Personenverwaltungs-apparat

Datenbankprojekt AuP vom 26.03.2021

BS1 Kempten

Julian Mader, Jeremy Gassner, Kevin Dusch – IT12c

Inhaltsverzeichnis

[1. Einleitung 2](#_Toc67674287)

[1.1 AuP Projekt 2](#_Toc67674288)

[1.2 Projektbeschreibung 2](#_Toc67674289)

[1.3 Projektziel 2](#_Toc67674290)

[1.4 Umgebung 2](#_Toc67674291)

[2. Analysephase 3](#_Toc67674292)

[2.1 Ist-Analyse 3](#_Toc67674293)

[2.2 Soll-Konzept 3](#_Toc67674294)

[3. Entwurfsphase 3](#_Toc67674295)

[3.1 Zielplattform 3](#_Toc67674296)

[3.2 Ermittlung der Funktionen 3](#_Toc67674297)

[3.3 Die Benutzeroberfläche 5](#_Toc67674298)

[4. Implementierungsphase 5](#_Toc67674299)

[4.1 Programmarchitektur 5](#_Toc67674300)

[4.2 Datenbankanbindung 5](#_Toc67674301)

[5. Testphase 6](#_Toc67674302)

[6. Abnahme- und Einführungsphase 6](#_Toc67674303)

[6.1 Abnahme 6](#_Toc67674304)

[6.2 Deployment und Einführung 6](#_Toc67674305)

[7. Fazit 6](#_Toc67674306)

[8. Anhänge 8](#_Toc67674307)

[8.1 Screenshots der Sprints 8](#_Toc67674308)

[8.2 Datenbank 21](#_Toc67674309)

[8.2.1 ER-Diagramm 21](#_Toc67674310)

[8.2.2 Datenbankschema 22](#_Toc67674311)

[8.3 Oberfläche 23](#_Toc67674312)

[8.4 Codeausschnitte 26](#_Toc67674313)

[8.5 Use Case Diagramme 29](#_Toc67674314)

# Einleitung

## AuP Projekt

Im Laufe der 12 Klasse wird ein Projekt im Bereich Datenbanken für das Fach AuP geschrieben. Hierbei steht es den Schülern komplett frei, welche Art von Anwendung sie schreiben, solange es einen Bezug zu einer Datenbank hat. Dieses Projekt stützt sich auf einen fiktiven Auftrag der Firma Soloplan GmbH.

## Projektbeschreibung

Die Firma Soloplan GmbH hat der Firma Gamadu den Auftrag erteilt, eine Anwendung zu entwickeln, welche der Personalabteilung die Möglichkeiten bietet, alle Verwaltungsaufgaben, welche derzeit durch mehrere Programme erledigt werden, in einem Programm zu verarbeiten.

Dabei sollen alle Zugriffe auf die Personen-, Abteilungs-, Positions-, Raum- und Vertragsdaten gewährleistet werden. Sie möchten eine möglichst übersichtliche und kompakte Oberfläche, damit alle Daten so gut es geht zusammen sichtbar sind.

## Projektziel

Ziel des Projektes ist, es alle personalbezogenen Aufgaben in einer Anwendung zu kombinieren. Hierbei sollen alle benötigten Funktionen, welche die Personalabteilung für die Pflege der Daten verwendet, implementiert sein.

## Umgebung

Im Projektablauf wird eine Desktop Anwendung geschrieben, welche auf Windows verfügbar sein soll. Diese Anwendung wird mit der Programmiersprache C# in Visual Studio entwickelt. Die Datenbank wird als MySQL-Datenbank realisiert.

# Analysephase

## Ist-Analyse

Die Soloplan GmbH verwaltet derzeit alle Daten in mehreren kleinen Programmen, welche nicht auf die gleiche Datenbank zugreifen. Durch die Speicherung auf mehreren Datenbanken sind inkonsistente und redundante Datensätze der Fall. Die Verwaltung des Personals wird allein von der Personalabteilung übernommen.

## Soll-Konzept

Anstatt vielen kleinen Programmen werden alle Funktionalitäten in einer großen Anwendung zusammengefasst. Dazu werden noch weitere Funktionen hinzugefügt, welche während der Entwicklung noch auftreten könnten. Diese Anwendung wird auf eine einheitliche Datenbank zugreifen, damit wird die Pflege der Daten erleichtert und alle Daten bleiben konsistent.

# Entwurfsphase

## Zielplattform

Die Anwendung wird in einer Windows Domäne verwendet.

## Ermittlung der Funktionen

Die Personalabteilung von der Soloplan GmbH hat eine Liste mit allen Funktionen geschrieben, welche sie derzeitig in ihren Verwaltungen verwenden (Use Cases siehe Anhang 8.5). Daraus konnte man folgende Module ableiten, welche die verschiedenen Funktionen beinhalten:

Mitarbeiterverwaltung:

Dieses Modul dient für die Verwaltung aller Mitarbeiter und deren Daten. Es dient als Kernstück der Verwaltung, da durch dieses alle Mitarbeiterzuordnungen vorgenommen werden können.

Funktionen:

* Auflistung aller Mitarbeiter
* Verwaltung von Mitarbeitern
* Zuordnung der Positionen, Abteilungen, Räume und Verträge

Abteilungsverwaltung:

Dieses Modul bietet der Personalabteilung die Möglichkeit, alle fachbezogenen Daten für die Abteilungen zu verwalten. Dazu gibt es eine Auswertungsfunktion, welche eine Auflistung aller zugeordneten Mitarbeiter der Abteilung ermöglicht.

Funktionen:

* Auflistung aller Abteilungen
* Verwalten von Abteilungen und deren Mitarbeiter

Positionsverwaltung:

Dieses Modul dient zur Verwaltung aller Positionen der Firma. Mit diesem Reiter wird es ermöglicht alle Daten der Positionen zu verwalten.

Funktionen:

* Auflistung aller Positionen
* Bearbeitung der Positionen und deren Mitarbeiter

Raumverwaltung:

Dieses Modul dient zur Verwaltung aller Räume der Firma. Mit diesem Reiter wird es ermöglicht alle Daten der Räume zu verwalten und diesem Mitarbeiter zu zuordnen.

Funktionen:

* Auflistung aller Räume
* Bearbeitung der Räume und deren Zuordnungen

## Die Benutzeroberfläche

Ziel ist es, eine klar strukturierte und einfache Oberfläche zu entwickeln, um die neue Applikation möglichst benutzerfreundlich bedienen zu können. Um dies zu erreichen, sollen die einzelnen Module mit einem Reiter-System dargestellt werden. Dadurch bekommt jedes Modul einen eigenen Tab (Siehe Anhang 8.3). Die Module folgen alle einem einheitlichen Aufbau.

# Implementierungsphase

## Programmarchitektur

Die Anwendung soll in späteren Stadien leicht erweiterbar und wartbar sein. Um diese Eigenschaften umzusetzen, wird hierbei eine modulare Programmstruktur verwendet. Jedes Modul wird als eigenes, austauschbares und einzeln testbares Projekt verwaltet. Um diese modulare Struktur umzusetzen, wird ein Framework namens Prism[[1]](#footnote-1) verwendet.

Des Weiteren wird das MVVM-Pattern[[2]](#footnote-2) (Model-View-ViewModel) für die Oberflächenanbindung verwendet (Beispiel in Anhang 8.4). Dieses Pattern ermöglicht eine lose Kopplung zwischen Oberfläche und Programmlogik. Bei der Implementierung wird hierbei wieder auf das Prism Framework zurückgegriffen.

## Datenbankanbindung

Für die Datenbankanbindung wurde auf das Dapper[[3]](#footnote-3) Framework zurückgegriffen (Beispiel in Anhang 8.4). Dieses Framework ist ein Micro-ORM und deckt das reine Speichern von Objekten in einer Datenbank ab. Hierbei ist Dapper hauptsächlich für das Mapping der Objekteigenschaften und der Tabellenspalten zuständig. Auf Datenbankebene wurde für jeden Objekttypen (bspw. eine Abteilung) jeweils eine Stored Procedure für jede der CRUD[[4]](#footnote-4)-Methoden angelegt. Dies ermöglicht es, dass auf Programmebene nur der Name der Stored Procedure und die nötigen Daten notwendig sind. Dies erhöht die Wartbarkeit der Software, da mögliche Änderungen der SQL-Befehle auf Datenbankebene vorgenommen werden können, ohne das Programm neu kompilieren zu müssen.

# Testphase

Während der Erstellung der Applikation wurden die einzelnen Module nach deren Fertigstellung getestet und Fehler ggf. behoben. Nach der Fertigstellung der Anwendung wurde diese auf jedem der Entwicklerrechner installiert und mit einer zentralen Datenbank verbunden. Es wurden Aktionen, wie bspw. die gleichzeitige Datenanlage oder Datenlöschung durchgeführt, um die parallele Nutzung der Software zu testen.

# Abnahme- und Einführungsphase

## Abnahme

Nachdem die gesamte Applikation fertig gestellt war, konnte diese dem Projektleiter der Soloplan GmbH übergeben werden.

## Deployment und Einführung

Der ausgelieferte Datenbankstruktur-Dump wurde auf einer MySQL-Datenbank aufgespielt, welche auf einem dedizierten Datenbankserver läuft. Die Software wurde danach auf jedem der Clients installiert. Bei der Installation der Clients musste die Datenbankverbindung entsprechend des Netzwerkes der Firma Soloplan GmbH angepasst werden.

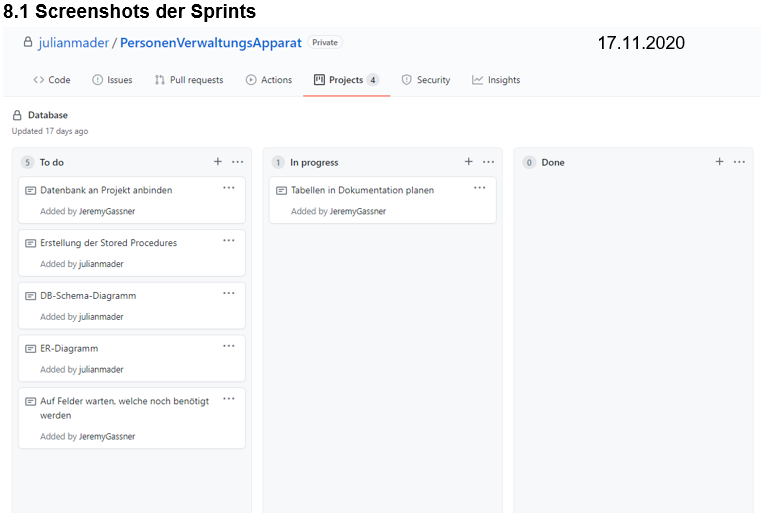
# Fazit

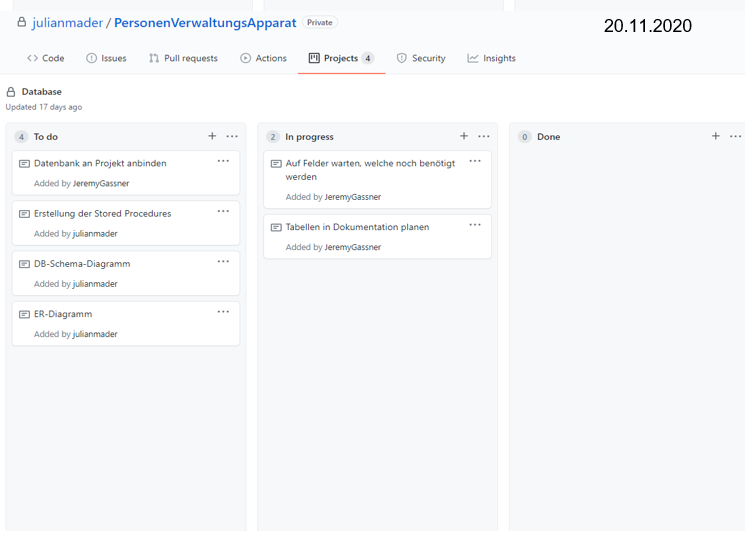
Rückblickend auf dieses Projekt kann festgehalten werden, dass alle zuvor erfassten Anforderung gemäß des Pflichtenheftes erfüllt werden konnten. Der zeitliche Rahmen konnte im Allgemeinen eingehalten werden. Die Testphase der Software hat jedoch mehr Zeit beansprucht als geplant. Dieser zusätzliche Zeitbedarf wurde jedoch durch die Zeitersparnis bei der Dokumentationserstellung kompensiert. Hierbei wurde die Erstellung der Dokumentation an die einzelnen Teammitglieder verteilt.

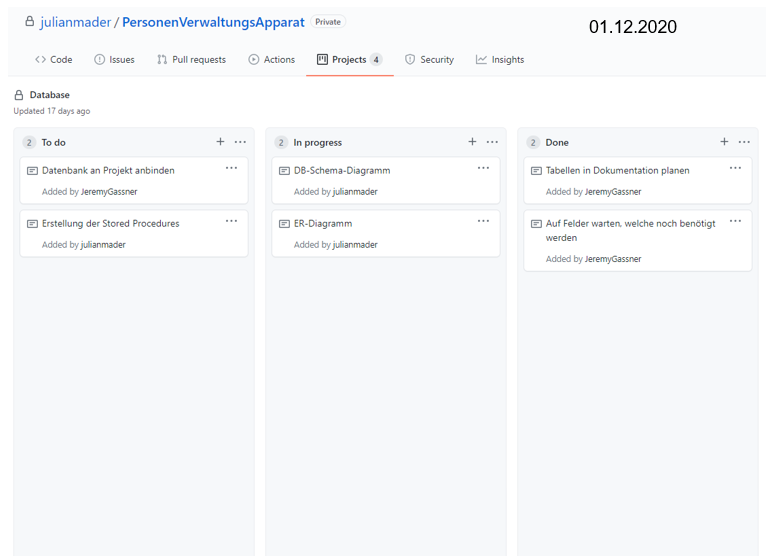
# Anhänge

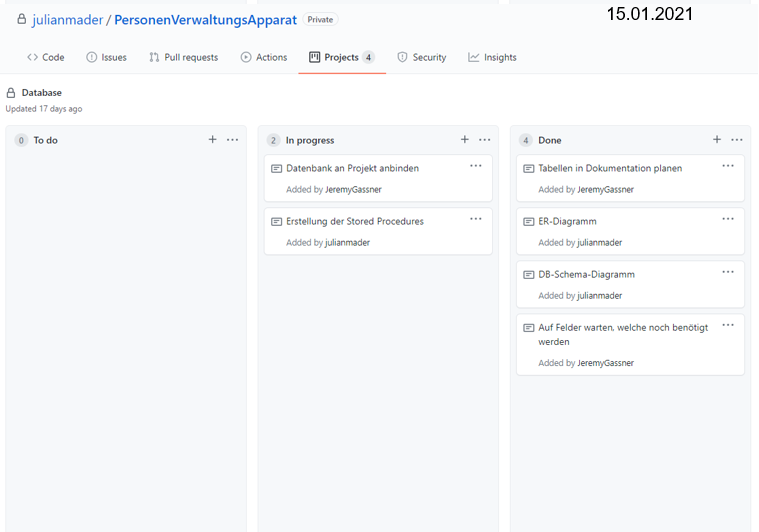
## Screenshots der Sprints

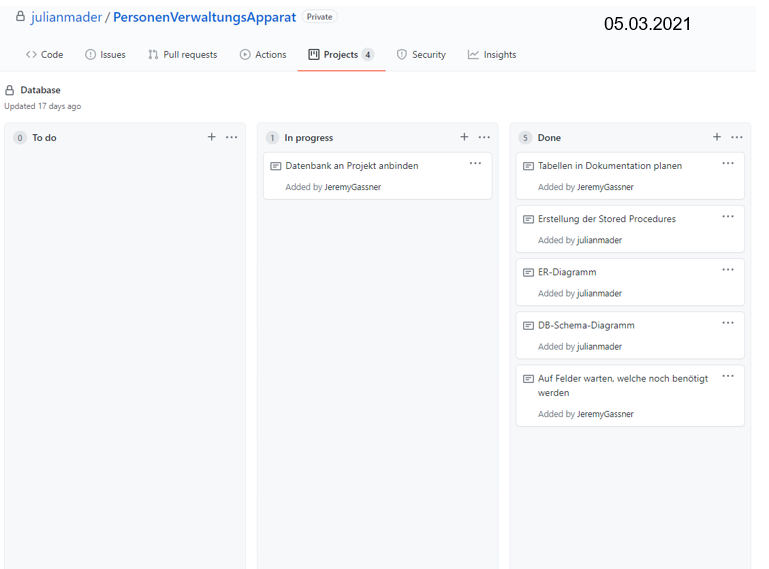
Sortiert nach Teilbereichen, anschließend nach Datum.

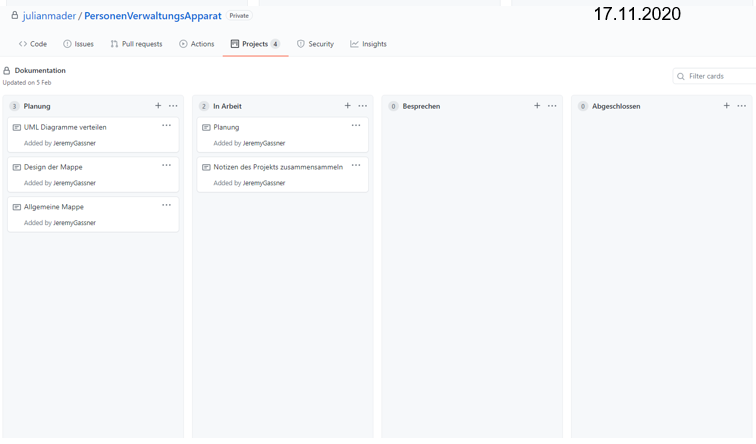


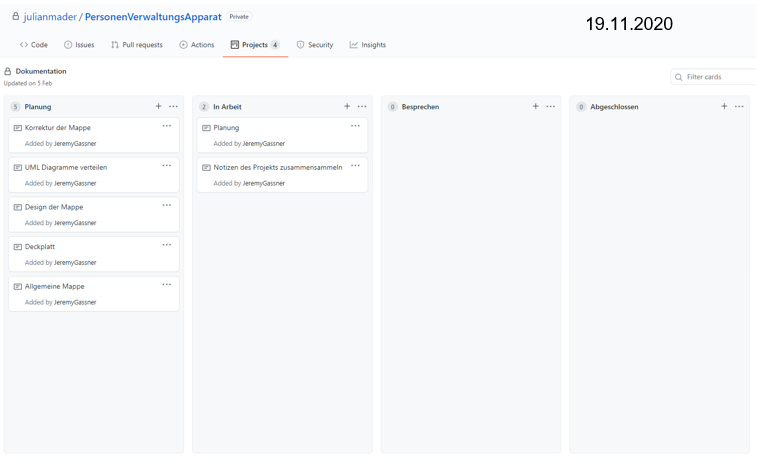


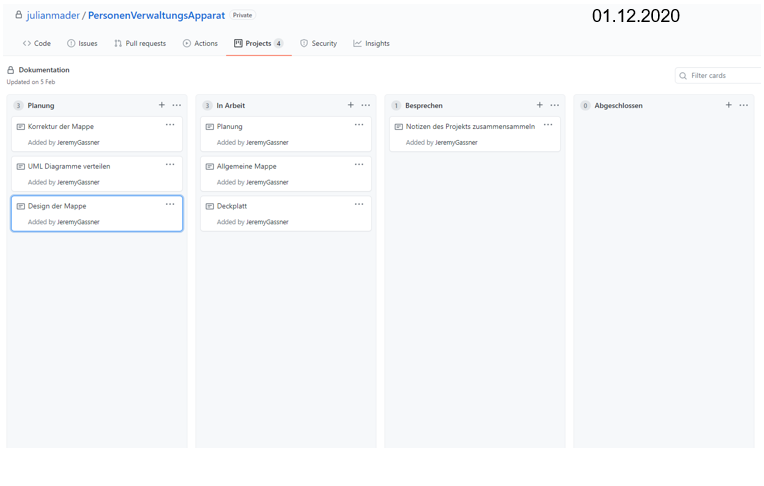


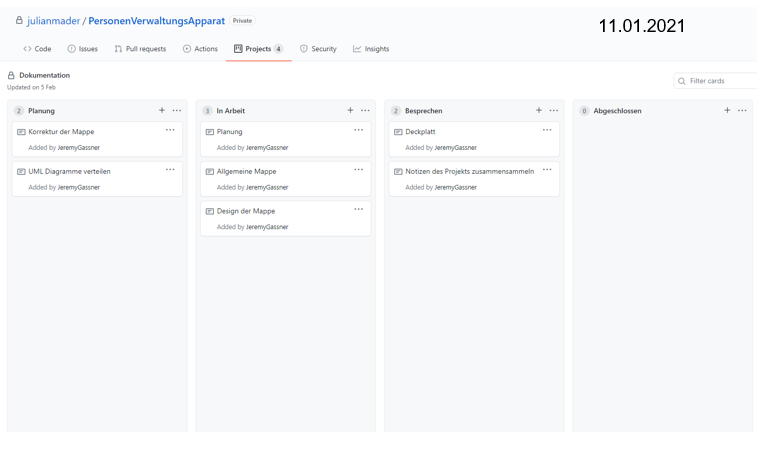


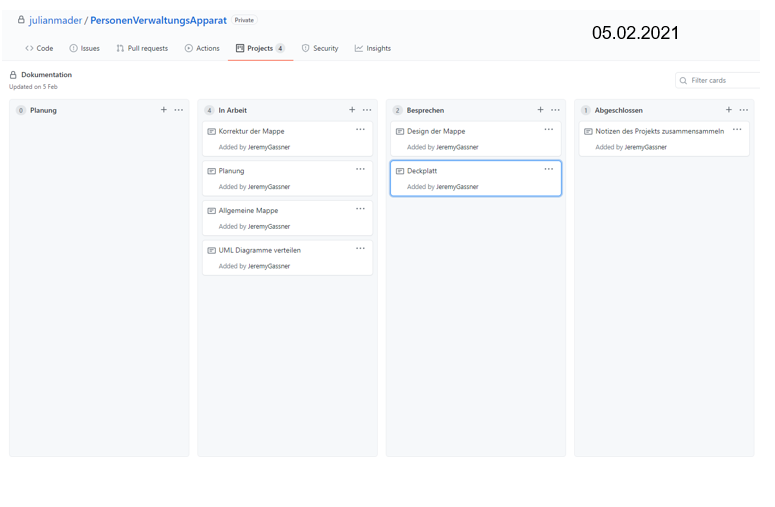


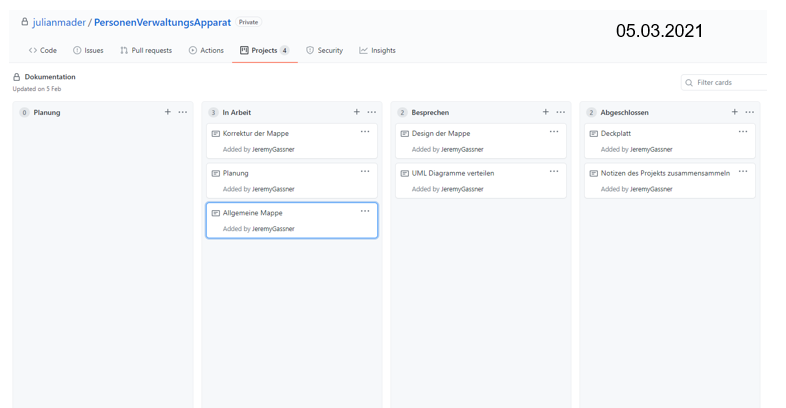


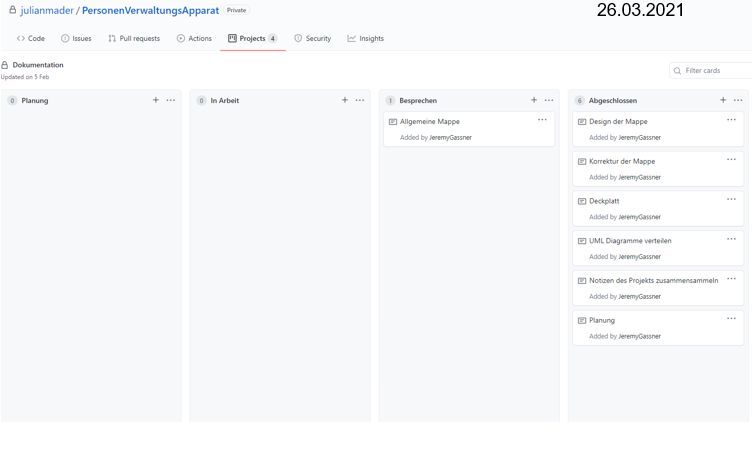


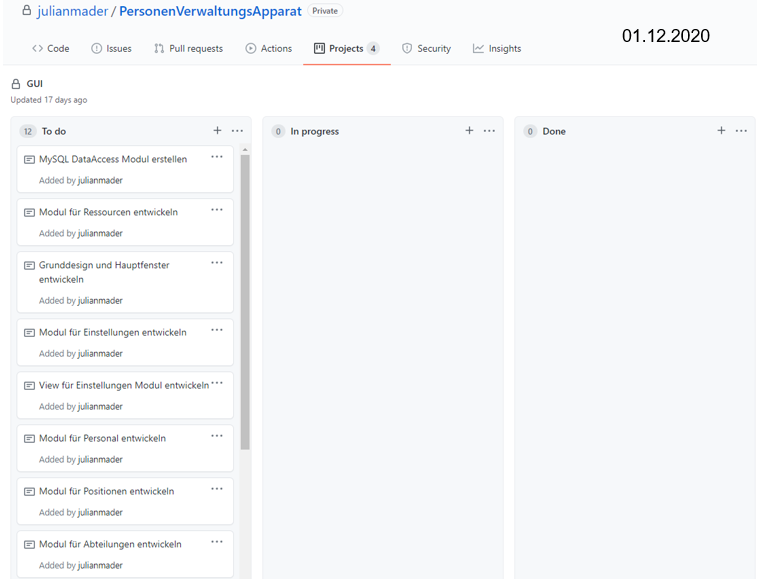


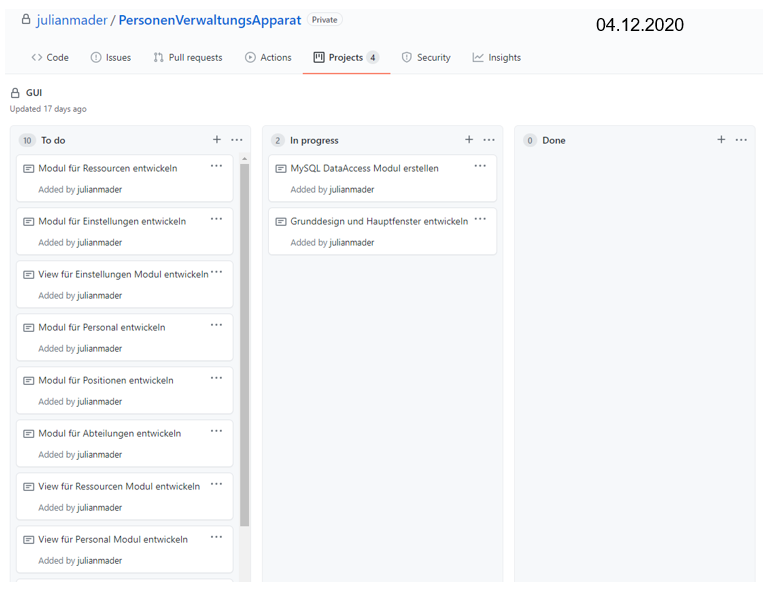


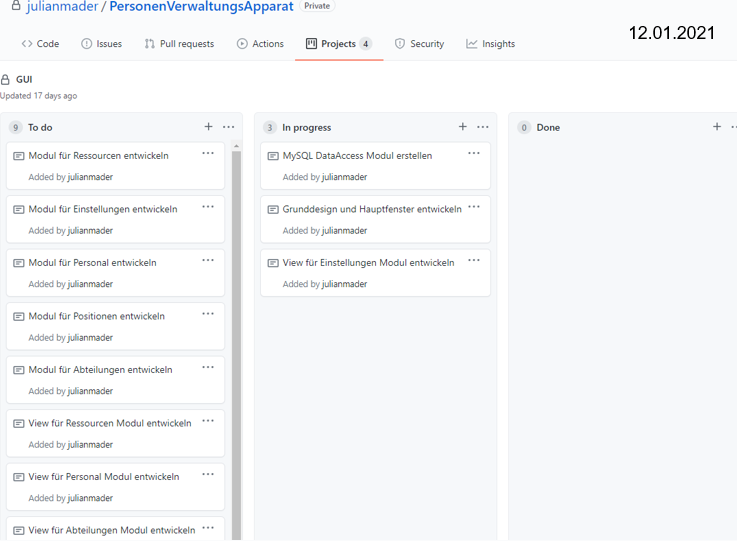


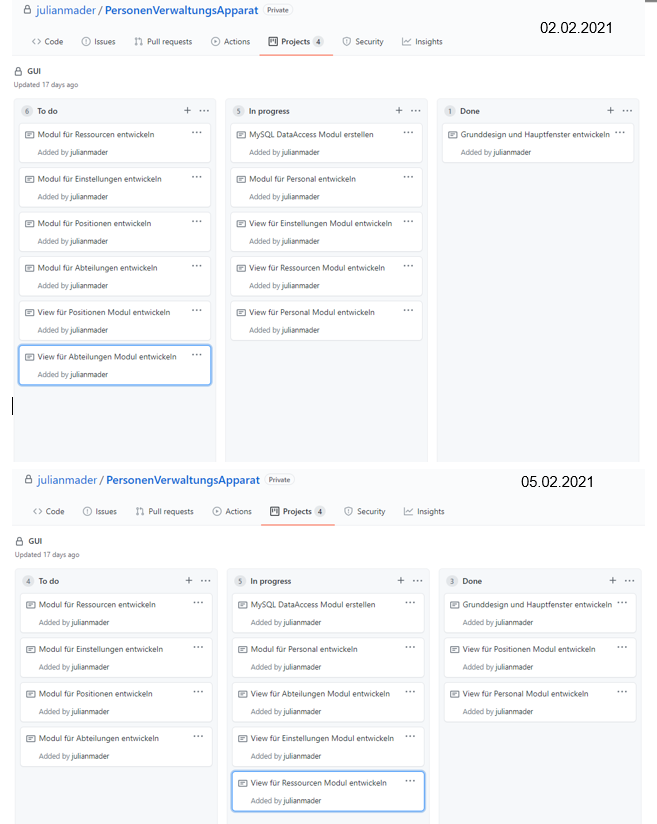
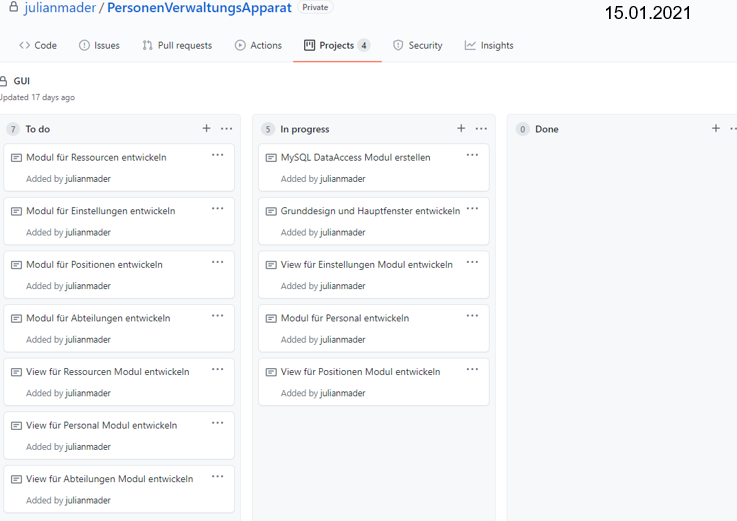


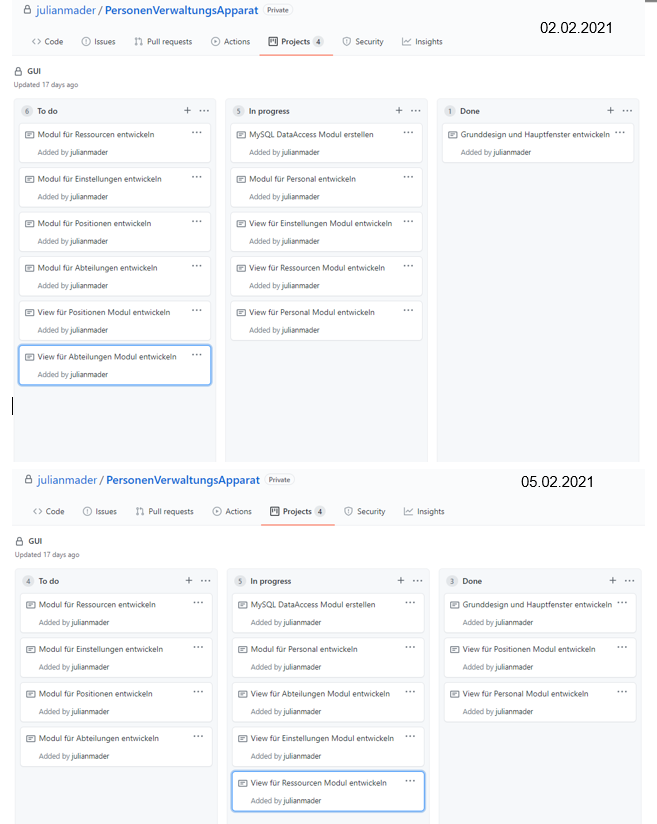


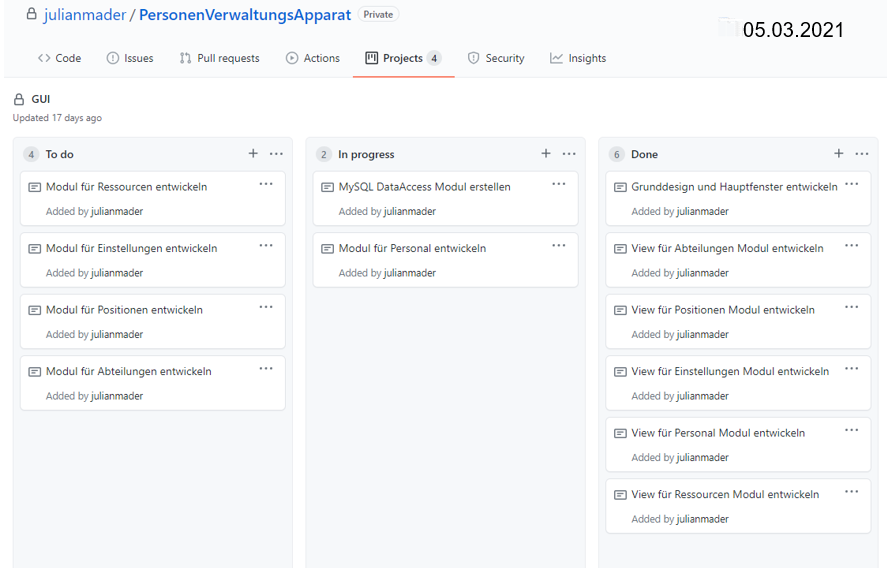


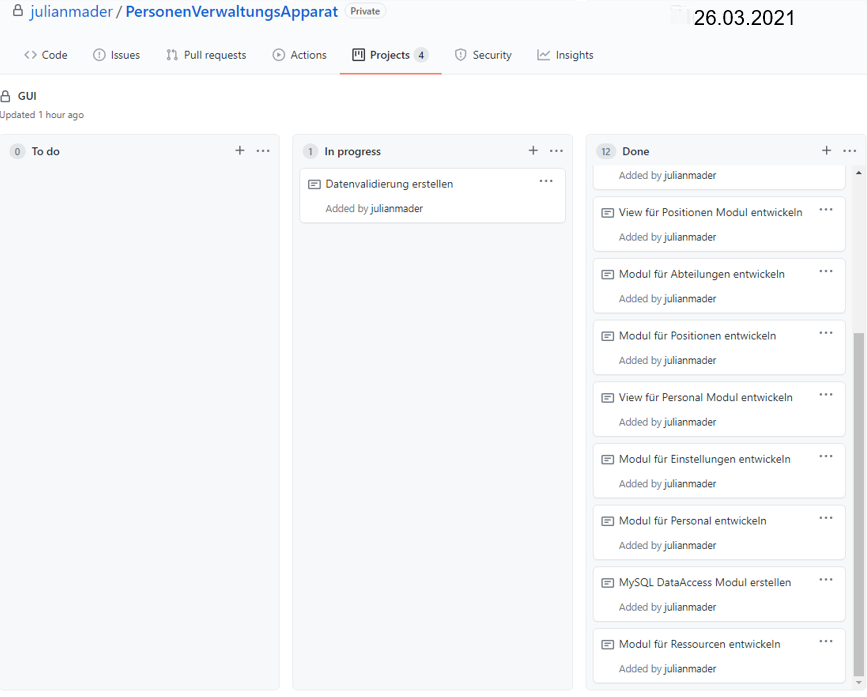


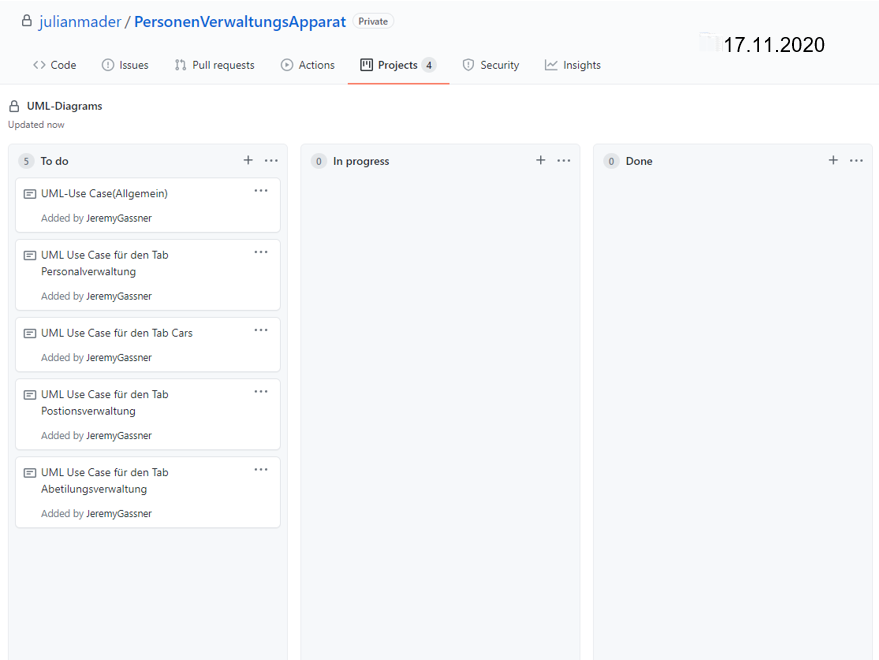


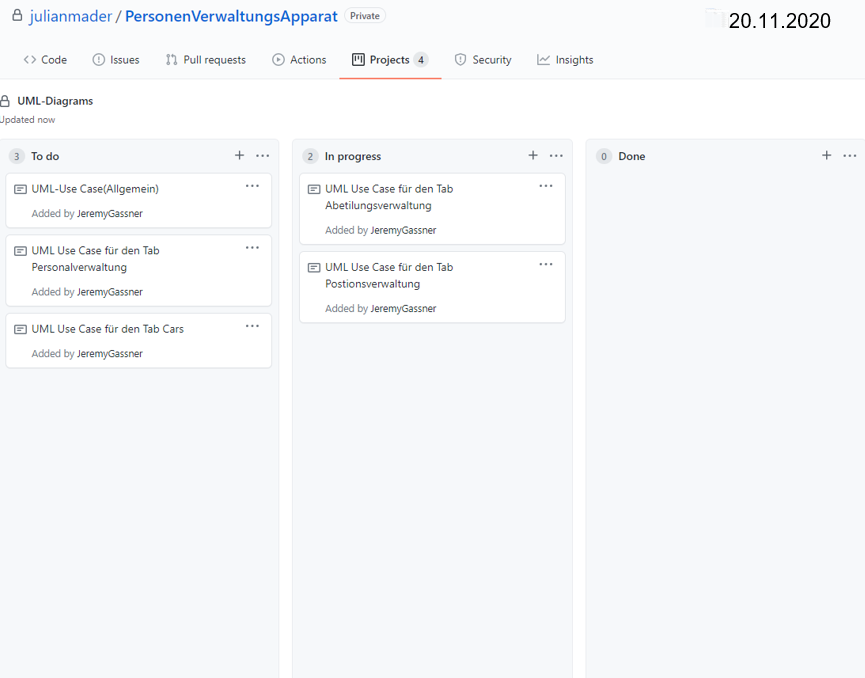


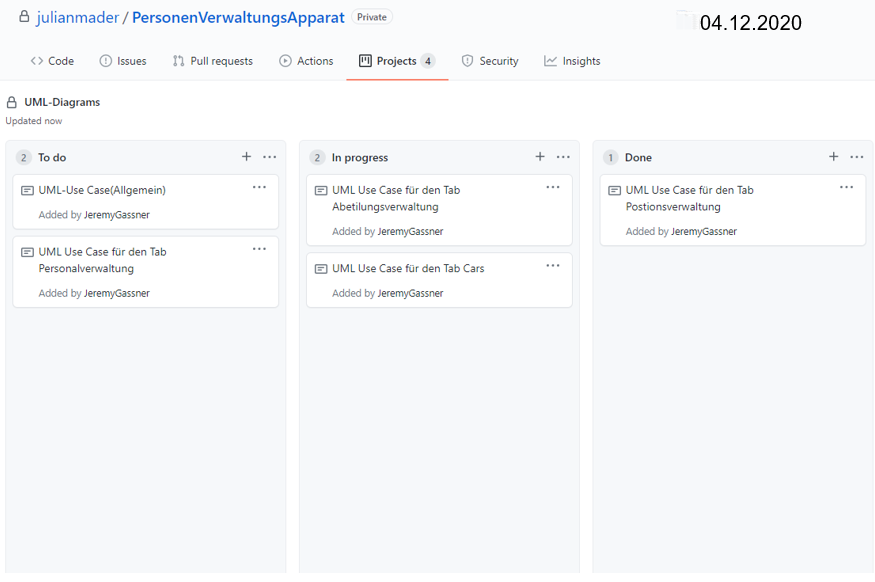


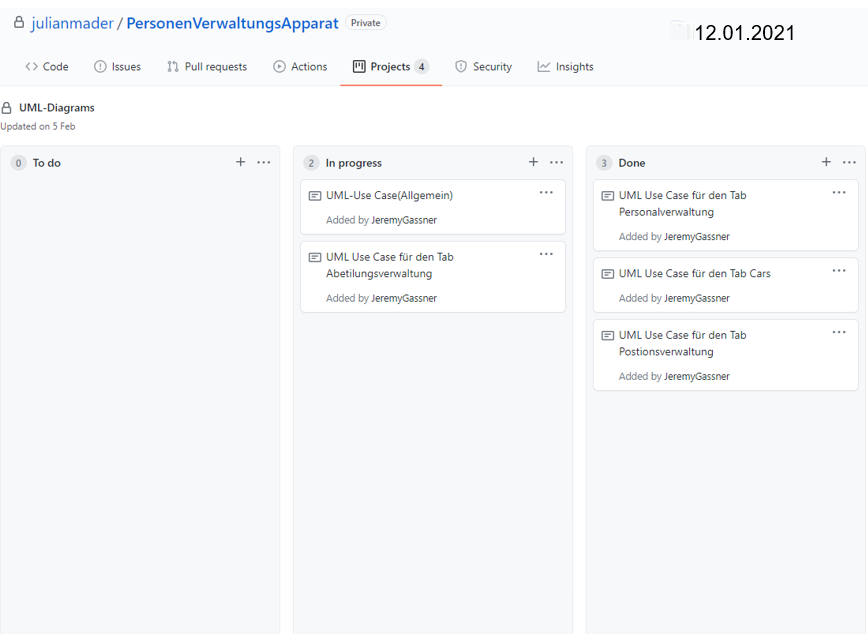


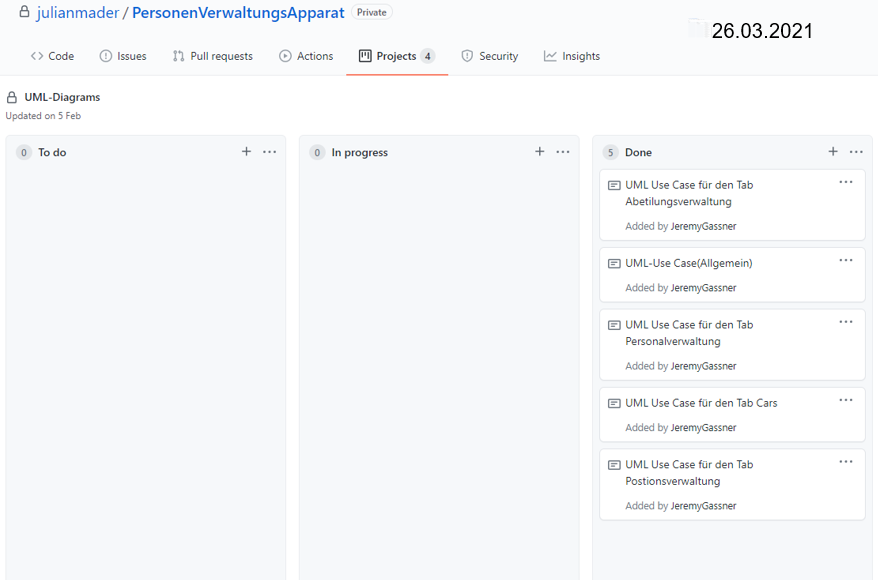






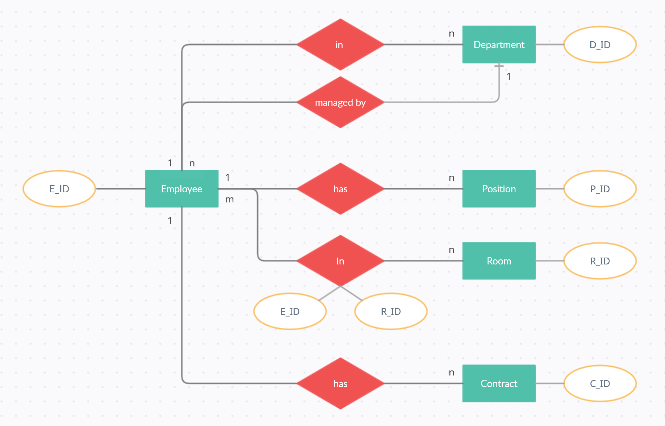




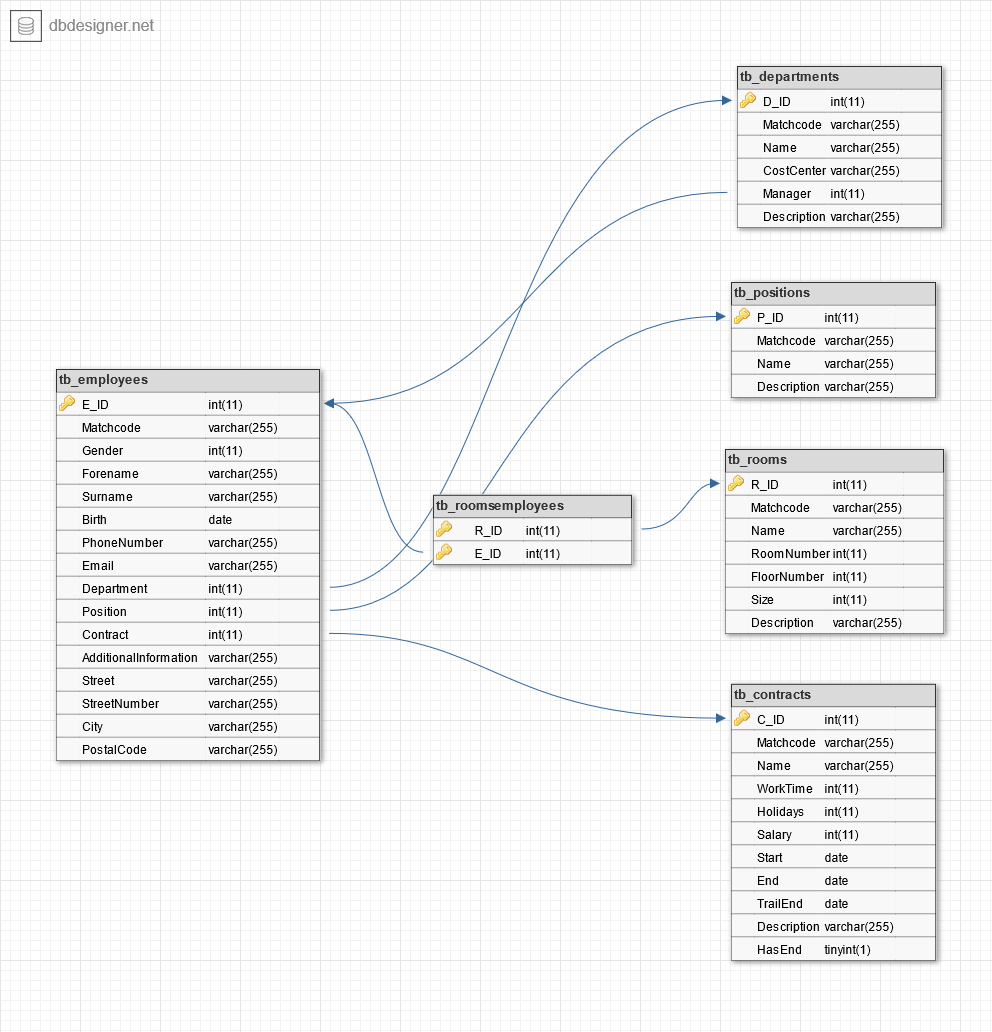


## Datenbank

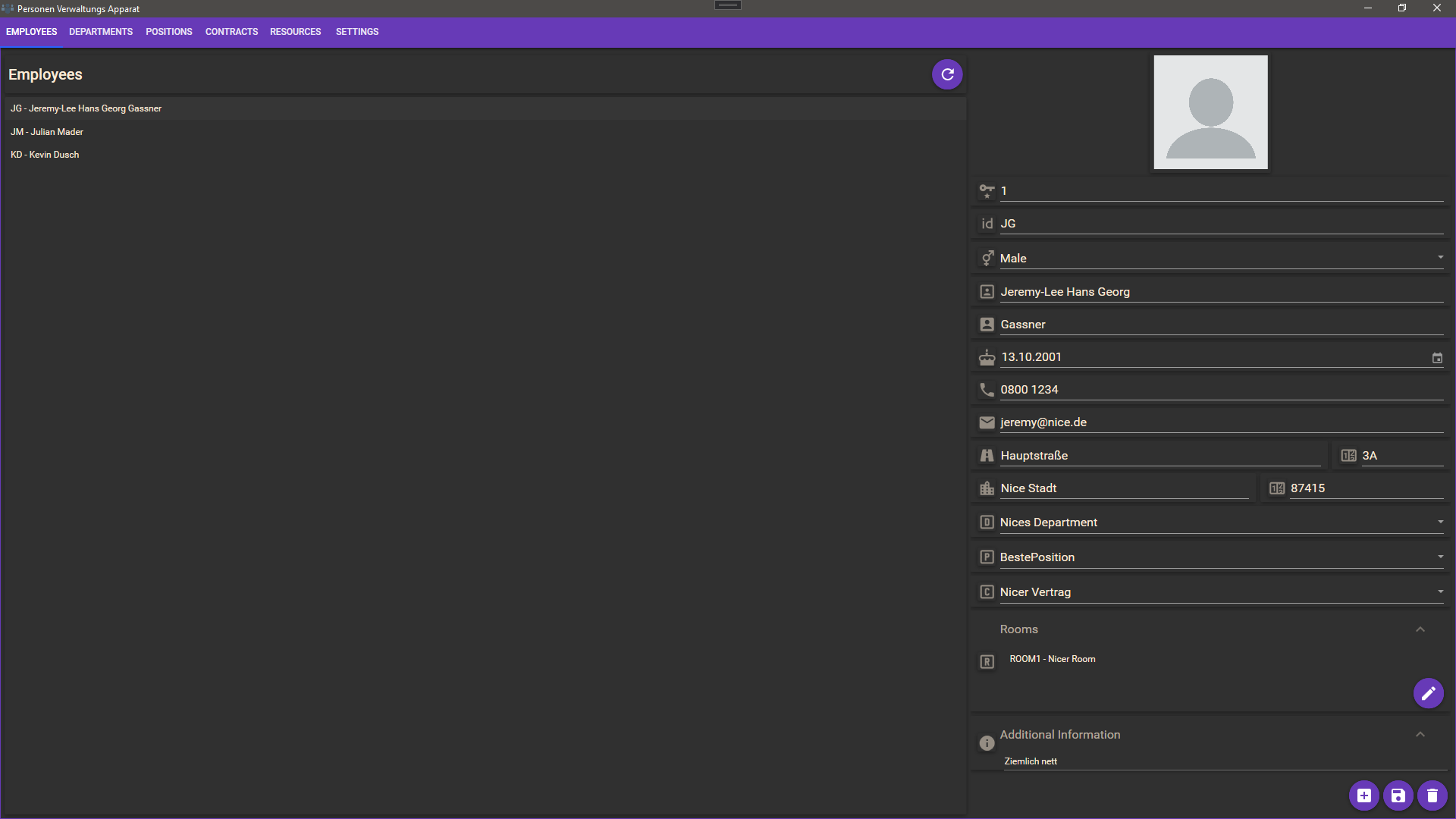
### ER-Diagramm

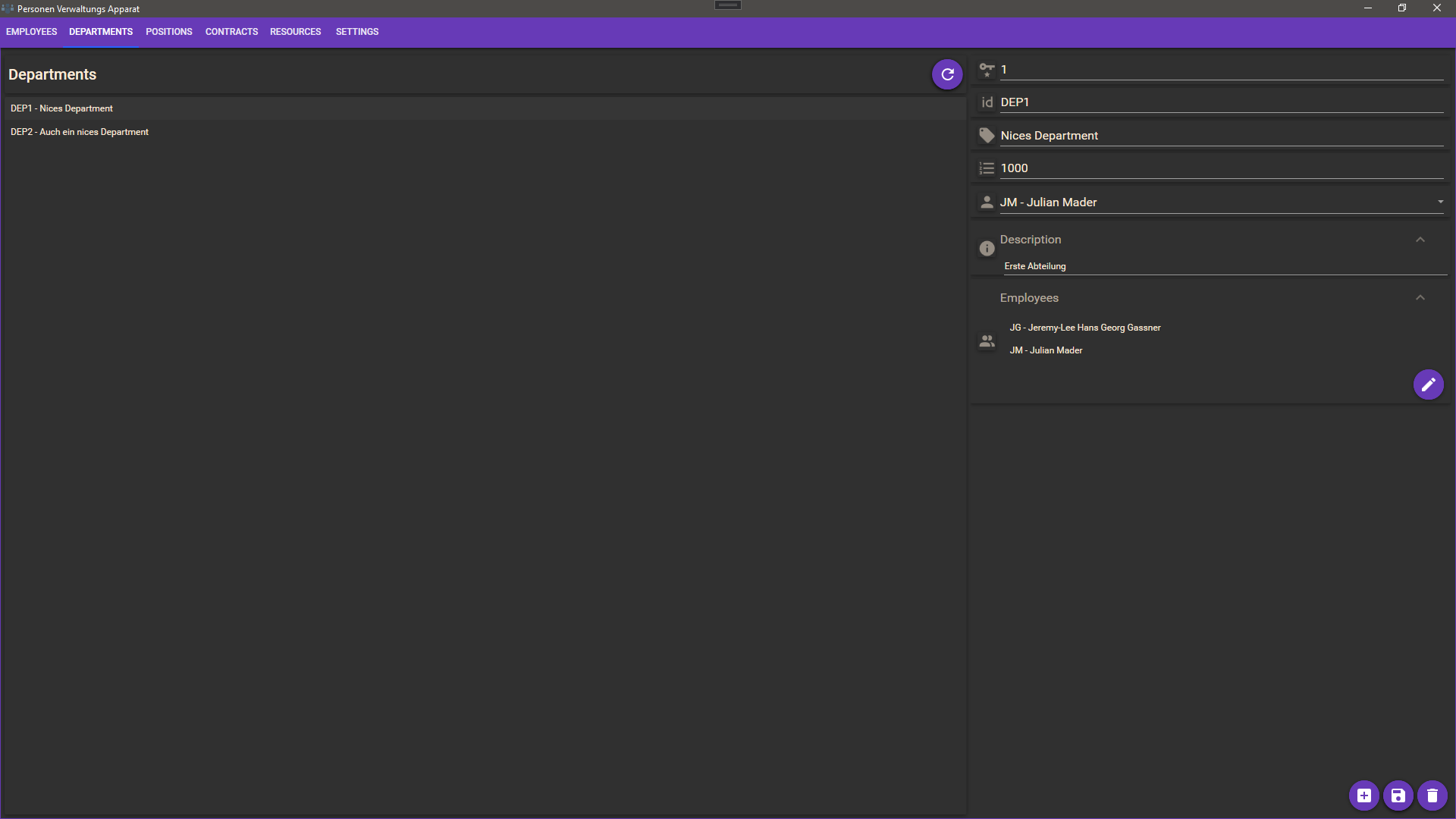


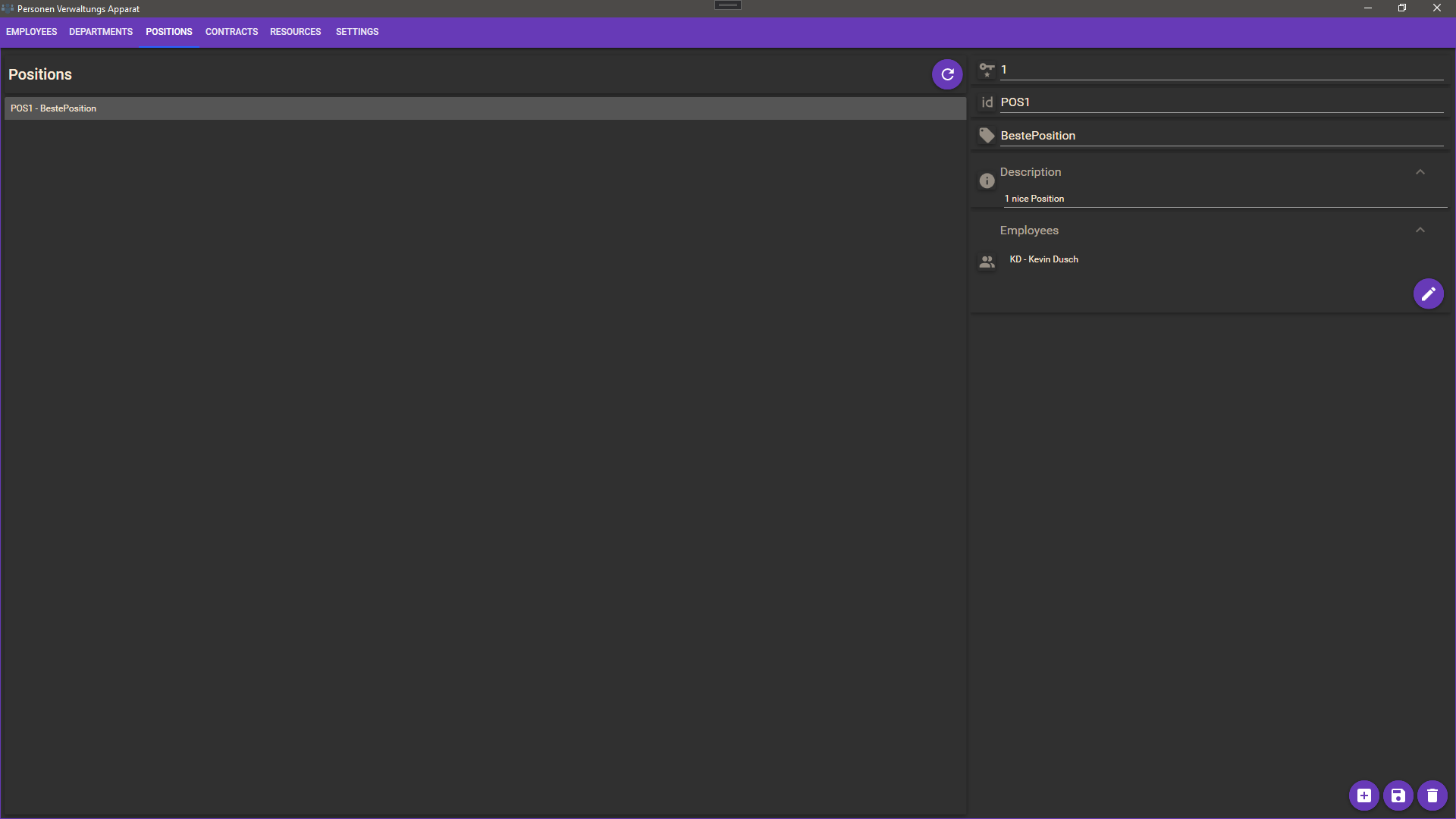
### Datenbankschema

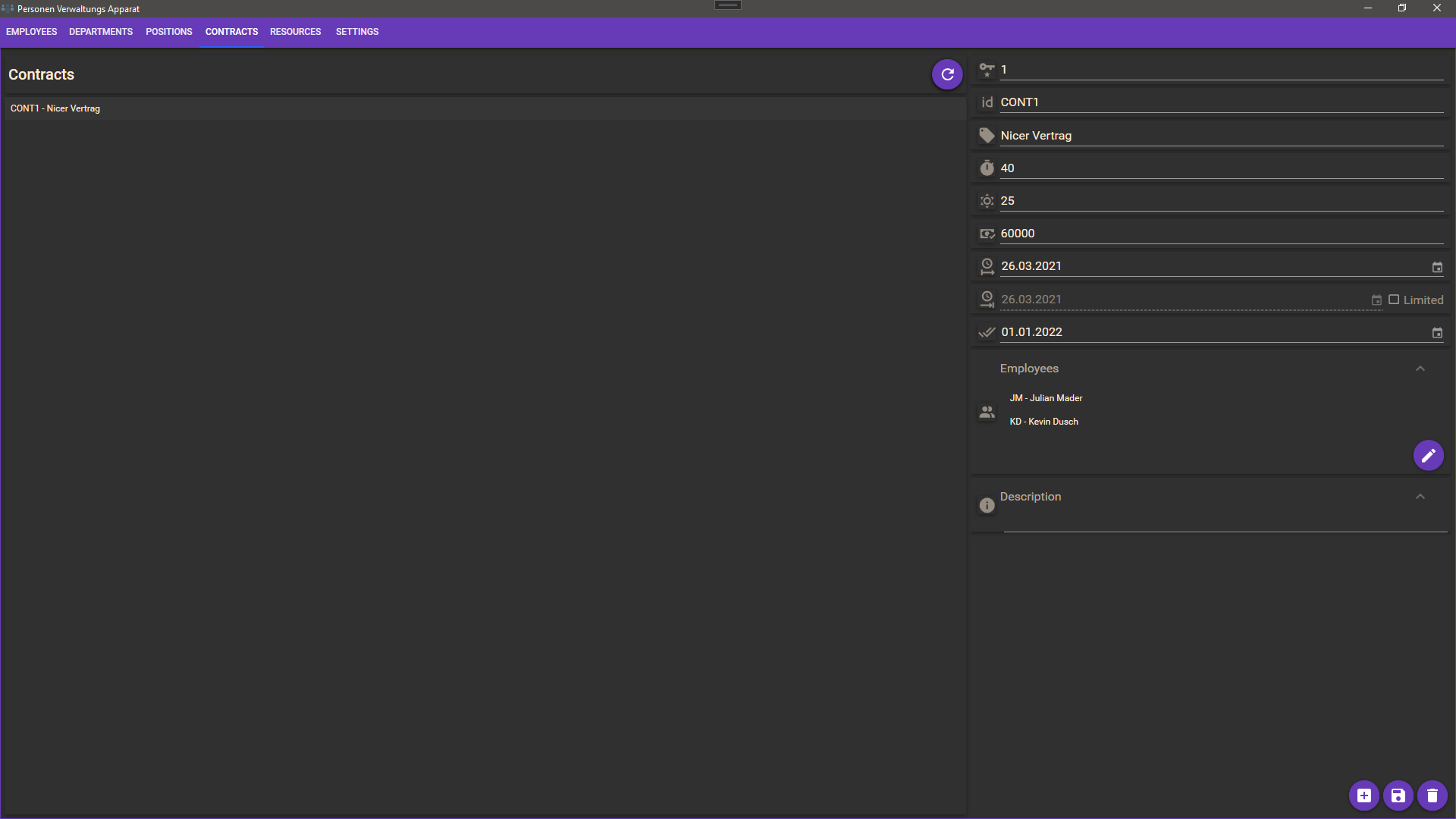


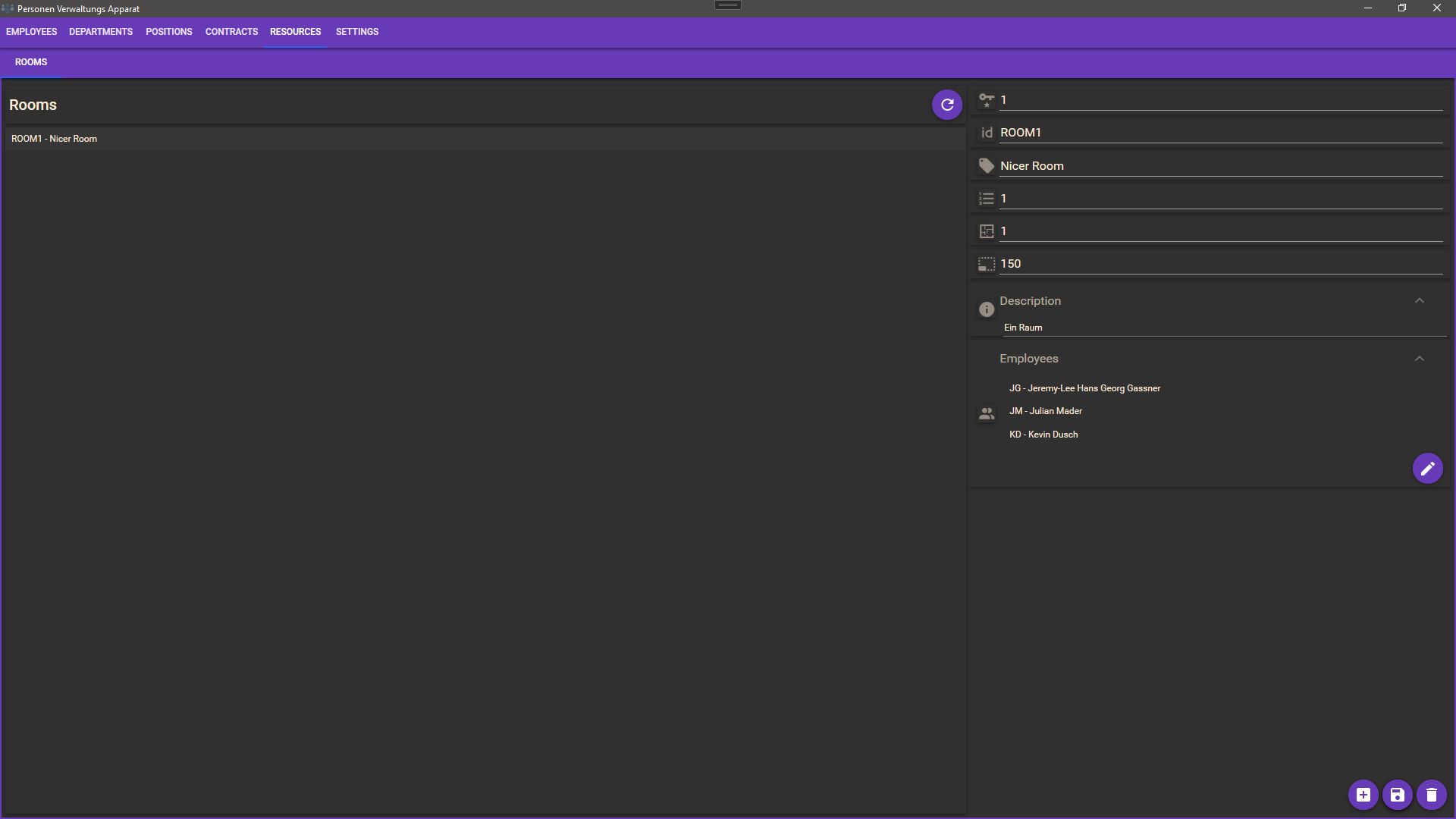
## Oberfläche



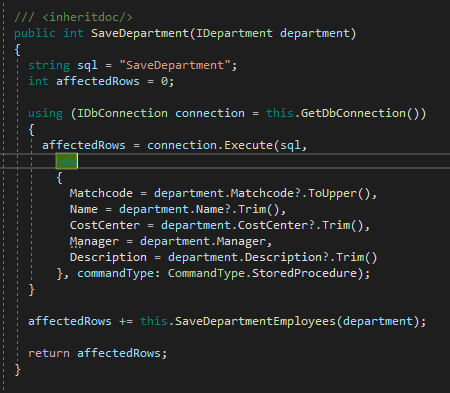




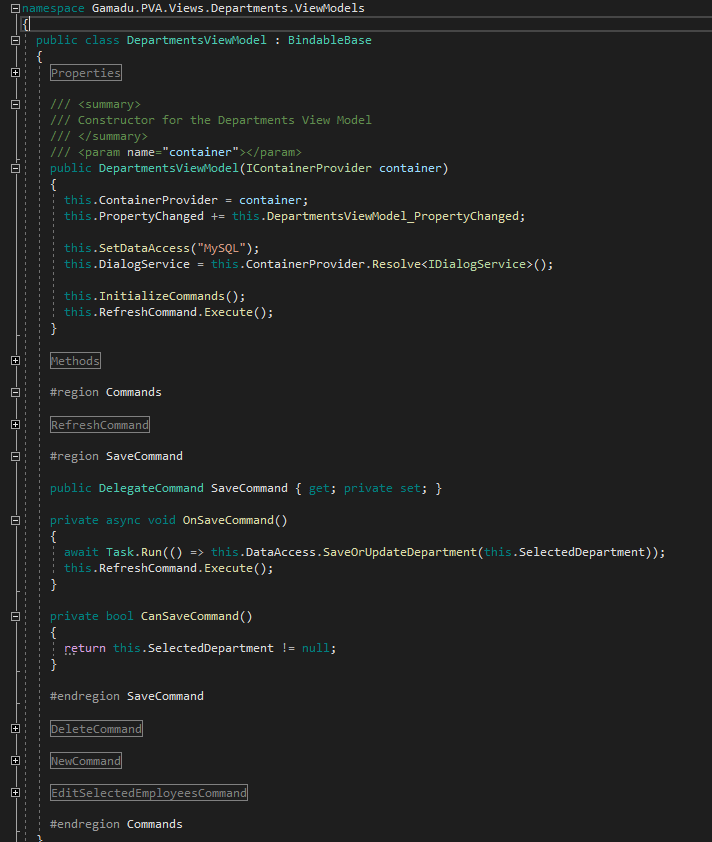




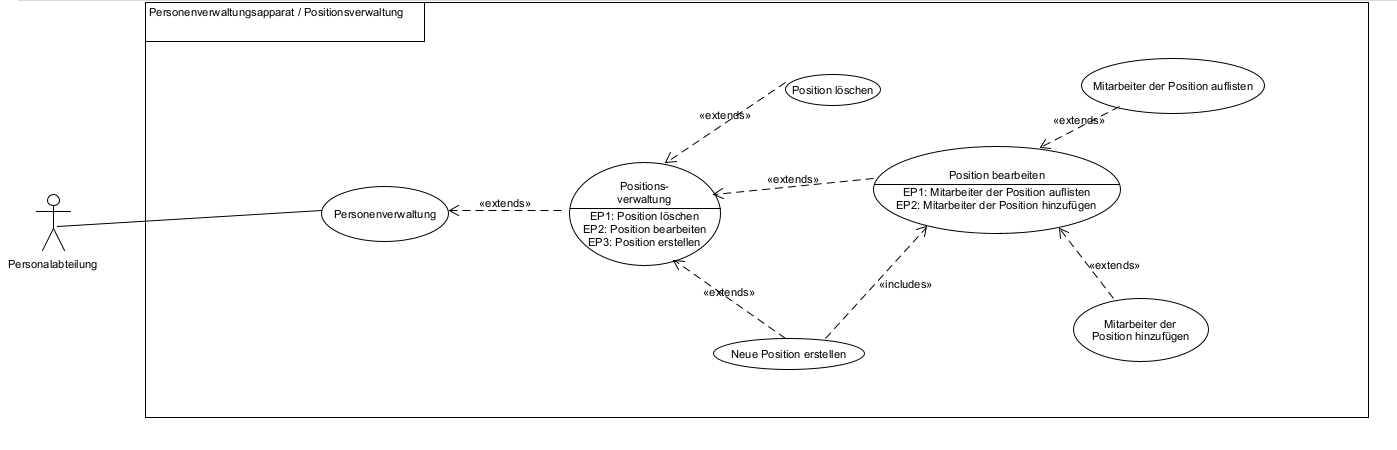
## Codeausschnitte

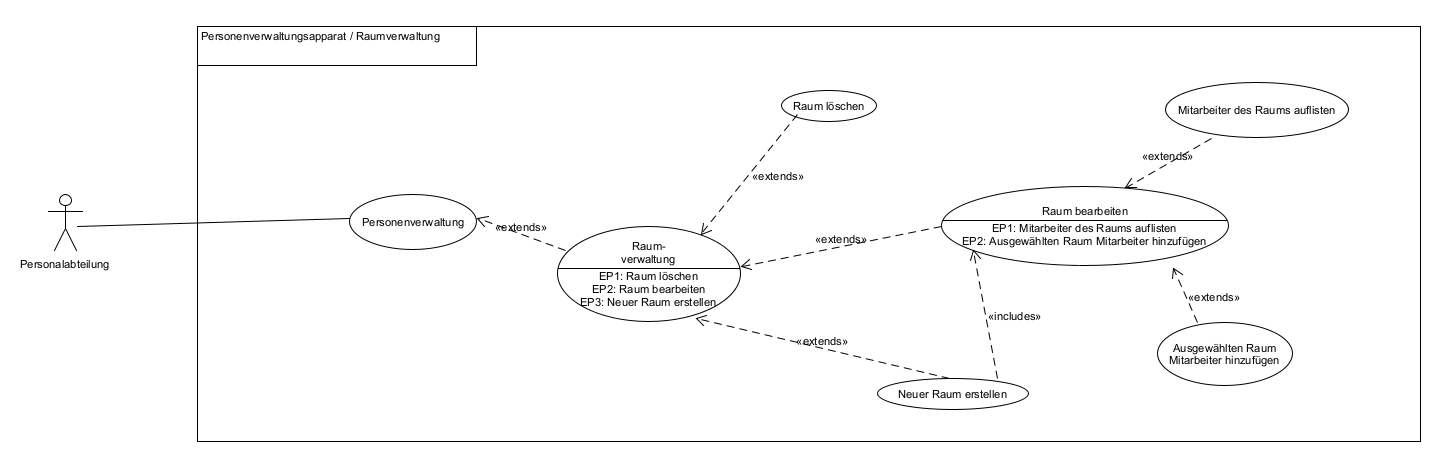


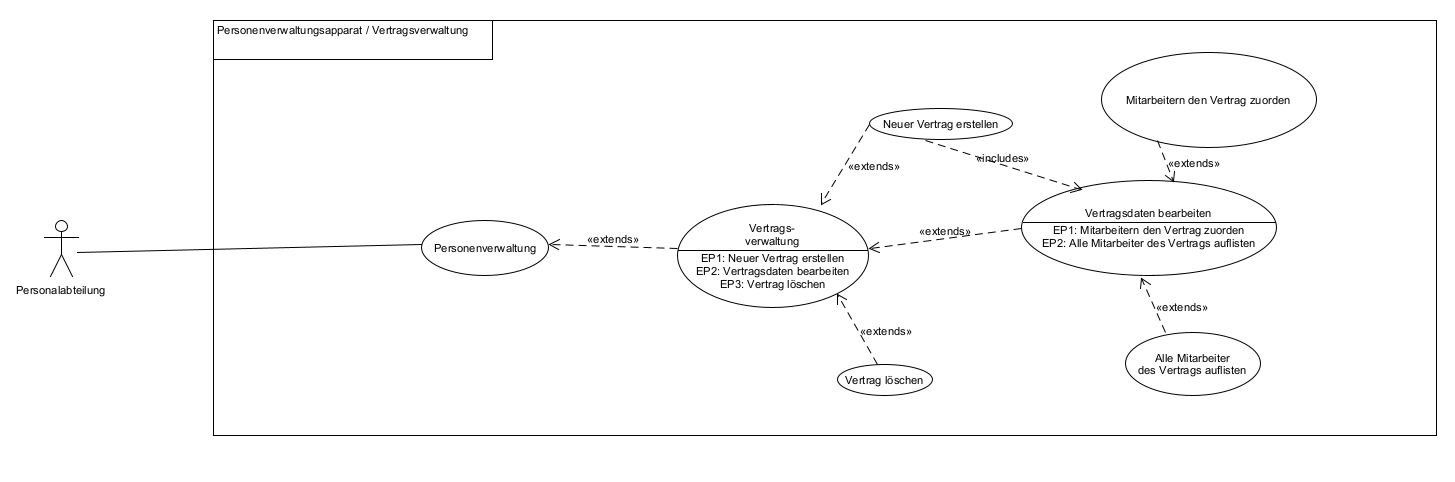


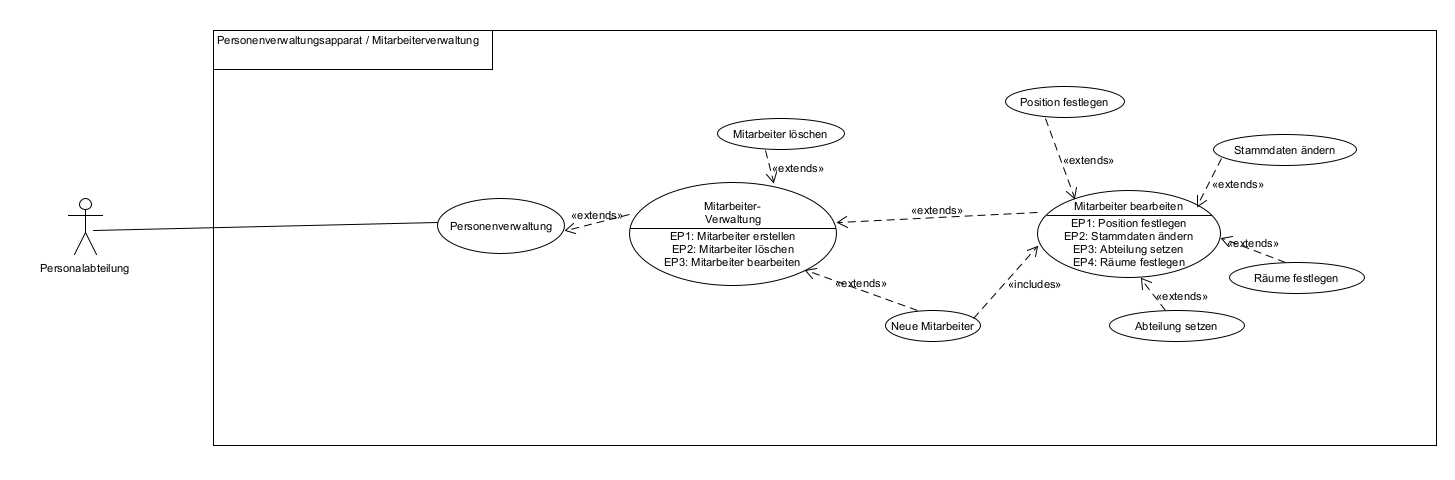


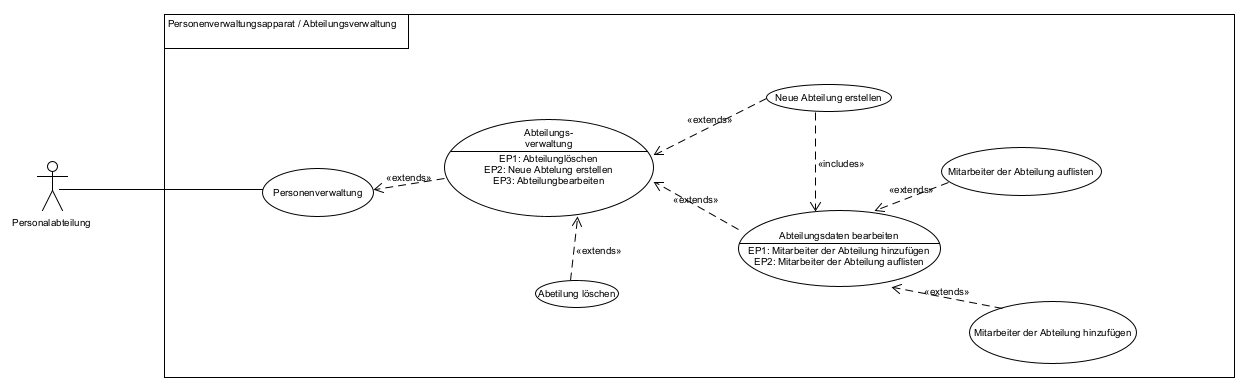
## Use Case Diagramme











1. https://prismlibrary.com/docs/index.html [↑](#footnote-ref-1)
2. https://de.wikipedia.org/wiki/Model\_View\_ViewModel [↑](#footnote-ref-2)
3. https://dapper-tutorial.net [↑](#footnote-ref-3)
4. https://de.wikipedia.org/wiki/CRUD [↑](#footnote-ref-4)