
GUI Development – Praktikum Aufgabe 2

Robin Biegel, Fabian Schuler, Jonathan Eck

biegelro@hs-albsig.de, schulefa@hs-albsig.de, eckjonat@hs-albsig.de

Arztpraxis Android App mit Anbindung an eine Datenbank

Inhaltsverzeichnis

Inhaltsverzeichnis	2
Java-Teil	3
Model	3
Helper	4
UI	4
Patientenbereich	4
Mitarbeiterbereich	7
WS (Webservice)	12
MainActivity	12
Res	12
Drawable	13
Layout	13
Menu	13
Navigation	13
Values	13
UML-Aktivitätsdiagramm	14
.....	15
.....	15
.....	15
.....	15
.....	16
UML-Klassendiagramm	17

Unser Projekt beschäftigt sich mit der Entwicklung einer Arztpraxis-Applikation für Android-Geräte. In dieser können sich Patienten einloggen, um eine Einsicht auf ihre Termine und Rezepte zu erhalten und Terminanfragen stellen zu können. Mitarbeiter der Arztpraxis (Arzt, Arzthelferin) können sich ebenfalls einloggen, um eine Übersicht über alle eingekommenen Anfragen, alle Patienten und Termine zu bekommen. Diese Kommunikation wird mit Hilfe der Einbindung einer Datenbank umgesetzt.

Java-Teil

Model

In dem Package `com.example.arztpraxis.model` werden alle Entities (mit einem Konstruktor, Getter- und Setter-Methoden, einer `copyData`- und einer `toString`-Methode) für die zugehörige Datenbank angelegt:

- Die Klasse „Adress“ verweist auf die Sql-Tabelle „app_adresse“ und den zugehörigen Spalten: id, plz, wohnort, strasse, hausnummer.
- Die Klasse „Disease“ verweist auf die Sql-Tabelle „app_krankheit“ und den zugehörigen Spalten: id, beschreibung.
- Die Klasse „Drug“ verweist auf die Sql-Tabelle „app_medikament“ und den zugehörigen Spalten: id, bezeichnung.
- Die Klasse „Position“ verweist auf die Sql-Tabelle „app_position“ und den zugehörigen Spalten: id, bezeichnung. (In unserem Fall gibt es die Positionen „Arzt“ und „Arzthelferin“)
- Die Klasse „Person“ verweist auf die Sql-Tabelle „app_person“ und den zugehörigen Spalten: id, vorname, name, geburtstag, adresseid, geschlecht.
- Die Klasse „Employee“ verweist auf die Sql-Tabelle „app_mitarbeiter“ und den zugehörigen Spalten: id, pos_id, personalnummer, personid.
- Die Klasse „Patient“ verweist auf die Sql-Tabelle „app_patient“ und den zugehörigen Spalten: id, versicherungsnummer, personid, krankenkasseid.
- Die Klasse „HealthInsurance“ verweist auf die Sql-Tabelle „app_krankenkasse“ mit den zugehörigen Spalten: id, name, adresseid
- Die Klasse „Prescription“ verweist auf die Sql-Tabelle „app_rezept“ mit den zugehörigen Spalten: id, patientid, mitarbeiterid, medikamentid, krankheitid, ausstelldatum.
- Die Klasse „Treatment“ verweist auf die Sql-Tabelle „app_behandlung“ mit den zugehörigen Spalten: id, bezeichnung.
- Die Klasse „Schedule“ verweist auf die Sql-Tabelle „app_terminplan“ mit den zugehörigen Spalten: id, patientid, mitarbeiterid, datum, behandlungid.
- Die Klasse „ScheduleRequest“ verweist auf die Sql-Tabelle „app_terminanfragen“ mit den zugehörigen Spalten: id, prioritae, notiz, patientid, mitarbeiterid.

Helper

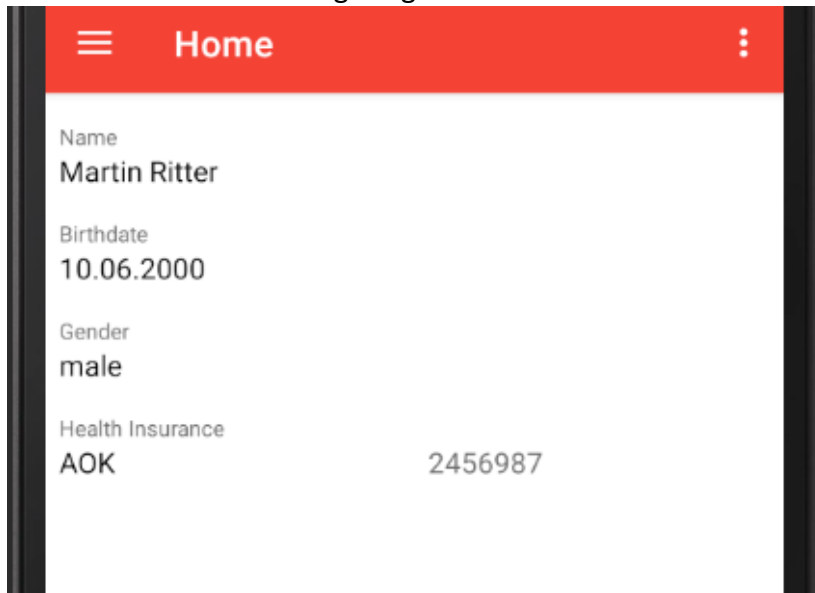
Im Package `com.example.arztpraxis.helper` wird eine Klasse „Helper“ erstellt. In dieser werden Strings implementiert, welche als Hilfsfunktionen dienen. Der String „formatDateTime“ hilft beim Formatieren des Date-Formats. Der String „getGender“ beinhaltet eine Switch-Case Fallunterscheidung, welche aus „m“ ein „male“, aus „w“ ein „female“ und standardmäßig ein „other“ zurückgibt.

UI

Im Package `com.example.arztpraxis.ui` werden alle benötigten Fragments, sowie die dazugehörigen ViewModels für die App erstellt.
Je nachdem, ob ein Patient oder ein Mitarbeiter eingeloggt ist, werden andere Fragmente angezeigt:

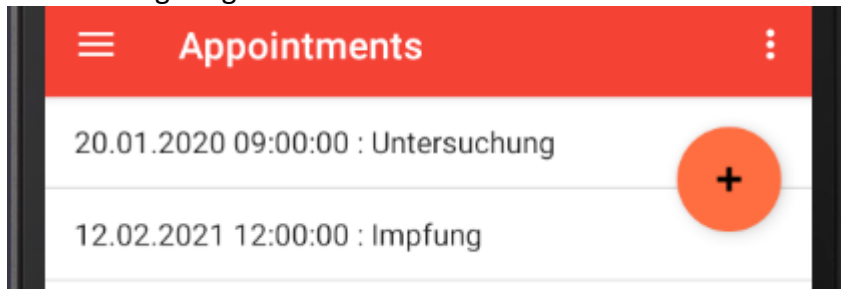
Patientenbereich

- HOME:
In das erste Package „home“ werden die Klassen „HomeFragment“ und „HomeViewModel“ implementiert. In diesem Fragment sollen alle persönlichen Daten eines Patienten angezeigt werden.



- APPOINTMENT

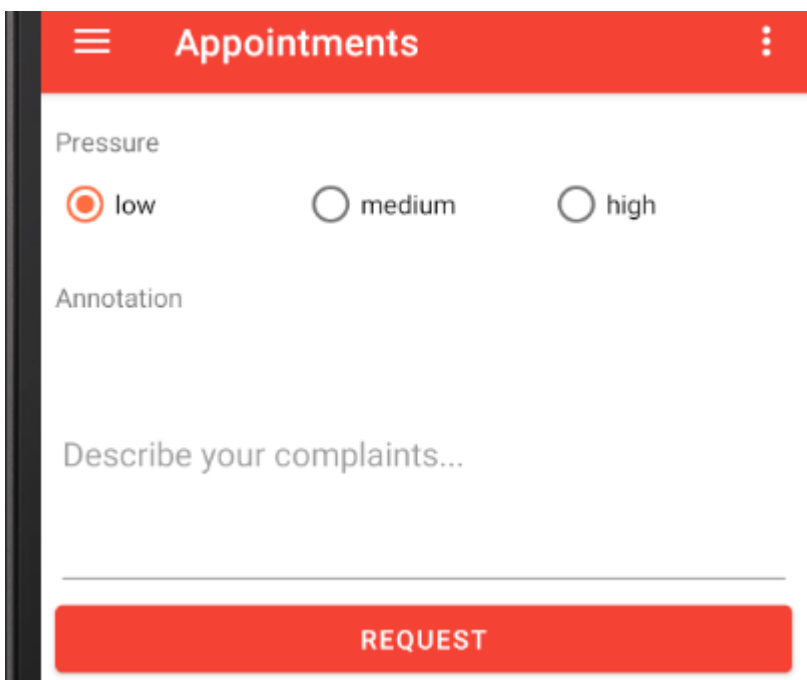
Das zweite Package „appointment“ beinhaltet die Klassen „AppointmentViewModel“ und „AppointmentFragment“, auf welchem eine listView für alle anstehenden Termine angezeigt wird.



Des Weiteren beinhaltet das Package die Klasse „AppointmentDetailFragment“ für eine detaillierte Ansicht der einzelnen Termine.



Die Klassen „AppointmentNewFragment“ und „AppointmentCreateFragment“ für das Erstellen und Abschicken einer Terminanfrage.

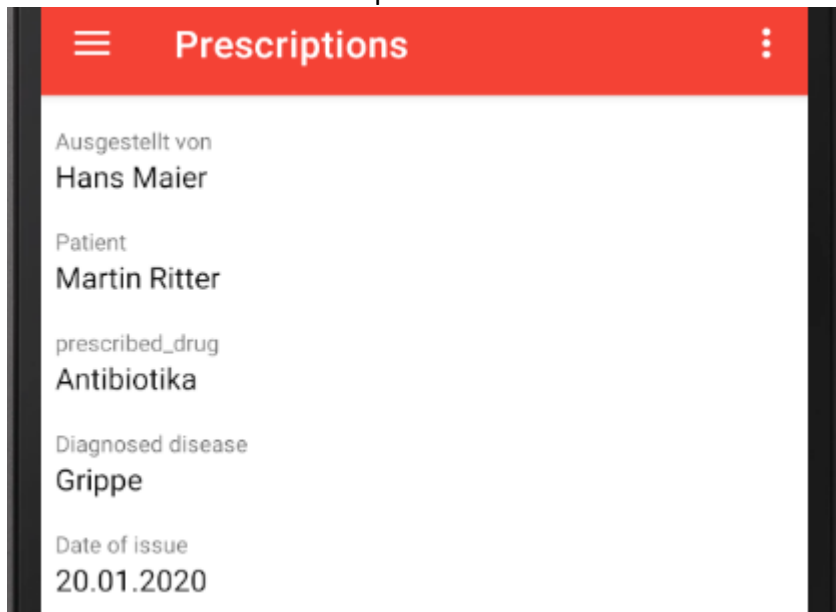


- **PRESCRIPTION:**

Das dritte Package „prescription“ besteht aus den Klassen „PrescriptionViewModel“ „PrescriptionFragment“ für eine listView aller Rezepte.



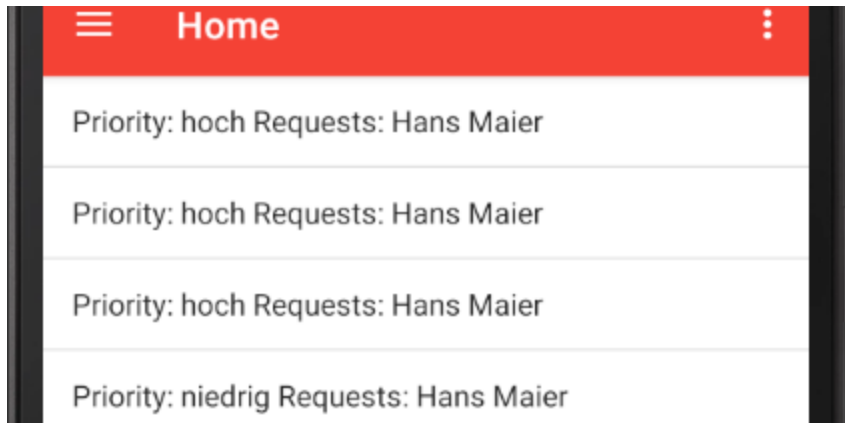
Des Weiteren existiert noch eine Klasse „PrescriptionDetailFragment“ für eine detaillierte Ansicht des Rezeptes.



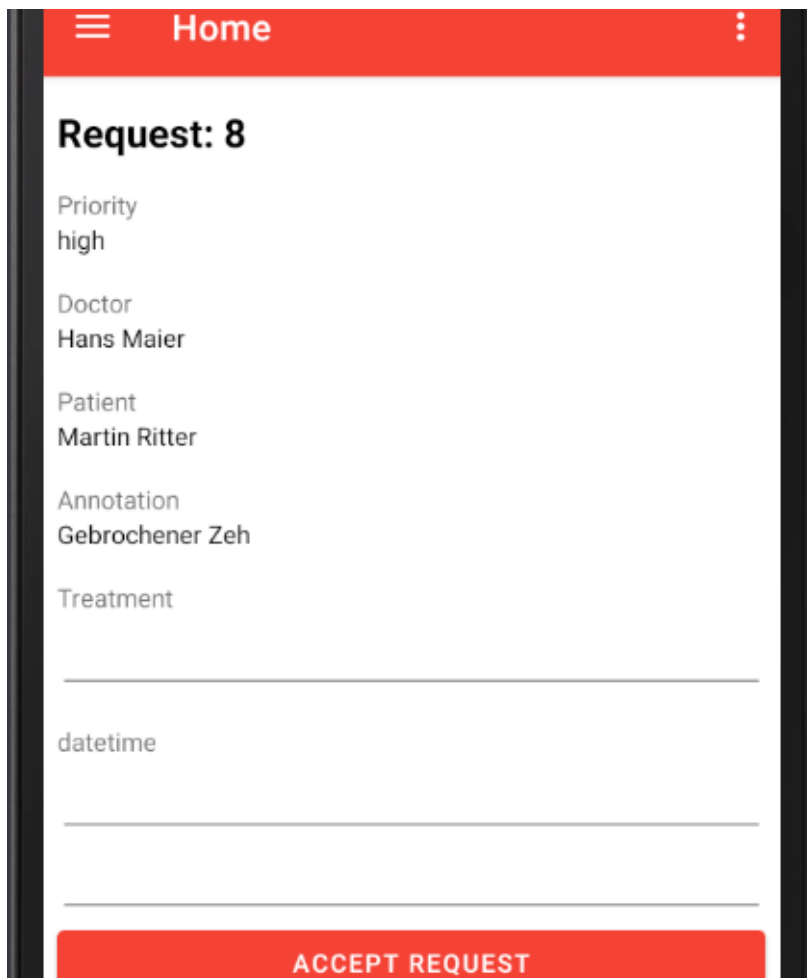
Mitarbeiterbereich

- HOME:

Aus Mitarbeitersicht wird ebenfalls das HomeFragment benutzt. Hierbei werden jedoch keine persönlichen Daten, sondern die Terminanfragen der Patienten aufgelistet.

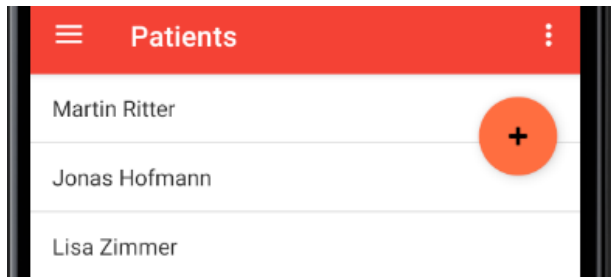


Zusätzlich wird das Package „adminHome“ erstellt und beinhaltet die Klassen „adminHomeDetailFragment“ und „adminHomeViewModel“. Dadurch kann eine Terminanfrage durch einen Klick detailliert angesehen werden und mit Ausfüllen der unteren Felder direkt akzeptiert werden.

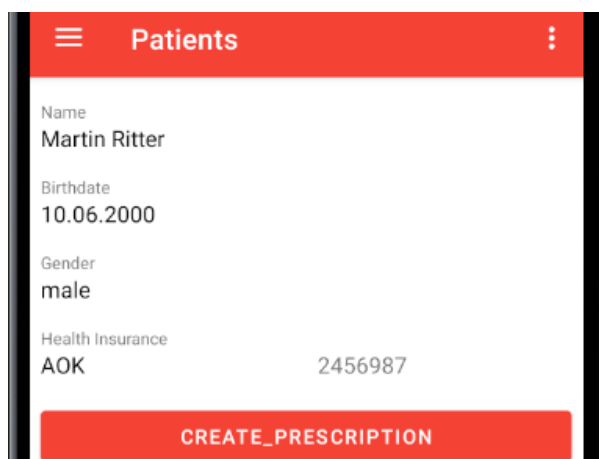


- PATIENTS:

Ein weiteres Package heißt „patient“ und enthält die Klassen „PatientViewModel“, „PatientFragment“, welche alle registrierten Patienten in einer Liste anzeigt.



Die Klasse „PatientDetailFragment“ ermöglicht eine detaillierte Ansicht auf den ausgewählten Patienten, mit der Möglichkeit, direkt von hier aus ein Rezept auszustellen.



Für diesen Button wird die Klasse „PrescriptionCreateFragment“ im Package „prescription“ genutzt. Hier kann man durch Ausfüllen des Formulars ein Rezept erstellen.

Patients

Patient

prescribed_drug

Diagnosed disease

CREATE_PRESCRIPTION

Zusätzlich wird noch eine Klasse „PatientAddFragment“ erstellt, welche ein vollständiges Formular zum Anlegen eines neuen Patienten zur Verfügung stellt.

← Add Patient

Name

Firstname Lastname

Birthdate

Gender

☒ male ☐ female ☐ other

Health Insurance

health_insurance_name

health_insurance_number

Address

Postcode

City

Street

Number

CREATE_PATIENT

- PRESCRIPTIONS:

Auch aus Mitarbeitersicht existiert das Fragment „PrescriptionFragment“, in welchem alle ausgestellten Rezepte des Mitarbeiters angezeigt werden.

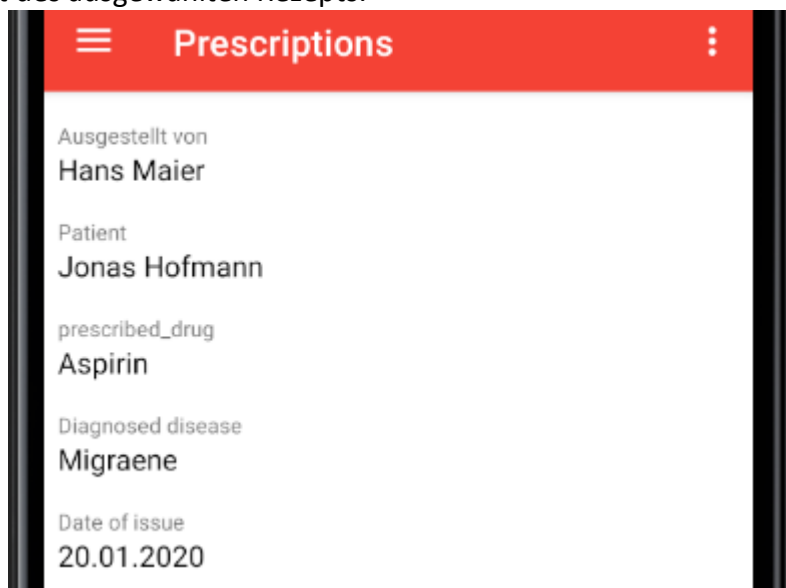
≡ Prescriptions

Antibiotika(Martin Ritter) - Mon Jan 20 09:15:00 GMT 2020

Aspirin(Jonas Hofmann) - Mon Jan 20 08:35:00 GMT 2020

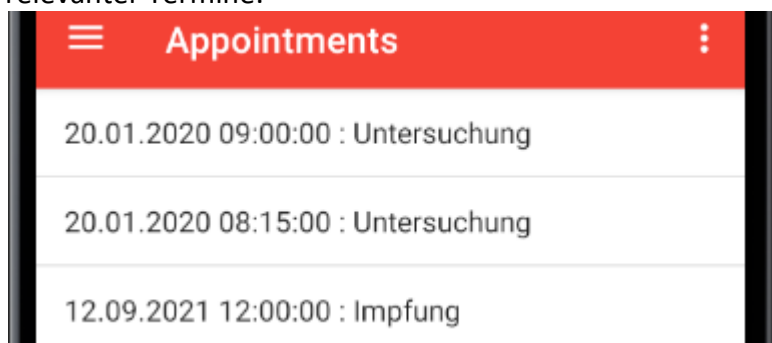
Antibiotika(Martin Ritter) - Mon Jan 20 00:00:00 GMT 2020

Auch hier wird die Klasse „PrescriptionDetailFragment“ verwendet für eine detaillierte Ansicht des ausgewählten Rezepts.

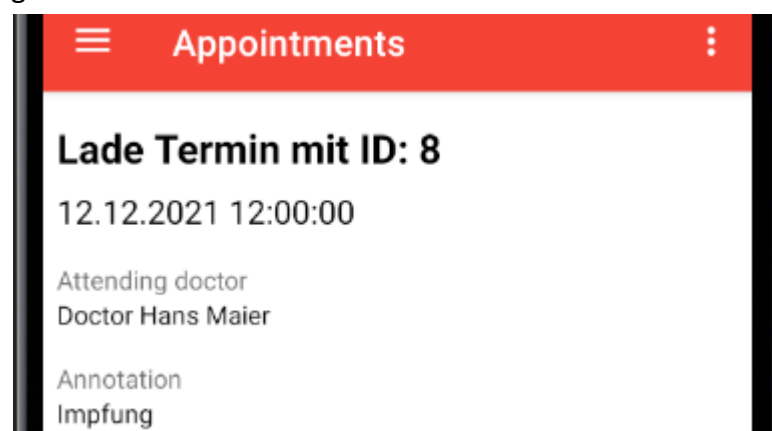


- APPOINTMENTS:

Auch der Mitarbeiter enthält die Klasse „AppointmentFragment“ mit einer Auflistung aller für ihn relevanter Termine.

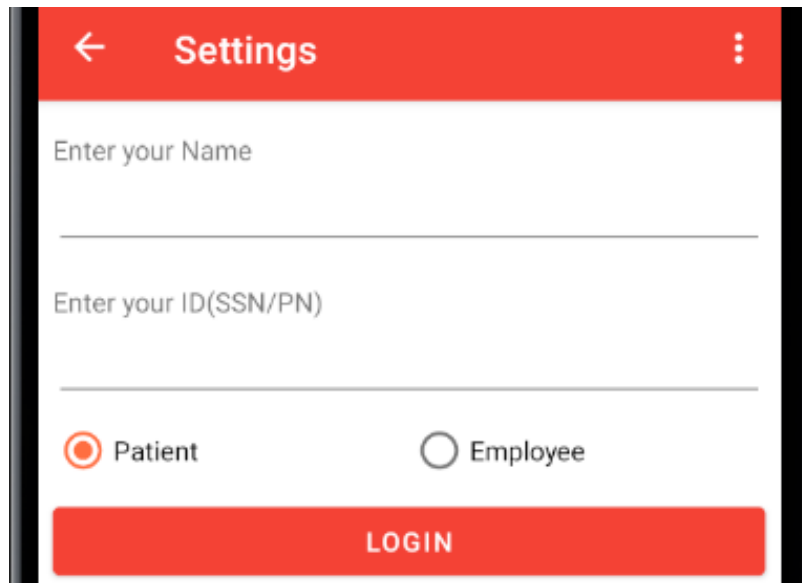


Mit dem „AppointmentDetailFragment“ wird eine detaillierte Ansicht eines Termins angezeigt.



- SETTINGS / LOGIN:

Zusätzlich wird noch ein „SettingsFragment“ erstellt, welches sowohl für den Patientenbereich, als auch für den Mitarbeiterbereich einsehbar ist. Dieses Fenster erscheint sowohl beim Start der App, als auch bei Aufruf der Settings, um den Benutzer wechseln zu können.



WS (Webservice)

Das Package „ws“ enthält neben einer „NoSuchRowException“ und einer „IllegalCreateException“ eine Klasse „InfrastrukturWebservice“. In dieser findet die Anbindung zu dem Wildfly-Server statt. Über die URL <http://141.87.68.X:8080/BuildingREST2/rest/app> können wir uns mit dem Dynamic Web Project „BuildingREST2“ in Eclipse verbinden und somit auf die Datenbank zugreifen.

Hier werden zusätzlich alle benötigten Requests (wie beispielsweise „public Person getPerson(long id) throws NoSuchRowException“) mit Hilfe der expliziten URL (In diesem Beispiel: oben genannte URL + „/persons/“ + id) durchgeführt.





MainActivity

Unsere App besteht aus nur einer Activity ohne Subactivities. Die MainActivity besteht aus der Konfiguration der App-Bar, welche in jedem Fragment sichtbar ist.

Res

Drawable

Im Package „drawable“ befinden sich alle benötigten Icons und Shapes.

- „ic_menu_home.xml“ und „ic_menu_adminhome.xml“ beschreiben das Home-Icon 
- „ic_menu_patient.xml“ beschreibt das Patient-Icon 
- „ic_menu_appointment.xml“ beschreibt das Appointment-Icon 
- „ic_menu_prescription.xml“ beschreibt das Prescriptions-Icon 
- „customshape.xml“ wird für die listViews erstellt
- „side_nav_bar.xml“ wird für die Navigation-Bar erstellt
- „ic_launcher_background.xml“ und „ic_launcher_foreground.xml“ beschreiben das App-Icon



Layout

Im Package „layout“ werden die Layouts, also die sichtbare Oberfläche aller Komponenten der App gestaltet: Die Side-Bar („activity_main.xml“), die App-Bar („app_bar_main.xml“), die allgemeine Content-Ansicht („content_main.xml“), der Navigation-Header („nav_header_main.xml“), sowie alle weiteren erstellten Fragmente („fragment_home.xml“, „fragment_patient.xml“, ...).

Menu

Im Package „menu“ befinden sich 2 XML-Dateien. Die „activity_main_drawer.xml“ definiert ID, Icon und Titel der einzelnen Side-Bar-Items.

Die „main.xml“ definiert ID und Titel des Settings-Items an der App-Bar.

Navigation

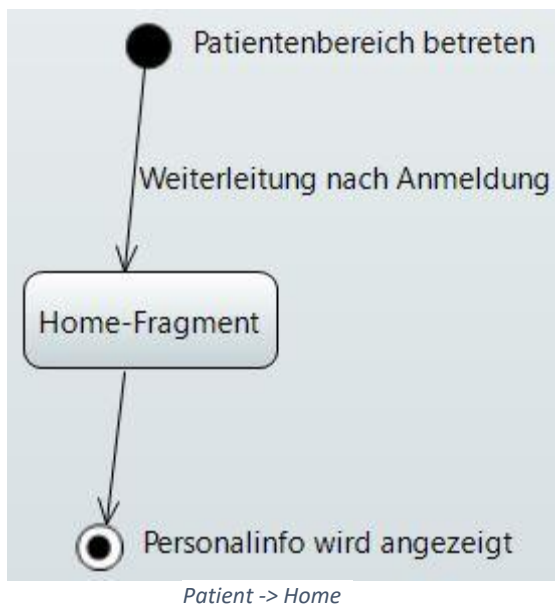
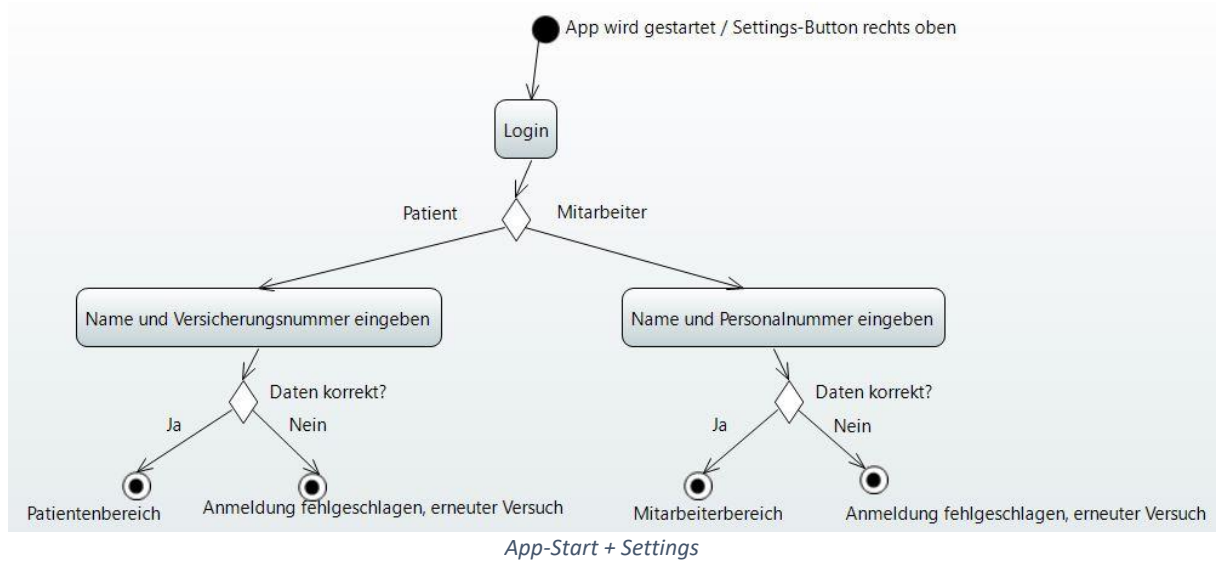
Im Package „navigation“ befindet sich eine „mobile_navigation.xml“, in welcher ID, Name, Label und Layout der Side-Bar-Fragmente definiert werden.

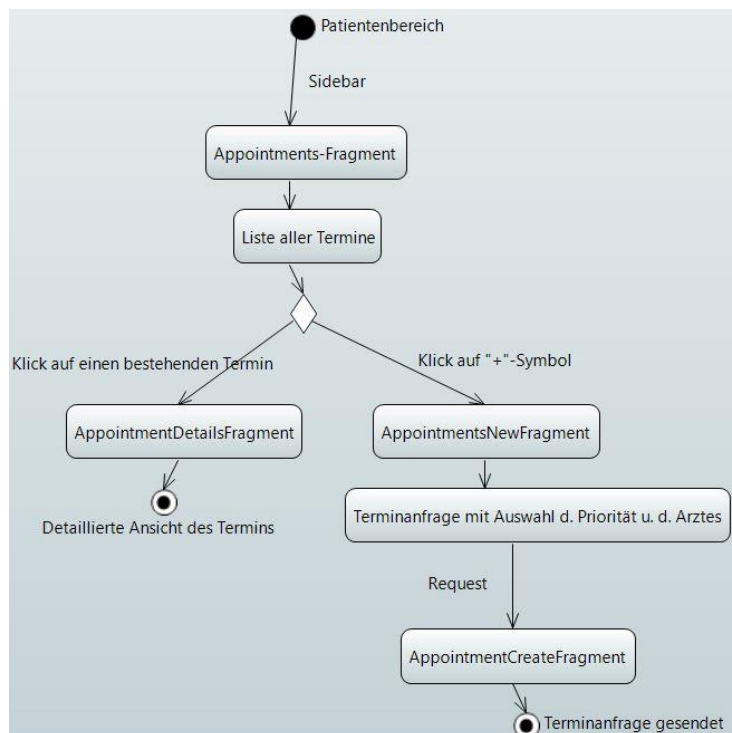
Values

Das Package values enthält mehrere XML-Dateien:

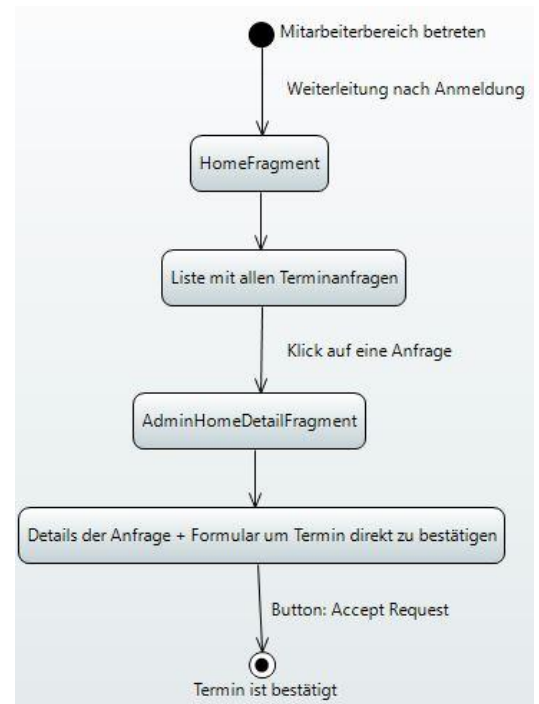
- In „colors.xml“ werden die Farben definiert
- In „dimens.xml“ werden die standardmäßigen Bildschirmränder genutzt
- In „strings.xml“ werden alle auf der App-Oberfläche sichtbaren Strings definiert (Standardmäßig auf Englisch und zusätzlich auf Deutsch)

UML-Aktivitätsdiagramm

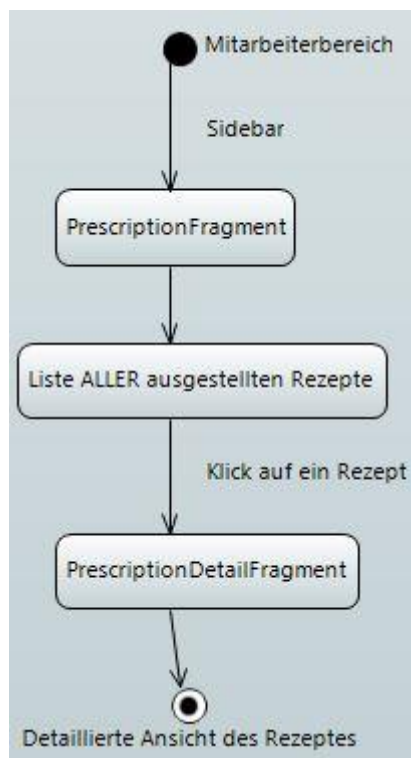




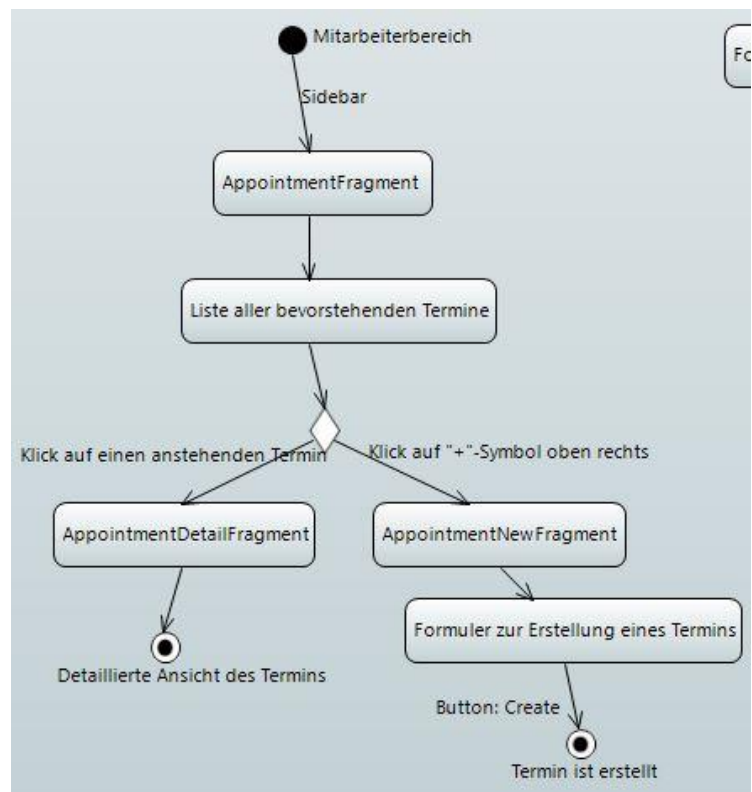
Patient -> Appointments



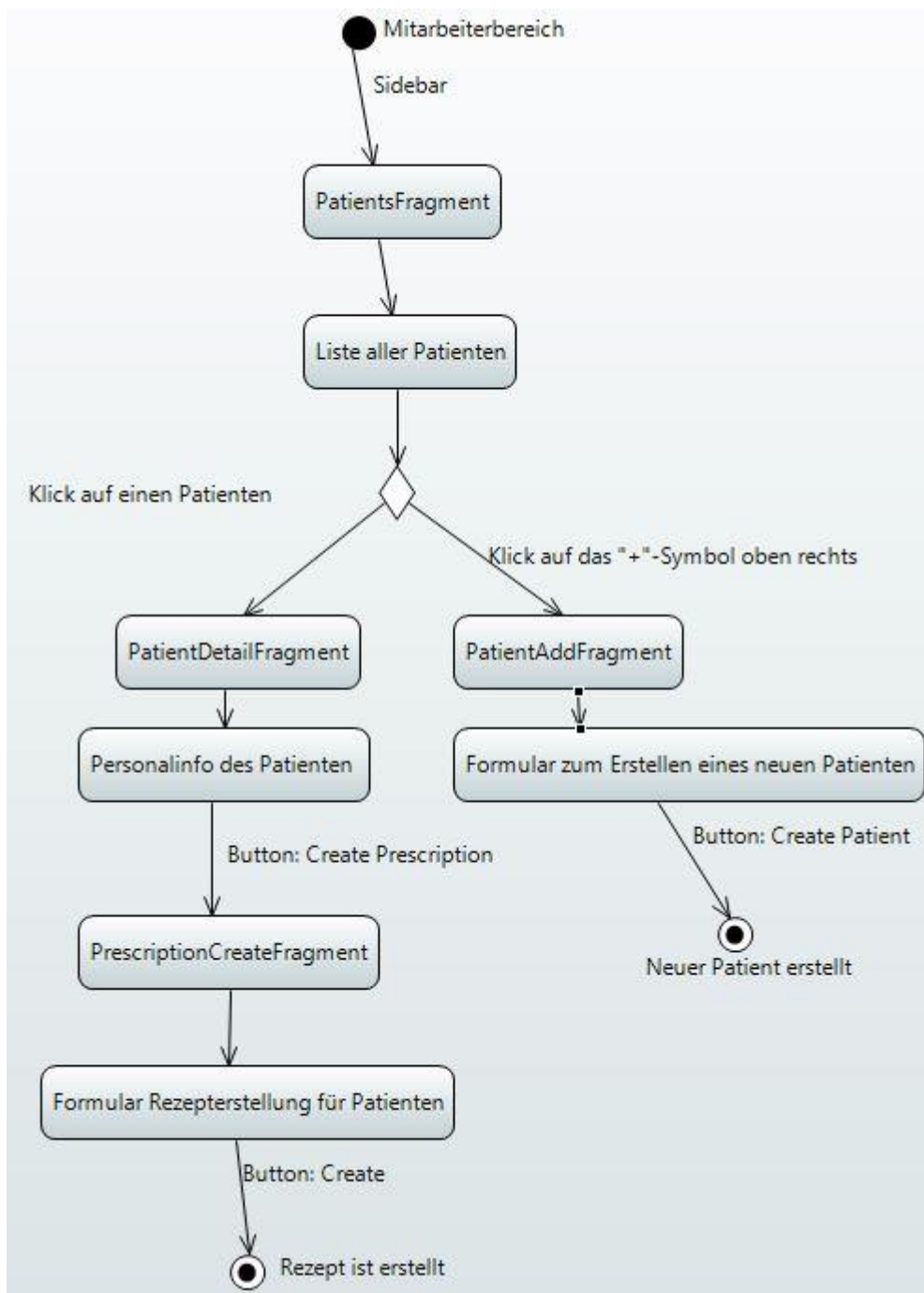
Mitarbeiter -> Home



Mitarbeiter -> Prescription



Mitarbeiter -> Appointment



Mitarbeiter -> Patients

UML-Klassendiagramm

