

Universität Regensburg

Lehrstuhl für Wirtschaftsinformatik III – Business Engineering

Seminar: Projektseminar

Dozentin: Leist, Susanne

Projektbetreuer: Berg, Volker



Universität Regensburg

Programmierung/Integration der vier Seminarvergabetools auf ein Webportal in Python

Projektseminar

von: **Scharlach**, Sabrina

Matrikel-Nr.: 2287302

Studienfach: Wirtschaftsinformatik B.Sc.

Adresse: Bruderwöhrdstraße 16

93055, Regensburg

E-Mail: Sabrina.Scharlach@stud.uni-regensburg.de

Inhaltsverzeichnis

1 Einleitung.....	3
2 Projektplanung	3
2.1 Anforderungsanalyse.....	3
2.2 Nutzergruppen und Use Cases	5
2.3 Gantt-Projektplan	6
2.4 Requirements, Virtualenv und Technologieauswahl	6
3 Projektdurchführung	7
3.1 Aufbau der Webapplikation.....	7
3.2 Architektur der Webapplikation (Frontend, Backend)	7
3.2.1 Model	8
3.2.2 View	14
3.2.3 Controller.....	22
4 Projektabschluss	27
4.1 User Tests.....	27
4.2 Next Steps.....	27
5 Quellen	29

1 Einleitung

Diese Dokumentation wurde im Rahmen meines Projektseminars "Programmierung/Integration der vier Seminarvergabetools auf ein Webportal in Python" erstellt. Sie dient als Beschreibung der entwickelten Webapplikation mit Schwerpunkt auf der Durchführung des Projekts, einschließlich des Projektplans und der Anforderungsanalyse, sowie auf der detaillierten Darstellung des Aufbaus der Flask-Webapplikation nach dem MVC-Prinzip (Model-View-Controller). Darüber hinaus soll sie als Ressource zur Unterstützung bei der Weiterentwicklung der Applikation genutzt werden können, indem sie notwendigen Informationen und Erklärungen der technischen Umsetzung bereitstellt.

Derzeit nutzt die Universität Regensburg vier separate Tools, um den jeweiligen Vergabeprozess für diese Art von Abschlussarbeiten oder Seminaren abzubilden. Diese Tools sollen in einem einheitlichen System integriert werden, das alle erforderlichen Funktionen für Studenten, Lehrstuhlmitarbeiter und Systemadministratoren umfasst und während des gesamten Vergabeprozess effizient unterstützt.

Der Vergabeprozess für Seminare und Abschlussarbeiten an der Uni Regensburg besteht aus den folgenden 3 Phasen:

1. **Registrierungsphase:** In der ersten Phase registrieren sich die Studenten für die Teilnahme am Vergabeprozess, indem sie ein Registrierungsformular ausfüllen. Diese Daten sind für das laufende Semester nicht mehr änderbar. Gleichzeitig tragen die Mitarbeiter die Themen für ihren Lehrstuhl in das System ein.
2. **Themenwahlphase:** In der zweiten Phase priorisieren die Studenten ihre gewünschten Themen nach den Themenvorstellungen, die in Phase 1 stattfanden. Nach der Anmeldung im System können die Studenten, falls erforderlich (z.B. bei Projekt- oder Praxisseminaren), Gruppen bilden und ihre bevorzugten Themen auswählen.
3. **Themenvergabe:** In der dritten Phase werden die Themen basierend auf den angegebenen Priorisierungen und durch Zufall den Studenten zugeteilt. Wenn es mehr Bewerber als verfügbare Plätze gibt, entscheidet das Los. Sowohl die Projektbetreuer als auch die Studenten werden über die Zuteilung informiert. Studenten, die kein Thema zugeteilt bekommen haben, haben die Möglichkeit, ein verbliebenes Thema zu wählen oder gegen freie Plätze zu tauschen.

2 Projektplanung

2.1 Anforderungsanalyse

Für die erfolgreiche Umsetzung des Seminarvergabesystems wurde eine detaillierte Anforderungsliste erstellt, die sowohl funktionale (F) als auch nicht-funktionale (NF) Anforderungen umfasst. Diese Liste dient als Grundlage für die Projektarbeit und ermöglicht es, den Projektstatus effektiv zu verfolgen und am Ende zu überprüfen, ob alle definierten Anforderungen erfüllt wurden.

Die Priorisierungen wurden nach der MoSCoW Methode umgesetzt:

- **M** (Must have - zwingend erforderlich)
- **S** (Should have - Wichtig, aber nicht der aktuelle Fokus)
- **C** (Could have - Wünschenswert, falls noch Zeit und Ressourcen übrig)
- und **W** (Won't have - Nicht geplant, aber in Zukunft):¹

¹ [What is MoSCoW Prioritization? | Overview of the MoSCoW Method \(productplan.com\)](https://productplan.com/moscow-prioritization/)

Anforderung	F / NF	Prio
Mit dem Vergabesystem sollen sich Nutzer auch ohne Login über Themen informieren können und diese nach Art oder einem Keyword filtern können.	F	S
Nutzer sollen sich über den Vergabeprozess informieren können, ohne eingeloggt zu sein.	F	S
Nutzer müssen sich für die Nutzung weitere Funktionen authentifizieren, indem sie ihre NDS-Kennung und ihr Passwort eingeben.	F	M
Studenten sollen sich für den Vergabeprozess registrieren können, indem sie ihre persönlichen Daten eingeben und die Art der Arbeit auswählen.	F	M
Studenten sollen für ihre ausgewählten Arten der Arbeit Themen priorisieren können.	F	M
Studenten sollen das Ergebnis der Themenvergabe einsehen können und erfahren, ob ihnen ein Thema zugewiesen wurde.	F	W
Studenten sollen ein verbliebenes Thema wählen können, falls ihnen kein Thema zugewiesen wurde.	F	W
Studenten werden über eine interaktiven Prozessabbildung auf ihre Aufgaben hingewiesen.	F	C
Mitarbeiter sollen Themen für ihren Lehrstuhl verwalten können, einschließlich Hinzufügen, Bearbeiten und Löschen von Themen.	F	M
Mitarbeiter sollen andere Mitarbeiter für ihren Lehrstuhl verwalten können.	F	M
Administratoren sollen Studenten verwalten können, einschließlich Hinzufügen, Bearbeiten und Löschen von Studenten.	F	M
Administratoren sollen die Priorisierungen der Studenten bearbeiten können, sowie deren ausgewählte Arten	F	M
Administratoren sollen Lehrstühle verwalten können, einschließlich Hinzufügen, Bearbeiten und Löschen von Lehrstühlen.	F	M
Administratoren sollen alle Mitarbeiter verwalten können, einschließlich Hinzufügen, Bearbeiten und Löschen von Lehrstühlen.	F	M
Administratoren sollen die Zeitdaten für die unterschiedlichen Phasen verwalten können.		M
Administratoren sollen den Themenvergabeprozess starten können.	F	W
Beteiligte (Studenten, Mitarbeiter) sollen über das Ergebnis der Themenvergabe informiert werden.	F	W
Die Webapplikation muss auf dem Flask Framework basieren und in Python entwickelt werden.	NF	M
Die Webapplikation muss nach dem Model-View-Controller (MVC) Prinzip aufgebaut sein.	NF	M
Die Webapplikation muss ein responsives Design haben, das in FIDS-Farben gestaltet ist und auf allen Bildschirmgrößen gut funktioniert.	NF	M
Die Webapplikation muss leicht verständlich und intuitiv bedienbar sein.	NF	M
Die Authentifizierung soll mithilfe des ldap-Servers der Uni Regensburg erfolgen.	NF	M
Es soll sichergestellt werden, dass keine SQL Injections möglich sind.	NF	S
Sensible Daten von Studenten und anderen Nutzern dürfen nicht auslesbar sein.		
Es muss sichergestellt werden, dass keine unautorisierten Nutzer auf Mitarbeiter- oder Adminbereiche zugreifen können (Zugriffskontrolle).	NF	S
Es soll sichergestellt werden, dass der Zugriff auf bestimmte Funktionen wie Registrierung und Themenwahl nur innerhalb einer bestimmten Zeitperiode möglich ist.	NF	S
Die Webapplikation soll auf einem Server deployed werden.	NF	W

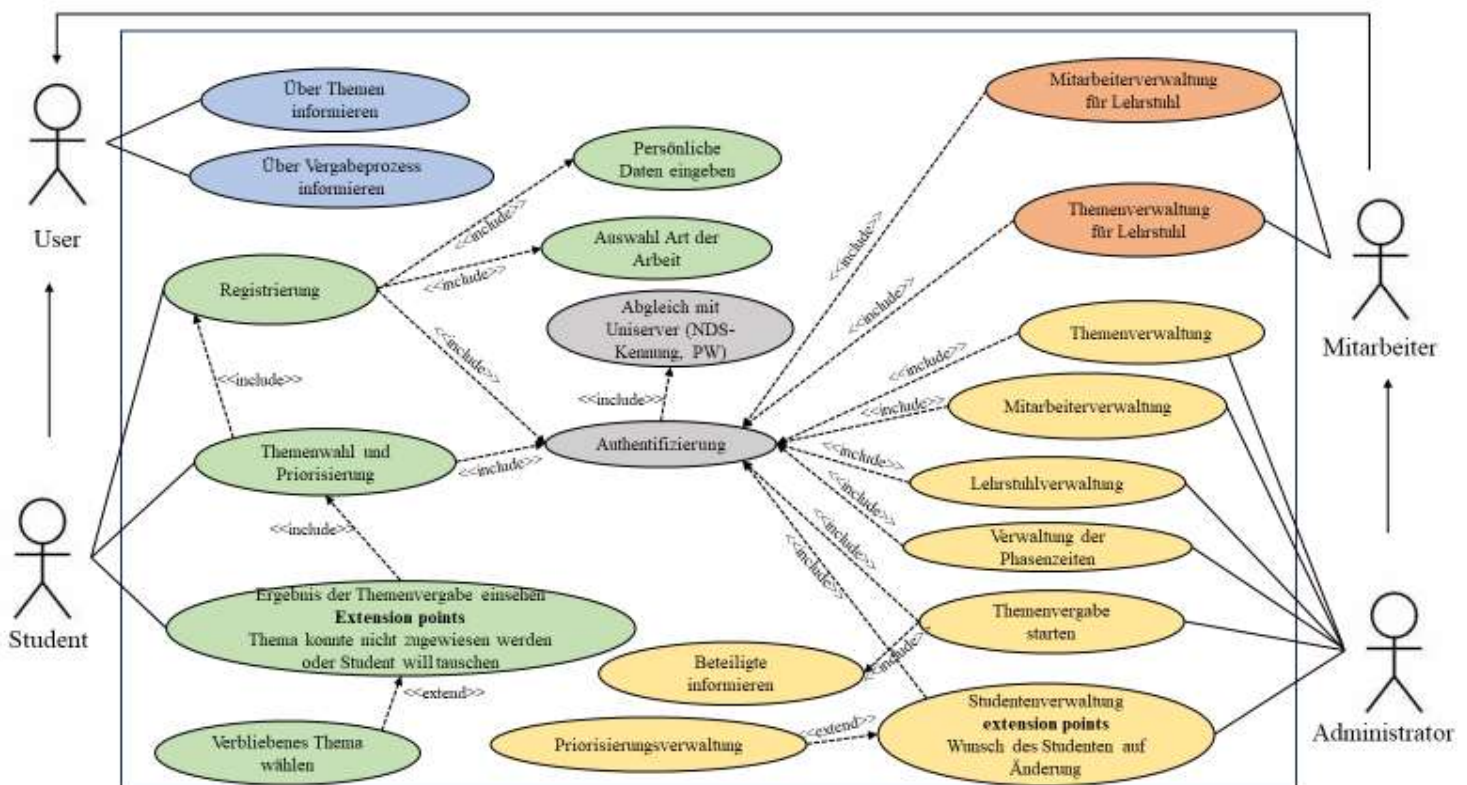
In Zusammenarbeit mit dem Projektbetreuer wurde im Vorfeld eine Priorisierung der Anforderungen vorgenommen. Um den Umfang des Projektseminars nicht zu überschreiten, wurde beschlossen, die Implementierung der Webapplikation zunächst auf Phase 1 (Registrierung) und Phase 2 (Themenwahl) zu fokussieren. Daher wurden die Anforderungen, die sich auf Phase 3 (Themenvergabe) und das Deployment beziehen, mit "W" (Won't have) gekennzeichnet.

2.2 Nutzergruppen und Use Cases

Bestandteil der Anforderungsanalyse ist nicht nur die Identifikation der benötigten Funktionen und technischen Systemanforderungen, sondern auch die der beteiligten Akteure. Innerhalb des Seminarvergabesystems wurden die folgenden 4 Akteure unterschieden:

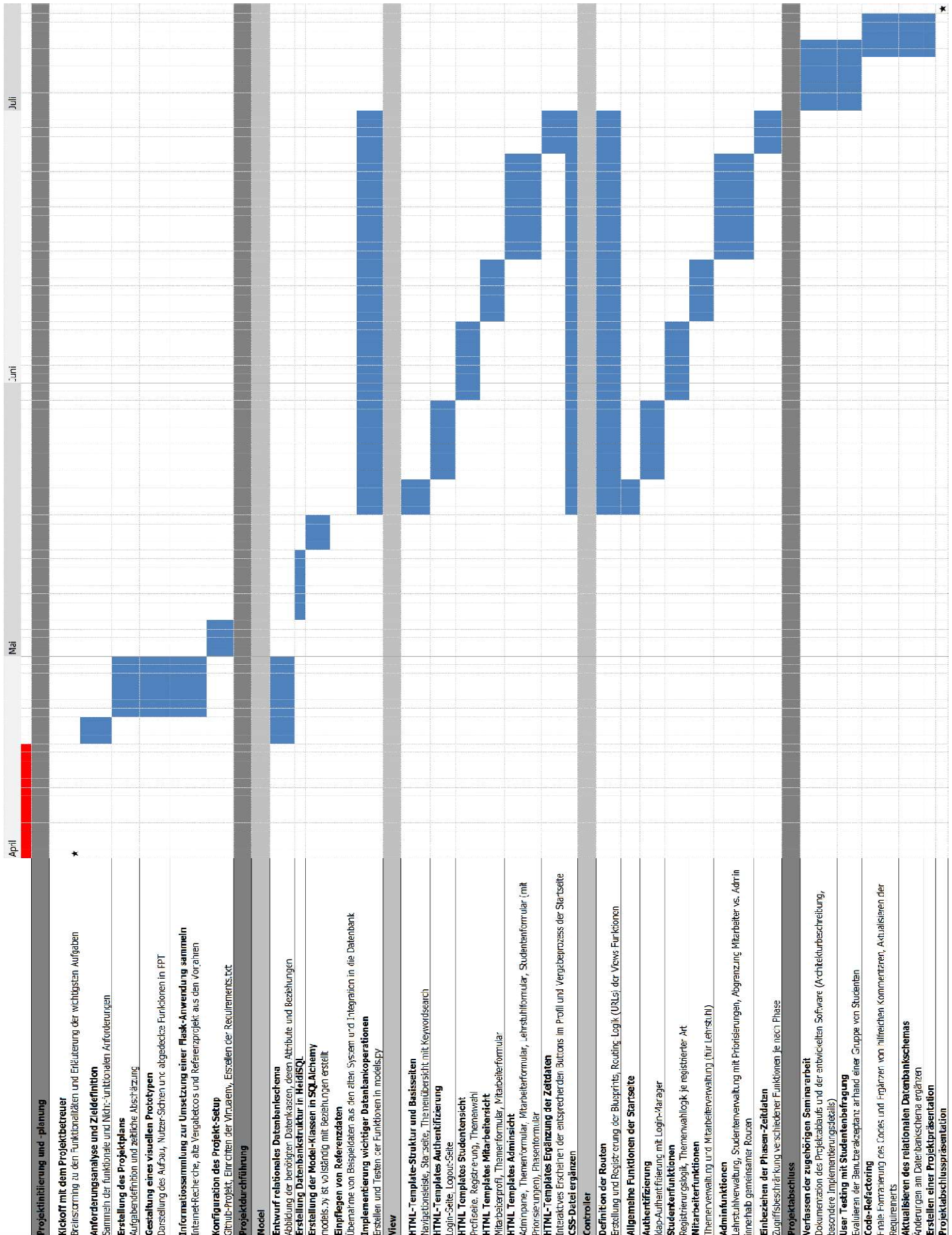
	Nutzer (ohne Authorisierung)	Studenten	Mitarbeiter	Administrator
Aufgaben	Informationen zum Vergabeprozess und zu den verfügbaren Themen abrufen.	Registrierung, Auswahl und Priorisierung von Themen, Einsehen der Vergabeergebnisse	Verwaltung der Themen und Betreuer des eigenen Lehrstuhls	Gesamte Verwaltung und Steuerung des Vergabeprozesses (Nutzer, Themen, Phasen)
Häufigkeit	selten, meist vor der Registrierung oder Themenwahl	einmalige bis dreimalige Nutzung während des Studiums	regelmäßig	regelmäßig

Ein UML Case Diagramm ist eine effektive Methode zur Darstellung der rollenbasierten Anforderungen, da es sich auf die Interaktionen zwischen einem System und seinen Benutzern fokussiert und so einen Überblick über die verschiedenen Verantwortlichkeiten und Aufgaben bietet.



2.3 Gantt-Projektplan

Der folgende Projektplan (siehe Projekt-Anhang) bildet die zeitliche Planung innerhalb den beiden bekannten Meilensteinen Projektkickoff (09.04.) und Projektpräsentation (12.07.) ab:



2.4 Requirements und Technologieauswahl

Zu Beginn des Projekts wurde eine Virtual Environment (virtuelle Umgebung) eingerichtet, die es ermöglicht, projektspezifische Python-Abhängigkeiten isoliert zu verwalten. Die Einrichtung und Installation der Abhängigkeiten basierten auf den Anforderungen des Projektbetreuers und des Referenzprojekts „Spindvergabesystem“. Diese Abhängigkeiten wurden anfangs angepasst und am Ende des Projekts aktualisiert.

Die Datei requirements.txt wird in Python-Projekten verwendet, um alle benötigten Pakete und deren spezifische Versionen aufzulisten. Dies ermöglicht es anderen Entwicklern (oder Nutzern des Codes), die notwendigen Abhängigkeiten einfach zu installieren, indem sie folgenden Befehl ausführen:

```
pip install -r requirements.txt
```

Sobald beispielsweise die Virtual Environment durch den Befehl „.venv\Scripts\activate.bat“ aktiviert ist, müssen lediglich die Abhängigkeiten aus der requirements.txt Datei mit pip installiert werden.²

Für das Seminarvergabetool sind folgende Python-Abhängigkeiten relevant:

Flask-Webframework	Flask == 3.0.3 Flask-Login == 0.6.3 Flask-SQLAlchemy == 3.1.1 Flask-WTF == 1.2.1 Jinja2 == 3.1.4 Werkzeug == 3.0.3 itsdangerous == 2.2.0 click == 8.1.7
Datenbank	SQLAlchemy == 2.0.31 mysqlclient == 2.2.4 greenlet == 3.0.3
Authentifizierung	python-ldap == 3.4.4 pyasn1 == 0.5.1 pyasn1-modules == 0.3.0
Weitere Hilfsbibliotheken	typing_extensions == 4.9.0 MarkupSafe == 2.1.5 blinker == 1.8.2 email-validator== 2.1.0.post1

3 Projektdurchführung

3.1 Start der Webapplikation

Da ein Deployment auf einem Server in diesem Projekt noch nicht vorgesehen ist, kann die Webapplikation lokal über die Datei run.py gestartet werden, welche eine main-Funktion zum Starten der App enthält. Über das Terminal kann die Webapplikation deshalb mit folgendem Befehl aus dem Projektverzeichnis heraus gestartet werden:

```
python run.py
```

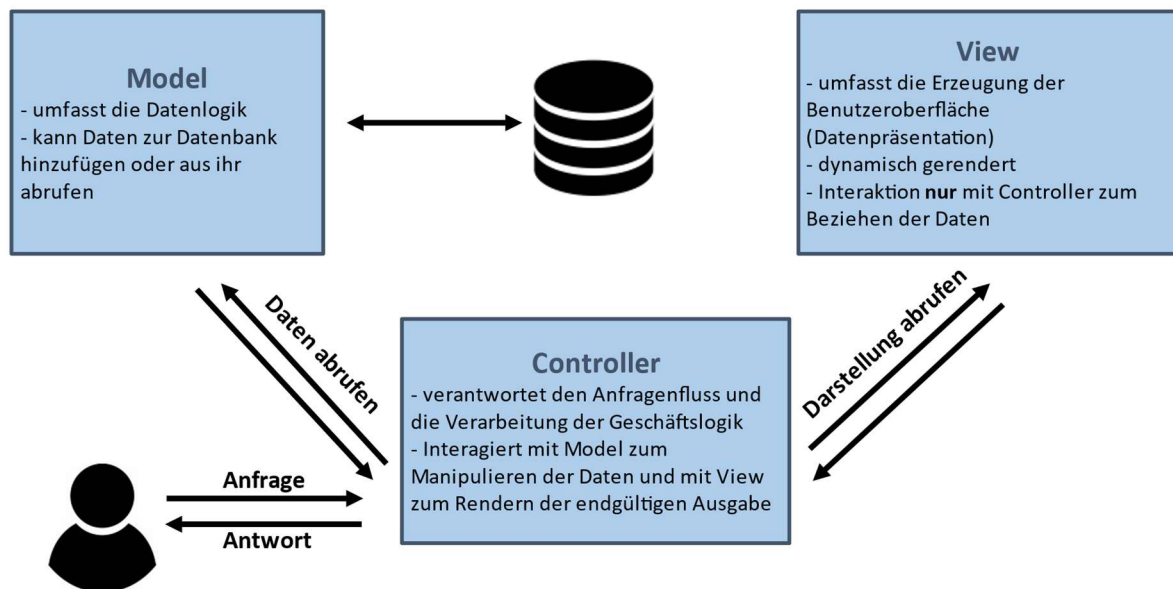
Der Zugriff erfolgt anschließend über den Webbrowser (localhost):

```
http://127.0.0.1:5000/
```

² [Setting Up Your Python Environment With Venv and requirements.txt - Frank's Blog \(frankcorso.dev\)](https://frankcorso.dev/setting-up-your-python-environment-with-venv-and-requirements.txt)

3.2 Architektur der Webapplikation (Frontend, Backend)

In den letzten Jahren sind Webanwendungen von einfachen HTML-Seiten zu komplexen Anwendungen gewachsen. Um diese Komplexität zu bewältigen, nutzen Entwickler meist das MVC (Model-View-Controller) Designmuster. MVC teilt eine Anwendung in drei Hauptkomponenten: Model, View Controller. Durch MVC werden somit Geschäftslogik, User-Interface Logik und Eingabelogik klar voneinander getrennt, was die Wartung und Testbarkeit der einzelnen Komponenten aufgrund ihrer Modularität erheblich erleichtert.³



Innerhalb meiner Web-Applikation wurde das MVC-Framework wie folgt umgesetzt:

3.2.1 Model

Die Model-Komponente meiner Webanwendung wurde mithilfe von SQLAlchemy, einer leistungsfähigen Python-Erweiterung zur objektrelationalen Abbildung (ORM), implementiert. SQLAlchemy dient als Schnittstelle zu meiner Datenbank und ermöglicht es, Datenbankoperationen in Python zu kapseln, indem die entsprechenden Aufrufe von Python-Klassen automatisch in SQL-Befehle umgewandelt werden.

Durch den Einsatz von SQLAlchemy als ORM-Framework wird eine klare Trennung zwischen den benötigten Daten und den auszuführenden Befehlen erreicht, was mehrere Vorteile mit sich bringt: Zum einen bietet SQLAlchemy eine erhöhte Sicherheit der Dateneingaben und verfügt über eingebaute Schutzmechanismen gegen SQL-Injections und zum anderen übernimmt SQLAlchemy die Umwandlung von Python-Ausdrücken in SQL-Abfragen.⁴

Datenbankmodell und Erstellung der mySQL-Datenbank

Einige Klassen des Datenbankmodells wurden zu Beginn vom Projektbetreuer vorgegeben. Diese Vorgaben wurden durch die Ergänzung weiterer Klassen und Attribute erweitert, um das finale

³ [MVC Framework Introduction - GeeksforGeeks](#)

⁴ [SQLAlchemy: Die mächtige ORM-Bibliothek für Python \(gpt5.blog\)](#)

Datenbankschema zu erstellen. Die Datenbank wurde im DBMS HeidiSQL entwickelt, indem die Tabellen manuell erstellt und die Beziehungen sowie Fremdschlüssel gepflegt wurden.

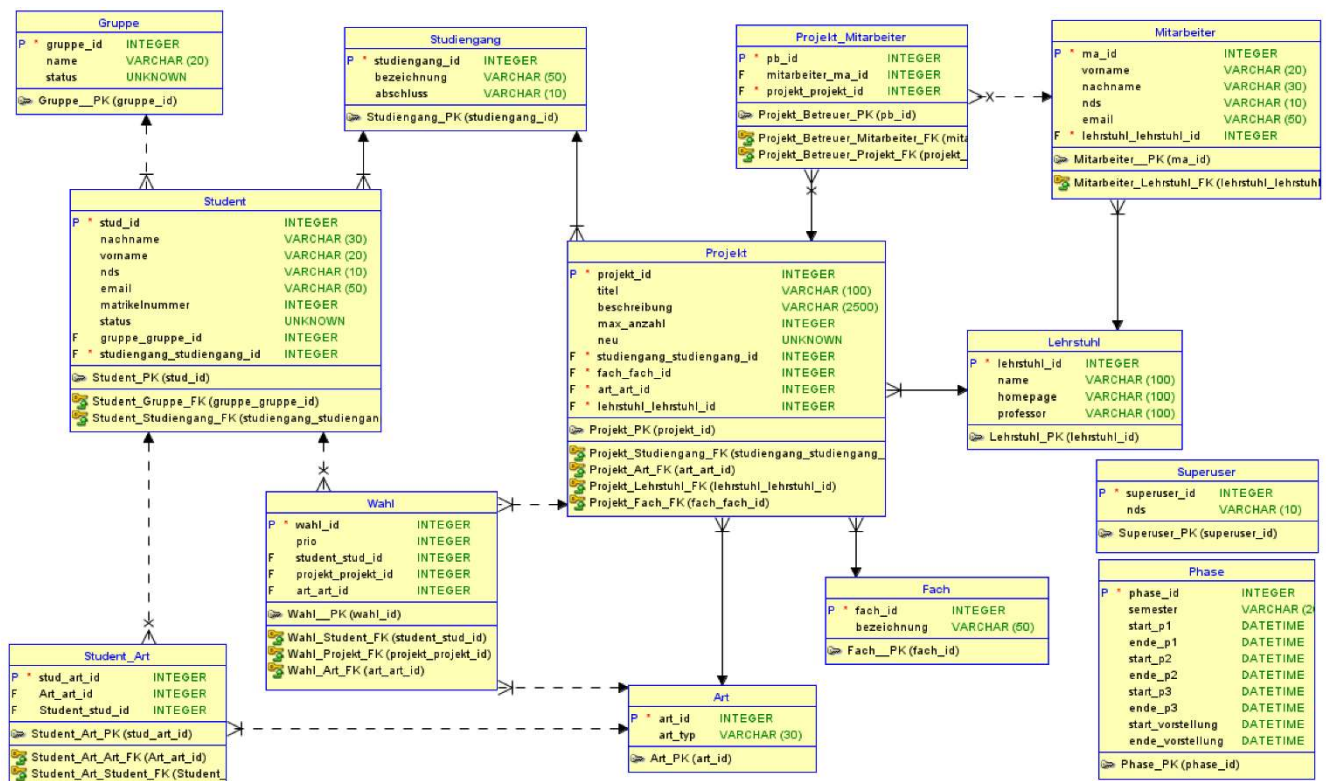


Abbildung 1: Relationales Datenbankmodell

Dieses Modell umfasst alle für die Vergabe notwendigen Daten, einschließlich der des Vergabeprozesses. Obwohl die Tabelle "Gruppe" bereits implementiert wurde, wird sie erst bei der Themenvergabe relevant, weshalb die Fremdschlüssel in der Tabelle "Student" vorerst standardmäßig auf NULL gesetzt wurden.

Um die Datenbank gegen Integritätsverlust zu schützen und Konsistenz zu gewährleisten, wurden außerdem bestimmte Arten von Löschrbeziehungen (Cascade und Restrict) in der Datenbank, als auch zu einem späteren Zeitpunkt in der `models.py` definiert:

Cascade-Beziehungen

Die Cascade-Beziehungen wurden implementiert, um verbundene Einträge in Zwischentabellen automatisch zu löschen und somit Dateninkonsistenzen zu vermeiden. Beispielsweise werden am Ende jedes Semesters alle Studenten gelöscht, wodurch auch die zugehörigen Einträge in den Zwischentabellen (Art-Auswahl, Priorisierungen der Wahl) entfernt werden.⁵

Student → Wahl

Student → Student_Art

Mitarbeiter → Projekt_Mitarbeiter

Projekt → Projekt_Mitarbeiter

⁵[MySQL – ON DELETE CASCADE-Einschränkung – GeeksforGeeks](#)

Die Umsetzung dieser Beziehungen ist im vorherigen Beispiel der Projektbetreuer ersichtlich. Es ist wichtig, dass das Cascade-Verhalten lediglich bei den Fremdschlüsseln der Beziehungstabelle definiert wird, um zu verhindern, dass auch die Parent-Objekte der Zwischentabelle gelöscht werden.

Restrict-Beziehungen

Um zu vermeiden, dass Daten, die in anderen Tabellen referenziert werden, versehentlich gelöscht werden, werden Restrict-Beziehungen eingesetzt. Ein Projekt soll beispielsweise **nicht** gelöscht werden können, wenn es in der Wahl-Tabelle referenziert wird (d.h. Studenten haben im aktuellen Semester bereits ihre Priorisierung abgegeben). Um dennoch veraltete Projekte zu kennzeichnen und aus neuen Vergabeprozessen herauszufiltern, wurde das boolesche Attribut `alt` eingeführt. Auch wird aufgrund der Cascade-Beziehung von Student-Wahl die Wahl-Tabelle am Ende jedes Semesters geleert.

Projekt → Wahl

No Action Beziehungen

Alle weiteren Tabellen verwenden die standardmäßigen NO ACTION-Beziehungen, die ähnlich wie RESTRICT-Beziehungen funktionieren. Diese Einstellungen sorgen dafür, dass keine wichtigen Daten im System gelöscht werden können. Zudem sind ein Großteil der Daten statisch, wie z.B. Studiengänge und Arten von Seminararbeiten, weshalb keine besonderen Löschbeziehungen implementiert wurden.

Datenbankkonfiguration mittels SQLAlchemy

Die Datenbank wurde ursprünglich mit HeidiSQL erstellt, sodass lediglich eine Konfiguration innerhalb der Webapplikation notwendig ist. Diese Konfiguration erfolgt in der Datei `__init__.py`, welche unter anderem SQLAlchemy einrichtet und durch die Definition der Verbindungsparameter mit der bestehenden Datenbank verknüpft, wodurch eine nahtlose Integration gewährleistet wird:

```
db_user = 'root'
db_password = '2024'
db_coded_password = quote_plus(db_password)
db_host = 'localhost'
db_database = 'seminarvergabetool'
app.config["SQLALCHEMY_DATABASE_URI"] =
f"mysql://{db_user}:{db_coded_password}@{db_host}/{db_database}"
app.config["SQLALCHEMY_TRACK_MODIFICATIONS"] = False
app.config["SECRET_KEY"] = "psc9BNcMna6mQx41zNILLaYY4x1TVXIj"
```

Nach der Konfiguration der Datenbankverbindung wird die Datenbankinstanz `db` aus SQLAlchemy mit der Flask-Webanwendung verknüpft. Obwohl ich das Datenbankschema bereits manuell in der Datenbank implementiert habe, ermöglicht der Befehl `create_all()` die automatische Erstellung der Tabellen und ihrer Beziehungen beim Starten der Webanwendung. Dies erfolgt anhand der in der `models.py` definierten Klassen.

```
db.init_app(app)
with app.app_context():
    db.create_all()
```

Models.py

Innerhalb der `models.py` sind alle Klasse der Datenbank mit ihren zugehörigen Attributen und Datentypen abgebildet. Zudem werden auch hier die Beziehungen untereinander anhand ihrer

Fremdschlüssel definiert. Die Beziehungen zwischen den einzelnen Klassentabellen wurden nach folgendem Schema umgesetzt:

1:n Beziehungen

Die Verknüpfung der 1:n Beziehungen wurde, wie bei Datenbanken üblich, auf der n-Seite als Fremdschlüssel definiert und auch dort die Beziehungsfunktion `db.relationship()` deklariert, welche den direkten Zugriff auf die verknüpften Daten ermöglicht und auf diese Weise Join-Operationen vermeiden kann.

Jedes Projekt ist beispielsweise einer Art zugeordnet, während zu einer Art mehrere Projekte existieren können. Mittels `projekt.art` kann nun auf die jeweilige Art eines Projektes zugegriffen werden und aufgrund des `backref` gilt dies auch für die Gegenseite der Beziehung, weswegen mit `art.projekte` eine Liste der Projekt-Objekte zurückgegeben wird, die zu dieser Art gehören.⁶

```
class Projekt(db.Model):
...
art_art_id = db.Column(db.Integer, db.ForeignKey('art.art_id'),
nullable=False)
...
art = db.relationship('Art', backref=db.backref('projekte',
lazy=True))
```

N:m Beziehungen

Die n:m Beziehungen wurden nicht direkt in der Zwischentabelle definiert, obwohl diese die Fremdschlüssel enthält. Stattdessen wurden die `db.relationship`-Beziehungen auf den Seiten der Parent-Tabellen definiert und durch die Angabe des `secondary`-Parameters wird auf die entsprechende Zwischentabelle verwiesen und die Parent-Tabellen verknüpft. Dadurch kann beispielsweise mit `mitarbeiter.betreute_projekte` direkt auf die Liste der dem Mitarbeiter zugeordneten Projekte zugegriffen werden und umgekehrt mit `projekt.betreuer` auf die dem Projekt zugeordneten Betreuer.

```
class Mitarbeiter(db.Model):
...
betreute_projekte = db.relationship('Projekt',
secondary='projekt_mitarbeiter', back_populates='betreuer')

class ProjektMitarbeiter(db.Model):
    __tablename__ = 'projekt_mitarbeiter'
...
mitarbeiter_ma_id = db.Column(db.Integer,
db.ForeignKey('mitarbeiter.ma_id', ondelete='CASCADE'),
nullable=False)
projekt_projekt_id = db.Column(db.Integer,
db.ForeignKey('projekt.projekt_id', ondelete='CASCADE'),
nullable=False)

class Projekt(db.Model):
...
betreuer = db.relationship('Mitarbeiter',
secondary='projekt_mitarbeiter', back_populates='betreute_projekte')
```

⁶ [Declaring Models — Flask-SQLAlchemy Documentation \(2.x\) \(palletsprojects.com\)](#)

Die Tabelle „Wahl“ stellt eine Ausnahme dieses Schemas dar, da sie drei verschiedene Fremdschlüssel enthält. Außerdem wird die Klasse „Wahl“ innerhalb der Implementierung der Webanwendung häufig direkt genutzt, um die Priorisierungen der Studenten abzubilden. Aus diesem Grund wurden die Beziehungsreferenzen direkt in der Klasse „Wahl“ definiert.

```
class Wahl(db.Model):
...
student_stud_id = db.Column(db.Integer,
db.ForeignKey('student.stud_id', ondelete='CASCADE'),
nullable=False)
projekt_projekt_id = db.Column(db.Integer,
db.ForeignKey('projekt.projekt_id', ondelete='RESTRICT'),
nullable=False)
art_art_id = db.Column(db.Integer, db.ForeignKey('art.art_id'),
nullable=False)

student = db.relationship('Student', backref=db.backref('wahl',
lazy=True, cascade='all, delete-orphan'))
projekt = db.relationship('Projekt', backref=db.backref('wahl',
lazy=True))
art = db.relationship('Art', backref=db.backref('wahl'), lazy=True)
```

Wichtige Funktionen der models.py

Um Datenbankoperationen zu vereinfachen und konkrete Funktionen isoliert testen zu können, wurden innerhalb der folgenden Hilfsmethoden zur Interaktion mit der Datenbank implementiert:

Studiengang	<ul style="list-style-type: none"> • <code>get_all_studiengaenge()</code>: Rückgabe einer Liste aller Studiengänge
Fach	<ul style="list-style-type: none"> • <code>get_all_faecher()</code>: Rückgabe einer Liste aller Fächer
Art	<ul style="list-style-type: none"> • <code>get_all_arten()</code>: Rückgabe einer Liste aller unterschiedlichen Arten von Seminararbeiten • <code>get_color()</code>: Rückgabe der Farbe, die einer Art zugeordnet ist (Mitarbeiter-Profil) • <code>get_badge()</code>: Rückgabe des Badges (=farbliches Abzeichen) mit dem Anfangsbuchstabe und der entsprechenden Farbe der Art (Themenübersicht, Admin-Profil)
Lehrstuhl	<ul style="list-style-type: none"> • <code>get_all_lehrstuehle()</code>: Rückgabe einer Liste aller Lehrstühle • <code>get_lehrstuhl_projekte(self)</code>: Rückgabe aller Projekte eines bestimmten Lehrstuhls (=self) • <code>create_lehrstuhl(name, homepage, professor)</code>: Erstellen eines Lehrstuhls • <code>edit_lehrstuhl(name, homepage, professor)</code>: Bearbeiten eines bestehenden Lehrstuhls • <code>delete_lehrstuhl()</code>: Löschen eines bestehenden Lehrstuhls
Student	<ul style="list-style-type: none"> • <code>get_selected_arten(self)</code>: Rückgabe einer Liste der ausgewählten Arten eines Student • <code>get_all_Students()</code>: Rückgabe einer Liste aller Studenten • <code>get_student(nds)</code>: Rückgabe des Student anhand der NDS-Kennung • <code>create_student(vorname, nachname, nds, email, matrikelnummer, studiengang_id)</code>: Erstellung eines Studenten

	<ul style="list-style-type: none"> • <code>edit_student(self, vorname, nachname, nds, email, matrikelnummer, studiengang_id)</code>: Bearbeiten eines Studenten • <code>delete_student()</code>: Löscht einen bestehenden Studenten. • <code>add_arten(self, arten_ids)</code>: Hinzufügen bestimmter Arten (<code>arten_ids</code>) zu den ausgewählten Arten eines bestehenden Studenten • <code>remove_art(self, art_id)</code>: Entfernen einer bestimmten Art (<code>art_id</code>) aus den ausgewählten Arten eines bestehenden Student
Projekt	<ul style="list-style-type: none"> • <code>get_projects_by(search_query, art_filter, lehrstuhl_filter)</code>: Rückgabe von Projekten gefiltert nach einem Stichwort, ihrer Art oder ihrem zugehörigen Lehrstuhl • <code>apply_keyword_search(projekte, search_query)</code>: Durchführung einer Stichwortsuche für eine Liste an Projekten anhand von Titel, Fach,... • <code>order_ascending_by_name(projekte)</code>: Alphabetisches Sortieren der Projekte nach ihrem Name • <code>group_by_art()</code>: Gruppieren der Projekte nach ihrer Art • <code>group_by_lehrstuhl(projekte)</code>: Erstellen eines Dictionary, welches für jeden Lehrstuhl seine Projekte geordnet nach der Art zurückgibt (Verwendung für Themensuche) • <code>create_projekt(titel, beschreibung, max_anzahl, studiengang_id, fach_id, art_id, lehrstuhl_id, betreuer_ids)</code>: Erstellen eines neuen Projektes • <code>add_projekt_betreuer(self, mitarbeiter_ids)</code>: Hinzufügen von Betreuer zu einem Projekt (Einträge der Zwischentabelle ProjektMitarbeiter) • <code>edit_projekt(self, titel, beschreibung, max_anzahl, studiengang_id, lehrstuhl_id, art_id, fach_id, betreuer_ids)</code>: Bearbeiten eines Projektes • <code>delete_projekt(self)</code>: Löschen eines Projektes
Wahl	<ul style="list-style-type: none"> • <code>get_priorities_for_art(student_id, art_id)</code>: Rückgabe der Prioritäten eines Studenten für eine Art • <code>get_all_priorities(student_id)</code>: Rückgabe eines Dictionary mit den ausgewählten Arten als Keys und deren zugehörigen Priorisierungen • <code>save_priorities(student_id, art_id, priorities)</code>: Speichern der Prioritäten eines Studenten für eine bestimmte Art • <code>delete_priorities(student_id, art_id)</code>: Löschen der Prioritäten eines Studenten für eine bestimmte Art
Phase	<ul style="list-style-type: none"> • <code>get_current()</code>: Gibt die aktuelle Phase zurück • <code>phase1_active(self)</code>: Prüfung, ob Phase 1 des Vergabeprozesses aktiv ist • <code>phase2_active(self)/phase3_active(self)</code>: analog Phase 1 • <code>create_phasen(semester, jahr, start_p1, ende_p1, start_p2, ende_p2, start_p3, ende_p3, start_vorstellung, ende_vorstellung)</code>: Erstellung der Phasendaten für ein bestimmtes Semester • <code>edit_phasen(self, start_p1, ende_p1, start_p2, ende_p2, start_p3, ende_p3, start_vorstellung, ende_vorstellung)</code>: Bearbeiten der bestehenden Phasendaten • <code>delete(self)</code>: Löschen einer Phase
User	<ul style="list-style-type: none"> • <code>is_student(self)</code>: Prüfung, ob User ein Student ist • <code>is_mitarbeiter(self)</code>: Prüfung, ob User ein Mitarbeiter ist
Superuser	<ul style="list-style-type: none"> • <code>is_admin(nds)</code>: Überprüfung anhand der NDS-Kennung, ob User Administratorberechtigung besitzt

3.2.2 View

Die View-Komponente ist verantwortlich für die Darstellung der Benutzeroberfläche des Seminarvergabesystems und deren UI-Logik. Diese Komponente umfasst die HTML-Templates, welche sich in einem speziellen Verzeichnis befinden und dynamisch durch die Controller gerendert werden, sowie das gesamte CSS-Layout und kleinere Java-Script Implementationen.

HTML-Templates

In dieser Anwendung werden die Views mithilfe des Jinja2-Template-Engines in HTML implementiert und es wird so ermöglicht, dynamische Inhalte wie beispielsweise Python-Variablen oder Daten aus der Datenbank in die Seiten zu integrieren und darzustellen. Die Templates sind in einem klaren Verzeichnisbaum organisiert und folgendermaßen strukturiert:

```
templates/
├── base.html
├── index.html
├── auth/
│   ├── login.html
│   └── logout.html
├── error/
│   ├── 404.html
│   ├── not_authorized.html
│   ├── out_of_period.html
│   └── themenwahl_not_possible.html
├── forms/
│   ├── confirm_delete.html
│   ├── lehrstuhl_form.html
│   ├── mitarbeiter_form.html
│   ├── phasen_form.html
│   ├── priorities_form.html
│   ├── projekt_form.html
│   └── student_form.html
├── info/
│   ├── datenschutz.html
│   ├── impressum.html
│   ├── regeln.html
│   └── themen.html
├── mitarbeiter/
│   ├── admin.html
│   ├── not_authorized.html
│   └── profil.html
├── student/
│   ├── already_registered.html
│   ├── profil.html
│   ├── register.html
│   └── themenwahl.html
```

Das Design der Webanwendung wurde im Vorfeld durch mit dem Betreuer durch einen visuellen Prototyp abgestimmt, da dieser insbesondere für die Strukturierung der Templates von entscheidender Bedeutung ist. Da die Webapplikation von Studierende als auch Mitarbeitende genutzt wird, welche das System lediglich zu bestimmten Zeitpunkten im Semester verwenden, war

es wichtig, eine intuitive und benutzerfreundliche Oberfläche zu entwickeln, die auch ohne umfassende Schulung leicht verständlich ist und einen einfachen Zugriff auf die benötigten Funktionen ermöglicht.

Besonderer Fokus wurde außerdem auf die Umsetzung eines responsiven Designs gelegt, das sich flexibel an unterschiedliche Bildschirmgrößen anpasst und eine konsistente Umsetzung des FIDS-Farbschemas.

Grundstruktur *base.html*

Nach dem Konzept des Template Inheritance umgesetzt, bei dem alle Seiten einer Webapplikation (Templates) von einer Basisvorlage erben. Im Mastertemplates *base.html* befinden sich Verweise zu externen CSS-Stylesheets, die Definition der Navigationsleiste und des Footers.

Der Hauptinhalt der Seiten wird mithilfe des `block`-Tags definiert, welchen die erbenden Templates mit ihren spezifischen Inhalten überschreiben.⁷

```
<div class="w3-section">
  {%block body%} {%endblock%}
</div>
```

Startseite *index.html*

Die Startseite der Webanwendung soll dem Nutzer möglichst schnell die wichtigsten und für ihn relevanten Informationen bereitstellen. Neben einem Login-Button und Beschreibungen der verschiedenen Bereiche (Themen, Regeln, Profil) werden Buttons mit den verschiedenen Projektarten angezeigt, über die der Nutzer direkt einen Überblick über die für ihn relevanten, angebotenen Themen erhalten kann. Zudem ist der Vergabeprozess visuell dargestellt.

Die Startseite der Webanwendung bietet je nach Anmeldestatus des Benutzers unterschiedliche Informationen und Optionen. Nicht angemeldete Benutzer werden aufgefordert, sich einzuloggen, während angemeldete Benutzer nach dem Login mit ihren RZ-Daten interaktiv auf ihre aktuellen Aufgaben hingewiesen werden. Diese Aufgaben umfassen beispielsweise die Registrierung in Phase 1 oder die Themenpriorisierung in Phase 2, die durch das Klicken auf entsprechende Buttons direkt erledigt werden können.



Abbildung 2: Interaktiver Vergabeprozess der Startseite

Durch den Einsatz der Jinja-Abfragen wird die aktuelle Phase visuell durch die Klasse `"w3-fids-color"` hervorgehoben. Gleichzeitig wird sichergestellt, dass die Buttons nur dann sichtbar sind, wenn die entsprechende Phase aktiv ist und der Benutzer die erforderlichen Berechtigungen besitzt, andernfalls bleiben die Buttons durch die Klasse `"w3-hide"` verborgen.

Im vorherigen Beispiel soll der Button zur Themenwahl beispielsweise nur dann angezeigt werden,

⁷ [Template Designer Documentation — Jinja Documentation \(2.10.x\) \(palletsprojects.com\)](https://palletsprojects.com/en/2.10.x/jinja/)

wenn die Phase 2 aktiv ist und der Student eingeloggt (is_authenticated) und für die Vergabe registriert (d.h. Student) ist:

```
<div id="p2" class="w3-card-4">
  <header class="w3-container {% if phase2_active %}w3-fids-color{% else
  %}w3-grey{% endif %}">
    <h5>Themenwahl</h5>
  </header>

  <div class="w3-container">
    <p>In der zweiten Phase können Sie ....<br>
      Zeitraum: {{ phasen.start_p2.strftime('%d.%m.%Y %H:%M') }} bis
      {{ phasen.ende_p2.strftime('%d.%m.%Y %H:%M') }}</p>
    {% if current_user.is_authenticated and current_user.is_student %}
      <a id="button-p2" href="{% url_for('bp_student.themenwahl') %}"
      class="w3-button w3-fids-color w3-margin w3-round {% if not phase2_active
      %}w3-hide{% endif %}">Zur Themenwahl</a>
    {% endif %}
  </div>
</div>
```

Authentifizierung auth/

Login Template: Formular, welches von der Login-Route zur Validierung der Nutzerdaten am ActiveDirectory

Logout Template: Erscheint bei erfolgreichem Logout

Fehler error/

404 Template: Verwendung, wenn URL nicht gefunden werden kann (ErrorHandler)

Not authorized Template: Verwendung, wenn ein Mitarbeiter nicht über die erforderlichen Berechtigungen verfügt, um bestimmte Aktionen durchzuführen

Out-of-Phase Template: Verwendung, wenn die Funktionalität nur in einer bestimmten Phase des Prozesses nutzbar ist und diese im Moment nicht aktiv ist

Themenwahl not Possible Template: Verwendung, wenn keine Registrierung zu einem Studenten mit mindestens 1 Art vorliegt

Formulare forms/

Um die Wiederverwendbarkeit meiner Templates zu erhöhen, habe ich meine Formulareingaben für Mitarbeiter (und Admins) zum Erstellen, Bearbeiten und Löschen verschiedener Datenbankobjekte in einen separaten Ordner ausgelagert. Dadurch können die Templates von unterschiedlichen Routen aus genutzt werden, und redundanter Code wird vermieden, indem identische bzw. ähnliche Funktionen der Mitarbeiter und der Admins gekapselt werden. Diese Auslagerung erleichtert es außerdem, einfache Formatvalidierungen bereits im View zu realisieren, z. B. durch die Definition von Required-Attributen (um Null-Werte in der Datenbank zu vermeiden) und Datentypen-Checks (z. B. durch die Angabe einer maximalen Länge).

Der Template Ordner „forms“ beinhaltet folgende Formulare:

- confirm_delete: Zum Bestätigen eines Löschvorgangs, um versehentliches Löschen zu vermeiden. Verfügbar für Projekt, Mitarbeiter, Lehrstuhl und Phasen (beim Starten eines neuen Semesters).
- lehrstuhl_form: Zum Erstellen/Bearbeiten eines Lehrstuhls.
- mitarbeiter_form: Zum Erstellen/Bearbeiten eines Mitarbeiters.

- phasen_form: Zum Erstellen/Bearbeiten der Phasendaten für das aktuelle Semester.
- priorities_form: Zum Erstellen/Bearbeiten der Priorisierung der Themen eines Studenten.
- projekt_form: Zum Erstellen/Bearbeiten eines Projekts.

Bei der Implementierung dieser universellen Formular-Templates wurden mittels Jinja2 folgende Fälle unterschieden:

1. Erstellung eines neuen Datenbankobjektes vs. Bearbeitung eines bestehenden Objekts

Am Beispiel des Projektformulars wird deutlich, dass je nach Szenario unterschiedliche Inhalte angezeigt werden. Dies beginnt z.B. mit der Überschrift, die dynamisch angepasst wird:

```
<h1><b>{% if projekt %}Thema bearbeiten{% else %}Thema einstellen{% endif %}</b></h1>
```

Bei bestehenden Projekten sollen die Formularfelder außerdem vorab ausgefüllt sein, um die Änderung von Daten zu erleichtern, ohne dass der Nutzer alle Informationen erneut eingeben muss. Daher wird innerhalb der Input-Tags folgender Wert festgelegt, der die Felder entweder mit Werten aus der Datenbank oder aus dem Request-Formular befüllt:

```
value="{{ request.form.titel or projekt.titel if projekt else '' }}"
```

Zuletzt muss das Formular auch hinsichtlich der Ziel-URL flexibel sein, um beide Fälle abzudecken:

```
<form method="POST" action="{% if projekt %}{{ url_for('bp_mitarbeiter.edit_projekt', projekt_id=projekt.projekt_id) }}{% else %}{{ url_for('bp_mitarbeiter.add_projekt') }} {% endif %}" class="w3-container w3-center">
```

```
...
```

```
</form>
```

2. Unterscheidung zwischen verschiedenen Nutzerrollen (Admin vs. Mitarbeiter)

Da Mitarbeiter nur die Berechtigung haben, Projekte und Mitarbeiter ihres eigenen Lehrstuhls zu bearbeiten, wird zwischen den Rollen Mitarbeiter und Admin unterschieden. Im Projektformular werden beispielsweise sowohl die Links der Buttons angepasst als auch sichergestellt, dass Mitarbeiter nur Projekte ihres eigenen Lehrstuhls verwalten können:

Abbildung 3: Formular zur Erstellung eines neuen Projektes (Mitarbeitersicht)

Thema bearbeiten

Titel

Aktueller Überblick über die IS-Forschung hinsichtlich Social Sustainability

Beschreibung

Theoretischer Hintergrund
 Methodisches Vorgehen: Literature Review\Strukturierte Aufarbeitung der aktuellen Forschungsliteratur im Bereich Social Sustainability in der IS-Forschung; Strukturierung anhand geeigneter Parameter (Thema: Menschenrechte, Diversität im Job, etc.; Stakeholder: Mitarbeiter*in, Konsument*in; Rolle von IS/ Umsetzung in Information System)
 Diskussion über Potentiale der IS im Bereich social sustainability

Studiengang

BWL (Bachelor)

Geeignet für

BWL mit Schwerpunkt WI

Art

Bachelorarbeit

Anzahl Studenten

1

Lehrstuhl

Lehrstuhl für Wirtschaftsinformatik III - Business Engineering (Leist)

Betreuer

Susanne Leist
 Volker Berg
 Markus Lang
 Moritz Hofmann

Aktualisieren

Löschen

Zurück

Abbildung 4: Formular zur Erstellung eines neuen Projektes (Adminsicht)

[Informationen info/](#)

Datenschutz Template: Datenschutzerklärung der Universität Regensburg, einschließlich der Informationen zu Datenerhebung, -verarbeitung, Nutzungszwecken, Nutzerrechten, Aufsichtsbehörden, sowie Cookies.

Impressum Template: Impressum des Instituts für Wirtschaftsinformatik

Regeln Template: Informationen für die Studenten zum Ablauf und den Regeln des Vergabeprozesses

Themen Template: Übersicht über alle angebotenen Themen der Lehrstühle, welche nach Art gefiltert werden können

Um die Usability der Themensuche für die Studenten zu erleichtern, wurde hier ein farbliches Schema integriert (z.B. Bachelorarbeiten werden in Blau dargestellt und mit einem „B“ markiert, Projektseminare in Gelb und mit einem „P“, usw.).

Alle

Bachelorarbeit

Projektseminar

Masterarbeit

Praxisseminar

Theor. Seminar

Alle durchsuchen...

Suchen

Pernul: Lehrstuhl für Wirtschaftsinformatik I - Informationssysteme

B

Analyse und Erhebung passender Visualisierungstechniken in Splunk für eine Live Digital Forensics Investigation

geeignet für: Wirtschaftsinformatik

Betreuer: Daniel Schlette

P

Design Principles für Cyber Threat Intelligence Observables im Identity Management

geeignet für: Informatik

Maximale Teilnehmer: 4

Betreuer: Sabri Hassan , Günther Pernul , Sabrina Friedl

M

Aufbau einer modernen Web-Plattform zur Analyse der Forschung im Kontext von Visualisierungen in der IT-Sicherheit

geeignet für: Wirtschaftsinformatik

Betreuer: Sabri Hassan

Abbildung 5: Farbliche Markierung und Badges für Projekte

Um dem Nutzer nur die für ihn relevanten Informationen zu präsentieren, zeigt die Themenliste zunächst nur die wesentlichen Details zu jedem Projekt an, wie Titel, Studiengang, Betreuer sowie die mögliche Anzahl an Studenten bei Projekt-, Praxis- und Theoretischen Seminaren. Über einen **Infoknopf** beim Lehrstuhl wird der Nutzer auf deren Webseite weitergeleitet, um detaillierte Informationen zu diesem zu erhalten. Zusätzlich wurde ein JavaScript implementiert, welches mithilfe eines Pfeilsymbols eine **erweiterte Ansicht** ermöglicht, wodurch die vollständige Projektbeschreibung angezeigt wird.

Um die Benutzerfreundlichkeit für den Studenten zu erhöhen, wurde innerhalb der models.py eine **Stichwortsuche** für die Themenübersicht implementiert. Diese Suche filtert die Projekte und gibt nur diejenigen zurück, bei denen das eingegebene Keyword im Titel, Fach, Studiengang (Bezeichnung, Abschluss) oder im Namen des Betreuers (Vor- und Nachname) enthalten ist:

```
@staticmethod
def apply_keyword_search(projekte, search_query):
    if search_query:
        search_filter = f"%{search_query}%"
        projekte = projekte.filter(
            (Projekt.titel.ilike(search_filter)) |
            # (Projekt.beschreibung.ilike(search_filter)) |
            (Fach.bezeichnung.ilike(search_filter)) |
            (Studiengang.bezeichnung.ilike(search_filter)) |
            (Studiengang.abschluss.ilike(search_filter)) |
            (Projekt.betreuer.any(Mitarbeiter.vorname.ilike(search_query)))
            (Projekt.betreuer.any(Mitarbeiter.nachname.ilike(search_query)))
        )
    return projekte
```

Die Beschreibung wurde hier vorerst aus der Suche herausgenommen, um die Anzahl der Ergebnisse einzugrenzen.

Mitarbeiter mitarbeiter/

Aufgrund der Auslagerung der Formulare für die Mitarbeiterfunktionen, wie beispielsweise das Einstellen eines Themas, umfassen die Mitarbeiter-Templates folgende Dokumente:

Profil Template: Informationen zum Mitarbeiter selbst, seinem Lehrstuhl und den Projekten des Lehrstuhls

Hier wurde besonders auf die konsistente Anwendung des Farbschemas zur Markierung der Projektarten geachtet, wie es bereits in der Themenübersicht umgesetzt wurde.


	Unsere Themen +
	Aktueller Überblick über die IS-Forschung hinsichtlich Social Sustainability (Markus Lang, Volker Berg) ✎
	Entwicklung eines nachhaltigen IT-Sicherheitskonzepts für KMUs (Daniel Konadl) ✎
	Funktionserweiterung der Fakultätsapplikation „Masterbewerbung“ von Zend (zf1) in Laminas (Volker Berg) ✎
	Förderung von nachhaltigem Handeln durch Gamification (Volker Berg, Markus Lang) ✎
	Untersuchung von Internet Email-Verkehr für eine Disclosure Attack (Janik Wörner) ✎

Abbildung 6: Farbliche Markierung der Projekte nach Art

Am Ende der Seite ermöglicht ein Toggle-Switch (Umschalter) den Wechsel zum Admin-Bereich, sofern die entsprechende Berechtigung vorliegt. Die Berechtigungsprüfung erfolgt dabei im Controller.



Abbildung 7: Umschalter auf Admin Panel

Admin Template: Darstellung der Admin-Funktionen, die das Erstellen, Bearbeiten und Löschen verschiedener Daten des Seminarvergabesystems umfassen, einschließlich der Verwaltung von Mitarbeitern, Lehrstühlen, Studenten und ihrer Priorisierungen, Phasen und Projekten.

Der Administrator kann mithilfe von Reitern auf der linken Seite durch verschiedene Verwaltungsoptionen navigieren. Im Reiter „Studenten“ kann er anschließend beispielsweise über das "+"-Symbol neue Studenten anlegen oder bestehende Einträge mithilfe der Icons neben den Namen bearbeiten.

Admin Profil

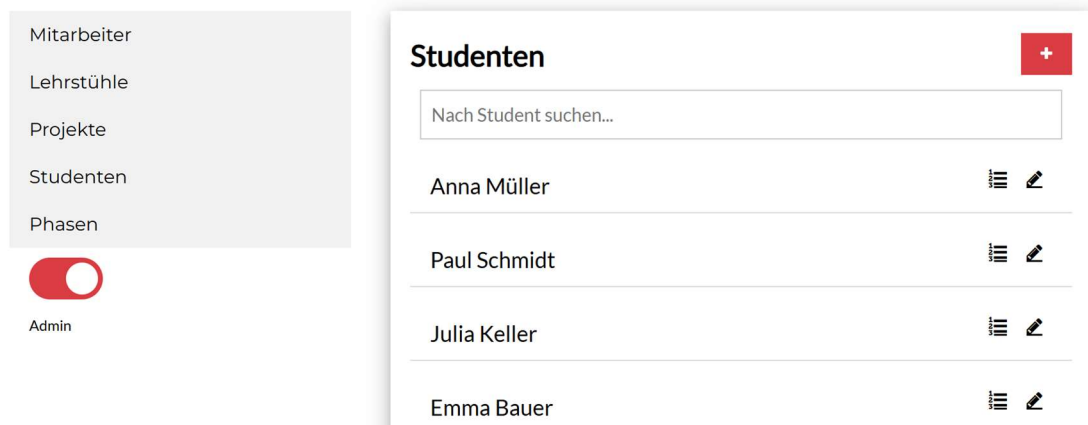


Abbildung 8: Adminprofil zur Studentenverwaltung

[Studenten student/](#)

Register Template: Formular zur Registrierung der Studenten für den Vergabeprozess, durch Abfrage zusätzliche Daten wie Matrikelnummer und Studiengang sowie der Auswahl der gewünschten Abschlussarbeiten oder Seminare

Already Registered Template: Bestätigung an den Studenten für die erfolgreiche Registrierung und die Anzeige der entsprechenden Registrierungsdaten

Profil Template: Informationen für den Studenten zu seinem Registrierungsstatus und seinen Priorisierungen

Die Studenten erhalten klare Rückmeldungen darüber, ob sie sich bereits registriert haben oder ob die Priorisierung der Themen bereits erfolgt ist. Besonders wichtig war es, dass sich auch hier die Schaltflächen für anstehende Aufgaben (Registrierung, Themenwahl) und Informationen (analog zur Startseite) dynamisch an die aktuell aktive Phase des Vergabeprozesses anpassen.

Je nachdem, ob ein Student die vorangegangenen Schritte wie die fristgerechte Registrierung erfüllt hat, werden ihm in Phase 2 die Buttons zur Themenwahl und zur Teilnahme am Vergabeprozess nur angezeigt, wenn die Voraussetzungen erfüllt sind:

Deine Daten

Name:

NDS-Kennung:

Email:

⚠ Sie sind nicht für den Vergabeprozess registriert ⚠

Deine Priorisierungen

Sie sind nicht für die Vergabe registriert und haben daher nicht die Berechtigung zur Themenwahl.

Deine Daten

Name:

NDS-Kennung:

Matrikelnummer:

Registriert für: ☒ Praxisseminar ☒ Masterarbeit

Themenwahl

Praxisseminar

Priorität 1: Extraction and Visualization of Industrial IoT Data

Priorität 2: Entwicklung eines nachhaltigen IT-Sicherheitskonzepts für KMUs

Priorität 3: Digital Honey: Konvertierung eines digitalen Zwillings in einen Honeypot

Masterarbeit

Priorisierung liegt aktuell noch nicht vor.

Abbildung 9: Studenten-Profil in der Themenwahlphase

Themenwahl Template: Formular zur Priorisierung der verfügbaren Seminare und Abschlussarbeiten, welche für die vom Studenten ausgewählten Arten möglich sind

Home Themen Regeln Profil

Themenwahl

Praxisseminar Masterarbeit

Verfügbare Projekte

Erkennung von Datenduplikaten
Heinrich: Lehrstuhl für Wirtschaftsinformatik II

Digital Honey: Konvertierung eines digitalen Zwillings in einen Honeypot
Pernul: Lehrstuhl für Wirtschaftsinformatik I - Informationssysteme

Extraction and Visualization of Industrial IoT Data
Schönig: Lehrstuhl für IoT-based Information Systems

Entwicklung eines nachhaltigen IT-Sicherheitskonzepts für KMUs
Leist: Lehrstuhl für Wirtschaftsinformatik III - Business Engineering

Prioritäten für Praxisseminar

⚠ Wichtig: Priorisierung nach Absenden nicht mehr änderbar! ⚠

Priorität 1:
Wählen Sie ein Thema

Priorität 2:
Wählen Sie ein Thema

Priorität 3:
Wählen Sie ein Thema

Speichern

Abbildung 10: Themenwahl in Phase 2

Zusätzlich können die Beschreibungen der jeweiligen Projekte durch einen Klick auf den Pfeil angezeigt werden, um weitere Informationen zu erhalten, falls vorher nicht ausreichend informiert. Das Formular zur Themenwahl ist als Sticky-Formular konzipiert, das beim Scrollen auf der rechten Seite sichtbar bleibt. Dies erleichtert es den Studenten bei einer langen Liste an verfügbaren Projekten, ihre Prioritäten einzustellen und abzusenden, ohne die Übersicht zu verlieren.

Styles

Für ein konsistentes und ansprechendes Layout nutzen die HTML-Templates das externe Stylesheet w3.css. Dieses moderne, responsive CSS-Framework bietet eine Vielzahl vordefinierter Stilklassen, die unter anderem Farben, Abstände, Container und Tabellen beinhalten. Die vordefinierten Klassen von w3.css können mittels class="w3.css-Element" direkt zugewiesen werden, wodurch keine neuen Styles innerhalb der HTML Templates definiert werden müssen.⁸

```
<h3 class="w3-left-align w3-text-red ">  
... Dieser Text ist linksbündig und rot  
</h3>
```

Um spezielle Stilelemente innerhalb der Webanwendung zu ergänzen, wurden spezielle CSS-Klassen zur Darstellung der Toggle Switch (Umschalter Admin/ Mitarbeiter), des Prozesses auf der Startseite (Pfeile) und zur Nutzung eines Sticky Forms (an einer Stelle positioniertes Formular in der Themenwahl) ergänzt.

3.2.3 Controller

Die wichtigste Rolle im MVC-Framework stellt wohl der Controller dar, da er für die gesamte Anwendungslogik verantwortlich ist. Diese Komponente umfasst also folgende drei Teile: Initialisierung der Applikation, Definition der URL-Routen und Ausführung der Views.⁹

Initialisierung und Konfiguration der Anwendung

Die Datei __init__.py initialisiert die Webanwendung und erstellt die benötigte Flask-Instanz.

```
app = Flask(__name__)
```

Hier erfolgt außerdem die Datenbankkonfiguration, welche bereits im Kapitel „Model“ erläutert wurde, sowie die Registrierung der vier Blueprints:

```
app.register_blueprint(bp_index, url_prefix="")  
app.register_blueprint(bp_auth, url_prefix="/auth")  
app.register_blueprint(bp_mitarbeiter, url_prefix="/mitarbeiter")  
app.register_blueprint(bp_student, url_prefix="/student")
```

Definition der URL-Routen mittels Blueprints

Blueprints gruppieren ähnliche Funktionen und Ansichten, um eine Anwendung zu organisieren. In dieser Anwendung werden Blueprints verwendet, um verschiedene Module zu trennen, wie zum Beispiel Studentenfunktionen und Mitarbeiterfunktionen, die jeweils unter einem spezifischen URL-Präfix registriert sind.¹⁰

⁸ [W3.CSS Home \(w3schools.com\)](https://www.w3schools.com/)

⁹ [Building a Flask Web Application \(Flask Part 2\) - Siv Scripts \(alysivji.github.io\)](https://github.com/sivscript/flask-part-2)

¹⁰ [Blueprints and Views — Flask Documentation \(2.3.x\) \(palletsprojects.com\)](https://palletsprojects.com/en/2.3.x/blueprints/)

Innerhalb des Seminarvergabesystems sind 4 Blueprints definiert, bei denen der Code jeweils in separaten Python-Modulen abgelegt ist mit ihren entsprechenden Routen implementiert.

Bp_index → index.py	Bp_mitarbeiter → mitarbeiter.py
Startseite / -- /regeln -- /datenschutz -- /impressum -- /themen -- /themen/<art_filter> -- /restricted -- /no_authorisation	/mitarbeiter -- /admin -- /add_projekt -- /edit/<projekt_id> -- /delete/<projekt_id> -- /add_mitarbeiter -- /edit_mitarbeiter/<ma_id> -- /delete/mitarbeiter/<ma_id> -- /add_lehrstuhl -- /edit_lehrstuhl/<lehrstuhl_id> -- /delete/lehrstuhl/<lehrstuhl_id> -- /add_student -- /edit_student/<stud_id> -- /edit_priorities/<stud_id> -- /delete_student/<stud_id> -- /add_phasen -- /edit_phasen -- /delete_phasen
Bp_student → student.py	Bp_auth → auth.py
/student -- /register -- /themenwahl	/auth/login /auth/logout

Bp_index

Dieser Blueprint bildet den Einstiegspunkt und das zentrale Modul der Applikation. Er umfasst die Startseite sowie verschiedene Informationsseiten, welche für alle Benutzer zugänglich und daher keine speziellen Zugriffskontrollen erfordern.

Die Routen und ihre Logik für die **Regeln** des Vergabeprozesses, des **Impressums**, den **Datenschutz** sowie **Fehlermeldungen** (restricted, no_authorisation) besteht nur aus dem Rendern der entsprechenden Templates, welche die entsprechenden Informationen enthalten.

Die **Themenübersicht** kann sowohl von Anfragen mit als auch ohne einen spezifischen Filter für die Art der Themen genutzt werden. So wird beispielsweise beim Klicken auf einen Art-Button auf der Startseite (Bachelorarbeit, Projektseminar, Masterarbeit, Praxisseminar oder theoretisches Seminar) oder durch Filterung über die Themenübersicht selbst die jeweilige Art über die URL an die Route übergeben:

```
@bp_index.route('/themen', defaults={'art_filter': None}, methods=['GET'])
@bp_index.route('/<art_filter>', methods=['GET'])
def themen(art_filter):
    ...
```

Die entsprechenden Daten zu den Themen werden anschließend mittels der in models.py definierten Methoden aus der Datenbank geladen und anhand der Filter (falls vorhanden) für Art und einem Suchkriterium gefiltert und abgerufen, um anschließend zum Rendern des Themen-Templates genutzt zu werden.

Die **Startseite** zeigt hauptsächlich statische Daten an, weshalb kaum Daten aus der Datenbank abgerufen werden müssen. Lediglich zur Darstellung des interaktiven Prozesses werden die Phasen-Zeiten des aktuellen Semesters abgerufen, um die aktuell aktive Phase hervorzuheben.

Bp_mitarbeiter

Ursprünglich waren zwei verschiedene Blueprints für Mitarbeiter und Administratoren vorgesehen. Aufgrund der vielen ähnlichen und sich überschneidenden Funktionen führte dies jedoch zu zahlreichen Code-Duplikaten. Daher wurde eine zentrale Datei für alle Funktionen geschaffen, indem die entsprechenden Funktionen zusammengeführt wurden. Dabei werden folgende drei Fälle unterschieden:

a) Gemeinsame, identische Routen

Da Administratoren ebenfalls Mitarbeiter eines Lehrstuhls sind, gibt es einige Funktionen, die für beide Rollen identisch sind, wie zum Beispiel das Mitarbeiterprofil und die daraus gerenderten Funktionen (z.B. Thema für seinen Lehrstuhl einstellen).

Diese Routen sind mit dem Decorator **@mitarbeiter_required** versehen, welcher in der helper.py definiert ist. Dieser Decorator stellt sicher, dass die Zugriffsrechte korrekt überprüft werden, um zu verhindern, dass beispielsweise Studenten über die Hauptseite durch manuelle Eingabe von /mitarbeiter in der URL auf Mitarbeiterfunktionen zugreifen können. Innerhalb des Decorator wird überprüft, ob ein Eintrag mit der NDS-Kennung des Nutzers in der entsprechenden Datenbankklasse Mitarbeiter vorhanden ist und der Nutzer somit die entsprechende Berechtigung besitzt.¹¹

```
def mitarbeiter_required(func):
    @wraps(func)
    def decorated_view(*args, **kwargs):
        if not current_user.is_mitarbeiter:
            return redirect(url_for('bp_index.no_authorisation'))
        return func(*args, **kwargs)
    return decorated_view
```

b) Admin-Spezifische Funktionen

Neben der Verwaltung der Themen und Mitarbeiter ihres eigenen Lehrstuhls, können Mitarbeiter keine weiteren Daten der Datenbank erstellen, bearbeiten oder löschen. Um die weiteren Funktionen wie Lehrstuhlverwaltung, Phasenverwaltung und Studentenverwaltung abzubilden, wurde ein Admin-Panel eingerichtet, welcher alle Admin-Funktionen bündelt.

Diese Funktionen sind mit dem Decorator **@superuser_required** versehen, der analog zu @mitarbeiter_required prüft, ob ein Eintrag in der superuser-Datenbankklasse vorhanden ist, was darauf hinweist, dass der Nutzer Admin-Berechtigungen besitzt.

c) Gemeinsame Routen mit unterschiedlichen Berechtigungen

Einige Funktionen, wie die Themen- und Mitarbeiterverwaltung, müssen je nach Kontext (Mitarbeiter- oder Admin-Bereich) unterschiedlich gehandhabt werden. Mitarbeiter können zwar Themen und Mitarbeiter verwalten, jedoch nur die ihres eigenen Lehrstuhls.

Diese gemeinsamen Routen sind ebenfalls mit dem Decorator @mitarbeiter_required versehen, es wurde allerdings zur Unterscheidung der Inhalte im Template eine **Session-Variable is_admin**

¹¹ [Die wichtigsten Sicherheitslücken in Web-Anwendungen: OWASP Top Ten | turingpoint](#)

eingeführt. Diese Session-Variable ermöglicht es, die Informationen im Hintergrund zu verwalten und über mehrere Anfragen hinweg zu speichern, ohne dass der Nutzer davon etwas bemerkt.¹²

Wenn ein Mitarbeiter beispielsweise aus seinem Profil heraus ein Thema hinzufügen möchte, werden die Formularbestandteile des Templates basierend auf der Variable `is_admin` gefiltert und auch die Routen der Buttons entsprechend angepasst. In der Route des Mitarbeiterprofils wird die Admin-Sitzungsvariable deshalb auf `False` gesetzt:

```
@bp_mitarbeiter.route("/", methods=['GET'])
@login_required
@mitarbeiter_required
def profil():
    ...
    session['is_admin'] = False
    return render_template("mitarbeiter/profil.html", ...)
```

In den gemeinsamen Routen wird der Wert der Sitzungsvariable `is_admin` in eine Python-Variable gespeichert, welche an das Template zur Nutzung von Jinja2-Abfragen übergeben wird und auch innerhalb der Route selbst genutzt wird, um unterschiedliche Logiken zu steuern:

```
@bp_mitarbeiter.route('/add_projekt', methods=['GET', 'POST'])
@login_required
@mitarbeiter_required
def add_projekt():
    ...
    is_admin = session.get('is_admin', False)

    if request.method == 'POST':
        ...
        lehrstuhl_id = request.form.get('lehrstuhl') if is_admin else
user.lehrstuhl_id

        return redirect(url_for('bp_mitarbeiter.admin' if is_admin else
'bp_mitarbeiter.profil'))

    return render_template('forms/projekt_form.html',
        ...
        is_admin=is_admin)
```

Beim Umschalten auf den Adminbereich wird der Session-Wert innerhalb der Admin-Route auf `True` gesetzt, wodurch alle Routen als Admin nutzbar sind und erst beim Zurückschalten auf das Mitarbeiterprofil durch die Profil-Route wieder auf `False` zurückgesetzt.

Bp_student

Die Hauptroute dieses Blueprints rendert das **Profil** eines Studenten. Dabei werden die persönlichen Daten und Informationen zur Registrierung und Priorisierung des Studenten abgerufen, sofern diese bereits vorhanden sind. Um die Interaktivität der Funktionen zu gewährleisten, werden auch die Phasendaten übergeben, sodass die Anzeige der zusätzlichen Informationen im Profil je nach aktueller Phase variiert (analog zum Vergabeprozess der Startseite).

¹² [Session data in Python Flask - Python Tutorial \(pythonbasics.org\)](https://pythonbasics.org/session-data-in-python-flask/)

Bei der **Registrierung** wird überprüft, ob bereits ein Eintrag in der Tabelle "Student" vorhanden ist, was bedeutet, dass der Student sich bereits registriert hat. In diesem Fall werden dem Studenten lediglich seine Daten angezeigt, da diese nach der Eingabe nicht mehr änderbar sind. Sollte noch keine Registrierung erfolgt sein, werden die persönlichen Daten des Studenten, die entweder aus dem „current_user“ oder den Formulareingaben stammen, gespeichert (Eintrag in der Klasse "Student" erstellt) und die Arten der Arbeiten, die der Student dieses Semester belegen möchte, hinzugefügt.

Die **Themenwahl**, welche nur verfügbar ist, wenn ein Student sich zuvor für mind. eine Art der Studienarbeit registriert hat,

Zur Sicherstellung der zeitlichen und rollenbasierten Beschränkung der Routen wurden auch hier Dekoratoren implementiert. Der **@login_required-Dekorator** sorgt dafür, dass nur authentifizierte Studenten der Universität Regensburg die Funktionen nutzen können. Zusätzlich verhindern die Dekoratoren **@within_registration_period** und **@within_prioritization_period**, dass Studenten die entsprechenden Routen außerhalb der festgelegten Fristen nutzen können, selbst wenn sie diese manuell in den Browser eingeben und werden stattdessen auf eine andere Seite weitergeleitet:

```
def within_registration_period(f):
    @wraps(f)
    def decorated_function(*args, **kwargs):
        current_phase = Phase.get_current()
        if not current_phase or not (current_phase.start_pl <=
datetime.now() <= current_phase.ende_pl):
            return redirect(url_for('bp_index.restricted'))
        return f(*args, **kwargs)

    return decorated_function
```

Bp-Auth

Neben den **Login- und Logout-Routen** wurde hier auch die komplette Authentifizierungslogik für das Seminarvergabesystem implementiert. Bei der Umsetzung dieser Authentifizierungsmethode habe ich mich an einem Referenzprojekt der Universität Regensburg orientiert, das ebenfalls die **Authentifizierung mittels python-ldap und LoginManager** realisiert.

LDAP ist ein Zugriffsprotokoll für schnelle Abfragen, Suchen, Änderungen und Autorisierungen in verteilten Verzeichnisdiensten. Die Universität Regensburg verwendet einen LDAP-Server, um verschiedene Informationen zu speichern, darunter Daten von Studenten und Mitarbeitern. Dieser LDAP-Server ermöglicht nun die Authentifizierung der Benutzer anhand ihrer NDS-Kennung und ihres Passworts. Darüber hinaus können weitere Benutzerdaten zu den Studenten und Mitarbeitern wie Vorname, Nachname und E-Mail-Adresse aus dem LDAP-Verzeichnis abgerufen werden.¹³

Zuerst wurde für die Sitzungsverwaltung und das Login-Handling der Flask-Login (LoginManager) integriert und die ein User-Loader implementiert, welche Benutzerdaten basierend auf der NDS-Kennung lädt und sie mit zusätzlichen Attributen (Vorname, Nachname, E-Mail) aus dem LDAP-Verzeichnis erweitert:

```
loginManager = LoginManager()
loginManager.login_view = "bp_auth.login"
```

```
@loginManager.user_loader
def load_user(nds):
```

¹³ [Was ist LDAP? Definition und Funktionsweise erklärt - IONOS](#)

```
user_data = extendNewUser(User(nds, "", "", ""))
return User(nds, user_data.vorname, user_data.nachname, user_data.mail)
```

Die Verbindung zum LDAP-Server der Universität Regensburg wird über das Python-Modul python-ldap hergestellt. Es wurden drei Methoden zur Verarbeitung der LDAP-Authentifizierung implementiert, welche eine SSL/TLS-Verbindung zum LDAP-Server „ldaps://ldapclient.uni-regensburg.de:636“ nutzen:

- getDN(nds): Gibt den Distinguished Name (eindeutiger Bezeichner im LDAP-Verzeichnis) eines Nutzers anhand seiner NDS-Kennung zurück
- checkPassword(dn, password): Überprüfung des Passworts anhand des LDAP-Verzeichnisses
- checkPasswordOfNds(nds): Kombination der beiden Methoden getDN(nds) und checkPassword(dn, password) zum Login des Users

Zuletzt wurde außerdem eine Methode zum Erweitern der Benutzerdaten implementiert, welche den „User“ anhand der LDAP-Einträge um Vorname, Nachname und Mail ergänzt, die beispielsweise über „current_user.mail“ abgerufen werden können.

4 Projektabschluss

4.1 User Tests

Um die Bedienbarkeit und Verständlichkeit der Webapplikation zu evaluieren, wurde eine Benutzerbefragung unter Studenten durchgeführt. Diese Studenten führten den Test direkt an meinem Rechner durch und navigierten live durch das System.

Folgende Punkte haben sich dabei herauskristallisiert:



Intuitiver To-Do-Prozess

Der Prozess auf der Startseite bietet eine einfache Navigation durch anklickbare To-Do-Buttons wie „Themenwahl“, wodurch Nutzer direkt zu relevanten Aufgaben gelangen, ohne das gesamte System durchsuchen zu müssen.

Benutzerfreundliches Interface

Projekte sind farblich markiert, was die Übersichtlichkeit und das Auffinden von Informationen verbessert.



Unverständlichkeit der Drag-and-Drop-Funktion

Die initial implementierte Drag-and-Drop-Funktion für die Themenwahl verwirrte die Studenten mehr, als dass sie nützlich war. Diese wurde daher entfernt.

4.2 Next Steps

Da der Projektumfang im Vorfeld auf die Grundstruktur sowie die Phase 1 und Phase 2 des Vergabeprozesses begrenzt wurde, wurden vier Schritte zur finalen Fertigstellung der Webapplikation identifiziert:

1. Umsetzung und Nutzung der Gruppen

Die Gruppe wurde datenbankseitig bereits implementiert, jedoch standardmäßig auf NULL gesetzt, da sie erst beim Themenvergabealgorithmus relevant wird. Daher muss innerhalb der Themenwahl für bestimmte Arten eine mögliche Gruppenwahl implementiert werden. Wenn ein Student einer Gruppe beitrifft, wird sein Status auf "False" gesetzt und der Student selbst für die Teilnahme gesperrt. Die Gruppe nimmt dann am Vergabeprozess teil und der Student wird aus dem individuellen Vergabeprozess entfernt.

2. Themenvergabealgorithmus

Im Phasenreiter des Adminbereichs kann der Administrator die Themenvergabe starten. Hierfür muss ein Algorithmus zur Themenvergabe entwickelt werden, der Priorisierungen, Gruppenbildung und Zufallsfaktoren berücksichtigt und kombiniert. Nach Abschluss der Themenvergabe sollen Betreuer und Studenten per Mail benachrichtigt werden.

3. Reporting im Admin Panel (optional)

Es kann ein Reporting-Tool zu den Statistiken des Vergabeprozesses implementiert werden. Dies umfasst beispielsweise die Anzeige der Anzahl der Studenten, die sich in Phase 1 für jede Art registriert haben, die Anzahl der priorisierten Themen in Phase 2 und die noch verfügbaren Plätze in Phase 3.

4. Deployment auf dem Server

Eine gängige Methode zum Deployment einer Flask-Anwendung auf einem Server ist mittels einem Nginx-Proxyserver und einem Gunicorn-Webserver. Das Virtual Environment (Requirements) ist bereits implementiert. Der nächste Schritt würde daher die Konfiguration von Gunicorn (WSGI-Webserver, der die Flask-Anwendung ausführt) und die Erstellung einer wsgi.py-Datei als Entry-Point umfassen. Diese Datei definiert die Standardschnittstelle für die Kommunikation zwischen der Anwendung und dem Webserver. Zuletzt erfolgt die Konfiguration von Nginx, welcher als Proxyserver zwischen dem Gunicorn-Server und dem Client fungiert und Anfragen entsprechend weiterleitet.¹⁴

¹⁴ [How To Serve Flask Applications with Gunicorn and Nginx on Ubuntu 18.04 | DigitalOcean](#)

5 Quellen

Blueprints and Views — Flask Documentation (2.3.x) (2023): [online]

<https://flask.palletsprojects.com/en/2.3.x/tutorial/views/>

Building a Flask Web Application (Flask Part 2) - Siv Scripts (2019): [online]

<https://alysivji.github.io/flask-part2-building-a-flask-web-application.html>

Declaring Models — Flask-SQLAlchemy Documentation (2.x) (2023): [online] <https://flask-sqlalchemy.palletsprojects.com/en/2.x/models/>

Die wichtigsten Sicherheitslücken in Web-Anwendungen: OWASP Top Ten | turingpoint (2021):

[online] <https://turingpoint.de/blog/die-wichtigsten-sicherheitsluecken-in-web-anwendungen-owasp-top-ten/>

How To Serve Flask Applications with Gunicorn and Nginx on Ubuntu 18.04 | DigitalOcean (2019):

[online] <https://www.digitalocean.com/community/tutorials/how-to-serve-flask-applications-with-gunicorn-and-nginx-on-ubuntu-18-04>

MVC Framework Introduction - GeeksforGeeks (2020): [online] <https://www.geeksforgeeks.org/mvc-framework-introduction/>

MySQL – ON DELETE CASCADE-Einschränkung – GeeksforGeeks (2019): [online]

<https://www.geeksforgeeks.org/mysql-on-delete-cascade-constraint/>

SQLAlchemy: Die mächtige ORM-Bibliothek für Python (gpt5.blog) (2023): [online]

<https://gpt5.blog/sqlalchemy/>

Session data in Python Flask - Python Tutorial (pythonbasics.org) (2020): [online]

<https://pythonbasics.org/flask-sessions/>

Setting Up Your Python Environment With Venv and requirements.txt | Frank Corso (2020)[online]:

<https://frankcorso.dev/setting-up-python-environment-venv-requirements.html>

Template Designer Documentation — Jinja Documentation (2.10.x) (2023): [online]

<https://jinja.palletsprojects.com/en/2.10.x/templates/#template-objects>

W3.CSS Home (w3schools.com) (2023): [online] <https://www.w3schools.com/w3css/default.asp>

Was ist LDAP? Definition und Funktionsweise erklärt - IONOS (2022): [online]

<https://www.ionos.de/digitalguide/server/knowhow/ldap/#c427199>

What is MoSCoW Prioritization? | Overview of the MoSCoW Method (productplan.com) (2023):

[online] <https://www.productplan.com/glossary/moscow-prioritization/>