

Entwurfsdokumentation (SAd)

Dieses Dokument beschreibt wesentliche Elemente der Softwarearchitektur, sowie andere übergreifende Aspekte des Systems für die Mitgliederdatenbank des StuRa. Hier werden im Folgenden auf die Ziele, Annahmen, die Architektonische Bedeutung, unsere Entscheidungen bzw. Einschränkungen und weitere Dinge eingegangen und Dokumentiert.

Mit Hilfe von verschiedenen Modellen und Entwürfen für die Architektur, soll die spätere Weiterentwicklung und Anpassung einfacher gemacht werden.

Architektonische Ziele und Philosophie

Das System ist eine bestehende Webanwendung zur Verwaltung der Kandidaten und Mitglieder des Stura der HTW Dresden durch einen Admin. Offizielle Mitglieder erhalten über einen Login Zugang zur Anwendung und können verschiedene Informationen einsehen, aus diesem Grund muss eine parallele Nutzung von mindestens 10 Personen gewährleistet werden. Des Weiteren ist nicht bekannt auf welchem Endgerät die Nutzung der Anwendung erfolgen wird, weshalb auf die Kompatibilität des Inhaltes mit deren Ansicht auf diversen Bildschirmgrößen geachtet werden muss.

Eine gute Bedienbarkeit wird durch eine übersichtliche und intuitive Benutzeroberfläche erzielt, welche zur Akzeptanz des Gesamtsystems durch die Mitglieder und des Admin beiträgt.

Die Anwendung ist eine weiterentwickelte Version des Projektes der Gruppe, von der wir das System übernommen haben. Andere Gruppen, aus zukünftigen Semestern, werden vermutlich ebenfalls an der Optimierung der sogenannten Mitgliederdatenbank arbeiten, sodass Aspekte, die unsere Gruppe im Rahmen des Projektes nicht umsetzen könnten, trotzdem umgesetzt werden können. Dazu zählen beispielsweise die Integrierung des Workloads der Mitglieder oder die Erstellung eines Organigramms aus den Mitgliedern der Datenbank.

Annahmen und Abhängigkeiten

Annahmen

- jeder Nutzer hat eine stabile Internetverbindung und nutzt einen aktuellen Browser (Firefox oder Chrome)
- Der Server auf dem die Webseite laufen soll, bietet ausreichen Ressourcen
 - Arbeitsspeicher: Verbund aus 3 Servern mit jeweils 72 GB RAM
 - Massenspeicher: ausreichend groß
 - Betriebssystem: Linux
- die bisher verwendete Datenbankmodelle und Frameworks können weiter verwendet werden
- die Mitgliederdatenbank wird in Zukunft funktional erweitert

Abhängigkeiten

- wir sind vom Laborbereich abhängig, der den Server mit seinen Ressourcen stellt

Entscheidungen, Einschränkungen und Begründungen

1. Wir nutzen Python als Programmiersprache, da die bestehenden Teile der Anwendung in dieser Sprache programmiert wurden und wir einen Mehraufwand im Sinne einer Umstrukturierung vermeiden wollten.
Außerdem ist sie objektorientiert und besitzt eine verhältnismäßig leichte Syntax.
2. Ebenso wie unsere Vorgänger nutzen wir Django als Framework. Es ist ebenfalls in Python verfasst und folgt einem Model-View-Presenter-Schema, aus dem eine leichte Erweiterbarkeit resultiert.
3. SQLite wird als Programmierbibliothek aufgegriffen, da es standardmäßig von Django unterstützt wird und von der Syntax sehr dem SQL-Standard ähnelt, mit dem einige Gruppenmitglieder schon intensiveren Kontakt hatten.
Abgesehen davon ist SQLite sowohl mit verschiedenen Betriebssystemen von Smartphones, sowie mit unterschiedlichsten Browsern kompatibel und unterstützt so das Erreichen von mehr Nutzern, die sich durch ihre verschiedenen Voraussetzungen auszeichnen.
4. Im Gegensatz zur Vorgängergruppe entscheiden wir uns dagegen das Framework Selenium im Zusammenhang mit den Tests zu verwenden, da hierbei der Einarbeitungsaufwand unverhältnismäßig groß wäre, da kein Mitglied unseres Teams damit bis jetzt in Kontakt gekommen ist. Die Tests werden stattdessen manuell erstellt.

Architektur-relevante Anforderungen

| Anforderung | Systemkomponente | Architekturmechanismen |
|-------------|---------------------------|------------------------|
| UC03 | Webanwendung | Erlernbarkeit |
| F1 | Webanwendung | Zugriffsschutz |
| F2 | Datenbank | Persistenz |
| U1, U2 | Interface | Erlernbarkeit |
| R1 | Webanwendung Datenbank | Archivierung |
| S1 | Webanwendung | Dokumentation |

Architekturmechanismen

Archivierung

- Zustand: Analysis

- Zweck: Daten dürfen bei Systemausfällen nicht verloren gehen
- Eigenschaften/ Attribute: Pro Woche soll ein Backup erstellt werden. Es sollen nie mehr als 3 Backups vorhanden sein
- Funktion: Backup der Datenbank wird auf dem Server angelegt

Dokumentation

- Zustand: Implementation
- Zweck: Die Anwendung soll erweiterbar sein/bleiben
- Eigenschaften/ Attribute: verständlich
- Funktion: Framework Django wird verwendet. Dokumentationen werden angefertigt

Erlernbarkeit

- Zustand: Implementation
- Zweck: Eine intuitive Nutzung soll erzielt werden
- Eigenschaften/ Attribute: einfach unkompliziert
- Funktion: Nutzerfreundliches Interface wird beibehalten

Persistenz

- Zustand: Implementation
- Zweck: Daten müssen zur Verfügung stehen für den Admin und die Mitglieder des Stura
- Eigenschaften/ Attribute: SQLite-Datenbank
- Funktion: Alle zu speichernden Daten müssen Platz in der Datenbank finden

Zugriffsschutz

- Zustand: Implementation (Vorgängersystem)
- Zweck: Login
- Eigenschaften/ Attribute: Nicht jeder Nutzer ist dazu berechtigt alle Daten einzusehen
- Funktion: Unterscheidung beim Login nach Mitglied Stura und Admin

Datenspeicherung

Die Speicherung der Daten erfolgt über die SQLite-Datenbank. Jeder Datensatz ist identifizierbar über einen Primärschlüssel in Form einer ID, die jedoch nicht durch die Webanwendung dem Admin oder den Mitgliedern präsentiert wird.

Webschnittstelle

Die von Django bereitgestellte Schnittstelle wird verwendet um Daten über die Webseite zu organisieren und zu verwalten. Die Weboberfläche wurde mittels HTML/CSS erstellt.

Informationsvermittlung

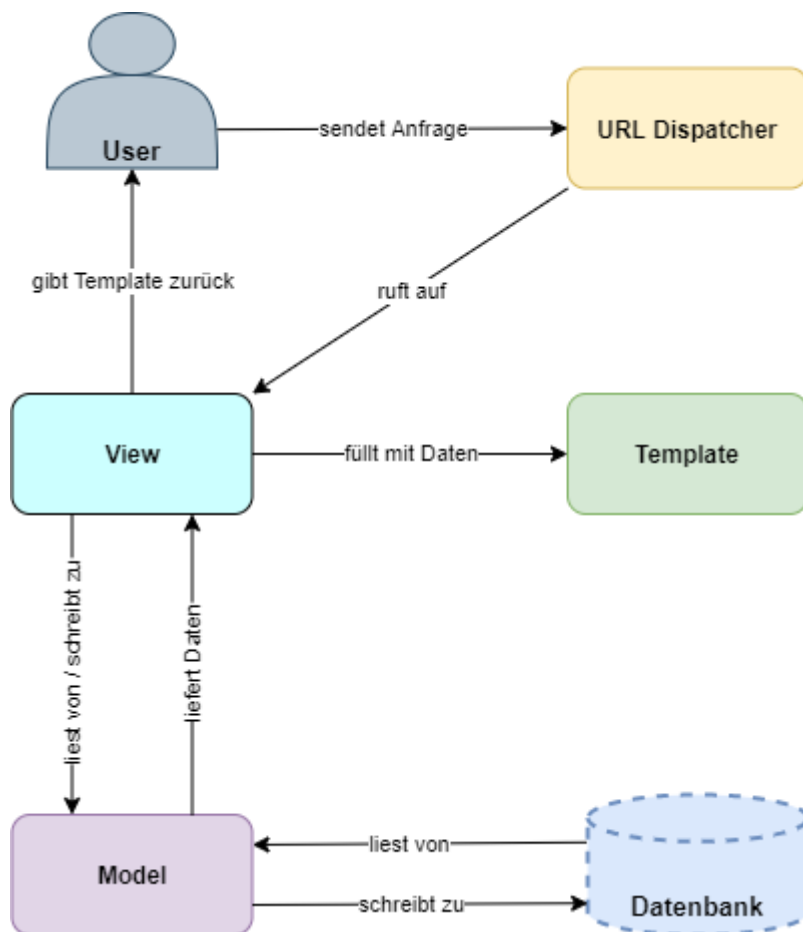
Bereitstellung der Daten für Organisations- und Informationszwecke auf der Webseite

Systemarchitektur (BMü)

Architekturmuster

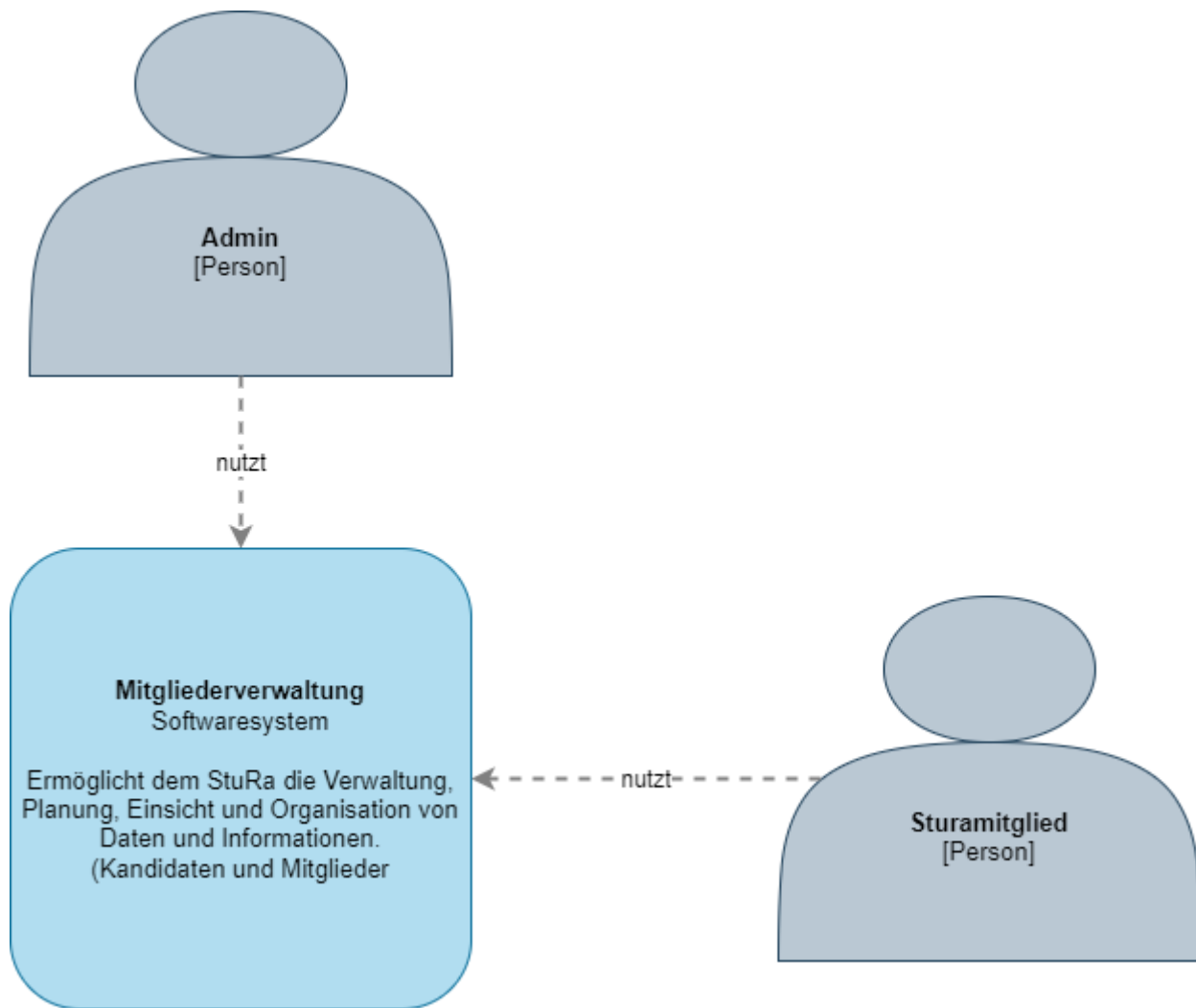
Model-View-Presenter (MVP)

- ergibt sich aus Django Framework
- **Model:** betreibt die Ansicht und ist für die Logik der Ansicht zuständig
- **View** (Ansicht): für Ein- und Ausgaben verantwortlich
- **Presenter:** Verbindung zwischen Model und View

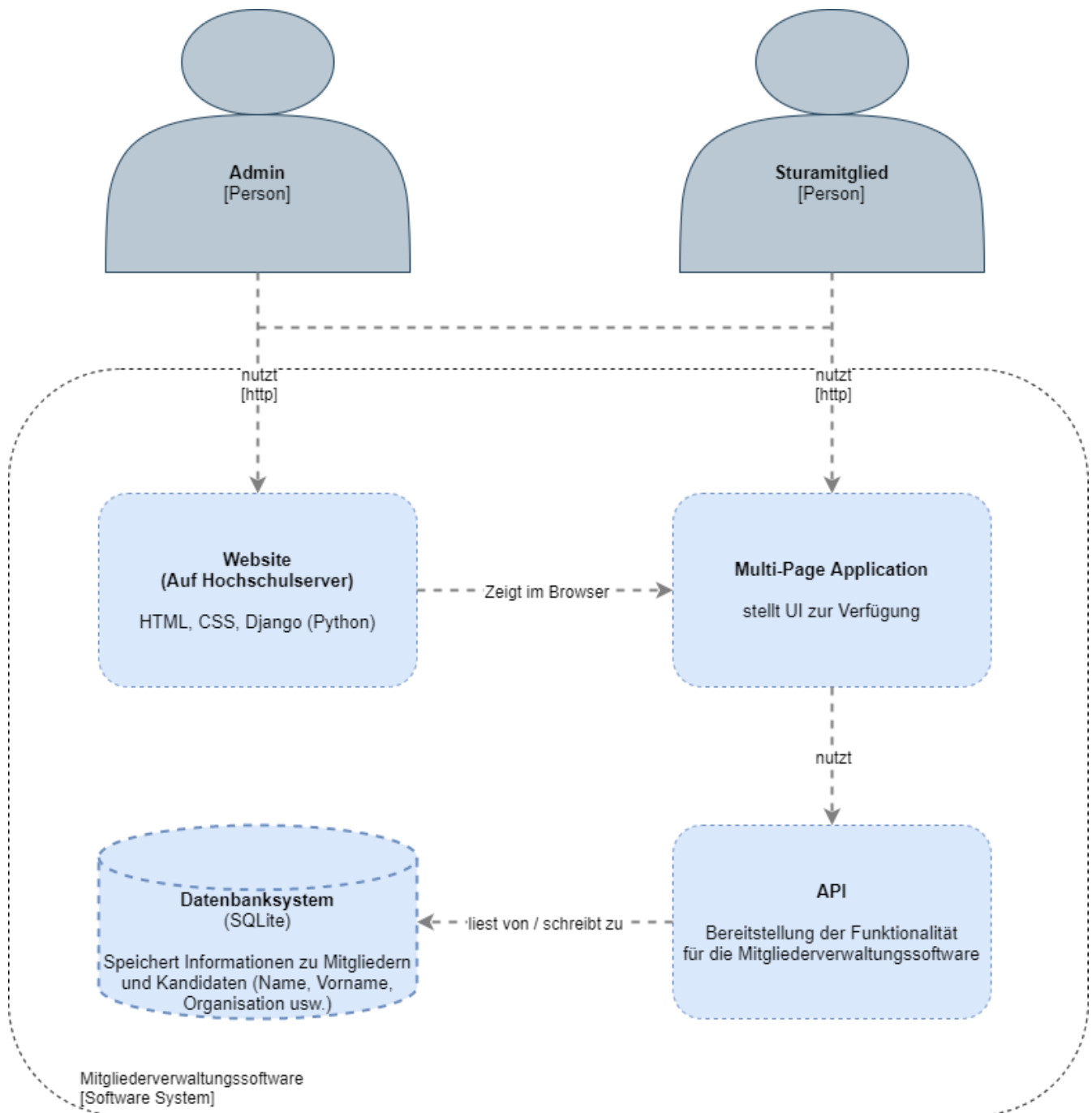


Logische Sicht

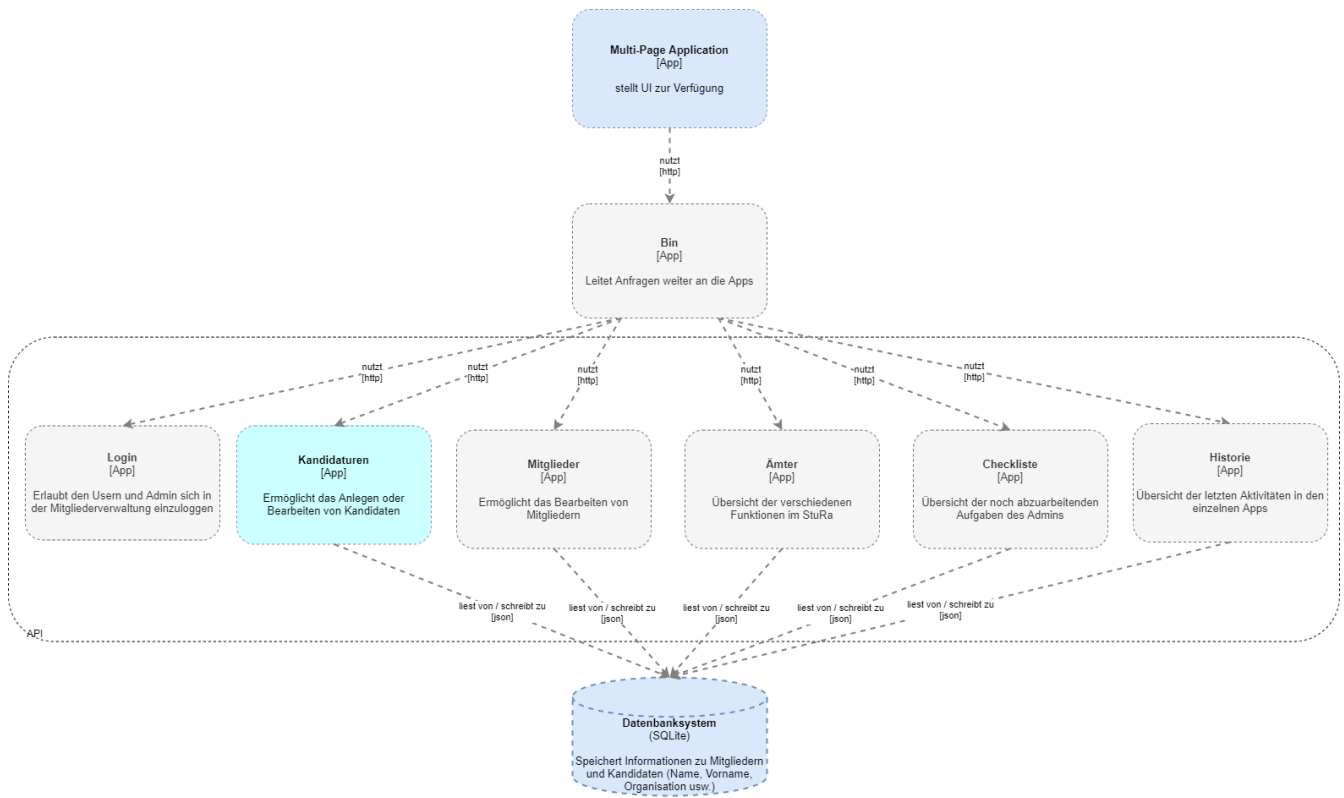
C4 Modelle



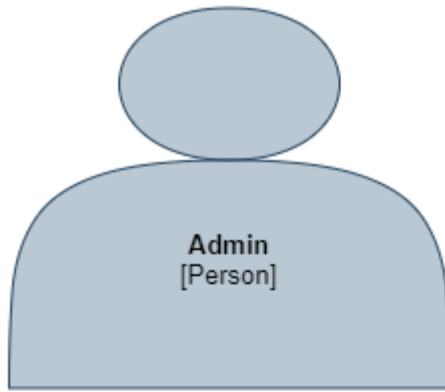
Level 1



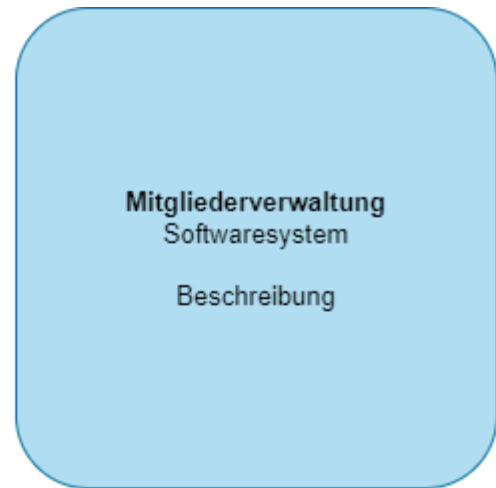
Level 2



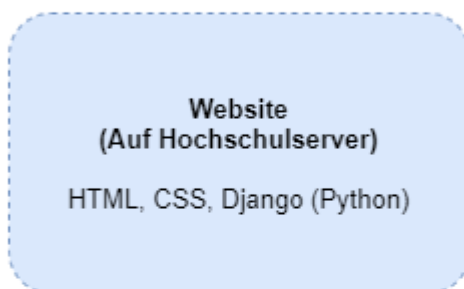
Level 3



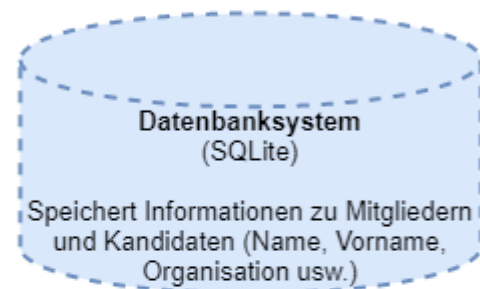
Nutzer des Softwaresystems



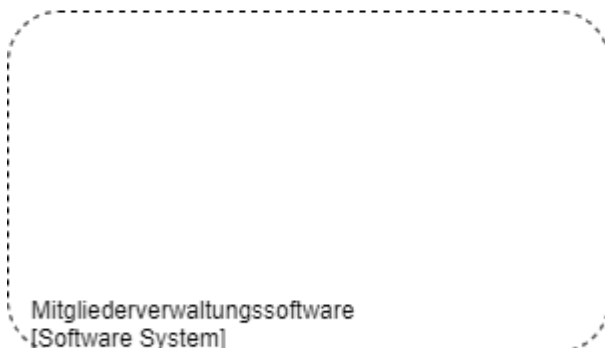
Das Softwaresystem



Komponente



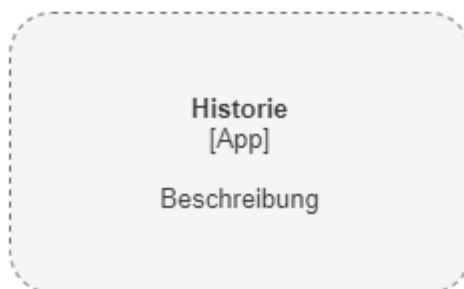
Datenbank



Zoom in Einzelteile



neu implementierte Django App

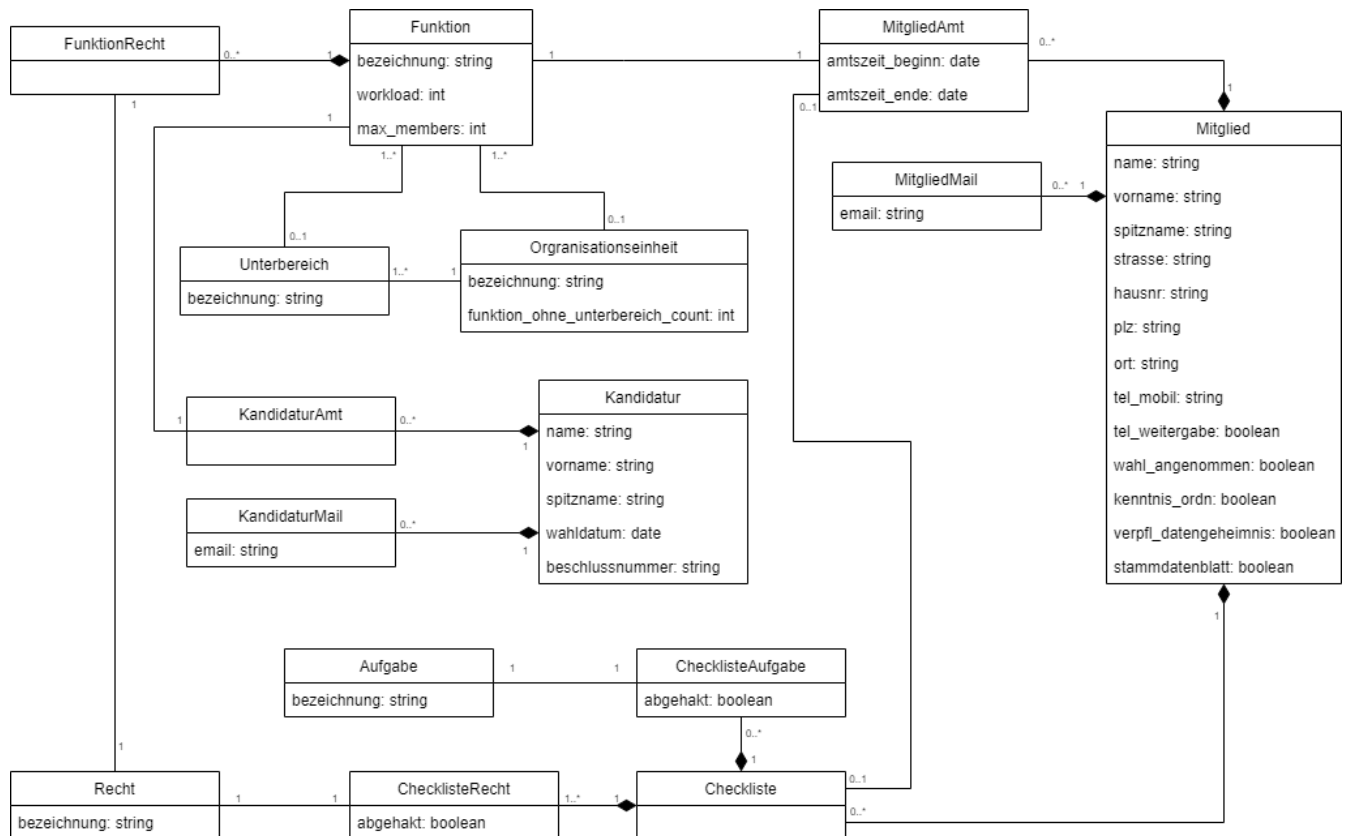


schon vorhandene Django App

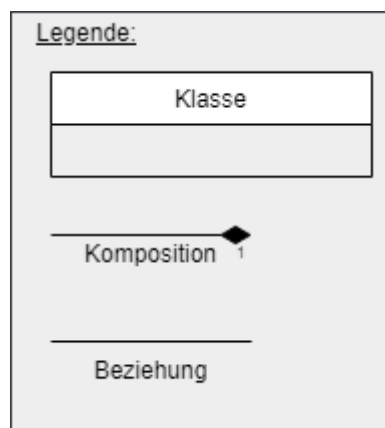


Beziehung

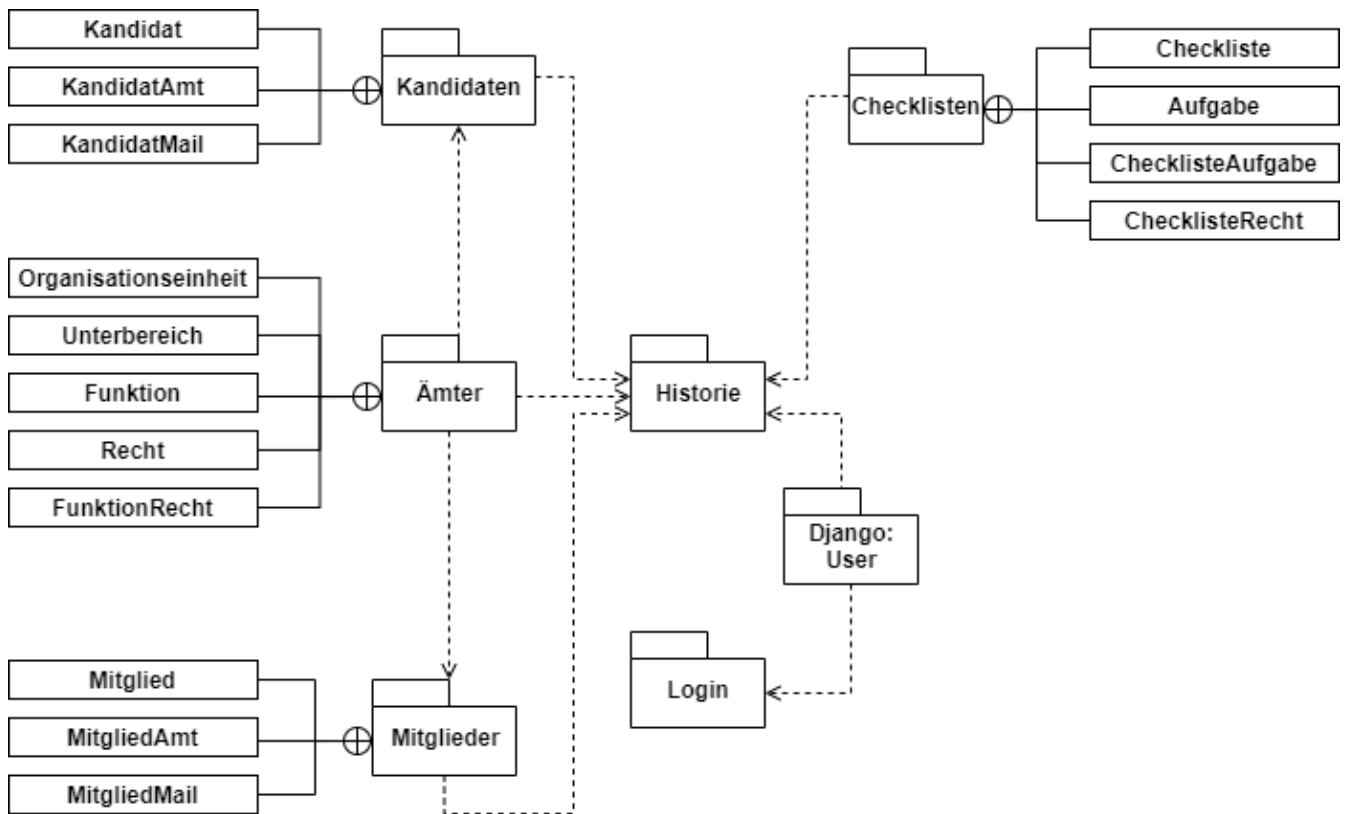
Klassendiagramm



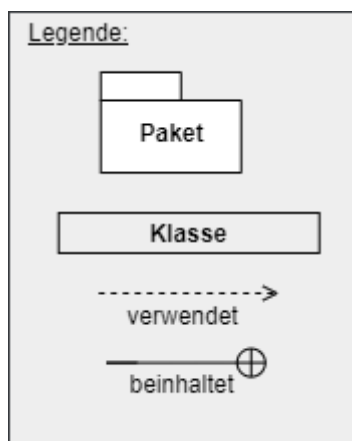
Klassendiagramm



Paketdiagramm

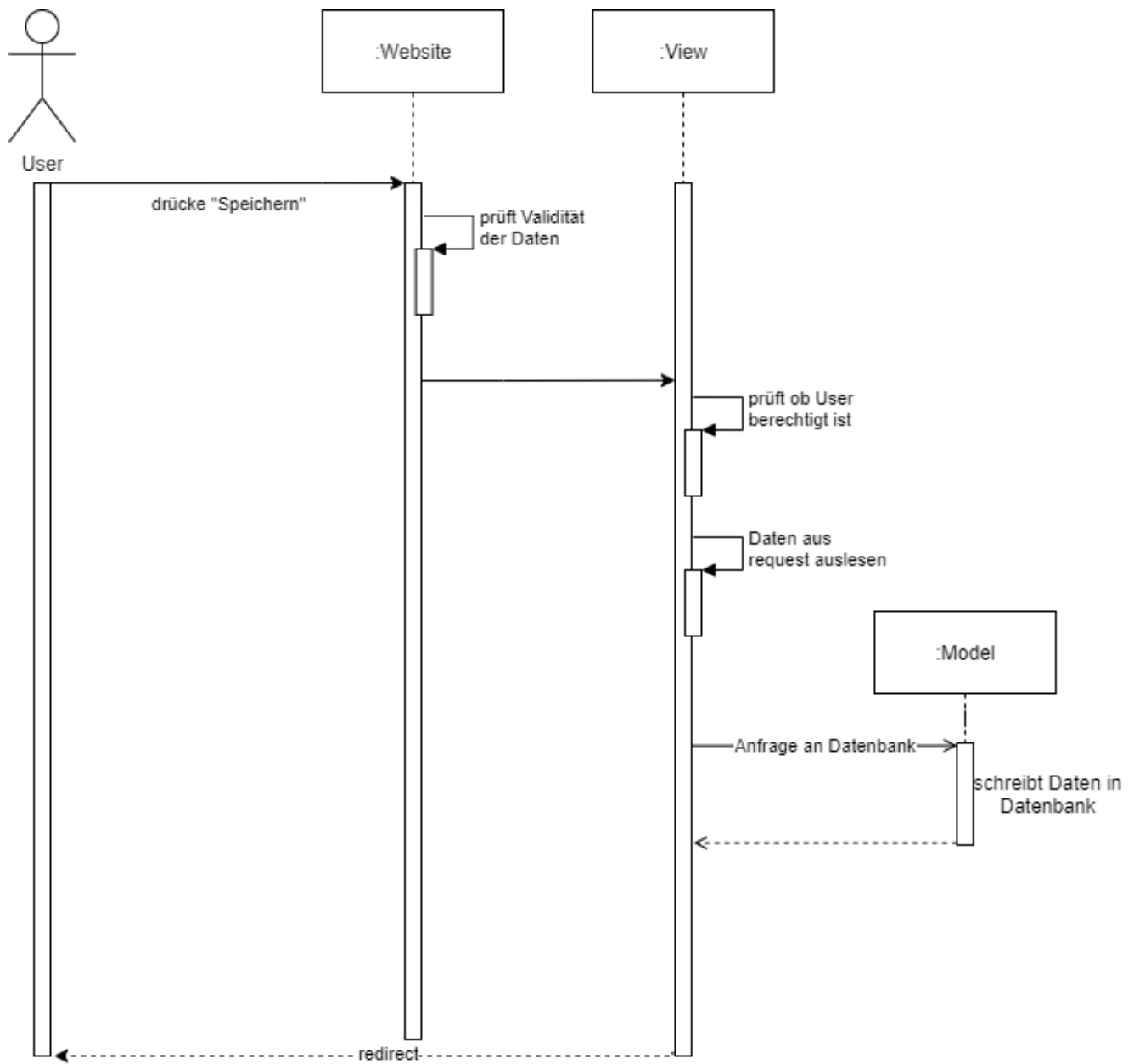


Paketdiagramm mit Klassen

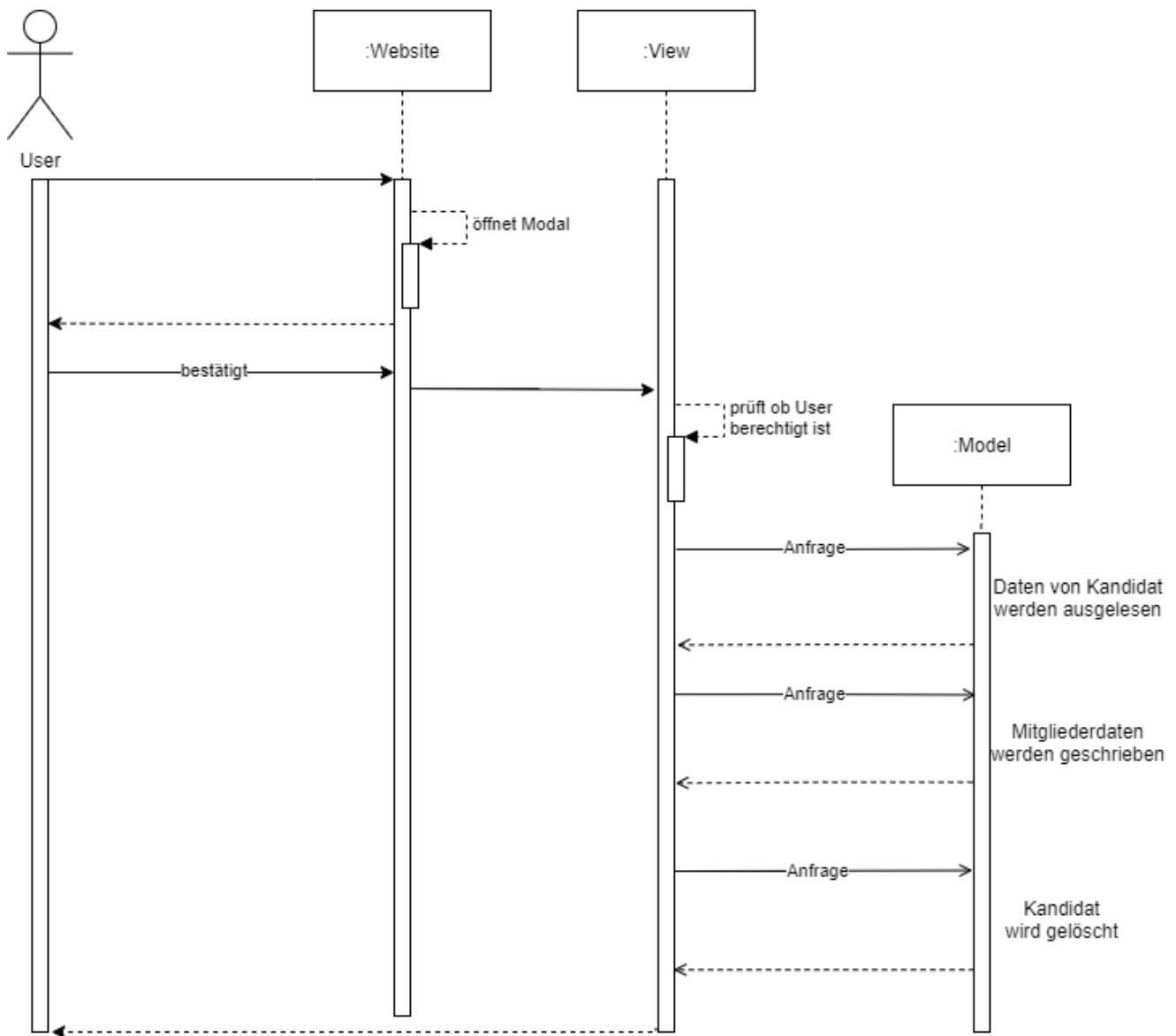


Ablaufsicht

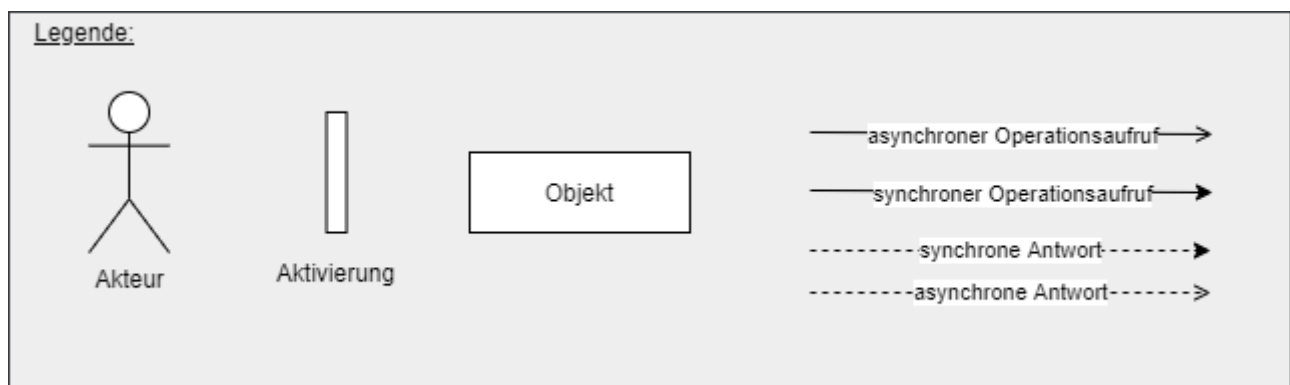
Sequenzdiagramme



UC01: Kandidat hinzufügen

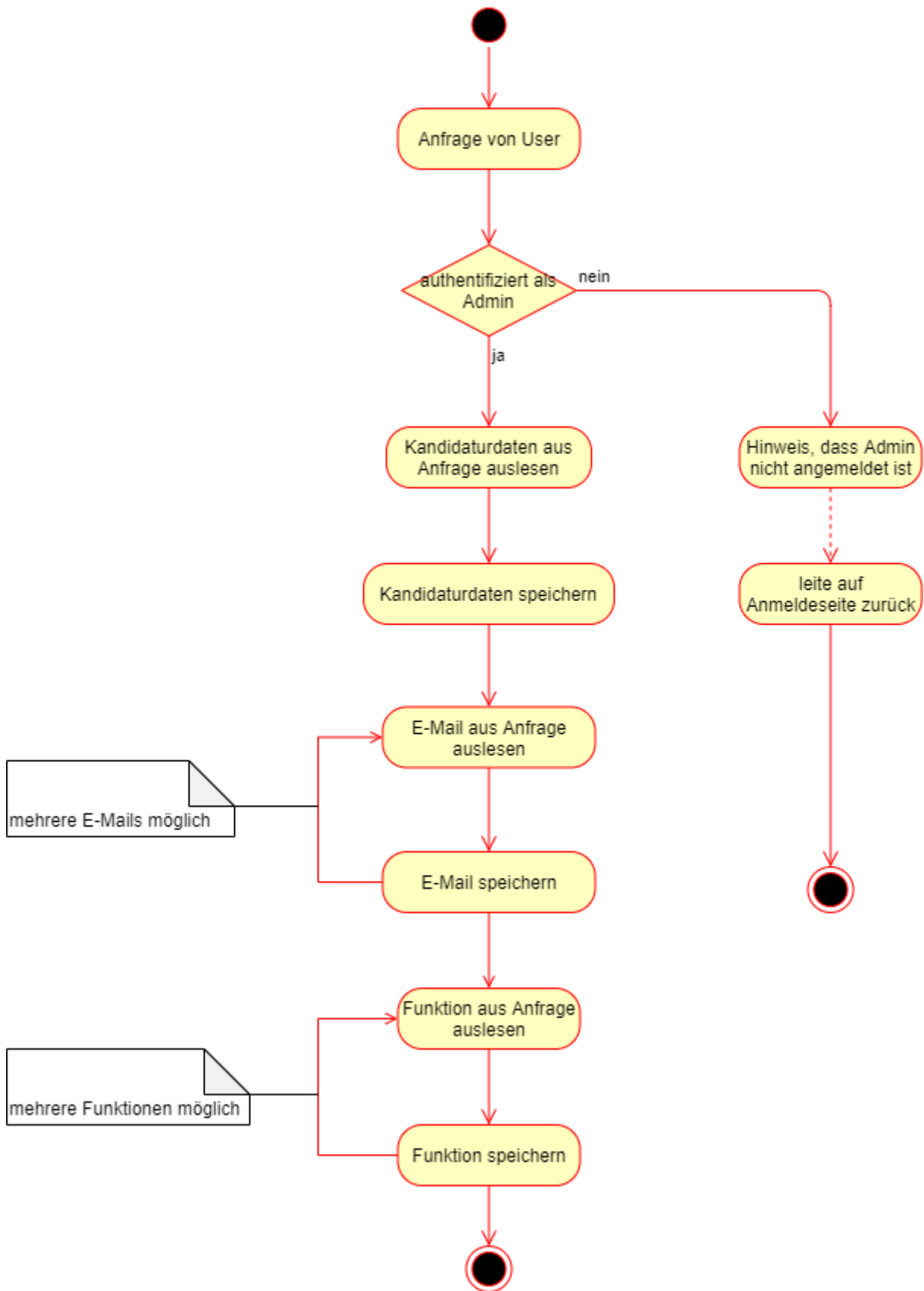


UC02: Kandidat aufnehmen

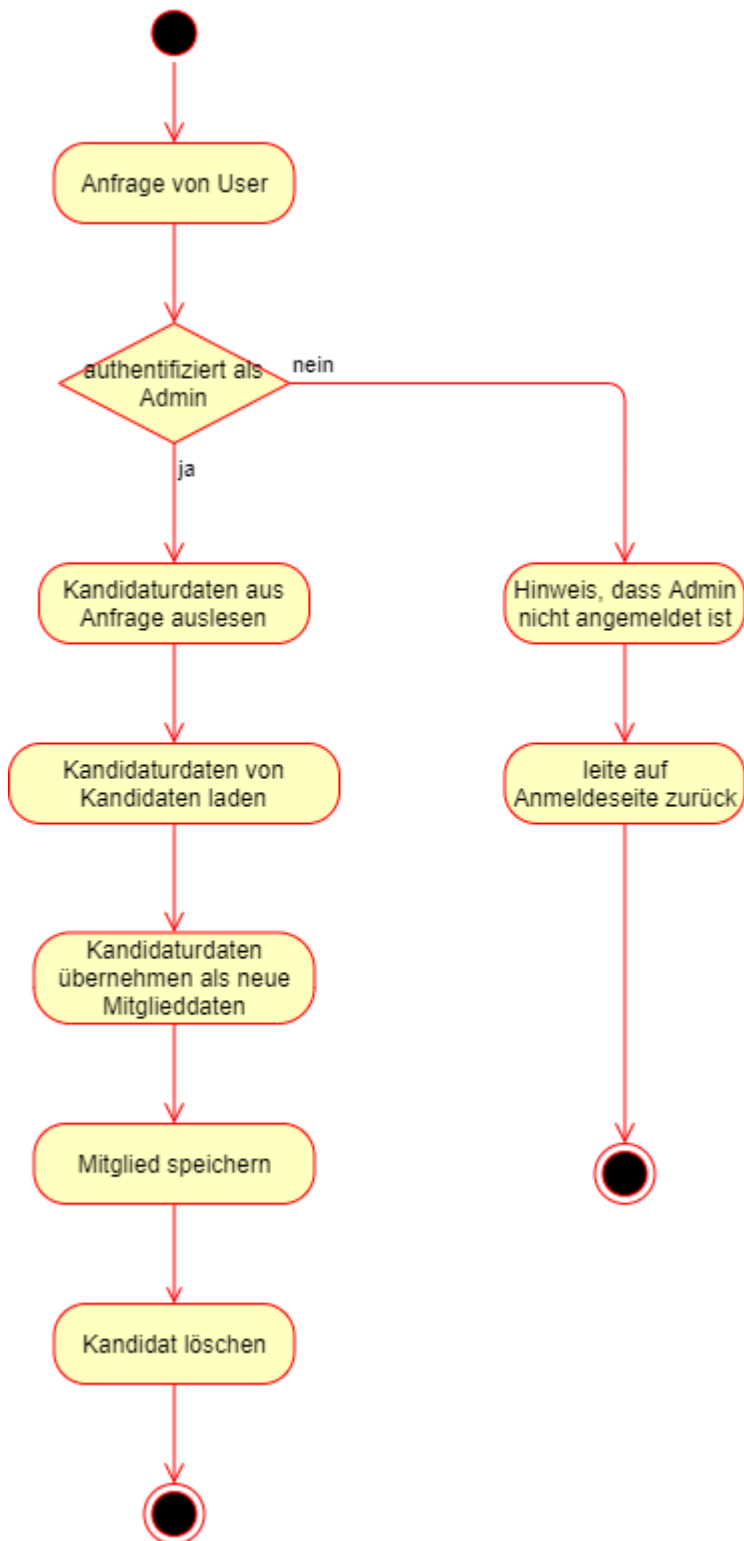


Szenarien

Aktivitätsdiagramme



UC01: Kandidat hinzufügen



UC02: Kandidat aufnehmen

