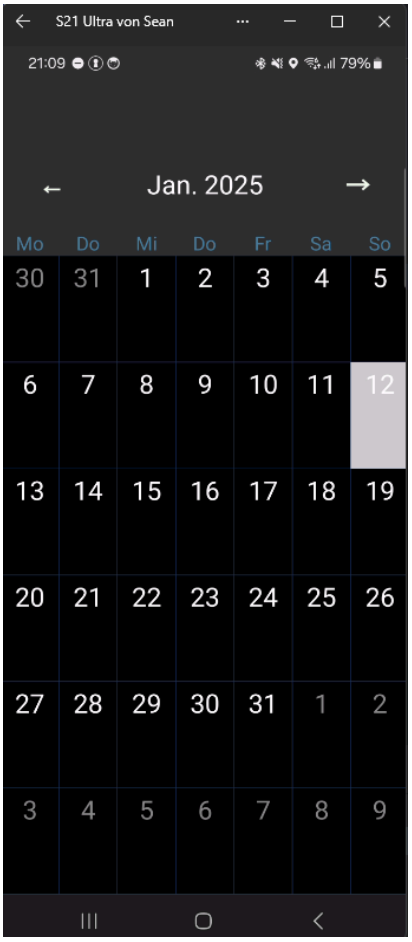
Sean Bien (589477)

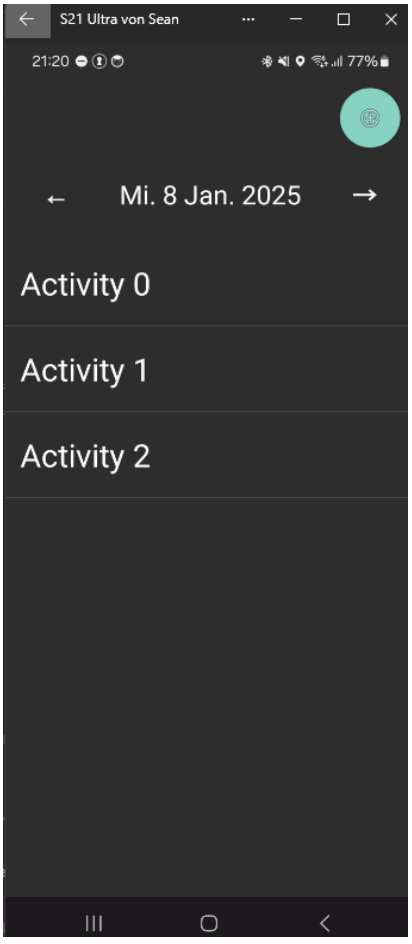
Kalender App

Beschreibung

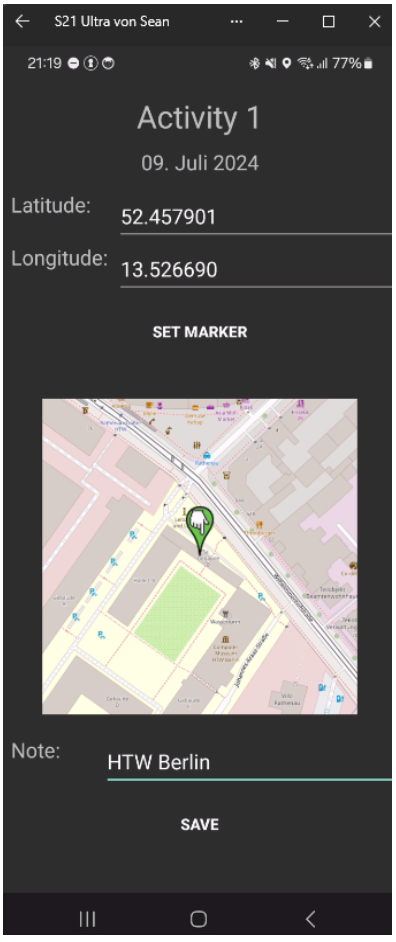
Bei der Applikation handelt es sich um einen Kalender, in dem man Aktivitäten für die jeweiligen Tage erstellen, anzeigen, bearbeiten und löschen kann.
Die Besonderheit ist das Einbinden von einer Karte, welche es dem User zulässt, einen Standort über die Latitude und Longitude anzugeben.



CalenderView

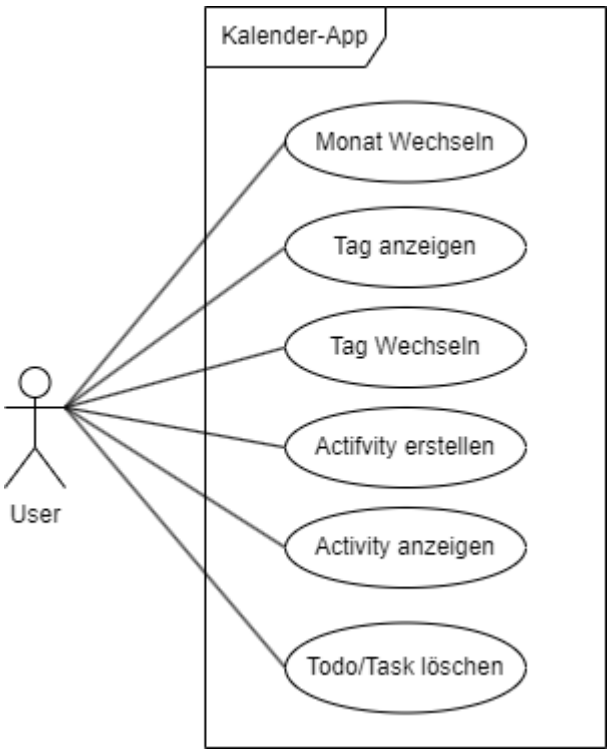


DayView



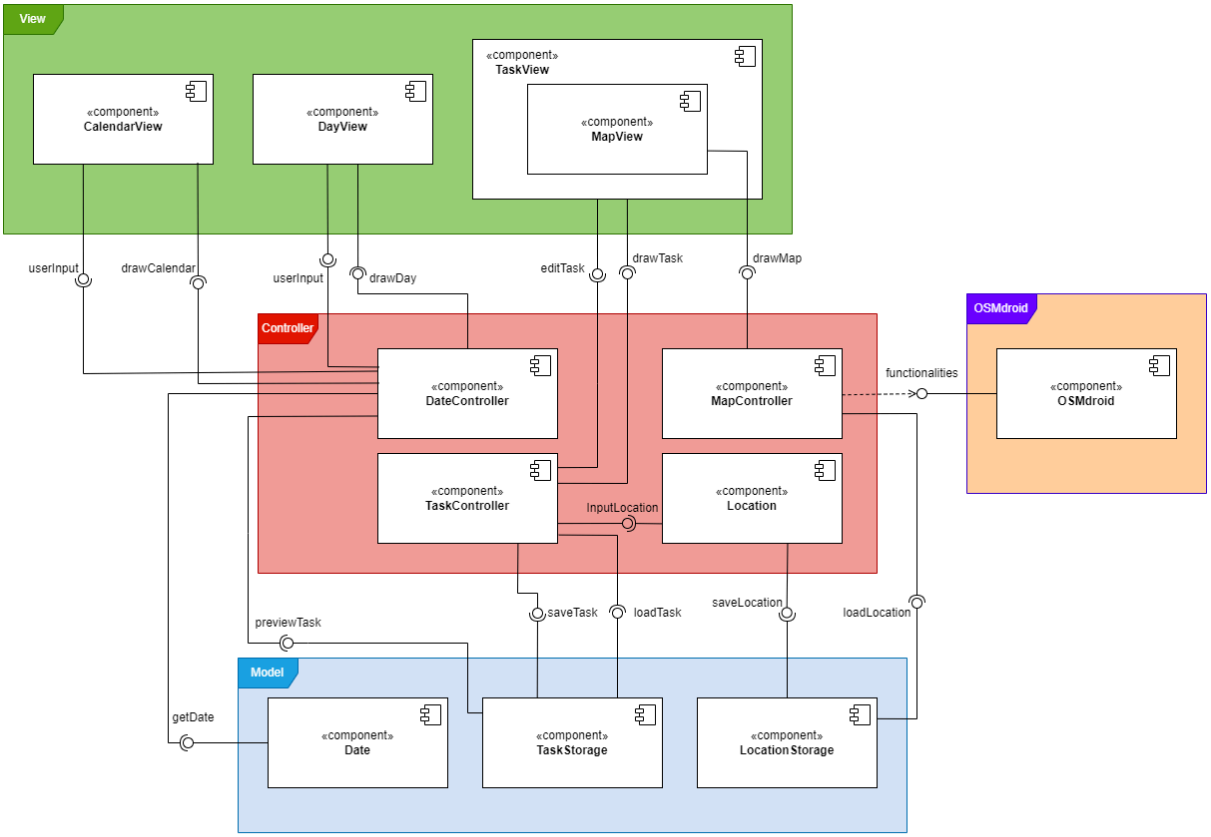
ActivityView

Use Cases



Für den User gibt es folgende Use-Cases:
Beim Öffnen der App gelangt man in die CalenderView, welche den aktuellen Monat anzeigt und den Tag hervorhebt. Hier kann der User durch Pfeil-Buttons die Monate wechseln (Vorheriger ‘←’, nächster ‘→’).
Beim Klick auf einen Tag öffnet sich die DayView, welcher den aktuellen Tag angezeigt, sowie eine Liste an angelegten Aktivitäten.
Durch einen Klick auf das “+” kann der User eine neue Aktivität erstellen. Der User gelangt nun in die ActivityView, wo er die Daten der jeweiligen Aktivität eingeben kann und durch das Drücken auf Save speichert. Das Verlassen der Ansicht passiert durch das Speichern oder durch die “Zurück”-Taste des Smartphones.
Auch kann der User in der DayView durch Klick auf eine Aktivität oder ToDo sich die jeweilige Task ansehen und ggf. bearbeiten.
Durch längeres Drücken auf eine Aktivität kann man eine Aktivität löschen.

Komponenten



Der Aufbau der Applikation basiert auf dem MVC (Model-View-Controller). Die jeweiligen Views haben einen entsprechenden Controller, welcher wiederum mit dem dazugehörigen Model verbunden ist.

View

In der View befinden sich die Komponenten, welche am Ende für den Nutzer als GUI sichtbar gemacht werden.
Hierzu gehört zum einen die *CalendarView*, welche den User den aktuellen Monat anzeigt und das aktuelle Datum farblich hervorhebt. Von hier gelangt der User durch den Klick von Pfeil-Buttons in einen anderen Monat oder durch Klick auf einen Tag in die *DayView*.
In der DayView werden alle Aktivitäten des jeweiligen Tages angezeigt. Beim Klick der jeweiligen Aktivität geht es in die *ActivityView*.
In der *ActivityView* werden die Details der jeweiligen Aktivität angezeigt, sowie eine Karte, welche von der *MapView* kommt. Die Karte soll zur Orientierung des Standortes dienen, indem der Standort mit einem Marker versehen ist, sodass man sieht, wo er sich befindet.

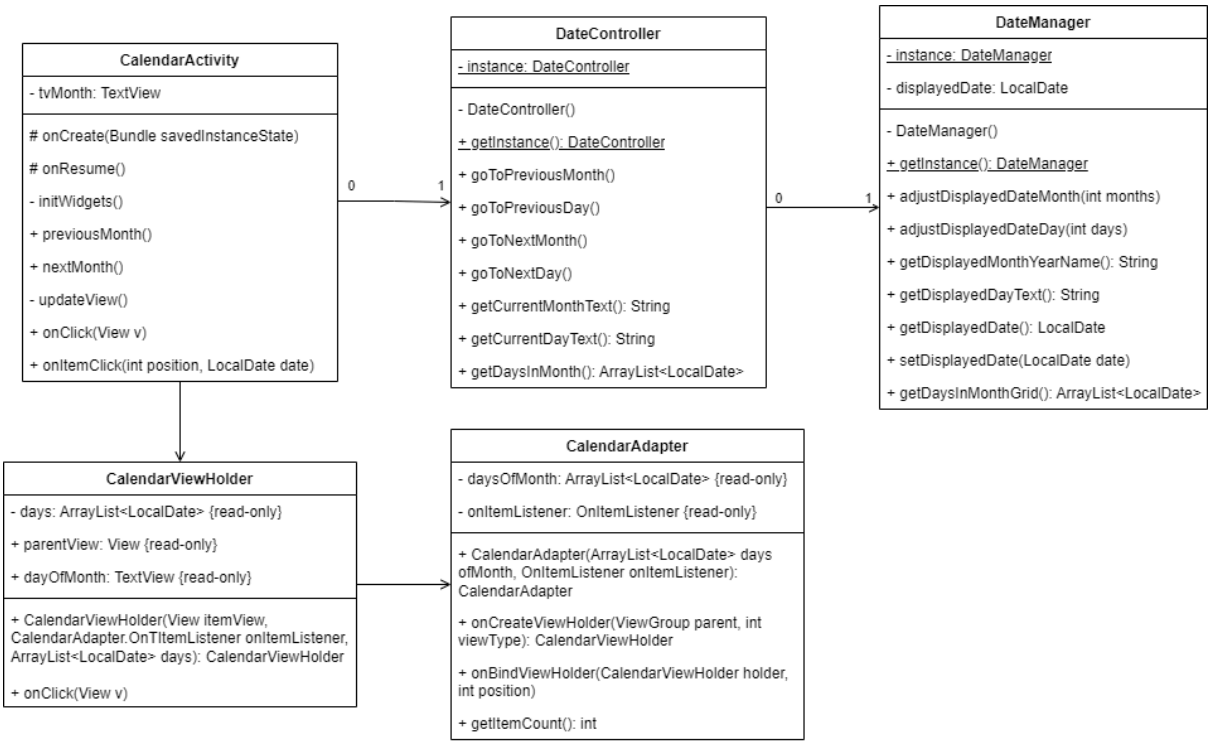
Controller

Die Controller dienen zur Informationsweitergabe der User-Inputs aus den Views, sowie die daraus resultierenden Ergebnisse (z.B. Wechsel der View, Abfragen etc.).
Hier sind zum einen die *Date*-, *Day*- und *TaskController*, welche eine ähnliche Funktionalität beinhalten.
Besonders erwähnenswert sind der *MapController* und *Location*; *Location* bekommt vom *TaskController* den Latitude und Longitude Input des Users und verarbeitet ihn so, dass dieser vom *MapController*, welche Funktionalitäten von *OSMdroid* implementiert, verarbeitet und der *TaskView* übermitteln kann.

Model

Das Model dient vor allem zur Speicherung, sowie Verarbeitung/ Berechnung des überwiegender Weise vom User bekommenen Inputs.
TaskStorage und *LocationStorage* dient vor allem zur Speicherung der Daten.
Während *Date* Berechnungen für den Kalender durchführt. Sprich er sorgt für die richtige Anzeige des Datums, welche er vom Betriebssystem übermittelt bekommt.

Klassendiagramm von Calendar



Dies dient zur Veranschaulichung des umgesetzten MVC im Fall von Calendar.
CalendarActivity dient hier als View und ist für den User-Input, sowie für die graphische ausgabe verantwortlich. *DateController* ändert entweder Werte aus dem *DateManager* und/oder gibt Daten zurück. *DateManager* ist hier für die Berechnungen bei User-Input zuständig, speichert aber auch den Wert des angezeigten Datums.
calendarViewHolder und *CalendarAdapter* ist für die zur Laufzeit erstellten Kalenders notwendig.

Tests

Model

Da die Funktionalitäten isoliert funktionieren sollen, werden die einzelnen Methoden des Models mit **Unit-Tests (JUnit)** durchgeführt. Vereinzelt werden auch manuelle Tests durchgeführt, da es sich hier zum großen Teil um die Speicherung von Daten handelt.

Date:

Hier geht es vor allem darum, dass die richtigen Tage eines Monats angezeigt werden. Dies habe ich punktuell mit JUnit-Tests überprüft.

```java
@Test
public void getDaysInMonthGridTestWithJan25() {
    DateManager dateManager = DateManager.getInstance();

    LocalDate firstJan = LocalDate.of( year: 2025,  month: 1,  dayOfMonth: 1);

    dateManager.setDisplayedDate(firstJan);
    ArrayList<LocalDate> result = dateManager.getDaysInMonthGrid();

    // Check if there are 42 Date Elements in result [6x7 Grid]
    assertEquals( expected: 42, result.size());

    // Check if result hast 31 Days from January 2025
    assertEquals( expected: 31, result.stream().
            filter(x -> x.getYear() == 2025 && x.getMonth() == Month.JANUARY).count());

    // Check if first of January 20205 is Wednesday
    assertEquals(firstJan, result.get(2));
}
```

Punktuell habe ich diese Funktion nur geprüft, da ich es als sinnlos finde, da die Daten der Tage aus der Java-Bibliothek kommen. Lediglich ist hier wichtig, dass die Daten richtig gespeichert werden, sodass sie auch korrekt angezeigt werden können. Auch habe ich hier überprüft, ob die Monate korrekt wechseln und der angezeigte Tag aktualisiert wird.

LocationStorage:

Hier habe ich manuell getestet, ob eine Location, welche ich gespeichert habe, dann auch beim Öffnen noch immer vorhanden ist.
Auch wurde über einen JUnit-Test getestet, ob die Werte für Longitude und Latitude gespeichert wurden.

TaskStorage:

Hier geht es vor allem darum, dass die Aktivitäten angelegt und abgespeichert sind. Dies habe ich auch wieder zum einen Manuell überprüft, zum anderen, aber ob die Werte wirklich abgespeichert wurden (Durch JUnit-Tests).

Controller

Controller habe ich nicht gesondert getestet, da sie bei mir nur für die Übermittlung von Daten zwischen dem Model und der View fungieren und z.B. keine eigenen Checks macht. Sprich die Controller werden schon bei den EspressoTests mitgetestet.

Da die Funktionalitäten von OSMdroid selbst getestet werden und ich davon ausgehen kann, dass diese auch funktionieren, werden diese Funktionalitäten nicht von mir getestet.

View

Diese werden vorrangig mit **Espresso** getestet. Hier wird vor allem geschaut, ob die Funktionalität der einzelnen Komponenten richtig funktioniert. Also ob z.B. ein Button das richtige Fenster öffnet etc.

DayView:

Hier habe ich mit Espresso getestet,

- ob man die Tage wechseln kann und ob sich dann die View aktualisiert
- ob die richtigen Aktivitäten für den richtigen Tag angezeigt werden (Dieser Test schlägt leider noch fehl, da die funktionalität nicht richtig funktioniert)
- Dass die *TaskActivity* beim klicken auf eine Task geöffnet wird.
- Dass bei längerem drücken eine Aktivität gelöscht wird
- Dass eine neue Task angelegt werden kann und diese nach dem Speichern angezeigt wird.
- Das ich mit der “Zurück”-Taste des Smartphones wieder in die *CalendarActivity* gelange

TaskView:

Hier habe ich folgendes getestet:

- Speichern von Aktivitäten und Überprüfung auf richtiges abspeichern
- Können Werte eingegeben werden (**Manueller Test**)
- Marker Hinzufügen (Überprüfung auf richtige Longitude und Latitude Wert [**Fehlerfälle**, wenn z.B. Latitude über 90 ist, aber auch Grenzfälle von Longitude und Latitude, also z.B. bei Longitude -180 funktioniert, aber -180.1 nicht).
  Ob der Marker dann angezeigt wird, habe ich manuell geprüft.
- Das ich mit der “Zurück”-Taste des Smartphones wieder in die *DayActivity* gelange

MapView:

Hier prüfe ich nur manuell, ob wirklich eine Karte angezeigt wird. Und ob die Funktionalitäten, welche ich möchte (Navigation mit Fingern, Zoom), auch funktionieren und ob ein Marker angezeigt wird.

Sean Bien (589477)

CalendarView:

```java
@Test
public void testNextMonthButtonUpdatesView() {
    // Click the next month button
    onView(withId(R.id.button_next_month)).perform(click());

    // Check that the tvMonth text has changed
    onView(withId(R.id.tv_month)).check(matches(not(withText(initialMonth))));
}

@Test
public void testDayClickOpensDayActivity() {
    // Set up an intent matcher
    Intents.init();

    // Click the first day in the RecyclerView
    onView(withId(R.id.rv_calendar))
            .perform(RecyclerViewActions.actionOnItemAtPosition( position: 0, click()));

    // Verify that the DayActivity is started
    intended(hasComponent(DayActivity.class.getName()));

    Intents.release();
}
```

einige Espresso Tests der Klasse *CalendarActivity*

- Überprüft wird, ob der Monat gewechselt wird und die view sich aktualisiert
- Wird beim Klicken eines Tages zur *DayActivity* gewechselt?