Hochschule Worms
Fachbereich Informatik
Studiengang Angewandte Informatik

Modul

"ENTWICKLUNG MOBILER ANWENDUNGEN" IM SOMMERSEMESTER 2020

Mobile Anwendung: Flashcards

Technische Dokumentation

Vorgelegt von:

Siawasch Sarmadi [Matrikel-Nr. 673944]

Robin Eschbach [Matrikel-Nr. 674705]

Ort, Abgabetermin

Worms, 31.08.2020

Vorgelegt bei: Stephan Kurpjuweit

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Glossar

Begriff	Bedeutung
App	Applikation
POJO	Plain Old Java Object Eine Klasse die abgetrennt von jeglichen Konventionen. Die mit der Datenbank in interagiert.
Dark Mode	Ein alternatives Farbschema, das sich auf dunkle Farben beschränkt und damit, vor Allem in Falle einer Sehbehinderung, unterstützend und angenehmer wirken kann.
User Story	Stellt eine einzelne Funktion der Software dar.
View Hierarchy	Die Relation zwischen den verschiedenen Elementen.
JVM	Java Virtual Machine.
XML	Extensible Markup Language.

Info:

Personas: (Bereits in der Designdokumentation)

Designentscheidungen: (Befinden sich in der Designdokumentation)

GitHub: https://github.com/MaxMad187/EMA_Flashcards

JavaDoc der Datenbank: https://0x5245.de/flashcards

2 Developer Guide

1. Programmiersprachen
    - **Kotlin** ist eine Programmiersprache, die hauptsächlich (aber nicht ausschließlich) auf der JVM ausgeführt wird und die folgenden wesentlichen Eigenschaften aufweist:
        - ausdruckstark (weniger Boilerplate-Code im Vergleich zu Java)
        - statisch typisiert (Datentyp ist zur Kompilierzeit bekannt)
        - objekt-orientiert
        - pragmatisch ausgelegt (Ideal zur "Lösung" alltäglicher Problemstellungen)
        - volle Interoperabilität mit Java Code (Java und Kotlin können kombiniert genutzt werden)
        - überall einsetzbar, wo momentan auch Java eingesetzt wird

      open source und unter Apache 2 Lizenz; gehostet auf GitHub
    - **XML**
        - Es ermöglicht die hierarchisch strukturierte Anzeige von Textdateien. Ein Merkmal der XML ist, dass die Namen von Strukturelementen in einer XML-Anwendung frei ausgewählt werden können.
2. Verwendete Zusatzbibliotheken App-Modul (app)
    - ConstraintLayout
        - androidx.constraintlayout:constraintlayout:1.1.3

      Umsetzung der Designs als XML Layout
    - Android Jetpack Navigation Component
        - androidx.navigation:navigation-fragment:2.3.0
        - androidx.navigation:navigation-ui:2.3.0
        - androidx.navigation:navigation-fragment-ktx:2.3.0
        - androidx.navigation:navigation-ui-ktx:2.3.0

      Umsetzung der Navigation
    - AdapterDelegates
        - com.hannesdorfmann:adapterdelegates4:4.2.0

      Bibliothek zur einfachen Darstellung von Listen mit hoher Erweiterbarkeit, z.B. um zukünftig weitere Kartentypen darstellen zu können.
    - Material Components
        - com.google.android.material:material:1.3.0-alpha02

      Komponenten zur Umsetzung des Material Design, z.B. Darstellung der Navigation, BottomAppBar, CardViews
3. Verwendete Zusatzbibliotheken Datenbank-Modul (database)

- Room Persistence Library
  - androidx.room:room-runtime:2.2.5
  - androidx.room:room-compiler:2.2.5
  - androidx.room:room-ktx:2.2.5
  - androidx.room:room-testing:2.2.5

  Bibliothek für eine einfache und für Android optimierte Verwendung einer SQLite Datenbank

4. Allgemeine Aufgaben
   - Event Handling
     - Eventgesteuerte Programmierung (Einstieg per Event)
   - Transaction-handling
     - Da nur einzelnen Änderungen in der Datenbank möglich sind, werden keine Transactions benötigt
   - Fehlermeldung
     - Meldung per Toast (Beim Erstellen der Karte falls front oder back leer ist)
   - Lokalisierung
     - Externalisierung aller Strings in einheitlich englischer Sprache
   - Benutzer/ Rechteverwaltung
     - Applikation benötigt keine Rechte
5. Funktionen basierend auf User Stories

   Funktionale Anforderungen.
   - Als Benutzer möchte ich Kartendecks erstellen, um ein bestimmtes Fach lernen zu können.
   - Als Benutzer möchte ich Kartendecks löschen können, um bereits abgeschlossene Fächer wieder zu entfernen.
   - Als Benutzer möchte ich einen Nachtmodus haben, um die Augen beim längeren Lernen zu schonen.
   - Als Benutzer möchte ich Karten überarbeiten können, um fehlerhafte Karten zu erneuern.
   - Als Benutzer möchte ich wissen, wie viele Karten sich in einem Kartendeck befinden, um zu wissen, wie umfangreich die Liste ist.
   - Als Benutzer möchte ich nach bestimmten Karten suchen, um sie anpassen zu können.
   - Als Benutzer möchte ich eine Übersicht über alle Kartendecks haben, um den Überblick zu bewahren.
   - Als Benutzer möchte ich nachträglich Karten eines Kartendecks erweitern können, um neue Materialien ergänzen zu können.

- Als Benutzer möchte ich nach Stichwörtern suchen können, um eine Auflistung eines Themas zu erhalten.

Nicht-Funktionale Anforderungen

- Als Benutzer möchte ich Karten nach einer bestimmten Zeit kategorisiert kriegen, damit ich automatisch an Karten erinnert werde, die ich schon länger nicht mehr angezeigt bekommen habe.
- Als Benutzer möchte ich einen Nachtmodus haben, um die Augen beim längeren Lernen zu schonen.

3 Besonderheiten in der Entwicklung

Ziel war es, Software zu entwickeln, die auf den neuesten Standards basiert und alle erforderlichen Qualitätsmerkmale aufweist.

- Behindertengerecht
    - Content descriptions, Dark Mode
- Responsiv auch auf sehr kleinen Geräten
    - Responsive durch flexible Constraints in den ConstraintLayouts
- An optisch relvanten Stellen speziell für Tablets angepasste Größen und Abstände (mit Hilfe von “dimens”, gesteuert über Resource Qualifier)
- Neueste Designinnovationen
- Performance schonend durch flache View Hierarchy mit Hilfe von ConstraintLayout
- Robustheit und Verständlichkeit durch qualitativ hochwertigen Code
- Gute Wiedererkennbarkeit für eine schnelle Eingewöhnung
- Einfache und gut verständliche Screens
- Dark Mode
- Eigenes Logo
- Beispielsweise können auch arabische, chinesische, japanische und koreanische Texte verwendet werden, da die Richtung der Texte berücksichtigt wird.
- Eine Datenbank Struktur in dritter Normalform

4 Rahmenbedingungen

Die einzige allgemeine Entwicklungsbedingung war, dass die Anwendung nativ sein musste. Dies wurde mit Android Studio implementiert. Ansonsten haben wir keine Bedingungen festgelegt oder uns eingeschränkt.

5 Risiken

Das größere Entwicklungsrisiko besteht darin, dass wir alle zusätzlich zu diesem Projekt andere Prüfungen und Projekte haben. Die Zeit, die dann dem Projekt gewidmet werden kann, ist leider in mehrere Projekte unterteilt. Ein weiteres Risiko bestand darin, dass die Kommunikation während der Corona-Zeit digital stattfinden musste.

6 Sichten auf die Architektur

Grundlegend ist Study Flashcards eine Single Activity App Die Architektur der App nutzt nur eine Activity und jedes Fragment stellt einen Screen dar.

6.1 Datenbank

Die Datenbank folgt einem einfachen Schema. Es gibt Flashcards (Karteikarten) und Repositories (Kartensätze). Diese stehen in einer Many-To-Many-Beziehung zueinander. Infolgedessen kann ein Kartensatz mehrere Karteikarten enthalten, eine Karteikarte kann jedoch auch in mehrere Kartensätze integriert werden. Aus Gründen der Skalierbarkeit wurde bewusst eine Many-To-Many-Beziehung anstelle einer One-To-Many-Beziehung ausgewählt.

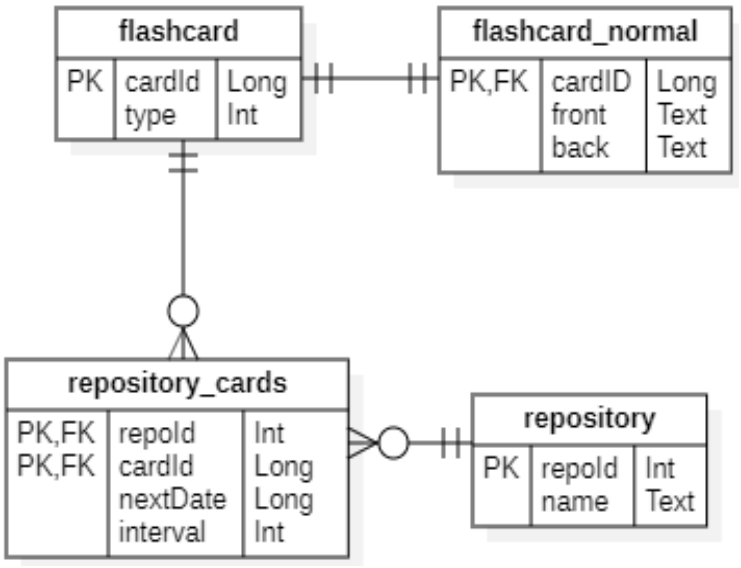

Abbildung 1 : ER-Modell

Auf diese Weise können neue Kartensätze erstellt werden, die alle Karten der alten enthalten, ohne dass die Karten kopiert werden müssen. Dies wäre in einem Prüfmodus oder ähnlichen Modi nützlich.

Im Hinblick auf die Erweiterbarkeit wurde die Lernkarte in Typen unterteilt. Der einzige Typ im Moment ist der normale Typ, bei dem eine Karteikarte eine vordere und eine hintere Abdeckung hat. Diese Entscheidung erleichtert das Hinzufügen neuer Typen, z. B. Multiple Choice.

Für die Benutzerfreundlichkeit des Datenbankmoduls existiert auch noch ein POJO um ein Kartenset mit allen Karten die zu diesem gehören, darzustellen.

6.2 Navigation

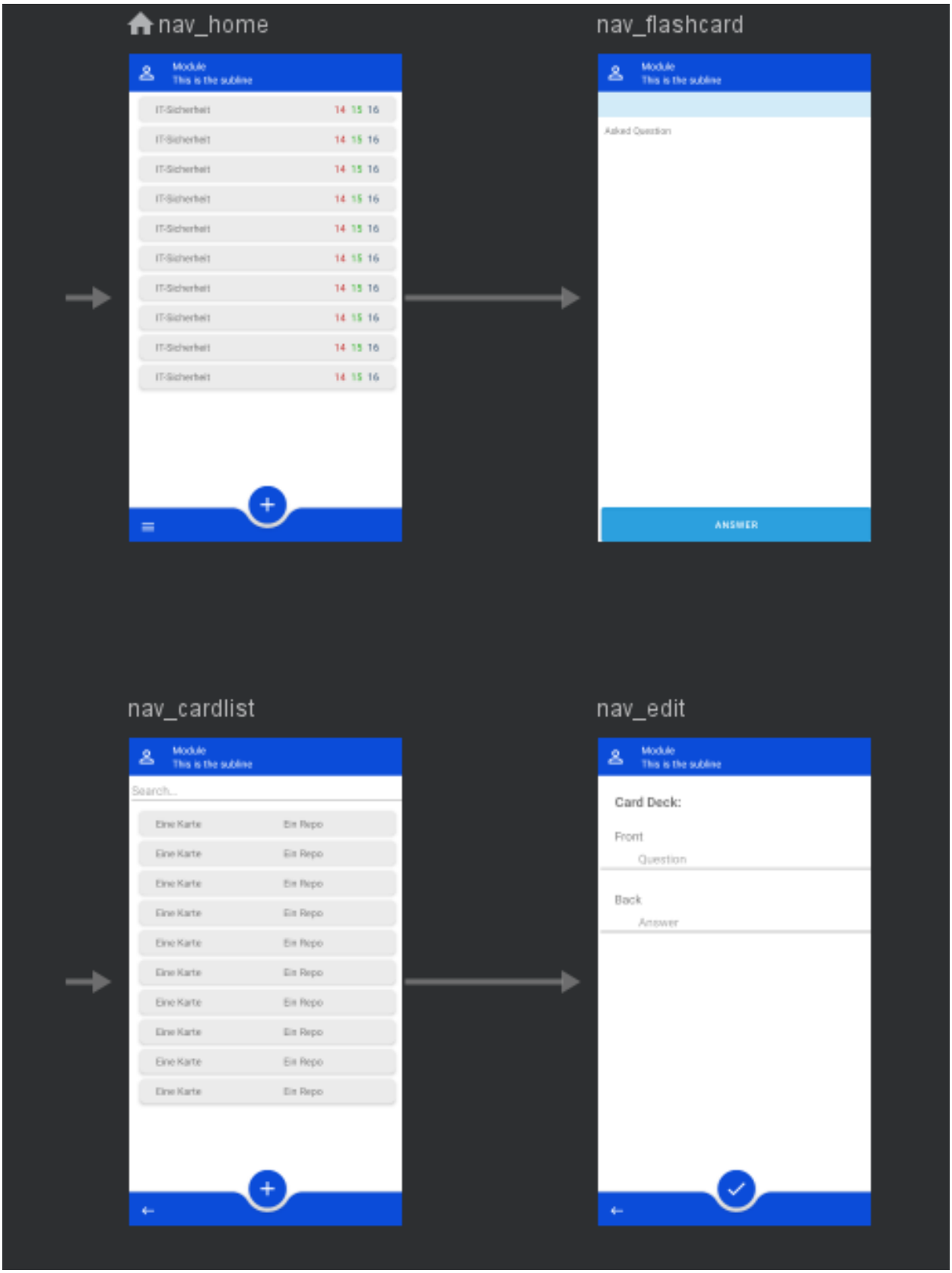

Abbildung 2 : Navigationsdiagramm

Die Navigation ist an sich ziemlich simpel gehalten.
Das Navigationsmenu bietet dem Benutzer die Möglichkeit entweder zu *nav_home* (Home Fragment) oder zu *nav*_cardlist (CardList Fragment) zu navigieren. Da das Navigationsmenu von überall erreichbar ist, sind auch diese Fragmente von überall erreichbar. Deswegen existiert ein globaler Navigationsknoten für die beiden.
Von *nav_home* kann der Benutzer durch Auswählen einen Kartensatzes zu *nav_flashcard* (Flashcard Fragment) steuern. Dieser Kartensatz wird dann als Argument an das Fragment übergeben.

Für die Navigiation von *nav_cardlist* zu *nav_edit* (Edit Fragment) bietet die Applikation zwei Möglichkeiten. Entweder der Benutzer erstellt eine neue Karteikarte, dabei wird dann null übergeben, oder der Benutzer wählt eine Karteikarte aus und möchte diese Bearbeiten. Dann wird die Karteikarte an *nav_edit* übergeben.

6.3 Aktivitätsdiagramm der Applikation

Die Applikation startet mit dem Home-Fragment. Von dort hat man verschiedene Aktionsoptionen. Solange man im Home-Fragment bleiben möchten, wird einem die Möglichkeit geboten mit der Liste der Repositories (Kartensätze) interagieren. Dabei ist es möglich neue Kartensätze zu erstellen oder zu importieren und vorhandene umzubenennen, zu löschen, zurückzusetzen oder zu exportieren. Beim Erstellen und Umbenennen ist ein Name erforderlich, den der Benutzer eingeben muss und beim Importieren und Exportieren muss der Benutzer einen Dateipfad auswählen. Alle diese Aktionen müssen entsprechend bestätigt werden.

Wenn ein Kartensatz ausgewählt ist, kann dies auch durch Senden an das Karteikarte gelernt werden. Die Karten werden nacheinander angezeigt. Die Antwort kann gesehen werde, nachdem die Vorderseite betrachtet wurde. Anschließen können aus den 3 verfügbaren Optionen eine Zeit ausgewählt werden, zu der die Karte erneut angezeigt wird. Falls bisher noch Karten vorhanden sind, wird der Vorgang wiederholt, und wenn nicht, wird man zum Home-Fragment zurückgeleitet.

Die Navigation kann wie jedes andere Fragment vom Home-Fragment aus geöffnet werden. In der Navigation besitzt man die Möglichkeit den Dark Mode ein- und ausschalten und zwischen Kartenlistenfragment und dem Home-Fragment wählen. Wie das Home-Fragment

bietet das Card-List-Fragment mehrere Optionen. Der Benutzer kann nach Karten suchen, neue Karten erstellen und vorhandene Karten bearbeiten, zurücksetzen oder löschen. Das Zurücksetzen und Löschen muss bestätigt werden, während das Erstellen und Bearbeiten den Benutzer zum Edit-Fragment weiterleiten. Von hier aus kann der Benutzer entweder zum Kartenlistenfragment zurücknavigieren oder die aktuellen Änderungen speichern. Beim Speichern gibt der Benutzer den entsprechenden Kartensatz sowie die Vorder- und Rückseite an.

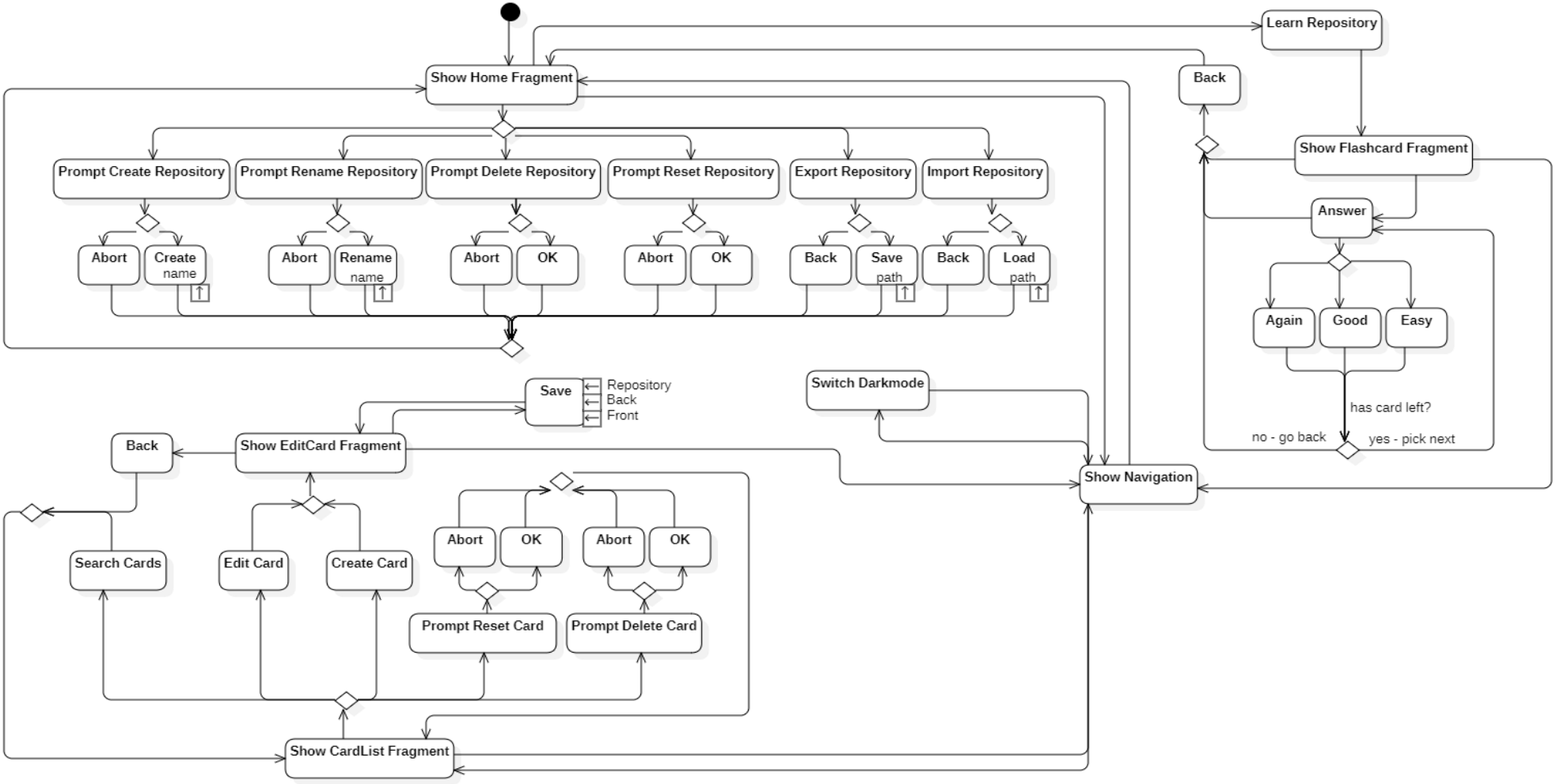

Abbildung 3 : Aktivitätsdiagramm

7 Zeitplan

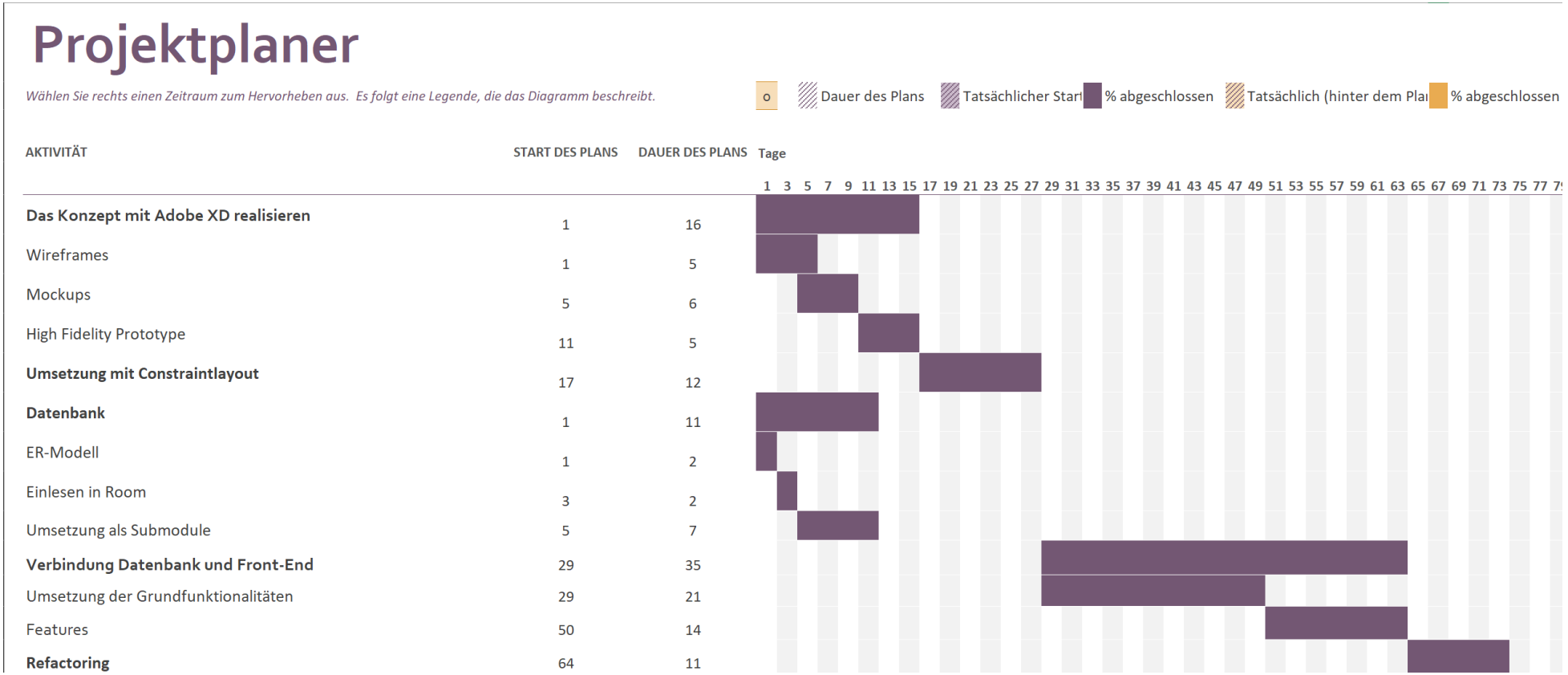

Abbildung 4 : Zeitplan