Software-Praktikum - Campus App

1 Kurzbeschreibung

Das Projekt „Uni Marburg Campus App“ soll ein mobiler Begleiter für alle Studierenden und Mitarbeiter der Universität Marburg sein. Sie soll ihnen dabei helfen schnell und einfach an aktuelle Informationen zu kommen, um damit den Uni-Alltag besser zu organisieren.
Die Kernanforderungen der Campus App sind zum einen auf den eigenen Veranstaltungsplan zuzugreifen, die ILIAS Aufgaben und die Foren Posts zu prüfen, aber auch sich den Veranstaltungskalender anzeigen zu lassen. Außerdem soll es möglich sein ein Feedback zu besuchten Veranstaltungen zu geben und sich zu Vorlesungsräumen/Gebäuden navigieren zu
lassen. Zudem soll es auch eine Such- und Filterfunktion geben, um andere Studierende zu finden,die dieselben Kurse wie Sie besuchen.
In Zukunft soll die Campus App auch in der Lage sein die Nutzer mit Kommilitonen zu verbinden, damit sie miteinander chatten können und damit direkt Einzel- bzw. Gruppengespräche auf der Plattform durchzuführen ohne Telefonnummer tauschen zu müssen. Des weiteren soll sich Veranstaltungsteilnehmer direkt vom Handy aus einchecken und Teilnehmer-Listen erstellt können. Zudem soll es möglich sein Hausaufgaben einzuscannen (fotografieren) und abzugeben. Außerdem soll es möglich sein die Buspläne und den Mensa-Speiseplan zu prüfen.
Die Campus App soll die Plattformen Ilias und Marvin verbinden, damit die Nutzer der App nur noch eine Plattform für die Organisation des Uni-Alltags benötigen.
Das Projekt wurde von der Philipps-Universität Marburg in Auftrag gegeben, um den Uni-Alltag zu verbessern.

Zeitliche Vorgaben
- bis 22.5. (Meilenstein 1)
    - Verfeinerung der Anforderungsbeschreibung mit spezifizierten Testfällen und Prozessmodellen
- bis 17.6. (Meilenstein 2)
    - OOP Model und Implementierungsentwurf
- bis 8.7. (Meilenstein 3)
    - Implementierung, Testfälle und Dokumentation
- bis 29.7.
    - Abschlusspräsentation

2 Entwicklungsprozess

Prozessmodell: Das geplante Prozessmodell (Wasserfallmodell) wird unter Punkt 4 - “Planung” - im Detail dargestellt. Das Einhalten dieser Ziele soll durch wöchentliche Meetings sichergestellt und kontrolliert werden.
Code Conventions: Es soll sich an den Code Conventions von Google orientiert werden. Dazu soll das CheckStyle Plugin in AndroidStudio verwendet werden.
Testen: Es wird sich vorgenommen, das Test-First Prinzip anzuwenden. An den Stellen wo das nicht umgesetzt werden kann, sollen die Tests zumindest möglichst früh geschrieben

werden, um während der Implementierung in fein-granularen Intervallen die Funktionalität und Auswirkung auf adjazente Programmteile überprüfen zu können.

3 Anforderungen

- **Definition des Zielsystems**

  Unterstützt werden sollen grundsätzlich alle Android-Endgeräte, die über die Android- Version 7.1 oder höher verfügen.

- **Funktionale Anforderungen**

  Soweit nicht explizit etwas anderes angegeben ist, wird bei allen spezifizierten Anforderungen der Nutzer der Anwendung als Akteur vorausgesetzt. Zudem wird für alle Anwendungsfälle, in denen der Nutzer als Akteur vorausgesetzt wird, die erfolgreiche Anmeldung des Nutzers am Server vorausgesetzt.

3.1 Anmeldung

3.1.1 Am Server anmelden

Kurzbeschreibung: Zur Nutzung der Applikation ist zunächst eine Anmeldung am Server notwendig.

Vorbedingung: Der Nutzer ist noch nicht angemeldet.

Ablauf: Zum Anmelden muss der Nutzer entweder seinen Benutzernamen eingeben oder aber die Email-Adresse, mit der er beim Server registriert ist. Zudem muss er sein Passwort eingeben. Bestätigt er die Daten, werden diese an den Server übermittelt.

Abbruchszenarien: Konnten die Daten vom Server nicht erfolgreich entgegengenommen werden, wird der Nutzer darüber informiert und die Anmeldung wird abgebrochen. Der Nutzer ist dann nicht am Server angemeldet.

Nachbedingung: Die Anmeldung wurde an den Server übermittelt und dort akzeptiert. Und die Überblicksseite geöffnet.

3.2 Veranstaltungsplan

3.2.1 Alle Veranstaltungen anzeigen

Kurzbeschreibung: Der Nutzer kann sich alle auf dem Server verfügbaren Veranstaltungen anzeigen lassen.

Vorbedingung: Der Nutzer ist auf dem Server angemeldet.

Ablauf: Dem Nutzer werden in einer Übersicht alle auf dem Server gespeicherten Veranstaltungen angezeigt. In der Übersicht werden jeweils die Veranstaltungsnummer, Veranstaltungsname, die Veranstaltungsform, der Studiengang, die Modulkürzel, Veranstaltungszeiten, Veranstaltungszeiträume, Veranstaltungsräume und Verantwortliche Dozenten angezeigt. Der Nutzer hat außerdem die Möglichkeit, die Veranstaltungen nach Namen, und Verantwortlichen Dozenten zu filtern (Anwendungsfall 3.2.3).

Nachbedingung: Dem Nutzer werden alle existierenden Veranstaltungen in einer Übersicht angezeigt.

3.2.2 Eigene Veranstaltungen anzeigen

Kurzbeschreibung: Der Nutzer kann sich alle Veranstaltungen anzeigen lassen, die er in diesem Semester belegt hat.

Vorbedingung: Keine

Ablauf: Dem Nutzer werden in einer Übersicht alle Veranstaltungen angezeigt, die in dem angemeldeten Nutzerkonto als belegt markiert sind. In der Übersicht werden jeweils die Veranstaltungsnummer, Veranstaltungsname, die Veranstaltungsform, der Studiengang, die Modulkürzel, Veranstaltungszeiten, Veranstaltungszeiträume, Veranstaltungsräume und Verantwortliche Dozenten angezeigt. Der Nutzer hat außerdem die Möglichkeit, die Veranstaltungen nach Veranstaltungsnamen, Veranstaltungsform und Verantwortlichen Dozenten zu filtern (Anwendungsfall 3.1.3)

Nachbedingung: Dem Nutzer werden alle seine angemeldeten Veranstaltungen in einer Übersicht angezeigt.

3.2.3 Angezeigte Veranstaltungen filtern

Kurzbeschreibung: Die Übersicht der Veranstaltungen kann nach Veranstaltungsname, Veranstaltungsform und verantwortlichen Dozenten gefiltert werden.

Vorbedingung: Eine Übersicht über alle Veranstaltungen (Anwendungsfall 3.1.1) oder über die eigenen Veranstaltungen (Anwendungsfall 3.1.2) wird angezeigt.

Ablauf: Der Nutzer bekommt die Möglichkeit ein Filterkriterium einzugeben, mit dem die Übersicht der Veranstaltungen gefiltert werden soll. Die Übersicht der Veranstaltungen soll dabei bei jeder Änderung des Filterkriteriums aktualisiert werden. In der Übersicht werden nur Veranstaltungen angezeigt, für die der Veranstaltungsname und/oder die Veranstaltungsform und/oder der verantwortliche

Dozent das Filterkriterium erfüllt. Löscht der Nutzer das Filterkriterium, werden entsprechend wieder alle Veranstaltungen angezeigt.

Nachbedingung: Eine entsprechend dem Filterkriterium gefilterte Übersicht der Veranstaltungen wird angezeigt.

3.2.4 Veranstaltung suchen

Kurzbeschreibung: Der Nutzer kann eine Suchanfrage formulieren und Veranstaltungen anzeigen lassen, die den gewünschten Suchkriterien entsprechen.

Vorbedingung: Eine Übersicht über alle Veranstaltungen (Anwendungsfall 3.1.1) wird angezeigt.

Ablauf: Der Nutzer bekommt die Möglichkeit eine Suchanfrage zu starten. Das Ergebnis enthält die Veranstaltungen, die den gewünschten Suchkriterien entspricht. Außerdem enthält das Ergebnis Information darüber, wann, wo und ob die gesuchte Veranstaltung in diesem Semester stattfindet.

Nachbedingung: Eine entsprechend der Suchkriterien gefilterte Übersicht der Veranstaltungen wird angezeigt.

3.3 Details von Veranstaltungen

3.3.1 Details anzeigen

Kurzbeschreibung: Der Nutzer kann nach Auswahl einer Veranstaltung auf Details zugreifen, wie auf den Name der Veranstaltung, Typ der Veranstaltung, Anbieter, Semester, Zeit, Tag, Raum und Teilnehmer.

Vorbedingung: Der Nutzer hat eine Veranstaltung ausgewählt.

Ablauf: Dem Nutzer werden Details der ausgewählten Veranstaltung angezeigt. Angezeigt werden Name der Veranstaltung, Typ der Veranstaltung, Anbieter, Semester, Zeit, Tag, Raum und Teilnehmer.

Nachbedingung: Dem Nutzer werden Details der Veranstaltung angezeigt.

3.3.2 Inhalte aus Ilias anzeigen

Kurzbeschreibung: Der Nutzer kann nach Auswahl einer Veranstaltung auf Unterlagen und Foren der Veranstaltung zugreifen.

Vorbedingung: Der Nutzer hat eine Veranstaltung ausgewählt.

Ablauf: Dem Nutzer werden verfügbare Inhalte aus Ilias angezeigt, wie Kursforum, Vorlesungsmaterial und Übungsblätter.

Nachbedingung: Dem Nutzer werden Unterlagen und Foren der Veranstaltung angezeigt.

3.3.3 Alle Teilnehmer anzeigen

Kurzbeschreibung: Der Nutzer kann sich die Teilnehmerliste von Veranstaltungen, die er selbst besucht, anzeigen lassen.

Vorbedingung: Der Nutzer hat eine Veranstaltung ausgewählt und die Teilnehmerliste geöffnet.

Ablauf: Dem Nutzer werden alle Teilnehmer der ausgewählten Veranstaltung chronologisch, nach Nachnamen sortiert angezeigt. Angezeigt werden jeweils der Name der Nutzerkonten. Der Nutzer erhält zudem die Möglichkeit, mit einer Suchanfrage nach Teilnehmern in der Veranstaltungen zu suchen. (Anwendungsfall 3.3.4).

Nachbedingung: Dem Nutzer werden alle Teilnehmer der ausgewählten Veranstaltung angezeigt.

3.3.4 Angezeigte Teilnehmer suchen

Kurzbeschreibung: Der Nutzer kann in der Teilnehmerliste, von Veranstaltungen, die er selbst besucht, andere Teilnehmer suchen.

Vorbedingung: Der Nutzer hat eine Veranstaltung ausgewählt und die Teilnehmerliste geöffnet.

Ablauf: Der Nutzer bekommt die Möglichkeit eine Suchanfrage zu starten. Das Ergebnis enthält die Nutzer, die den gewünschten Suchkriterien entspricht.

Nachbedingung: Dem Nutzer werden alle Teilnehmer mit dem gesuchten Namen angezeigt.

3.3.5 Details über Raum

Kurzbeschreibung: Der Nutzer kann von der Detailansicht der Veranstaltungen auf die Detailansicht des Raumes wechseln.

Vorbedingung: Der Nutzer hat eine Veranstaltung ausgewählt.

Ablauf: Der Nutzer wählt zunächst die Detailansicht Raum aus (Anwendungsfall 3.5.1) Der Nutzer erhält zudem die Möglichkeit mit der Eingabe der Zeit und des

Datums nach leeren Räumen zu suchen (Anwendungsfall 3.5.2) oder zu dem Vorlesungsraum/Gebäude zu navigieren (Anwendungsfall 3.5.3)

Nachbedingung: Dem Nutzer wird der Zeitplan des Raumes in einer Übersicht angezeigt.

Abweichung: Aufgrund von Designentscheidungen kann man der Nutzer nicht durch die Eingabe von Zeit und Datum nach einem leeren Raum suchen.

3.3.6 Ilias Aufgaben und Foren überprüfen

Kurzbeschreibung: Der Nutzer kann seine Ilias Aufgaben und Forum Posts überprüfen.

Vorbedingung: Der Nutzer ist auf dem Server angemeldet.

Ablauf: Dem Nutzer werden in einer Übersicht alle Neuigkeiten, wie neue Übungen und neue Forenposts angezeigt mit Uploaddatum und -uhrzeit angezeigt.

Nachbedingung: lias Aufgaben und Forum Posts werden angezeigt.

3.4 Vorlesungskalender

3.4.1 Vorlesungskalender anzeigen

Kurzbeschreibung: Der Nutzer kann sich seinen Vorlesungskalender (Stundenplan) mit seinen belegten Veranstaltungen in dem aktuellen Semester anzeigen lassen.
Vorbedingung: Der Nutzer ist auf dem Server angemeldet und hat seinen Vorlesungskalender ausgewählt.

Ablauf: Dem Nutzer werden seine gewählten Veranstaltungen in einem Vorlesungskalender angezeigt. Er bekommt Informationen, wann und wo seine Veranstaltungen stattfinden.

Nachbedingung: Dem Nutzer wird sein Vorlesungskalender angezeigt.

3.5 Raumpläne

3.5.1 Zeitplan für einen Raum

Kurzbeschreibung: Der Nutzer kann sich den Zeitplan eines Raumes anzeigen lassen.

Vorbedingung: Der Nutzer ist auf dem Server angemeldet und hat die Raumpläne ausgewählt.

Ablauf: Der Nutzer gibt den Namen eines Raums ein und ihm wird zunächst der Zeitplan des Raumes angezeigt. Der Nutzer erhält zudem die Möglichkeit, zu dem Vorlesungsraum/Gebäude zu navigieren (Anwendungsfall 3.5.3) und durch Eingabe der Zeit und des Datums nach leeren Räumen zu suchen (Anwendungsfall 3.5.1)

Nachbedingung: Dem Nutzer wird der Zeitplan des Raums angezeigt.

Abweichungen: Aufgrund von Designentscheidungen kann man der Nutzer nicht durch die Eingabe von Zeit und Datum nach einem leeren Raum suchen.

3.5.2 Leere Räume suchen

Kurzbeschreibung: Der Nutzer kann nach leeren Räumen suchen.

Vorbedingung: Der Nutzer ist auf dem Server angemeldet und hat die Raumpläne ausgewählt.

Ablauf: Der Nutzer gibt Zeit und Datum ein und ihm wird zunächst eine Übersicht von leeren Räumen angezeigt. Der Nutzer erhält zudem die Möglichkeit, zu dem Vorlesungsraum/Gebäude zu navigieren (Anwendungsfall 3.5.3).

Nachbedingung: Dem Nutzer werden leere Räume zu den Suchkriterien angezeigt.

Abweichung: Aufgrund von Designentscheidungen kann der Nutzer nicht durch eingabe von Zeit und Datum nach leeren Räumen suchen. Der Nutzer kann stattdessen einen Raum eingeben und sich dann den Raumplan ansehen. Anhand des Raumplans kann er sehen, ob der Raum zu der gewünschten Zeit frei oder besetzt ist.

3.5.3 Zu den Vorlesungsräumen/Gebäuden navigieren

Kurzbeschreibung: Der Nutzer kann sich den Weg zu einem Vorlesungsraum/Gebäude anzeigen lassen.

Vorbedingung: Der Nutzer ist auf dem Server angemeldet.

Ablauf: Der Nutzer gibt die Raumnummer/ Gebäudenummer des gesuchten Raum/Gebäude in das Suchfeld ein und startet die Suche.

Nachbedingung: Der Nutzer wird auf Google Maps weitergeleitet und die Route zum gesuchten Raum/ Gebäude wird angezeigt.

Abweichung: Der Nutzer wird erst auf die Detailansicht des gesuchten Raums geleitet, wo der Raumplan und die Adresse des Raums angezeigt wird. In der Detailansicht kann er sich dann den Raum auf Maps anzeigen lassen und sich zum Gebäude navigieren lassen.

3.5.4 Feedback zu Vorlesungen geben

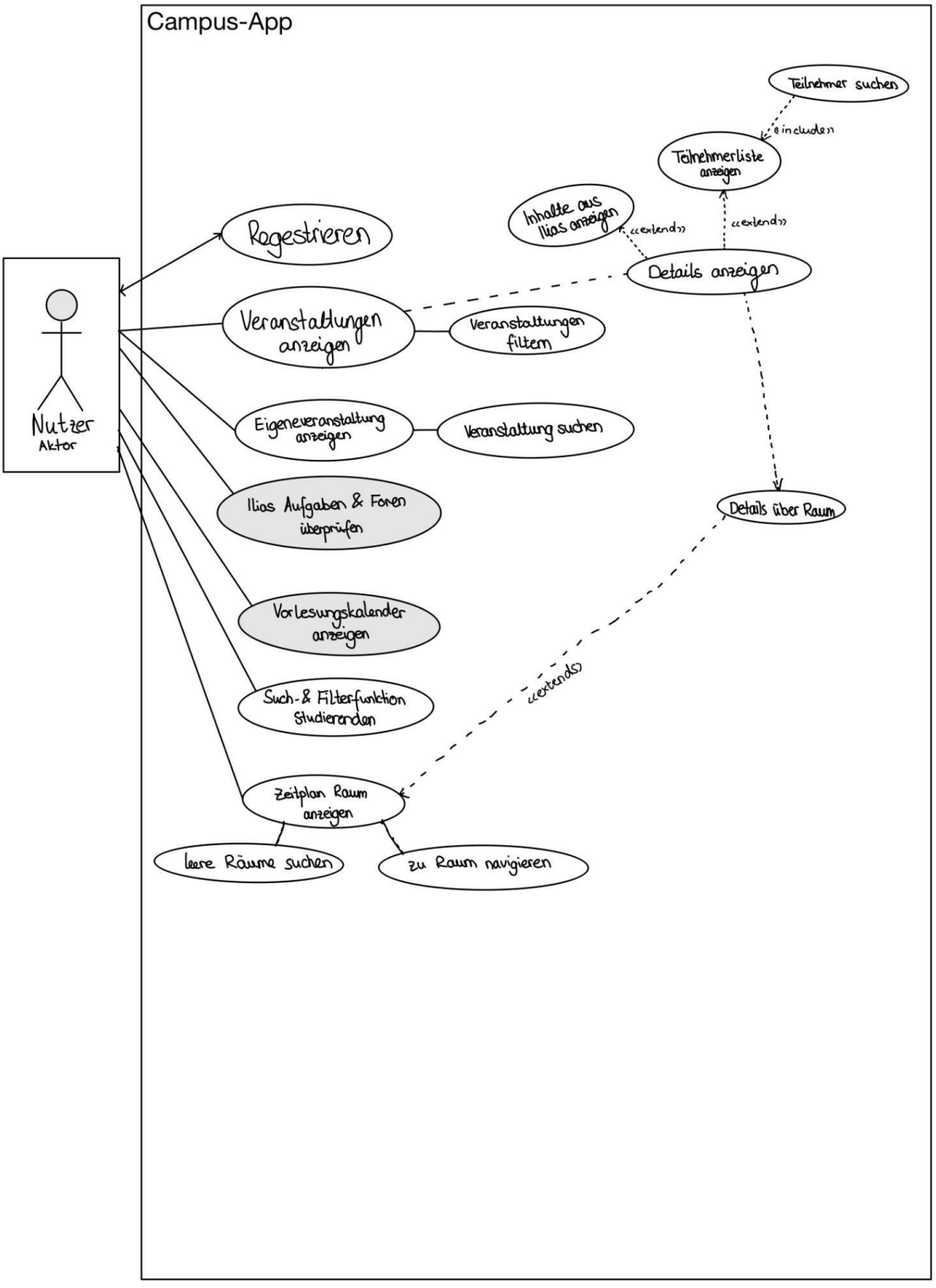
Kurzbeschreibung: Der Nutzer kann Feedbacks zu Vorlesungen geben, die er besucht hat.

Vorbedingung: Die Detailansicht der Vorlesung, zu der ein Feedback gegeben werden soll, ist geöffnet.

Ablauf: Der Nutzer gibt in ein Textfeld sein Feedback zur Vorlesung ein und drückt dann den Button „speichern“.

Nachbedingung: Das Feedback wird auf dem Server gespeichert und kann von weiteren Nutzern eingesehen werden.

Use Case Diagramm



- **Nicht-Funktionale Anforderungen**

  Externe Qualität des Systems

  Performanz:
  Die Bedienung der App sollte korrekt und ohne größere Verzögerungen erfolgen. Die Such- und Filterfunktionen, das Durchklicken von Detailansichten, die Anzeige des Vorlesungsplans soll ohne größere Verzögerungen möglich sein.

  Usability:
  Die App sollte einfach und verständlich aufgebaut werden, sodass Nutzer bereits bei der ersten Bedienung der App in der Lage sind diese in vollem Umfang korrekt zu nutzen. Die App sollte selbsterklärend aufgebaut sein.
  Das heißt der Nutzer sollte keine Probleme beim Verständnis der Funktionen haben, um in Folge dessen auch Fehler vermeiden zu können.

  Sicherheit: Wenn sich der User erfolgreich einloggt, soll idealerweise der Server einen access token erzeugen und über SLL an die App senden. Dieser soll dann für eine begrenzte Zeit gültig sein und kann für die Kommunikation mit dem Server verwendet werden.

  Portierbarkeit: Unterstützt werden sollen grundsätzlich alle Android-Endgeräte, die über die Android- Version 7.1 oder höher verfügen. Die App sollte in jeder Auflösung darstellbar sein. Zum Beispiel sollte die App auch für Tablets angeboten werden.

  Zuverlässigkeit: Die App soll bei allen Eingaben des Nutzers geeignet reagieren und nicht abstürzen. Die Applikation soll stabil laufen, auch wenn der Nutzer unvorhergesehene oder falsche Eingaben macht.

  Wartbarkeit: Durch ein entsprechende Softwarearchitektur soll die Campus-App von anderen Programmierern einfach erweiterbar und einfach wartbar sein.

  Rückmeldung an den Nutzer: Der Nutzer der App muss bei jeder Aktion immer ein baldmöglichstes Feedback bekommen. Insbesondere sind Probleme bei der Kommunikation mit dem Server und Fehler bei der Ausführung von durch den Nutzer veranlassten Aktionen durch verständliche Fehlermeldungen anzuzeigen. Wenn möglich, sollten Fehlermeldungen den Nutzer bei der Behebung der Fehler unterstützen.

  Benutzeroberfläche: Die Benutzeroberfläche der Applikation soll mindestens die Sprachen Deutsch und Englisch unterstützen. Zudem soll eine unkomplizierte Erweiterung um zusätzliche Sprachen durch das Design der Applikation ermöglicht werden.
  Die Benutzeroberfläche der Applikation sollte sich an etablierten Bedienkonzepten orientieren, um Nutzern eine möglichst intuitive Bedienung zu ermöglichen. Die

Design- Guidelines für Android (https://developer.android.com/design) sollten dabei eingehalten werden.

Interne Qualität der Codes

Da die App ggf. erweitert oder verbessert wird, sollte der Code gut dokumentiert und kommentiert sein, sodass neue Mitglieder der App-Entwicklung ihn schnell verstehen können.

- **Glossar**

  Nutzer: Die Nutzer der App sind Studierende und Mitarbeiter der Universität Marburg.
  Veranstaltung: Unter Veranstaltungen werden Vorlesungen, Übungen, Tutorien, Seminare, etc. zusammengefasst.
  Veranstaltungsteilnehmer: Die Veranstaltungsteilnehmer nehmen an einer Veranstaltung teil.
  Kurs: Ein Kurs ist eine Untermenge der Veranstaltungen.
  Kursteilnehmer: Die Kursteilnehmer nehmen an einem Kurs teil.
  Studienräume: Unter Studienräumen versteht man Räume, in denen eine Veranstaltung stattfindet.
  Forum: Im Forum kann man sich über Neuigkeiten, Fragen, Problemen, etc. austauschen.
  Chat: Im Chat stellt man Kontakt zu anderen Nutzern her und kann miteinander Informationen austauschen.

4 Planung

4.1 Meilensteine

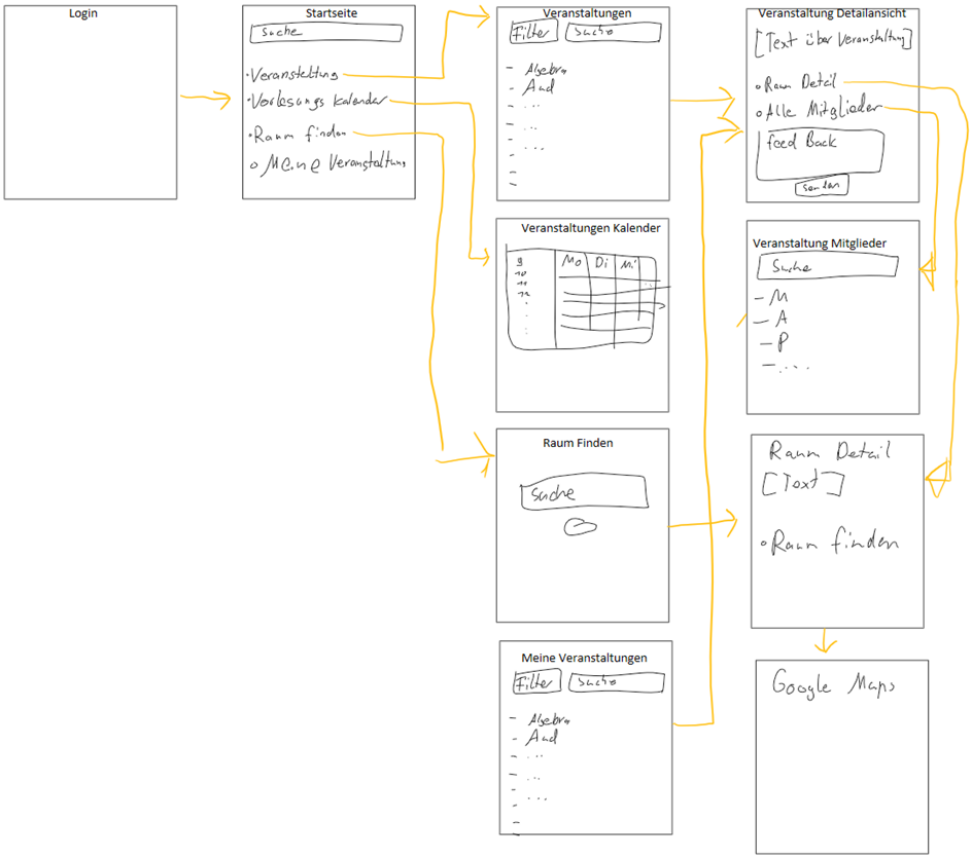
- bis 22.5. (Meilenstein 1)
  - Verfeinerung der Anforderungsbeschreibung mit spezifizierten Testfällen und Prozessmodellen
- bis 17.6. (Meilenstein 2)
  - OOP Model und Implementierungsentwurf
- bis 8.7. (Meilenstein 3)

    - Implementierung, Testfälle und Dokumentation
- bis 29.7. (Meilenstein)
    - Abschlusspräsentation

4.2 Detaillierte Beschreibung des Meilenstein 1

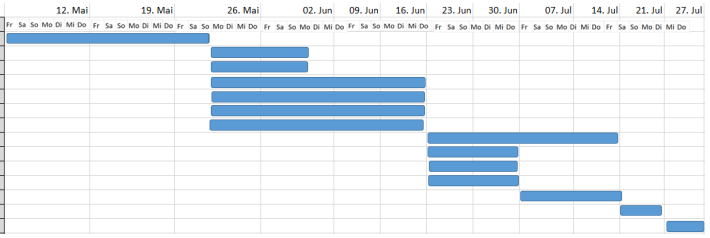
Als ersten Masnahme wurde eine Teambesprechung geplannt um eine Übersicht über das Projekt sowie den vorstehenden Meilenstein zu bekommen. Es wurden die Meilensteine besprochen um eine gemeinsame Vision zu haben.
Für den Meilenstein 1 wurden die zu erledigenden aufgaben in Gruppen eingeteilt. Die Kurzbeschreibung und Entwicklungsprozesse wurden zunächst von allen Teammitgliedern gemeinsam am ersten Tag beschrieben, um ein besseres Verständnis für das Projekt zu bekommen. Für die übrigen Aufgaben wurden zwei Wochen geplant. Im folgenden wurden die Anforderungen von Christine und Annie spezifiziert. Die Zeitplanung und Priorisierung der einzelnen Anforderungen und Dokumente wurde von Alex erstellt. Der genaue Entwurf der Datenstrukturen und die umzusetzende Architektur des Projektes wurde von Christoph mithilfe von UML entworfen.

4.3 Interface Struktur



4.4 Verfügbarkeitsplanung / Anforderungspriorisierung / Zeitplanung

Anforderungs Nr.	Anforderung/Dokumentname	Dauer
	Meilenstein1_Dokument	22 Tage
3,1,1	Login	8 Tage
3,1,2	Startseite	8 Tage
3,2,2	Meine Veranstaltungen Anzeigen	18 Tage
3,2,1	Alle Veranstaltungen Anzeigen	18 Tage
3,5,3	Nach einem Raum Suchen	18 Tage
3,4,1	Veranstaltungskalender von Ilias anzeigen	18 Tage
3,3,1	Veranstaltungs Detailansicht	15 Tage
3,3,5	Raum Detailansicht	7 Tage
3,2,4	Nach Veranstaltungen Suchen	7 Tage
3,2,3	Nach Veranstaltungen Filtern	7 Tage
3,3,5	Zu einem Raum Navigieren	7 Tage
3,3,3	Alle Mitglieder Anzeigen	3 Tage
3,3,4	Nach Mitlgiedern Suchen	3 Tage

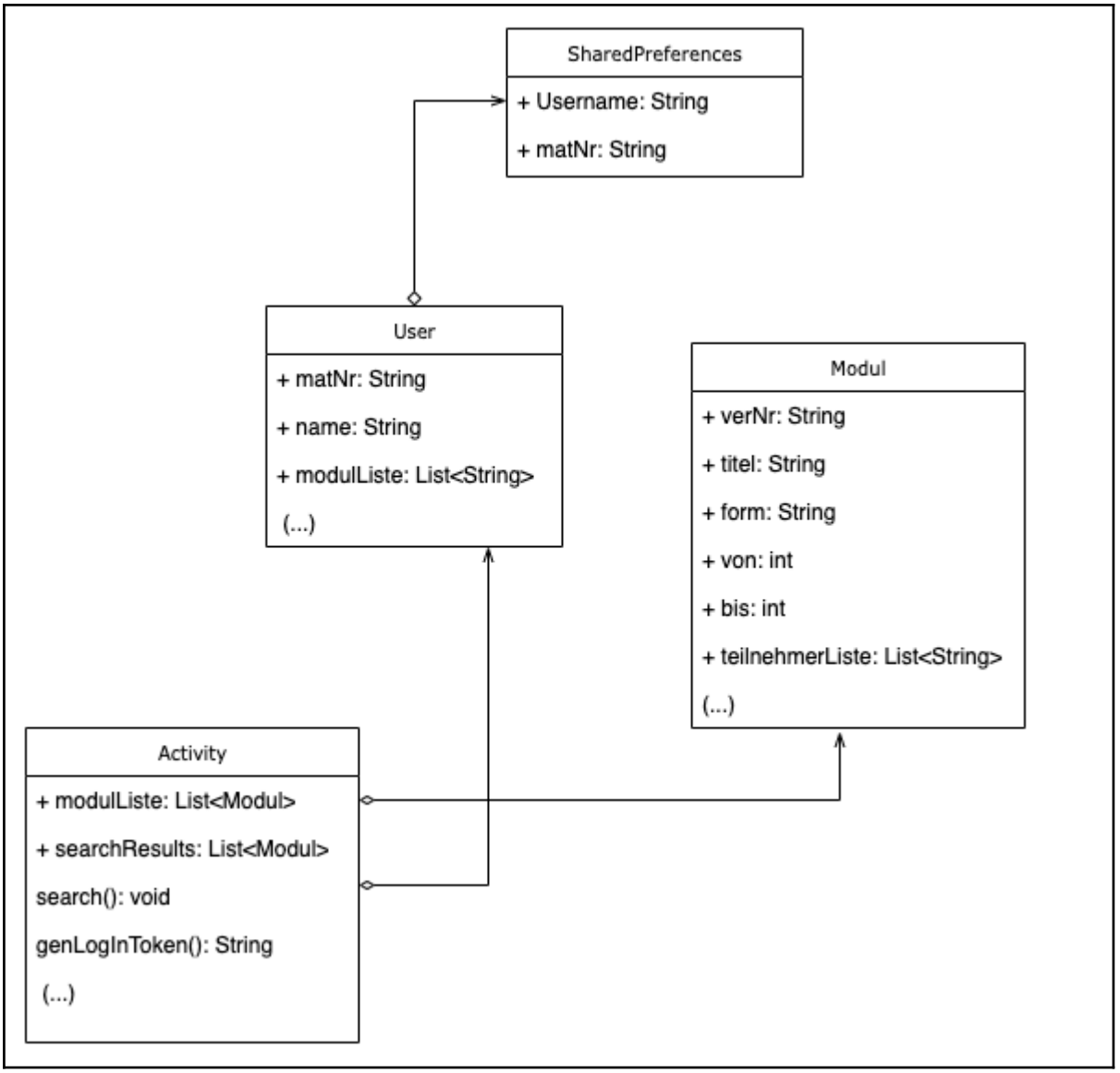


Das Diagramm Zeigt nicht nur den Aktuellen Meilenstein, sondern auch die in naher Zukunft anstehenden Anforderungen. Die Anforderung wurden nach Priorität geordnet, die Anforderungen die zuerst erledigt werden müssen stehen in dem Zeitstrahl weiter Links. Das Diagramm zeigt zudem dass gewisse Anforderung parallel bearbeitet werden können. Zeitgleich wird die Verfügbarkeit der einzelnen Anforderungen dargestellt indem.

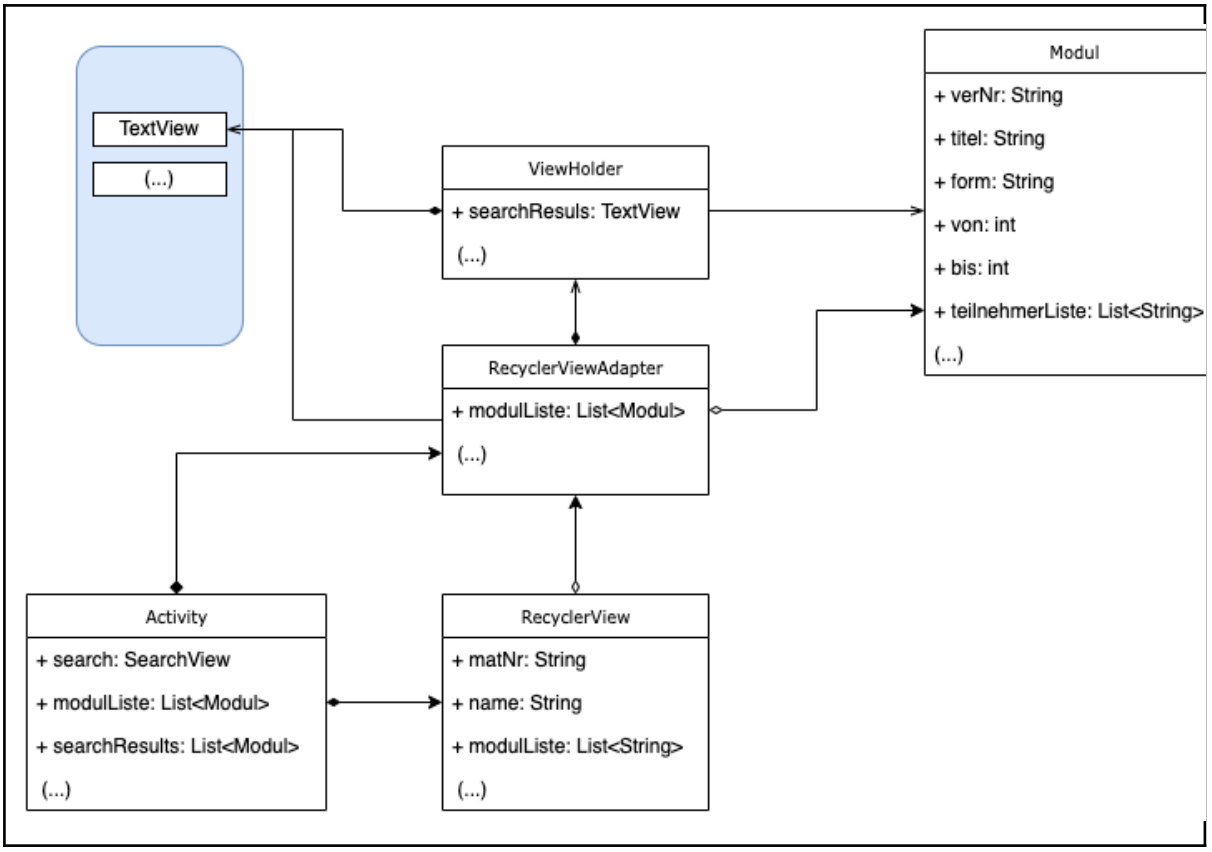
5 Entwurf

UML Diagramm 1:

UML_1 zeigt einen Ausschnitt der grundlegenden Klassenstruktur, die unsere wichtigsten Daten in der App abbilden soll. Die Activity liest den aktuellen Modulplan über die API ein. Das eingelesene xml-Dokument wird geparsed und aus je einer Zeile im Modulplan eine Modul-Instanz erzeugt. Diese werden in einer Liste gespeichert, die dann als Grundlage für die Suchfunktion dienen kann. SharedPreferences ermöglicht es Nutzerinformationen, wie Nutzername und Matrikelnummer über den App-Lifecycle hinaus lokal zu persistieren.

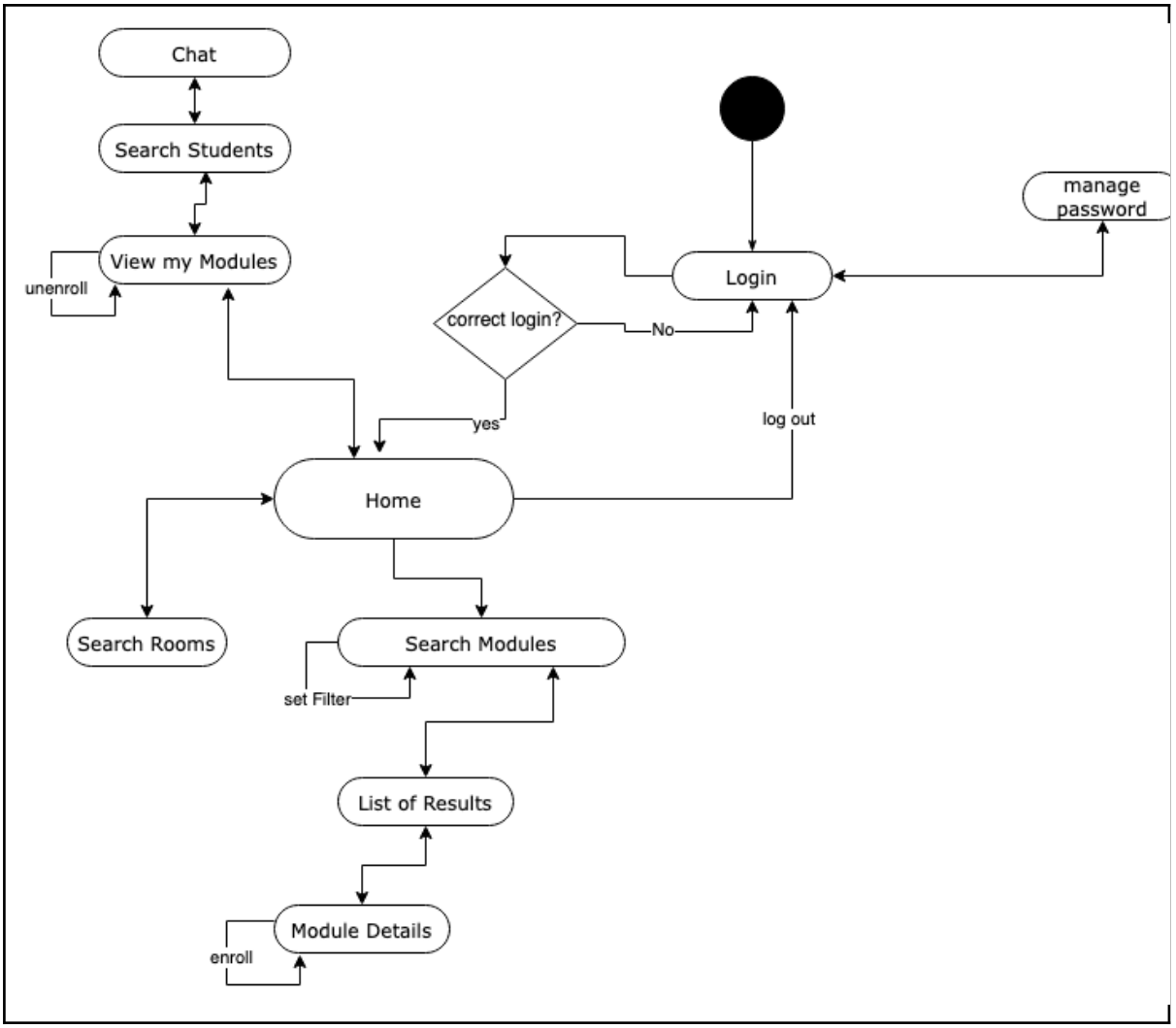


UML Diagramm 2:
UML_2 zeigt einen Ausschnitt der konkreten Android-Komponenten zur Umsetzung der Modulsuche. Zur Darstellung der Suchergebnisse wird, für bessere Performance, ein RecyclerView mit ViewHolder-Pattern verwendet.



UML Diagramm 3:

UML_3 zeigt die Kernfunktionen als Ablaufdiagramm.



6 Testen

Anwendungsfall: 3.3.5 Details über Raum
Testname: Erfolgreiches anzeigen der Details eines Raums
Vorbedingung: Der Nutzer hat die Detailansicht einer Veranstaltung geöffnet.
Ablauf: Der Nutzer klickt auf den Button Raumdetails.
Erwartetes Verhalten: Das Fenster mit den Raumdetails inklusive des Zeitplan des Raumes wird angezeigt.
Tatsächliches Verhalten: Das erwartete Verhalten tritt ein.

Anwendungsfall: 3.5.1 Zeitplan für einen Raum
Testname: Erfolgreiche anzeigen des Zeitplan eines Raumes

Vorbedingung: Der Nutzer hat die Raumsuche geöffnet.
Ablauf: Der Nutzer gibt die Raumnummer des Raums ein, deren Zeitplan er sucht, und drückt dann auf den Button „suche“.
Erwartetes Verhalten: Die Raumdetails werden angezeigt mit den Zeitplan, an dem der Raum belegt ist.
Tatsächliches Verhalten: Das erwartete Verhalten tritt ein.

Anwendungsfall: 3.5.2 Leere Räume suchen
Testname: Erfolgreiche Suche eines Rums
Vorbedingung: Der Nutzer hat die Raumsuche geöffnet.
Ablauf: Der Nutzer gibt eine Raumnummer ein, die existiert, und drückt dann auf den Button „suche“.
Erwartetes Verhalten: Raumdetails des gesuchten Raums werden angezeigt.
Tatsächliches Verhalten: Das erwartete Verhalten tritt ein.

Anwendungsfall: 3.5.2 Leere Räume suchen
Testname: Fehlschlagen der Raumsuche
Vorbedingung: Der Nutzer hat die Raumsuche geöffnet.
Ablauf: Der Nutzer gibt eine Raumnummer ein, die nicht existiert, und drückt dann auf den Button „suche“.
Erwartetes Verhalten: Es wird ein Hinweis angezeigt, dass der Raum nicht existiert.
Tatsächliches Verhalten: Das erwartete Verhalten tritt ein.

Anwendungsfall: 3.5.3 Zu den Vorlesungsräumen/ Gebäuden navigieren
Testname: Erfolgreiches Navigieren zum Raum
Vorbedingung: Der Nutzer hat die Raumdetails eines Raums geöffnet.
Ablauf: Der Nutzer drückt auf den Button „Auf Maps anzeigen“.
Erwartetes Verhalten: Der Nutzer wird auf Google Maps weitergeleitet und der gesuchte Raum wird angezeigt.
Tatsächliches Verhalten: Das erwartete Verhalten tritt ein.

Anwendungsfall: 3.2.3 Alle Veranstaltungen filtern
Testname: Erfolgreiches Filtern
Vorbedingung: “Alle Veranstaltungen” ist bereits geöffnet und die Veranstaltungen werden angezeigt
Interaktion: Der Nutzer wählt ein Filter aus
Erwartetes Verhalten: Die Veranstaltungen werden maßgeblich des angewendeten Filters gefiltert.
Tatsächliches Verhalten: Wie erwartet

Anwendungsfall: 3.2.4 Veranstaltung suchen
Testname: Erfolgreiche Suche
Vorbedingung: “Alle Veranstaltungen” ist bereits geöffnet und die Veranstaltungen werden angezeigt
Interaktion: Der Nutzer gibt in die Suche ein Suchbegriff ein (Name oder Dozent)
Erwartetes Verhalten: Die Veranstaltungen, die den Suchbegriff enthalten, werden angezeigt.

Tatsächliches Verhalten: Wie erwartet

Anwendungsfall: 3.3.1 Details anzeigen
Testname: Details anzeigen
Vorbedingung: “Alle Veranstaltungen” ist bereits geöffnet und die Veranstaltungen werden angezeigt
Interaktion: Der Nutzer klickt eine Veranstaltung an.
Erwartetes Verhalten: Die Details der Veranstaltung werden angezeigt.
Tatsächliches Verhalten: Wie erwartet

Anwendungsfall: 3.3.1 Details anzeigen
Testname: Von Details zur Raumsuche
Vorbedingung: Die Details einer Veranstaltung werden angezeigt.
Interaktion: Raumplan wird angeklickt.
Erwartetes Verhalten: Raumsuche mit dem Raum der Veranstaltung im Suchfeld wird angezeigt.
Tatsächliches Verhalten: Wie erwartet

7 Benutzer- und Entwicklerdokumentation

Abschlussbericht:

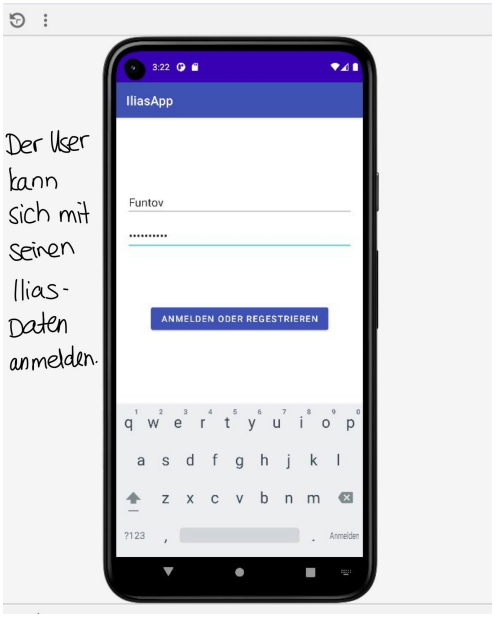
1. Kurzbeschreibung

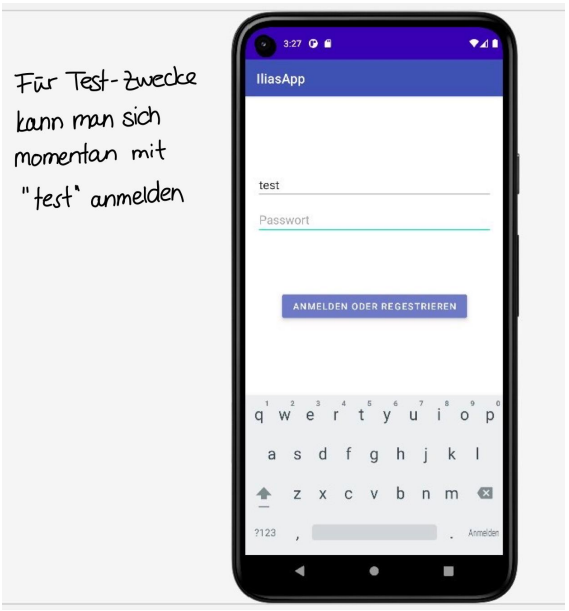
Die Campus-App kann mit Android-Endgeräten, die über die Android-Version 7.1 oder höher verfügen, genutzt werden.

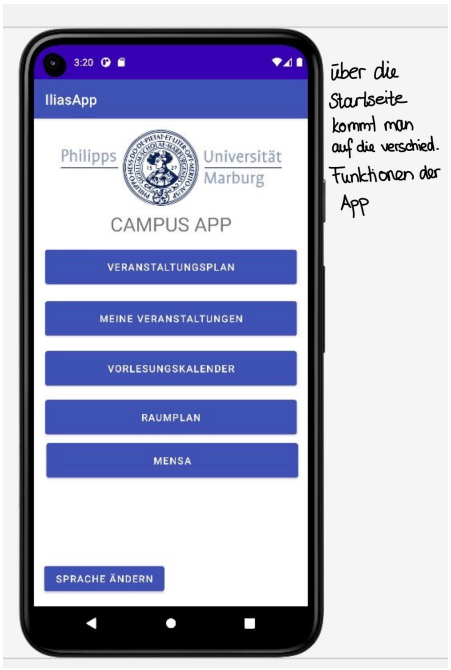
Damit an der Campus-App weitergearbeitet werden kann benötigt man vorzugsweise Java und Android Studio in der Version 2021.2.1.

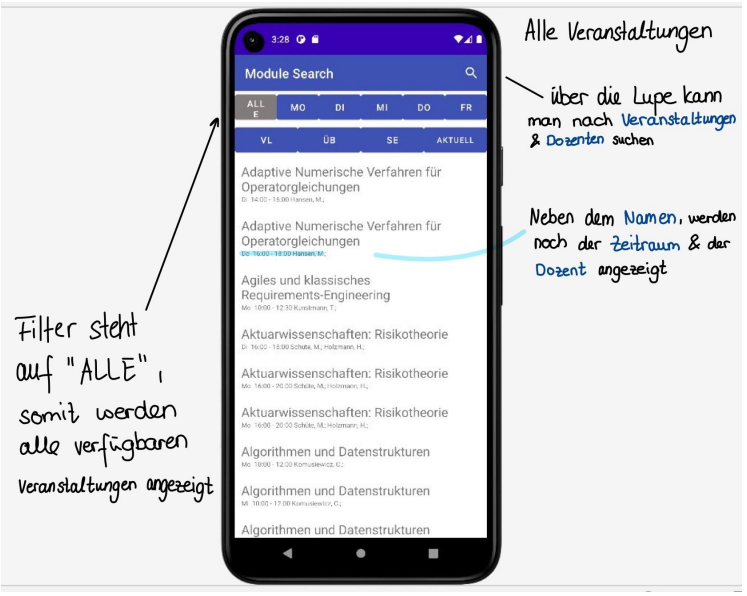
Funktionalität der App
Mit der App kann man über eine Startseite auf alle Veranstaltungen, eigene Veranstaltungen, Veranstaltungskalender, Raumnavigation und Mensaplan zugreifen. Die detaillierten Funktionen der App kann man unter Punkt zwei “User Anleitung” einsehen.

2. User Anleitung

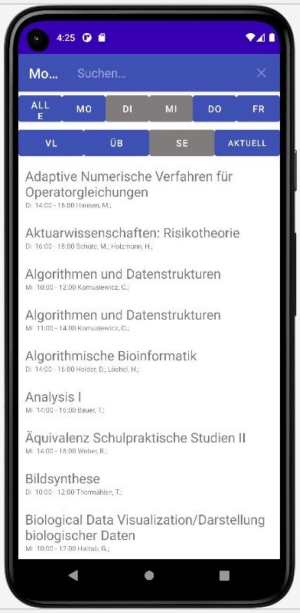


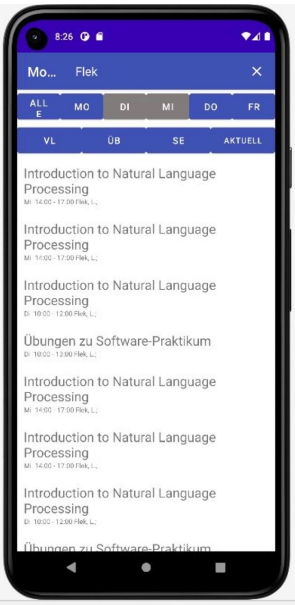




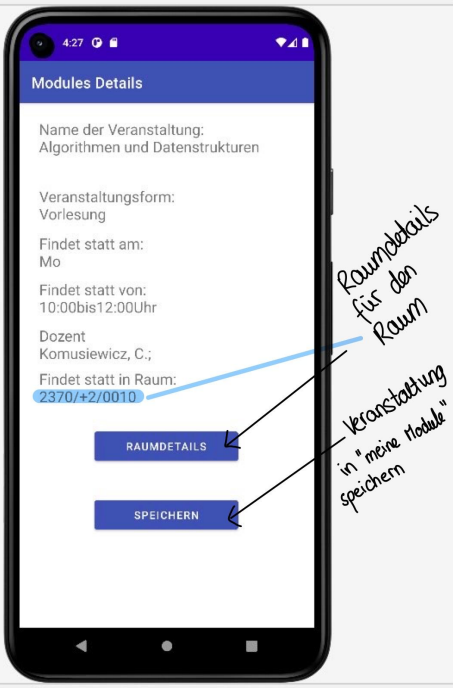


Der User hat die Möglichkeit mehrere Filter auf einmal auszuwählen





Der User hat die Möglichkeit Filter & Suche gleichzeitig anzuwenden

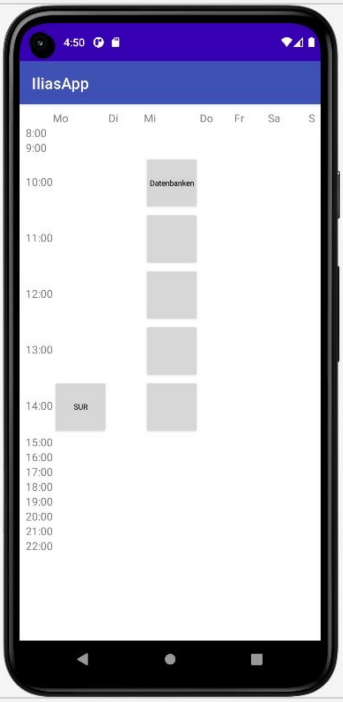


Raumdetails für den Raum

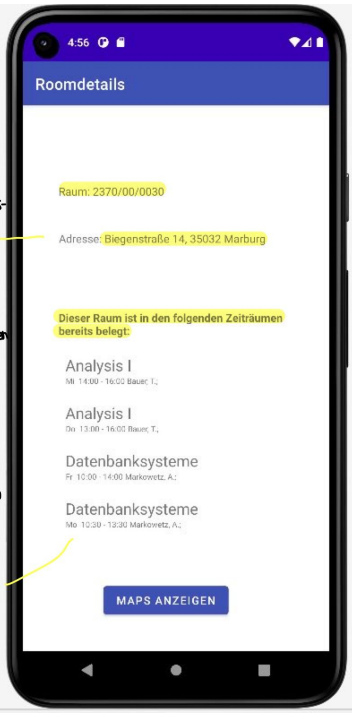
Veranstaltung in "meine Module" speichern

Über meine Veranstaltungen, kann der User die Veranstaltungen sehen, die er zuvor über den Speicher-Button gespeichert hat

nspection



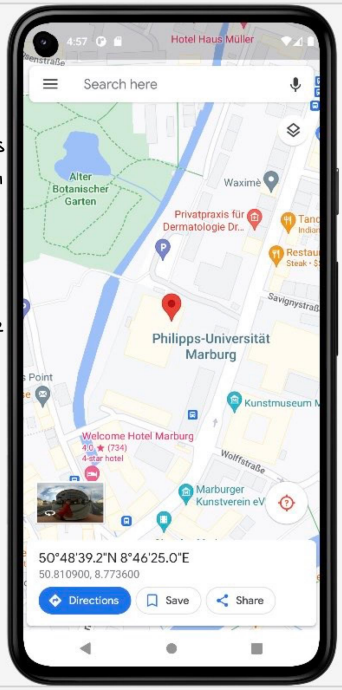
Stundenplan mit Beispiel Vorlesungen SuR und Datenbanken

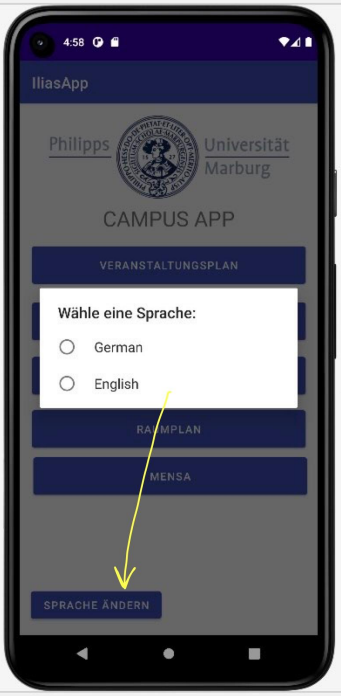


Zu dem gesuchten oder über die Veranstaltungsdetails ausgewählten Raum lässt sich die Adresse anzeigen

Außerdem kann man einsehen, zu welchen Zeiträumen der Raum bereits belegt ist.

Wenn der User über die Raumdetails auf Maps klickt, wird man auf Google-Maps weitergeleitet und zu der Adresse des Raums geführt.





über einen Button lässt sich die Sprache ändern

3. Entwicklerdokumentation:

3.1. Kurzbeschreibung der Komponenten

3.1.1. Login-Package:

Für den Login-Prozess wurde eine Abwandlung des (sehr komplizierten) Android Studio Templates für Login-Activities verwendet.

Login :
Android Activity, die den Login-Bildschirm darstellt, Passwort und Nutzername an LoginRepository weitergibt und
die UI mit den User-Daten ausfüllt, wenn die Validierung erfolgreich ist.

LoginRepository:
Gibt die Userinformationen zur Validierung an LoginDataSource weiter und gibt als Ergebnis eine Instanz von LoginResult an Login zurück.

LoginDataSource:
Sendet einen login request an die Ilias-test API. Bei erfolgreicher Validierung der Nutzerdaten durch die API wird eine LoggedInUser Instanz zurückgegeben.

LoggedInUser:
Modelliert einen validen Nutzer mit Nutzerdaten und einer validen Session ID.

LoginResult:
Modelliert einen erfolgreichen oder nicht erfolgreichen Login-Versuch über die Subklassen (1) Success die ein LoggedInUser Objekt verwaltet und (2) Error, die eine Exception als Begründung für den Fehlschlag beinhaltet.

3.1.2. Data-Package:

Um die Datenverwaltung von den Android Activities mit der Darstellung und Programmlogik etwas zu trennen, haben wir eine separate Klassenstruktur für die Datenverwaltung implementiert, die installiert werden kann um an den benötigten stellen leichter wiederverwendet werden zu können.

ModulSearchData:
Datenhaltungsklasse, welche die Modulinformationen von der Modulliste.xls mithilfe von JSoup einliest und aus jeder Zeile eine Modul-Instanz erzeugt. Die fertige Liste wird als ArrayList verwaltet.

Modul:
Modelliert ein Modul im Veranstaltungskatalog.

Gebäude:
Modelliert ein Uni-Gebäude für die Raumsuche.

DataAPIRequest:
Stellt einen request an die getCourseXML Methode der Ilias-test API. (Leider beinhaltet die test-api nur die Informationen für den chat-container, daher wird die Klasse nicht verwendet.)

UserRequest:
Stellt einen request an die searchUser Methode der Ilias-test API. (Leider erlaubt die API keinen Zugriff.)

FeedReaderContract:
Legt die Struktur der DatenBank für gespeicherte Module fest.

FeedReaderDbHelper:
Subklasse von SQLiteOpenHelper. Stellt sicher, dass Operationen wie die Erzeugung und Update der Datenbank nur ausgeführt werden, wenn nötig und nicht beim Start der App.

3.1.3. Zentrale Android Activities:

HomeScreen:
Hauptmenü — stellt Buttons für alle Funktionsbereiche der App dar.

Modulsuche:
Verwaltet den gesamten Modulkatalog in einem RecyclerView. Ein SearchView und mehrere Filterfunktionen machen die Modulliste durchsuchbar.

MeineVeranstaltungen:
Stellt alle Module dar, die in der sqLite Datenbank gespeichert wurden.

Kalender:
Stellt einen Kalender dar mit allen in MeineVeranstaltungen gespeicherten Modulen.

RecyclerViewAdapter:
Setzt das ViewHolder Pattern um.
Der Adapter greift auf die Daten in ModulSearchData zu und übergibt sie dem ViewHolder im RecyclerView um effizientes Recycling zu ermöglichen.

RaumDetails:
Darstellung aller für den User relevanten Informationen zu einem ausgewählten Modul. Greift auf eine sqLite Datenbank zu in der um Module zur Darstellung in MeineVeranstaltungen gespeichert oder wieder entfernt werden.

RaumSuche:
Darstellung der Raumdetails und Adressen von Uni Gebäuden.

4. Erfahrungsbericht:

Was gut lief:
Unsere Strategie die Aufgaben aufzuteilen, hat sich erstaunlich positiv auf die gemeinsame Bearbeitung der Aufgabe ausgewirkt und es entstand auch der Eindruck, dass an vielen Stellen jeder seine Stärken einbringen konnte. Durch die Aufteilung der Aufgaben hatte jedes Teammitglied einen eigenen Modularen Teil, an dem relativ selbstständig und ohne übermäßige Absprache gearbeitet werden konnte. An den Stellen wo diese Teile zusammengesetzt werden mussten bzw. aufeinander aufbauen, hat dies auch erstaunlich reibungslos funktioniert — das war zum Beispiel der Fall bei den Filter-Buttons, in der Modulsuche, welche von Christine, die einen großen Teil der Präsentation und UI designed und entwickelt hat, zum von Christoph implementierten RecyclerView hinzugefügt wurde.

Ein weiterer Vorteil dieser Aufteilung war, dass nicht jedes Teammitglied in die verwendeten Frameworks gleichermaßen tief eingearbeitet sein musste, sich damit jeder ein wenig spezialisieren konnte und so gemeinsam relativ effizient an der App gearbeitet werden konnte.

Schwierigkeiten:

Die Arbeit mit der Ilias/Marvin API hat sich als unerwartet aufwändig herausgestellt. Den Fehler den wir dabei gemacht haben, war uns nicht im Vornherein ausreichend über unseren Tutor und die Dokumentation über die Möglichkeiten der APIs zu informieren. So hatten wir z.B. zuerst eine Anfrage für die Modul-Informationen und die Nutzerinformationen geplant, nur um dann nach der fertigen Implementierung herauszufinden, dass diese Anfragen mit der test-api gar nicht möglich waren.

5. Weiterentwicklung der App

Aufgrund der Probleme mit der API konnten ein paar geplante Anforderungen nicht umgesetzt werden und können in der Zukunft noch implementiert werden:

3.3.2 Inhalte aus Ilias anzeigen
3.3.3 Alle Teilnehmer anzeigen/3.3.4 Angezeigte Teilnehmer suchen
3.3.6 Ilias Aufgaben und Foren überprüfen
3.5.2 Leere Räume suchen
3.5.4 Feedback zu Vorlesungen geben
Außerdem könnte die App um noch folgende weitere Funktionen erweitert werden:

- mit Kommilitonen verbinden und miteinander chatten
- Einzel- bzw. Gruppengespräche direkt auf der Plattform führen
- Hausaufgaben einscannen (fotografieren) und abgeben
- Busfahrpläne anzeigen
- Veranstaltungsteilnehmer direkt vom Handy einchecken und Teilnehmerlisten erstellen