Projektbericht: Hochschule Veranstaltungsplaner

Inhaltsverzeichnis

1. Einleitung

1.1 Einleitung

Das Projekt “Hochschule Veranstaltungsplaner” zielt darauf ab, eine Softwarelösung zu entwickeln, die die Planung und Organisation von Lehrveranstaltungen an der Hochschule automatisiert. Das Hauptziel ist es, sicherzustellen, dass jeder Mitarbeiter und Student den aktuellen Unterrichtsplan kennt und weiß, welche Lehrveranstaltung er wann besuchen oder betreuen soll.

1.2 Rahmenbedingungen

- **System**: Es wird eine Applikation erwartet, die Einsatzpläne für Studenten und Mitarbeiter mithilfe eines gegebenen Datensatzes entwickelt und ausgibt. Der Stundenplan beinhaltet: Die Lehrperson, den Raum, die Uhrzeit und das gelehrte Fach. Das System soll (dynamisch) Unterrichtsausfälle wahrnehmen, eine verfügbare Vertretungsperson für diesen Zeitraum auswählen und diese für die Professoren und Studenten anzeigen können. Die Bedingungen für die Verfügbarkeit des Professors sind unter dem Punkt “Professoren” weiter ausgeführt.
- **Veranstaltungen**: Das System beachtet die verschiedenen Studiengänge. Die Dauer der Veranstaltungen soll minimal 2 und maximal 4 Schulstunden lang sein. Veranstaltungen können in zwei verschiedenen Campus stattfinden (A,B). Sonderveranstaltungen sollen im Stundenplan nicht beachtet werden.
- **Mitarbeiter**: Es gibt drei verschiedene Mitarbeiter: Präsident, Professoren, Sekretäre. Mitarbeiter können keine Überstunden machen. Es soll genug Speicherplatz für 100 Mitarbeiter geben. Überstunden sind nicht erlaubt. Es gibt eine Überkapazität an Mitarbeitern.
- **Professoren**: Die maximale Vorlesungszeit beträgt 16 Stunden pro Woche pro Professor. Ein Professor kann zu jeder Zeit höchstens ein Fach unterrichten. Professoren müssen regelmäßig den Vertretungsplan abrufen. Jeder Professor kann jedes Fach unterrichten.
- **Optimal**: Getrennte Stundenplan Ansichten: einmal welche für die Professoren und einmal die Ansichten für die Studenten. Eine Filterfunktion beim Stundenplan z.B. für: Raum, Student, Uhrzeit. Ein eigenes GUI für die Verwendung programmieren (alternativ reicht auch das Terminal).

2. Anforderungen

2.1 Funktionale Anforderungen

- **Veranstaltungsplanung**: Die Software muss in der Lage sein, einen Stundenplan zu erstellen.
- **Vertretungsplan:** Die Software muss in der Lage sein einen Vertretungsplan zu erstellen, indem verfügbare Dozenten in den zu vertretenden Veranstaltungen eingesetzt werden. Der Vertretungsplan enthält alle aktuellen Änderungen und Abweichungen vom regulären Stundenplan. Dabei ist folgendes zu beachten:
  - 16 Stunden Wochenarbeitszeit
  - Kein zeitgleiches Auftreten eines Dozenten
- **Verwaltung von Mitarbeitern**: Das System muss in der Lage sein das Hinzufügen oder Löschen von Mitarbeitern zu ermöglichen
- **Ausfallmeldefunktion**: Dozenten müssen die Möglichkeit haben sich im System krank zu melden, sodass eine Vertretung für die Veranstaltung organisiert werden kann

2.2 Nichtfunktionale Anforderungen

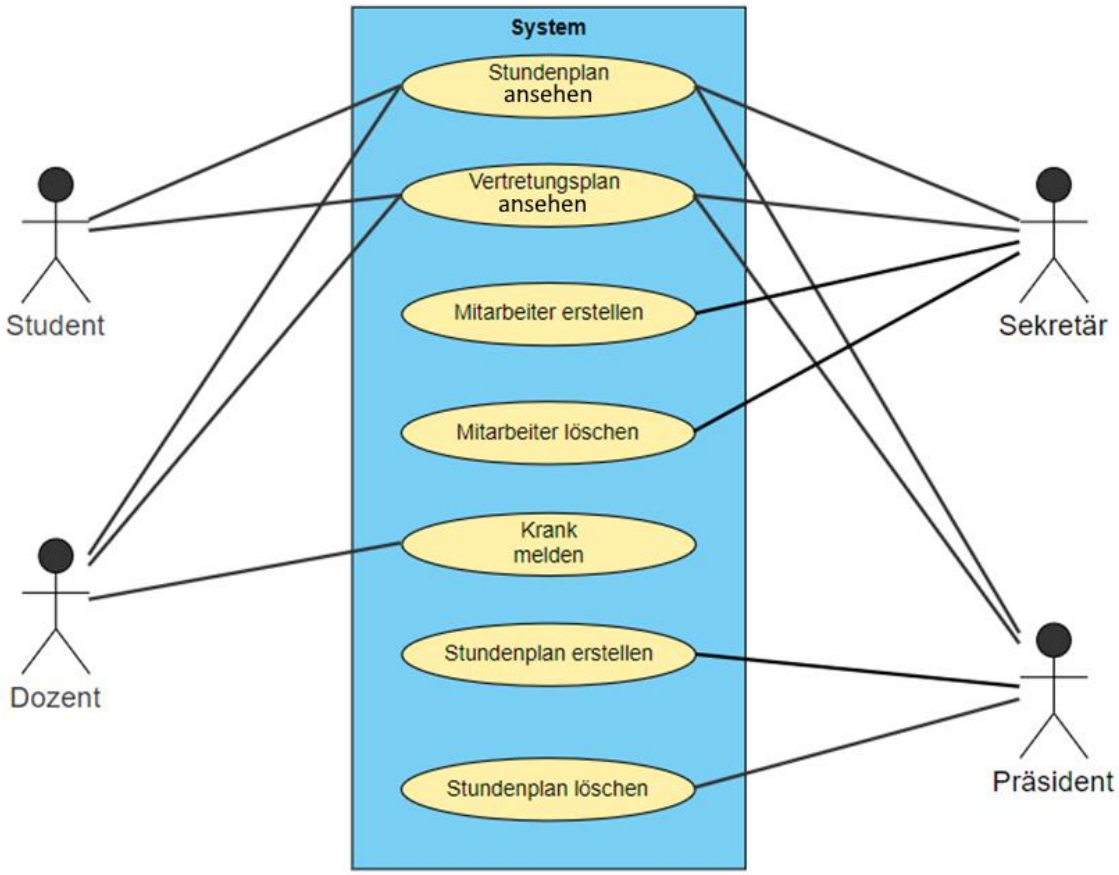
- **Benutzerfreundlichkeit**: Das System soll einfach zu benutzen sein, sodass keine speziellen Kenntnisse zur Nutzung erforderlich sind.
- **Zuverlässigkeit**: Die Software sollte zuverlässig sein und korrekte und aktuelle Informationen zu den Veranstaltungen liefern.

3. Use Case (Anwendungsfälle)

3.1 Akteure

- **Dozenten**: Ein Dozent kann seinen eigenen Einsatzplan einsehen und seinen Ausfall an das System melden.
- **Sekretariat**: Ein Sekretär kann den Stundenplan / Vertretungsplan einsehen, sowie Mitarbeiter erstellen und löschen.
- **Studenten**: Ein Student kann den Stundenplan und den Vertretungsplan einsehen.
- **System**: Das System generiert den Stundenplan und den Vertretungsplan und gibt diese aus und sorgt dafür, verfügbare Vertretungskräfte zu finden.
- **Präsident**: Der Präsident initialisiert das System mit den notwendigen Daten, indem er lediglich die Erstellung des Stundenplans einleitet. Das Löschen des Stundenplans ist auch möglich.

3.2 Use Case Diagramm



3.3 Textuelle Erläuterung zu Use Cases

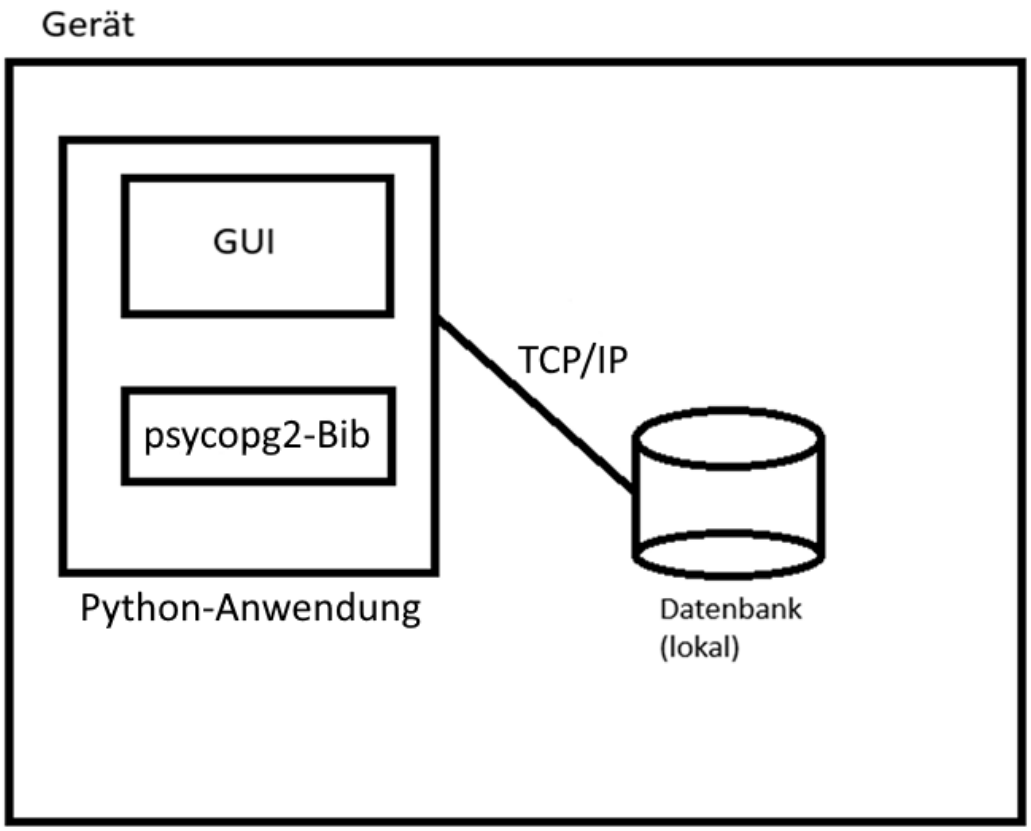
- **Stundenplan ansehen:**
  Voraussetzung: Nutzer ist angemeldet + System befüllt mit Daten
  Hauptszenario:
  1) Der Nutzer startet das Programm.
  2) Er gibt seine ID ein
  3) Die Option "Stundenplan einsehen" wird gewählt
  4) System gibt Stundenplan in der Konsole aus
  Ausnahmekriterium: Präsident hat die Erstellung des Stundenplans noch nicht eingeleitet
- **Vertretungsplan ansehen:**
  Voraussetzung: Nutzer ist angemeldet + System befüllt mit Daten
  Hauptszenario:
  1) Der Nutzer startet das Programm.
  2) Er gibt seine ID ein
  3) Die Option "Vertretungsplan einsehen" wird gewählt
  4) System gibt Vertretungsplan in der Konsole aus
  Ausnahmekriterium: Es gibt keine Vertretungen

- **Mitarbeiter erstellen:**
  Voraussetzung: Sekretär ist angemeldet
  Hauptszenario:
  1) Der Sekretär startet das Programm.
  2) Er gibt seine ID ein
  3) Die Option "Mitarbeiter verwalten" wird gewählt
  4) Die Option „Mitarbeiter hinzufügen“ wird gewählt
  5) Eingabe: Rolle des neuen Mitarbeiters
  6) Eingabe: Vorname
  7) Eingabe: Nachname
  8) Eingabe: Nummer
  9) System fügt den neuen Mitarbeiter in die Datenbank ein
  Ausnahmekriterium: /
- **Mitarbeiter Löschen:**
  Voraussetzung: Sekretär ist angemeldet
  Hauptszenario:
  1) Der Sekretär startet das Programm.
  2) Er gibt seine ID ein
  3) Die Option "Mitarbeiter verwalten" wird gewählt
  4) Die Option „Mitarbeiter löschen“ wird gewählt
  5) Eingabe: ID des zu löschenden Mitarbeiters
  6) System löscht den Mitarbeiter der entsprechenden ID aus der Datenbank
  Ausnahmekriterium: Es existiert kein Mitarbeiter mit der eingegebenen ID
- **Krankmelden:**
  Voraussetzung: Dozent ist angemeldet
  Hauptszenario:
  1) Der Dozent startet das Programm.
  2) Er gibt seine ID ein
  3) Die Option "Krankmelden" wird gewählt
  4) Wahl des Wochentags
  5) System meldet den Dozenten krank
  Ausnahmekriterium: Dozent hat keine Veranstaltungsstunden gehabt
- **Stundenplan erstellen:**
  Voraussetzung: Präsident ist angemeldet
  Hauptszenario:
  1) Präsident startet das Programm.
  2) Er gibt seine ID ein
  3) Die Option "System initialisieren" wird gewählt
  4) Die Option „Stundenplan erstellen“ wird gewählt
  5) System leitet die Generierung des Stundenplans ein
  Ausnahmekriterium: Es existiert bereits ein Stundenplan

- **Stundenplan löschen:**
  Voraussetzung: Präsident ist angemeldet
  Hauptszenario:
  1) Präsident startet das Programm.
  2) Er gibt seine ID ein
  3) Die Option "System initialisieren" wird gewählt
  4) Die Option „Stundenplan löschen“ wird gewählt
  5) System leitet die Löschung des Stundenplans ein
  Ausnahmekriterium: Es existiert noch kein Stundenplan

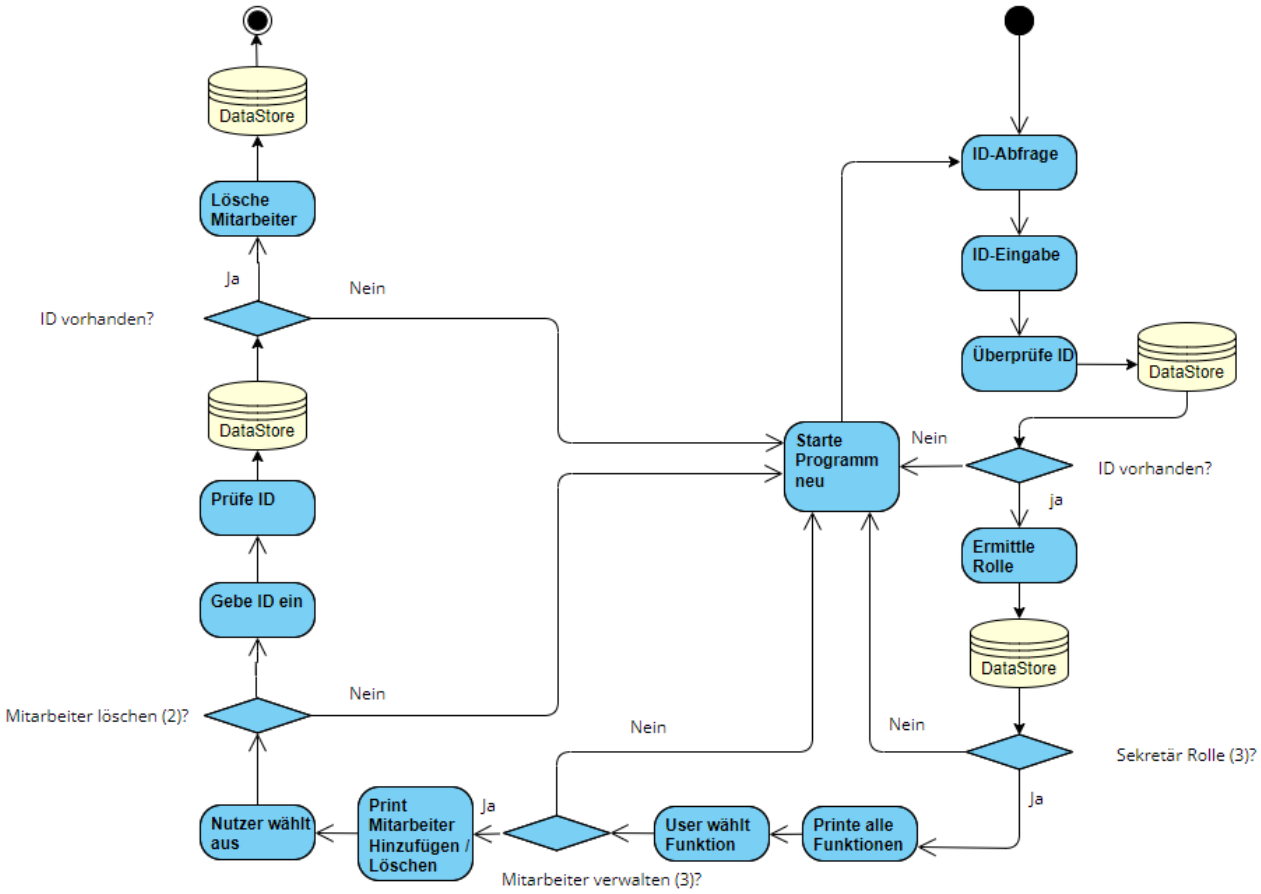
4. Architektur

Für dieses Projekt wird ein Python-Programm entwickelt, das mit einer PostgreSQL-Datenbank interagiert. Die Verwendung von Python ermöglicht eine souveräne Ausführung, während PostgreSQL eine robuste und zuverlässige Lösung für die Datenhaltung bietet. - **Architektur-Muster**: Client-Server - Client: Python-Programm stellt Anfragen an Server - Server: PostgreSQL, bearbeitet und stellt Daten bereit



5. Geschäftsprozesse

5.1 Geschäftsprozess „Mitarbeiter Löschen“



Zu Beginn wird die ID abgefragt. Das System überprüft dann die eingegebene ID und ermittelt, bei vorhandener Gültigkeit, die Rolle, zu der die ID gehört. Falls diese ungültig ist, wird das Programm neugestartet. Wenn nun die entsprechende Rolle der Rolle des Sekretärs gleicht, so werden alle verfügbaren Funktionen ausgegeben:

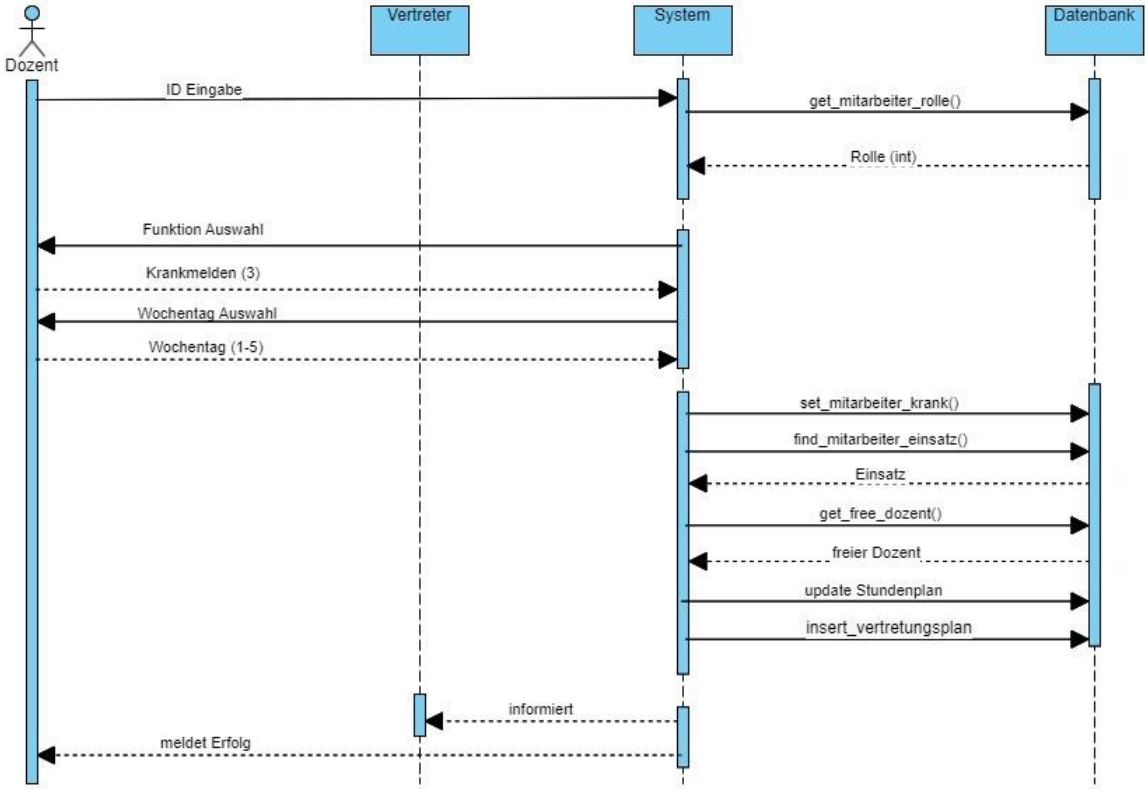
1. *Stundenplan ansehen*
2. *Vertretungsplan ansehen*
3. ***Mitarbeiter verwalten***

Nun überprüft das System, ob die Eingabe <3> erfolgt, was der Auswahl *„Mitarbeiter verwalten“* entspricht. Ist dies der Fall, so werden folgende Funktionen ausgegeben:

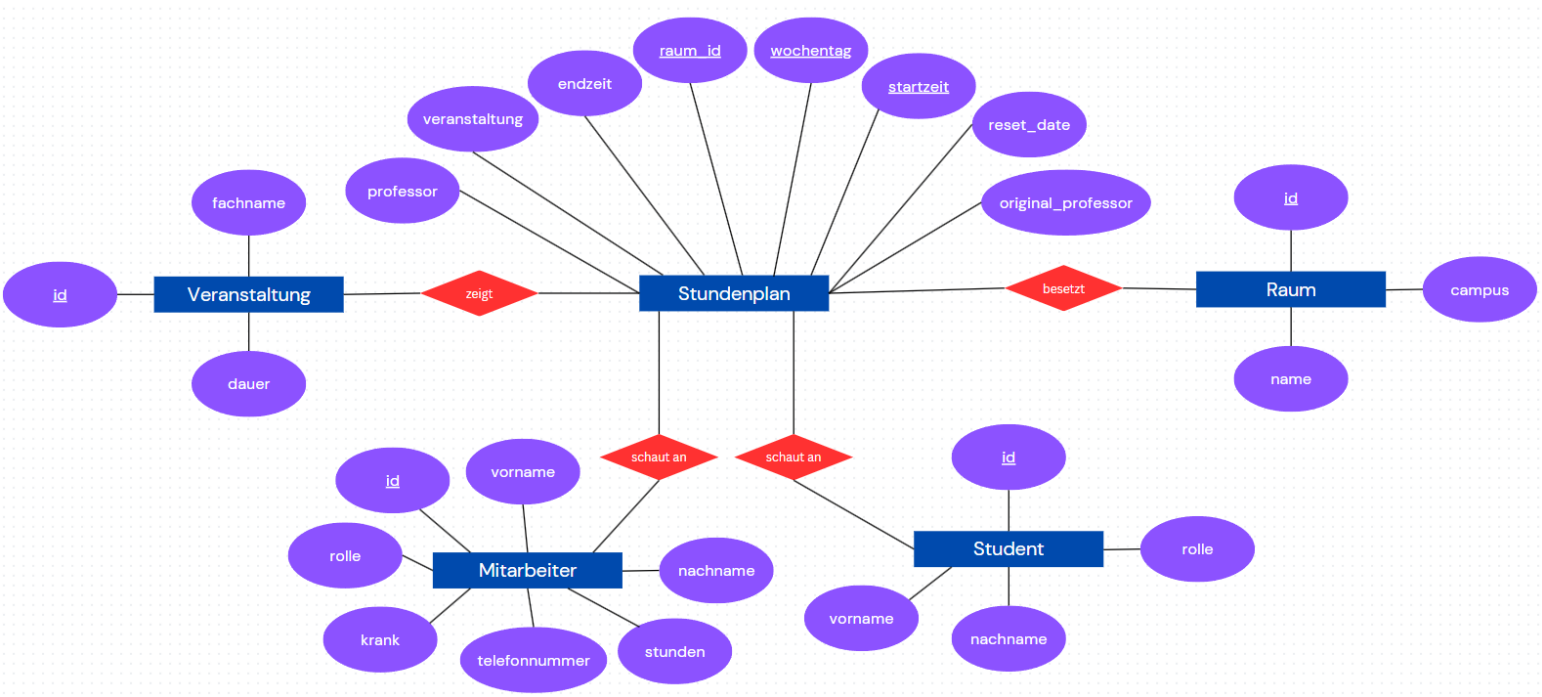
1. *Mitarbeiter hinzufügen*
2. ***Mitarbeiter löschen***

Durch die Eingabe <2> wird die ID des zu löschenden Mitarbeiters abgefragt, sodass eine Eingabe mit anschließender ID-Prüfung möglich ist. Ist die ID vorhanden, wird der zugehörige Mitarbeiter gelöscht und der Prozess ist abgeschlossen. Bei einem Fehlschlag, wird das Programm neugestartet.

5.2 Geschäftsprozess „Krankmeldung einer Lehrperson“



6. Datenmodell



7. GUI

Die Kommunikation zwischen Nutzer und System ist ein Dialog, bei dem das System zunächst eine Aufforderung stellt oder ein Auswahlmenü ausgibt.

Aufforderung
Das System fordert den Nutzer auf eine bestimmte, keine beliebige, Eingabe zu tätigen (zB. ID)

Menüpunkt
Das System stellt einen Katalog von Funktionen bereit. Zu jeder Funktion gibt es eine Nummer, die diese Funktion aufruft.

Stundenplan
Es folgen Darstellungen des GUIs in verschiedenen Anwendungsfällen bei Benutzung des Systems:

```
Geben Sie Ihre ID ein: 15

Willkommen Herr Zimmermann!
(1) Stundenplan ansehen
(2) Vertretungsplan ansehen
(3) Stundenplan initialisieren
Ihre Wahl: 3

(1) Stundenplan erstellen
(2) Stundenplan löschen
Ihre Wahl: 1

Stundenplan erfolgreich erstellt!
```

Präsident - Stundenplan-Initialisierung

Nach Eingabe der entsprechenden ID erfolgt die Anmeldung des Präsidenten und die Erstellung des Stundenplans kann eingeleitet werden.

Prozedur: siehe 3.3 - Stundenplan erstellen

```
Geben Sie Ihre ID ein: 2

Willkommen Prof. Schmidt!
(1) Stundenplan ansehen
(2) Vertretungsplan ansehen
(3) Krank melden
Ihre Wahl: 3

Wochentag der Krankmeldung:
(1) Montag
(2) Dienstag
(3) Mittwoch
(4) Donnerstag
(5) Freitag
Ihre Wahl: 1

Sie haben sich erfolgreich krankgemeldet.
```

Dozent - Krankmeldung

Nach Eingabe der entsprechenden ID erfolgt die Anmeldung des Dozenten und die Krankmeldung kann nach Eingabe des Wochentags erfolgen.

Prozedur: siehe 3.3 - Krankmelden

```
Geben Sie Ihre ID ein: 18

Willkommen Julia!
(1) Stundenplan ansehen
(2) Vertretungsplan ansehen
Ihre Wahl: 1

Stundenplan:
Wochentag  Zeit        Veranstaltung             Professor          Raum
=============================================================================================
Montag     8-10        Mathe I                   Müller             Raum A1
Montag     8-12        Webentwicklung HTML/CSS   Weber              Raum A2
Montag     8-12        Gegenwartssoziologie      Wolf               Raum A3
Montag     8-12        Nachhaltiges Management   Wagner             Raum A4
Montag     12-14       Statistik                 Wolf               Raum A1
Montag     12-16       Marketing                 Richter            Raum A2
Montag     14-16       Marketing-Strategien      Schäfer            Raum A1
Dienstag   8-10        Philosophie Aufklärung    Schmidt            Raum A2
Dienstag   8-10        Literaturgeschichte       Weber              Raum A1
Dienstag   10-12       Medizinische Ethik        Becker             Raum A2
Dienstag   10-14       Informatik Grundlagen     Schmidt            Raum A1
Dienstag   14-16       Physikalische Chemie      Klein              Raum A1
Mittwoch   8-12        Internationales Recht     Koch               Raum A2
Mittwoch   8-12        Lineare Algebra           Müller             Raum A3
Mittwoch   8-10        Architektur Geschichte    Becker             Raum A4
```

Student - Stundenplan-Ansicht

Nach Eingabe der entsprechenden ID, kann die Auswahl zur Anzeige des Stundenplans getroffen werden.

Prozedur: siehe 3.3 - Stundenplan ansehen

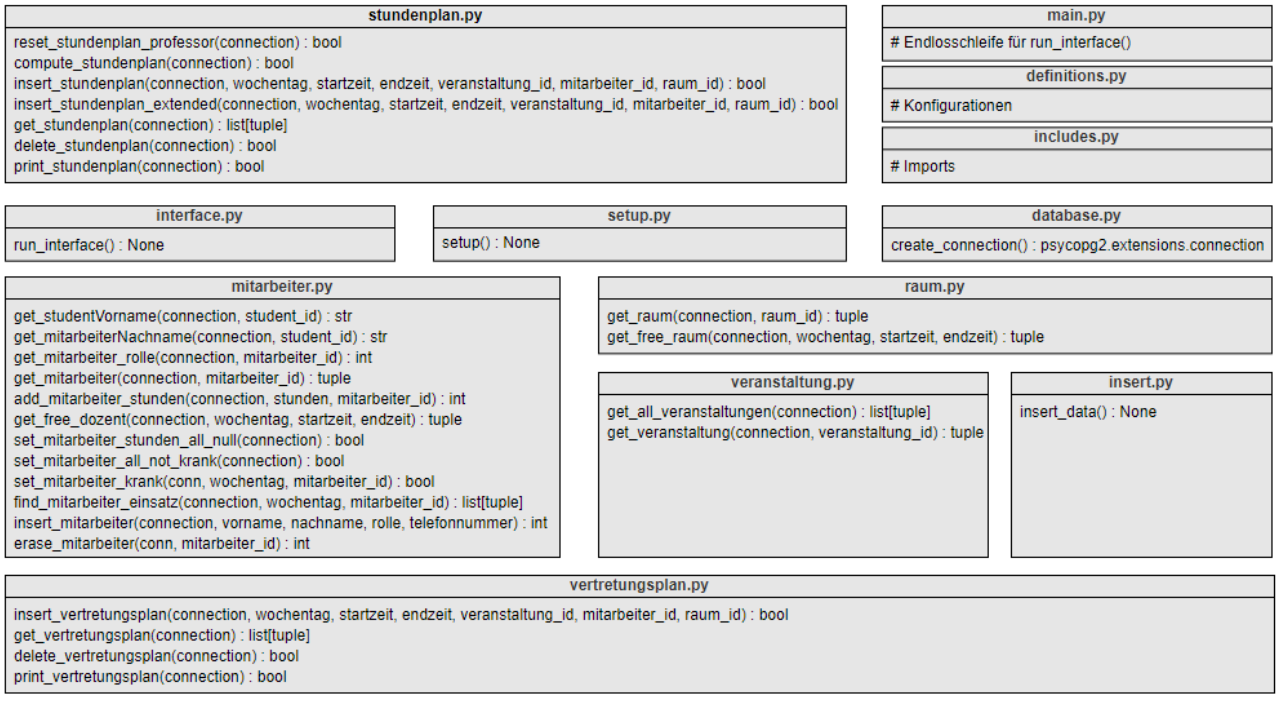
```
Geben Sie Ihre ID ein: 14

Willkommen Frau Schwarz!
(1) Stundenplan ansehen
(2) Vertretungsplan ansehen
(3) Mitarbeiter verwalten
Ihre Wahl: 3

(1) Mitarbeiter hinzufügen
(2) Mitarbeiter löschen
Ihre Wahl: 1

Geben Sie die Rolle des neuen Mitarbeiters ein (2 = Dozent, 3 = Sekretär, 4 = Praesident): 2

Geben Sie den Vornamen des neuen Mitarbeiters ein: Arthur

Geben Sie den Nachnamen des neuen Mitarbeiters ein: Schopenhauer

Geben Sie die Nummer des neuen Mitarbeiters ein: 420
Erfolgreich hinzugefügt.
```

Sekretär - Mitarbeiter hinzufügen

Nach Eingabe der entsprechenden ID, kann ein Mitarbeiter durch Eingabe seiner Attributwerte erstellt werden.

Prozedur: siehe 3.3 - Mitarbeiter erstellen

8. Moduldiagramm



9. Implementierung

- **Veranstaltungsplanung - Erstellung des Stundenplans**
  - Funktion 1) `compute_stundenplan(connection)`
    Umsetzung: Diese Funktion berechnet und initialisiert den Stundenplan basierend auf den aktuellen Veranstaltungen und den Verfügbarkeiten der Dozenten und Räume.
  - Funktion 2) `print_stundenplan(connection)`
    Umsetzung: Diese Funktion gibt den aktuellen Stundenplan aus und stellt somit sicher, dass die Informationen für die Benutzer sichtbar und überprüfbar sind.
- **Weitere (sekundäre) Funktionen für die Veranstaltungsplanung**
  - Funktion 3) `insert_stundenplan(connection, wochentag, startzeit, endzeit, veranstaltung_id, mitarbeiter_id, raum_id)`
    Umsetzung: Diese Funktion fügt neue Einträge zum Stundenplan hinzu, was zur Verwaltung und Organisation des Stundenplans beiträgt.
  - Funktion 4) `insert_stundenplan_extended(connection, wochentag, startzeit, endzeit, veranstaltung_id, mitarbeiter_id, raum_id)`
    Umsetzung: Diese Funktion fügt initiale Einträge in den Stundenplan ein. Die Einträge enthalten auch den ursprünglichen Professor, um spätere Resets bei Vertretungen zu ermöglichen.
  - Funktion 5) `get_stundenplan(connection)`
    Umsetzung: Die Funktion gibt alle Stundenplan Einträge Sortiert nach dem Wochentag und der Startzeit zurück.
  - Funktion 6) `delete_stundenplan(connection)`
    Umsetzung: Löscht Einträge aus dem Stundenplan. Dient der Verwaltung des Stundenplans
  - Funktion 7) `reset_stundenplan_professor(connection)`
    Umsetzung: Diese Funktion setzt den Vertreter zu auf den originalen Professor zurück, aktualisiert die Arbeitsstunden und den Gesundheitsstatus des Professors und löscht den Eintrag im Vertretungsplan.
- **Vertretungsplan - Erstellung und Verwaltung**
  - Funktion 1) `print_vertretungsplan(connection)`
    Umsetzung: Diese Funktion gibt den aktuellen Vertretungsplan aus und stellt sicher, dass die Informationen für die Benutzer sichtbar und überprüfbar sind.
  - Funktion 2) `insert_vertretungsplan(connection, wochentag, startzeit, endzeit, veranstaltung_id, mitarbeiter_id, raum_id)`
    Umsetzung: Diese Funktion fügt Einträge in den Vertretungsplan ein.

- **Weitere (sekundäre) Funktionen für den Vertretungsplan**
  - Funktion 3) `get_vertretungsplan(connection)`
    Umsetzung: Diese Funktion gibt alle aktuellen Einträge im Vertretungsplan sortiert nach Wochentag und Startzeit zurück.
  - Funktion 4) `delete_vertretungsplan(connection)`
    Umsetzung: Diese Funktion löscht alle Einträge im Vertretungsplan, was zur Verwaltung und Aktualisierung des Plans beiträgt.
- **Verwaltung von Mitarbeitern**
  - Funktion 1) `insert_mitarbeiter(connection, vorname, nachname, rolle, telefonnummer)`
    Umsetzung: Fügt in die Datenbank einen neuen Mitarbeiter mit einem Vornamen, Nachnamen, einer Rolle und der dazugehörigen Telefonnummer ein.
  - Funktion 2) `erase_mitarbeiter(conn, mitarbeiter_id)`
    Umsetzung: Identifiziert einen Mitarbeiter mithilfe seiner ID und löscht ihn anschließend aus der Datenbank.
    Abweichung: Es gibt eine Vertretungslogik, die für jede unterrichtende Stunde des gelöschten Professors einen neuen Professor findet.
- **Ausfall-Meldefunktion mit Vertretung**
  - Funktion 1) `set_mitarbeiter_krank(conn, wochentag, mitarbeiter_id)`
    Umsetzung: Ändert den Gesundheitsstatus des kranken Mitarbeiters. Danach wird für jede Stunde, die der Dozent unterrichtet hat eine verfügbare Vertretung gefunden.
    Abweichung: Es wurde festgelegt, dass der Professor sich pro Krankmeldung nur für einen Tag krankmelden kann. Dazu werden Einträge in dem Vertretungsplan für die zu vertretenden Stunden erstellt. Dazu wird ein Reset-Datum von 7 Tagen festgelegt, das als Zeitstempel fungiert, um den ursprünglichen Professor in den Stundenplan wieder einzusetzen und den Vertretungsauftrag zu löschen.
- **Weitere (sekundäre) Funktionen für die Krankmeldung**
  - Funktion 2) `find_mitarbeiter_einsatz(connection, wochentag, mitarbeiter_id)`
    Umsetzung: Diese Funktion findet alle Stunden, an denen ein Mitarbeiter an einem bestimmten Wochentag gearbeitet hat. Sie wird für die Krankmeldung benutzt.
  - Funktion 3) `get_free_dozent(connection, wochentag, startzeit, endzeit)`
    Umsetzung: Diese Funktion liefert einen freien Dozenten für einen bestimmten Wochentag und sowohl Start-, als auch Endzeit. Es wird dazu eine Datenabfrage mit Filterkriterien in der Datenbank getätigt.

10. Test

- **Datei - test_database.py**
  - Funktion 1) `test_create_connection_success`
    Umsetzung: Überprüft, ob die Funktion create_connection erfolgreich eine Datenbankverbindung erstellt und stellt sicher, dass die Verbindung nicht None ist. Falls die Verbindung erfolgreich erstellt wurde, wird versucht, die Verbindung zu schließen, wobei ein Fehler beim Schließen der Verbindung einen Testfehler auslöst.
- **Datei - test_interface.py**
  - Funktion 1) `test_run_interface`
    Umsetzung: Überprüft die Funktion run_interface, indem verschiedene Abhängigkeiten wie Datenbankverbindung, Abfragen der Mitarbeiterrolle und des Studentenvornamens sowie die Benutzereingaben simuliert werden, und stellt sicher, dass die Funktion ohne Fehler ausgeführt wird und None zurückgibt.
- **Datei - test_mitarbeiter.py**
  - Funktion 1) `test_get_mitarbeiter_rolle_success`
    Umsetzung: Überprüft, ob die Funktion get_mitarbeiter_rolle erfolgreich die Rolle eines Mitarbeiters aus der Datenbank abruft und die erwartete Rolle zurückgibt, wenn eine gültige Mitarbeiter-ID übergeben wird.
  - Funktion 2) `test_get_mitarbeiter_rolle_failure`
    Umsetzung: Testet, ob die Funktion get_mitarbeiter_rolle None zurückgibt, wenn eine ungültige Mitarbeiter-ID übergeben wird und die Datenbankabfrage fehlschlägt.
  - Funktion 3) `test_get_mitarbeiter_success`
    Umsetzung: Überprüft, ob die Funktion get_mitarbeiter erfolgreich die Details eines Mitarbeiters aus der Datenbank abruft und die erwarteten Werte zurückgibt, wenn eine gültige Mitarbeiter-ID übergeben wird.
  - Funktion 4) `test_get_mitarbeiter_failure`
    Umsetzung: Testet, ob die Funktion get_mitarbeiter None zurückgibt, wenn eine ungültige Mitarbeiter-ID übergeben wird und die Datenbankabfrage fehlschlägt.
  - Funktion 5) `test_add_mitarbeiter_stunden_success`
    Umsetzung: Überprüft, ob die Funktion add_mitarbeiter_stunden erfolgreich die Stunden eines Mitarbeiters erhöht und den neuen Stundenwert zurückgibt, wenn eine gültige Mitarbeiter-ID übergeben wird.
  - Funktion 6) `test_add_mitarbeiter_stunden_failure`
    Umsetzung: Testet, ob die Funktion add_mitarbeiter_stunden None zurückgibt und ein Rollback durchführt, wenn eine ungültige Mitarbeiter-ID übergeben wird.

- Funktion 7) `test_free_dozent_success`
  Umsetzung: Überprüft, ob die Funktion get_free_dozent erfolgreich einen freien Dozenten aus der Datenbank abruft und die erwarteten Werte zurückgibt, wenn gültige Parameter übergeben werden.
- Funktion 8) `test_free_dozent_failure`
  Umsetzung: Testet, ob die Funktion get_free_dozent None zurückgibt und ein Rollback durchführt, wenn kein freier Dozent gefunden wird.
- Funktion 9) `test_set_mitarbeiter_stunden_all_null_success`
  Umsetzung: Überprüft, ob die Funktion set_mitarbeiter_stunden_all_null erfolgreich die Stunden aller Mitarbeiter auf null setzt und True zurückgibt.
- Funktion 10) `test_set_mitarbeiter_all_not_krank_success`
- Umsetzung: Überprüft, ob die Funktion set_mitarbeiter_all_not_krank erfolgreich den Krankheitsstatus aller Mitarbeiter auf "nicht krank" setzt und True zurückgibt.
- Funktion 11) `test_set_mitarbeiter_krank_success`
  Umsetzung: Überprüft, ob die Funktion set_mitarbeiter_krank erfolgreich den Krankheitsstatus eines Mitarbeiters setzt und True zurückgibt, wenn gültige Parameter übergeben werden.
- Funktion 12) `test_set_mitarbeiter_krank_failure`
  Umsetzung: Testet, ob die Funktion set_mitarbeiter_krank None zurückgibt und ein Rollback durchführt, wenn ungültige Parameter übergeben werden.
- Funktion 13) `test_find_mitarbeiter_einsatz_success`
  Umsetzung: Überprüft, ob die Funktion find_mitarbeiter_einsatz erfolgreich die Einsätze eines Mitarbeiters abruft und die erwarteten Werte zurückgibt, wenn gültige Parameter übergeben werden.
- Funktion 14) `test_find_mitarbeiter_einsatz_failure`
  Umsetzung: Testet, ob die Funktion find_mitarbeiter_einsatz None zurückgibt und ein Rollback durchführt, wenn keine Einsätze für die übergebenen Parameter gefunden werden.
- Funktion 15) `test_insert_mitarbeiter_success`
  Umsetzung: Überprüft, ob die Funktion insert_mitarbeiter erfolgreich einen neuen Mitarbeiter in die Datenbank einfügt und die neue Mitarbeiter-ID zurückgibt.
- Funktion 16) `test_insert_mitarbeiter_failure`
  Umsetzung: Testet, ob die Funktion insert_mitarbeiter None zurückgibt und ein Rollback durchführt, wenn das Einfügen eines neuen Mitarbeiters fehlschlägt.
- Funktion 17) `test_erase_mitarbeiter_success`
  Umsetzung: Überprüft, ob die Funktion erase_mitarbeiter erfolgreich einen Mitarbeiter aus der Datenbank löscht und True zurückgibt, wenn eine gültige Mitarbeiter-ID übergeben wird.

- Funktion 18) `test_erase_mitarbeiter_failure`
  Umsetzung: Testet, ob die Funktion erase_mitarbeiter None zurückgibt und ein Rollback durchführt, wenn eine ungültige Mitarbeiter-ID übergeben wird.

- **Datei - test_raum.py**
  - Funktion 1) `test_get_raum_success`
    Umsetzung: Überprüft, ob die Funktion get_raum erfolgreich einen Raum aus der Datenbank abruft und die erwarteten Werte zurückgibt, wenn eine gültige Raum-ID übergeben wird.
  - Funktion 2) `test_get_raum_failure`
    Umsetzung: Testet, ob die Funktion get_raum None zurückgibt, wenn eine ungültige Raum-ID übergeben wird und die Datenbankabfrage fehlschlägt.
  - Funktion 3) `test_get_free_raum_success`
    Umsetzung: Überprüft, ob die Funktion get_free_raum erfolgreich einen freien Raum aus der Datenbank abruft und die erwarteten Werte zurückgibt, wenn die übergebenen Zeiten und Wochentag gültig sind.
  - Funktion 4) `test_get_free_raum_failure`
    Umsetzung: Testet, ob die Funktion get_free_raum None zurückgibt, wenn kein freier Raum gefunden wird, der den übergebenen Zeiten und dem Wochentag entspricht.

- **Datei - test_stundenplan.py**
  - Funktion 1) `test_reset_stundenplan_professor_success`
    Umsetzung: Überprüft, ob die Funktion reset_stundenplan_professor erfolgreich ausgeführt wird und einen positiven Rückgabewert liefert, wenn die Verbindung zur Datenbank erfolgreich hergestellt wird.
  - Funktion 2) `test_reset_stundenplan_professor_error`
    Umsetzung: Simuliert einen Datenbankverbindungsfehler und stellt sicher, dass die Funktion reset_stundenplan_professor None zurückgibt, wenn die Verbindung fehlschlägt.
  - Funktion 3) `test_compute_stundenplan_success`
    Umsetzung: Testet, ob die Funktion compute_stundenplan korrekt arbeitet und einen positiven Rückgabewert liefert, wenn die Datenbankabfragen erfolgreich sind.
  - Funktion 4) `test_compute_stundenplan_error`
    Umsetzung: Überprüft, ob die Funktion compute_stundenplan None zurückgibt, wenn ein Fehler bei der Datenbankverbindung auftritt.
  - Funktion 5) `test_insert_stundenplan_success`
    Umsetzung: Überprüft, ob die Funktion insert_stundenplan erfolgreich ist und einen positiven Rückgabewert liefert, wenn gültige Eingabewerte übergeben werden.

- Funktion 6) `test_insert_stundenplan_invalid_wochentag`
  Umsetzung: Testet, ob die Funktion insert_stundenplan None zurückgibt, wenn ein ungültiger Wochentag (6) übergeben wird.
- Funktion 7) `test_insert_stundenplan_invalid_startzeit`
  Umsetzung: Überprüft, ob die Funktion insert_stundenplan None zurückgibt, wenn eine ungültige Startzeit (7 Uhr) übergeben wird.
- Funktion 8) `test_insert_stundenplan_invalid_endzeit`
  Umsetzung: Stellt sicher, dass die Funktion insert_stundenplan None zurückgibt, wenn eine ungültige Endzeit (17 Uhr) übergeben wird.
- Funktion 9) `test_insert_stundenplan_invalid_veranstaltung_id`
  Umsetzung: Testet, ob die Funktion insert_stundenplan None zurückgibt, wenn eine ungültige Veranstaltungs-ID (0) übergeben wird.
- Funktion 10) `test_insert_stundenplan_invalid_mitarbeiter_id`
  Umsetzung: Überprüft, ob die Funktion insert_stundenplan None zurückgibt, wenn eine ungültige Mitarbeiter-ID (0) übergeben wird.
- Funktion 11) `test_insert_stundenplan_exception`
  Umsetzung: Simuliert einen Datenbankverbindungsfehler und stellt sicher, dass die Funktion insert_stundenplan None zurückgibt und ein Rollback durchgeführt wird.
- Funktion 12) `test_insert_stundenplan_extended_success`
  Umsetzung: Überprüft, ob die Funktion insert_stundenplan_extended erfolgreich ist und einen positiven Rückgabewert liefert, wenn gültige Eingabewerte übergeben werden.
- Funktion 13) `test_insert_stundenplan_extended_invalid_wochentag`
  Umsetzung: Testet, ob die Funktion insert_stundenplan_extended None zurückgibt, wenn ein ungültiger Wochentag (6) übergeben wird.
- Funktion 14) `test_insert_stundenplan_extended_invalid_startzeit`
  Umsetzung: berprüft, ob die Funktion insert_stundenplan_extended None zurückgibt, wenn eine ungültige Startzeit (7 Uhr) übergeben wird.
- Funktion 15) `test_insert_stundenplan_extended_invalid_endzeit`
  Umsetzung: Stellt sicher, dass die Funktion insert_stundenplan_extended None zurückgibt, wenn eine ungültige Endzeit (17 Uhr) übergeben wird.
- Funktion 16) `test_insert_stundenplan_extended_invalid_veranstaltung_id`
  Umsetzung: Testet, ob die Funktion insert_stundenplan_extended None zurückgibt, wenn eine ungültige Veranstaltungs-ID (0) übergeben wird.
- Funktion 17) `test_insert_stundenplan_extended_invalid_mitarbeiter_id`
  Umsetzung: Überprüft, ob die Funktion insert_stundenplan_extended None zurückgibt, wenn eine ungültige Mitarbeiter-ID (0) übergeben wird.
- Funktion 18) `test_insert_stundenplan_extended_exception`
  Umsetzung: Simuliert einen Datenbankverbindungsfehler und stellt sicher, dass die Funktion insert_stundenplan_extended None zurückgibt und ein Rollback durchgeführt wird.

- Funktion 19) `test_get_stundenplan_success`
  Umsetzung: Überprüft, ob die Funktion get_stundenplan erfolgreich ist und die erwarteten Daten zurückgibt, wenn die Datenbankabfrage erfolgreich ist.
- Funktion 20) `test_get_stundenplan_exception`
  Umsetzung: Simuliert einen Datenbankverbindungsfehler und stellt sicher, dass die Funktion get_stundenplan None zurückgibt und ein Rollback durchgeführt wird.
- Funktion 21) `test_delete_stundenplan_success`
  Umsetzung: Überprüft, ob die Funktion delete_stundenplan erfolgreich ist und einen positiven Rückgabewert liefert, wenn die Datenbankabfrage erfolgreich ist.
- Funktion 22) `test_delete_stundenplan_exception`
- Umsetzung: Simuliert einen Datenbankverbindungsfehler und stellt sicher, dass die Funktion delete_stundenplan None zurückgibt.
- Funktion 23) `test_print_stundenplan_success`
  Umsetzung: Überprüft, ob die Funktion print_stundenplan erfolgreich ist und True zurückgibt, wenn alle Datenbankabfragen und Hilfsfunktionen erfolgreich sind.
- Funktion 24) `test_print_stundenplan_error`
  Umsetzung: Simuliert einen Datenbankverbindungsfehler und stellt sicher, dass die Funktion print_stundenplan None zurückgibt.

- **Datei - test_veranstaltung.py**
  - Funktion 1) `test_get_all_veranstaltungen_success`
    Umsetzung: Überprüft, ob die Funktion get_all_veranstaltungen erfolgreich alle Veranstaltungen aus der Datenbank abruft und die erwarteten Werte zurückgibt.
  - Funktion 2) `test_get_veranstaltung_success`
    Umsetzung: Testet, ob die Funktion get_veranstaltung erfolgreich eine spezifische Veranstaltung aus der Datenbank abruft und die erwarteten Werte zurückgibt, wenn eine gültige Veranstaltungs-ID übergeben wird.
  - Funktion 3) `test_get_veranstaltung_failure`
    Umsetzung: Überprüft, ob die Funktion get_veranstaltung None zurückgibt, wenn eine ungültige Veranstaltungs-ID übergeben wird und die Datenbankabfrage fehlschlägt.

- **Datei - test_vertretungsplan.py**
  - Funktion 1) `test_insert_vertretungsplan_success`
    Umsetzung: Überprüft, ob die Funktion insert_vertretungsplan erfolgreich ist und einen positiven Rückgabewert liefert, wenn gültige Eingabewerte übergeben werden.

- Funktion 2) `test_insert_vertretungsplan_invalid_wochentag`
  Umsetzung: Testet, ob die Funktion insert_vertretungsplan None zurückgibt, wenn ein ungültiger Wochentag (6) übergeben wird.
- Funktion 3) `test_insert_vertretungsplan_invalid_startzeit`
  Umsetzung: Überprüft, ob die Funktion insert_vertretungsplan None zurückgibt, wenn eine ungültige Startzeit (7 Uhr) übergeben wird.
- Funktion 4) `test_insert_vertretungsplan_invalid_endzeit`
  Umsetzung: Stellt sicher, dass die Funktion insert_vertretungsplan None zurückgibt, wenn eine ungültige Endzeit (17 Uhr) übergeben wird.
- Funktion 5) `test_insert_vertretungsplan_invalid_veranstaltung_id`
  Umsetzung: Testet, ob die Funktion insert_vertretungsplan None zurückgibt, wenn eine ungültige Veranstaltungs-ID (0) übergeben wird.
- Funktion 6) `test_insert_vertretungsplan_invalid_mitarbeiter_id`
  Umsetzung: Überprüft, ob die Funktion insert_vertretungsplan None zurückgibt, wenn eine ungültige Mitarbeiter-ID (0) übergeben wird.
- Funktion 7) `test_insert_vertretungsplan_exception`
  Umsetzung: Simuliert einen Datenbankverbindungsfehler und stellt sicher, dass die Funktion insert_vertretungsplan None zurückgibt und ein Rollback durchgeführt wird.
- Funktion 8) `test_get_vertretungsplan_success`
  Umsetzung: Überprüft, ob die Funktion get_vertretungsplan erfolgreich ist und die erwarteten Daten zurückgibt, wenn die Datenbankabfrage erfolgreich ist.
- Funktion 9) `test_get_vertretungsplan_exception`
  Umsetzung: Simuliert einen Datenbankverbindungsfehler und stellt sicher, dass die Funktion get_vertretungsplan None zurückgibt und ein Rollback durchgeführt wird.
- Funktion 10) `test_delete_vertretungsplan_success`
  Umsetzung: Überprüft, ob die Funktion delete_vertretungsplan erfolgreich ist und einen positiven Rückgabewert liefert, wenn die Datenbankabfrage erfolgreich ist.
- Funktion 11) `test_delete_vertretungsplan_exception`
  Umsetzung: Simuliert einen Datenbankverbindungsfehler und stellt sicher, dass die Funktion delete_vertretungsplan None zurückgibt.
- Funktion 12) `test_print_vertretungsplan_success`
  Umsetzung: Überprüft, ob die Funktion print_vertretungsplan erfolgreich ist und True zurückgibt, wenn alle Datenbankabfragen und Hilfsfunktionen erfolgreich sind.
- Funktion 13) `test_print_vertretungsplan_error`
  Umsetzung: Simuliert einen Datenbankverbindungsfehler und stellt sicher, dass die Funktion print_vertretungsplan None zurückgibt.