anstehende Wartungsintervalle an
3. Wartungsart wird gewählt: Inspektion, Routine-Wartung, Reparatur
4. Durchgeführte Arbeiten werden dokumentiert: Austausch von Teilen, Ölwechsel, Kalibrierung
5. Techniker erfasst Ersatzteile aus Lager (automatische Bestandsbuchung)
6. Betriebsstundenzähler wird aktualisiert, nächster Wartungstermin wird berechnet

Betroffene Felder: MachineMaintenanceLog (Date, Type, WorkPerformed, ReplacedParts, NextMaintenanceDate, OperatingHours)

Auswirkungen: Wartungshistorie ist lückenlos dokumentiert. Ausfallzeiten werden minimiert. Verschleißteile-Bedarf wird prognostiziert.

13.1.3 Produktionskapazität planen

Zweck: Überwachung der Maschinenauslastung und Kapazitätsplanung

Ablauf:

1. Produktionsleiter öffnet Kapazitätsübersicht
2. System zeigt Auslastung pro Maschine: Laufzeit, Stillstand, Wartung
3. Produktionsaufträge werden auf freie Kapazitäten verteilt
4. Engpässe werden visualisiert (Maschinen mit >90% Auslastung)
5. Alternative Fertigungswege werden vorgeschlagen
6. Kapazitätsreport zeigt OEE (Overall Equipment Effectiveness)

Betroffene Felder: MachineCapacity (AvailableHours, UtilizedHours, DowntimeHours, OEE)

Auswirkungen: Produktionsplanung ist realistisch. Maschinenauslastung wird optimiert. Liefertermine sind verlässlich.

13.1.4 Maschinenhistorie auswerten

Zweck: Analyse von Leistungsdaten und Verschleißmustern für vorausschauende Wartung

Ablauf:

1. Benutzer wählt Maschine und öffnet Historie-Ansicht
2. System zeigt Timeline: Produktionsaufträge, Wartungen, Störungen
3. Produktivitätskennzahlen werden analysiert: Output pro Stunde, Ausschussrate, Stillstandzeiten
4. Verschleißmuster werden erkannt (z.B. Leistungsabfall vor Wartung)
5. Predictive Maintenance-Empfehlungen werden generiert
6. Export der Daten für externe Analysen möglich

Betroffene Felder: MachineHistory (Date, ProductionOrders, Downtime, OutputQuality, MaintenanceEvents)

Auswirkungen: Ungeplante Ausfälle werden reduziert. Wartungskosten sinken. Produktionseffizienz steigt kontinuierlich.

13.2 Produktionsaufträge (Production Orders)

Module Path: `src/centron/Centron.WPF.UI/Modules/Production/ProductionOrder`
Controller: `ProductionOrderManagementAppModuleController`
ViewModel: `ProductionOrderManagementViewModel`
Category: Produktion
Description: Erstellung und Verwaltung von Produktionsaufträgen mit Materialplanung, Fertigungssteuerung und Status-Tracking

Use Cases

13.2.1 Produktionsauftrag erstellen

Zweck: Anlegen neuer Fertigungsaufträge aus Kundenbestellungen oder Lagerbedarfen

Ablauf:

1. Benutzer öffnet Produktionsaufträge und klickt "Neuer Auftrag"
2. Quelle wird gewählt: Kundenauftrag, Lagerbestand-Auffüllung, oder manuelle Erstellung
3. Artikel, Menge und gewünschter Fertigstellungstermin werden erfasst
4. System zeigt Stückliste (BOM) und prüft Materialverfügbarkeit
5. Fertigungsschritte werden automatisch generiert (Routing)
6. Maschinen werden vorgeschlagen basierend auf Kapazität
7. Auftrag wird freigegeben und an Produktion weitergeleitet

Betroffene Felder: ProductionOrder (OrderNumber, ArticleI3D, Quantity, DueDate, Status, SourceType)

Auswirkungen: Produktionsplanung ist transparent. Materialengpässe werden frühzeitig erkannt. Durchlaufzeiten sind kalkulierbar.

13.2.2 Material reservieren und bereitstellen

Zweck: Materialallokation und Lagerbuchung für Produktionsaufträge

Ablauf:

1. Produktionsplaner öffnet genehmigten Auftrag
2. System zeigt benötigte Materialien aus Stückliste (BOM)
3. Verfügbarkeit wird geprüft: Auf Lager / Bestellt / Fehlmenge
4. Lagerist reserviert Material für diesen Auftrag (exklusive Reservierung)
5. Kommissionierung wird ausgelöst: Artikel werden zusammengestellt
6. Bereitstellungsplatz wird zugewiesen (z.B. Produktionsinsel 3)
7. Status wechselt auf "Material bereitgestellt"

Betroffene Felder: ProductionOrderMaterial (ArticleI3D, RequiredQuantity, ReservedQuantity, PickingLocation, Status)

Auswirkungen: Materialengpässe werden vermieden. Produktion kann ohne Unterbrechung starten. Lagerbestand ist präzise.

13.2.3 Produktionsfortschritt erfassen

Zweck: Tracking des Fertigungsstatus und Rückmeldung von abgeschlossenen Arbeitsschritten

Ablauf:

1. Maschinenbediener scannt Produktionsauftrag-Barcode oder wählt Auftrag aus Liste
2. System zeigt offene Fertigungsschritte: Zuschnitt, Montage, Qualitätsprüfung
3. Bediener startet Arbeitsschritt (Zeiterfassung beginnt)
4. Bei Abschluss wird Menge rückgemeldet: Gutmenge / Ausschuss
5. System bucht Materialverbrauch und Maschinenzeit
6. Nächster Fertigungsschritt wird freigegeben
7. Bei letztem Schritt: Fertigprodukt wird ins Lager gebucht

Betroffene Felder: ProductionOrderStep (StepNumber, Status, StartTime, EndTime, GoodQuantity, ScrapQuantity, MachineI3D)

Auswirkungen: Produktionsstatus ist jederzeit transparent. Durchlaufzeiten sind messbar. Ausschussquoten werden dokumentiert.

13.2.4 Produktionsauftrag abschließen

Zweck: Finalisierung des Auftrags mit Qualitätsprüfung und Lagerbuchung

Ablauf:

1. Qualitätsprüfer öffnet fertiggestellten Auftrag
2. Endprüfung wird durchgeführt: Maße, Funktion, Optik
3. Gutmenge und Ausschuss werden final bestätigt
4. Fertigprodukte werden ins Fertigwarenlager gebucht
5. Kosten werden kalkuliert: Material, Maschinenzeit, Personal
6. Abweichungen zur Planung werden dokumentiert (Zeit/Kosten)
7. Auftrag wird geschlossen, Lieferung an Kunde wird ausgelöst

Betroffene Felder: ProductionOrder (Status=Completed, ActualCost, ActualDuration, QualityCheckPassed, CompletedDate)

Auswirkungen: Nachkalkulation ist möglich. Produktionseffizienz ist messbar. Artikel sind verfügbar für Auslieferung.

14. Stammdaten (Master Data)

14.1 Belegkonditionen (Document Terms)

Module Path: `src/centron/Centron.WPF.UI/Modules/Administration/ReceiptConditions`
Controller: `ReceiptConditionManagementAppModuleController`
ViewModel: `ReceiptConditionManagementViewModel`
Category: Stammdaten
Description: Verwaltung von Zahlungs-, Liefer- und Dokumentbedingungen für Bestellungen und Rechnungen

Use Cases

14.1.1 Zahlungsbedingungen definieren

Zweck: Erstellung von Zahlungsbedingungen für Geschäftspartner

Ablauf:

1. Admin öffnet Zahlungsbedingungen-Management
2. Neue Zahlungsbedingung wird erstellt: Name, Skonto-Prozentsatz, Skonto-Tage, Zahlungsfrist
3. Beispiele: "Netto 30 Tage", "2/10 Netto 30", "Prepayment"
4. Bedingungen werden gespeichert und sind sofort verfügbar
5. Automatische Mahnläufe können konfiguriert werden

Betroffene Felder: PaymentCondition (Name, DiscountPercentage, DiscountDays, PaymentTermDays)

Auswirkungen: Konsistente Zahlungsbedingungen. Skonto-Berechnungen automatisiert. Mahnprozesse werden gesteuert.

14.1.2 Lieferbedingungen konfigurieren

Zweck: Verwaltung von Lieferkonditionen (Incoterms, Versandart)

Ablauf:

1. Admin erstellt Lieferbedingung: Name, Incoterm (EXW, FOB, CIF, DDP), Kosten-Verantwortung
2. Versandart wird zugeordnet: Paket, Palette, Container, etc.
3. Lieferzeiten werden definiert: Express, Standard, Slow
4. Versicherungsoptionen können aktiviert werden
5. Bedingungen sind in Bestellungen auswählbar

Betroffene Felder: DeliveryCondition (Name, Incoterm, ShippingType, DeliveryTime, InsuranceIncluded)

Auswirkungen: Versandinformationen sind standardisiert. Kundenerwartungen sind klar. TCO wird berechenbar.

14.1.3 Rabatte und Skonti verwalten

Zweck: Definition von Rabatt-Strukturen für Mengen oder zeitlich begrenzte Promotionen

Ablauf:

1. Admin erstellt Rabattstaffel: Ab 100 Stück -5%, ab 500 -10%, ab 1000 -15%
2. Zeitliche Rabatte können konfiguriert werden: Frühjahrs-Rabatt: -8% vom 01.03 bis 30.04
3. Kunde-spezifische Rabatte können definiert werden
4. System wendet Rabatte automatisch in Bestellungen an
5. Rabatt-Report zeigt gewährte Rabatte

Betroffene Felder: DiscountMatrix (Quantity, DiscountPercentage, ValidFrom, ValidTo), CustomerDiscount

Auswirkungen: Preisgestaltung wird flexibel. Volumen-Anreize werden gesetzt. Kundenakquisition wird unterstützt.

14.1.4 Standardbedingungen zuweisen

Zweck: Zuweisung von Standard-Zahlungs- und Lieferbedingungen an Kunden und Lieferanten

Ablauf:

1. Admin öffnet Kunde/Lieferant
2. Standard-Zahlungsbedingung wird zugewiesen (z.B. "Netto 30 Tage")
3. Standard-Lieferbedingung wird zugewiesen (z.B. "FOB, Paket")
4. Standard-Rabatt wird zugewiesen (falls vorhanden)
5. Diese Bedingungen werden automatisch in neuen Bestellungen vorgefüllt
6. Benutzer kann Bedingungen pro Bestellung anpassen

Betroffene Felder: Account (DefaultPaymentConditionI3D, DefaultDeliveryConditionI3D, DefaultDiscountI3D)

Auswirkungen: Bestellungen werden schneller erstellt. Konsistente Partner-Behandlung. Fehler werden minimiert.

14.1.5 Bedingungen-Reports und Audit

Zweck: Analyse und Tracking von Bedingungen-Verwendung

Ablauf:

1. Admin kann Reports anfordern: "Welche Zahlungsbedingungen werden am häufigsten verwendet?"
2. Bericht zeigt: Verwendete Bedingungen, Häufigkeit, durchschnittliche Zahlungsdauer
3. Abweichungen werden identifiziert (z.B. "Kundennummer XYZ erhält 50% Rabatt vs. 5% Standard")
4. Audit-Trail zeigt Änderungen an Bedingungen
5. Reports können exportiert werden

Betroffene Felder: ConditionAuditLog (ConditionI3D, OldValue, NewValue, ChangedByI3D, ChangedDate)

Auswirkungen: Bedingungen werden transparent. Compliance wird dokumentiert. Verhandlungen werden datengetrieben.

14.2 Data Updater

Module Path: `src/centron/Centron.WPF.UI/Modules/Massenupdates`
Controller: `MassUpdatesAppModuleController`
ViewModel: `MassUpdatesViewModel`
Category: Stammdaten
Description: Massen-Datenaktualisierung für Artikel, Bestellungen und Rechnungen mit Batch-Verarbeitung

Use Cases

14.2.1 Artikel-Preise stapelweise aktualisieren

Zweck: Massen-Preisanpassung für multiple Artikel gleichzeitig

Ablauf:

1. Admin öffnet Data Updater Modul
2. Template wird gewählt: "Artikel Preise aktualisieren"
3. Artikel-Filter wird gesetzt: Material-Gruppe, Kategorie, Preis-Range
4. Änderungsregel wird definiert: "+10% auf alle Preise" oder "auf Minimum 49,99€"
5. Vorschau zeigt betroffene Artikel und neue Preise

6. Nach Bestätigung werden Preise aktualisiert
7. Update-Report wird dokumentiert

Betroffene Felder: Article (PurchasePrice, ListPrice), UpdateLog (BatchID, RecordsUpdated, UpdateDate)

Auswirkungen: Preismanagement wird beschleunigt. Fehler durch Einzelerfassung entfallen. Konsistenz wird gewährleistet.

14.2.2 Artikel-Attribute in Batch ändern

Zweck: Massen-Änderung von Artikel-Eigenschaften (Kategorie, Materialgruppe, Lieferant)

Ablauf:

1. Admin erstellt Filter: "Alle Artikel der Kategorie 'Veraltet'"
2. Ziel-Kategorie wird gewählt: "Sonderangebote"
3. Template wird angewendet
4. Alle 250 betroffenen Artikel werden verschoben
5. Audit-Log dokumentiert Massenperation

Betroffene Felder: Article (MaterialGroupI3D, CategoryI3D, DefaultSupplierI3D)

Auswirkungen: Katalog-Neuorganisation wird schnell durchgeführt. Lagerverwaltung wird aktualisiert. Verkaufsansichten werden neu sortiert.

14.2.3 Bestellungen-Massendaten-Korrektur

Zweck: Batch-Korrektur von fehlerhaften Bestellungsdaten

Ablauf:

1. Admin identifiziert Problem: "Alle Bestellungen von 2025-01-15 haben falsches Lagerort"
2. Filter wird gesetzt: "Bestellungen vom 2025-01-15"
3. Korrektur wird definiert: "Lagerort ändern von 'Lagel A' zu 'Lager A'"
4. 1.250 Bestellungen werden korrigiert
5. Korrektur-Report wird versendet

Betroffene Felder: ReceiptTable (StorageLocationI3D), UpdateLog

Auswirkungen: Fehler werden flächendeckend korrigiert. Reporting wird genauer. Finanzbuchhaltung ist konsistent.

14.2.4 Validierung und Plausibilitätsprüfung

Zweck: Automatische Validierung von Updates vor Durchführung

Ablauf:

1. Admin plant Update: "Alle Lieferantenkürzel aktualisieren"
2. System führt automatisch Validierungen durch:
   - Sind alle Zieldaten vorhanden?
   - Sind Datentypen konsistent?
   - Gibt es Konflikte mit Constraints?
3. Validierungs-Report wird angezeigt
4. Wenn 99% valide: Update wird durchgeführt
5. Fehlerhafte Datensätze werden protokolliert

Betroffene Felder: ValidationLog (ErrorType, RecordID, ErrorMessage)

Auswirkungen: Datenqualität wird garantiert. Datenbankintegrität wird gewährleistet. Rollback wird nicht nötig.

14.2.5 Update-Historie und Rollback

Zweck: Verfolgung und ggf. Rückgängigmachung von Massen-Updates

Ablauf:

1. Admin kann Update-Historie anschauen: "Wer hat wann was geändert?"
2. Bei Bedarf: "Rollback durchführen" wird geklickt
3. System stellt vorherige Datenstände wieder her
4. Rollback wird dokumentiert
5. Benachrichtigungen werden versendet

Betroffene Felder: UpdateHistory (OriginalValue, NewValue, RolledBack, RollbackDate, RolledBackByI3D)

Auswirkungen: Fehlerhafte Updates können korrigiert werden. Audit-Trail ist vollständig. Compliance wird unterstützt.

14.3 Kostenträger/Kostenstellen (Cost Centers)

Module Path: `src/centron/Centron.WPF.UI/Modules/PayersAndCostCenter`
Controller: `PayersAndCostCenterAppModuleController`
ViewModel: `PayersAndCostCenterAppModuleControllerViewModel`
Category: Stammdaten
Description: Verwaltung von Kostenstellen und Kostenträgern für innerbetriebliche Kostenrechnung und Budgetierungskontrolle

Use Cases

14.3.1 Kostenstelle erstellen und strukturieren

Zweck: Definition von Kostenstellen für Kostenrechnung und Budget-Planung

Ablauf:

1. Controller erstellt Kostenstelle: "Sales", "Administration", "Produktion"
2. Hierarchie wird definiert: Main-Kostenstelle → Sub-Kostenstellen
3. Verantwortlicher wird zugewiesen
4. Budget wird zugeordnet
5. Kostenstelle ist in Belegen selektierbar

Betroffene Felder: CostCenter (Name, ParentCostCenterI3D, ManagerI3D, Budget)

Auswirkungen: Kostenverantwortung wird klar. Budgets werden gesteuert. Kosten werden verursachergerecht verteilt.

14.3.2 Kostenträger und Projekt-Kostenrechnung

Zweck: Verfolgung von Kosten auf Projekte oder Produkte

Ablauf:

1. Project Manager erstellt Kostenträger für Projekt "Website Redesign"
2. Alle Kosten werden diesem Kostenträger zugeordnet
3. Externe Kosten, Personalkosten, Materialkosten werden erfasst
4. Report zeigt Gesamt-Projektkosten und Rentabilität
5. Abweichungen von Projektbudget werden gewarnt

Betroffene Felder: CostCarrier (ProjectI3D, TotalCost, CostCenter), CostAllocation

Auswirkungen: Projektrentabilität wird genau berechnet. Budgets werden eingehalten. Management kann informiert entscheiden.

14.3.3 Budgetierung und Forecast

Zweck: Planung und Überwachung von Budgets pro Kostenstelle

Ablauf:

1. Controller definiert für 2025: "Sales: 500k€, Administration: 200k€"
2. Monthly Forecast wird erstellt basierend auf bisherigem Spend
3. System warnt bei Budgetüberschreitungen (80%, 95%, 100%)
4. Manager kann Forecast anpassen
5. Prognose wird in Finanzberichte integriert

Betroffene Felder: CostCenterBudget (Amount, ForecastAmount, ActualSpent, VarianceAmount)

Auswirkungen: Budget-Kontrolle wird proaktiv. Überraschungen werden vermieden. Finanzierungssicherheit ist gegeben.

14.3.4 Kosten-Umlagen und Verteilungen

Zweck: Verteilung von Gemeinkosten auf Kostenstellen

Ablauf:

1. CFO definiert Verteilschlüssel: "Miete anteilig nach Fläche"
2. System berechnet automatisch Anteile je Kostenstelle
3. Miete wird verteilt: Produktion 60%, Admin 30%, Sales 10%
4. Verteilte Kosten werden als Kostenrechnung verbucht
5. Rentabilität pro Kostenstelle wird genau berechnet

Betroffene Felder: CostAllocation (CostCenterId3D, AllocationPercentage, AllocatedAmount)

Auswirkungen: Kosten werden verursachergerecht zugeordnet. Rentabilität wird transparent. Entscheidungen werden faktenbasiert.

14.3.5 Kostenstellen-Reporting und Analyse

Zweck: Analyse von Kostenverantwortung und Leistung

Ablauf:

1. Manager kann Report anfordern: "Kostenentwicklung Sales 2024 vs 2025"
2. System zeigt: Budget, Forecast, Actual, Abweichung
3. Trends werden identifiziert: "Personalkosten steigen um 8%"
4. Vergleiche zwischen Kostenstellen möglich
5. Reports können geplant und per Email versendet werden

Betroffene Felder: CostCenterReport (CostCenterId3D, Period, BudgetAmount, ActualAmount, VariancePercentage)

Auswirkungen: Kostenmanagement wird datengetrieben. Effizienzpotenziale werden identifiziert. Optimierungen werden gezielt umgesetzt.

14.4 Länderverwaltung (Country Management)

Module Path: `src/centron/Centron.WPF.UI/Modules/Administration/CountryManagement`
Controller: `CountryManagementAppModuleController`
ViewModel: `CountryManagementViewModel`
Category: Stammdaten
Description: Verwaltung von Ländern, Regionen, Steuersätze und länderspezifische Regulations-Anforderungen

Use Cases

14.4.1 Länder und Regionen konfigurieren

Zweck: Definition von Ländern mit regionalen Abgrenzungen und Eigenschaften

Ablauf:

1. Admin erstellt Land: Deutschland
2. Regionen werden hinzugefügt: Baden-Württemberg, Bayern, Berlin, etc.
3. Lieferzonen werden definiert: Nord, Süd, Ost, West
4. Standardwährung wird gesetzt: EUR
5. Sprache wird zugeordnet: Deutsch

Betroffene Felder: Country (Name, Currency, Language), CountryRegion (Name, CountryI3D)

Auswirkungen: Versand wird optimal organisiert. Währungskonvertierung wird automatisiert. Kundensprache wird respektiert.

14.4.2 Steuersätze pro Land verwalten

Zweck: Definition von MwSt-Sätze und Steuerkategorien pro Land

Ablauf:

1. Admin konfiguriert Steuersätze für Deutschland: 19% Standard, 7% ermäßigt, 0% Export
2. Für Schweiz: 7.7% Standard, 2.5% reduziert
3. Für Österreich: 20% Standard, 10% reduziert
4. Gültigkeitsdatum wird definiert (von/bis)
5. System wendet Steuersätze automatisch an

Betroffene Felder: TaxRate (CountryI3D, TaxPercentage, TaxCategory, ValidFrom, ValidTo)

Auswirkungen: Rechnungen werden korrekt versteuert. Compliance wird gewährleistet. Automatische Kalkulation funktioniert.

14.4.3 Versand- und Lieferbestimmungen pro Land

Zweck: Definition von länder- und regionenspezifischen Versand-Regelungen

Ablauf:

1. Admin definiert: "Deutschland: Versand in 2-3 Tagen, Porto 4,99€"
2. "Österreich: Versand in 3-5 Tagen, Porto 7,99€"
3. "Schweiz: Nur Kurier möglich, Porto 15,99€"
4. System warnt bei nicht-lieferbaren Ländern
5. Versandinformationen werden in Bestell-Bestätigung angezeigt

Betroffene Felder: CountryDeliveryInfo (CountryI3D, DeliveryDays, ShippingCost, IsShippingAvailable)

Auswirkungen: Versandkosten sind transparent. Liefersicherheit ist gegeben. Kundenerwartungen sind klar.

14.4.4 Compliance und Regulations-Anforderungen

Zweck: Dokumentation von länderspezifischen regulatorischen Anforderungen

Ablauf:

1. Admin konfiguriert für Deutschland: "DSGVO-Compliance erforderlich"
2. Für Schweiz: "DataProtectionAct", "MwSt-Nummern erforderlich"
3. System erinnert Admin bei Implementierung neuer Features

4. Compliance-Checks werden in Prozesse integriert
5. Audit-Reports zeigen Compliance-Status

Betroffene Felder: CountryRegulation (CountryI3D, RegulatoryRequirement, Description, ImplementedI3D)

Auswirkungen: Compliance wird gewährleistet. Risiken werden minimiert. Geschäftsfortbestand ist sicher.

14.4.5 Länderpräferenzen und Konfiguration

Zweck: Aktivierung/Deaktivierung von Ländern und Verwaltung von Landinformationen

Ablauf:

1. Admin kann Länder aktivieren/deaktivieren: "Ist Deutschland aktiv? Ja"
2. Währungs-Konvertierung wird konfiguriert: "EUR zu CHF Rate 0.95"
3. Telefon-Formate werden definiert: "+49..." für Deutschland
4. Adressformate werden definiert: "Postleitzahl vor Stadt"
5. Konfiguration wird gespeichert und sofort wirksam

Betroffene Felder: Country (IsActive, ExchangeRate, PhoneFormat, AddressFormat)

Auswirkungen: Internationale Geschäftstätigkeit wird unterstützt. Dateneingabe wird automatisiert. Fehler werden minimiert.

14.5 Mehrwertsteuer (VAT/Sales Tax)

Module Path: `src/centron/Centron.WPF.UI/Modules/Warehousing`
Controller: `ValueAddedTaxAppModuleController`
ViewModel: `ValueAddedTaxViewModel`
Category: Stammdaten
Description: Verwaltung von Mehrwertsteuersätze, Steuerkategorien und Steuerbefreiungen für Artikel und Transaktionen

Use Cases

14.5.1 Steuerkategorien und Sätze definieren

Zweck: Definition von Mehrwertsteuerkategorien mit Sätzen

Ablauf:

1. Admin erstellt Steuerkategorie: "Standardartikel" = 19%
2. Weitere Kategorien: "Lebensmittel" = 7%, "Medikamente" = 0%
3. Jeder Artikel wird Kategorie zugeordnet
4. System wendet Steuersatz automatisch in Rechnungen an
5. Kategorien sind durchsuchbar und verwaltbar

Betroffene Felder: TaxCategory (Name, TaxRate, Description), ValueAddedTax (ArticleI3D, TaxCategoryI3D)

Auswirkungen: Rechnungen werden automatisch korrekt versteuert. Fehler werden eliminiert. Compliance wird garantiert.

14.5.2 Steuerbefreiungen und Ausnahmen

Zweck: Verwaltung von Steuerbefreiungen für Export oder spezielle Kunden

Ablauf:

1. Admin erstellt Ausnahmeregelung: "Export außerhalb EU = 0% MwSt"
2. Bedingung: Lieferadresse ist außerhalb EU
3. System wendet automatisch 0% MwSt an

4. Kunde muss EU-MwSt-Nummer angeben
5. System validiert MwSt-Nummer

Betroffene Felder: TaxExemption (Condition, TaxRate, ValidFrom, ValidTo), CustomerTaxID

Auswirkungen: Export-Umsätze werden korrekt berechnet. Compliance mit Umsatzsteuer-Richtlinie. Kostenersparnis für Exporte.

14.5.3 Steuersätze nach Lieferland

Zweck: Anwendung länderspezifischer Steuersätze auf Lieferungen

Ablauf:

1. Admin konfiguriert: "Lieferung nach Österreich = 20% MwSt"
2. "Lieferung nach Schweiz = 7.7% MwSt"
3. "Lieferung nach Luxemburg = 17% MwSt"
4. System bestimmt automatisch MwSt-Satz basierend auf Lieferadresse
5. In Rechnungen wird korrekter Satz angezeigt

Betroffene Felder: CountryTaxRate (CountryI3D, TaxPercentage), ReceiptTable (DeliveryCountryI3D, AppliedTaxRate)

Auswirkungen: Mehrländer-Verkauf wird korrekt abgewickelt. Rechnungen sind landeskonform. Steuererklärung wird erleichtert.

14.5.4 Steuersatz-Änderungen verwalten

Zweck: Verwaltung von Steuersatz-Änderungen (z.B. Januar 2024: 19% → 20%)

Ablauf:

1. Admin erfasst Satzänderung: "Ab 01.01.2024: 19% → 20%"
2. Gültigkeitsdatum wird gesetzt
3. System erkennt automatisch: Alte Rechnungen 19%, neue Rechnungen 20%
4. Übergangszeitraum kann konfiguriert werden
5. Reports zeigen Auswirkungen der Änderung

Betroffene Felder: TaxRateHistory (OldRate, NewRate, EffectiveDate, ChangedByI3D), ValueAddedTax (ValidFrom, ValidTo)

Auswirkungen: Gesetzesänderungen werden umgesetzt. Fehler werden vermieden. Audit-Trail ist dokumentiert.

14.5.5 Steuerzahlungen und Abrechnung

Zweck: Reporting und Verarbeitung von Steuerzahlungen

Ablauf:

1. Controller kann Steuerzahlung Report anfordern: "MwSt-Schuld Q1 2025"
2. System berechnet automatisch: Eingangssteuern - Ausgangssteuern = Zahllast
3. Report zeigt Details pro Steuerkategorie
4. System kann Steuererklärung-Export generieren (ELSTER-Format)
5. Zahlungsanweisung kann ausgedruckt werden

Betroffene Felder: TaxPayment (Period, InputTax, OutputTax, PaymentAmount, PaymentDate), ReceiptTable (AppliedTaxRate, TaxAmountI3D)

Auswirkungen: Steuerzahlungen werden präzise berechnet. Zahlfristen werden eingehalten. Compliance ist garantiert.

14.6 Projektpreis Import (Project Price Import)

Module Path: `src/centron/Centron.WPF.UI/Modules/ProjectPriceImport`
Controller: `ProjectPriceImportAppModuleController`
ViewModel: `ProjectPriceImportViewModel`
Category: Stammdaten
Description: Import und Verwaltung von Projekt-spezifischen Preisen aus externen Quellen mit Validierung und Anwendung auf Bestellungen

Use Cases

14.6.1 Projekt-Preis-Tabelle importieren

Zweck: Import von Customer-spezifischen oder Projekt-spezifischen Preisen

Ablauf:

1. Admin navigiert zu "Projekt-Preis Import"
2. Excel-Datei wird hochgeladen mit Spalten: Artikelnummer, Kundennummer, Projekt, Preis
3. System validiert Datei: Alle Artikel existieren? Alle Kunden existieren?
4. Vorschau wird angezeigt: "100 Artikel für Projekt 'Website' werden importiert"
5. Import wird durchgeführt, Projekte werden aktualisiert

Betroffene Felder: ProjectPrice (ArticleI3D, ProjectI3D, SpecialPrice, ValidFrom, ValidTo), ImportLog

Auswirkungen: Projekt-spezifische Preise sind verfügbar. Bestellungen werden mit korrekten Preisen berechnet. Kundenabos werden erfüllt.

14.6.2 Projekt-Preise in Bestellungen anwenden

Zweck: Automatische Anwendung von Projekt-Preisen auf Bestellpositionen

Ablauf:

1. Benutzer erstellt Bestellung für Projekt "Website"
2. System erkennt: Für diesen Kunden + dieses Projekt existieren spezielle Preise
3. Bestellposition wird mit Projekt-Preis berechnet (nicht Standard-Preis)
4. In Bestellung wird angezeigt: "Mit Projekt-Rabatt: 15€ statt 20€"
5. System dokumentiert Preis-Anwendung

Betroffene Felder: ReceiptItems (PriceI3D, AppliedProjectPriceI3D, OriginalPrice, DiscountPercentage)

Auswirkungen: Kunden-Vereinbarungen werden eingehalten. Rabatte werden automatisch gewährt. Beschwerde werden minimiert.

14.6.3 Preis-Effektivität und Rabatt-Tracking

Zweck: Analyse von angewendeten Projekt-Preisen und Rabatten

Ablauf:

1. Admin kann Report anfordern: "Rabatte nach Projekt für Q1 2025"
2. System zeigt: "Projekt 'Website': 1.200€ Rabatt auf 50 Bestellungen"
3. Durchschnittlicher Rabatt wird berechnet: 24€ pro Bestellung
4. Entwicklung wird verfolgt: Rabatte steigen oder sinken?
5. Management kann Rabatt-Strategien datengetrieben anpassen

Betroffene Felder: ProjectPrice (SpecialPrice, DiscountPercentage), PricingReport (ProjectI3D, TotalDiscount, AverageDiscount)

Auswirkungen: Rabatt-Strategien werden optimiert. Profitabilität wird kontrolliert. Preismanagement wird datengetrieben.

14.6.4 Gültigkeitsdauern und Ablauf

Zweck: Verwaltung von Gültigkeit von Projekt-Preisen

Ablauf:

1. Admin legt fest: "Diese Preise gültig vom 01.01.2025 bis 31.03.2025"
2. System automatisch am 01.04.2025: "Diese Preise sind abgelaufen"
3. Bestellungen nach 31.03 verwenden wieder Standard-Preise
4. Benachrichtigung wird versendet: "Projekt-Preise werden neu verhandelt"
5. Preis-Erneuerung kann administrativ ausgelöst werden

Betroffene Felder: ProjectPrice (ValidFrom, ValidTo), SystemNotification (ExpirationWarning, ProjectI3D)

Auswirkungen: Preis-Verhandlungen werden zeitgerecht geführt. Preisexplosionen werden vermieden. Kundenbeziehungen bleiben gepflegt.

14.6.5 Sonder-Vereinbarungen verwalten

Zweck: Verwaltung von Sonder-Vereinbarungen für bestimmte Artikel oder Kunden

Ablauf:

1. Admin erstellt Sonder-Vereinbarung: "Kunde ABC erhält 25% Rabatt auf alle IT-Artikel"
2. Bedingung: Nur für Projekt "Netzwerk-Upgrade"
3. System wendet automatisch 25% Rabatt an
4. Vereinbarung ist zeitlich begrenzt: "Gültig bis 31.12.2025"
5. Reports zeigen Einhaltung der Vereinbarung

Betroffene Felder: SpecialAgreement (CustomerI3D, ArticleCategoryI3D, ProjectI3D, DiscountPercentage, ValidFrom, ValidTo)

Auswirkungen: Sonder-Vereinbarungen werden automatisch eingehalten. Kundenvertrauen wird gestärkt. Streitfälle werden vermieden.

14.7 Reportverwaltung (Report Management)

Module Path: `src/centron/Centron.WPF.UI/Modules/Reports/ReportManagement`
Controller: `ReportEngineAppModuleController`
ViewModel: `ReportEngineAppModuleControllerViewModel`
Category: Stammdaten
Description: Verwaltung von Geschäftsberichten mit Erstellung, Planung und Distribution von Auswertungen

Use Cases

14.7.1 Report-Templates erstellen

Zweck: Erstellung von wiederverwendbaren Report-Vorlagen

Ablauf:

1. Report Designer öffnet Report-Tool
2. Report-Template wird erstellt: "Verkaufs-Übersicht nach Region"
3. Query wird definiert: SELECT Umsatz, Region, Kunde FROM...
4. Layout wird formatiert: Header, Tabellen, Diagramme
5. Template wird gespeichert und ist sofort nutzbar

Betroffene Felder: Report (Name, Query, Layout, CreatedByI3D), ReportTemplate (TemplateXML)

Auswirkungen: Reports werden standardisiert. Erstellung wird beschleunigt. Konsistenz wird gewährleistet.

14.7.2 Reports zeitlich planen

Zweck: Zeitgesteuerte automatische Report-Erstellung und Versand

Ablauf:

1. Manager plant Report: "Verkaufs-Report täglich um 6:00 Uhr"
2. Report-Parameter werden gesetzt: "Zeige Umsatz letzte 24 Stunden"
3. Versand wird konfiguriert: "Per Email an: sales-team@company.com"
4. Nach Planung läuft Report automatisch
5. Reports werden archiviert

Betroffene Felder: ReportSchedule (ReportI3D, SchedulePattern, ScheduledTime, Recipients), ReportDelivery

Auswirkungen: Manager erhalten automatisch aktuelle Daten. Entscheidungen werden schneller getroffen. Konsistente Reporting wird sichergestellt.

14.7.3 Report-Filter und Dimensionen

Zweck: Flexibles Filtering von Reports für verschiedene Perspektiven

Ablauf:

1. Benutzer öffnet Report "Verkaufs-Übersicht"
2. Filter werden angewendet: "Nur Region 'Süd'" und "Nur Top 10 Kunden"
3. Zeitraum wird gefiltert: "Letzten 12 Monate"
4. Drilldown möglich: Click auf Region zeigt Kundendetails
5. Report wird nach Filterung neu berechnet

Betroffene Felder: ReportFilter (ReportI3D, FilterName, FilterValue, FilterOperator)

Auswirkungen: Reports werden personalisierbar. Benutzer können tiefere Analysen durchführen. Fragen werden schneller beantwortet.

14.7.4 Report-Verteilung und Genehmigung

Zweck: Automatische Verteilung mit Genehmigungslogik für sensitive Reports

Ablauf:

1. Finance Manager erstellt Report "Profitabilität nach Projekt"
2. Genehmigungsregel wird gesetzt: "Muss von CFO genehmigt werden"
3. Report wird an CFO zur Genehmigung versendet
4. CFO genehmigt oder lehnt ab
5. Nach Genehmigung wird Report an Management-Team versendet

Betroffene Felder: ReportApproval (ReportI3D, ApproverI3D, Status, ApprovedDate), ReportDistribution

Auswirkungen: Sensitive Informationen sind geschützt. Governance wird eingehalten. Verteilung ist kontrolliert.

14.7.5 Report-Archivierung und Compliance

Zweck: Langzeitarchivierung von Reports für Compliance-Anforderungen

Ablauf:

1. System archiviert täglich Reports automatisch
2. Aufbewahrungsfrist wird konfiguriert: "5 Jahre für Finanz-Reports"
3. Benutzer können alte Reports abrufen
4. Audit-Trail zeigt wer Report wann aufgerufen hat
5. Compliance-Reports zeigen Aufbewahrungsstatus

Betroffene Felder: ReportArchive (ReportI3D, ArchiveDate, RetentionEndDate), ReportAccessLog (UserI3D, AccessDate)

Auswirkungen: Compliance-Anforderungen erfüllt. Historische Daten verfügbar. Audit-Trail vorhanden.

14.8 Warengruppenverwaltung (Product Group Management)

Module Path: `src/centron/Centron.WPF.UI/Modules/Warehousing/MaterialGroupManagement`
Controller: `MaterialGroupAppModuleController`
ViewModel: `MaterialGroupManagementMainView`
Category: Stammdaten
Description: Verwaltung von Produktgruppen und Material-Kategorien für Klassifizierung und Segmentierung von Artikeln

Use Cases

14.8.1 Warengruppen-Hierarchie aufbauen

Zweck: Strukturierung von Artikeln in hierarchische Kategorien

Ablauf:

1. Admin erstellt Hauptgruppe: "IT Hardware"
2. Untergruppen werden hinzugefügt: "Computer", "Peripherie", "Netzwerk"
3. Weitere Verschachtelung: "Computer" → "Laptops", "Desktops", "Server"
4. Jeder Artikel wird einer Leaf-Kategorie zugeordnet
5. Hierarchie ist nutzbar für Navigation und Reporting

Betroffene Felder: MaterialGroup (Name, ParentMaterialGroupI3D, Level, Description), MaterialGroupHierarchy

Auswirkungen: Katalog wird organisiert. Navigation wird intuitiv. Verkaufsanalyse wird möglich.

14.8.2 Warengruppen-Eigenschaften und Standards

Zweck: Definition von Standard-Eigenschaften pro Warengruppe

Ablauf:

1. Admin definiert für Gruppe "Laptops":
   - Standard-Lagerdauer: 90 Tage
   - Standard-Margenerwartung: 20%
   - Standard-Lieferant: Lenovo
2. Neue Artikel dieser Gruppe erben diese Standards
3. Reports können pro Gruppe analysiert werden
4. Abweichungen werden identifiziert

Betroffene Felder: MaterialGroup (DefaultShelfLife, DefaultMargin, PreferredSupplierI3D), MaterialGroupArticle

Auswirkungen: Standards werden konsistent. Artikel-Management wird erleichtert. Kalkulation wird standardisiert.

14.8.3 Steuern und Konditionen pro Warengruppe

Zweck: Zuweisung von Steuersätzen und Konditionen auf Warengruppen-Ebene

Ablauf:

1. Admin definiert: "Lebensmittel" = 7% MwSt
2. "Technische Geräte" = 19% MwSt
3. "Bücher" = 7% MwSt

4. System wendet automatisch Steuersätze an basierend auf Warengruppe
5. Neue Artikel erben automatisch Steuersatz der Gruppe

Betroffene Felder: MaterialGroup (TaxCategoryI3D, DefaultPaymentConditionI3D), ValueAddedTax

Auswirkungen: Steuern werden korrekt berechnet. Fehler werden vermieden. Compliance wird gewährleistet.

14.8.4 Warengruppen-Leistung und Analytics

Zweck: Analyse von Umsatz, Rentabilität und Performance pro Warengruppe

Ablauf:

1. Manager kann Report anfordern: "Umsatz nach Warengruppe Q1 2025"
2. System zeigt: "IT Hardware: 500k€", "Office Supplies: 200k€", "Sonstiges: 100k€"
3. Rentabilität wird berechnet: "IT Hardware: 22% Marge", "Office: 18%"
4. Trends werden angezeigt: "IT Hardware wächst +15% YoY"
5. Unterperformance wird identifiziert

Betroffene Felder: MaterialGroupPerformance (MaterialGroupI3D, Revenue, Margin, Growth%), SalesAnalytics

Auswirkungen: Portfolio wird datengetrieben optimiert. Ressourcen werden gezielt eingesetzt. Profitabilität wird maximiert.

14.8.5 Warengruppen-Umstrukturierung und Umbau

Zweck: Verwaltung von Änderungen an Warengruppen-Struktur

Ablauf:

1. Admin plant Umstrukturierung: "Merging von 'Old Models' in 'Refurbished'"
2. Artikel werden neu zugeordnet
3. 250 Artikel werden verschoben
4. System validiert: Keine Artikel werden verloren
5. Nach Bestätigung wird Struktur aktualisiert, alte Gruppe wird archiviert

Betroffene Felder: MaterialGroup (Status, ArchivedDate, SuccessorGroupI3D), MaterialGroupArticle (MigratedDate)

Auswirkungen: Katalog wird aktuell. Organisationsänderungen werden abgebildet. Kontinuität wird gewährleistet.

15. Verträge (Contracts)

15.1 Dynamischer Datenimport - Verträge (Dynamic Data Import - Contracts)

Module Path: `src/centron/Centron.WPF.UI/Modules/Sales/SpecialArticleToContractImport`
Controller: `SpecialArticleToContractImportAppModuleController`
ViewModel: `SpecialArticleToContractImportViewModel`
Category: Verträge
Description: Automatischer Import von Vertragspositionsdaten aus externen Quellen für die Abrechnung

Use Cases

15.1.1 Externe Datenquellen verbinden und konfigurieren

Zweck: Verbindung zu externen Datenquellen (Lieferanten-APIs, Gateway-Systeme) aufbauen und konfigurieren für automatisierte Datenabfrage

Ablauf:

1. Benutzer öffnet Dynamischer Datenimport Modul
2. Wählt externe Datenquelle aus (z.B. Custom Gateway, Lieferanten-System)
3. Konfiguriert API-Credentials und Verbindungsparameter
4. Testet Verbindung zur Datenquelle
5. Speichert Konfiguration für wiederholte Nutzung
6. System validiert Verbindungsdetails

Betroffene Felder: Gateway-Typ, API-Endpoint, Authentication-Token, Verbindungs-Timeout, Retry-Logik

Auswirkungen:

- Ermöglicht vollautomatisierte Datenabfrage ohne manuelle Eingriffe
- Reduziert Fehler bei der Dateneingabe
- Erhöht Verarbeitungsgeschwindigkeit von Vertragsabrechnungen

15.1.2 Vertragspositionsdaten automatisch abrufen und importieren

Zweck: Automatisierter Abruf von Vertragspositionsdaten aus externen Systemen und Import in c-entron

Ablauf:

1. Benutzer definiert Abfrage-Filter (Kunde, Vertragstyp, Zeitraum)
2. System stellt Query zur konfigurierten Datenquelle
3. Externe Daten werden formatiert und validiert
4. Vertragspositionsdaten werden mit bestehenden Verträgen verknüpft
5. System zeigt Preview der zu importierenden Daten
6. Benutzer bestätigt Import oder korrigiert Zuordnungen
7. Daten werden in Contract-Tabelle eingefügt

Betroffene Felder: ContractI3D, PositionNumber, Description, Quantity, UnitPrice, Currency, ImportDate, SourceSystem

Auswirkungen:

- Automatische Anreicherung von Vertragsabrechnungsdaten
- Wegfall von manuellen Copy-Paste-Operationen
- Erhöhte Datenqualität durch standardisierte Imports
- Zeitersparnis bei regelmäßigen Abrechnungsläufen

15.1.3 Automatische Preisberechnung aus mehreren Quellen

Zweck: Berechnung von korrekten Preisen für Vertragspositionen basierend auf Daten aus mehreren Quellen

Ablauf:

1. System lädt Artikel-Stammdaten und Kundenpreise
2. Ruft externe Preisquellen ab (Lieferanten, Marktpreise, Spezialtarife)
3. Berechnet beste/durchschnittliche/kundenspezifische Preise
4. Wendet Rabatt-/Bonusregeln an
5. Berücksichtigt Währungsumrechnung und Steuern
6. Hinterlegt berechnete Preise in Vertragspositionen
7. Dokumentiert Preisquelle für Audit-Trail

Betroffene Felder: UnitPrice, CurrencyCode, DiscountPercent, TaxRate, CalculatedPrice, PriceSource

Auswirkungen:

- Sichert Preisgenauigkeit bei Abrechnungen
- Automatische Anwendung von Kundenvergünstigungen

- Verhindert Abweichungen zwischen Quelle und Abrechnung
- Unterstützt Multi-Währungs-Szenarien

15.1.4 Fehlerbehandlung und Validierung bei Datenabweichungen

Zweck: Erkennung und Behebung von Datenfehlern während des Imports, um Abweichungen zu minimieren

Ablauf:

1. System führt Validierungsprüfungen durch (Formatprüfung, Wertebereich, Referenz-Integrität)
2. Identifiziert fehlende oder ungültige Daten
3. Erzeugt Fehler-Report mit Details zur Abweichung
4. Bietet Benutzer Optionen: Datensatz überspringen, Korrektur vorschlagen oder manuell bearbeiten
5. Speichert korrigierte Daten zur Wiederholung
6. Benutzer erhält Zusammenfassung importierter vs. fehlgeschlagener Datensätze
7. Fehlerbehandlung wird geloggt für Compliance

Betroffene Felder: ValidationErrorType, ErrorMessage, DataQuality, FieldValidation, ReferenceIntegrity

Auswirkungen:

- Verhindert fehlerhafte Vertragsabrechnungen
- Zentraler Überblick über Datenqualitätsprobleme
- Dokumentation aller Import-Fehler für Audit
- Ermöglicht iterative Korrektur und Neuimporte

15.1.5 Massen-Import und Planungsintervalle

Zweck: Automatisierte Planung und Durchführung von regelmäßigen Massen-Importen ohne Benutzerinteraktion

Ablauf:

1. Benutzer erstellt Import-Profil mit Zeitplan (täglich, wöchentlich, monatlich)
2. Definiert Datenquelle, Filter und Validierungsregeln
3. Stellt Zeitpunkt für automatische Durchführung ein
4. System führt Import zu geplanter Zeit aus
5. Sendet Benachrichtigung über Import-Erfolg/-Fehler
6. Erstellt regelmäßig Audit-Report über alle durchgeführten Importe
7. Benutzer kann jeden Import-Lauf einsehen und ggfs. rückgängig machen

Betroffene Felder: ScheduleType, ScheduledTime, RecurrencePattern, LastExecutionDate, NextScheduledRun, AutomationLevel

Auswirkungen:

- Vollständige Automatisierung von regelmäßigen Importen
- Keine manuellen Eingriffe für wiederkehrende Prozesse
- Konsistente Datenqualität über alle Import-Läufe
- Zentrale Überwachung von Import-Aktivitäten

15.1.6 Import-Ergebnisse analysieren und nachbearbeiten

Zweck: Detaillierte Analyse von durchgeführten Importen und Möglichkeit zur Nachbearbeitung

Ablauf:

1. Benutzer öffnet Import-Verlauf und sucht spezifischen Import-Lauf
2. System zeigt Import-Statistiken (Anzahl importiert, aktualisiert, fehlerhafte)
3. Benutzer kann einzelne Datensätze durchsehen und Änderungen prüfen

4. Kann fehlgeschlagene Datensätze erneut verarbeiten
5. Kann Daten vor Abrechnung noch korrigieren (z.B. Preise anpassen)
6. Exportiert Import-Report für Dokumentation
7. Archiviert Import-Protokoll für zukünftige Audits

Betroffene Felder: ImportResult, SuccessCount, ErrorCount, UpdatedCount, ModificationDate, ApprovalStatus

Auswirkungen:

- Volle Transparenz über durchgeführte Datenoperationen
- Möglichkeit zur Qualitätskontrolle vor Abrechnung
- Audit-Sicherheit durch vollständige Dokumentation
- Nachverfolgung von Datenänderungen möglich

15.2 Klick-Zählerverwaltung (Click Counter Management)

Module Path: `src/centron/Centron.WPF.UI/Modules/Finances/DeviceClickCounter`
Controller: `DeviceClickCounterAppModuleController`
ViewModel: `DeviceClickCounterViewModel`
Category: Verträge
Description: Verwaltung von Klick-Zählern für Geräteabrechnung (Kopier-/Drucker-Seiten)

Use Cases

15.2.1 Click-Counter-Lesevorgänge manuell erfassen und verarbeiten

Zweck: Erfassung von manuell abgelesenen Klick-Zähler-Werten (z.B. von Kopierern) für Abrechnung

Ablauf:

1. Servicetechniker erstellt neuen Counter-Leseeintrag
2. Erfasst Geräteseriennummer und Lesedatum
3. Gibt aktuelle Zähler-Lesewert ein (z.B. "156.234 Kopien")
4. Optional: Erfasst Zusatzinformationen (Fehler, Wartung, Toner-Wechsel)
5. System validiert Plausibilität (Zähler sollte nur steigen, nicht sinken)
6. Speichert Lesedatum und -wert als Abrechnungsgrundlage
7. Berechnet verbrauchte Kopien seit letztem Lesedatum

Betroffene Felder: DeviceSerialNumber, ReadingDate, CounterValue, CounterDifference, ReadingType, LocationInfo, TechnicianID

Auswirkungen:

- Dokumentation von Geräte-Nutzung für kundengerechte Abrechnung
- Plausibilitätskontrolle verhindert fehlerhafte Zählerstände
- Audit-Trail für alle Counter-Lesevorgänge
- Basis für zeitgenauere Abrechnung

15.2.2 Zählerstände aus Excel/CSV importieren

Zweck: Automatischer Massenimport von Zählerständen aus Excel- oder CSV-Dateien (z.B. von Kunde oder Lieferant)

Ablauf:

1. Benutzer öffnet Import-Dialog für Excel/CSV-Dateien
2. Wählt Datei mit Zählerständen aus (Format: Geräte-ID, Zähler, Datum)
3. System parst Datei und validiert Spaltenformat
4. Zeigt Preview der zu importierenden Daten
5. Benutzer korrigiert ggfs. Zuordnungen (Spalten-Mapping)

6. Stellt Lesedatum und Validierungsregeln ein
7. System importiert Zähler und verknüpft mit Verträgen
8. Erstellt Import-Report mit Erfolgs-/Fehlerquote

Betroffene Felder: FileName, ImportDate, ParsedFormat, ColumnMapping, ValidationRules, ImportStatus

Auswirkungen:

- Schnelle Verarbeitung von großen Zählerdatenmengen
- Manuelles Abschreiben entfällt
- Standardisierte Verarbeitung von Lieferanten-Daten
- Zeitersparnis bei regelmäßigen Importen

15.2.3 Zähler mit Verträgen verknüpfen und verwalten

Zweck: Zuordnung von Zählergeräten zu Service-Verträgen für die Verbrauchsabrechnung

Ablauf:

1. Benutzer öffnet Vertrag und navigiert zu Counter-Verwaltung
2. Wählt Gerät aus (nach Seriennummer oder Gerätebezeichnung)
3. Definiert Zählerverhältnis und Abrechnungsfrequenz
4. Setzt Start- und Enddatum der Counter-Zuordnung
5. Optional: Erstellt mehrere Counter pro Gerät (z.B. Farb- und Schwarzweiß-Seiten)
6. System speichert Counter-Konfiguration
7. Zeigt Abrechnung-Simulationen basierend auf aktuellen Zählerständen

Betroffene Felder: ContractI3D, DeviceI3D, CounterStartDate, CounterEndDate, CounterMultiplier, BillingFrequency

Auswirkungen:

- Klare Zuordnung zwischen Geräten und Verträgen
- Ermöglichung von flexibler Abrechnung je Gerät
- Verhindert Doppel-Abrechnungen oder fehlende Geräte
- Basis für automatisierte Rechnungserstellung

15.2.4 Counter-Leseverlauf und Trend-Analyse

Zweck: Überwachung des Counter-Leseverlaufs und Analyse von Nutzungstrends zur Optimierung

Ablauf:

1. Benutzer wählt Gerät/Counter und Zeitraum aus
2. System zeigt alle Lesevorgänge chronologisch
3. Berechnet durchschnittliche Nutzung pro Monat/Woche
4. Erstellt Trendgrafik (Nutzungsentwicklung)
5. Identifiziert Spitzen oder Ausfälle
6. Vergleicht Nutzung mit Kundenhistorie und Vertrag
7. Berechnet erwartete Abrechnung basierend auf Trend

Betroffene Felder: ReadingDate, CounterValue, AverageUsage, UsageTrend, AnomalyDetection, PredictedNextReading

Auswirkungen:

- Früherkennung von Gerätedefekten oder Störungen
- Optimierungsmöglichkeiten bei Vertragsplanung
- Bessere Vorhersage zukünftiger Abrechnungen
- Kundenkommunikation über Nutzungsmuster

15.2.5 Verschiedene Counter-Formate verarbeiten (Riverbird, docuForm)

Zweck: Unterstützung verschiedener Zähler-Formate von unterschiedlichen Lieferanten und Systemen

Ablauf:

1. Benutzer wählt Counter-Format aus (z.B. Riverbird, docuForm, Standard)
2. System lädt entsprechendes Import-Profil
3. Parser konvertiert Format in Standard c-entron Counter-Format
4. Validiert Zählerstände nach Format-spezifischen Regeln
5. Benutzer überprüft konvertierte Daten
6. System speichert Counter unter Standard-Format
7. Dokumentiert Original-Format für Audit

Betroffene Felder: CounterFormat, FormatVersion, OriginalData, ConversionStatus, ValidationRules

Auswirkungen:

- Flexibilität bei verschiedenen Lieferanten
- Zentralisierte Counter-Verwaltung trotz unterschiedlicher Formate
- Vereinheitlichte Abrechnung unabhängig von Quelle
- Fehlerquellen bei Format-Konvertierung minimiert

15.2.6 Abrechnungsvorschläge generieren und freigeben

Zweck: Erstellung von Abrechnungsvorschlägen basierend auf Zählerständen zur Rechnungserstellung

Ablauf:

1. Benutzer startet Abrechnungsprozess für Zeitraum
2. System ruft letzte bestätigte Zählerstände ab
3. Berechnet Differenzen zum Abrechnungszeitraum
4. Wendet Preistabellen und Rabatte an
5. Erstellt Abrechnungsvorschlag mit Details
6. Benutzer prüft Vorschlag und gibt ggfs. Feedback
7. Genehmigt Abrechnungsvorschlag
8. System erstellt Rechnungspositionen automatisch

Betroffene Felder: BillingPeriod, LastConfirmedReading, CurrentReading, QuantityDifference, UnitPrice, TotalAmount, ApprovalStatus

Auswirkungen:

- Automatische, fehlerfreie Abrechnung von Counter-Verbrauch
- Transparenz über Abrechnungsgrundlagen für Kunde
- Schnellere Rechnungserstellung
- Nachverfolgbarkeit von Abrechnungsdaten

15.3 Statischer Datenimport - Verträge (Static Data Import - Contracts)

Module Path: `src/centron/Centron.WPF.UI/Modules/Sales/SpecialArticleImport`
Controller: `SpecialArticleImportAppModuleController`
ViewModel: `SpecialArticleToContractViewModel`
Category: Verträge
Description: Manuelle und dateibasierte Eingabe von Vertragspositionsdaten (statischer Datenimport)

Use Cases

15.3.1 Vertragspositionsdaten manuell erfassen und eingeben

Zweck: Manuelle Eingabe von Vertragspositionsdaten für Fälle, bei denen automatisierter Import nicht möglich ist

Ablauf:

1. Benutzer öffnet Vertrags-Details und wählt "Neue Position hinzufügen"
2. Wählt Artikel/Leistung aus oder erstellt Ad-Hoc-Position
3. Gibt Positionsnummer, Beschreibung, Menge und Preis ein
4. Optional: Erfasst Zusatzinformationen (Einheit, Rabatt, Steuerkategorie)
5. System validiert Eingabewerte (z.B. nicht-negative Mengen, gültige Preise)
6. Benutzer speichert Position
7. Position wird der Vertragsliste hinzugefügt und ist sofort für Abrechnung verfügbar

Betroffene Felder: ContractI3D, PositionNumber, Description, Quantity, UnitPrice, Unit, DiscountPercent, TaxCategory, EnterDate, EnteredByID

Auswirkungen:

- Flexible Erfassung für Spezial- und Einmaligkeitspos
- Keine Abhängigkeit von Artikel-Stammdaten
- Schnelle Eingabe für Ad-Hoc-Abrechnungen
- Manuelle Qualitätskontrolle möglich

15.3.2 Vertragspositionsdaten aus statischen Dateien importieren

Zweck: Einmalig oder regelmäßig Vertragspositionsdaten aus statischen Dateiquellen (Excel, CSV, Text) importieren

Ablauf:

1. Benutzer erstellt oder lädt Datenquelle-Datei
2. Wählt "Statischen Import" und definiert Quell-Datei
3. System erkennt Dateiformat und Spaltenstruktur
4. Zeigt Spalten-Mapping-Dialog (Zuordnung zu c-entron Feldern)
5. Benutzer validiert und korrigiert Mappings
6. System führt Test-Import für erste Zeile durch
7. Benutzer bestätigt Mapping und startet Volumen-Import
8. System importiert alle Datensätze und erstellt Vertrags-Positionen

Betroffene Felder: FileName, SourceFormat, ColumnMapping, ParsedRows, ImportedRows, ErrorRows, ImportDate

Auswirkungen:

- Massenimport von Vertragspositionsdaten
- Reduzierte manuelle Dateneingabe
- Wiederholbare Import-Prozesse definierbar
- Dokumentation der Datenquelle für Audit

15.3.3 Artikel-Zuordnung und Enrichment bei Import

Zweck: Automatische oder manuelle Zuordnung von importierten Positionen zu bestehenden Artikeln und Anreicherung mit Zusatzinformationen

Ablauf:

1. System versucht automatisch, importierte Positionsbeschreibungen zu Artikeln zuzuordnen
2. Nutzt Fuzzy-Matching auf Artikel-Namen und Beschreibungen
3. Bei unsicheren Matches zeigt System Vor-/Rückschläge zur manuellen Auswahl
4. Benutzer kann auch manuell Artikel für Position auswählen
5. Wenn Artikel zugeordnet: System lädt Artikel-Eigenschaften (Standardpreis, Steuern, Rabatte)

6. Optional: Enriched Positionen mit Artikel-Eigenschaften oder behält importierte Werte
7. Benutzer bestätigt finale Zuordnung

Betroffene Felder: PositionDescription, ArticleI3D, MatchConfidence, EnrichedFrom, StandardPrice, ArticleTax

Auswirkungen:

- Automatische Verknüpfung zu Stammdaten
- Konsistente Verwendung von Standardpreisen/-steuern
- Fehlerreduktion durch Validierung gegen Artikel-Stamm
- Datenqualität-Verbesserung

15.3.4 Validierung und Qualitätskontrolle vor Übernahme

Zweck: Umfassende Validierung importierter Daten vor Übernahme in Verträge, um Abweichungen zu verhindern

Ablauf:

1. System führt automatische Validierungsprüfungen durch:
   - Format-Prüfung (gültige Zahlen, Datentypen)
   - Referenz-Integrität (Verträge existieren, Artikel gültig)
   - Geschäftsregeln (z.B. Mindestmenge, Max-Preis)
   - Duplikat-Erkennung (Positionen doppelt?)
2. Erstellt Validierungs-Report mit Fehlern/Warnungen
3. Kategorisiert Fehler: "Kritisch" (Import abbrechen), "Warnung" (Import mit Hinweis)
4. Benutzer sieht Details jedes Fehlers mit Korrektur-Optionen
5. Kann Fehler-Datensätze überspringen oder korrigieren
6. Lädt korrigierte Daten neu zur Validierung
7. Nach erfolgreicher Validierung: Import genehmigen

Betroffene Felder: ValidationResult, ErrorType, ErrorSeverity, ErrorMessage, FieldName, SuggestedFix

Auswirkungen:

- Verhindert fehlerhafte Vertragspositionsdaten
- Früherkennung von Datenqualitätsproblemen
- Dokumentation von Validierungsfehlern
- Verbesserte Compliance und Audit-Sicherheit

15.3.5 Import-Vorschau und Änderungs-Tracking

Zweck: Transparente Vorschau von zu importierenden Daten vor finaler Übernahme

Ablauf:

1. Benutzer sieht Import-Preview mit tabellarischer Darstellung
2. Kann in der Preview Werte direkt editieren (Inline-Editing)
3. System zeigt ursprüngliche und editierte Werte nebeneinander
4. Highlights Unterschiede zu bestehenden Vertrags-Positionen (falls Update)
5. Zeigt Auswirkungen pro Zeile (z.B. Rechnungsbetrag-Änderung)
6. Benutzer kann einzelne Zeilen aus Import ausschließen
7. Kann Änderungen speichern als Template für zukünftige Importe
8. Nach Bestätigung werden Änderungen übernommen mit Audit-Timestamp

Betroffene Felder: OriginalValue, EditedValue, FieldModified, ChangeType, PreviewStatus, ChangeTimestamp

Auswirkungen:

- Volle Transparenz über zu importierende Änderungen

- Möglichkeit zur Feinabstimmung vor Übernahme
- Änderungs-Nachverfolgung für Compliance
- Fehlerreduktion durch visuelle Validierung

15.3.6 Import-Verlauf und Rollback-Funktionalität

Zweck: Dokumentation aller durchgeführten Importe mit Möglichkeit zur Rückgängigmachung

Ablauf:

1. System speichert alle durchgeführten Importe mit Datum, Benutzer, Details
2. Benutzer kann Import-Verlauf aufrufen und älteren Import auswählen
3. Zeigt Details des Imports: Datensätze, Änderungen, Betroffene Verträge
4. Benutzer kann Rollback durchführen (Import rückgängig machen)
5. System prüft, ob Rückgängig möglich ist (z.B. nicht, wenn Rechnungen bereits erstellt)
6. Bei Rollback: Werden alle Änderungen zurückgefahren
7. Original-Zustand wird hergestellt und geloggt
8. Audit-Trail dokumentiert Rollback mit Begründung

Betroffene Felder: ImportID, ImportDate, ImportedByID, ImportStatus, ChangeCount, RollbackStatus, RollbackDate, RollbackReason

Auswirkungen:

- Vollständige Nachverfolgbarkeit aller Importe
- Möglichkeit zur Fehlerkorrektur durch Rollback
- Compliance-sicherheit durch Audit-Trail
- Vertrauenswürdigkeit der Import-Funktionalität

Appendix: Dokumentations-Struktur und Erweiterungsanleitung

Dokumentations-Status (Stand: 2025-11-04)

Kapitel	Modul-Anzahl	Dokumentiert	Detailgrad	Status
1. Abrechnung	6	6	100%	✅ Vollständig
2. Administration	12	1	8%	🔄 2.3 detailliert
3. Adressen/CRM	7	0	0%	⏳ Zu bearbeiten
5. Buchhaltung/Finanzen	7	1	14%	🔄 5.4 detailliert
8. Helpdesk	5	1	20%	🔄 8.5 detailliert
Kapitel 4, 6, 7, 9-15	52	0	0%	⏳ Zu bearbeiten
GESAMT	**93**	**12**	**13%**	🎯 Ziel: 100%

Vorlage für neue Module

Modul-Header

```
## X.Y [Modulname] ([English Translation])

**Modulpfad**: `src/centron/Centron.WPF.UI/Modules/[Kategorie]/[Modul]/`
**Controller**: `[ModulnameAppModuleController]`
**ViewModel**: `[HauptViewModelName]`
**Schnittstelle**: `I[Modulname]Logic`
**Kategorie**: [Kategorie-Name]
**Beschreibung**: [Deutsche Beschreibung, was das Modul macht]
**Lizenz**: `LicenseGuids.[Modulname]` OR `LicenseGuids.Centron`
**Rechte**: `UserRightsConst.[Category].[MODULE_NAME]`

### Modul-Architektur
[Kurze Erklärung der Architektur, Workflow, Haupt-Features]

### Vollständige Use Cases
```

Use Case Template

```
#### X.Y.Z [Use Case Name]

**Zweck**: [Deutsche Beschreibung des Geschäftszwecks]

**Ablauf aus Benutzersicht**:
1. Benutzer macht Aktion A
2. System zeigt/verarbeitet B
3. Ergebnis ist C

**Betroffene Felder/Daten**:
- `PropertyName` (Type): Erklärung
- `Feld2` (Type): Erklärung

**Auswirkungen**:
- [Was ändert sich in der DB]
- [UI-Updates]
- [Business-Konsequenzen]
```

Anleitung zum Ergänzen fehlender Module

Schritt 1: Module im Code lokalisieren

```
# Im Source-Verzeichnis Module suchen
find src/centron/Centron.WPF.UI/Modules -name "*ModuleController.cs" | grep -i [suchtext]

# Beispiel: Mahnung-Modul
find src -name "*Dunning*"
# → src/centron/Centron.WPF.UI/Modules/Finances/Dunning/DunningOverviewAppModuleController.cs
```

Schritt 2: ViewModel und Interface recherchieren

- Controller-Datei öffnen
- Nach `ILogic` Interface suchen (z.B. `IDunningLogic`)
- Zugehörige ViewModel identifizieren (z.B. `DunningOverviewViewModel`)

Schritt 3: Use Cases aus Code extrahieren

Im Code nach folgendem suchen:

- Public Commands (z.B. `LoadCommand`, `SaveCommand`, `DeleteCommand`)
- Public Methods im `ILogic` Interface
- Property-Namen die UI-Features andeuten
- UI-Element Bindings im XAML

Beispiel - IDunningLogic Methoden:

```
Task<Result<IList<DunningDTO>>> GetDunningList(int customerId);
Task<Result<bool>> GenerateDunningLetter(int dunningId);
Task<Result<bool>> SendDunningEmail(int dunningId, string email);
// → 3 Use Cases: Mahnliste laden, Mahnbrief generieren, E-Mail versenden
```

Schritt 4: Detailliert dokumentieren

Pro Use Case folgende Info sammeln:

- Was macht der Endbenutzer konkret? (Geschäfts-Perspektive)
- Welche Schritte sind erforderlich?
- Welche Daten/Felder sind beteiligt?
- Was ändert sich als Ergebnis?

Konventionen und Standards

Dokumentations-Konventionen

- **Modulpfad**: Vollständiger Quellcode-Pfad (absolut)
- **Controller**: Hauptcontroller-Klasse
- **ViewModel**: Primäres ViewModel
- **Schnittstelle**: Service Logic Interface (Konvention: `I[Name]Logic`)
- **Code-Referenz**: `[Datei.cs:Zeilennummer]` Format

Lizenz-Typen

- `LicenseGuids.Centron`: Full System License (alle Module)
- `LicenseGuids.[Modulname]`: Individuelle Modul-Lizenz
- Mehrfach möglich: `LicenseGuids.X` OR `LicenseGuids.Y` OR `LicenseGuids.Centron`

Verbindungs-Typen

- **SqlServer**: Direkte DB-Verbindung (Desktop-Modus mit BL-Layer)
- **CentronWebServices**: REST API Connection (Web-Service-Modus mit WSLogic)
- Die meisten Module unterstützen beide automatisch

Sprach- und Namenkonventionen

- **Kapitel-Titel**: Deutsch + English Translation in Klammern
- **Use Case Beschreibung**: Auf Deutsch, Geschäfts-fokussiert
- **Property-Namen**: Original C# Namen (PascalCase, unverändert)
- **Feld-Namen**: Original SQL/Database Namen
- **UI-Text**: Wie tatsächlich im User Interface angezeigt

Prioritäten für weitere Dokumentation

🔴 KRITISCH (Geschäftskritisch, täglich)

- **Kapitel 6**: Controlling/Analytics (8 Module) - KPIs, Dashboards, Berichte
- **Kapitel 5**: Zahlungseingang, OPOS, SEPA (3 Module) - Kernfinanz
- **Kapitel 8**: Ticket-Details, TaskManagement, Checklisten (4 Module) - Support-Kern

🟠 WICHTIG (Häufig genutzt)

- **Kapitel 3**: Adressen/CRM (7 Module) - Kundenstamm
- **Kapitel 7**: Einkauf/EDI (4 Module) - Beschaffung
- **Kapitel 10**: Logistik (4 Module) - Lager/Kommissionierung
- **Kapitel 14**: Stammdaten (8 Module) - Master Data

🟡 OPTIONAL (Spezialisiert)

- **Kapitel 2, 4, 9, 11, 13, 15**: Weitere Module (insgesamt 25)

🔵 DEPRECATED (Nicht aktiv)

- **Kapitel 12**: Passwort Manager - wird nicht mehr empfohlen

FAQ - Häufig gestellte Fragen

Q: Wie detailliert müssen Use Cases sein?
A: Orientieren Sie sich an Kapiteln 1.1, 2.3, 5.4, 8.5 - mindestens 5-10 Use Cases pro Modul mit Benutzer-Workflow und Feldern.

Q: Sollen Code-Snippets eingebunden werden?
A: Nur wenn sie dem Verständnis helfen. Fokus liegt auf Geschäfts-Logik, nicht technische Implementierung.

Q: Wer sollte diese Dokumentation erweitern?
A: Business Analyst, Product Manager, oder Entwickler mit Geschäfts-Verständnis. Nicht nur reine Entwickler.

Q: Wie wird es mit neuen Features aktualisiert?
A: Nach jedem Feature-Release die neuen Use Cases hinzufügen und Versionsnummer erhöhen.

Q: Kann ich die Struktur/Vorlage anpassen?
A: Ja! Die Struktur ist flexibel. Wichtig ist: Konsistenz innerhalb eines Kapitels.

Dokumentations-Kontext

- **Version**: 2025.1.0.0
- **Letzter Update**: 2025-11-04
- **Fortschritt**: 13% dokumentiert (30-40 Stunden geschätzte Restarbeit für 100%)
- **Geplante Fertigstellung**: Q1 2025
- **Kontakt für Fragen**: Product Team, Tech Lead

End of Documentation