HANDLUNGSORIENTIERTES JAVA-PROJEKT FI 37

Prototyp-Entwicklungsprojekt

Exposee

Ein Java-Anwendungssetup mit einer grafischen Benutzeroberfläche (GUI) und einer Datenbank zur Ersetzung der Verwaltung von Mitarbeitern und Kunden mittels Excel-Tabellen, wobei XML-Dateien als Zwischenmedium für den Datentransfer verwendet werden.

Projektbetreuer: Hr. Hebold

Feras Alshaekh Ebraheem
feras.kh.ebraheem@gmail.com

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

1 Einleitung

1.1 Hintergrund

Die ShareEmp Ltd. verwaltet bislang ihre Mitarbeiter, externen Mitarbeiter und Kunden ausschließlich in Excel-Tabellen. Dies führt zu manuellen, fehleranfälligen Prozessen und mangelnder Datenkonsistenz.

1.2 Motivation

Ziel des Projekts ist es, einen Prototypen einer Java-Anwendung zu entwickeln, der die bisher in Excel erfassten Daten in einer relationalen Datenbank verwaltet. XML-Dateien dienen als Zwischenspeicher, um die Daten aus Excel zu importieren.

1.3 Ziele

- Entwicklung einer GUI-basierten Java-Anwendung mit Datenbankanbindung
- Automatisierung der Datenerfassung, -speicherung und -abfrage
- Verbesserung der Datenkonsistenz und Benutzerfreundlichkeit

---

1.4 Zeitrahmen

- Projektbeginn: 17.02.2025
- Projektende: 11.03.2025
- Projektantrag: 28.02.2025, 15:00 Uhr
- Projektdokumentation: 11.03.2025, 15:00 Uhr

---

2 Projektbeschreibung

2.1 Ist-Analyse

- **Aktuelle Situation:**
  - Daten werden in Excel-Tabellen verwaltet, was zu redundanten Einträgen, manuellen Fehlern und zeitaufwändigen Aktualisierungen führt.
  - Es existieren keine automatisierten Prozesse zur Datensicherung oder -analyse.
- **Probleme und Herausforderungen:**
  - Hoher manuelle Aufwand bei der Datenpflege.
  - Fehlende Integration zwischen den verschiedenen Excel-Tabellen (Mitarbeiter, externe Mitarbeiter, Kunden).
  - Mangel an Übersichtlichkeit und Datenkonsistenz.

2.2 Soll-Konzept

- **Lösungsansatz:**
  - Entwicklung einer Java-basierten Anwendung, die eine relationale Datenbank (MySQL) verwendet, um die bisher in Excel erfassten Daten zu zentralisieren.
  - Einsatz von XML-Dateien als Schnittstelle zwischen den alten Excel-Daten und der neuen Datenbank.
- **Funktionale Anforderungen:**
  - **Datenimport:**
    - Import aller Mitarbeiterdaten (Elemente <person> im XML-Dokument) in die Tabelle „Mitarbeiter“ mithilfe der Klasse *DataImporter*.
    - Import der Firmendaten aus Elementen <firma> in die Tabelle „Firmen“.

- Import externer Mitarbeiter aus Elementen <externe_mitarbeiter> in die entsprechende Tabelle.
- Import von Kundendaten (Element <kunde>) in die Tabelle „Kunden“.
- Import von Beziehungen zwischen Mitarbeitern und Kunden (Element <mitarbeiter_kunden>).

- **Datenanzeige und -bearbeitung:**
  - Anzeige der importierten Daten in einer TableView (JavaFX), mit dynamisch erzeugten Spalten.
  - Suchfunktionalitäten in der Tabelle (z. B. Suche nach „Vorname“).
  - Möglichkeit zum Hinzufügen, Ändern und Löschen von Datensätzen.
- **Datenexport:**
  - Export der aktuellen Daten aus der Datenbank in eine XML-Datei.
- **Fehlerbehandlung und Validierung:**
  - Sicherstellung, dass alle Muss-Felder (z. B. Mitarbeitennummer, Vorname, Nachname) vorhanden sind.
  - Verwendung von PreparedStatements und Exception-Handling zur Vermeidung von SQL-Injection und zur Gewährleistung der Datenbankkonsistenz.
- **Benutzerschnittstelle:**
  - Ein intuitives GUI, das den Anwender durch alle Funktionen führt und eine klare Navigation (z. B. Buttons für Import, Export, Suchen, Hinzufügen, Löschen) bietet.

3 Durchführung

3.1 Technische Umsetzung:

- **Schichtenarchitektur:**
  - **Präsentationsschicht:** JavaFX-basierte GUI, die dem Anwender die Interaktion mit dem System ermöglicht (z. B. EmployeeManagementApp).
  - **Logikschicht:** Enthält Klassen wie DataImporter und die Domänenklassen (Mitarbeiter, ExterneMitarbeiter etc.), die die Geschäftslogik implementieren.

  - **Datenzugriffsschicht:** DBHelper und entsprechende SQL-Abfragen zur Interaktion mit der MySQL-Datenbank.

- **Verwendete Technologien:**
  - Java, JavaFX für die Benutzeroberfläche.
  - MySQL als relationale Datenbank.
  - XML (DOM) zur Datenübertragung zwischen Excel und der Datenbank.
- **Diagramme:**
  - UML-Klassendiagramme zur Darstellung der Objektbeziehungen (z. B. Vererbung von Person zu Mitarbeiter, ExterneMitarbeiter und Kunde).
  - ER-Diagramm zur Darstellung des Datenbankschemas.
  - Use-Case-Diagramm zur Visualisierung der Hauptfunktionen des Systems.

3.2 Implementierungsschritte:

- **Phase 1:** Anforderungsanalyse und Erstellung des Datenbankentwurfs.
- **Phase 2:** Entwicklung der Kernklassen:
  - DBHelper für die Datenbankverbindung.
  - DataImporter für den Import von XML-Daten.
  - Domänenklassen (Mitarbeiter, ExterneMitarbeiter, Firma, Kunde, MitarbeiterKunden, Person).
- **Phase 3:** Implementierung der GUI (EmployeeManagementApp):
  - Aufbau der Hauptoberfläche mit TableView, Buttons und weiteren Steuerelementen.
  - Integration von Funktionen zum Laden, Suchen, Hinzufügen, Löschen, Importieren und Exportieren von Daten.
- **Phase 4:** Testen und Fehlerbehebung:
  - Entwicklung und Ausführung von Unit-Tests (JUnit_Tests) für einzelne Module.
  - Durchführung von Integrationstests und Anpassung der Schnittstellen.

3.3 Benutzeroberfläche (GUI):

Layout:

- Eine seitliche Leiste (Sidebar) mit Schaltflächen für „Show Tables“, „XML zu Datenbank“, „Datenbank zu XML“ und „Beenden“.
- Eine zentrale Anzeige (TableView) für die Daten.
- Zusätzliche Funktionen:
    - Suchen: Über ein ComboBox zur Auswahl der Suchspalte und ein Textfeld/Eingabedialog zur Eingabe des Suchbegriffs.
    - Hinzufügen und Löschen von Datensätzen über separate Dialogfenster.

4 Testprotokolle

4.1 Unit-Tests:

4.1.1 DBHelperTest

- **Ziel:** Überprüfung, ob die Methode {@code DBHelper.getConnection()} eine gültige Datenbankverbindung zurückgibt.
- **Testkriterien:**
    - Die zurückgegebene Verbindung darf nicht {@code null} sein.
    - Die Verbindung sollte offen sein (d.h. {@code isClosed()} gibt {@code false} zurück).

4.1.2 ExterneMitarbeiterTest

- **Ziel:** Sicherstellen, dass Objekte der Klasse {@code ExterneMitarbeiter} korrekt validiert werden.
- **Tests:**
    - **Gültiger externer Mitarbeiter:** Setzen von gültigen Werten für {@code ExterneID} und {@code FirmaID} und anschließendes Überprüfen, ob diese Werte korrekt zurückgegeben werden.

- **Ungültige ExterneID:** Versuch, eine negative {@code ExterneID} zu setzen, was eine {@code IllegalArgumentException} mit der Meldung „Die externe ID muss eine positive Zahl sein.“ auslöst.
- **Ungültige FirmaID:** Versuch, {@code FirmaID} mit dem Wert 0 zu setzen, was ebenfalls eine Ausnahme mit der entsprechenden Fehlermeldung auslöst.

4.1.3 FirmaTest

- **Ziel:** Überprüfung der Validierung und der Getter/Setter für Objekte der Klasse {@code Firma}.
- **Tests:**
  - **Gültige Firma:** Setzen von gültigen Werten für {@code FirmaID}, {@code Name} und {@code Adresse} und Überprüfung, ob diese korrekt gespeichert werden.
  - **Ungültige FirmaID:** Setzen einer negativen {@code FirmaID} soll eine Ausnahme mit der Meldung „Die Firmen-ID muss eine positive Zahl sein.“ auslösen.
  - **Ungültiger Firmenname:** Setzen eines leeren Namens soll eine Ausnahme mit der Meldung „Der Firmenname darf nicht leer sein.“ auslösen.
  - **Ungültige Adresse:** Setzen einer leeren Adresse soll eine Ausnahme mit der Meldung „Die Adresse der Firma darf nicht leer sein.“ auslösen.

4.1.4 KundeTest

- **Ziel:** Validierung und korrekte Funktionsweise der Klasse {@code Kunde}.
- **Tests:**
  - **Gültiger Kunde:** Erzeugen eines {@code Kunde}-Objekts, Setzen von gültigen Werten für {@code KundenID} und {@code Branche} und Überprüfung der Rückgabewerte.
  - **Ungültige KundenID:** Setzen einer negativen {@code KundenID} soll eine Ausnahme mit der Meldung „Die Kunden-ID muss eine positive Zahl sein.“ auslösen.

- **Leere Branche:** Setzen eines leeren Strings als Branche soll eine Ausnahme mit der Meldung „Die Branche darf nicht leer sein.“ auslösen.
- **Null als Branche:** Setzen von {@code null} für die Branche soll ebenfalls eine Ausnahme mit der gleichen Fehlermeldung auslösen.

4.1.5 MitarbeiterKundeTest

- **Ziel:** Überprüfung der Beziehungen zwischen Mitarbeitern und Kunden.
- **Tests:**
  - **Gültiger Datensatz:** Erzeugen eines {@code MitarbeiterKunde}-Objekts mit gültigen Werten für {@code MitarbeiterID} und {@code KundenID} und Überprüfung, ob die Werte korrekt gespeichert werden.
  - **Ungültige MitarbeiterID:** Setzen einer negativen {@code MitarbeiterID} soll eine Ausnahme mit der Meldung „Die Mitarbeiter-ID muss eine positive Zahl sein.“ auslösen.
  - **Ungültige KundenID:** Setzen eines ungültigen Wertes (z. B. 0) für {@code KundenID} soll eine Ausnahme mit der Meldung „Die Kunden-ID muss eine positive Zahl sein.“ auslösen.

4.1.6 MitarbeiterTest

- **Ziel:** Sicherstellen, dass die Setter und Getter der Klasse {@code Mitarbeiter} korrekt funktionieren.
- **Tests:**
  - **Gültiger Mitarbeiter:** Erzeugen eines {@code Mitarbeiter}-Objekts mit einer gültigen ID und Überprüfung, ob die ID korrekt zurückgegeben wird.
  - **Ungültiger Mitarbeiter:** Setzen einer negativen ID soll eine Ausnahme mit der Meldung „Die Mitarbeiter-ID muss eine positive Zahl sein.“ auslösen.

4.1.7 PersonTest

- **Ziel:** Testen der grundlegenden Getter und Setter in der abstrakten Klasse {@code Person} (durch Erzeugen eines anonymen Unterobjekts).
- **Tests:**
    - Überprüfung der Setter und Getter für alle relevanten Attribute (Vorname, Nachname, Straße, PLZ, Ort, Telefon, E-Mail).
    - Ein umfassender Test, der alle Werte setzt und anschließend überprüft, ob sie korrekt abgerufen werden.

4.2 Manueller Test

4.2.1 Anmeldung bei MySQL über PowerShell

1. **PowerShell öffnen und MySQL anmelden:**
    - Öffnen Sie PowerShell.
    - Geben Sie ein : PS C:\Windows\system32> mysql -u root -p
    - Wenn Sie zur Eingabe des Passworts aufgefordert werden, geben Sie ein: 123456789feras
2. **Überprüfen der Datenbank:**
    - In der MySQL-Konsole:
        - mysql> show databases;
        - mysql> use shareempdb;
        - mysql> show tables;

      Hinweis: Falls noch keine Tabellen vorhanden sind, wird angezeigt:
      Empty set (0.00 sec)

4.2.2 Ausführen der Anwendung in IntelliJ

1. **Starten der EmployeeManagementApp:**
    - Öffnen Sie IntelliJ und führen Sie die Klasse feras.EmployeeManagementApp aus.
2. **Funktionen in der Anwendung:**
    - **Show Tables:**
        - Drücken Sie die Taste „Show Tables“.
        - Erwartetes Ergebnis: Fehlermeldung – *"Es gibt keine Tabellen in der Datenbank!"*
    - **XML to Datenbank:**
        - Drücken Sie die Taste „XML to Datenbank“.
        - Wählen Sie im Projektordner die Datei shareempdb.xml aus und klicken Sie auf „Open“.
        - Es erscheint eine Nachricht:
          "Die Tabellen wurden erfolgreich erstellt!" – klicken Sie auf OK.
        - Anschließend folgt eine weitere Nachricht:
          "Alle Daten wurden erfolgreich importiert!" – klicken Sie ebenfalls auf OK.
    - **Show Tables (erneut):**
        - Drücken Sie erneut die Taste „Show Tables“.
        - Nun öffnet sich ein Fenster mit dem Titel „Verfügbare Tabellen“, das alle Tabellennamen (als Schaltflächen) anzeigt, die aus der Datei shareempdb.xml in die Datenbank importiert wurden.
    - **Tabelle auswählen:**
        - Klicken Sie auf eine der angezeigten Schaltflächen, z. B. auf die Schaltfläche für die Tabelle „Mitarbeiter“.
        - Daraufhin werden die Inhalte der gewählten Tabelle in der Anwendung angezeigt.

4.2.3 Überprüfung in MySQL

1. **Tabelleninhalte prüfen:**

- Kehren Sie zu MySQL in PowerShell zurück:
  mysql> show tables;

Jetzt sollten die importierten Tabellen sichtbar sein.

2. **Daten abrufen:**

- Führen Sie Abfragen aus, um den Inhalt der Tabellen zu überprüfen:

  - mysql> select * from mitarbeiter;
  - mysql> select * from externe_mitarbeiter;
  - mysql> select * from kunden;
  - mysql> select * from firmen;

  Alle Abfragen sollten die entsprechenden Daten anzeigen.

4.2.4 Testen der GUI-Funktionalitäten in der EmployeeManagementApp

1. **Löschen von Datensätzen:**

   - Wählen Sie in der Tabelle „Mitarbeiter“ eine Zeile mit der Maus aus.

   - Drücken Sie die Taste „Löschen“.

     - Es erscheint eine Bestätigungsmeldung:
       "Sind Sie sicher? Möchten Sie die ausgewählte Zeile wirklich löschen?"

     - Bestätigen Sie mit OK.

     - Es erscheint anschließend eine Erfolgsmeldung:
       "Datensatz erfolgreich gelöscht!"

   - Um zu überprüfen, ob der Datensatz auch aus der Datenbank entfernt wurde, klicken Sie erneut auf die Schaltfläche „Mitarbeiter“ oder führen Sie in MySQL die Abfrage aus: mysql> SELECT * FROM mitarbeiter;

2. **Hinzufügen von Datensätzen:**

- Klicken Sie in der Anwendung auf die Schaltfläche „Add“ (nachdem Sie z. B. die Tabelle „Mitarbeiter“ ausgewählt haben).

  - Ein neues Fenster mit dem Titel *"Neuen Datensatz hinzufügen zu mitarbeiter"* erscheint.

  - In diesem Fenster sind die Spaltennamen der Tabelle aufgeführt; geben Sie neben jedem Feld die gewünschten Daten ein.

  - Klicken Sie auf „Save“.

- Es erscheint eine Erfolgsmeldung:
  "Der Datensatz wurde erfolgreich hinzugefügt!"

- Bestätigen Sie mit OK.

- Überprüfen Sie anschließend in der Anwendung oder in MySQL (z. B. mit SELECT * FROM mitarbeiter;), ob der neue Datensatz vorhanden ist.

- **Hinweis zu Fehlerfällen:**

  - Bei Eingabe einer bereits existierenden ID (z. B. 1) erscheint eine Fehlermeldung:
    "SQL Error: Duplicate entry '1' for key 'mitarbeiter.PRIMARY'"

  - Wird auf „Save“ geklickt, ohne dass alle erforderlichen Daten eingegeben wurden (z. B. leere ID), erscheint eine Fehlermeldung:
    "SQL Error: Incorrect integer value: '' for column 'MitarbeiterID' at row 1"

3. **Suchen nach Datensätzen:**

- Neben der Schaltfläche „Add“ befindet sich ein Dropdown-Menü „Suchen nach“, dessen Inhalt sich automatisch an die Spaltennamen der aktuell ausgewählten Tabelle anpasst.

- Wählen Sie z. B. die Spalte „Ort“ aus, und klicken Sie auf die Schaltfläche „Suche“.

  - Es erscheint ein Eingabedialog mit dem Titel *"Suche in Ort"*, in dem Sie z. B. „Berlin“ eingeben können (die Suche ist nicht case-sensitive).

  - Nach Bestätigung werden alle Zeilen angezeigt, in denen die Spalte „Ort“ den Wert „Berlin“ enthält.

- **Fehlerszenarien:**

  - Wird auf „Suche“ geklickt, ohne eine Suchspalte auszuwählen, erscheint die Meldung:
    "Bitte wählen Sie eine Suchspalte aus!"

  - Wird ein nicht vorhandener Wert eingegeben, erscheint im Tabellenbereich die Meldung:
    "No content in table."

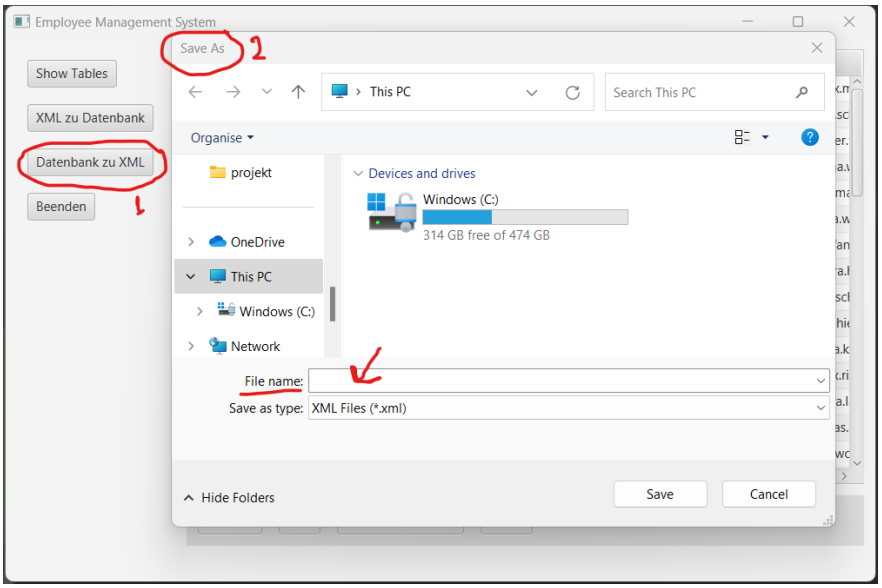
4.2.5 Exportieren der Daten von der Datenbank in eine XML-Datei

1. **Datenbank zu XML exportieren:**

   - Drücken Sie in der Anwendung die Schaltfläche „Datenbank zu XML“.

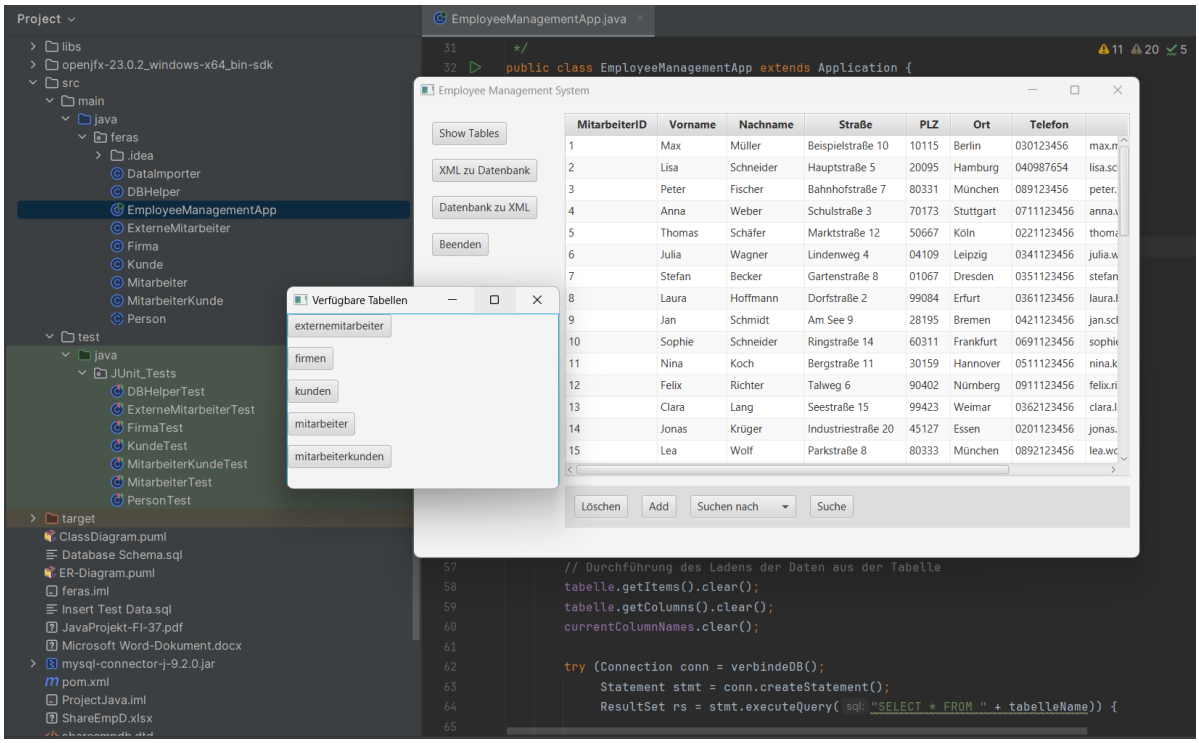
Testprotokolle

- Es öffnet sich ein Fenster zur Auswahl des Speicherorts. Wählen Sie einen Ordner aus, geben Sie einen Dateinamen ein und klicken Sie auf „Save“.
- Es erscheint eine Erfolgsmeldung:
  "XML-Export erfolgreich!"
- Bestätigen Sie mit OK.
- Nun wurde eine XML-Datei erstellt, die alle Daten der Datenbank shareempdb enthält.



4.2.6 Bild des Programms

Ein Bild, das das Erscheinungsbild des Programms nach dem Start, dem Import der Daten und dem Öffnen des Fensters "Show Tables" zeigt.



5 Diagramme

A1 ER-Diagramm

Ein **ER-Diagramm** mit **PlantUML**, das die grundlegende Struktur einer Datenbank zur Verwaltung von Informationen über Mitarbeiter, Kunden und Unternehmen zeigt.

1. **Entitäten (Entities):**

   - **Mitarbeiter:**
     Enthält als Primärschlüssel die MitarbeiterID (vom Typ int) sowie Attribute wie Vorname, Nachname, Straße, PLZ, Ort, Telefon und Email.

   - **Externe Mitarbeiter:**
     Besitzt die ExterneID als Primärschlüssel und ähnliche Attribute wie Mitarbeiter. Zusätzlich enthält diese Entität ein Attribut FirmaID, das auf die zugehörige Firma verweist.

   - **Firmen:**
     Diese Entität hat die FirmaID als Primärschlüssel und speichert den Firmennamen sowie die Adresse.

Diagramme

- **Kunden:**
  Enthält die KundenID als Primärschlüssel und weitere Attribute wie Vorname, Nachname, Straße, PLZ, Ort, Telefon, E-Mail und Branche.

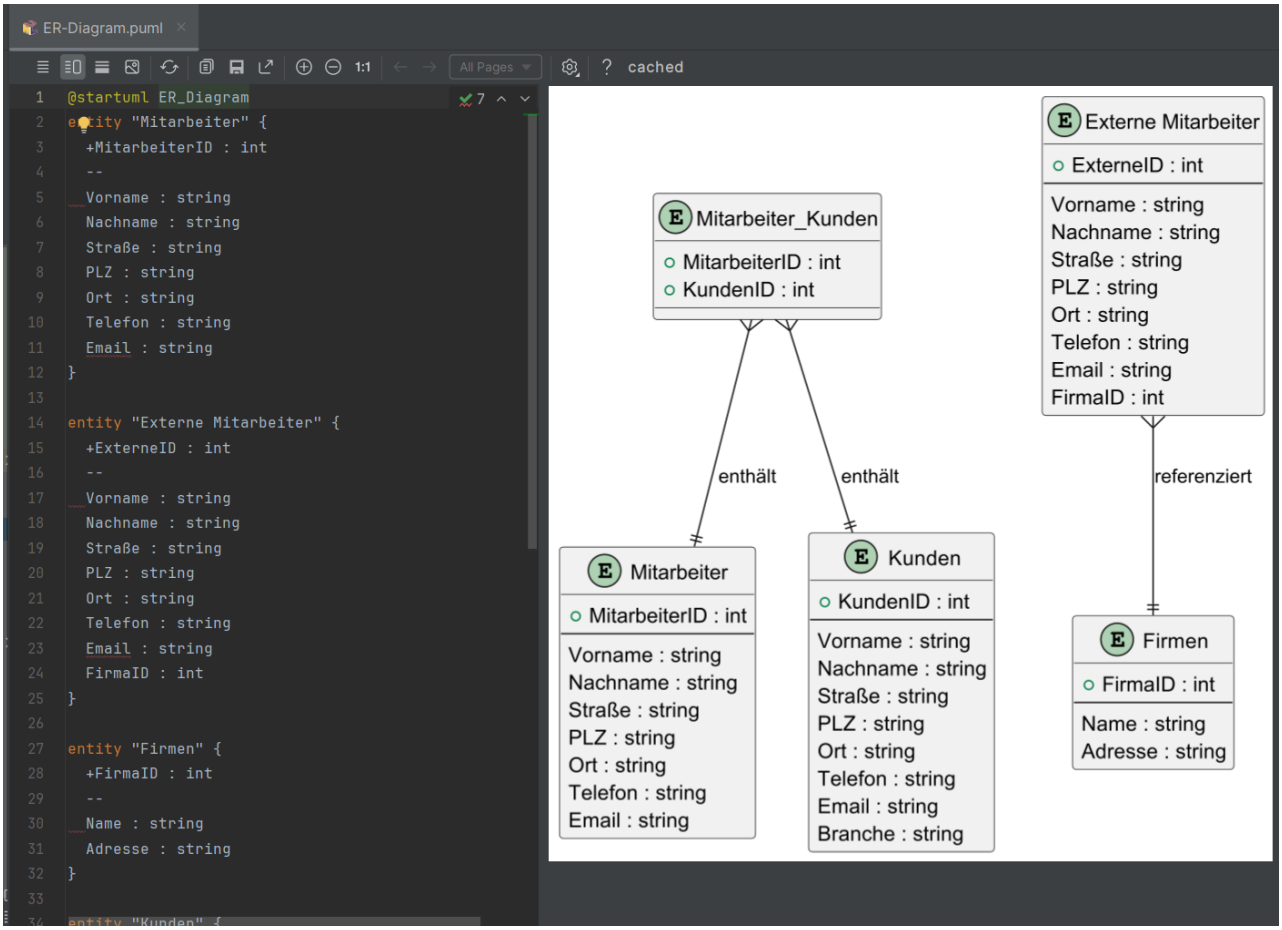
- **Mitarbeiter_Kunden:**
  Dient als Beziehungstabelle, die die MitarbeiterID und die KundenID speichert, um eine Viele-zu-Viele-Beziehung zwischen Mitarbeitern und Kunden herzustellen.

2. **Beziehungen (Relationships):**

- **Mitarbeiter_Kunden – Mitarbeiter:**
  Die Beziehung zeigt an, dass ein Datensatz in der Tabelle Mitarbeiter_Kunden einen bestimmten Mitarbeiter enthält.

- **Mitarbeiter_Kunden – Kunden:**
  Ebenso wird hier der zugehörige Kunde referenziert.

- **Externe Mitarbeiter – Firmen:**
  Diese Beziehung drückt aus, dass ein externer Mitarbeiter eine Firma referenziert, zu der er gehört.



A2 UML-Klassendiagramme

Ein **Diagramm** mit **PlantUML**, das die Architektur der Anwendung und die allgemeine Struktur der verschiedenen Komponenten des Mitarbeiterverwaltungssystems zeigt. Im Folgenden findet sich eine Zusammenfassung, die die Komponenten des Diagramms und ihre Beziehungen erklärt.

5.1.1 1. Paket "Anwendung":

- **EmployeeManagementApp:**
    - Repräsentiert die Hauptanwendung für das Employee Management System.
    - Enthält UI-Elemente wie eine TableView (zur Anzeige von Datensätzen), ein VBox-Container (für Buttons), eine ComboBox (für die Suche) und verschiedene Buttons (z. B. für Suche, Import, Export).
    - Bietet Methoden zur:
    - Datenbankverbindung (verbindeDB),
    - Datensatzzustand laden (ladeDaten),
    - Suchen in der Tabelle (searchInTable),
    - Importieren von XML-Daten in die Datenbank (importiereDatenAusXML),
    - Anzeigen der vorhandenen Tabellennamen (ladeTabellenNamen),
    - Exportieren der Daten in eine XML-Datei (exportiereDatenNachXML und exportiereTabelle),
    - sowie zur Anzeige von Fehlermeldungen und Erfolgsmeldungen (zeigeFehler und zeigeErfolg).
    - Die Methoden **start** und **main** dienen dem Start der Anwendung.

5.1.2 2. Paket "Hilfsklassen":

- **DBHelper:**
    - Eine Hilfsklasse, die eine statische Methode **getConnection()** bereitstellt, um eine Verbindung zur Datenbank herzustellen.
- **DataImporter:**

- Eine Klasse, die statische Methoden zum Importieren von Daten aus XML-Dateien in die Datenbank bietet.
- Enthält Methoden wie:
  - **importFirmen** (Import der Firmendaten),
  - **importMitarbeiter** (Import der Mitarbeiterdaten),
  - **importExterneMitarbeiter** (Import der Daten externer Mitarbeiter),
  - **importKunden** (Import der Kundendaten),
  - **importMitarbeiterKunden** (Import der Beziehungen zwischen Mitarbeitern und Kunden).

5.1.3 3. Paket "Datenmodell":

- **Person (abstrakte Klasse):**
  - Definiert gemeinsame Attribute für alle Personen, wie Vorname, Nachname, Straße, PLZ, Ort, Telefon und E-Mail.
- **Mitarbeiter:**
  - Erbt von **Person** und besitzt zusätzlich die MitarbeiterID als Primärschlüssel.
- **ExterneMitarbeiter:**
  - Erbt von **Person** und verfügt über die ExterneID als Primärschlüssel sowie über das Attribut FirmaID, das die Zugehörigkeit zu einer Firma anzeigt.
- **Kunde:**
  - Erbt von **Person** und enthält als zusätzlichen Identifikator die KundenID sowie das Attribut Branche.
- **Firma:**
  - Enthält die FirmaID als Primärschlüssel, den Firmennamen (Name) und die Adresse.
- **MitarbeiterKunde:**
  - Eine Beziehungsklasse, die die Verbindung zwischen Mitarbeitern und Kunden abbildet, indem sie sowohl die MitarbeiterID als auch die KundenID speichert.

5.1.4 4. Beziehungen zwischen den Entitäten:

- **ExterneMitarbeiter → Firma:**
    - Die ExterneMitarbeiter-Entität referenziert die Firma über das Attribut FirmaID.
- **MitarbeiterKunde → Mitarbeiter und Kunde:**
    - Die MitarbeiterKunde-Entität enthält die MitarbeiterID und KundenID, um eine Viele-zu-Viele-Beziehung zwischen Mitarbeitern und Kunden herzustellen.

A3 Use-Case-Diagramm

Dieses **PlantUML-Diagramm** ist ein **Use-Case-Diagramm**, das die Hauptfunktionen des Employee Management Systems aus der Sicht des Benutzers darstellt. Im Einzelnen:

- **Akteur "Benutzer":**
  Der Benutzer interagiert mit dem System.
- **Anwendungsfälle innerhalb des Systems:**
    - **Tabellen anzeigen:** Der Benutzer kann sich die verfügbaren Tabellen im System anzeigen lassen.
    - **XML zu Datenbank importieren:** Der Benutzer importiert Daten aus einer XML-Datei in die Datenbank.
    - **Datenbank zu XML exportieren:** Der Benutzer exportiert Daten aus der Datenbank in eine XML-Datei.

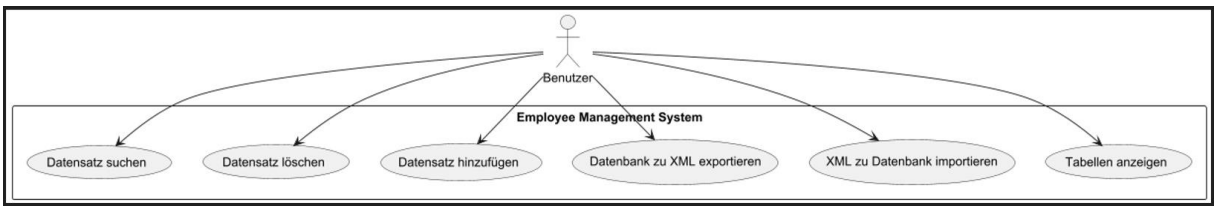
Diagramme

- **Datensatz hinzufügen:** Der Benutzer kann neue Datensätze in das System einfügen.
- **Datensatz löschen:** Der Benutzer kann bestehende Datensätze aus dem System entfernen.
- **Datensatz suchen:** Der Benutzer hat die Möglichkeit, nach bestimmten Datensätzen zu suchen.

```
@startuml
actor Benutzer

rectangle "Employee Management System" {
  usecase "Tabellen anzeigen" as UC1
  usecase "XML zu Datenbank importieren" as UC2
  usecase "Datenbank zu XML exportieren" as UC3
  usecase "Datensatz hinzufügen" as UC4
  usecase "Datensatz löschen" as UC5
  usecase "Datensatz suchen" as UC6
}

Benutzer --> UC1
Benutzer --> UC2
Benutzer --> UC3
Benutzer --> UC4
Benutzer --> UC5
Benutzer --> UC6
@enduml
```



A4 JavaDoc

Im Ordner "javadoc" des Projektverzeichnisses befindet sich eine Datei namens "index.html". Diese Datei öffnet eine Webseite, über die die Dokumentation angezeigt wird.

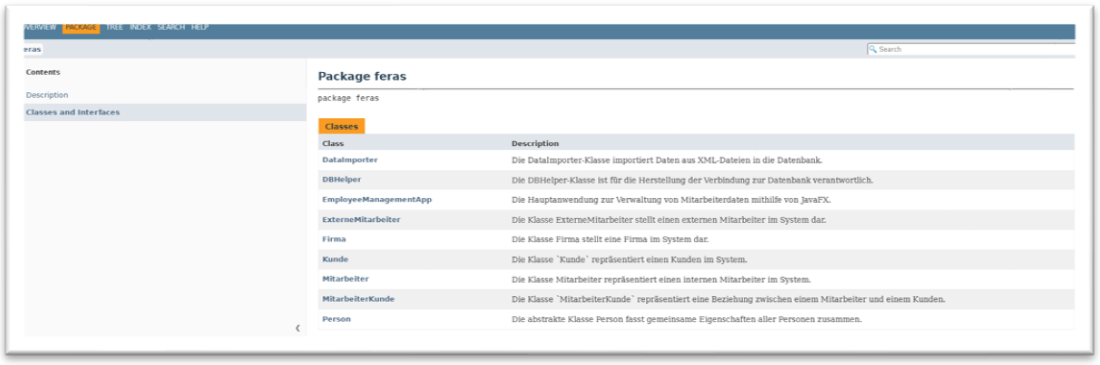
1. **Klassenübersicht:**
   - Zeigt eine Liste aller Klassen, Schnittstellen und Pakete im Projekt.
   - Enthält eine kurze Beschreibung jeder Klasse basierend auf der /** ... */-Dokumentation.
2. **Detailansicht für Klassen:**

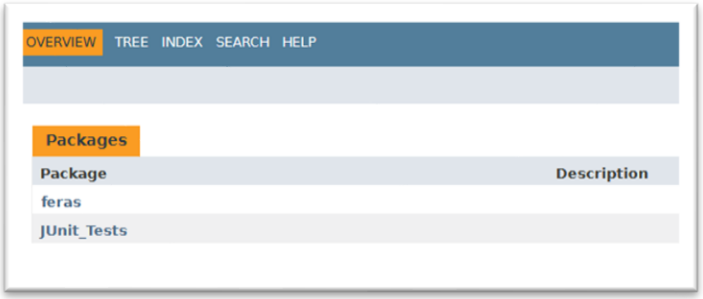
Diagramme

- Enthält den vollständigen Klassennamen und die Beschreibung.
- Zeigt Vererbungshierarchien und implementierte Schnittstellen an.
- Listet alle Konstruktoren, Methoden und Felder mit ihrer Dokumentation.

3. **Methodenübersicht:**
   - Zeigt eine Liste aller Methoden in einer Klasse mit Beschreibung, Parameter- und Rückgabewerten.
   - Falls vorhanden, werden auch @throws-Anmerkungen für Fehlerbehandlungen angezeigt.
4. **Feldübersicht:**
   - Zeigt alle Klassenvariablen (Felder) und ihre Beschreibungen an.
5. **Konstruktorübersicht:**
   - Falls eine Klasse Konstruktoren enthält, werden sie mit den dazugehörigen Parametern und Beschreibungen angezeigt.
6. **Verlinkung zwischen Dokumentationen:**
   - Falls im Code @see, @param, oder @return verwendet wurde, erstellt Javadoc Verweise innerhalb der Dokumentation.
   - @author und @version erscheinen in einer separaten Sektion.
7. **Zusätzliche HTML-Formatierung:**
   - Falls Markdown-ähnliche Formatierungen oder HTML-Tags (<b>, <i>, <ul>, <li>, usw.) in den Javadoc-Kommentaren verwendet wurden, werden diese in der generierten HTML-Dokumentation korrekt dargestellt.

Diagramme


Package feras
package feras
Classes
Class
Description
DataImporter
Die DataImporter-Klasse importiert Daten aus XML-Dateien in die Datenbank.
DBHelper
Die DBHelper-Klasse ist für die Herstellung der Verbindung zur Datenbank verantwortlich.
EmployeeManagementApp
Die Hauptanwendung zur Verwaltung von Mitarbeiterdaten mithilfe von JavaFX.
ExterneMitarbeiter
Die Klasse ExterneMitarbeiter stellt einen externen Mitarbeiter im System dar.
Firma
Die Klasse Firma stellt eine Firma im System dar.
Kunde
Die Klasse `Kunde` repräsentiert einen Kunden im System.
Mitarbeiter
Die Klasse Mitarbeiter repräsentiert einen internen Mitarbeiter im System.
MitarbeiterKunde
Die Klasse `MitarbeiterKunde` repräsentiert eine Beziehung zwischen einem Mitarbeiter und einem Kunden.
Person
Die abstrakte Klasse Person fasst gemeinsame Eigenschaften aller Personen zusammen.


OVERVIEW
TREE
INDEX
SEARCH
HELP
Packages
Package
Description
feras
JUnit_Tests

6 Fazit und Ausblick

6.1 Fazit

- **Erreichte Ziele:**
  • Entwicklung eines funktionalen Prototyps zur Verwaltung von Mitarbeiter- und Kundendaten.
  • Erstellung einer benutzerfreundlichen, interaktiven Oberfläche, die für alle Arten von Nutzern geeignet ist.
  • Möglichkeit, Daten aus XML-Dateien mit nur einem Klick in die Datenbank zu importieren, mit einer intuitiven Oberfläche zur Auswahl der Datei von jedem Speicherort auf dem Computer.
  • Möglichkeit, Daten aus der Datenbank in eine XML-Datei zu exportieren, mit der Option, die Datei an einem beliebigen Ort auf dem Computer zu speichern, um die Daten nach Änderungen an der Datenbank zu sichern.
  • Eine einfache Benutzeroberfläche zur Verwaltung der in der Datenbank gespeicherten Daten, mit Schaltflächen zum Löschen, Suchen oder Hinzufügen von Datensätzen.

6.2 Kritische Reflexion

- Der Prototyp demonstriert eine klare Trennung in Module, wobei die Anwendung, Hilfsklassen und das Datenmodell sauber voneinander getrennt sind.
- Die Implementierung der Datenbankanbindung über DBHelper und die Verwendung von PreparedStatements sorgt für grundlegende Sicherheit, allerdings fehlt ein umfassendes Transaktionsmanagement.
- Die XML-Import- und Exportfunktionen funktionieren, können jedoch bei unvorhergesehenen XML-Strukturen oder fehlerhaften Daten zu Problemen führen.
- Die Verwendung von JavaFX für die GUI liefert eine funktionale Oberfläche, die jedoch in Bezug auf Benutzerfreundlichkeit und modernes Design noch verbessert werden könnte.
- Fehler- und Erfolgsmeldungen werden über Alerts angezeigt, was in einer Testumgebung hilfreich ist, aber in einer produktiven Anwendung weiter verfeinert werden sollte.
- Die Domänenklassen (Mitarbeiter, ExterneMitarbeiter, Firma, Kunde, MitarbeiterKunde, Person) nutzen Vererbung sinnvoll, was die Wiederverwendbarkeit und Wartbarkeit des Codes unterstützt.
- Es existiert eine solide Basis an Unit-Tests, die kritische Funktionen und Validierungen der Domänenklassen überprüfen.
- Das Exception-Handling deckt typische Fehlerfälle ab, könnte aber detailliertere Rückmeldungen für den Endnutzer bieten.
- Insgesamt erfüllt der Prototyp die Kernanforderungen, weist jedoch Verbesserungspotenzial in den Bereichen GUI-Design, Robustheit und Erweiterbarkeit auf.
- Kritisch zu betrachten ist auch, dass bei zunehmender Datenmenge und Nutzerzahl weitere Optimierungen notwendig sein könnten.

6.3 Ausblick

- Zukünftig sollte die Benutzeroberfläche durch den Einsatz moderner UI-Frameworks und responsivem Design weiter optimiert werden.
- Eine Erweiterung des Transaktionsmanagements und der Fehlerprotokollierung ist notwendig, um eine höhere Datenintegrität und Stabilität im Mehrbenutzerbetrieb zu gewährleisten.
- Erweiterte Such- und Filterfunktionen könnten die Benutzerinteraktion verbessern und flexiblere Abfragen ermöglichen.
- Der Import-/Export-Prozess sollte robuster gestaltet werden, um auch verschiedene XML-Formate oder fehlerhafte Datenstrukturen zu handhaben.
- Die Integration weiterer Datenformate (z. B. CSV) könnte den Datenaustausch mit bestehenden Systemen vereinfachen.
- Eine Implementierung von Benutzer-Authentifizierung und Rollenmanagement wäre sinnvoll, um den Zugriff auf sensible Daten zu steuern.
- Weitere Integrationstests und automatisierte Testverfahren sollten entwickelt werden, um die Systemstabilität auch in komplexen Szenarien sicherzustellen.
- Performance-Optimierungen und Skalierungskonzepte sollten in Betracht gezogen werden, um den Prototyp für den produktiven Einsatz vorzubereiten.
- Eine ausführlichere Entwicklerdokumentation (JavaDoc) und eine kontinuierliche Code-Refaktorierung würden die Wartbarkeit langfristig verbessern.
- Insgesamt bietet der Prototyp eine solide Basis, die durch gezielte Weiterentwicklungen zu einem voll funktionsfähigen System ausgebaut werden kann.

7 Literaturverzeichnis

- **Oracle Java SE Documentation**
  Oracle Corporation. „Java SE Documentation“. Abgerufen von: https://docs.oracle.com/en/java/javase/
- **OpenJFX Documentation**
  OpenJFX. „OpenJFX Documentation“. Abgerufen von: https://openjfx.io/
- **MySQL 8.0 Reference Manual**
  MySQL Documentation. „MySQL 8.0 Reference Manual“. Abgerufen von: https://dev.mysql.com/doc/refman/8.0/en/
- **Jakarta XML Binding (JAXB) Reference Implementation**
  Eclipse Foundation. „Jakarta XML Binding (JAXB) RI“. Abgerufen von: https://eclipse-ee4j.github.io/jaxb-ri/
- **PlantUML Documentation**
  PlantUML. „PlantUML Documentation“. Abgerufen von: https://plantuml.com/
- **JUnit 5 User Guide**
  JUnit. „JUnit 5 User Guide“. Abgerufen von: https://junit.org/junit5/docs/current/user-guide/
- **Apache Maven Documentation**
  Apache Software Foundation. „Maven: The Complete Reference“. Abgerufen von: https://maven.apache.org/guides/index.html
- **IHK Leitfaden zur Projektdokumentation**
  Industrie- und Handelskammer (IHK).