

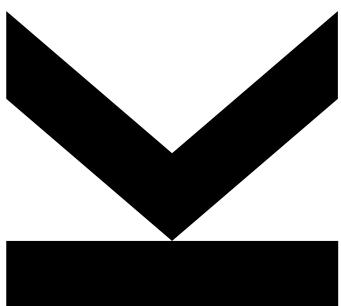
Eingereicht von
Dino Osmanagic
Daniel Breimaier
Thomas Pausch
Lukas Gruber
Daniel Hochgatterer

Angefertigt am 5.01.2022



GRUPPENNR: 04

KursNr: 258.321



Projektdokumentation
im Rahmen des Kurses
PR Data & Knowledge Engineering
im Bachelorstudium
Wirtschaftsinformatik

INHALTSVERZEICHNIS

1.	Projektanforderungen	4
1.1.	Ausgangssituation	4
1.2.	Zielsetzung	4
1.3.	Zuständigkeiten	4
1.4.	Projektstrukturplan.....	5
2.	Technische Dokumentation	10
2.1.	Anforderungen.....	10
2.1.1.	Frontend.....	10
2.1.2.	Webservice Rest Service.....	10
2.1.3.	Datenbank	10
2.1.4.	Datenmodell	10
2.1.5.	Zielsysteme	11
3.	Technische Lösungsarchitektur Frontend	12
3.1.	BPMN Diagramm Frontend	12
3.2.	Technische Umsetzung	12
4.	Technische Lösungsarchitektur REST-Service	24
4.1.	BPMN Diagramm Rest-Service	24
4.2.	Projektstruktur/ Technische Umsetzung	25
4.3.	Fortschritt, Planung	27
4.4.	Einteilung im Rest-Team	27
4.5.	Filtermethoden für Entity Klassen	28
4.6.	Exception Handling.....	28
4.7.	Anmerkungen zur Implementierung.....	29
4.7.1.	Verwendung der Werte 1 und 2 für True respektive False	29
4.8.	Eingabeüberprüfungen im Service.....	30
4.8.1.	Employee Tabelle/Entity:	30
4.8.2.	Targetconfig Tabelle/Entity:	30
4.8.3.	Targetstate Tabelle/Entity:	31
4.9.	Automatisch durch den Service vergebene Werte	31
4.9.1.	Employee Entity/Tabelle	32
4.9.2.	Targetconfig Entity/Tabelle	32
4.9.3.	Targetstate Entity/Tabelle	32
4.9.4.	Department Entity/Tabelle	32

4.10. Nicht auffangbare Falscheingaben.....	33
4.10.1. Eingabe nicht vorhandener Fremdschlüssel	33
4.10.2. Bearbeitung des Passwortes im UI	33
5. Technische Lösungsarchitektur Datenbank & Zielsysteme	34
5.1. Datenbank.....	34
5.1.1. Rahmenbedingungen der Synchronisierung der Zielsysteme:	34
5.1.2. Tabelle „employee“	35
5.1.3. Tabelle „department“	36
5.1.4. Tabelle „targetconfig“	36
5.1.5. Tabelle „targetstate“	37
5.2. Zielsysteme	37
5.2.1 CSV und JSON.....	37
5.2.1. Events	39
5.2.2. LDAP	40
5.2.3. MYSQL.....	40
5.2.4. Synchronisierung starten	40
5.2.5. Funktionsprüfung/Testdaten	41
6. Anmerkungen zu den eingesetzten Software Artefakten, Technologien & Tools.....	41
6.1. GitHub	41
6.1.1. Frontend-Team.....	41
6.1.2. Rest-Service-Team.....	41
6.2. Angular	42
6.3. Entwicklungsumgebungen.....	42
6.4. Spring Initializer Rest Team.....	42
6.5. Rest Service	42
6.6. Hibernate/JPA Rest Team	42
6.7. Postman Rest Team.....	42
6.8. MySQL-Datenbank	43
6.9. Zielsysteme	43
6.9.1. CSV und JSON	43
6.9.2. OPENLDAP	43
6.9.3. MYSQL.....	43
7. Installations- & Konfigurationsanleitung	44
7.1. Datenbank und Zielsysteme	44
7.2. Rest Service	51
7.3. Front End	54

8. Use Cases.....	57
8.1. Use Cases 1: Anlage eines neuen Mitarbeiters.	57
8.2. Use Case 2: Abteilungswechsel eines Mitarbeiters:.....	59
8.3. Use Case 3: Austritt eines Mitarbeiters:.....	61
9. Reflexion Projekt.....	63

1. Projektanforderungen

1.1. Ausgangssituation

In einem fiktiven Konzern werden für natürliche Personen Benutzerkonten in diversen IT-Systemen angelegt und verwaltet. Bei Ein- und Austritten (oder Organisationswechseln) entsteht manueller Aufwand bei der Anlage und Pflege der Benutzerkonten.

1.2. Zielsetzung

Es soll eine Applikation umgesetzt werden, die die Anlage, Änderung und Löschung von Benutzerkonten in angebundenen Zielsystemen automatisiert. Zur Interaktion mit der Anwendung soll ein einfaches Frontend erstellt werden, das mindestens folgende Funktionen anbietet:

- Neuanlage, Bearbeiten, Löschen von Personen
- Neuanlage, Bearbeiten, Löschen von Organisationseinheiten
- Umzug von Personen in andere Organisationseinheiten

Es sollen (mandantenspezifische) Zielsysteme (z.B. CSV-Dateien, SQL-Datenbanken, LDAP-Stores) konfiguriert werden können, die automatisiert mit Änderungen an Userdaten versorgt werden. Das Hinzufügen von Mandanten und Zielsystemen soll konfiguratorisch, ohne Anpassung von Programmcode, möglich sein.

Aus Security-Gründen erhält das Frontend keinen direkten Zugriff auf die Datenbank mit Identitätsdaten, sondern nur auf eine „Pufferdatenbank“, in die aus dem Frontend generierte Requests (Neuanlagen, ...) eingefügt werden können.

1.3. Zuständigkeiten

- Frontend: Dino Osmanagic
- REST Service: Daniel Hochgatterer, Lukas Gruber,
- Datenbank & Zielsysteme: Daniel Breimaier, Thomas Plausch

1.4. Projektstrukturplan

Bereich	Arbeitspaket	Verantwortlicher
Projekt	Aufbau des ursprünglichen GIT-Repository	Dino Osmanagic
Projekt	Aufbereitung von Präsentationen	Gesamtes Team
Projekt	Aufbereitung der Dokumentation	Gesamtes Team
Projekt	Erstellung Geschäftsprozess-Visualisierung	Dino Osmanagic
Projekt	Erstellung Architektur-Visualisierung	Dino Osmanagic
Projekt	Erstellung Installationsanleitung Front End	Dino Osmanagic
Datenbank	Datenbanksystemwahl	Thomas Pausch, Daniel Breimaier
Datenbank	Datenbankschema	Thomas Pausch, Daniel Breimaier
Datenbank	Visualisierung Datenbank	Daniel Breimaier
Datenbank	Datenbank für Zusammenarbeit bereitstellen	Thomas Pausch
Datenbank	Erstellung und Import der Beispieldatensätze	Daniel Breimaier
Datenbank	Diverse Prozeduren zur einfachen Verwaltung erstellen	Daniel Breimaier
Datenbank	Dokumentieren von Rahmenbedingungen	Daniel Breimaier, Thomas Pausch
Datenbank	Erstellen einer Installationsanleitung der Systeme	Daniel Breimaier, Thomas Pausch
Datenbank	Erstellen einer Nutzeranleitung	Daniel Breimaier, Thomas Pausch
Datenbank	Erstellung der Prozeduren für den Import der Testdatensätze	Daniel Breimaier
Datenbank	Testen der Funktionalität (Insert-, Update-, Delete-statements) in der Datenbank	Daniel Breimaier
Zielsysteme	Auswahl von Technologien für CSV Zielsystem	Daniel Breimaier
Zielsysteme	Auswahl von Technologien für JSON Zielsystem	Daniel Breimaier
Zielsysteme	Auswahl von Technologien für LDAP Zielsystem	Thomas Pausch
Zielsysteme	Auswahl von Technologien für MYSQL Zielsystem	Thomas Pausch
Zielsysteme	Implementierung einer automatisierten und dynamischen Synchronisierung eines CSV Formats	Daniel Breimaier
Zielsysteme	Implementierung einer automatisierten und dynamischen Synchronisierung eines JSON Formats	Daniel Breimaier
Zielsysteme	Testen der Funktionsfähigkeit und Korrektheit der Zielsysteme	Daniel Breimaier
Zielsysteme	Aufsetzen eines OpenLDAP Servers	Thomas Pausch
Zielsysteme	Entwickeln eines automatisierten und dynamischen Synchronisierungsservices für LDAP Server	Thomas Pausch
Zielsysteme	Aufsetzen eines weiteren MYSQL Zielservers	Thomas Pausch
Zielsysteme	Entwickeln eines automatisierten und dynamischen Synchronisierungsservices für MYSQL Server	Thomas Pausch
Front-End	Erstellung Wireframe	Dino Osmanagic
Front-End	Auswahl kompatible Front-End Technologie	Dino Osmanagic
Front-End	Aufbau Front-End Grundstruktur	Dino Osmanagic

Front-End	Umsetzung Schematischer Login	Dino Osmanagic
Front-End	Herstellung Verbindung zum REST Service	Dino Osmanagic
Front-End	Umsetzung Neuanlage von Personen	Dino Osmanagic
Front-End	Umsetzung Bearbeiten von Personen	Dino Osmanagic
Front-End	Umsetzung Löschen von Personen	Dino Osmanagic
Front-End	Umsetzung Umzug von Personen in andere Organisationseinheiten	Dino Osmanagic
Front-End	Umsetzung Neuanlage von Organisationseinheiten	Dino Osmanagic
Front-End	Umsetzung Bearbeiten von Organisationseinheiten	Dino Osmanagic
Front-End	Umsetzung Löschen von Organisationseinheiten	Dino Osmanagic
Front-End	Umsetzung Neuanlage von Zielsystem-Konfiguration	Dino Osmanagic
Front-End	Umsetzung Bearbeiten von Zielsystem-Konfiguration	Dino Osmanagic
Front-End	Umsetzung Löschen von Zielsystem-Konfiguration	Dino Osmanagic
Front-End	Umsetzung Neuanlage von Zielsystem-Status	Dino Osmanagic
Front-End	Umsetzung Bearbeiten von Zielsystem-Status	Dino Osmanagic
Front-End	Umsetzung Löschen von Zielsystem-Status	Dino Osmanagic
Rest Service	Erstellung der Targetstate Entity im Rest Service	Daniel Hochgatterer
Rest Service	Erstellung der Department Entity im Rest Service	Daniel Hochgatterer
Rest Service	Erstellung Converter für ENUM Kompatibilität im Rest Service (nicht in der Endversion vorhanden)	Daniel Hochgatterer
Rest Service	Erstellung eigener Annotation für die Verwendung von ENUM Klassen im Restservice (nicht in der Endversion vorhanden)	Daniel Hochgatterer
Rest Service	Einbindung der ENUM Klasse Type in den Restservice (nicht in der Endversion vorhanden)	Daniel Hochgatterer
Rest Service	Einbindung UserAction-Logging (nicht in der Endversion vorhanden)	Daniel Hochgatterer, Lukas Gruber
Rest Service	Erstellung Department-Repository	Daniel Hochgatterer
Rest Service	Erstellung Targetstate-Repository	Daniel Hochgatterer
Rest Service	Erstellung Post-Mapping für Departments	Daniel Hochgatterer
Rest Service	Erstellung Fehlerbehandlung für Falscheingaben bei Post-Mapping für Departments	Daniel Hochgatterer
Rest Service	Erstellung Post-Mapping für Targetstates	Daniel Hochgatterer
Rest Service	Erstellung Fehlerbehandlung für Falscheingaben bei Post-Mapping für Targetstates	Daniel Hochgatterer
Rest Service	Überarbeiten der gestamten Targetstate Funktionalitäten im Restservice aufgrund von Datenbankänderungen	Daniel Hochgatterer
Rest Service	Erstellung Update(Patch)-Mapping für Departments	Daniel Hochgatterer

Rest Service	Erstellung Fehlerbehandlung für Falscheingaben bei Patch-Mapping für Departments	Daniel Hochgatterer
Rest Service	Erstellung Update(Patch)-Mapping für Targetstates	Daniel Hochgatterer
Rest Service	Erstellung Fehlerbehandlung für Falscheingaben bei Patch-Mapping für Targetstates	Daniel Hochgatterer
Rest Service	Überarbeitungen für Targetstate Patch/Post-Mapping aufgrund von Datenbankänderungen	Daniel Hochgatterer
Rest Service	Implementierung der GET-Mappings für Departments	Daniel Hochgatterer
Rest Service	Implementierung der GET-Mappings für Targetstates.	Daniel Hochgatterer
Rest Service	Suchfunktion-Methoden für UI zur Verfügung stellen. (GET-Mappings für alle Variablen der jeweiligen Klassen)	Daniel Hochgatterer, Lukas Gruber
Rest Service	Erstellung Delete-Mapping für Departments	Daniel Hochgatterer
Rest Service	Erstellung Delete-Mapping für Targetstates	Daniel Hochgatterer
Rest Service	Erstellung DeleteAll-Mapping für Departments	Daniel Hochgatterer
Rest Service	Erstellung DeleteAll-Mapping für Targetstates	Daniel Hochgatterer
Rest Service	Erstellung DeleteAll-Inactive Mapping für Targetstates	Daniel Hochgatterer
Rest Service	Erstellung Präsentation für eigens erstellte Rest-Funktionen sowohl für End-, als auch Zwischenpräsentation	Daniel Hochgatterer, Lukas Gruber
Rest Service	Regelmäßiges Testen der Funktionalitäten via Postman API	Daniel Hochgatterer, Lukas Gruber
Rest Service	Gemeinsame Rest-Service weite Bugfixes	Daniel Hochgatterer, Lukas Gruber
Rest Service	Erstellung der Employee Entity Klasse im Rest Service	Lukas Gruber
Rest Service	Erstellung der TargetConfig Entity Klasse im Rest Service	Lukas Gruber
Rest Service	Erstellung Employee Repository Klasse	Lukas Gruber
Rest Service	Erstellung TargetConfig Repository Klasse	Lukas Gruber
Rest Service	Implementierung des Post Mappings für EmployeeEntity im Employee Controller	Lukas Gruber
Rest Service	Fehlerbehandlungen für Falscheingaben bei Post Mappings für EmployeeEntitys im Employee Controller	Lukas Gruber
Rest Service	Implementierung des Post Mappings für TargetConfigEntity im TargetConfigController	Lukas Gruber

Rest Service	Fehlerbehandlungen für Falscheingaben bei Post Mappings für TargetConfigEntitys im TargetConfig Controller	Lukas Gruber
Rest Service	Überarbeitung der TargetConfig Klassen aufgrund von Änderungen an der Datenbank	Lukas Gruber
Rest Service	Erstellung von (Update)Patch Mappings für TargetConfigEntitys im TargetConfigController	Lukas Gruber
Rest Service	Implementierung von Fehlerbehandlung für Falscheingaben für Patch Mapping von TargetConfigEntitys im TargetConfig Controller	Lukas Gruber
Rest Service	Erstellung von (Update) Patch Mappings für EmployeeEntitys im Employee Controller	Lukas Gruber
Rest Service	Implementierung von Fehlerbehandlung für Falscheingaben für Patch Mapping von EmployeeEntitys im EmployeeController	Lukas Gruber
Rest Service	Erstellung der Exception Klassen für Employee und TargetConfig Entitäten	Lukas Gruber
Rest Service	Erstellung der Advice Klassen für Employee und TargetConfig Enitäten	Lukas Gruber
Rest Service Rest	Implementierung von Filtermethoden (Get Mappings) für Variablen von Employee	Lukas Gruber
Rest Service	Implementierung von Filtermethoden (Get Mappings) für Variablen von TargetConfig	Lukas Gruber
Rest Service	Implementierung fortlaufender ID für Employee Entity	Lukas Gruber
Rest Service	Implementierung fortlaufender ID für TargetConfig Entity	Lukas Gruber
Rest Service	Erstellung sämtlicher Delete Mappings für Employee Klasse	Lukas Gruber
Rest Service	Erstellung sämtlicher Delete Mappings für TargetConfig Klasse	Lukas Gruber
Rest Service	Regelmäßiges Testen der Funktionalitäten des Rest Services mittels Postman API	Lukas Gruber, Daniel Hochgatterer
Rest Service	Verfassen der Dokumentation für den Rest Service Kapitel 4 inkl. Subkapitel, sowie Kapitel 6 verwendete Technologien für Rest + Installationsanleitung Rest	Lukas Gruber, Daniel Hochgatterer
Rest Service	Verbindung des Rest Service mit der Datenbank über Application.property File im Java Projekt	Lukas Gruber
Rest Service	Implementierung automatischer Generierung von Loginnamen und Passwörtern bei Nichteingabe in Employee	Lukas Gruber
Rest Service	Recherche über mögliche Umsetzung eines Logging Systems	Lukas Gruber, Daniel Hochgatterer

Rest Service	BPMN Modell Schnittstelle, Rest	Lukas Gruber, Daniel Hochgatterer

2. Technische Dokumentation

2.1. Anforderungen

2.1.1. Frontend

Abbildung folgender Funktionen zur Verwaltung von Personen, Organisationseinheiten und Zielsystemen:

- Neuanlage, Bearbeiten, Löschen von Personen
- Neuanlage, Bearbeiten, Löschen von Organisationseinheiten
- Umzug von Personen in andere Organisationseinheiten
- Konfiguration von (mandantenspezifische) Zielsystemen
- Hinzufügen von Mandaten und Zielsystemen

2.1.2. Webservice | Rest Service

Der User darf bei Benutzung der Software keinen direkten Zugriff auf die Identitätsdatenbank des Unternehmens haben. Somit wurde das Team mit dem Problem konfrontiert, eine Schnittstelle zu implementieren, welche dem Frontend sämtliche Funktionen, zum Einfügen/Bearbeiten von Mitarbeiter/Abteilungsdatensätzen, ohne direkten Datenbankzugriff ermöglicht.

2.1.3. Datenbank

Als Anforderung an die Datenbank bzw. das Datenbanksystem wurde eine konsistente Datenbasis für die Verwaltung von Mitarbeiterdaten bei Miterbeitereintritten, Abteilungswechsel von Mitarbeiten und Mitarbeiteraustritten definiert. Die Datenbank verwaltet Mitarbeiter mehrere Organisationen mit beliebigen Eigenschaften. Diese Mitarbeiter können angelegt (bei Miterbeitereintritten), bearbeitet (bei Abteilungswechsel des Mitarbeiters) und gelöscht (bei Austritt eines Mitarbeiters) werden.

Darüber hinaus werden in der Datenbank die Einstellungen für die Schnittstellen zu den Zielsystemen gespeichert.

2.1.4. Datenmodell

Das Datenbankschema besteht aus folgenden Tabellen:

- Employee_table, diese Tabelle beinhaltet die Daten des Mitarbeiters. Jeder Mitarbeiter ist genau einem Mandanten zugeordnet und kann über die internen Mitarbeiter_ID identifiziert werden.
- Department_table, hier werden die einzelnen Mandanten bzw. Departments verwaltet.
- Targetconfig_table, in dieser Tabelle werden die Konfigurationsmöglichkeiten definiert, welche Daten an die Zielsysteme exportiert werden sollen.
- Targetstate_table, diese Tabelle bildet die Schnittstelleninformationen zu den einzelnen Zielsystemen ab.

2.1.5. Zielsysteme

CSV

Für dieses Zielsystem wurde ein Dateiexport ins das CSV-Format definiert, das die Daten automatisiert direkt auf dem entsprechenden Ablagepfad des jeweiligen Departments exportiert.

JSON

Das JSON Zielsystem wird ebenfalls über einen automatisierten Export auf den jeweiligen Ablagepfad des hinterlegten Departments exportiert.

LDAP

LDAP wird mittels eines Synchronisierungsservices in gewissen Zeitabständen automatisch mit den konfiguriertem LDAP Server synchronisiert. Der Service prüft hierbei mittels der Felder „last_Updated“ und „last_synced“ ob Änderungen übertragen werden müssen und überträgt diese, wenn notwendig. Dabei ist das Mappings auf den LDAP Server vorgegeben und ermöglicht so die übernahmen von allen Daten aus der zentralen Datenbank.

MYSQL

MSQL wird mittels demselben Synchronisierungsservices wie LDAP synchronisiert dabei wird wiederum auf Änderungen in der zentralen Datenbank geachtet und bei relevanten Änderungen diese in die Ziel Datenbank mittels JDBC übertragen. Dabei werden wiederum nur die in der zentralen Datenbank konfigurierten Departments mit deren Credentials angegebenen Datenbank

3. Technische Lösungsarchitektur Frontend

3.1. BPMN Diagramm Frontend

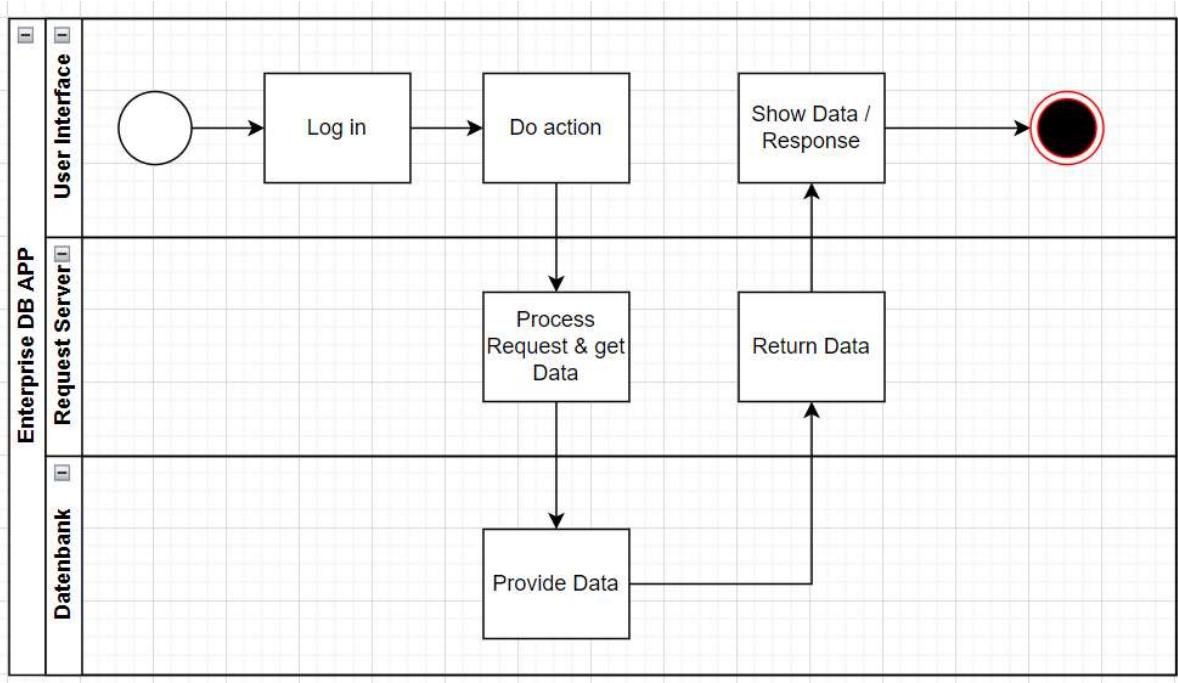


Abbildung 1: BPMN Frontend Diagramm

3.2. Technische Umsetzung

Anfangs wurde im Zuge der Evaluierung der Front-End Technologie die Nutzung von JavaFX in Betracht gezogen, was aus diversen Gründen schnell wieder verworfen wurde. Schlussendlich fiel die Entscheidung auf Angular 13 mit Bootstrap als "HTML, CSS & JS Framework".

Die Hauptbestandteile sind das "Routine.module", die "Pages", "Services", "Environment" und "Index".

Im "Index" ist die Verbindung zum Bootstrap CDN (Content Delivery Network) hinterlegt.

Im "Environment" ist als "Base-URL" Parameter der "Localhost Port" 8080 definiert, sodass dies bei Änderungen am Port nur an dieser Stelle geändert werden muss.

Im "Package.json" liegen die grundlegenden Konfigurationsdetails von "Angular Version 13".

Im "App.Module" liegen die definierten Komponenten des Projekts. Es umfasst das Basismodul zum Starten der Applikation wie auch diverse andere Module, die Bestandteil der Software sind. Dazu gehören die Module der jeweiligen Seiten zu "Employee", "Department", "Targetconfig" und "Targetstate", wie auch Formulardaten, HTTP Services und das Routing.

Die Basis Struktur ist im "app-routing.module.ts" hinterlegt, wo die Pfade der jeweiligen Seiten definiert sind. Die Webapplikation gliedert sich in die Seiten "Login", "Home (Employee)", "Single Employee View", "Department", "Targetconfig", "Targetstate" und "404".

Innerhalb von "Pages" ist die Struktur der jeweiligen Seiten als "HTML, CSS und Typescript" Datei definiert. Nachdem mit dem Bootstrap Framework gearbeitet wurde, wird kein eigener CSS Code pro Seite genutzt.

Innerhalb von "Services" wird die Verbindung zum REST Service und dahingehend dem Back End hergestellt, sodass die Daten entsprechend verarbeitet werden können. Für jeden der Teilbereiche "Employee", "Department", "Target Config" und "Target State" gibt es einen eigenen Service, der die Daten REST Service verarbeitet.

Ein prototypischer Login und Logout ist auf in der Webapplikation möglich. Nachdem die Möglichkeit zum Login im späteren Projektverlauf aufgekommen ist, wurde dies nur nachträglich ergänzt. Daher ist es im Gegensatz zum Regelfall, nicht die Startseite, der Applikation.

The form consists of a title 'Please sign in' at the top. Below it are two input fields: 'Loginname' and 'Password'. Both fields have placeholder text ('login name' and 'Password' respectively). A blue 'Sign in' button is positioned below the password field. At the bottom right of the form area, there is small text 'DKE PR'.

Abbildung 2: Login-Seite

Die Startseite der Webapplikation bildet die "Employee"-Übersicht mit Möglichkeit zur Anlage, Bearbeitung, Ansicht und Löschen von Mitarbeitern. Außerdem ist es die Startseite und ermöglicht eine Navigation zu den weiteren Seiten "Department", "Targetconfig" und "Targetstate".

Beim Anlegen von Mitarbeitern werden folgende Felder abgefragt:

- Sozialversicherungsnummer
- Vorname
- Nachname
- Login Name
- Passwort
- Start-Datum (erster Arbeitstag)
- End-Datum (letzter Arbeitstag)
- Department ID (Zuordnung zu einer Abteilung)
- Status (aktiv, inaktiv)

Nach erfolgter Eingabe der Felder wird mittels Buttons der Request ausgeführt und die Daten übermittelt. Etwaige Fehlermeldungen werden in der Browser-Konsole ausgegeben.

EnterpriseDbApp Home Department Target config Target state Logout

Enterprise Database Application

Managing employees and departments.

DKE PR

Add Employee Department Targetconfig Targetstate

Social Security Number 1234567891	First name First name	Last name last	Start Date TT.MM.JJJJ, --:--	End Date TT.MM.JJJJ, --:--					
Login name login	Password Password	Department ID 0	Active						
Add Employee									
ID	First name	Last name	Userlogin	Social Security Number	Department	Status	Start Date	End Date	Edit
1	Max	Mustermann	M.Mustermann	1234567684	1	Active	2000-07-21		View Edit delete
2	Maier	Peter	P.Maier	1234657685	1	Active	2000-07-21		View Edit delete
3	Doe	John	J.Doe	1234657686	1	Active	2000-07-21		View Edit delete
4	Ralf	Müller	R.Mueller	1234657687	1	Active	1990-01-13		View Edit delete
5	Frank	Eisenhof	F.Eisenhof	1234657688	1	Inactive	1995-05-31		View Edit delete

Abbildung 2: Front-End: Startseite

Beim Ändern von Mitarbeitern steht die Möglichkeit zur Änderung aller hinterlegten Informationen zu Verfügung.

EnterpriseDbApp Home Department Target config Target state Logout

DKE PR

Add Employee Department Targetconfig Targetstate

Social Security Number 1234567891	First name First name	Last name last	Start Date TT.MM.JJJJ, --:--	End Date TT.MM.JJJJ, --:--								
Login name login	Active											
Add Employee												
<div style="border: 1px solid #ccc; padding: 5px;"> <p>Change User</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 25%;">First Name Max</td> <td style="width: 25%;">Department 1</td> <td style="width: 25%;">Start Date TT.MM.JJJJ, --:--</td> <td style="width: 25%;">End Date TT.MM.JJJJ, --:--</td> </tr> <tr> <td>SVNR 1234567684</td> <td>Last Name Mustermann</td> <td>Password *****</td> <td>Status active</td> </tr> </table> <p style="text-align: right;">Close Save changes</p> </div>					First Name Max	Department 1	Start Date TT.MM.JJJJ, --:--	End Date TT.MM.JJJJ, --:--	SVNR 1234567684	Last Name Mustermann	Password *****	Status active
First Name Max	Department 1	Start Date TT.MM.JJJJ, --:--	End Date TT.MM.JJJJ, --:--									
SVNR 1234567684	Last Name Mustermann	Password *****	Status active									
ID	First name	Last name	Userlogin	Social Security Number								
1	Max	Mustermann	M.Mustermann	1234567684								
2	Maier	Peter	P.Maier	1234657685								
3	Doe	John	J.Doe	1234657686								
4	Ralf	Müller	R.Mueller	1234657687								
5	Frank	Eisenhof	F.Eisenhof	1234657688								
6	Anja	Lupin	A.Lupin	1234057690								
7	Dirk	Lempe	D.Lempe	1234057691								
8	Iris	Müller	I.Mueller	1234657692								

Abbildung 3: Front-End: Mitarbeiter bearbeiten

Das Löschen von Mitarbeitern erfolgt mittels Klicks auf den “Delete”-Button, der das Löschen mittels Erfolgsmeldung bestätigt.

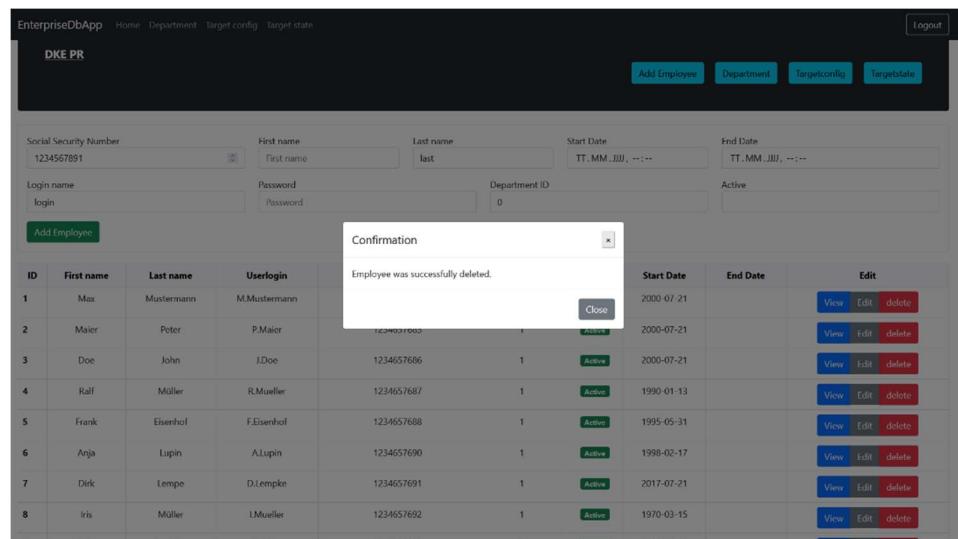


Abbildung 4: Front-End: Mitarbeiter löschen

Die Organisationseinheiten wurden als Abteilungen abgebildet und werden unter "Departments" ausgegeben.

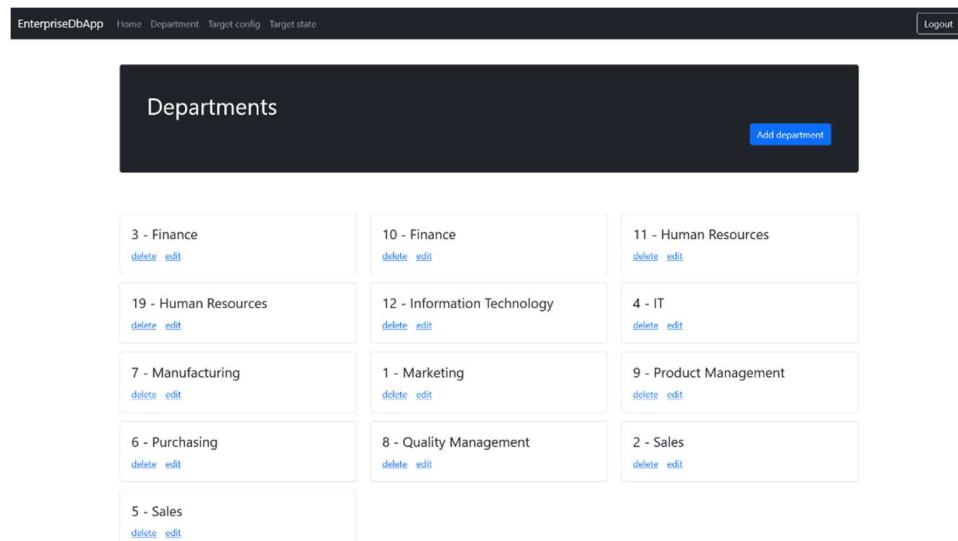


Abbildung 5: Front-End: Abteilungen

Es besteht die Möglichkeit des Anlegenes neuer Abteilungen durch Angabe des Namens. Daraufhin wird eine neue Abteilung angelegt und automatisch mit einer ID versehen.

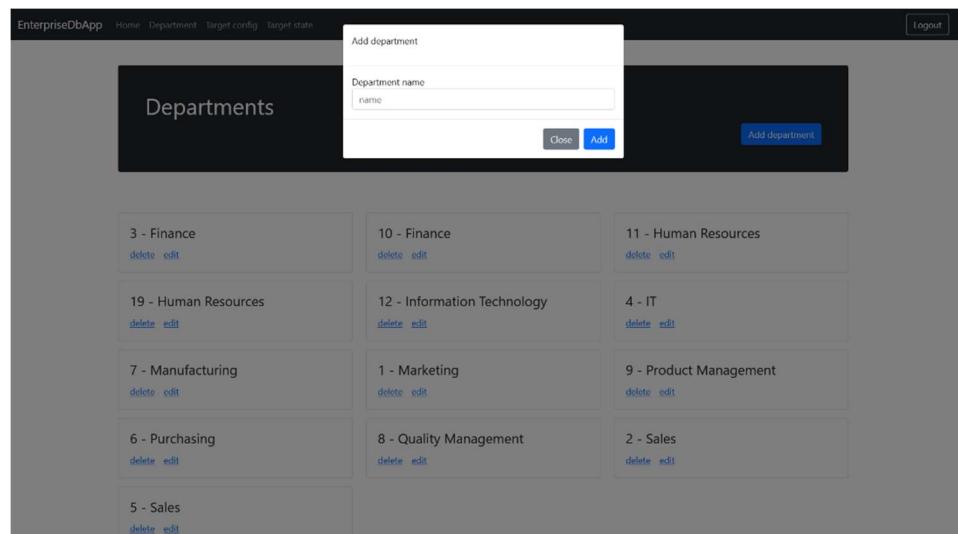


Abbildung 6: Front-End: Abteilungen anlegen

Das Bearbeiten von Abteilungen umfasst nur die Möglichkeit zur Namensänderung.

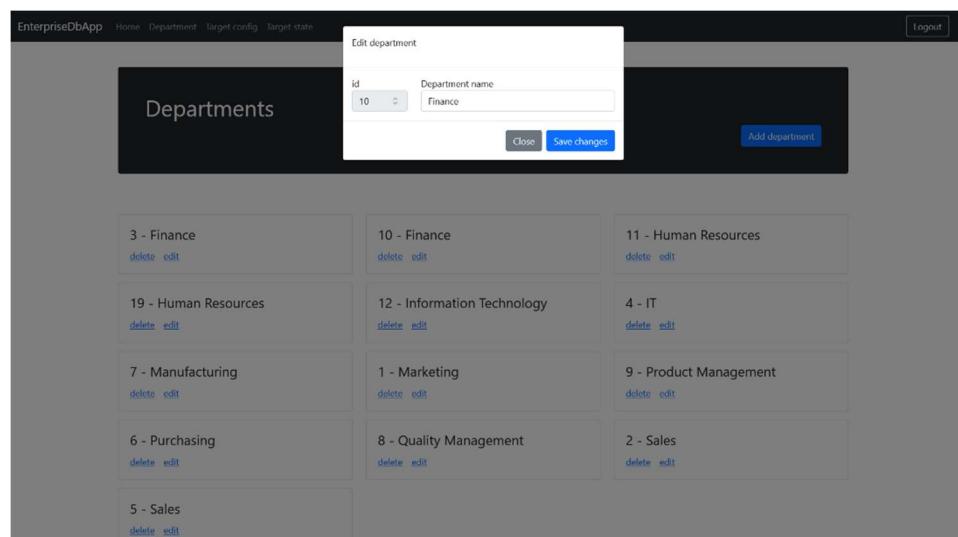


Abbildung 7: Front-End: Mitarbeiter bearbeiten

Das Löschen von Abteilungen erfordert ein erneutes Bestätigen, woraufhin es gelöscht wird.

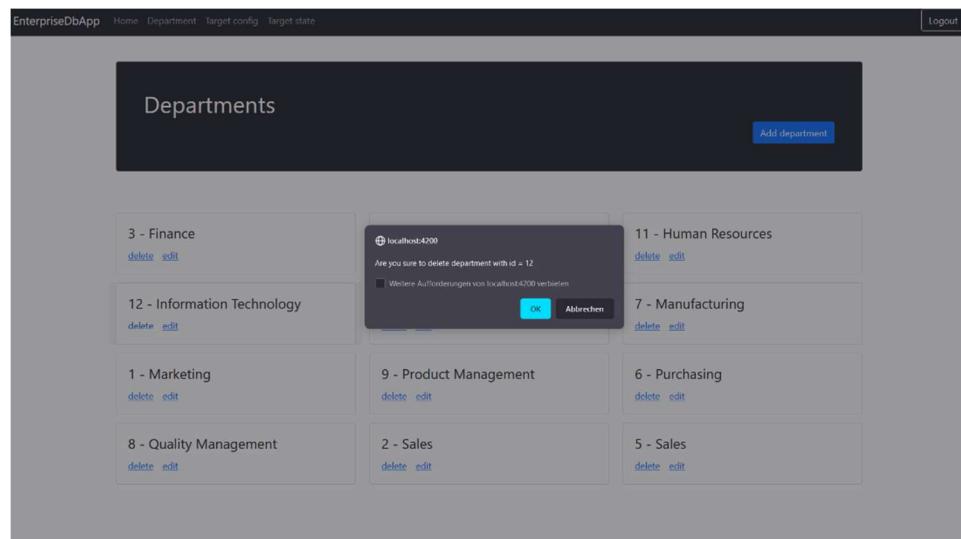


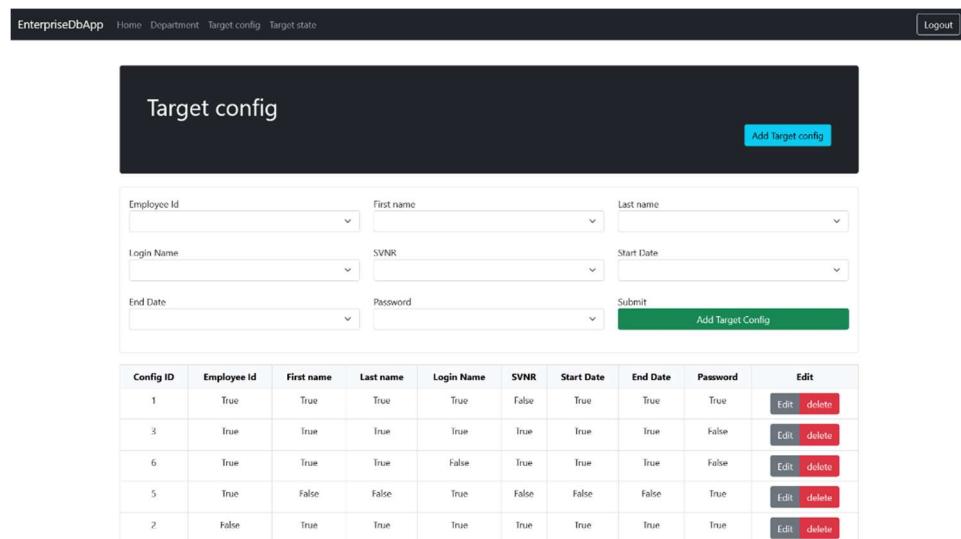
Abbildung 8: Front-End: Mitarbeiter löschen

Die Zielsystem-Konfigurations-Seite “Target-Config” umfasst die Möglichkeit zum Erstellen, Bearbeiten und Löschen von Zielsystem-Konfigurationen.

Das Anlegen der Zielsystem-Konfiguration erfordert die Wahl folgender der Parameter:

- Employee ID
- Vorname
- Nachname
- Login Name
- Sozialversicherungsnummer
- Passwort
- Start-Datum

Mittels Auswahl, ob der jeweilige Parameter in der jeweiligen Konfiguration inkludiert werden soll (true/false pro Parameter) wird eine neue Zielsystem-Konfiguration und einzigartiger ID angelegt.



Config ID	Employee Id	First name	Last name	Login Name	SVNR	Start Date	End Date	Password	Edit
1	True	True	True	True	False	True	True	True	Edit Delete
3	True	True	True	True	True	True	True	False	Edit Delete
6	True	True	True	False	True	True	True	False	Edit Delete
5	True	False	False	True	False	False	False	True	Edit Delete
2	False	True	True	True	True	True	True	True	Edit Delete

Abbildung 9: Front-End: Zielsystem-Konfiguration

Das Bearbeiten der Zielsystem-Konfiguration umfasst die Möglichkeit zum Ändern der Ausgabe der jeweiligen Parameter und dem Wechsel auf “true/false”. Mittels Klick auf “Speichern” kann die Eingabe bestätigt werden.

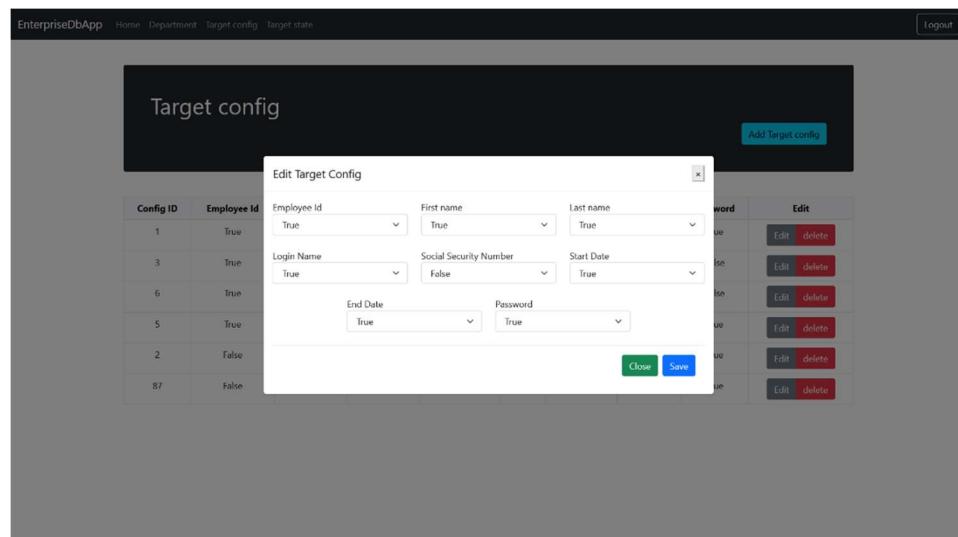


Abbildung 10: Front-End: Zielsystem-Konfiguration bearbeiten

Das Löschen von Zielsystem-Konfigurationen erfolgt mittels Klick auf “Delete”, was eine Information und Bestätigung zur Durchführung der Aktion nach sich zieht.

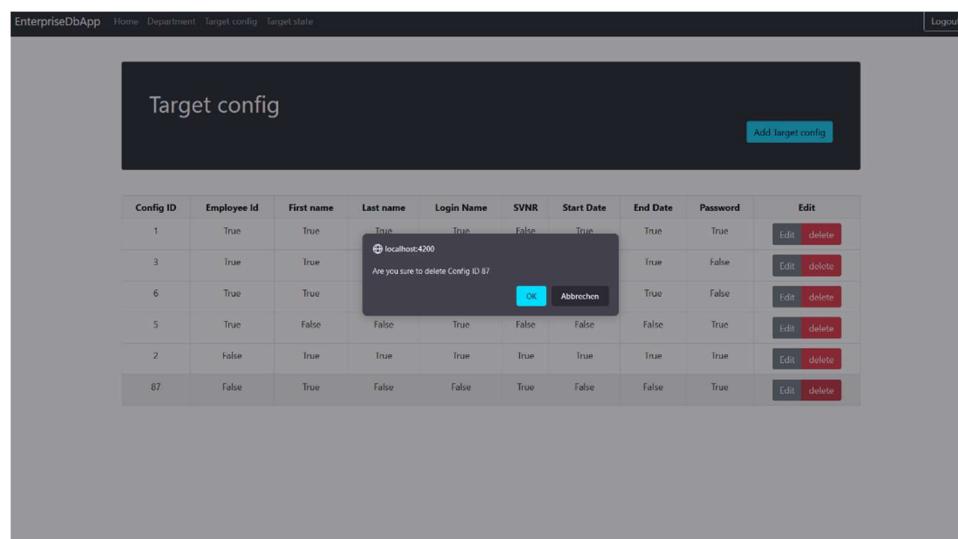


Abbildung 11: Front-End: Zielsystem-Konfiguration löschen

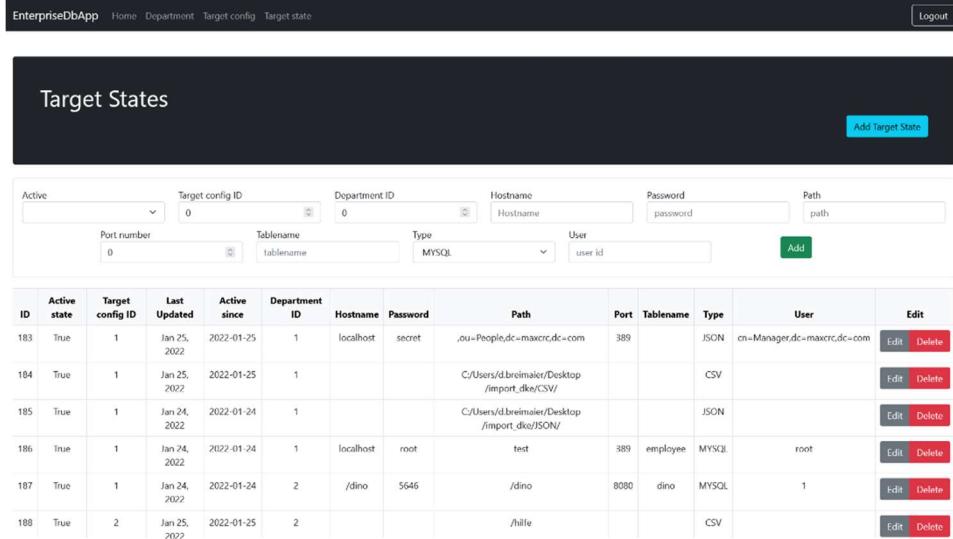
Der Zielsystem-Zustand wird auf der “Targetstate”-Seite abgebildet, welche das Anlegen, Bearbeiten und Löschen vom jeweiligen “Targetstate” ermöglicht.

Im späteren Verlauf des Projekts wurde die Differenzierung der Eingabeparameter nach Typ ergänzt, sodass beim Anlegen eines neuen “Targetstate” unterschiedliche Auswahlfelder angezeigt werden.

Bei MySQL werden folgende Parameter abgefragt:

- Typ (MySQL, LDAP, JSON, CSV)
- Status (aktiv, inaktiv)
- Target Config ID

- Department ID
- Path
- Hostname
- Password
- Port
- Tablename
- User



ID	Active state	Target config ID	Last Updated	Active since	Department ID	Hostname	Password	Path	Port	Tablename	Type	User	Edit
183	True	1	Jan 25, 2022	2022-01-25	1	localhost	secret	,ou=People,dc=maxcrc,dc=com	389		JSON	cn=Manager,dc=maxcrc,dc=com	<button>Edit</button> <button>Delete</button>
184	True	1	Jan 25, 2022	2022-01-25	1			C:/Users/d.breimaijer/Desktop/import_dke/CSV/			CSV		<button>Edit</button> <button>Delete</button>
185	True	1	Jan 24, 2022	2022-01-24	1			C:/Users/d.breimaijer/Desktop/import_dke/JSON/			JSON		<button>Edit</button> <button>Delete</button>
186	True	1	Jan 24, 2022	2022-01-24	1	localhost	root	test	389	employee	MYSQL	root	<button>Edit</button> <button>Delete</button>
187	True	1	Jan 24, 2022	2022-01-24	2	/dino	5646	/dino	8080	dino	MYSQL	1	<button>Edit</button> <button>Delete</button>
188	True	2	Jan 25, 2022	2022-01-25	2			/hilfe			CSV		<button>Edit</button> <button>Delete</button>

Abbildung 12: Front-End: Zielsystem-Zustand

Bei CSV und JSON werden folgende Parameter abgefragt:

- Typ (mySQL, LDAP, JSON, CSV)
- Status (aktiv, inaktiv)
- Target Config ID
- Department ID
- Path

Target States													
<input style="float: right; margin-right: 10px;" type="button" value="Add Target State"/>													
Active		Target config ID		Department ID		Path		Type					
ID	Active state	Target config ID	Last Updated	Active since	Department ID	Hostname	Password	Path	Port	Tablename	Type	User	Edit
183	True	1	Jan 25, 2022	2022-01-25	1	localhost	secret	,ou=People,dc=maxcrc,dc=com	389		JSON	cn=Manager,dc=maxcrc,dc=com	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
184	True	1	Jan 25, 2022	2022-01-25	1			C:/Users/d.breimaijer/Desktop/import_dke/CSV/			CSV		<input type="button" value="Edit"/> <input type="button" value="Delete"/>
185	True	1	Jan 24, 2022	2022-01-24	1			C:/Users/d.breimaijer/Desktop/import_dke/JSON/			JSON		<input type="button" value="Edit"/> <input type="button" value="Delete"/>
186	True	1	Jan 24, 2022	2022-01-24	1	localhost	root	test	389	employee	MySQL	root	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
187	True	1	Jan 24, 2022	2022-01-24	2	/dino	5646	/dino	8080	dino	MySQL	1	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
188	True	2	Jan 25, 2022	2022-01-25	2			/hilfe			CSV		<input type="button" value="Edit"/> <input type="button" value="Delete"/>
189	True	2	Jan 25, 2022	2022-01-25	2	TestHost	testpw	/hilfe	8080		LDAP	Daniel	<input type="button" value="Edit"/> <input type="button" value="Delete"/>

Abbildung 13: Zielsystem-Zustand CSV

Target States													
<input style="float: right; margin-right: 10px;" type="button" value="Add Target State"/>													
Active		Target config ID		Department ID		Path		Type					
ID	Active state	Target config ID	Last Updated	Active since	Department ID	Hostname	Password	Path	Port	Tablename	Type	User	Edit
183	True	1	Jan 25, 2022	2022-01-25	1	localhost	secret	,ou=People,dc=maxcrc,dc=com	389		JSON	cn=Manager,dc=maxcrc,dc=com	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
184	True	1	Jan 25, 2022	2022-01-25	1			C:/Users/d.breimaijer/Desktop/import_dke/CSV/			CSV		<input type="button" value="Edit"/> <input type="button" value="Delete"/>
185	True	1	Jan 24, 2022	2022-01-24	1			C:/Users/d.breimaijer/Desktop/import_dke/JSON/			JSON		<input type="button" value="Edit"/> <input type="button" value="Delete"/>
186	True	1	Jan 24, 2022	2022-01-24	1	localhost	root	test	389	employee	MySQL	root	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
187	True	1	Jan 24, 2022	2022-01-24	2	/dino	5646	/dino	8080	dino	MySQL	1	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
188	True	2	Jan 25, 2022	2022-01-25	2			/hilfe			CSV		<input type="button" value="Edit"/> <input type="button" value="Delete"/>
189	True	2	Jan 25, 2022	2022-01-25	2	TestHost	testpw	/hilfe	8080		LDAP	Daniel	<input type="button" value="Edit"/> <input type="button" value="Delete"/>

Abbildung 14: Front-End: Zielsystem-Zustand: JSON

Bei LDAP werden folgende Parameter abgefragt:

- Typ (MySQL, LDAP, JSON, CSV)
- Status (aktiv, inaktiv)
- Target Config ID
- Department ID
- Path
- Hostname
- Password
- Port
- User

EnterpriseDbApp Home Department Target config Target state Logout

Target States

Add Target State

Active	Target config ID	Department ID	Hostname	Password	Path
Active	0	0	Hostname	password	path
	Port number	Type	User		
	0	LDAP	user id		
					Add

ID	Active state	Target config ID	Last Updated	Active since	Department ID	Hostname	Password	Path	Port	Tablename	Type	User	Edit
183	True	1	Jan 25, 2022	2022-01-25	1	localhost	secret	,ou=People,dc=maxcrc,dc=com	389		JSON	cn=Manager,dc=maxcrc,dc=com	<button>Edit</button> <button>Delete</button>
184	True	1	Jan 25, 2022	2022-01-25	1			C:/Users/d.breimaijer/Desktop/import_dke/CSV/			CSV		<button>Edit</button> <button>Delete</button>
185	True	1	Jan 24, 2022	2022-01-24	1			C:/Users/d.breimaijer/Desktop/import_dke/JSON/			JSON		<button>Edit</button> <button>Delete</button>
186	True	1	Jan 24, 2022	2022-01-24	1	localhost	root	test	389	employee	MYSQL	root	<button>Edit</button> <button>Delete</button>
187	True	1	Jan 24, 2022	2022-01-24	2	/dino	5646	/dino	8080	dino	MYSQL	1	<button>Edit</button> <button>Delete</button>
188	True	2	Jan 25, 2022	2022-01-25	2			/hilfe			CSV		<button>Edit</button> <button>Delete</button>

Abbildung 15: Front-End: Zielsystem-Zustand LDAP

Das Ändern von einem “Targetstate” ermöglicht das Ändern der hinterlegten Daten. Leider kam die Differenzierung der Eingabe nach Typ relativ spät im Projekt auf, weshalb nicht mehr ausreichend viel Zeit für eine gesonderte Darstellung der Änderungsmaske nach Typ zur Verfügung stand. Dies würde unter anderem gesonderte Modals pro Typ bedeuten.

EnterpriseDbApp Home Department Target config Target state Logout

Target States

Add Target State

Active	Target config ID	Department ID	Hostname	Password	Path
Active	0	0	localhost	*****	,ou=People,dc=maxcrc,dc=com
	Port number	Type	Path	Port	
	0	JSON	389		
		User			Add

ID	Active state	Target config ID	Last Updated	Active since	Department ID	Hostname	Password	Path	Port	Tablename	Type	User	Edit
183	True	1	Jan 25, 2022	2022-01-25	1	localhost	secret	,ou=People,dc=maxcrc,dc=com	389		JSON	cn=Manager,dc=maxcrc,dc=com	<button>Edit</button> <button>Delete</button>
184	True	1	Jan 25, 2022	2022-01-25	1			C:/Users/d.breimaijer/Desktop/import_dke/CSV/			CSV		<button>Edit</button> <button>Delete</button>
185	True	1	Jan 24, 2022	2022-01-24	1			C:/Users/d.breimaijer/Desktop/import_dke/JSON/			JSON		<button>Edit</button> <button>Delete</button>
186	True	1	Jan 24, 2022	2022-01-24	1	localhost	root	test	389	employee	MYSQL	root	<button>Edit</button> <button>Delete</button>
187	True	1	Jan 24, 2022	2022-01-24	2	/dino	5646	/dino	8080	dino	MYSQL	1	<button>Edit</button> <button>Delete</button>
188	True	2	Jan 25, 2022	2022-01-25	2			/hilfe			CSV		<button>Edit</button> <button>Delete</button>

Change Target state

ID	Active	Target Config ID	Department ID
183	True	1	1
Hostname	Password	Path	Port
localhost	*****	,ou=People,dc=maxcrc,dc=com	389
Tablename	Type	User	
	JSON	cn=Manager,dc=maxcrc,dc=com	

Close Save changes

Abbildung 16: Front-End: Zielsystem-Zustand bearbeiten

Das Löschen des jeweiligen “Targetstate” zieht eine entsprechende Erfolgsmeldung nach sich.

EnterpriseDbApp													Logout
ID	Active	Target State ID	Created Date	Last Modified	Target State	Host	Port	Secret	Path	Protocol	DB Type	User	Action
183	True	1	Jan 25, 2022	2022-01-25	1	localhost	secret	,ou=People,dc=maxorc,dc=com	389	JSON	cn=Manager,dc=maxorc,dc=com		Edit Delete
184	True	1	Jan 25, 2022	2022-01-25	1			C:/Users/lbreinmaier/Desktop/_import_dkr/CSV/		CSV			Edit Delete
185	True	1	Jan 24, 2022	2022-01-24	1			C:/Users/lbreinmaier/Desktop/_import_dkr/JSON/		JSON			Edit Delete
186	True	1	Jan 24, 2022	2022-01-24	1	localhost	root	test	389	employee	MYSQL	root	Edit Delete
187	True	1	Jan 24, 2022	2022-01-24	2	/dino	5646	/dino	8080	dino	MYSQL	1	Edit Delete
188	True	2	Jan 25, 2022	2022-01-25	2	Confirmation				CSV			Edit Delete
189	True	2	Jan 25, 2022	2022-01-25	2	Target state was successfully deleted.				0	LDAP	Daniel	Edit Delete
192	True	2	Jan 25, 2022	2022-01-25	2			/path	0	LDAP	Daniel		Edit Delete
193	True	1	Jan 25, 2022	2022-01-25	2			/path	0	CSV			Edit Delete
194	True	1	Jan 25, 2022	2022-01-25	2			/path	0	JSON			Edit Delete
195	True	1	Jan 25, 2022	2022-01-25	2	host	password	/path	8080	LDAP	1		Edit Delete
196	False	1	Jan 25, 2022		2	host	password	/path	8080	tablename	MYSQL	1	Edit Delete

Abbildung 17: Front-End: Zielsystem-Zustand löschen

4. Technische Lösungsarchitektur REST-Service

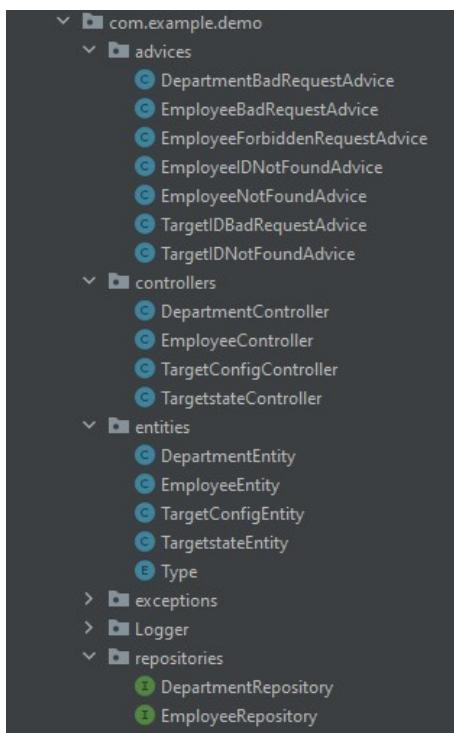
Als Schnittstelle zwischen dem Userinterface und der Identitätsdatenbank wurde in unserem Projekt ein REST Service ausgewählt, welcher mittels des SPRING Initializrs aufgesetzt und an die, durch die Datenbank vorgegebenen Tabellen angepasst wurde. Dieser Abschnitt des Projekts wurde von den Teammitgliedern Lukas Gruber und Daniel Hochgatterer übernommen und vollständig implementiert. Somit beinhaltet der Rest Service sämtliche HTTP GET, SET, PATCH, POST & DELETE Methoden, inkl. aller benötigten REST-Annotationen, für die, in der Datenbank erstellten, Tabellen Employee, Departments, Targetstate & Targetconfig. Die beiden letzteren Tabellen sind für den Export in die verschiedenen Zielsysteme (MYSQL, JSON, LDAP & CSV) zuständig.

4.1. BPMN Diagramm Rest-Service

Aufgrund der Größe des BPMN Diagramms für den Rest Service konnte dieses nicht in der Dokumentation eingefügt werden. Das Diagramm befindet sich als „rest_service_bpmn“ png Datei im “Dokumentation” Ordner in der Abgabe.

4.2. Projektstruktur/ Technische Umsetzung

Als Projektstruktur wurde das Preset des Spring Initializers verwendet. Hierbei werden direkt alle benötigten Maven-Dependencies automatisiert in das Projekt geladen und das Projekt wird, wie auf dem folgenden Screenshot ersichtlich in Controller für das Methodenhandling, Entities, für die in der Datenbank behandelten Tabellen, und Exceptions&Advices für das Exceptionhandling, auf das im Punkt 4.6 Exception Handling genauer eingegangen wird, eingeteilt. Die Entity Klassen im Entitypackage beinhalten somit eine Repräsentation der Tabellen in Form von Java Objekten inkl. Get/Set Methoden für alle benötigten Tabellen. Das HTTP-Methodenhandling wurde für jede Entityklasse in einem eigenen Controller im „controllers“ Package umgesetzt. Diese beiden Packages kommunizieren über die erstellten Repositories im „repositories“ Package. Das Zusammenspiel dieser Komponenten wird im Folgenden anhand der für die Umsetzung der Mitarbeiterverwaltung benötigten Klassen veranschaulicht. So stellt die EmployeeEntity Klasse eine Repräsentation der Employee Tabelle aus der Datenbank dar, das Mapping dieser Java Klasse zu der Tabelle in der Datenbank erfolgt hierbei über die @Table Annotation. In dieser Klasse werden, wie in Abbildung 18 ersichtlich, die Attribute der Employee Tabelle mittels @Column Annotationen an korrespondierende Java Variablen gekoppelt. Der Primary Key der Tabelle wird über die @Id Annotation der Java Variable „employeeid“ zugewiesen. Hierbei wird die Hibernate Bibliothek verwendet um eine fortlaufende ID als Primärschlüssel der Employee Datensätze zu definieren. Diese Zuweisung der Attribute der Tabellen zu Java Variablen erlaubt im späteren Verlaufe die Manipulation sowie Erstellung von Mitarbeiterdaten in der Datenbank durch JSON Objekte.



```

@Entity
@Table(name = "employee")
public class EmployeeEntity {

    //id wird über einen generator automatisch generiert und ist eine fortlaufende nummer
    @Id @GeneratedValue(strategy = GenerationType.AUTO, generator = "employeeid_sequence")
    @SequenceGenerator(name = "employeeid_sequence", allocationSize=1)
    int employeeid;

    @Column(name ="svnr")
    private String svnr;
    @Column(name ="active")
    private int active;
    @Column(name ="first_name")
    private String firstName;
    @Column(name ="last_name")
    private String lastName;
    @Column(name ="login_name")
    private String loginName;
    @Column(name ="password")
    private String password;
    @Column(name ="start_date")
    private Date start_date;
    @Column(name ="end_date")
    private Date end_date;
    @Column(name ="department_id")
    private int departmentId;
    @Column(name ="last_changed")
    private Instant lastChanged;
}

```

Abbildung 18: Employee Entity

Abbildung 19: Java Projektstruktur

Die Controller Klassen sind für das Methodenhandling des Services zuständig. Als Beispiel kann hierbei die Post Methode „newEmployeeEntity“ genannt werden.

Hierbei wird ein Ausschnitt dieser Methode in Abbildung 21 dargestellt. Diese Methode wird dazu verwendet um einen neuen Mitarbeiter in die Employee Tabelle der Datenbank einzufügen. Hierbei befinden sich die Informationen eines neu anzulegenden Mitarbeiters in einem sogenannten RequestBody. Abbildung 20 veranschaulicht einen solchen RequestBody.

```

    "svnr": 4216110699,
    "active": 1,
    "firstName": "Max",
    "lastName": "Mustermann",
    "start_date": "2011-11-11",
    "end_date": null,
    "department": 3
  }
}

```

Abbildung 20: JSON Requestbody

```

@PostMapping("/employee")
EmployeeEntity newEmployeeEntity(@RequestBody EmployeeEntity employeeEntity) {
    if (ObjectUtils.isEmpty(employeeEntity.getSVNR())) {
        throw new EmployeeBadRequestException("SVNR should not be empty!");
    }
    if (employeeEntity.getSVNR().length() != 10 || !employeeEntity.getSVNR().matches(regex: "[0-9]{10}")) {
        throw new EmployeeBadRequestException("SVNR formatted incorrectly!");
    }
    EmployeeEntity toSave = repository.save(temp);
    return toSave;
}

```

Abbildung 21: Employee Post Methode

Genauer genommen handelt es sich bei diesem Requestbody um ein JSON Objekt welches einen neuen Mitarbeiter, in Form einer EmployeeEntity, repräsentiert. Im Zuge eines Post Requests wird anhand der Werte des JSON Objekts durch den Service eine neuer Mitarbeiterdatensatz erstellt. Hierbei verwendet der Service die in der EmployeeEntity Klasse festgelegten Mappings zwischen den Attributen der Tabelle und den EmployeeEntity Objektvariablen um die Informationen in die korrekten Felder einzutragen.

Im Zuge dessen gilt es zu erwähnen, dass die grundlegenden Funktionalitäten für die Manipulation und Abfrage der Daten durch ein Interface bereitgestellt werden. Hierbei besitzt jede Entität ein eigenes Interface welche das sogenannte JpaRepository Interface erweitern. Dieses JpaRepository beinhaltet grundlegende Funktionalitäten zur Datenbankabfrage und -manipulation. Als Beispiele können das Suchen sämtlicher Datensätze in einer Tabelle, die Suche eines bestimmten Datensatzes über dessen ID oder das Löschen eines Datensatzes genannt werden. Damit für die einzelnen Entitäten neben den bereits kurz erwähnten Standardfunktionen auch spezifischere Abfrage und Manipulationsmethoden zur Verfügung stehen, müssen diese im jeweiligen Repository deklariert werden. Als Beispiel dient hierfür das EmployeeRepository, dieses Repository stellt Methoden zur Verfügung die unter Anderem die Suche eines Mitarbeiters über dessen Vornamen, Aktivitätsstatus, Nachnamen oder dessen zugehörige Abteilung ermöglichen. Dieses Repository ist hierbei in Abbildung 22 ersichtlich.

```

public interface EmployeeRepository extends JpaRepository<EmployeeEntity, Integer> {

    List<EmployeeEntity> findByFirstName(String firstName);
    List<EmployeeEntity> findByLastName(String lastName);
    List<EmployeeEntity> findByActive(int active);
    List<EmployeeEntity> findByDepartmentId(int department_Id);
    List<EmployeeEntity> findByLoginName(String login_name);

    @Transactional
    List<EmployeeEntity> removeByActive(int active);
}

```

Abbildung 22: EmployeeRepository

Die Geschäftslogik dieser Methoden muss hierbei nicht explizit implementiert werden.

Der Rest Service erkennt hierbei über den Namen der jeweiligen Methode nach welchem Suchkriterium gesucht werden muss. So erkennt der Service beispielsweise, dass es sich bei der findByFirstName Methode um eine Suche anhand des Vornamens handelt.

Damit Controller Klassen Zugriff auf diese Methoden haben wird das jeweilige Repository im Controller Konstruktor instanziert, dieser Vorgang ist in Abbildung 23 ersichtlich.

```
EmployeeController(EmployeeRepository repository) { this.repository = repository; }
```

Abbildung 23: Initialisierung eines Controllers mit dem Repository

Somit hat der Controller nun Zugriff auf die Methoden des Repositorys und kann sie im Zuge der Controller Methoden anwenden.

Da die Software Userdaten einsehen lässt, werden von dem REST-Service Passwörter direkt über den MD5 Verschlüsselungsalgorithmus verschlüsselt und nur auf diese Art und Weise in der Datenbank abgespeichert.

4.3. Fortschritt, Planung

Der Fortschritt des Projekts wurde mit dem Team stets über ein restteaminternes Github Repository geteilt und war jederzeit für das gesamte Team einsehbar. Des Weiteren wurde der Fortschritt im gesamten Projekt und spezifisch auf die aufgeteilten Aufgaben, in regelmäßigen Abständen, über online Meetings, mit dem Team kommuniziert. Diese Meetings fanden auf der Konferenzplattform Discord statt und wurden zuvor über eine gruppeninterne Whatsapp-Gruppe geplant.

4.4. Einteilung im Rest-Team

Die Zuständigkeitsbereiche zwischen Lukas Gruber und Daniel Hochgatterer wurden anhand der, im Service, zu implementierenden Datenbanktabellen eingeteilt. Somit wurden von jedem der beiden Teammitglieder zwei, zuvor von Daniel Breimeier und Thomas Pausch erstellte Tabellen, im REST-Service, in Form von Entities mit sämtlichen HTTP-Methoden, umgesetzt. Hierbei übernahm Daniel Hochgatterer die Implementierung der Department und der Targetstate Tabelle. Lukas Gruber war im Zuge dieses Projekts für die Implementierung des Services bezüglich der Employee und TargetConfig Tabelle verantwortlich. Beide Entwickler implementierten hierbei für ihre Tabellen die benötigten Entity, Repository, Controller, Exception sowie Advice Klassen.

Die Dokumentation der Schnittstelle wurde von beiden Teammitgliedern durchgeführt.

Ebenso arbeiteten beide Teammitglieder gemeinsam an der Implementierung eines Logging-Systems, welches jedoch, aufgrund der fehlenden Anforderung des Projektauftraggebers, nicht vollständig umgesetzt wurde. Bezuglich der Präsentation des Projektsfortschritts und der Powerpointgestaltung übernahm jeder der Beteiligten den von ihm umgesetzten Inhalt.

4.5. Filtermethoden für Entity Klassen

Das Rest Team implementierte für die Entitätsklassen (Employee, Department, Targetconfig und Targetstate) Filtermethoden. Bei diesen Filtermethoden handelt es sich um Get Methoden die beispielsweise die Suche nach Mitarbeitern anhand deren Vornamen, Nachnamen, Aktivitätsstatus oder zugewiesenen Department erlauben. Diese Methoden wurden jedoch vom Verantwortlichen für das Front End nicht in die UI eingebunden.

4.6. Exception Handling

Wie man in den Abbildungen 24 und 25 bereits erkennen kann, wurde vom Rest Team auch das komplette Exceptionhandling im Service implementiert. Somit wird in jeder Situation auf Falscheingaben des Users reagiert und dieser über diverse „Exceptions & Advices“ über diese Fehler informiert. Hier wurde vom Team auf wesentlich mehr Falscheingaben/Fehlermeldungen geprüft als der User im Userinterface durch Usereingaben überhaupt hervorrufen kann. Somit ist im Rest-Service eine komplette Fehlerbehandlung durch Usereingaben vorhanden, diese wurde jedoch von dem UserInterface Verantwortlichen nicht umgesetzt und deswegen ist im UI kein Exceptionhandling vorhanden.

In unserem Rest Service befinden sich für jede Entitätsart, also Employee, Department, Targetconfig sowie Targetstate, mehrere Exceptionklassen. Hierbei werden zum Einen Falscheingaben des Users über Badrequest Exceptions vermerkt, zum Anderen werden Suchanfragen welche kein Ergebnis liefern über NotFound Exceptions gekennzeichnet. Hierbei war es dem Team wichtig möglichst spezifische und genaue Exceptions in Fehlersituationen zu werfen. Deshalb wurde von der Implementierung zweier eher generell gehaltenen Exceptionklassen abgesehen und es wurden stattdessen für jede Entitätsart eigenständige Exceptionklassen implementiert. Diese klare Trennung und Namensgebung der Exceptions ermöglicht somit, dass in Fehlersituationen schnell die Fehlerursache ersichtlich ist.

So wird beispielsweise bei der Falscheingabe von Mitarbeiterinformationen anstelle einer generellen Badrequest Exception eine für die Situation angemessene EmployeeBadRequestException geworfen und somit ist schneller ersichtlich das eine Falscheingabe bei einer Employee Entität aufgetreten ist.

Da es sich bei unserem Programm um einen Webservice handelt war es den Rest Service Entwicklern wichtig, http Statuscodes in Fehlersituationen mitauszugeben. Hierbei beinhalten BadRequestExceptions den korrespondierenden 400 Statuscode und NotFoundExceptions den 404 http Statuscode. Dieses Verhalten wird mit sogenannten Advice Klassen erzielt. In diesen Adviceklassen werden die jeweiligen Exceptions über die @ExceptionHandler Annotationen an die jeweiligen Advices gebunden. Diese Koppelung erlaubt es nun die Exceptions mit entsprechenden http Statuscodes zu erweitern. Ein Beispiel für eine solche Advice Klasse ist in Abbildung 25 ersichtlich.

```
package com.example.demo.exceptions;

public class EmployeeBadRequestException extends RuntimeException {

    public EmployeeBadRequestException(String error) { super(error); }
}
```

Abbildung 24: EmployeeBadRequestException Klasse

```

@ControllerAdvice
public class EmployeeBadRequestAdvice {

    @ExceptionHandler(EmployeeBadRequestException.class)
    public ResponseEntity<ExceptionResponse> customException(EmployeeBadRequestException ex) {
        ExceptionResponse response=new ExceptionResponse();
        response.setErrorCode("BAD_REQUEST");
        response.setErrorMessage(ex.getMessage());
        response.setTimestamp(LocalDateTime.now());

        return new ResponseEntity<ExceptionResponse>(response, HttpStatus.BAD_REQUEST);
    }
}
  
```

Abbildung 25: EmployeeBadRequestAdvice Klasse

So werden dieser Exception unter anderem der Bad Request http Errorcode und der Zeitpunkt des Fehlerauftretts angefügt. Im Zuge dessen gilt es zu erwähnen, dass jede von den Entwicklern implementierte Exception ein eigenständiger Advice zugewiesen wird. Dies erlaubt es, dass in späteren Versionen des Services noch genauere Informationen an die jeweilige Exceptions gebunden werden können.

Abschließend lässt sich somit behaupten, dass der implementierte Rest Service die komplette Datenbank mit allen benötigten HTTP Methoden abdeckt.

4.7. Anmerkungen zur Implementierung

Im Folgenden werden einige Designentscheidungen der Entwickler des Services behandelt.

4.7.1. Verwendung der Werte 1 und 2 für True respektive False

Wie während der Implementierung des Services ersichtlich geworden ist, führte die Verwendung von der Zahl 0 als Repräsentation für den Booleanwert False zu Problemen. In unserem Projekt gibt es in der Datenbank einige Attribute die einen Status repräsentieren und dementsprechend auf True oder False gesetzt werden sollen. Diese umfassen folgende Tabellen Attribute:

- Das „active“ Attribut in der Employee Tabelle
- Alle Attribute der “Targetconfig” Tabelle
- Das Active Attribut in der Targetstate Tabelle

Für diese Attribute mussten die Entwickler den Wert „2“ für False verwenden, da die Verwendung des Wertes 0 als Repräsentation für den Booleanwert False zu Problemen führte. Diese Probleme traten hierbei bei „Patch“ Methoden, also Methoden zur Veränderung bestehender Datensätze, auf. Betrachtet man die Fehlerüberprüfung des Active Felds in der Patch Methode für die TargetState Tabelle, so wird ersichtlich, dass dieses Feld nur gesetzt wird sollte der Wert nicht 0, also nicht leer, sein.

```

if(newEmployeeData.getActive() != 0){
    emp.setActive(newEmployeeData.getActive());
}
  
```

Abbildung 26: Ausschnitt Patch Methode – Target State

Variablen die in dem gesendeten JSON Objekt nicht definiert wurden, werden vom Rest Service automatisch mit dem Wert null oder 0 für Integer Variablen versehen.

Der Wert 0 konnte dementsprechend nicht für False verwendet werden, da somit die genannte Fehlerüberprüfung entfallen würde und der Active Status immer auf 0 gesetzt werden würde auch wenn diese Angabe vom User nicht gemacht wurde. Somit wurde von den Entwicklern der Wert 1 für True festgelegt und der Wert 2 für False.

4.8. Eingabeüberprüfungen im Service

Zur Sicherstellung korrekter Datensätze kontrolliert der Service Benutzereingaben für gewisse Attribute. Im Folgenden werden die Attribute der Entitäten kurz erklärt.

4.8.1. Employee Tabelle/Entity:

- **employeeid:** Einzigartige ID (Primärschlüssel) eines Employees, wird vom Service automatisch vergeben und kann nicht verändert werden
- **SVNR/svnr:** Darf nur eine exakt 10 Stellen lange Nummer sein, dieses Feld ist verpflichtend
- **Active:** Aktivitätsstatus eines Mitarbeiters. Darf entweder den Wert 1 oder 2 erhalten, dieses Feld ist verpflichtend
- **first_name/firstName:** Darf nur aus Buchstaben (A-Z) bestehen und beginnt mit einem Großbuchstaben, dieses Feld ist verpflichtend. Doppelnamen sind möglich und können über einen White Space, Bindestrich oder Punkt voneinander getrennt werden
- **last_name/lastName:** Darf nur aus Buchstaben (A-Z) bestehen und beginnt mit einem Großbuchstaben, dieses Feld ist verpflichtend. Doppelnamen sind möglich und können über einen White Space, Bindestrich oder Punkt voneinander getrennt werden
- **start_date:** Das Eintrittsdatum eines Mitarbeiters muss vor dessen Austrittsdatum (bei Angabe) sein. Dieses Feld ist verpflichtend
- **end_date:** Austrittsdatum eines Mitarbeiters. Muss bei der Erstellung nicht angeben werden, muss bei Eingabe allerdings nach dem Eintrittsdatum sein
- **login_name/loginName und password:** Sind grundsätzlich nicht verpflichtend, müssen bei der Eingabe allerdings aus mindestens einem Buchstaben bestehen, somit werden Eingaben welche nur aus White Spaces bestehen unterbunden. Sollte keine Eingabe oder eine Eingabe bestehend aus reinen White Spaces erfolgen wird automatisch ein Loginname und ein Passwort generiert. Dies wird in Punkt 4.9.1 genauer erläutert. Bei der Veränderung der Zugangsdaten eines bestehenden Mitarbeiters sollen diese Felder nicht leer sein. Sollte das Passwort Feld leer sein so wird das Passwort nicht verändert und es bleibt das bestehende Passwort im Datensatz. Eine leere Veränderungsangabe des Loginnamen eines Mitarbeiters wird durch den Service mithilfe einer BadRequest Exception unterbunden
- **last_changed/lastChanged:** Nicht durch den Benutzer befüllbar
- **department_id/departmentId:** Muss ein vorhandenes Department (Fremdschlüssel) referenzieren, nur numerische Eingaben möglich

4.8.2. Targetconfig Tabelle/Entity:

Sämtliche Eingaben sind auf die Werte 1, für True, und 2, für False beschränkt. Die Kombination einer Targetconfig ist einzigartig und 1 Kombination kann somit nur einmal in der DB vorhanden sein.

4.8.3. Targetstate Tabelle/Entity:

- Das Feld „Active“ darf entweder den Wert 1 oder 2 erhalten.
- Das Feld „Active“ wird bei Erstellung eines Targetstates auf 1 (true) gesetzt.
- Das Feld „Active_since“ und „Last_updated“ werden automatisch gesetzt.
- Das PostMapping der Targetstate Tabelle differenziert je nach Typ (MYSQL, CSV, LDAP oder JSON) zwischen verschiedenen Eingabeüberprüfungen. Als Beispiel werden bei JSON nur der Pfad, die Config-ID und die Department-ID benötigt, bei MYSQL jedoch andere Werte wie z. B. Port und Username
- Fremdschlüssel: Targetconfig-ID bezogen auf die Targetconfig Tabelle und Department-ID bezogen auf die Department Tabelle.

4.9. Automatisch durch den Service vergebene Werte

Der Rest Service vergibt sämtliche IDs für die Tabellen automatisch in Form einer fortlaufenden Nummer. Hierbei werden die IDs für die Employee, Department, Targetconfig und Targetstate Tabelle vom Service gehandhabt. Hierfür wurde der Sequenz Generator der Bibliothek Hibernate verwenden. Damit dies umgesetzt werden kann müssen die entsprechenden Variablen der Entitäten mit einer entsprechenden Annotation vermerkt werden, dies ist in der folgenden Abbildung ersichtlich.

```
//id wird über einen generator automatisch generiert und ist eine fortlaufende nummer
private @Id @GeneratedValue(strategy = GenerationType.AUTO, generator = "employeeid_sequence")
@SequenceGenerator(name = "employeeid_sequence", allocationSize=1)
int employeeid;
```

Abbildung 27: ID Generator

In dieser Abbildung wird die ID der Employee Entity an den in der „employeeid_sequence“ Tabelle gespeicherten Wert gekoppelt. Diese Tabelle beinhaltet die als nächstes zu vergebende ID. Hierbei befinden sich in der Datenbank eigenständige Sequenztabellen für die Employee, Department, Targetconfig und Targetstate Tabelle. Um diese kurz zu erwähnen werden diese nun kurz aufgelistet:

- **employeeid_sequence:** ID Generatortabelle für die Employee Tabelle
- **departmentid_sequence:** ID Generatortabelle für die Department Tabelle
- **targetid_sequence:** ID Generatortabelle für die Targetconfig Tabelle
- **targetstateid_sequence:** ID Generatortabelle für die Targetstate Tabelle

Der Service vergibt des Weiteren bei der Erstellung einer Entity für manche nicht durch den Benutzer befüllte Felder eigenständig Werte. Diese werden im Folgenden kurz aufgezählt.

4.9.1. Employee Entity/Tabelle

- **employeeid:** eine automatisch generierte einzigartige ID
- **login_name:** Sollte der Benutzer nicht selbst einen Login Namen definieren so wird dieser aus dem Anfangsbuchstaben des Vornamens, gefolgt von einem Unterstrich, dem Nachnamen und anschließend der ID generiert
- **password:** Sollte der Benutzer nicht selbst ein Passwort definieren so wird dieses aus dem Anfangsbuchstaben des Vornamens, gefolgt von einem Unterstrich, dem Nachnamen und anschließend der ID generiert
- **end_date:** Das Austrittsdatum wird bei der Erstellung eines Mitarbeiters auf null gesetzt, sollte dies nicht definiert werden
- **lastChanged/last_changed:** Nicht im UI ersichtlich und definiert wann der jeweilige Datensatz zuletzt geändert wurde, wird automatisch auf die Zeit der letzten Änderung gesetzt

4.9.2. Targetconfig Entity/Tabelle

- Targetconfig_id: eine automatisch generierte einzigartige ID

4.9.3. Targetstate Entity/Tabelle

- Active: Wird automatisch bei der Erstellung auf 1, also True gesetzt
- Activesince: Definiert seit wann ein Targetstate aktiv ist und wird automatisch auf diesen Zeitpunkt gesetzt
- Lastupdated: Definiert wann der jeweilige Datensatz zuletzt geändert wurde, wird automatisch auf die Zeit der letzten Änderung gesetzt
- Targetstate-ID: eine automatisch generierte, einzigartige ID.
- Last-synced: Dieser Wert wird ausschließlich vom jeweiligen Zielsystem vergeben.

4.9.4. Department Entity/Tabelle

- Department-ID: eine automatisch generierte, einzigartige ID.

4.10. Nicht auffangbare Falscheingaben

Im Folgenden werden kurz einige Falscheingaben erläutert, die durch den Service nicht abgefangen werden können.

4.10.1. Eingabe nicht vorhandener Fremdschlüssel

Einige Attribute der Tabellen sind Foreign Keys, so beinhaltet beispielsweise die Employee Tabelle das Attribut „department_id“ welches als Referenz auf einen Department Datensatz fungiert. Gibt der Benutzer hierbei eine nicht in der Department Tabelle vorhandene ID ein so liefert der Service hierbei einen 500er http Fehler. Dieser Umstand betrifft des Weiteren die Targetstate Tabelle mit Foreign Keys für eine Targetconfig und ein Department. Die Entwickler versuchten solche Falscheingaben im Service aufzufangen allerdings war hierbei eine saubere Umsetzung nicht möglich. Aus diesem Grund wurde der Verantwortliche für das UI darum gebeten für diese Felder in der UI ein Dropdownfeld zur Auswahl der entsprechenden Departments zu implementieren um dahingehenden Falscheingaben bereits im Front End zu verhindern. Der Verantwortliche für das UI verwies hierbei auf seine begrenzte Zeit und implementierte diese Umsetzung nicht.

4.10.2. Bearbeitung des Passwortes im UI

Ein weiterer Aspekt, der dem Rest Team im Zuge des Testens der UI Funktionalitäten aufgefallen ist, ist der Umstand, dass die Passwortfelder beim Bearbeiten von Mitarbeitern in der UI mit dem Hash Wert der Passwörter gefüllt werden. Dies ist hierbei ein Problem beim Bearbeiten von Mitarbeitern. So befindet sich im Eingabefeld für das Passwort standardmäßig der gehashte Wert des Passwortes. Hierbei wird dieser Hashwert bei der Bearbeitung eines neuen Mitarbeiters als neues Passwort angesehen und dementsprechend durch den Service neu gehashed, sollte im UI diese standardgesetzte Eingabe nicht entfernt werden. Der neu generierte Hashwert wird anschließend durch den Service in die Datenbank gespeichert. Der Verantwortliche für die UI wurde hierbei auf diesen Umstand hingewiesen und das Rest Team schlug ihm die Lösung vor dieses Bearbeitungstextfeld nicht mit dem in der Datenbank befindlichen gehaschten Passwort zu befüllen sondern leer zu lassen. Dies hat den Grund, dass der Service bei Nichteingabe eines neuen Passwortes kein neues Passwort setzt und dementsprechend das alte Passwort unverändert bleibt. Der Verantwortliche der UI konnte diese Lösung jedoch nicht umsetzen.

5. Technische Lösungsarchitektur Datenbank & Zielsysteme

5.1. Datenbank

Als Anforderung an die Datenbank bzw. das Datenbanksystem wurde eine konsistente Datenbasis für die Verwaltung von Mitarbeiterdaten bei Miterbeitereintritten, Abteilungswechsel von Mitarbeitern und Mitarbeiteraustritten definiert. Die Datenbank verwaltet Mitarbeiter mehrere Organisationen mit beliebigen Eigenschaften. Diese Mitarbeiter können erstellt (bei Miterbeitereintritten), bearbeitet (bei Abteilungswechsel des Mitarbeiters) und gelöscht (bei Austritt eines Mitarbeiters) werden.

Bei einem Eintritt eines Mitarbeiters werden die Mitarbeiterdaten über die GUI in die Datenbank eingegeben. Bei einem Abteilungswechsel eines Mitarbeiters wird der Benutzer in der alten Abteilung deaktiviert und ein neuer Benutzer für die neue Abteilung mit denselben Eigenschaften erstellt. Dadurch bleibt der Verlauf des Unternehmens- und der Abteilungsmitarbeiters erhalten und Abteilungswechsel von Mitarbeiter bleiben dadurch nachvollziehbar. Außerdem wird dadurch verhindert, dass veraltete Berechtigungen und Eigenschaften in die Zielsysteme übertragen werden.

Bei Austritt eines Mitarbeiters aus dem Unternehmen sollte dieser nicht aus dem System gelöscht werden, sondern lediglich auch inaktiv und dementsprechend ein Enddatum gesetzt werden. Das dient wiederum den Nachvollziehbarkeit. Durch automatisierte Vorgänge werden von der Datenbank ausgehend verschiedene Zielsysteme synchronisiert. Die Konfiguration der Zielsystemen ist in der Datenbank hinterlegt und können angepasst und auch erweitert werden. Dabei werden die Mitarbeiter der jeweiligen Abteilung zu allen aktiven Zielsystemen mit der Abteilung synchronisiert. Dabei bietet unsere Applikation den Export zu vier verschiedene Zielsysteme an (CSV-, JSON-Format, MYSQL-Datenbank und zu LDAP). Pro Zielsystem wurden zudem unterschiedliche Konfigurationsmöglichkeiten definiert, in denen bestimmt wird, welche Daten in welches Zielsystem exportiert werden sollen. Diese Konfigurationen kann manuell bearbeitet bzw. erweitert werden und unterstützen somit die Flexibilität für die Exporte in die unterschiedlichen Zielsysteme.

5.1.1. Rahmenbedingungen der Synchronisierung der Zielsysteme:

Die Synchronisierung ist lediglich unidirektional vorgesehen. Damit gesagt ist die zentrale Datenbank immer die „Wahrheit“. Jede Änderung der zentralen Datenbank, wenn relevant, wird auf das konfigurierte Zielsystem synchronisiert. Dabei ist zu beachten, dass nur Mitarbeiter synchronisiert werden, welche nach der letzten Synchronisierung bearbeitet wurden. Es wird eine korrekte Konfiguration vorausgesetzt. Das bedeutet, wenn in der „targetconfig“ einer zugehörigen „targetstate“ Konfiguration Felder aktiviert sind die nicht auf zum Beispiel der MYSQL Ziieldatenbank übertragen werden können, da diese Felder in der angegebenen Tabelle nicht verfügbar sind, wird keine Synchronisierung durchgeführt.

Eine stets korrekte Konfiguration wird vorausgesetzt und auch soweit möglich in dem zur Verfügung gestellten Frontend limitiert. Die Identitätsdatenbank (central_db_t3) wurde mit der Open Source Lösung „MYSQL“ umgesetzt und setzt sich aus folgenden Tabellen und Stored Procedures zusammen, die nun im Detail beschrieben werden.

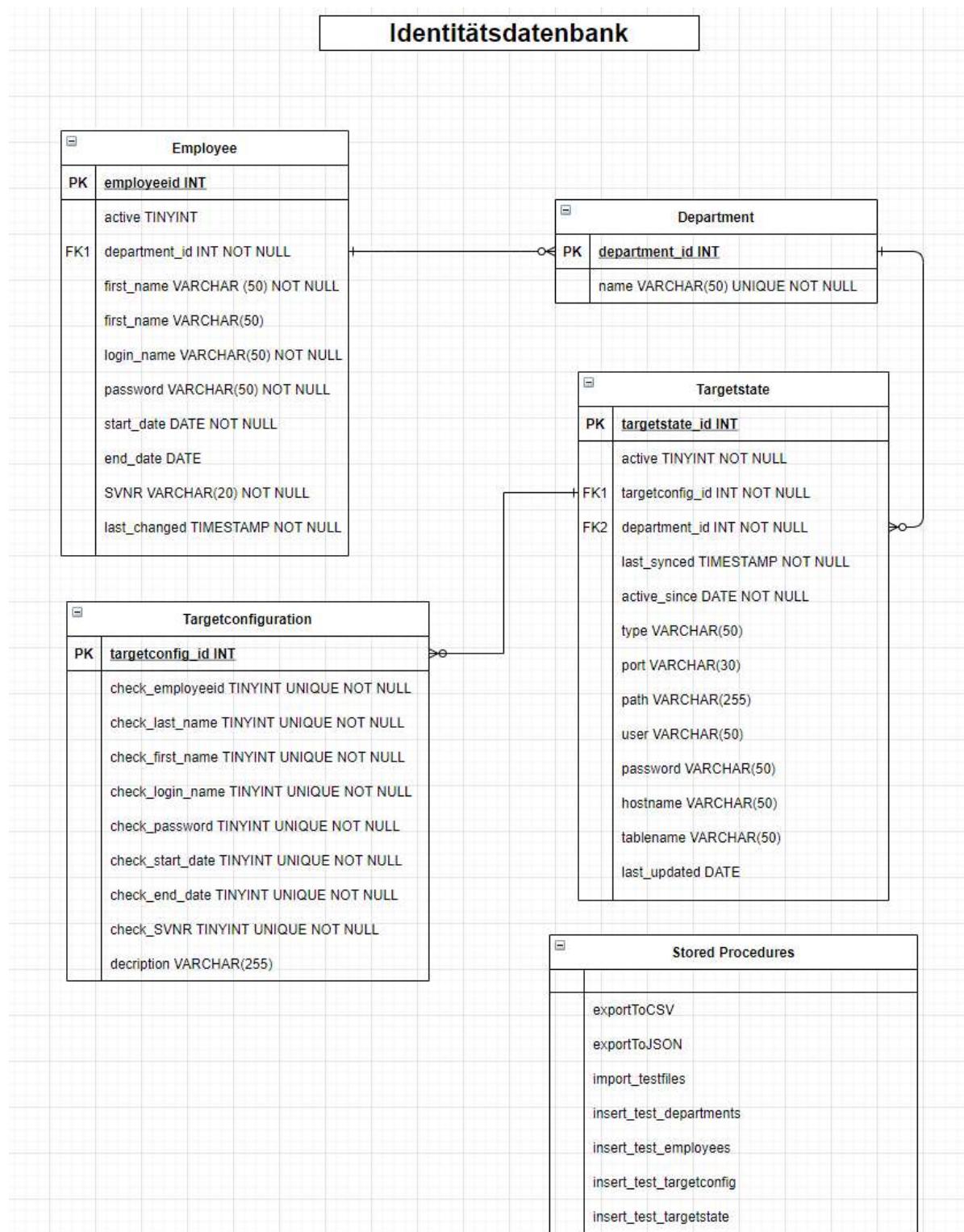


Abbildung 28: Tabellenschema

5.1.2. Tabelle „employee“

In dieser Tabelle befinden sich alle notwendigen Mitarbeiterinformationen, als Primärschlüssel wird die fortlaufende „employeeid“ verwendet. In dieser Tabelle müssen bis auf „first_name“ und „end_date“ alle Felder befüllt werden.

Column	Type	Nullable	Indexes
employeeid	int	NO	PRIMARY
active	tinyint	NO	
last_name	varchar(50)	NO	
first_name	varchar(50)	YES	
login_name	varchar(50)	NO	
password	varchar(50)	NO	
start_date	date	NO	
end_date	date	YES	
department_id	int	NO	FK_department
SVNR	varchar(20)	NO	
last_changed	timestamp	NO	

Abbildung 29: Tabelle "employee"

5.1.3. Tabelle „department“

Hier werden die Informationen der jeweiligen Abteilung gespeichert, der Primärschlüssel wurde auf das Feld „department_id“ gesetzt. Das Feld „name“ beinhaltet die Bezeichnung der Abteilung und wurde auf UNIQUE gesetzt, da keine identischen Departments angelegt werden sollen.

name	varchar(50)	NO	name_UNIQUE
------	-------------	----	-------------

Abbildung 31: Tabelle "department"

5.1.4. Tabelle „targetconfig“

In dieser Tabelle wird definiert, welche Mitarbeiterdaten an die Zielsysteme exportiert werden, dabei wird in den „check“-Feldern mittels Tinyint (1 = True/2 = False) der gewünschte Wert gesetzt. Diese Konfiguration kann anschließend in der Tabelle „targetstate“ ausgewählt werden. Um zu verhindern, dass doppelte Einträge angelegt werden, wurde die Kombination der „check“-Felder auf UNIQUE gesetzt. Des weiteren existiert das Feld „description“ in der die Beschreibung der Konfiguration gespeichert ist.

Column	Type	Nullable	Indexes
targetconfig_id	int	NO	PRIMARY
check_employeeid	tinyint	NO	check_UNIQUE
check_last_name	tinyint	NO	check_UNIQUE
check_first_name	tinyint	NO	check_UNIQUE
check_login_name	tinyint	NO	check_UNIQUE
check_password	tinyint	NO	check_UNIQUE
check_start_date	tinyint	NO	check_UNIQUE
check_end_date	tinyint	NO	check_UNIQUE
check_SVNR	tinyint	NO	check_UNIQUE
description	varchar(255)	YES	

Abbildung 32: Tabelle "targetconfig"

5.1.5. Tabelle „targetstate“

Diese Tabelle dient zur Konfiguration der Zielsysteme, dabei ist zu beachten, dass je nach Zielsystem einige Felder als Pflichtfelder definiert wurden. Als Primärschlüssel dient das Feld „targetstate_id“. Der Timestamp „last_synced“ wird dabei auf den Zeitpunkt der Ausführung des Exports in das Zielsystems gesetzt, dadurch lässt sich nachvollziehen, wann der Export in das jeweilig konfigurierte Zielsystem durchgeführt wurde. Der Timestamp „active_since“ wird gesetzt, sobald eine Konfiguration aktiv gesetzt wurde (entweder direkt bei der Anlage der targetstate, wenn er im selben Zug auf aktiv gesetzt wird, oder wenn es sich um eine inaktiven Datensatz handelt, bei dessen Aktivierung). Der Timestamp „last_updated“ wird wiederum gesetzt, wenn eine Änderung in der jeweiligen Targetstate durchgeführt wurde.

Im Feld „type“ wird das jeweilige Zielsystem eingetragen (CSV, JSON, LDAP oder MYSQL). Abhängig von dieser Auswahl sind anschließend die Felder „port“, „path“, „user“, „password“, „hostname“ und „tablename“ jeweils Pflichtfelder.

Dabei ist anzumerken, dass pro Abteilung (Feld „department_id“) und ausgewählter Konfiguration der zu exportierenden Mitarbeiterfelder (Feld „targetconfig_id“) mehrere Zielsysteme (Feld „type“) ausgewählt werden können.

Zum Beispiel für die Abteilung „Sales“ dieselben Mitarbeiterdaten (die über die Tabelle targetconfig gesetzt wurden) mehrere Zielsysteme definiert werden.

Die Felder „port“, „path“, „user“, „password“, „hostname“ und „tablename“, werden zur Konfiguration der Zielsysteme verwendet und werden in der Eingabe durch das Frontend beschränkt welche dieser Felder für den gewählten Zieltyp benötigt werden.

Zum Beispiel benötigt das Zielsystem MYSQL alle dieser Felder außer „tablename“ und CSV nur das „path“ Feld, die Erstellung einer Targetstate wird dementsprechend auch so limitiert. Die Nutzung der benötigten Felder wird in dem Zielsystem weiter beschrieben.

Column	Type	Nullable	Indexes
targetstate_id	int	NO	PRIMARY
active	tinyint	NO	
targetconfig_id	int	YES	FK_targetstate_central_db_t3.targetconfig
last_synced	timestamp	NO	
active_since	date	NO	
department_id	int	NO	FK_Targetstate_department
type	varchar(50)	NO	
port	varchar(30)	YES	
path	varchar(255)	YES	
user	varchar(50)	YES	
password	varchar(50)	YES	
hostname	varchar(50)	YES	
tablename	varchar(50)	YES	
last_updated	date	YES	

Abbildung 33: Tabelle "targetstate"

5.2. Zielsysteme

5.2.1 CSV und JSON

Die Zielsysteme „CSV“ und „JSON“ benötigen nur das Konfigurationsfeld „Path“ aus der Tabelle „targetstate“, hier wird Pfad angegeben, auf dem die Daten abgelegt werden sollen.

Die Exporte für die Zielsysteme JSON und CSV wurden in der Datenbank über Stored Procedures umgesetzt und über zeitlichen Events in der Datenbank getriggert. Dafür ist das „Path“ Feld der „targetstate“ Tabelle notwendig, darüber hinaus müssen die Felder „department_id“, „tagetconfig_id“ und „active“ gesetzt werden, um die zu exportierenden Konfigurationen daraus ermitteln zu können.

Innerhalb der Stored Procedure wird eine temporäre Tabelle erstellt und in diese die Mitarbeiterdaten geladen, hierbei werden die Einstellungen in der Tabelle „targetconfig“ berücksichtig und nur die in dieser Tabelle definierten Daten auf den angegebenen Pfad („Path“ aus „targetstate“) im jeweiligen Datenformat für die jeweilig definierte Abteilung exportiert.

Das bedeutet zum Beispiel, wenn in der Tabelle „targetconfig“ definiert wurde, dass alle Mitarbeiterdaten der Abteilung „Sales“ für das Zielsystem „CSV“ exportiert werden sollen, wird dies in den Stored Procedures geprüft und die Daten der Sales-Mitarbeiter anschließend auf das entsprechende Laufwerk abgelegt. Dabei wird durch folgende Logik in den Stored Procedures verwendet.

Sobald ein neuer Mitarbeiter eingefügt oder verändert wurde, wird in der Tabelle „employee“ in das Feld „last_changed“ der aktuelle Timestamp gesetzt, in der Tabelle „targetstate“ gibt es das Feld „last_synced“, dass bei jedem Durchlauf der Synchronisation ebenfalls einem Timestamp setzt. Sobald das Feld „last_changed“ in der Employee-Table zeitlich vor dem Feld „last_synced“ in der Targetstate-Table liegt, wird dieser Mitarbeiter in das Zielsystem exportiert, ansonsten nicht.

Dieselbe Funktionalität bietet aus das Zielsystem „JSON“, hierbei werden die Mitarbeiterdaten in ein JSON-Format exportiert.

Damit der Datenbankservice auch die benötigten Schreibrechte besitzt, um auf die ausgewählten Pfade die Daten der Zielsysteme abzulegen, muss unter dem jeweiligen Betriebssystem für den Benutzer „Netzwerkdienst“ die Schreibrechte definiert werden (hier als Beispiel unter Windows 10):

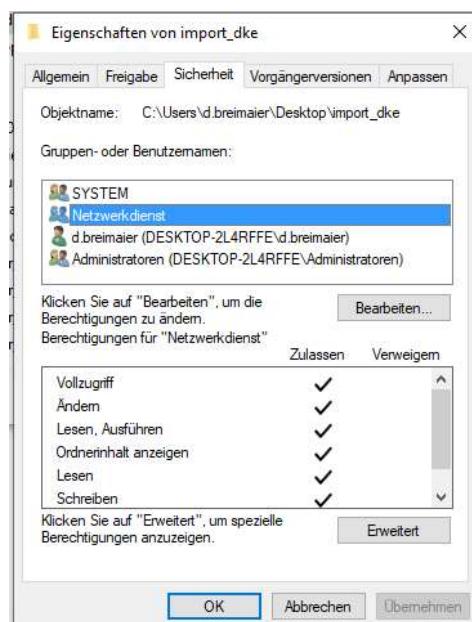


Abbildung 34: Sicherheitseinstellungen unter Windows 10

Außerdem muss auf dem Datenbankserver die Einstellung „secure -file -priv“ deaktiviert bzw. auf NULL gesetzt werden um die Funktion „SELECT INTO OUTFILE“, die innerhalb der Stored Procedures verwendet wird, verwenden zu können.



Abbildung 35: secure-file-priv Einstellung

5.2.1. Events

Die Stored Procedures „exportToCSV“ und „exportToJson“ werden über in der Datenbank definierte Events ausgeführt, da die Ausführung von Triggern durch Update- und Insert-statements in der Datenbank aus Sicherheitsgründen deaktiviert wurde. Diese Events werden automatisch nach einem vorher definierten Zeitintervall ausgeführt (aus Demonstrationszwecken in diesem Fall alle 5 Minuten). In einer Produktivumgebung sollte dieses Intervall deutlich erhöht werden, um die Datenbankauslastung nicht unnötig zu erhöhen.

Db	Name	Definer	Time zone	Type	Execute at	Interval value	Interval field
central_db_t3	autom_export_CSV	root@localhost	SYSTEM	RECURRING	NULL	5	MINUTE
central_db_t3	autom_export_JSON	root@localhost	SYSTEM	RECURRING	NULL	5	MINUTE

Abbildung 36: Events in MySQL

Um das Zeitintervall der automatisierten Exporte für die CSV und JSON Dateiformate zu ändern, muss folgender SQL-Code einfach neu im DBMS ausgeführt werden, dabei muss das gewünschte Zeitintervall angegeben werden.

```

DROP EVENT IF EXISTS `autom_export_CSV`;
CREATE EVENT autom_export_CSV
ON SCHEDULE EVERY 5 MINUTE -- Hier das Zeitintervall einstellen
STARTS CURRENT_TIMESTAMP
DO
CALL `central_db_t3`.`exportToCSV`();

;

DROP EVENT IF EXISTS `autom_export_JSON`;
CREATE EVENT autom_export_JSON
ON SCHEDULE EVERY 5 MINUTE -- Hier das Zeitintervall einstellen
STARTS CURRENT_TIMESTAMP
DO
CALL `central_db_t3`.`exportToJson`();
;
```

Abbildung 37: Erstellung MySQL Events

5.2.2. LDAP

Das Zielsystem LDAP benötigt folgende Konfigurationsfelder aus der Tabelle „targetstate“:

- **Port:** der Port an dem der LDAP Service läuft, zum Beispiel: „389“
- **Path:** der Pfad in den die Daten geschrieben werden sollen, zum Beispiel: „ou=People,dc=dkePr,dc=at“
- **User:** Der Username mit welchem die Dateien in den LdapServer geschrieben werden sollten (benötigt genügend Rechte), zum Beispiel: „cn=Manager,dc=dkePr,dc=at“
- **Password:** Das Passwort des Users, der verwendet wird um Daten in den LdapServer zu schreiben, zum Beispiel: „secret“
- **Hostname:** Der Host Name oder die IP Adresse des Host auf welchem der LdapServer läuft, zum Beispiel: „localhost“

Grundsätzlich ist es möglich verschiedenste LdapServer zu synchronisieren, jedoch ist die Synchronisierung abhängig von dem Mapping für den jeweiligen LdapServer.

Als Schnittstelle zwischen Applikation und LDAP-Server wird die Open Source-Software „OpenLDAP“ verwendet, um einen lokalen Ldap Server zur Verfügung zu stellen und diesen mit der Datenbank zu synchronisieren. Es kann entweder der fertig konfigurierte OpenLdap Server gestartet werden oder eine neue Konfigurierung des LDAP Servers durchgeführt werden.

5.2.3. MYSQL

Das Zielsystem MYSQL benötigt folgende Konfigurationsfelder aus der Tabelle „targetstate“:

- **Port:** der Port unter welchem auf die Zieldatenbank zugegriffen werden kann, zum Beispiel: „389“
- **Path:** der Pfad zu der gewünschten Datenbank in welche die Daten geschrieben werden sollen (exklusive dem Tabellennamen) auf dem Host, zum Beispiel: „/central_db_t2“
- **User:** Der Username mit welchem die Dateien in die Datenbank geschrieben werden sollten (benötigt genügend Rechte), zum Beispiel: „DKETeam“
- **Password:** Das Passwort des User, der verwendet wird um Daten in die Datenbank zu schreiben, zum Beispiel: „mydkepassword“
- **Hostname:** Der Host Name oder die Ip Adresse des Host auf welchem die Datenbank läuft, zum Beispiel: „dkeprw21.mysql.database.azure.com“ oder localhost
- **Tablename:** Der Tabellenname in welche die Daten geschrieben werden sollen, zum Beispiel: „employee“

Um MYSQL zu synchronisieren, muss nur dieselbe Synchronisation gestartet werden wie für LDAP und die jeweilige Konfiguration in der zentralen Datenbank eingetragen und auf aktiv gesetzt werden. Dabei ist zu beachten, dass die zugehörige Targetconfig auch zu der konfigurierten Tabelle des Ziel MySQL Servers passt

5.2.4. Synchronisierung starten

Bevor die Main Methode der Synchronize_DKE.java gestartet und damit auch die Synchronisation durchgeführt wird, sollte die Datenbankkonfiguration der Identitätsdatenbank (central_db_t3) geprüft und eventuell abgeändert werden. Ebenfalls sollte überprüft werden, ob die Schnittstelle zu Ldap aktiv ist mit dem Parameter „ldapConfigured“.

Im Quellcode wurde die Möglichkeit definiert, entweder die cloudbasierte Datenbank zu nutzen (in der allerdings die Zielsysteme für CSV und JSON aus Sicherheitsgründen nicht aktivierbar sind) oder die lokale Datenbank.

Dementsprechend muss der jeweilige Anwendungsfall im Quellcode aktiviert werden bzw. bei der Verwendung der lokalen Datenbank den Usernamen und das Passwort dementsprechend abgeändert werden.

Sobald die Main Methode läuft, wird alle 5 Minuten auf Änderungen in der „Employee“ Tabelle geprüft und wenn eine Änderung erkannt wurde, werden auch die Zielsysteme aktualisiert. Achtung: Es ist nicht vorgesehen, dass ein Employee aus der zentralen Datenbank gelöscht wird, es ist lediglich vorgesehen, dass dieser auf Inaktiv gesetzt wird („is active“ != 1). Erst nachdem alle Zielsysteme diese Änderung synchronisiert bekommen haben, kann auch der Employee aus der Datenbank endgültig gelöscht werden (nicht vorgesehen)

5.2.5. Funktionsprüfung/Testdaten

Um die Applikation zu testen, wurden einige Testdaten mitgeliefert, um die Funktionalität der Applikation zu veranschaulichen. Um diese Daten zu importieren, muss innerhalb des DBMS die Stored Procedure „import_testfiles“ mit dem Aufruf „call import_testfiles()“ ausgeführt werden (diese ruft wiederum die Stored Procedures „test_insert_departments“, „test_insert_employees“, „test_insert_targetconig“ und „test_insert_targetstate“ auf. Durch dessen Aufruf werden in den vorhandenen Tabellen die Testdaten eingefügt. Anzumerken ist hierbei, dass diese Testdaten direkt in die Datenbank geschrieben werden, das bedeutet die Mitarbeiterpasswörter werden nicht verschlüsselt und im Klartext gespeichert. Da es sich dabei um fiktive Mitarbeiterdaten handelt ist dieser Aspekt nicht weiter relevant, da bei der Eingabe der Mitarbeiter über die Applikation die Passwörter verschlüsselt werden. Des Weiteren ist anzumerken, dass die Testdaten für die Zielsysteme Exportpfade, Logindaten und Ports beinhalten, die möglicherweise auf andere Server/Clients nicht vorhanden sind, daher müssen diese dementsprechend manuell angepasst werden.

6. Anmerkungen zu den eingesetzten Software Artefakten, Technologien & Tools

6.1. GitHub

6.1.1. Frontend-Team

Initial wurde auf <https://github.com/galvadino/enterprise-db-app> ein Repository für das gesamte Team angelegt. Im späteren Verlauf entschied sich das Rest Service Team die Umsetzung statt in einem separaten Ordner in einem separaten Projekt fortzuführen.

6.1.2. Rest-Service-Team

Das Rest Service Team entschied sich dazu ein eigenständiges, vom Frontend unabhängiges Repository für die Verwaltung des Codes zu verwenden. Dadurch konnten Arbeiten am Code unabhängig vom UI durchgeführt und Codekonflikte vermieden werden.

Des Weiteren kann durch die Verwendung eines eigenständigen Repositories die Commit Historie des Rest Teams leichter nachvollzogen werden. Diese ist hierbei im folgenden Repository ersichtlich: <https://github.com/DKEProjekt2021/DKE-Rest>. Für die Abgabe und Installation des Projektes wurde der fertig implementierte Rest Service in das Gesamtrepository der Gruppe übersiedelt welches unter dem Link in Punkt 6.1.1 aufrufbar ist.

Da beide Teammitglieder schon Erfahrungen mit der Verwendung von GitHub haben konnte dieses Tool leicht in die Projektstruktur des Rest Teams implementiert werden.

6.2. Angular

Als Front-End Technologie wird Angular 13, aufgrund von bestehenden Erfahrungen und der steigenden Popularität im Bereich des Web Development genutzt.

6.3. Entwicklungsumgebungen

Für die Umsetzung des Front-End in Angular wurde Visual Studio Code genutzt.

6.4. Spring Initializer Rest Team

Wie bereits erwähnt verwendete das Rest Team den Spring Initializer für die Erstellung des Restservice-Java Projekts. Mit dem Spring Initializer konnte das Rest Team unkompliziert die für das Projekt benötigten Dependencies auswählen und der Spring Initializer erstellte automatisiert ein Java Projekt mit der benötigten Struktur. Somit war der Spring Initializer ein wertvolles Software Artefakt und reduzierte die Zeit die das Team investieren musste um benötigte Software Dependencies zu finden.

6.5. Rest Service

Die Verwendung eines Rest Services als Schnittstelle zwischen dem Front End und der Datenbank stellte für das Rest Team die richtige Entscheidung dar. So konnte dieses durch die Verwendung eines Rest Service alle benötigten Anforderungen implementieren und des Weiteren noch verschiedenste Fehlerüberprüfungen und andere Mechaniken implementieren. Der generelle Aufbau eines Rest Service half uns zudem dabei unseren Code übersichtlich zu gestalten. So sind die Aufgabenbereiche der einzelnen Klassen klar feststellbar und die einzelnen Klassen können einfach kategorisiert werden. Fehler im Programmcode konnten somit relativ schnell bestimmten Klassen zugeordnet werden und dementsprechend schnell behoben werden. Des Weiteren konnten mit dem Rest Service http Statuscode zu den Fehlermeldungen hinzugefügt werden.

6.6. Hibernate/JPA Rest Team

Die JPA/Hibernate Bibliothek wurde vom Rest Team dafür verwendet um Repräsentationen der benötigten Tabellen im Java Code zu implementieren. Hierbei handelt es sich um die bereits erläuterten Entitätsklassen. Des Weiteren wurde die Hibernate Bibliothek dazu verwendet um für die entsprechenden Tabellen fortlaufende IDs als Primärschlüssel zu generieren. Hibernate/JPA ermöglichte es dem Rest Team somit die Informationen innerhalb der Tabellen auf Java Objekte zu mappen und erlaubte die Manipulation und Erstellung von Datensätzen über JSON Objekte. Somit stellt die Hibernate/JPA Bibliothek einen fundamentalen Teil unserer Projektstruktur dar.

6.7. Postman Rest Team

Das Rest Team verwendete die API Plattform Postman zur Überprüfung der Funktionalitäten des Rest Service. Hierbei ermöglichte Postman dem Rest Team eine schnelle und unkomplizierte Kontrolle der Funktionsfähigkeit der einzelnen Service Methoden. Des Weiteren konnte das Rest Team dank Postman einfach kontrollieren ob bei Falscheingaben durch den Benutzer die richtigen HTTP Statuscodes vom Service gemeldet werden.

6.8. MySQL-Datenbank

Da bereits ein Großteil des Teams Erfahrungen mit der quelloffenen MySQL Datenbank hatte, wurde diese auch als Datenbanktechnologie ausgewählt. Für die Anlage und Bearbeitung der Datenbank(en) und der verschiedenen Tabelle, Stored Procedures, etc. wurde die MySQL Workbench genutzt.

6.9. Zielsysteme

6.9.1. CSV und JSON

Die für die Exporte der Zielsysteme „CSV“ und „JSON“ verwendeten Stored Procedures werden direkt in der zentralen Datenbank verwaltet und sind daher leichter zu adaptieren, außerdem sind sie performanter als eigens geschriebene Methoden in der Applikation. Des Weiteren verringern sie die Kommunikation zwischen Server und Applikation auf ein Minimum. Allerdings muss geprüft werden, ob die Zugriffsberechtigung für den Netzwerkdienst der Datenbank, wie eingangs beschrieben, auf allen notwendigen Ablageorte eingestellt wurde. Außerdem sind einige Funktionen der Stored Procedures (wie zum Beispiel das Feature „SELECT * INTO OUTFILE“) innerhalb cloudehosteter Datenbanken deaktiviert (im Fall der Azure Cloud).

6.9.2. OPENLDAP

Der größte Vorteil von OpenLDAP ist die einfache Installation und Aufsetzung des Servers im Lokalen Umfeld. Besonders LDAP Server sind meist kostenpflichtig und benötigen besondere Hardwareausführungen wodurch das Testen im kleinen Rahmen wie dieses Projektes deutlich schwieriger gestaltet hätte. Durch OpenLDAP war es möglich Einblicke in die Grundstrukturen von LDAP zu erhalten wodurch das Verständnis viel detaillierter ausfällt. Durch die Nutzung des bekannten Apache Directory Browser konnte das Wissen der Serverfunktionen mit der mehr bekannten Nutzererfahrung in Verbindung gebrachte werden.

6.9.3. MYSQL

Aufgrund des Bekanntheitsgrads von MYSQL sind für die meisten möglichen Konfigurationen und Sonderbedingungen bereits Erfahrungen im Netz zu finden. Besonders von Vorteil ist auch, dass durch ein Studentenkonto eine Cloud Datenbank sehr einfach aufgesetzt werden kann und dadurch perfekt von überall aus gleichzeitig mit dieser arbeiten zu können. Ein Nachteil von MYSQL wurde bei der Zusammenarbeit erkannt, dass MYSQL Windows benötigt und nicht verlässlich unter UNIX Systemen läuft. Außerdem bietet MYSQL eine besonders hohe Kompatibilität mit verschiedensten Systemen, wodurch eine deutlich größere Auswahl entsteht im Gegensatz zu Alternativen. Dies gilt sowohl für die zentrale Datenbank als auch für das Zielsystem.

7. Installations- & Konfigurationsanleitung

7.1. Datenbank und Zielsysteme

Download des gesamten Projektes von GitHub <https://github.com/galvadino/enterprise-db-app>

Die Identitätsdatenbank (central_db_t3) sowie die Zieldatenbank (mysqltarget) befinden sich unter „Datenbank-Zielsysteme/Datenbank_DKE“ und können einfach in ein MySQL DBMS importiert werden.

Unter dem Ordner „Datenbank-Zielsysteme“ befinden sich der Konfigurierte LDAP Server „OpenLDAP“, der LDAP Browser Apache Directory Studio, der SynchronisierungsService als Java Projekt und Konfigurationsdateien für einen selbstkonfigurierten LDAP Server.

Es kann entweder der fertig konfigurierte OpenLdap Server gestartet werden oder eine neue Konfigurierung des LDAP Servers durchgeführt werden.

Konfigurierten LDAP Server starten:

Die Zip Datei „OpenLDAP.zip“ auf einem Windows Gerät entpacken. In dem entpackten Ordner unter „run“ die Batch Datei run.bat ausführen.

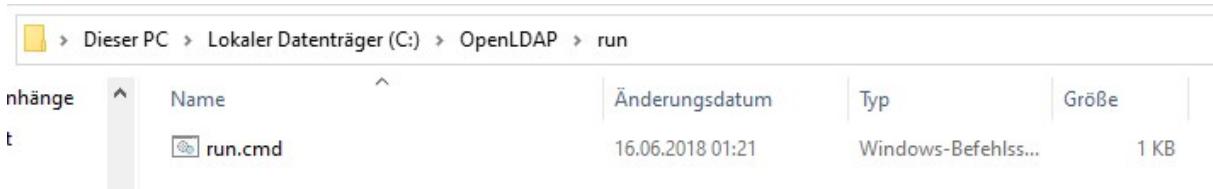


Abbildung 38: Start des OpenLdap Services

Damit wird der LDAP Server local unter dem Port 389 gestartet und kann mit jedem beliebigen Ldap Browser verbunden werden.

Alternativ: Neuinstallation des OpenLdap Servers:

OpenLdap kann kostenlos für Linux unter : <https://www.openldap.org/software/download/> , für Windows unter : <https://www.maxcdn.de/download/> heruntergeladen werden. Nach dem installieren laut der Anleitung für Linux: <https://www.openldap.org/doc/admin24/quickstart.html> , für Windows : <https://www.maxcdn.de/openldap-for-windows-installation/> kann der Ldap-Server konfiguriert werden.

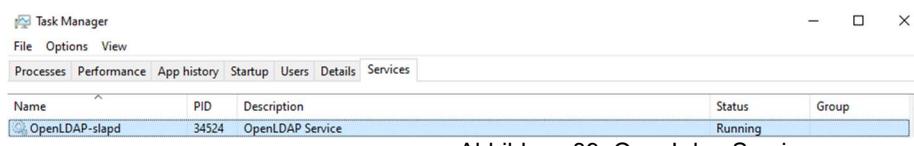


Abbildung 39: OpenLdap Service

Zuerst sollte die Slapd.conf bearbeiten in dem root Verzeichnis wo LDAP installiert wurde bearbeitet werden, um die gewünschte Root in der Datenbank zu erzeugen.

Ein Beispiel ist in dem Projekt beigelegt „slapd.conf“ und beinhaltet den datenbanktyp, den root Suffix, die rootDn, das rootPasswort, und diversen Imports.

Um diese Konfiguration zu übernehmen, sollte der Ldap Service neu gestartet werden (laut Betriebssystem den openldap-slapd service neu starten)

Danach kann eine Root Domain mit den Ldap Client Tools welche sich in dem gleichnamigen Ordner in der Ldap Installation befinden hinzugefügt werden. Um Einträge hinzuzufügen muss eine „.ldif Datei“ erstellt werden wo die neuen Einträge aufgelistet sind. Ein Beispiel ist in dem Projektordner beigelegt „dkePr.ldif“.

Diese Einträge der Datei kann danach mit einem Konsolenbefehl und den LdapClientTools eingefügt werden.

```
ldapmodify -a -x -D "cn=Manager,dc=<MY-DOMAIN>,dc=<COM>" -W <PASSWORD> -f example.ldif
```

konkret:

```
ldapmodify -a -D "cn=Manager,dc=dkePr,dc=at" -W secret -f dkePr.ldif
toolname add      Domain user with enough priv. u.password Path to file
```

Weitere Details ist der Schnellstartanleitung von OpenLdap zu entnehmen.

<https://www.openldap.org/doc/admin24/quickstart.html>

<https://www.openldap.org/software//man.cgi>

Damit ist der OpenLdap Server bereit und lässt sich zum Beispiel mit einem Ldap Browser nutzen.

Konfigurierten LdapServer nutzen:

Stellen sie sicher, dass entweder der Service (bei Neuinstallation) oder der Hintergrundprozess slapd.bat von Ldap (bei der Übernahme des Konfigurierten Servers) gestartet sind.

Damit ist es notwendig das Apache Directory Studio zu installieren und die Verbindung zwischen Applikation und LDAP-Datenbank zu erstellen. Eine Instanz dieses Browsers ist auch in den Projektordner enthalten. Dieser Browser kann durch Entpacken der heruntergeladenen ZIP Datei anschließendem ausführen der exe gestartet werden. Dieser ist auch für andere Betriebssysteme verfügbar unter: <https://directory.apache.org/studio/>

Sollte es beim Starten des Apache Directory Studio zu folgender Fehlermeldung kommen, ist wie folgt vorzugehen.

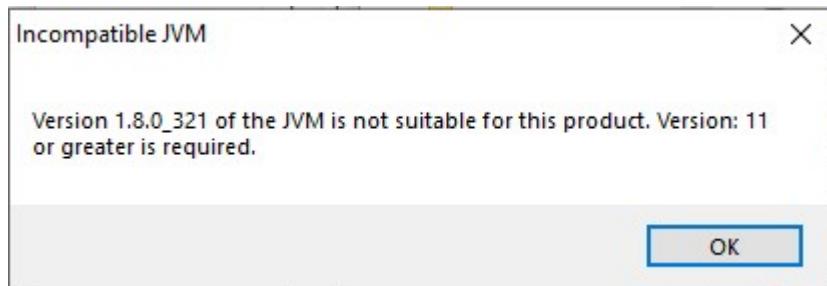


Abbildung 40: Fehlermeldung Apache Directory Studio

Im Installationsordner des Apache Directory Studio die Konfigurationsdatei mittels Editors öffnen.

Dieser PC > Lokaler Datenträger (C:) > Programme > Apache Directory Studio				
	Name	Änderungsdatum	Typ	Größe
	configuration	22.01.2022 14:33	Dateiordner	
	features	22.01.2022 14:33	Dateiordner	
	p2	22.01.2022 14:33	Dateiordner	
	plugins	22.01.2022 14:33	Dateiordner	
	ApacheDirectoryStudio.exe	17.07.2021 20:07	Anwendung	417 KB
	ApacheDirectoryStudio.ini	22.01.2022 14:39	Konfigurationsein...	1 KB
	artifacts.xml	17.07.2021 20:07	XML-Dokument	117 KB

Abbildung 41: Pfad zur zu ändernden Datei

Und wie folgt abändern (dabei muss der Installationspfad der aktuellen JDK angegeben werden).

```
"""
#Uncomment_to_configure_Java_version_to_use
#https://directory.apache.org/studio/faqs.html#how-to-set-the-java-vm-to-use
-vm
C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe
#/usr/lib/jvm/java-11-openjdk/bin/java

-vmargs
-Dosgi.requiredJavaVersion=11
###
```

Abbildung 42: Änderungen der Datei „ApacheDirectoryStudio.ini“

Dann muss eine Verbindung im Apache Directory wie folgt angelegt werden:

Schritt 1: Anlage einer Verbindung im Apache Directory Studio

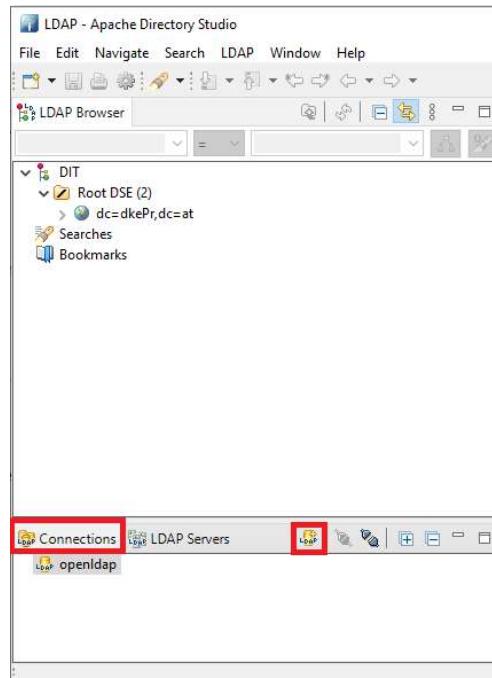


Abbildung 43: Hinzufügen einer neuen Verbindung

Schritt 2: Eingabe der Verbindungsdaten (da der Service lokal auf einem Rechner läuft muss als Hostname „localhost“ und als Portnummer „398“ angegeben werden).

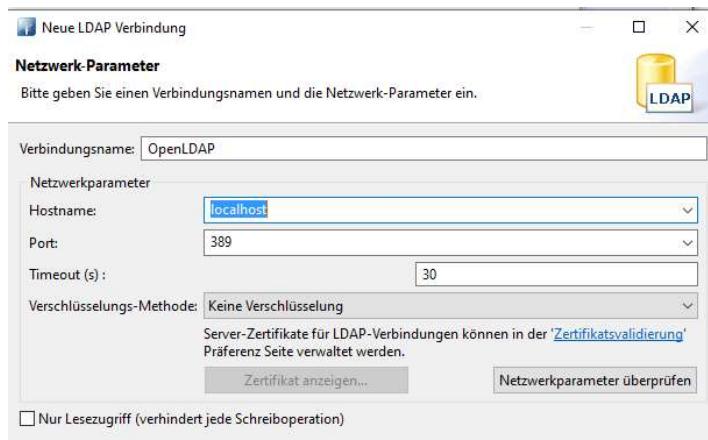


Abbildung 44: Eingabe der Verbindungsdaten

Schritt 3: Eingabe des Benutzers und des Passworts des LDAP Directory Services:

User: cn=Manager,dc=dkePr,dc=at

Password: secret

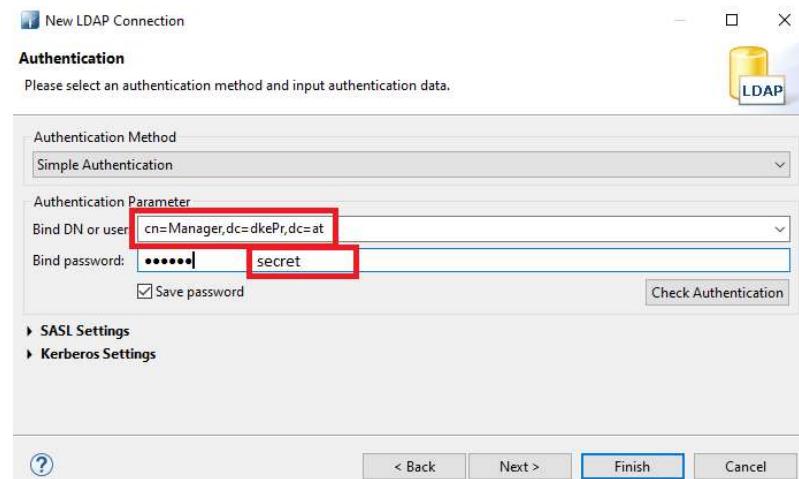


Abbildung 45: Eingabe der Authentifizierungsdaten

Nachdem die Verbindungsdaten eingegeben wurden, kann die Verbindung hinzugefügt werden und die Ldap Datenbank wird angezeigt.

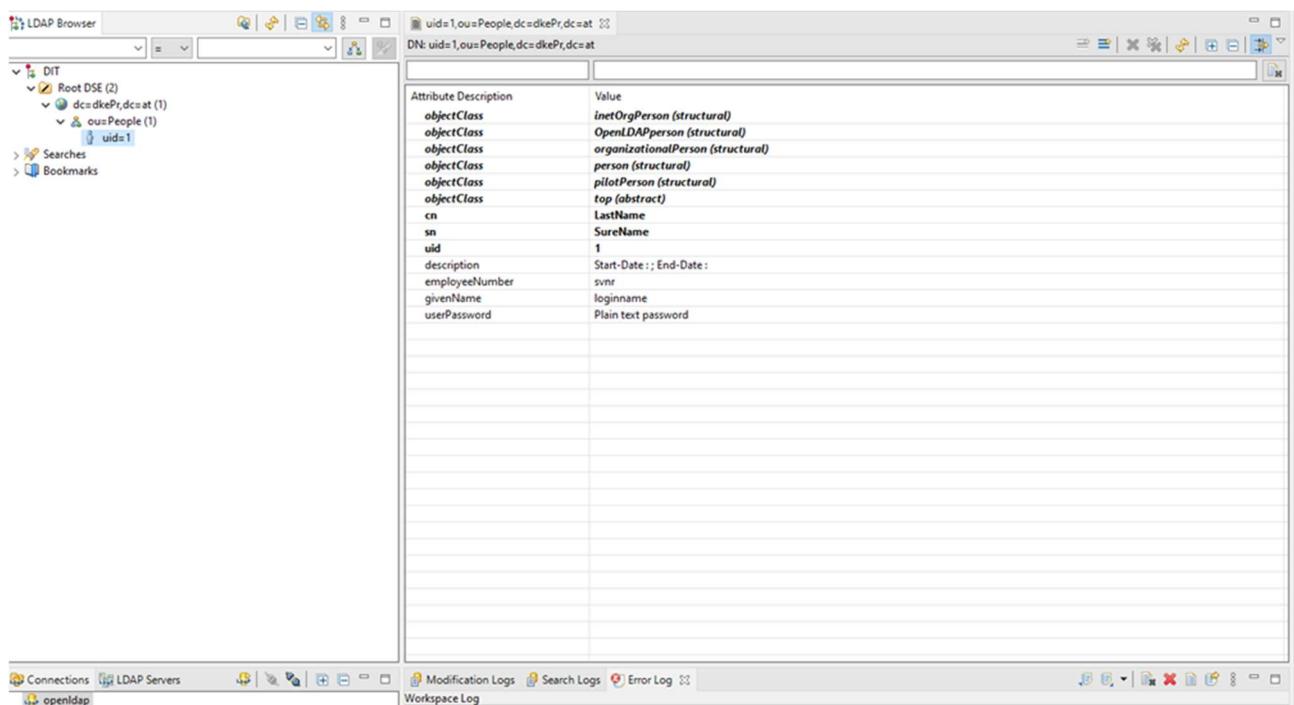


Abbildung 46:Verbundener LDAP Server + Struktur

In Abbildung 46 ist auch die verwendete Struktur der LDAP Datenbank zu sehen und die verwendeten Attribute wie die „Employees“ gespeichert werden.

Die Nutzerdaten werden als

- inetOrgPerson (structural),
- OpenLDAPperson (structural),
- organizationalPerson (structural),
- person (structural),
- pilotPerson (structural),
- top (abstract),

abgespeichert wobei die Einträge in der ou=People gespeichert werden und jeder Eintrag eine eindeutige „uid“ hat welche die employee_id darstellt (da diese auch in der Datenbank der Primary Key ist).

- „uid“: Mitarbeiternummer des Mitarbeiters „employee_id“
- „cn“: Nachname des Mitarbeiters „lastname“
- „sn“: Vorname des Mitarbeiters „firstname“
- „description“: Wird genutzt, um Start- und Enddatum des Mitarbeiters zu speichern „StartDate“+“EndDate“
- „employeeNumber“: die Svnr des Mitarbeiters „Svnr“
- „givenName“: Wird genutzt, um den Login Namen des Mitarbeiters zu speichern „loginname“
- „userPassword“: Hier wird das gehashte Passwort des Mitarbeiters gepeichert. (das gehashte passwort als plaintext) „password“

Die Attribute sind in Abbildung 46 auch in einem dargestellten Beispieleintrag beschrieben.

Sobald die Verbindung aufgebaut wurde, kann im Programmcode der Datenbank Applikation „Synchronize_DKE.java“ und „LdapServer.java“ geöffnet werden.

Um die Funktion zu testen, kann die main der LdapServer.java aufgerufen, welche die Konfigurationsdaten zu beginn nutzt.

```
//Configuration for test methode (main)
private final String hostname = "localhost";
private final String port = "389";
private final String authuser = "cn=Manager,dc=dkePr,dc=at";
private final String authpassword = "secret";
private final String context = "ou=People,dc=dkePr,dc=at";
```

Abbildung 47: Testkonfiguration LDAP

Damit sind die Einrichtung und Inbetriebnahme des LDAP-Servers beziehungsweise dessen Schnittstelle abgeschlossen und die Synchronisation zwischen Server und Applikation wird alle 5 min durchgeführt sobald sich in der Tabelle „targetstate“ Daten für das Zielsystem „LDAP“ befinden.

Weiters siehe „Synchronisierung starten“.

MYSQL

Das Zielsystem MYSQL benötigt folgende Konfigurationsfelder aus der Tabelle „targetstate“:

Port: der Port unter welchem auf die Zieldatenbank zugegriffen werden kann, zum Beispiel: „389“

Path: der Pfad zu der gewünschten Datenbank in welche die Daten geschrieben werden sollen (exklusive dem Tabellennamen) auf dem Host, zum Beispiel: „/central_db_t2“

User: Der Username mit welchem die Dateien in die Datenbank geschrieben werden sollten (benötigt genügend Rechte), zum Beispiel: „DKETeam“

Password: Das Passwort des User, der verwendet wird um Daten in die Datenbank zu schreiben, zum Beispiel: „mydkepassword“

Hostname: Der Host Name oder die Ip Adresse des Host auf welchem die Datenbank läuft, zum Beispiel: „dkeprw21.mysql.database.azure.com“ oder localhost

Tablename: Der Tabellenname in welche die Daten geschrieben werden sollen, zum Beispiel: „employee“

Um MYSQL zu synchronisieren, muss nur die Synchronisation gestartet werden und die jeweilige Konfiguration in der zentralen Datenbank eingetragen und auf aktiv gesetzt werden.

Synchronisierung starten:

Bevor die Main Methode der Synchronize_DKE.java gestartet und damit auch die Synchronisation durchgeführt wird, sollte die Datenbankkonfiguration der Identitätsdatenbank (central_db_t3) geprüft und eventuell abgeändert werden. Ebenfalls sollte überprüft werden, ob die Schnittstelle zu Ldap aktiv ist mit dem Parameter „ldapConfigured“.

Im Quellcode wurde die Möglichkeit definiert, entweder die cloudbasierte Datenbank zu nutzen (in der allerdings die Zielsysteme für CSV und JSON aus Sicherheitsgründen nicht aktivierbar sind) oder die lokale Datenbank.

Dementsprechend muss der jeweilige Anwendungsfall im Quellcode aktiviert werden bzw. bei der Verwendung der lokalen Datenbank den Usernamen und das Passwort dementsprechend abgeändert werden.

```
//Source Database configuration file for Cloud Database (CentralDB)
/*
public static final String USER = "DKETeam";
public static final String PWD = "mydkepassword";
public static final String CONNECT_STRING = "jdbc:mysql://dkeprw21.mysql.database.azure.com:3306/central_db_t3";
*/
//Source Database configuration file for local Database (CentralDB)
public static final String USER = "root";
public static final String PWD = "root";
public static final String CONNECT_STRING = "jdbc:mysql://localhost:3306/central_db_t3";
public static final boolean ldapConfigured = true;
```

Abbildung 48: Datenbankkonfiguration

Sobald die Main Methode läuft, wird alle 5 Minuten auf Änderungen in der „Employee“ Tabelle geprüft und wenn eine Änderung erkannt wurde, werden auch die Zielsysteme aktualisiert.

Achtung: Es ist nicht vorgesehen, dass ein Employee aus der zentralen Datenbank gelöscht wird, es ist lediglich vorgesehen, dass dieser auf Inaktiv gesetzt wird („is active“ != 1). Erst nachdem alle Zielsysteme diese Änderung synchronisiert bekommen haben, kann auch der Employee aus der Datenbank endgültig gelöscht werden (nicht vorgesehen)

7.2. Rest Service

- Der Rest Service befindet sich im Gesamt Repository der Gruppe unter dem Ordner "Rest". Link für das Gesamtrepository: <https://github.com/galvadino/enterprise-db-app>

 TheFreYzer Add RestService	2971415 13 minutes ago	20 commits
 Datenbank-Zielsysteme	Add Zielsysteme + Reorganize	27 minutes ago
 Dokumentation	Add Zielsysteme + Reorganize	27 minutes ago
 FrontEnd/enterprise-db-app	Final commit / fro complete	3 days ago
 REST	Add RestService	13 minutes ago
 README.md	Project scope	10 days ago

Abbildung 49 REST Ordner in Github

- Nachdem Download des gesamten Projekts befindet sich der Rest Service in folgendem Ordner:

 Datenbank-Zielsysteme	30.01.2022 11:37	Dateiordner
 Dokumentation	30.01.2022 11:37	Dateiordner
 FrontEnd	30.01.2022 11:37	Dateiordner
 REST	30.01.2022 11:42	Dateiordner
 README.md	30.01.2022 11:37	MD-Datei

Abbildung 50 Rest Ordner 1

- Diesen Ordner gilt es nun in eine Entwicklungsumgebung (z. B. IntelliJ) zu importieren. (Über Menüpunkt File -> Open Auswahl des „REST“ Ordners)

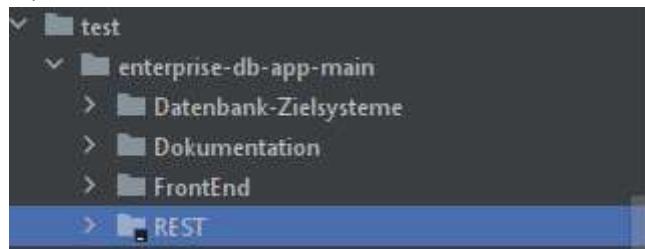


Abbildung 51 REST Ordner 2

4. Das Projekt sollte nun in der Entwicklungsumgebung folgende Struktur haben:

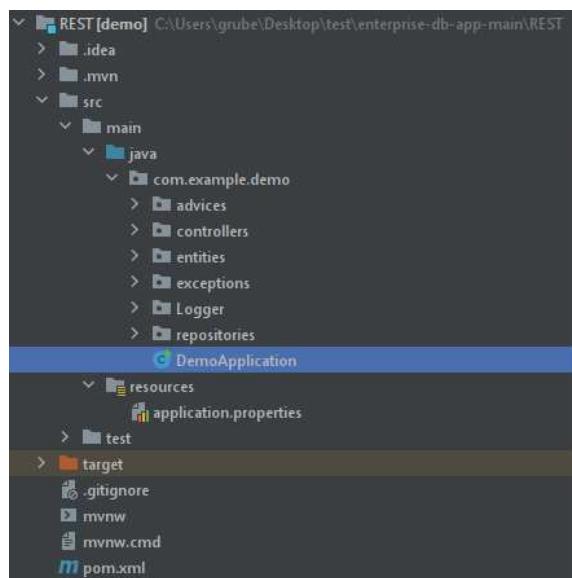


Abbildung 52 Rest Struktur

5. Start der Anwendung über die Mainklasse „DemoApplication“. (Rechtsklick auf diese Klasse -> Run)

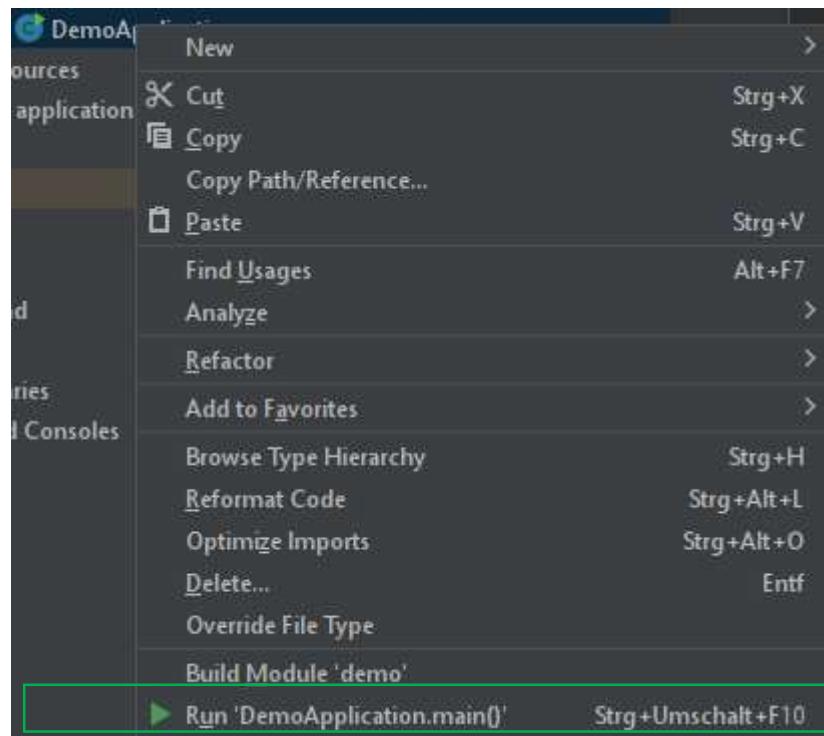


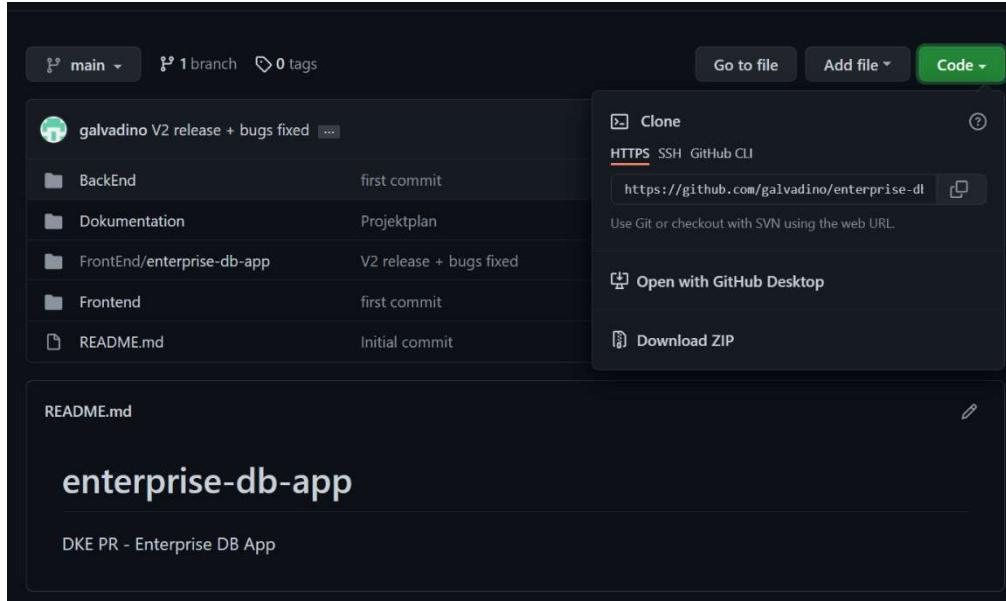
Abbildung 53 Demo Application Start

Sobald die Konsole die Rückmeldung gibt, dass der Rest-Service erfolgreich gestartet wurde kann das Frontend im Browser geöffnet werden. Der Service läuft lokal auf Port 8080. Dieser Port gehört dementsprechend freigehalten bzw. muss sonst der Port in der Properties Datei im Rest-Projekt geändert werden. Des Weiteren gilt es zu erwähnen das eine aktuelle Java Version zur Verwendung des Services benötigt wird. Die Rest Service Entwickler arbeiteten hierbei mit der Java Version 17.

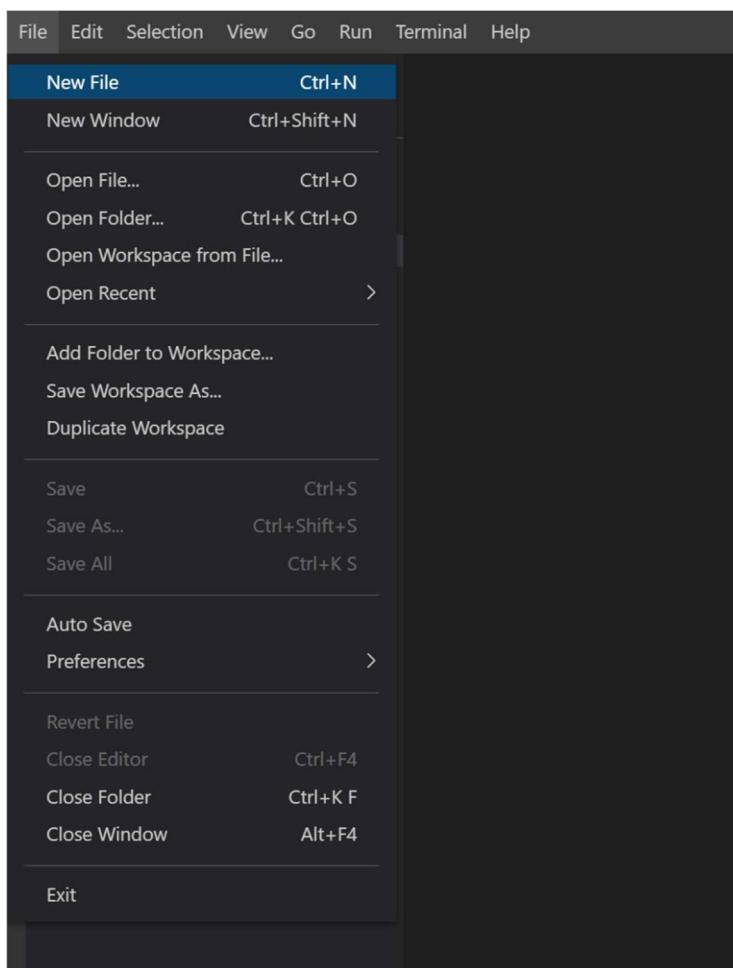
Anmerkung: Firefox macht Probleme mit der generellen Aktualisierung der Datensätze im UI.
Dementsprechend Chrome, Edge oder ähnliches verwenden. Auch hier muss in dem Fall nach Änderungen bzw. Frontendoperationen die Seite aktualisiert werden.

7.3. Front End

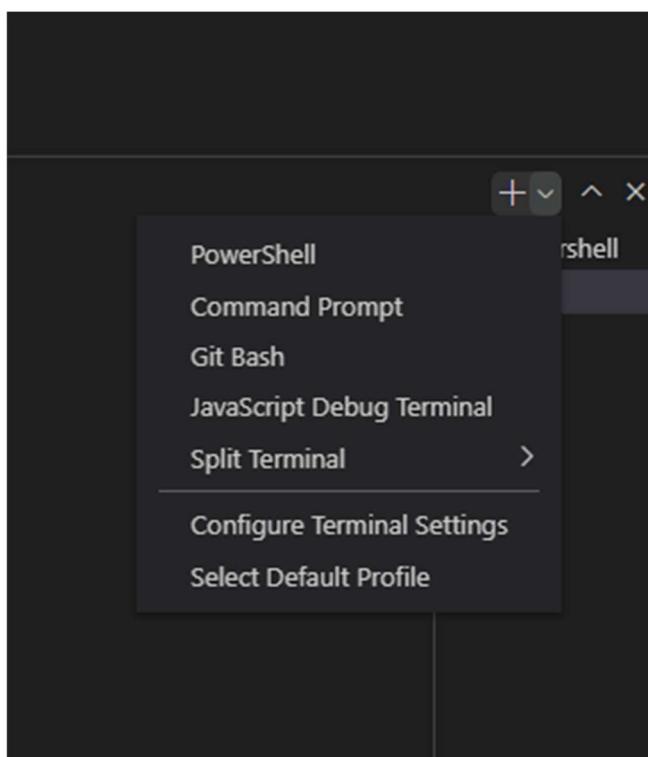
1. Download des Frontend ZIP von GitHub (<https://github.com/galvadino/enterprise-db-app>)



2. Installation des Angular CLI "Command Line Interface" mittels Ausführung von "npm install -g @angular/cli" auf der Windows Kommandozeile
3. Entpackgen der ZIP und Import des "enterprise-db-app" Ordners als Projekt in eine Entwicklungsumgebung wie z.B. Visual Studio Code. Pfad lautet wie folgt: enterprise-db-app-main\FrontEnd\enterprise-db-app



4. Öffnen eines neuen “Command Prompt” Terminals innerhalb der Entwicklungsumgebung zum Starten (unten rechts) oder via “New Terminal” (Screenshot aus Visual Studio Code). Ausführung des Befehls “npm install” zur Installation der notwendigen Module.



5. Start der Anwendung via Befehl in Kommandozeile "ng serve - open" bzw. "ng s - o"

```
C:\Users\dinos\OneDrive\Dokumente\GitHub\enterprise-db-app\FrontEnd\enterprise-db-app>ng s -o
✓ Browser application bundle generation complete.

Initial Chunk Files | Names      | Size
vendor.js           | vendor    | 2.47 MB
polyfills.js        | polyfills | 339.14 kB
main.js              | main      | 227.37 kB
styles.css, styles.js | styles   | 212.87 kB
runtime.js          | runtime   | 6.87 kB

| Initial Total | 3.23 MB

Build at: 2022-01-30T11:08:37.966Z - Hash: 0985a8f72c2aaaaea - Time: 5665ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

✓ Compiled successfully.
```

6. Automatisches Öffnen der Webapplikation in einem Browserfenster als localhost. Applikation ist für Google Chrome optimiert. Empfehlung der Nutzung von diesem. Sollte sich das Fenster nicht automatisch öffnen bitte folgenden Link in Google Chrome öffnen: <http://localhost:4200/>

8. Use Cases

8.1. Use Cases 1: Anlage eines neuen Mitarbeiters.

Die fiktive Mitarbeiterin „Miriam Musterfrau“ wird über die Applikation angelegt.

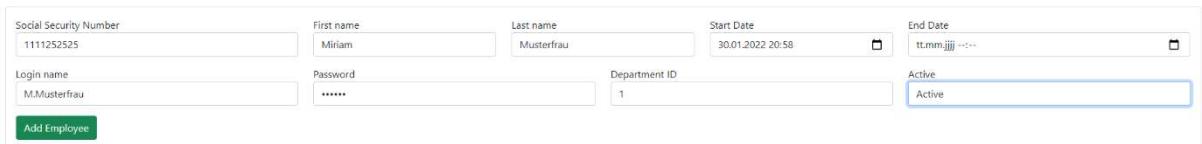


Abbildung 54: Anlage eines neuen Mitarbeiters

Nach der Anlage des Mitarbeiters ist der Datensatz in der Liste vorhanden.

54	Miriam	Musterfrau	M.Musterfrau	1111252525	1	Active	2022-01-30			View	Edit	Delete

Abbildung 55: Datensatz des neuen Mitarbeiters

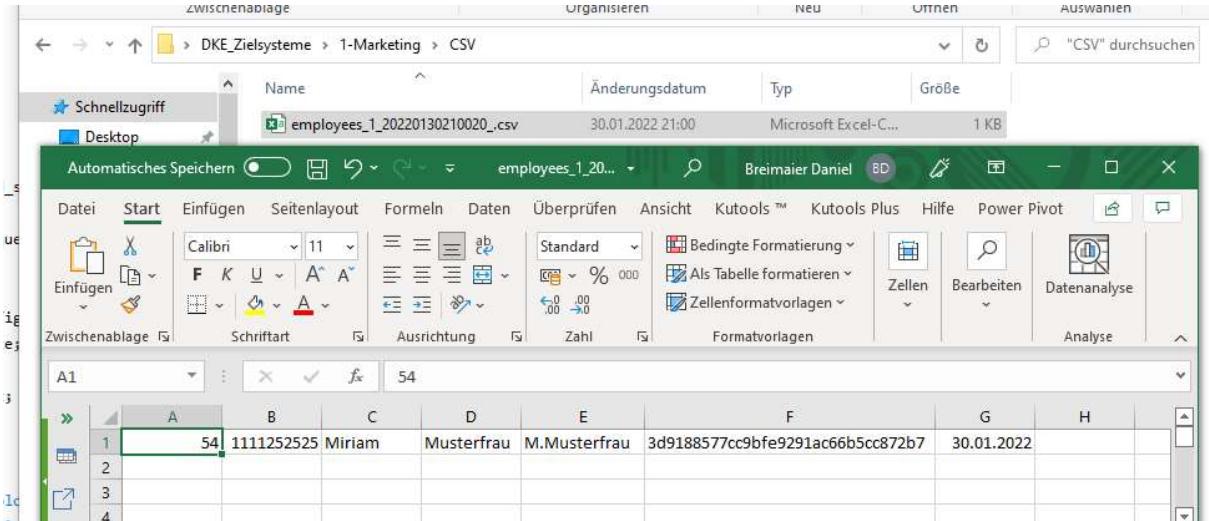
Da der Mitarbeiter in der Abteilung „Marketing“ einfügt wird, werden nun alle Zielsysteme für, die für diese Abteilung konfiguriert sind, aktualisiert.

Die für die Abteilung „Marketing“ definierten Zielsysteme sind „MySQL“, „JSON“ und „CSV“. Laut der Konfigurationstabelle werden dabei jene Mitarbeiterinformationen exportiert, die jeweils in den Targetkonfigurationen definiert sind.

ID	Active state	Type	Last Updated	Active since	Target config ID	Department ID	Path	Hostname	Password	Port	Tablename	User	Edit
1	True	MySQL			7	1	/mysqltarget	localhost	root	3306	targetemployee	root	Edit Delete
2	True	CSV			1	1	C:/Users/d.breimaier/Desktop/DKE_Zielsysteme/1-Marketing/CSV/						Edit Delete
3	True	JSON			1	1	C:/Users/d.breimaier/Desktop/DKE_Zielsysteme/1-Marketing/JSON/						Edit Delete

Abbildung 56: Targetkonfigurationen in der Applikation

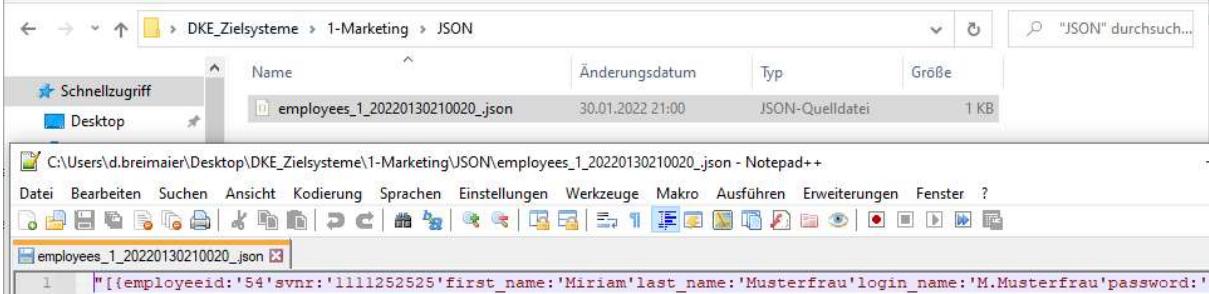
CSV:



	A	B	C	D	E	F	G	H
1	54	1111252525	Miriam	Musterfrau	M.Musterfrau	3d9188577cc9bfe9291ac66b5cc872b7	30.01.2022	
2								
3								
4								

Abbildung 57: Zielsystem "CSV" mit neuem Mitarbeiter

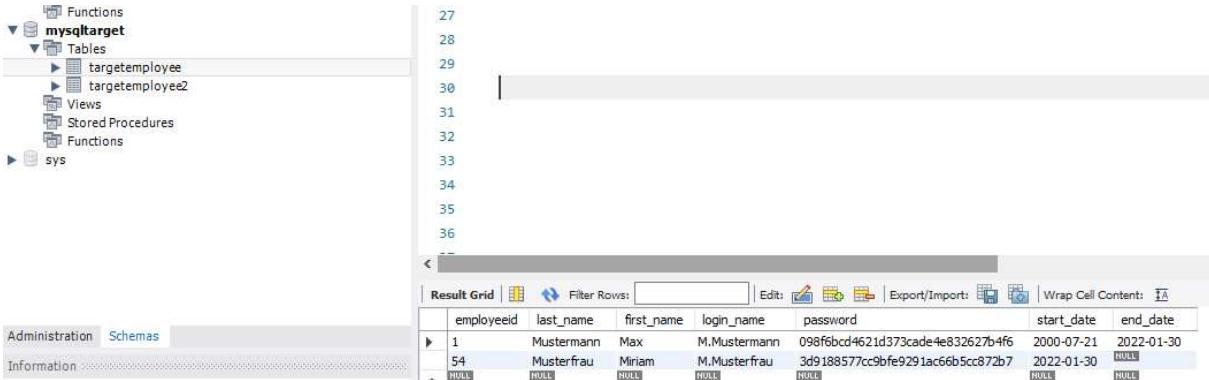
JSON:



```
[{"employeeid": "54", "svnr": "1111252525", "first_name": "Miriam", "last_name": "Musterfrau", "login_name": "M.Musterfrau", "password": "3d9188577cc9bfe9291ac66b5cc872b7"}]
```

Abbildung 58: Zielsystem "JSON" mit neuem Mitarbeiter

MYSQL (Tabelle „targetemployee“):



	employeeid	last_name	first_name	login_name	password	start_date	end_date
▶	1	Mustermann	Max	M.Mustermann	098f6bcd4621d373cade4e832627b4f6	2000-07-21	2022-01-30
▶	54	Musterfrau	Miriam	M.Musterfrau	3d9188577cc9bfe9291ac66b5cc872b7	2022-01-30	NULL

Abbildung 59: Zielsystem "MySQL" mit neuem Mitarbeiter

8.2. Use Chase 2: Abteilungswechsel eines Mitarbeiters:

Der fiktive Mitarbeiter „Peter Maier“ wechselt mit 01.02.2022 von der Abteilung „Marketing“ in die Abteilung „Manufacturing“.

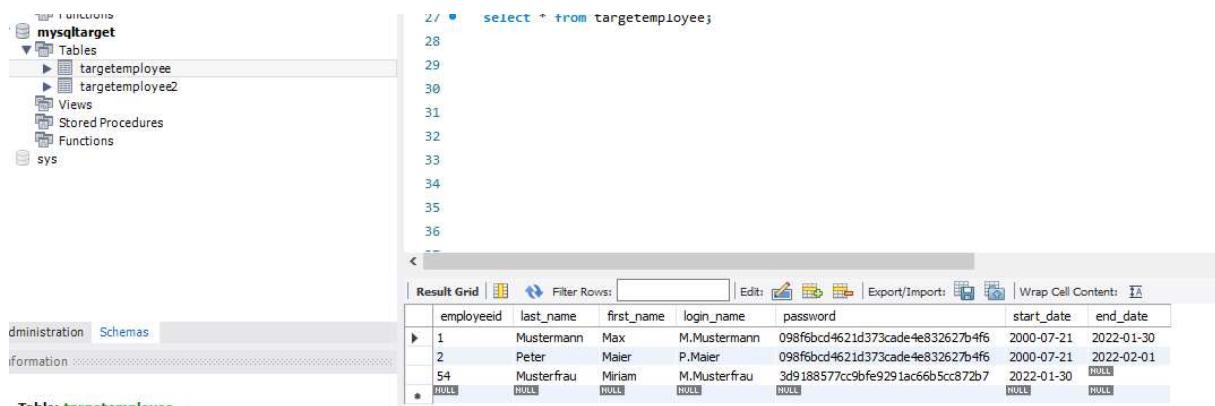
Dabei wird für den bereits existierenden existierende Mitarbeiter „Peter Mair, Mitarbeiter-ID 2“ ein Enddatum gesetzt:

2	Maier	Peter	P.Maier	123456785	1	Active	2000-07-21	2022-02-01	View	Edit	Delete
---	-------	-------	---------	-----------	---	--------	------------	------------	----------------------	----------------------	------------------------

Abbildung 60: Abteilungswechsel des Mitarbeiters "Peter Maier"

Dadurch werden wieder die definierten Zielsysteme der Marketingabteilung aktualisiert und das entsprechende Enddatum für den Mitarbeiter gesetzt:

MySQL:



The screenshot shows the MySQL Workbench interface. On the left, the schema browser displays the database `mysqltarget` with tables `targetemployee` and `targetemployee2`. The `targetemployee` table is selected. In the center, a SQL editor window shows the query `select * from targetemployee;`. Below it, the results grid displays the following data:

employeeid	last_name	first_name	login_name	password	start_date	end_date
1	Mustermann	Max	M.Mustermann	098f6bcd4621d373cade4e832627b4f6	2000-07-21	2022-01-30
2	Peter	Maier	P.Maier	098f6bcd4621d373cade4e832627b4f6	2000-07-21	2022-02-01
54	Musterfrau	Miriam	M.Musterfrau	3d9188577cc9bfe9291ac66b5cc872b7	2022-01-30	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL

Abbildung 61: Zielsystem "MySQL" mit nach dem Abteilungswechsel

CSV:

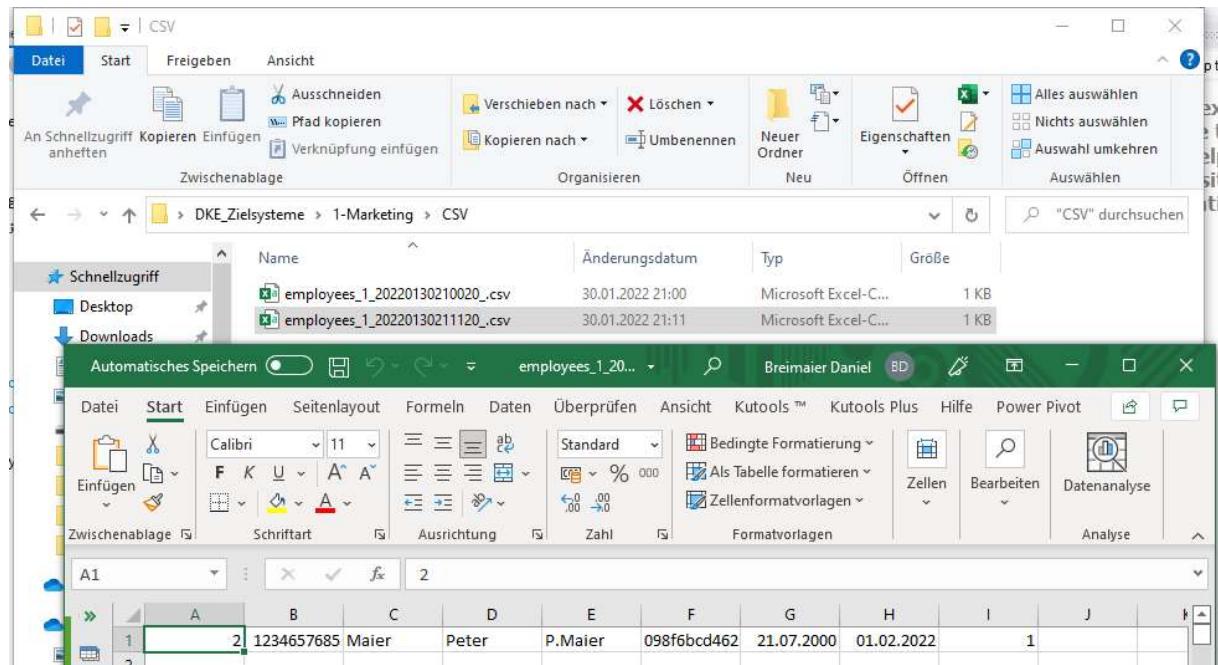


Abbildung 62: Zielsystem "CSV" nach dem Abteilungswechsel

JSON:

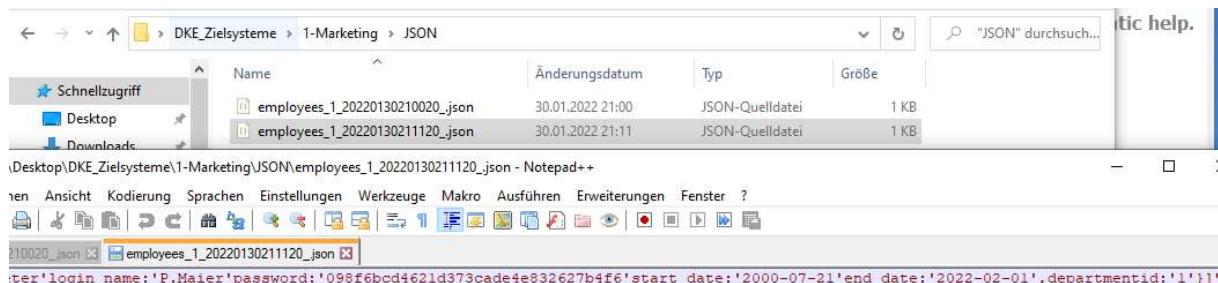


Abbildung 63: Zielsystem "JSON" nach dem Abteilungswechsel

Anschließend wird der Mitarbeiter in der Applikation neu eingegeben und für die Abteilung 7 – Manufacturing angelegt:

55	Peter	Maier	P.Maier	1234657685	7	Active	2022-02-01		View	Edit	Delete
----	-------	-------	---------	------------	---	--------	------------	--	------	------	--------

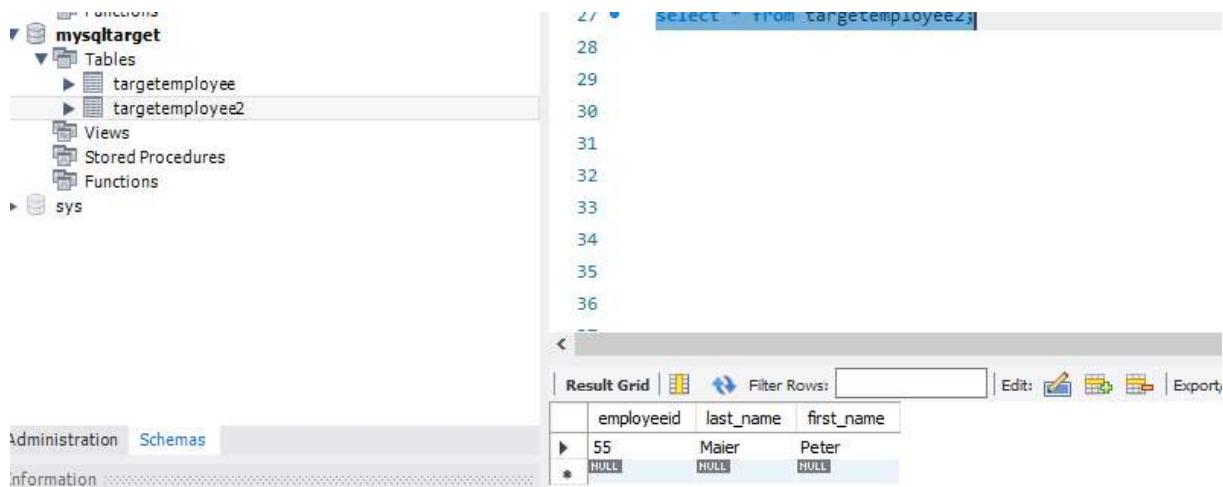
Abbildung 64: Mitarbeiter in der neuen Abteilung

Für diese Abteilung sind nun die Zielsysteme „MySQL“ und „LDAP“ definiert:

10	True	MySQL		8	7	/mysqltarget	localhost	root	3306	targetemployee2	root	Edit	Delete
11	True	LDAP		1	7	,ou=People,dc=dkePr,dc=at	localhost	secret	389		cn=Manager,dc=dkePr,dc=at	Edit	Delete

Abbildung 65: Zielsysteme der Abteilung "Manufacturing"

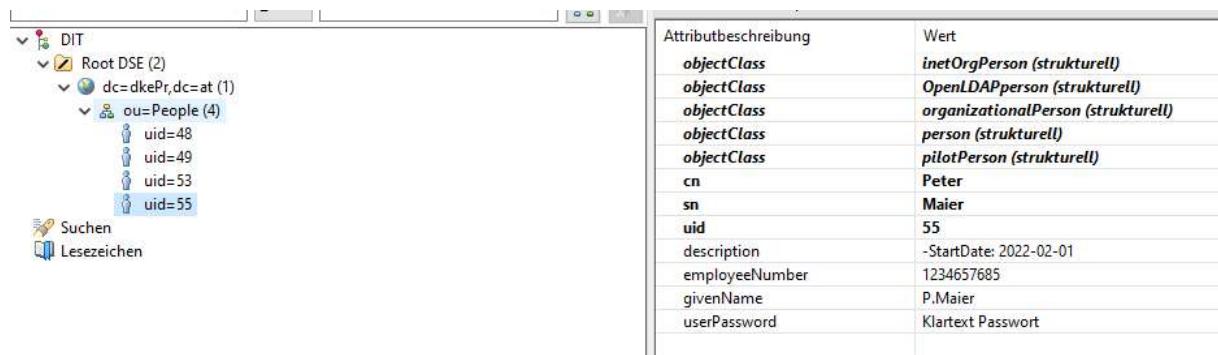
Diese Zielsysteme werden wieder über die Applikation aktualisiert:
 MYSQL(„targetemployee2“):



The screenshot shows the MySQL Workbench interface. On the left, the schema browser displays the database 'mysqltarget' with tables 'targetemployee' and 'targetemployee2'. The 'targetemployee2' table is selected. On the right, the SQL editor window contains the query 'SELECT * FROM targetemployee2;'. Below it, the result grid shows a single row with columns 'employeedid', 'last_name', and 'first_name'. The values are 55, Maier, and Peter respectively.

Abbildung 66: "MySQL" Zielsystem nach dem Abteilungswechsel

LDAP:



The screenshot shows an LDAP management interface. On the left, a tree view of the directory structure under 'DIT' shows 'Root DSE (2)' and 'dc=dkePr,dc=at (1)'. Under 'dc=dkePr,dc=at (1)', there is an 'ou=People (4)' entry which contains four entries labeled 'uid=48', 'uid=49', 'uid=53', and 'uid=55'. On the right, a detailed view of the 'uid=55' entry is shown in a table. The table has two sections: 'Attributbeschreibung' (Attribute Description) and 'Wert' (Value). The 'Attributbeschreibung' column lists attributes like 'objectClass', 'cn', 'sn', 'uid', 'description', 'employeeNumber', 'givenName', and 'userPassword'. The 'Wert' column provides their corresponding values, such as 'inetOrgPerson (strukturell)', 'Peter', 'Maier', '55', and 'P.Maier'.

Abbildung 67: "LDAP" Zielsystem nach dem Abteilungswechsel

8.3. Use Case 3: Austritt eines Mitarbeiters:

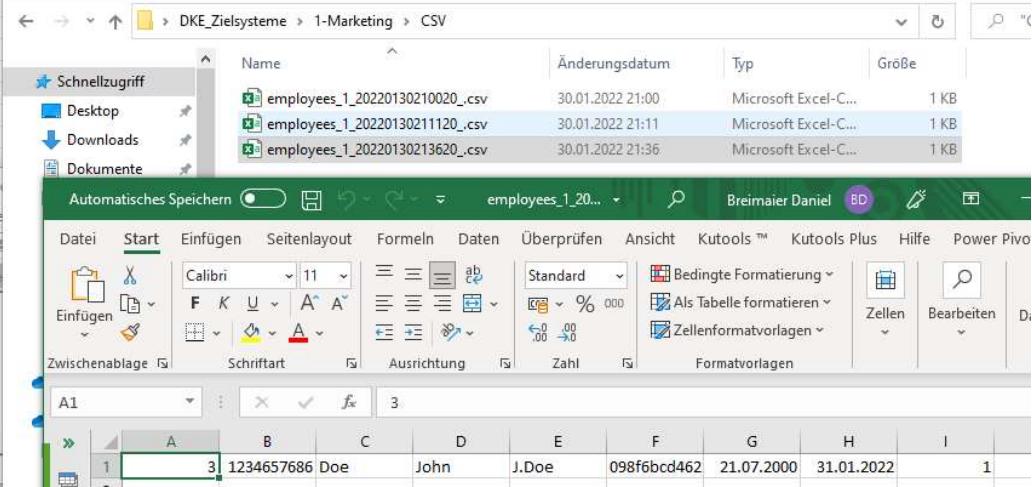
Der Mitarbeiter „John Doe“ ID: 3 verlässt mit 31.01.2022 das Unternehmen bzw. die Abteilung 1 Marketing. Im User Interface wird das Enddatum dementsprechend gesetzt:

3	Doe	John	J.Doe	1234657686	1	Active	2000-07-21	2022-01-31	View	Edit	Delete
---	-----	------	-------	------------	---	--------	------------	------------	----------------------	----------------------	------------------------

Abbildung 68: Austritt eines Mitarbeiters

Anschließend werden wieder die konfigurierten Zielsysteme automatisch aktualisiert und das Enddatum gesetzt.

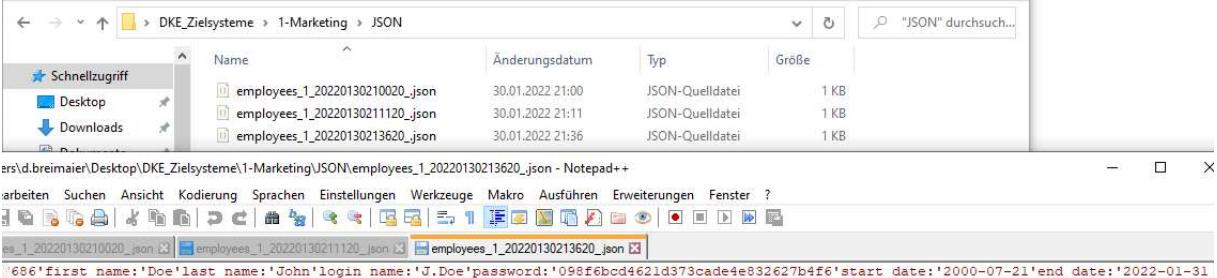
CSV:



Employee ID	Last Name	First Name	Login Name	Password	Start Date	End Date
1	Doe	John	J.Doe	098f6bcd4621d373cade4e832627b4f6	21.07.2000	31.01.2022

Abbildung 69: Zielsystem "CSV" nach dem Austritt

JSON:

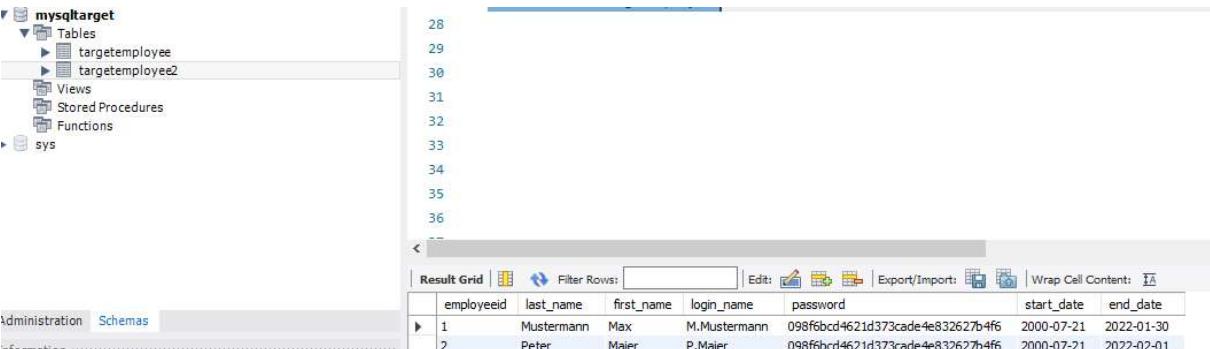


```

{
  "first_name": "Doe",
  "last_name": "John",
  "login_name": "J.Doe",
  "password": "098f6bcd4621d373cade4e832627b4f6",
  "start_date": "2000-07-21",
  "end_date": "2022-01-31"
}
  
```

Abbildung 70: Zielsystem "JSON" nach dem Austritt

MYSQL:



Employee ID	Last Name	First Name	Login Name	Password	Start Date	End Date
1	Mustermann	Max	M.Mustermann	098f6bcd4621d373cade4e832627b4f6	2000-07-21	2022-01-30
2	Peter	Maier	P.Maier	098f6bcd4621d373cade4e832627b4f6	2000-07-21	2022-02-01

Abbildung 71: Zielsystem "MySQL" nach dem Austritt

9. Reflexion Projekt

Durch die Vielseitigkeit des Projektes war es möglich, dass jede Person des Teams seine Stärken am besten nutzen konnte. Die enge Zusammenarbeit, welche durch die vielen Abhängigkeiten und Schnittstellen notwendig war, wurde durch regelmäßige Meetings ermöglicht. Dadurch konnte das gesamte Team sich gut abstimmen und keine Person driftete trotz des unterschiedlichen Fokus ab. Die Zusammenarbeit gelang gut und besonders beim Testen der einzelnen Komponenten und Anleitungen wurde sich gegenseitig und auch Fokusübergreifend unterstützt. Dadurch konnte auch sichergestellt werden, dass jede Person sich in den anderen Technologien, die verwendet wurden, auch etwas einfinden konnte und dadurch diese besser zu verstehen gelernt hat. Es gab in dem Team keine großen Konflikte und jeder beteiligte hat die Aufgaben, die zu Beginn fair verteilt wurden, termingerecht erfüllt.

ABBILDUNGSVERZEICHNIS

Abbildung 1: BPMN Frontend Diagramm	12
Abbildung 2: Front-End: Startseite	15
Abbildung 3: Front-End: Mitarbeiter bearbeiten.....	15
Abbildung 4: Front-End: Mitarbeiter löschen	16
Abbildung 5: Front-End: Abteilungen	16
Abbildung 6: Front-End: Abteilungen anlegen.....	17
Abbildung 7: Front-End: Mitarbeiter bearbeiten.....	17
Abbildung 8: Front-End: Mitarbeiter löschen	18
Abbildung 9: Front-End: Zielsystem-Konfiguration	18
Abbildung 10: Front-End: Zielsystem-Konfiguration bearbeiten.....	19
Abbildung 11: Front-End: Zielsystem-Konfiguration löschen	19
Abbildung 12: Front-End: Zielsystem-Zustand	20
Abbildung 13: Zielsystem-Zustand CSV.....	21
Abbildung 14: Front-End: Zielsystem-Zustand: JSON	21
Abbildung 15: Front-End: Zielsystem-Zustand LDAP	22
Abbildung 16: Front-End: Zielsystem-Zustand bearbeiten.....	22
Abbildung 17: Front-End: Zielsystem-Zustand löschen	23
Abbildung 18: Employee Entity	25
Abbildung 19: Java Projektstruktur.....	25
Abbildung 20: JSON Requestbody.....	26
Abbildung 21: Employee Post Methode	26
Abbildung 22: EmployeeRepository	26
Abbildung 23: Initialisierung eines Controllers mit dem Repository	27
Abbildung 24: EmployeeBadRequestException Klasse.....	28
Abbildung 25: EmployeeBadRequestAdvice Klasse	29
Abbildung 26: Ausschnitt Patch Methode – Target State	29
Abbildung 27: ID Generator	31
Abbildung 28: Tabellenschema.....	35
Abbildung 29: Tabelle "employee"	36
Abbildung 30: Tabelle "employee"	36
Abbildung 31: Tabelle "department"	36
Abbildung 32: Tabelle "targetconfig"	36
Abbildung 33: Tabelle "targetstate"	37
Abbildung 34: Sicherheitseinstellungen unter Windows 10	38
Abbildung 35: secure-file-priv Einstellung	39
Abbildung 36: Events in MySQL	39
Abbildung 37: Erstellung MySQL Events.....	39
Abbildung 38: Start des OpenLdap Services.....	44
Abbildung 39: OpenLdap Service.....	44
Abbildung 40: Fehlermeldung Apache Directory Studio	45
Abbildung 41: Pfad zur zu ändernden Datei	46
Abbildung 42: Änderungen der Datei „ApacheDirerctoryStudio.ini“	46
Abbildung 43: Hinzufügen einer neuen Verbindung	46
Abbildung 44: Eingabe der Verbindungsdaten	47
Abbildung 45: Eingabe der Authentifizierungsdaten	47

Abbildung 46: Verbundener LDAP Server + Struktur	48
Abbildung 47: Testkonfiguration LDAP	49
Abbildung 48: Datenbankkonfiguration	50
Abbildung 49 REST Ordner in Github	51
Abbildung 50 Rest Ordner 1.....	51
Abbildung 51 REST Ordner 2	51
Abbildung 52 Rest Struktur	52
Abbildung 53 Demo Application Start.....	52
Abbildung 54: Anlage eines neuen Mitarbeiters	57
Abbildung 55: Datensatz des neuen Mitarbeiters	57
Abbildung 56: Targetkonfigurationen in der Applikation	57
Abbildung 57: Zielsystem "CSV" mit neuem Mitarbeiter	58
Abbildung 58: Zielsystem "JSON" mit neuem Mitarbeiter.....	58
Abbildung 59: Zielsystem "MySQL" mit neuem Mitarbeiter.....	58
Abbildung 60: Abteilungswechsel des Mitarbeiters "Peter Maier".....	59
Abbildung 61: Zielsystem "MySQL" mit nach dem Abteilungswechsel	59
Abbildung 62: Zielsystem "CSV" nach dem Abteilungswechsel	60
Abbildung 63: Zielsystem "JSON" nach dem Abteilungswechsel	60
Abbildung 64: Mitarbeiter in der neuen Abteilung.....	60
Abbildung 65: Zielsysteme der Abteilung "Manufacturing"	60
Abbildung 66: "MySQL" Zielsystem nach dem Abteilungswechsel	61
Abbildung 67: "LDAP" Zielsystem nach dem Abteilungswechsel	61
Abbildung 68: Austritt eines Mitarbeiters	61
Abbildung 69: Zielsystem "CSV" nach dem Austritt.....	62
Abbildung 70: Zielsystem "JSON" nach dem Austritt.....	62
Abbildung 71: Zielsystem "MySQL" nach dem Austritt	62