

Dokumentation

Software Praktikum – Gruppe 2

Erstellt von:

Perry Dettke  
Jana Drawert  
Rosalie Kripp  
Raphael Müller  
Jacqueline Schmid  
Kai Schmied

[1 Anforderungsanalyse 2](#_Toc107686735)

[Zielsystem 2](#_Toc107686736)

[2 Meilensteine 3](#_Toc107686737)

[Aufgabenverteilung 3](#_Toc107686738)

[Unterstützende Programme 4](#_Toc107686739)

[3 Use Case Diagramm 5](#_Toc107686740)

[Einloggen 5](#_Toc107686741)

[Registrieren 5](#_Toc107686742)

[Ausloggen 6](#_Toc107686743)

[Profil bearbeiten 6](#_Toc107686744)

[Profil löschen 6](#_Toc107686745)

[Projekt erstellen 6](#_Toc107686746)

[Projekt löschen 6](#_Toc107686747)

[Aktivität erstellen 6](#_Toc107686748)

[Aktivität löschen 6](#_Toc107686749)

[Arbeitszeitbuchung erstellen 6](#_Toc107686750)

[Kommen 6](#_Toc107686751)

[Gehen 6](#_Toc107686752)

[Pause 6](#_Toc107686753)

[Pause Beenden 7](#_Toc107686754)

[Sonderbuchung 7](#_Toc107686755)

[Arbeitszeitbuchung bearbeiten 7](#_Toc107686756)

[Arbeitszeitbuchung löschen 7](#_Toc107686757)

[Projektbuchung erstellen 7](#_Toc107686758)

[Projektbuchung bearbeiten 7](#_Toc107686759)

[Projektbuchung löschen 7](#_Toc107686760)

[Auswertung anzeigen 7](#_Toc107686761)

[4 Klassendiagramm 8](#_Toc107686762)

[BusinessObject: 8](#_Toc107686763)

[Person 9](#_Toc107686764)

[Projekt 10](#_Toc107686765)

[Aktivitäten 11](#_Toc107686766)

[Arbeitszeitkonto 12](#_Toc107686767)

[Buchung 13](#_Toc107686768)

[Ereignis 14](#_Toc107686769)

[PersonenProjekt 14](#_Toc107686770)

[Zeitintervall 15](#_Toc107686771)

[Mapper 16](#_Toc107686772)

[5 ER-Diagramm 17](#_Toc107686773)

[6 Installation der fertigen Anwendung 18](#_Toc107686774)

[Voraussetzungen 18](#_Toc107686775)

[Lokal 18](#_Toc107686776)

[Cloud 19](#_Toc107686777)

[7 Nutzerhandbuch 19](#_Toc107686778)

[Abbildungsverzeichnis 26](#_Toc107686779)

[Ehrenwörtliche Erklärung 27](#_Toc107686780)

# 1 Anforderungsanalyse

## Zielsystem

Die zu erstellende Software soll Studenten und Mitarbeiter der Hochschule der Medien die Möglichkeit bieten, ihre geleistete Zeit in einem Projekt zu dokumentieren. Hierbei muss jeder Anwender einen Account anlegen und einen Benutzernamen festlegen. In seinem Account kann er nun alle Projekte anschauen, bearbeiten und löschen, die er selbst angelegt hat. Ebenfalls sieht er alle Projekte, in welchen er beteiligt ist. Zusätzlich kann jede Person Aktivitäten für sein Projekt anlegen und geleistete Stunden auf sein Projekt buchen. Jede Person hat auch eine Übersicht über seine geleisteten Stunden und Urlaubs – und Krankheitstage in diesem Jahr.

Die Authentifizierung erfolgt über Google Firebase und die Anwendung soll über Google Cloud gehostet werden. Die App verfügt über eine Clientseite, geschrieben mit dem Framework React und eine Serverseite, die das Python Webframework Flask nutzt. Client und Server kommunizieren über eine REST-API. Die Daten werden in einer MySQL-Datenbank gespeichert und von dort aus abgerufen.

# 2 Meilensteine

Für den ersten Meilenstein galt es, sich in einer Gruppe aus 5-6 Personen zu finden und diese Zusammenstellung anzugeben. Die vorliegende Anwendung wurde von sechs Personen erstellt, deren Namen auf dem Deckblatt genannt sind.

Der zweite Meilenstein umfasste die Erstellung des Use-Case-Diagramms, das die verschiedenen Möglichkeiten abbildet, wie die Anwendung genutzt werden kann. Das aktuelle und angepasste Use-Case-Diagramm befindet sich unter 3.

Im Rahmen des dritten Meilensteins wurde dann das Klassen- und das Entity-Relationship-Diagramm erstellt. Diese repräsentieren die verwendeten Klassen der Anwendung und den Aufbau der Datenbank. Im Laufe der Entwicklung wurden beide Diagramme mehrmals angepasst und überarbeitet. Die aktuellen Versionen sind unter 4., beziehungsweise 5. Abgebildet.

Der vierte Meilensteil besteht aus der Abgabe des Peer-Feedbacks, das dazu dient, die eigenen Gruppenmitglieder hinsichtlich Mitwirkung, um Team einzuschätzen. Es dient vor allem dazu, den Gruppenmitgliedern Rückmeldung zu ihrer Performance zu geben und Probleme frühzeitig zu erkennen.

Der fünfte Meilenstein ist die Abgabe des vollständigen Source-Codes der Anwendung, sowie diese Dokumentation. Der Source-Code soll mit Kommentaren entsprechend umfangreich dokumentiert sein.

Die beiden letzten Meilensteine umfassen die Präsentation fertigen Zielsystems in der Cloud und das individuelle Fachgespräch. Diese werden hier in der Dokumentation nicht behandelt.

### Aufgabenverteilung

Die folgende Auflistung zeigt auf welche Aufgaben innerhalb des Teams eingeteilt wurden.

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

### Unterstützende Programme

Die folgende Auflistung zeigt auf welche Dienste für den Entwicklungsprozess der Applikation und bei der Organisation des Projektes genutzt wurden.

|  |  |
| --- | --- |
|  | Modellierung der Diagramme |
| StarUML | Zur Modellierung der Use Cases und Klassendiagramme |
| Draw.io | Tool für das Modellieren der ER-Diagramme |
| Balsamiq | Erstellung für ein Beispiel (Mockup) als Vorlage für die GUI |
|  | **Kommunikation und Organisation der Gruppenarbeit** | |
| WhatsApp | Gruppenkommunikation für Besprechungen und Anmerkungen | |
| Teams | Hier wurden alle Dateien gespeichert und die Kalenderfunktion als Meetingraum genutzt | |
| GitHub | Verwaltung des Projekts | |
| Miroboard | Aufteilung der Aufgaben und Notieren der nächsten Aufgaben | |
|  | **Entwicklung der Applikation** | |
| Visual Studio Code | Entwicklungsumgebung | |
| MySQL | Datenbank und MySQL Connector | |
| App Engine | Google App Engine für Deployment der Applikation auf Web Server | |

# 3 Use Case Diagramm

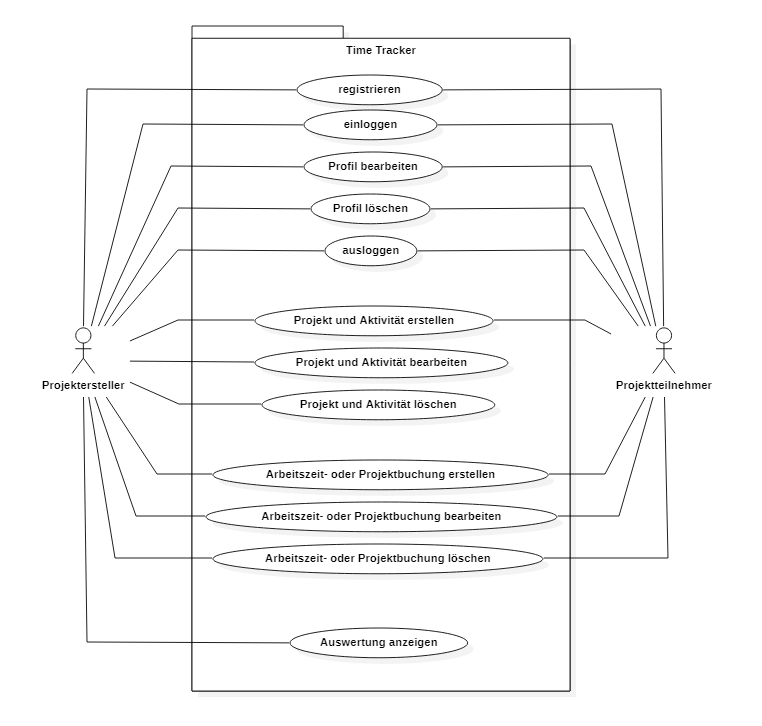


Abbildung 1: Use Case Diagramm / Eigene Darstellung

In Abbildung 1 ist das aktuelle Use Case Diagramm des Programms zu sehen. Es kennt zwei Akteure, den Projektersteller und den Projektbeteiligten. Der *Projektersteller* kennt insgesamt 12 Use Cases. Der *Projektteilnehmer* kennt insgesamt 11 Use Cases.

## Einloggen

Der Nutzer kann sich über FireBase einloggen. Die Aufforderung dazu erscheint automatisch, wenn kein FireBase-Token in den Cookies des Nutzers gefunden wird.

## Registrieren

Wenn ein Nutzer sich zum ersten Mal einloggt, wird seine FireBase-ID nicht in der Datenbank gefunden. Ist dies der Fall, wird er dazu aufgefordert, einen Account zu erstellen. Dabei muss er persönliche Daten angeben, wie zum Beispiel seinen Vor- und Nachnamen, seine E-mail und seinen Benutzername. Diese Daten werden dann zusammen mit seiner FireBase-ID in der Datenbank abgespeichert.

## Ausloggen

Wenn der Nutzer sich ausloggen will, muss er nur den entsprechenden Button anklicken. Er wird dann automatisch von FireBase ausgeloggt. Will er den Dienst wieder nutzen, muss er sich erneut einloggen.

## Profil bearbeiten

Will der User sein Profil anpassen, kann er auf den entsprechenden Button klicken. Daraufhin hat er die Möglichkeit, die Werte anzupassen und die alten Daten zu überschreiben.

## Profil löschen

Der Nutzer hat die Möglichkeit seinen Account zu löschen. Die Einträge werden aus der Datenbank entfernt und wenn der Nutzer die Webseite wieder öffnet, muss er erneut durch den Prozess der Anmeldung.

## Projekt erstellen

Der Nutzer hat die Möglichkeit ein neues Projekt anzulegen. Diese Daten werden dann in der Datenbank abgespeichert.

Projekt bearbeiten

Will der User sein Projekt anpassen, kann er auf den entsprechenden Button klicken. Daraufhin hat er die Möglichkeit, die Werte anzupassen und die alten Daten zu überschreiben.

## Projekt löschen

Der Nutzer hat die Möglichkeit sein Projekt zu löschen. Hierbei werden auch die dazugehörigen Aktivitäten gelöscht. Die Einträge werden aus der Datenbank entfernt.

## Aktivität erstellen

Der Nutzer hat die Möglichkeit eine neue Aktivität zu seinem Projekt anzulegen. Diese Daten werden dann in der Datenbank abgespeichert.

Aktivität bearbeiten

Will der User eine Aktivität anpassen, kann er auf den entsprechenden Button klicken. Daraufhin hat er die Möglichkeit, die Werte anzupassen und die alten Daten zu überschreiben.

## Aktivität löschen

Der Nutzer hat die Möglichkeit eine Aktivität zu löschen. Die Einträge werden aus der Datenbank entfernt.

## Arbeitszeitbuchung erstellen

Der Nutzer kann verschiedene Buchungen erstellen.

### Kommen

Beginnt der Nutzer seinen Arbeitstag, klickt er auf den Button Kommen. Dieser Zeitstempel wird dann in der Datenbank abgespeichert.

### Gehen

Beendet der Nutzer seinen Arbeitstag, klickt er auf den Button Gehen. Dieser Zeitstempel wird dann in der Datenbank abgespeichert.

### Pause

Beginnt der Nutzer seine Pause, klickt er auf den Button Pause. Dieser Zeitstempel wird dann in der Datenbank abgespeichert.

### Pause Beenden

Beendet der Nutzer seine Pause, klickt er auf den Button Pause Beenden. Dieser Zeitstempel wird dann in der Datenbank abgespeichert.

### Sonderbuchung

Hier kann der User eine Sonderbuchung wie Urlaub oder Krankheit tätigen. Diese Daten werden dann in der Datenbank abgespeichert.

## Arbeitszeitbuchung bearbeiten

Der Nutzer hat die Möglichkeit seine Arbeitszeitbuchungen zu bearbeiten hierbei kann er auf den entsprechenden Button klicken. Daraufhin hat er die Möglichkeit, die Werte anzupassen und die alten Daten zu überschreiben.

## Arbeitszeitbuchung löschen

Will der User eine Arbeitszeitbuchung löschen, kann er auf den entsprechenden Button klicken. Die Einträge werden aus der Datenbank entfernt.

## Projektbuchung erstellen

Der Nutzer hat die Möglichkeit ein neue Zeitbuchung für eine Aktivität anzulegen. Diese Daten werden dann in der Datenbank abgespeichert.

## Projektbuchung bearbeiten

Will der User seine Projektbuchung anpassen, kann er auf den entsprechenden Button klicken. Daraufhin hat er die Möglichkeit, die Werte anzupassen und die alten Daten zu überschreiben.

## Projektbuchung löschen

Will der User eine Projektbuchung löschen, kann er auf den entsprechenden Button klicken. Die Einträge werden aus der Datenbank entfernt.

## Auswertung anzeigen

Nur der Projektersteller kann sich die Auswertungsseite anzeigen lassen.

# 4 Klassendiagramm

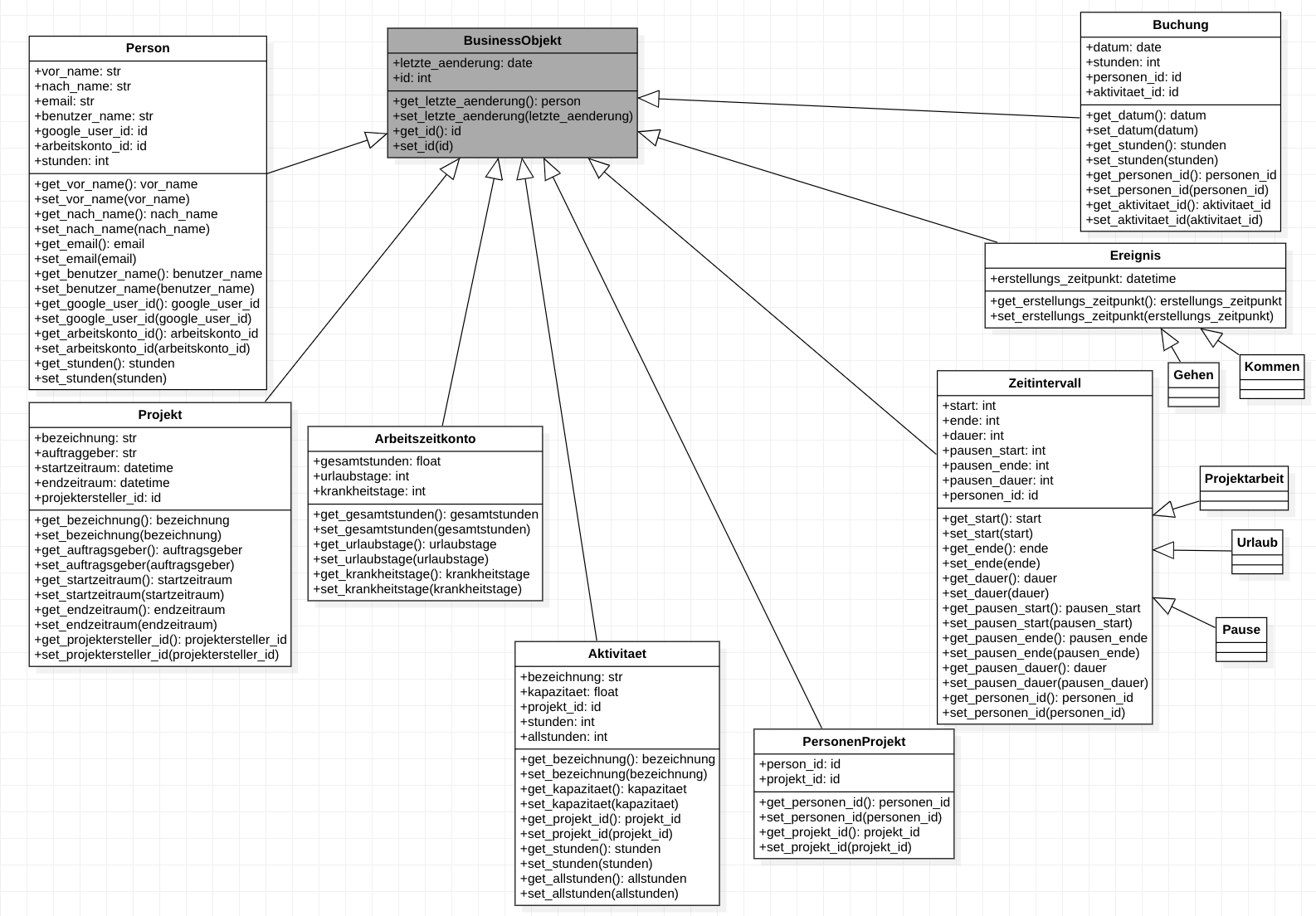


Abbildung 2: Klassendiagramm / Eigene Darstellung

## BusinessObject:

Die BusinessObject Klasse ist eine Abstrakte Klasse, von der sämtliche anderen Klassen erben. Sie enthält lediglich zwei Variablen, nämlich die ID und die Erstellungszeit, sowie die zugehörigen Getter und Setter.

**Attribute:**

*BusinessObject.*\_id

Enthält die ID des Objekts. Die ID wird von der Datenbank beim ersten Schreiben des Objektes vergeben und danach immer mit ausgelesen.

*BusinessObject.*\_letzte\_aenderung

Enthält den Zeitpunkt der Erstellung des Objekts. In der Praxis wird dieser Zeitpunkt beim ersten Schreiben des Objekts in die Datenbank festgehalten und danach immer mit ausgelesen.

**Methoden:**

*BusinessObject.*get\_id()

Gibt die ID des Objektes aus.

*BusinessObject.*set\_id(id)

Schreibt die ID.

*BusinessObject.*get\_letzte\_aenderung()

Gibt die Erstellungszeit aus.

*BusinessObject.*set\_letzte\_aenderung (letzte\_aenderung)

Schreibt die Erstellungszeit.

## Person

Die Klasse Person bildet einen Nutzer ab. Sie hält dafür Informationen wie den Vor- und Nachnamen, die E-mail, den Benutzername, seine ArbeitszeitkontoID, seine Arbeitsstunden sowie einen Verweis auf das zugehörige Profil. Außerdem erbt die Klasse von der abstrakten Klasse BusinessObject.

**Attribute:**

*Person*.\_vor\_name

Enthält den Vornamen des Nutzers.

*Person*.\_nach\_name

Enthält den Nachnamen des Nutzers.

*Person*.\_email

Enthält das Geburtsdatum des Nutzers.

*Person*.\_benutzer\_name

Enthält das Fachsemester des Nutzers.

*Person*.\_google\_user\_id

Enthält die ID des zugehörigen Profils.

Person.\_arbeitszeitkonto\_id

Enthält die ID des zugehörigen Arbeitszeitskontos.

Person.\_stunden

Enthält die Arbeitsstunden des Nutzers.

**Methoden:**

*Person*.get\_vor\_name()

Gibt den Vornamen aus.

*Person*.set\_vor\_name(vor\_name)

Schreibt den Vornamen.

*Person*.get\_nach\_name()

Gibt den Nachnamen aus.

*Person*.set\_nach\_name(nach\_name)

Schreibt den Nachnamen.

*Person*.get\_benutzer\_name()

Gibt den Benutzername aus.

*Person*.set\_benutzer\_name(benutzer\_name)

Schreibt den Benutzername.

*Person*.get\_email()

Gibt die E-Mail aus.

*Person*.set\_email(email)

Schreibt die E-Mail.

*Person*.get\_google\_user\_id()

Gibt die ID des zugehörigen Profils aus.

*Person*.set\_google\_user\_id(google\_user\_id)

Schreibt die ID des zugehörigen Profils.

*Person*.get\_arbeitszeitkonto\_id()

Gibt die ID des zugehörigen Arbeitszeitkontos aus.

*Person*.set\_ arbeitszeitkonto\_id (arbeitszeitkonto\_id)

Schreibt die ID des zugehörigen Arbeitszeitkontos.

*Person*.get\_stunden()

Gibt die Stunden aus.

*Person*.set\_stunden(stunden)

Schreibt die Stunden.

## Projekt

Die Klasse Projekt bildet alle Projekte vom Nutzer ab. Sie hält dafür Informationen wie die Bezeichnung, den Auftraggeber, den Startzeitraum, den Endzeitraum sowie einen Verweis auf den zugehörigen Projektersteller. Außerdem erbt die Klasse von der abstrakten Klasse BusinessObject.

**Attribute:**

*Projekt*.\_bezeichnung

Enthält die Bezeichnung des Projekts.

*Projekt*.\_auftraggeber

Enthält den Auftraggeber des Projekts.

*Projekt*.\_startzeitraum

Enthält den Startzeitraum des Projekts.

*Projekt*.\_endzeitraum

Enthält den Endzeitraum des Projekts.

*Projekt*.\_projektersteller\_id

Enthält die ID des zugehörigen Projekterstellers.

**Methoden:**

*Projekt*.get\_bezeichnung

Gibt die Bezeichnung aus.

*Projekt*.set\_bezeichnung(bezeichnung)

Schreibt die Bezeichnung.

*Projekt*.get\_auftraggeber

Gibt den Auftraggeber aus.

*Projekt*.set\_ auftraggeber (auftraggeber)

Schreibt den Auftraggeber.

*Projekt*.get\_startzeitraum

Gibt den Startzeitraum aus.

*Projekt*.set\_startzeitraum(startzeitraum)

Schreibt den Startzeitraum.

*Projekt*.get\_endzeitraum

Gibt den Endzeitraum aus.

*Projekt*.set\_endzeitraum(endzeitraum)

Schreibt den Endzeitraum.

*Projekt*.get\_projekterstellerID

Gibt die ProjekterstellerID aus.

*Projekt*.set\_projekterstellerID(projekterstellerID)

Schreibt die Projektersteller ID.

## Aktivitäten

Die Klasse Aktivitäten bildet alle Aktivitäten in einem Projekt ab. Sie hält dafür Informationen wie die Bezeichnung, die Kapazität, die Stunden, alle Stunden sowie einen Verweis auf das zugehörige Projekt. Außerdem erbt die Klasse von der abstrakten Klasse BusinessObject.

**Attribute:**

Aktivitaet.\_bezeichnung

Enthält die Bezeichnung der Aktivität.

Aktivitaet.\_kapazitaet

Enthält die Kapazität der Aktivität.

Aktivitaet.\_projekt\_id

Enthält die ID des zugehörigen Projekts.

Aktivitaet.\_stunden

Enthält die Arbeitsstunden der Aktivität.

Aktivitaet.\_allstunden

Enthält alle Stunden der Aktivität.

**Methoden:**

*Aktivitaet*.get\_bezeichnung()

Gibt die Bezeichnung aus.

*Aktivitaet*.set\_bezeichnung(bezeichnung)

Schreibt die Bezeichnung.

*Aktivitaet*.get\_kapazitaet ()

Gibt die Kapazitaet aus.

*Aktivitaet*.set\_kapazitaet (kapazitaet)

Schreibt die Kapazitaet.

*Aktivitaet*.get\_projektID()

Gibt die ProjektID aus.

*Aktivitaet*.set\_projektID (projektID)

Schreibt die Projekt ID.

*Aktivitaet*.get\_stunden()

Gibt die Stunden aus.

*Aktivitaet*.set\_stunden(stunden)

Schreibt die Stunden.

*Aktivitaet*.get\_allstunden()

Gibt alle Stunden aus.

*Aktivitaet*.set\_allstunden(allstunden)

Schreibt alle Stunden.

## Arbeitszeitkonto

Die Klasse Arbeitszeitkonto bildet das Arbeitszeitkonto des Nutzers ab. Sie hält dafür Informationen wie die Gesamtstunden sowie die Urlaubs- und Krankheitstage. Außerdem erbt die Klasse von der abstrakten Klasse BusinessObject.

**Attribute:**

Arbeitszeitkonto.\_gesamtstunden

Enthält die Gesamtstunden des Arbeitzeitkontos.

Arbeitszeitkonto.\_urlaubstage

Enthält die Urlaubstage des Arbeitzeitkontos.

Arbeitszeitkonto.\_krankheitstage

Enthält die Krankheitstage des Arbeitzeitkontos.

**Methoden:**

*Arbeitszeitkonto*.get\_gesamtstunden()

Gibt die Gesamtstunden aus.

*Arbeitszeitkonto*.set\_gesamtstunden(gesamtstunden)

Schreibt die Gesamtstunden.

*Arbeitszeitkonto*.get\_urlaubstage()

Gibt die Urlaubstage aus.

*Arbeitszeitkonto*.set\_urlaubstage(urlaubstage)

Schreibt die Urlaubstage.

*Arbeitszeitkonto*.get\_krankheitstage()

Gibt die Krankheitstage aus.

*Arbeitszeitkonto*.set\_krankheitstage(krankheitstage)

Schreibt die Krankheitstage

## Buchung

Die Klasse Buchung bildet die Buchungen des Nutzers ab. Sie hält dafür Informationen wie das Datum, die Stunden sowie einen Verweis auf die zugehörige Person und Aktivität. Außerdem erbt die Klasse von der abstrakten Klasse BusinessObject.

**Attribute:**

Buchung.\_datum

Enthält das Datum der Buchung.

Buchung.\_stunden

Enthält die Stunden der Buchung.

Buchung.\_person\_id

Enthält die ID der zugehörigen Person.

Buchung.\_aktivitaet\_id

Enthält die ID der zugehörigen Aktivität.

**Methoden:**

*Buchung.*get\_datum()

Gibt das Datum aus.

*Buchung*.set\_datum(datum)

Schreibt das Datum.

*Buchung.*get\_stunden()

Gibt die Stunden aus.

*Buchung*.set\_stunden(stunden)

Schreibt die Stunden.

*Buchung.*get\_person\_id()

Gibt die Personen ID aus.

*Buchung*.set\_person\_id(person\_id)

Schreibt die Personen ID.

*Buchung.*get\_aktivitaeten\_id()

Gibt die Aktivitaeten ID aus.

*Buchung*.set\_aktivitaeten\_id(aktivitaeten\_id)

Schreibt die Aktivitaeten ID.

## Ereignis

Die Klasse Ereignis bildet die Ereignisse ab. Sie hält dafür Informationen wie den Erstellungszeitpunkt. Außerdem erbt die Klasse von der abstrakten Klasse BusinessObject.

**Attribute:**

Ereignis.\_erstellungs\_zeitpunkt

Enthält den Erstellungszeitpunkt des Ereignisses.

**Methoden:**

*Ereignis.*get\_erstellungs\_zeitpunkt()

Gibt den Erstellungszeitpunkt aus.

*Ereignis*.set\_erstellungs\_zeitpunkt(erstellungs\_zeitpunkt)

Schreibt den Erstellungszeitpunkt.

## PersonenProjekt

Die Klasse PersonenProjekt bildet die Verbindung zwischen der Klasse Person und der Klasse Projekt ab. Sie hält dafür Informationen wie die Personen ID und die Projekt ID. Außerdem erbt die Klasse von der abstrakten Klasse BusinessObject.

**Attribute:**

PersonenProjekt.\_person\_id

Enthält die ID der zugehörigen Person.

PersonenProjekt.\_projekt\_id

Enthält die ID des zugehörigen Projekts.

**Methoden:**

PersonenProjekt*.*get\_person\_id()

Gibt die Personen ID aus.

PersonenProjekt.set\_person\_id(person\_id)

Schreibt die Personen ID.

PersonenProjekt*.*get\_projekt\_id()

Gibt die Projekt ID aus.

PersonenProjekt.set\_projekt\_id(projekt\_id)

Schreibt die Projekt ID.

## Zeitintervall

Die Klasse Zeitintervall bildet die Zeitintervalle der Zeitintervallbuchungen ab. Sie hält dafür Informationen wie den Start, das Ende, die Dauer, den Pausenstart, das Pausenende, die Pausendauert sowie einen Verweis auf die zugehörige Person. Außerdem erbt die Klasse von der abstrakten Klasse BusinessObject.

**Attribute:**

Zeitintervall.\_start

Enthält den Start eines Zeitintervalls.

Zeitintervall.\_ende

Enthält das Ende eines Zeitintervalls.

Zeitintervall.\_dauer

Enthält die Dauer eines Zeitintervalls.

Zeitintervall.\_pausen\_start

Enthält den Start einer Pause.

Zeitintervall.\_pausen\_ende

Enthält das Ende einer Pause.

Zeitintervall.\_pausen\_dauer

Enthält die Dauer einer Pause.

Zeitintervall.\_personen\_id

Enthält die ID der zugehörigen Person.

**Methoden:**

Zeitintervall*.*get\_start()

Gibt den Start aus.

Zeitintervall.set\_start(start)

Schreibt den Start.

Zeitintervall*.*get\_ende()

Gibt das Ende aus.

Zeitintervall.set\_ende(ende)

Schreibt das Ende.

Zeitintervall*.*get\_dauer()

Gibt die Dauer aus.

Zeitintervall.set\_dauer(dauer)

Schreibt die Dauer.

Zeitintervall*.*get\_pausen\_start()

Gibt den Pausenstart aus.

Zeitintervall.set\_pausen\_start(pausen\_start)

Schreibt den Pausenstart.

Zeitintervall*.*get\_pausen\_ende()

Gibt das Pausenende aus.

Zeitintervall.set\_pausen\_ende(pause\_nende)

Schreibt das Pausenende.

Zeitintervall*.*get\_pausen\_dauer()

Gibt die Dauer der Pause aus.

Zeitintervall.set\_pausen\_dauer(pausen\_dauer)

Schreibt die Dauer der Pause.

Zeitintervall*.*get\_person\_id()

Gibt die Personen ID aus.

Zeitintervall.set\_person\_id(person\_id)

Schreibt die Personen ID.

## Mapper

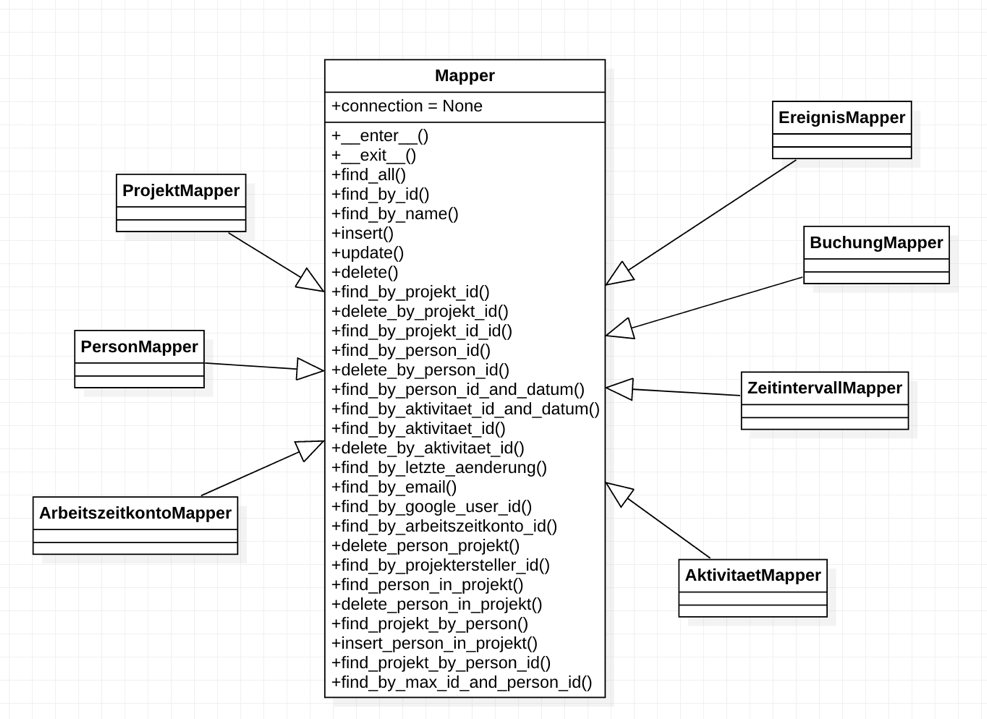


Abbildung 3:Mapperklasse / Eigene Darstellung

Die Datenbankschicht stellt die Verbindung der Applikationslogik zur Datenbank her. Die Mapper-Klassen stellen dabei sicher, dass jeweils nur eine Instanz der Klasse generiert wird. Für jedes Geschäftsobjekt existiert ein Mapper, der den Bezug zur entsprechenden Entität in der Datenbank herstellt. Die Klasse Mapper kapselt die Verbindung zur Datenbank und ein Objekt wird in allen Mappern verwendet.

# 5 ER-Diagramm

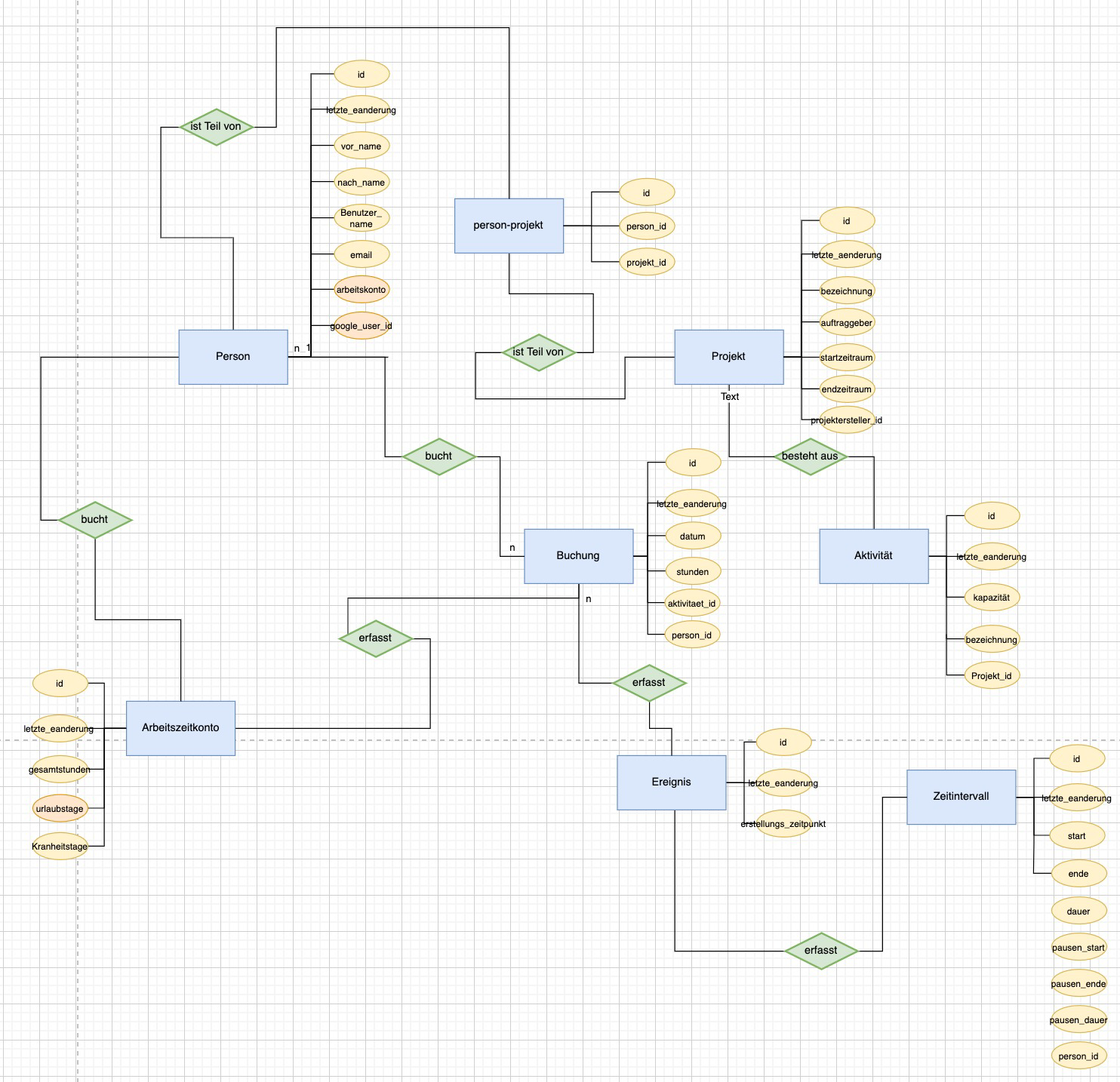


Abbildung 4: ER Diagramm / Eigene Darstellung

Das ER-Diagramm gibt im Wesentlichen die Struktur des Klassendiagramms wieder, da jede Klasse durch eine Tabelle repräsentiert wird.

Jede Tabelle besitzt einen Primärschlüssel, der von der Datenbank automatisch vergeben und hochgezählt wird und einen Timestamp (letzte\_aenderung), der den Erstellungszeitpunkt des Datensatzes wiedergibt. Innerhalb der Datenbank wurden alle n:m-Beziehungen aufgelöst und alle Relationen von Fremdschlüssel-Beziehungen repräsentiert.

Die Erstellung der Datenbank erfolgt über die Datei *dump.sql* im Ordner *database*.

# 6 Installation der fertigen Anwendung

Das GitHub-Repository mit dem Source Code befindet sich unter <https://github.com/Perry-Dettke/sw-praktikum-timetracker>.

## Voraussetzungen

Um die Anwendung lokal ausführen zu können werden folgende Programme beziehungsweise Laufzeitumgebungen benötigt:

* Python
* Node.js
* Terminal (z.B. in VSC oder Pycharm)
* Web-Browser

Darüber hinaus werden einige Python-Module benötigt, die über das Programm pip installiert werden können. Das Anlegen eines Virtual Environments ist empfohlen.

* Flask
* Flask-restx
* Flask-Cors
* Mysql-connector-python

Um die benötigten Node-Packages zu installieren, im Terminal in den Ordner */reactfrontend* navigieren und **npm install** ausführen. Welche Module installiert werden, kann in der Datei */reactfrontend/package.json* eingesehen werden.

## Lokal

Um die App lokal zu installieren, muss zuerst die MySQL-Datenbank lokal angelegt werden. Dies kann über ein Tool wie MySQL Workbench. Für genauere Hinweise ist die die Dokumentation des Herstellers zu konsultieren. Anschließend müssen in der Datei *Mapper.py* im Ordner */backend/server/db/* noch die entsprechenden Login-Informationen angegeben werden.

Als nächster Schritt muss die Authentifizierung mit Google Firebase eingerichtet werden. Dazu in Firebase ein Projekt für eine Web-App anlegen. In diesem Zug wird automatisch eine Konfiguration erstellt, die dann in die Datei */reactfrontend/firebaseconfig.js* kopiert werden muss. Bei Schwierigkeiten kann die Dokumentation seitens Firebase konsultiert werden.

Danach kann der React-Client kompiliert werden. Dazu im Terminal in den Ordner *reactclient* navigieren und den Befehl **npm run build** ausführen. Das fertige Kompilat steht dann im neuen Order *build* zur Verfügung. Der Inhalt dieses Ordners muss nun nach *backend/static/reactclient* kopiert oder verschoben werden.

Anschließend kann der Flask-Server gestartet werden, indem die Datei ‘main.py’ ausgeführt wird. Die Anwendung steht dann im Browser unter <http://localhost:5000/static/reactclient/index.html> zur Verfügung. Die REST-API befindet sich unter [http://localhost:5000](http://localhost:5000/).

## Cloud

Um die App auf Google Cloud bereit zu stellen, muss zuerst über die Google Cloud Console ein neues Projekt angelegt werden. Danach in Google Cloud eine MySQL-Datenbank erstellen und über die Datei *dump.sql* befüllen. In der Datei */backend/server/db/Mapper.py* wieder die entsprechenden Login-Informationen angeben.

Als nächstes muss die Google App Engine aktiviert und auf dem lokalen Rechner die Google Cloud SDK installiert werden. Hier müssen noch die beiden Komponenten **app-engine-python** und **app-engine-python-extras** installiert und das gerade erstellte Projekt ausgewählt werden. Es wird ausdrücklich auf die Dokumentation des Herstellers verwiesen. Anschließend die Firebase-Konfiguration erstellen und den React-Client kompilieren (siehe oben). Dann kann die Anwendung mit der Google Cloud SDK deployed werden. Dazu in den Ordner backend navigieren und den Befehl **gcloud app deploy** ausführen.

Ist der Prozess abgeschlossen, kann die Anwendung mit **gcloud app browse** genutzt werden.

# 7 Nutzerhandbuch

Die App kann über die folgenden Links online aufgerufen werden:

<https://sw-timetracker-355011.ey.r.appspot.com/static/reactclient/index.html>

Öffnet der Nutzer die App, muss er sich immer erst mit seinem Google Account anmelden.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung 5:Login / Eigene Darstellung

Hat der Nutzer noch keinen Account, muss er sich erst registrieren, in dem die Personendaten und seinen Benutzernamen erfasst werden.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung 6:Popup Anmeldung / Eigene Darstellung

Nun wird der Nutzer auf die Home Seite weitergeleitet.

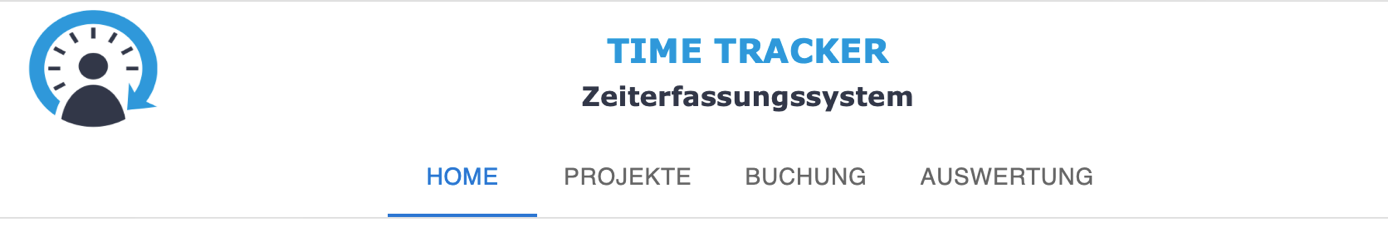


Abbildung 7:Menüleiste / Eigene Darstellung

Auf der Homeseite kann der Nutzer sein Profil sehen, bearbeiten und löschen.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung 8:Mein Profil / Eigene Darstellung

Ebenfalls auf der Homeseite sieht der Nutzer sein Arbeitszeitkonto sowie seine Stempelfunktion.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung 9:Mein Arbeitszeitkonto / Eigene Darstellung

In seinem Arbeitszeitkonto kann der Nutzer seine geleisteten Stunden nach einem Zeitraum filtern. Dazu schreibt er in das Feld Monat ein Datum und nutzt er den Button Zeitraum suchen zum Ausführen.

Der Button Sonderbuchung öffnet ein Popup-Menü, in dem er eine Art von Buchung auswählen kann und die Anzahl der Tage.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung 10:Popup Sonderbuchung / Eigene Darstellung

In seiner Stempelfunktion hat der Nutzer die Möglichkeit, Einzustempeln (über den Button Kommen), Auszustempeln (über den Button Gehen), seine Pause zu beginnen (über den Button Pause) und seine Pause zu beenden (über den Button Pause Beenden).



Abbildung 11:Stempelfunktion / Eigene Darstellung

Über den Reiter Projekte kann der Nutzer seine aktuellen Projekte sehen und ebenfalls neue Projekte, über den Button Neues Projekt anlegen, anlegen.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung 12:Projekteseite / Eigene Darstellung

Der Button Neues Projekt anlegen öffnet ein Popup-Menü, in dem er alle Daten für ein neues Projekt eingeben kann.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung 13:Popup Neues Projekt anlegen / Eigene Darstellung

Nach dem Anlegen eines Projekts hat der Nutzer die Möglichkeit das Projekt, über den Button Projekt Bearbeiten, zu bearbeiten oder das Projekt, über den Button Projekt löschen, zu löschen. Ebenfalls hat der Nutzer die Möglichkeit eine neue Aktivität anzulegen. Drückt der Nutzer den Button Aktivität anlegen, öffnet ein Popup-Menü, in dem er alle Daten für ein neue Aktivität eingeben kann.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung 14:Projekte Übersicht / Eigene Darstellung

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung 15:Popup Neue Aktivität anlegen / Eigene Darstellung

Nach dem Anlegen der Aktivität, kann der Nutzer diese auch bearbeiten oder wieder löschen.

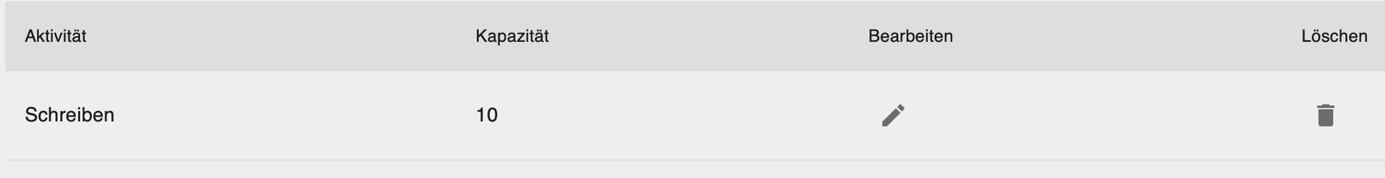


Abbildung 16:Aktivitäten anzeigen / Eigene Darstellung

In dem Reiter Buchung kann der Nutzer neue Buchungen für sein Projekt anlegen. Der Button Projekt-buchung Erstellen öffnet ein Popup-Menü, in dem er erstmal ein Projekt auswählen muss.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung 17:Popup Buchung anlegen / Eigene Darstellung

Danach öffnet sich ein weiteres Popup-Menü, in dem er nun alle Daten der Projektbuchung eintragen muss.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung 18:Popup Neue Projektbuchung / Eigene Darstellung

Nach dem Anlegen der Buchung, kann der Nutzer diese auch bearbeiten oder wieder löschen.

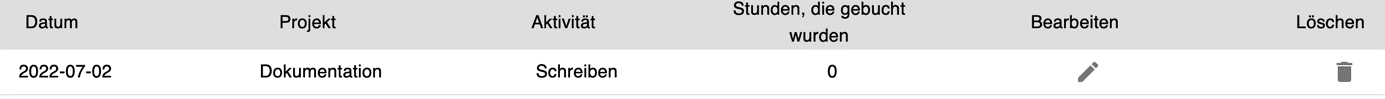


Abbildung 19: Buchung anzeigen/ Eigene Darstellung

In dem Reiter Auswertung kann der Nutzer sich alle Projekte anzeigen lassen, welche er selbst erstellt hat. Hierbei kann der Nutzer in den Projekten die Ist- Stunden und die Restkapazität nach einem Zeitraum filtern. Dazu schreibt er in die Felder Start und Ende jeweils ein Datum und nutzt er den Button Zeitraum suchen zum Ausführen.



Abbildung 20:Zeitraumfiltern Projekt / Eigene Darstellung

Zusätzlich kann der Nutzer sich die Stunden einzelnen Personen, innerhalb eines Zeitraums, einer Aktivität anzeigen lassen. Dazu nutzt er den Button Zeitraum suchen zum Ausführen.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung 21: Zeitraumfiltern Aktivität / Eigene Darstellung

# Abbildungsverzeichnis

[Abbildung 1: Use Case Diagramm / Eigene Darstellung 5](https://buds365.sharepoint.com/teams/Software-Praktikum2022/Freigegebene%20Dokumente/General/Projektbericht/Timetracker_Dokumentation.docx#_Toc107686781)

[Abbildung 2: Klassendiagramm / Eigene Darstellung 8](#_Toc107686782)

[Abbildung 3:Mapperklasse / Eigene Darstellung 16](#_Toc107686783)

[Abbildung 4: ER Diagramm / Eigene Darstellung 17](#_Toc107686784)

[Abbildung 5:Login / Eigene Darstellung 19](#_Toc107686785)

[Abbildung 6:Popup Anmeldung / Eigene Darstellung 20](#_Toc107686786)

[Abbildung 7:Menüleiste / Eigene Darstellung 20](#_Toc107686787)

[Abbildung 8:Mein Profil / Eigene Darstellung 20](#_Toc107686788)

[Abbildung 9:Mein Arbeitszeitkonto / Eigene Darstellung 21](#_Toc107686789)

[Abbildung 10:Popup Sonderbuchung / Eigene Darstellung 21](#_Toc107686790)

[Abbildung 11:Stempelfunktion / Eigene Darstellung 21](#_Toc107686791)

[Abbildung 12:Projekteseite / Eigene Darstellung 22](#_Toc107686792)

[Abbildung 13:Popup Neues Projekt anlegen / Eigene Darstellung 22](#_Toc107686793)

[Abbildung 14:Projekte Übersicht / Eigene Darstellung 23](#_Toc107686794)

[Abbildung 15:Popup Neue Aktivität anlegen / Eigene Darstellung 23](#_Toc107686795)

[Abbildung 16:Aktivitäten anzeigen / Eigene Darstellung 23](#_Toc107686796)

[Abbildung 17:Popup Buchung anlegen / Eigene Darstellung 23](#_Toc107686797)

[Abbildung 18:Popup Neue Projektbuchung / Eigene Darstellung 24](#_Toc107686798)

[Abbildung 19: Buchung anzeigen/ Eigene Darstellung 24](#_Toc107686799)

[Abbildung 20:Zeitraumfiltern Projekt / Eigene Darstellung 24](#_Toc107686800)

[Abbildung 21: Zeitraumfiltern Aktivität / Eigene Darstellung 25](#_Toc107686801)



# Ehrenwörtliche Erklärung

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Vorname: | Perry | Name: Dettke |  | Matrikel-Nr.: 38543 |
| Vorname: | Jana | Name: Drawert |  | Matrikel-Nr.: 39560 |
| Vorname: | Rosalie | Name: Kripp |  | Matrikel-Nr.: 39054 |
| Vorname: | Raphael | Name: Müller |  | Matrikel-Nr.: 38073 |
| Vorname: | Kai | Name: Schmied |  | Matrikel-Nr.: 39122 |
| Vorname: | Jacqueline | Name: Schmid |  | Matrikel-Nr.: 39061 |

Hiermit versichern wir ehrenwörtlich, dass wir die vorliegende Dokumentation zum IT-Projekt selbständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Hilfsmittel benutzt haben. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen wurden, sind in jedem Fall unter Angabe der Quelle kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden.

Wir haben die Bedeutung der ehrenwörtlichen Versicherung und die prüfungsrechtlichen Folgen (§ 26 Abs. 2 der Bachelor-SPO (6-Semester), § 24 Abs. 2 Bachelor-SPO (7-Semester), § 23 Abs. 2 Master-SPO (3 Semester) bzw. §19 Abs. 2 Master-SPO (4 Semester und berufsbegleitend) der HdM) einer unrichtigen oder unvollständigen ehrenwörtlichen Versicherung zur Kenntnis genommen.



Neuhausen, 02.07.22

.................................................. ..........................................................

Ort, Datum Unterschrift

Ein Bild, das Text enthält.

Automatisch generierte BeschreibungStuttgart, 2.07.22

.................................................. ..........................................................

Ort, Datum Unterschrift

Stuttgart, 2.07.22

.................................................. ..........................................................

Ort, Datum Unterschrift

Fellbach, 02.07.22



.................................................. ..........................................................

Ort, Datum Unterschrift

Backnang, 02.07.22



.................................................. ..........................................................

Ort, Datum Unterschrift

Schönaich, 02.07.22



.................................................. ..........................................................

Ort, Datum Unterschrift