Musterprojekt zur Erstellung einer Android-App

Dieses Musterprojekt zeigt beispielhaft den Weg zu einer fertigen Android-App. Dabei werden nicht nur die nötigen Programmierschritte durchgeführt, sondern auch die Planungsüberlegungen und deren Ausführungen dokumentiert. Hierdurch erhält der Leser einen Eindruck, wie ein Projekt geplant und durchgeführt werden kann.

1 Installationsanleitung

Die Installation der Anwendung auf einem Smartphone oder Tablet kann mit Hilfe der beigelegten Datei `Lernkarten.apk` erfolgen. Hierzu muss diese auf das Gerät kopiert werden und von dort aus die Installation gestartet werden.

Alternativ dazu ist eine direkte Installation über eine USB-Verbindung möglich. Hierzu muss einerseits der entsprechende Gerätetreiber installiert sein und anderseits die SDK-Tools aus dem Android SDK[1]. Anschließend kann über das Terminal oder ein Command-Fenster mit dem Befehl

```
adb install Lernkarten.apk
```

die App auf dem Smartphone oder Tablet installiert werden.

Möchte man sich den Quellcode des Projektes näher ansehen, ist die Installation von Android Studio zu empfehlen. Dies ist die von Google präferierte Entwicklungsumgebung für Android-Apps. Diese kann ebenfalls unter dem angegebenen Link heruntergeladen werden. Für die Installation von Android Studio ist die Installation eines aktuellen Java Development Kit Voraussetzung.

2 Projektverzeichnis

Das Projektverzeichnis hat folgende Struktur und Inhalt:

- code: Der vollständige Quellcode der Android App. Das Projekt kann mit Android Studio geöffnet werden
- doc: Dokumente und Projektartefakte, die im Rahmen des Projektes erstellt wurden bzw. sie ergänzen:
  - Dieses Dokument (`musterprojekt.pdf`).
  - Die Projektbeschreibung des Auftraggebers (`projektbeschreibung.pdf`).
  - Das Exposé (`expose.pdf`).
  - Die Spezifikation, inkl. Domänenmodell, technischer Analyse und Anwendungsfällen (`spezifikation.pdf`).
  - Der Entwurf (`entwurf.pdf`).

[1] Download unter: `http://developer.android.com/sdk/index.html`

- apk:
  - Die lauffähige Anwendung.
  - Die Benutzerdokumentation (als Bestandteil der Anwendung).

Autor: Christian Heigele, Michael Lüttebrandt, Projektgruppe Softwaretechnik.

Projektbeschreibung

Der Verlag Lern & Erfolg GmbH ist ein aufstrebendes Unternehmen im Handel mit Lernkarten zur Vorbereitung auf Prüfungen in verschiedenen Bereichen. Da in den letzten Jahren mobile Plattformen immer mehr an Bedeutung gewonnen haben, möchten wir nun auch eine eigene App anbieten. Diese soll kostenlos zur Verfügung gestellt werden. Langfristig soll es die Möglichkeit geben, für einzelne Themenbereiche Lernkarten kostenpflichtig zu erwerben. Geräte mit dem Betriebssystem Android sind sehr verbreitet. Daher soll die App zunächst für diese Plattform entwickelt werden.

Da es in diesem Bereich bereits Lösungen gibt und der Verlag mit seinen qualitativ hochwertigen Lernkarten möglichst schnell auf den Markt kommen möchte, ist es wichtig, dieses Projekt kurzfristig zu realisieren. Daher wurde ein Zeithorizont von 3 Monaten geplant.

Nach den Vorstellungen von unserem Verlag sind die Eckdaten des Projektes folgende:

- Lernkarten können verwaltet werden.
- Karten können einem Thema zugeordnet werden.
- Eine Lernkarte kann sowohl Text als auch Bilder enthalten.
- Es wird eine Suche nach verschiedenen Themen angeboten.
- Lernkarten können zufällig oder nach einer bestimmten Reihenfolge gelernt werden.
- Eine Statistik soll dem Benutzer zeigen, welche Fortschritte gemacht wurden.
- Es soll die Möglichkeit bestehen, neue Karten von einem Server nachzuladen.
- Es sind Karten mit nur einer Seite möglich. Dies ist zum Beispiel beim Lernen von einzelnen Bildern sinnvoll.
- Antworten können aus einfachem Text bestehen. Es kann aber auch eine Multiple-Choice-Auswahl angezeigt werden.
- Um möglichst viele Geräte anzusprechen, soll es auch ein Layout für Nutzer von einem Tablet geben.
- Das Layout soll an die neusten Android-Designrichtlinien (Material-Design) angepasst sein.

Autor: Michael Lüttebrandt, Projektgruppe Softwaretechnik.

Exposé

1 Auftraggeber

Das Projekt wurde von der Lern & Erfolg GmbH in Auftrag gegeben. Die Lern & Erfolg GmbH ist einer der führenden Verlage im Handel mit Lernkarten und möchte nun seine Geschäfte auf mobile Geräte ausweiten. In diesem Zusammenhang soll eine App für Android herausgebracht werden.

2 Zeitraum

Für die Durchführung des Projektes stehen lediglich drei Monate zur Verfügung. Hierdurch soll schnellstmöglich der Vertrieb mit kostenpflichtigen Lernkarten über die App realisiert werden.

3 Beschreibung

Grundlage für diese Beschreibung ist die vom Auftraggeber erhaltene Projektbeschreibung, die seine Sichtweise mit den wichtigsten Funktionen der Anwendung zusammenfasst.

Das Projektziel ist eine mobile Anwendung mit der Lernkarten trainiert und verwaltet werden können. Neben der Auswahl von vorgefertigten Lernkarten zu verschiedenen Themen können auch eigene Karten vom Benutzer erstellt werden.

Um am Anfang möglichst viele Kunden zu binden, soll die App zunächst kostenlos mit bestimmten Lernpaketen erhältlich sein. Nach der Einführungsphase werden kostenpflichtige Themen folgen.

Eigene Lernkarten können ein- oder zweiseitig sein und neben einem Text auch Bilder enthalten.

Damit ein Benutzer seine Lernaktivitäten verfolgen kann, wird eine Statistik bereitgestellt.

Um Themen und Lernkarten leichter aufzufinden, wird eine Such- & Sortierfunktion angeboten.

Die Darstellung der Anwendung soll auch für größere Displays optimiert sein, sodass ein Benutzer auch ein Tablet zum Lernen verwenden kann.

4 Infrastruktur

Die Anwendung wird für Android entwickelt. Hierzu kommen das Android SDK mit einer SQLite Datenbank zum Einsatz. Für die Entwicklung wird Android Studio verwendet.

Spezifikation

Dieses Dokument beschreibt die Spezifikation des Lernkarten-Projektes. Zu Beginn wird die Anwendungsdomäne mit Hilfe eines Glossars, eines Datenmodells und einiger Beispieldaten beschrieben. Es folgen Skizzen (Mock-Ups) zu den einzelnen Screens und die Definition der einzelnen Anwendungsfälle. Das Dokument schließt mit der Erläuterung der technischen Aspekte ab.

1 Domänenanalyse

1.1 Beispiel

Um zu verdeutlichen, was sich hinter dem in der Domänenanalyse erstellten Datenmodell verbirgt, werden nun zwei Beispiele dargestellt. Diese sollen zeigen, wie ein Stapel, eine Lernkarte und die dazugehörigen Antworten aussehen und welchen Inhalt ihre Attribute haben können.

1.1.1 Beispiel 1: Testansätze

Stapel:

Bezeichnung	Softwaretechnik
Beschreibung	Lerninhalte zu Softwaretechnik

Tabelle 1: Beispieldaten Stapel - „Softwaretechnik“

Lernkarte:

Frage	Wie nennt man in der Softwaretechnik einen Test, bei dem nur eine Betrachtung von außen berücksichtigt wird und alle Kenntnisse über die genaue Implementierung vernachlässigt werden
Anzahl Richtig	1
Anzahl Falsch	3
Zuletzt Richtig	false
Zuletzt Gespielt	1462106700 (Unix Zeitstempel: 01.05.2016 12:45)

Tabelle 2: Beispieldaten Lernkarte - „Testansätze“

Antworten:

Antwort 1	Black-Box-Test	true
Antwort 2	White-Box-Test	false
Antwort 3	Green-Box-Test	false
Antwort 4	Red-Box-Test	false

Tabelle 3: Beispieldaten Antworten

1.1.2 Beispiel 2: Städte

Stapel:

Bezeichnung	Städte in Europa
Beschreibung	Zuordnung von Städten in Europa zu deren Ländern.

Tabelle 4: Beispieldaten Stapel - „Softwaretechnik“

Lernkarte:

Frage	Welche Stadt liegt in Spanien?
Anzahl Richtig	4
Anzahl Falsch	2
Zuletzt Richtig	true
Zuletzt Gespielt	1462106700 (Unix Zeitstempel: 01.05.2016 12:45)

Tabelle 5: Beispieldaten Lernkarte - „Städte“

Antworten:

Antwort 1	Mailand	false
Antwort 2	Madrid	true
Antwort 3	Barcelona	true
Antwort 4	Berlin	false

Tabelle 6: Beispieldaten Antworten

1.2 Glossar

Das Glossar beschreibt die wichtigsten Begriffe, die im Datenmodell verwendet werden:

Stapel Ein Stapel ist eine Sammlung von Karten, die zusammen gelernt werden können. In der Regel gibt es einen Stapel für ein Thema. Ein Stapel wird innerhalb der Implementierung auch als „Deck“ bezeichnet.

Karte Eine Lernkarte oder innerhalb der Implementierung auch „Flashcard“ enthält eine Frage oder einen Hinweis auf eine Antwort, die gelernt werden soll. Zusätzlich werden Statistiken zu jeder Karte gespeichert.

Antwort Jede Karte kann nur eine, aber auch mehrere Antworten enthalten, die nach dem Multiple-Choice-Prinzip ausgewählt werden können.

ZuletztRichtig Hält fest, ob die Lernkarte beim letzten Lernen korrekt beantwortet wurde.

ZuletztGespielt Legt fest, wann die Lernkarte zuletzt gelernt wurde. Das Lernen der Lernkarten wird im Folgenden auch als „spielen“ bezeichnet.

1.3 Datenmodell

Die Analyse der Projektbeschreibung hat ein Datenmodell ergeben, es ist in Abbildung 1 zu sehen.
Ein Stapel enthält neben der Bezeichnung und der genauen Beschreibung auch die Lernzeit. Dies ist die Gesamtzeit, die ein Benutzer mit dem Lernen von diesem Stapel aufgewendet hat.
Zu einem Stapel gehören mehrere Karten, welche neben der Frage auch weitere Daten für eine Statistik enthalten. Dazu gehört die Anzahl, wie oft die Karte korrekt oder falsch beantwortet war genauso wie der Zeitstempel, wann die Karte das letzte Mal richtig beantwortet bzw. gelernt wurde.
Zu einer Karte kann es mehrere Antworten geben (evtl. auch keine bei einer einseitigen Lernkarte). Die Antwort enthält einen Text für die Lösung. Außerdem gibt es ein Attribut, welches besagt, ob diese Antwort auch die korrekte Antwort ist. Das ist wichtig, um bei einer Multiple-Choice-Lösung die richtige Antwort definieren zu können.
Sowohl eine Karte als auch eine Antwort können mehrere Bilder enthalten. Zu diesen wird ein Name, sowie der MimeType und die Groesse als Meta-Informationen festgelegt.

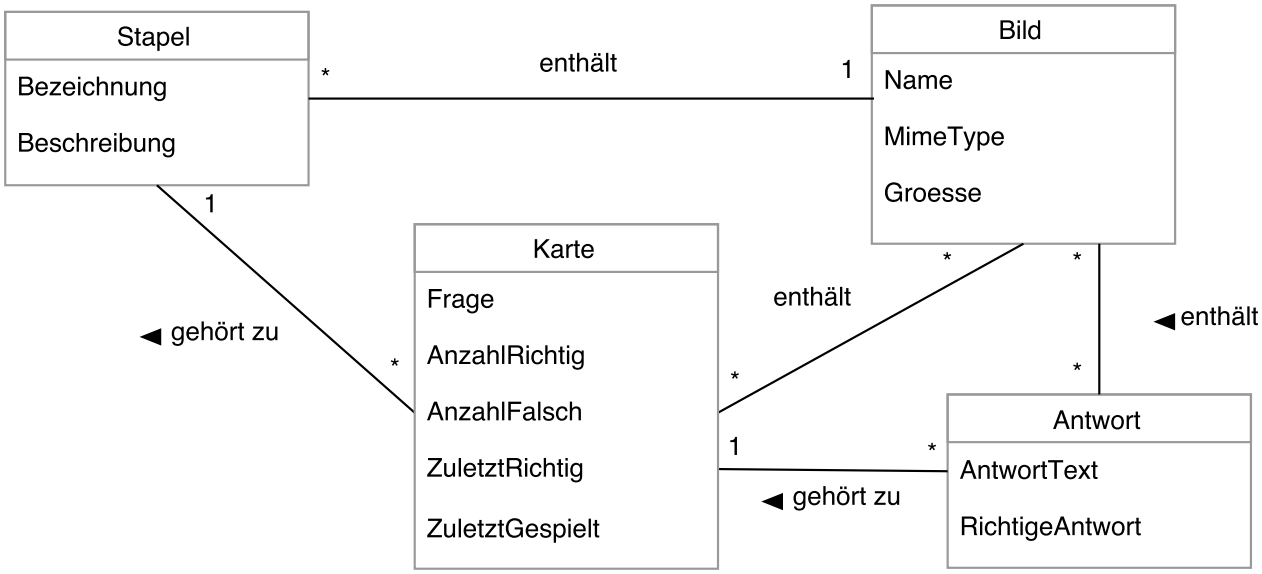

Abbildung 1: Domänenmodell

1.4 Benutzeroberfläche

Zur ersten Visualisierung der GUI (auch für den Auftraggeber) wurden einige Skizzen mit Hilfe des Mockup-Tools Pencil erstellt. Diese sollen einen Überblick über das grundsätzliche Aussehen der Anwendung geben. Die Struktur und der Aufbau der App sollen hiermit schon zu Beginn möglichst intuitiv geplant werden.
Es erfolgt eine Unterscheidung zwischen der Oberfläche auf einem Smartphone und dem Layout auf einem Tablet. Hier ist es wichtig, je nach Geräteklasse den Bereich des Screens möglichst gut auszunutzen.

1.4.1 Smartphone

Das Layout soll möglichst gut an das mit Android 5 erschienene „Material Design“ angepasst werden. Dies ermöglicht dem Benutzer ein einheitliches Bedienungserlebnis auch über die Grenzen von verschiedenen Apps hinweg. Auf fast allen Seiten der Anwendung ist über den auf der oberen linken Seite platzierten Button ein Seitenmenü aufzurufen, welches verschiedene Optionen bereithält. Der Lernmodus soll soweit wie möglich im Vollbildmodus stattfinden, um eine möglichst geringe Ablenkung zu erreichen.

Home

Der Home-Screen besteht hauptsächlich aus einer Liste von verschiedenen Stapeln mit Lernkarten. Diese sollen in der Grundeinstellung nach der letzten Benutzung sortiert sein, sodass der Anwender schnell auf die aktuellen Lernaufgaben zugreifen kann. Neue Karten und Stapel können in der jeweiligen Liste über ein Plus-Symbol am unteren Rand erstellt werden.
Zu jedem Stapel gibt es ein Untermenü, über das die Bearbeitung und das Löschen möglich ist. Zusätzlich gibt es eine Suchfunktion.

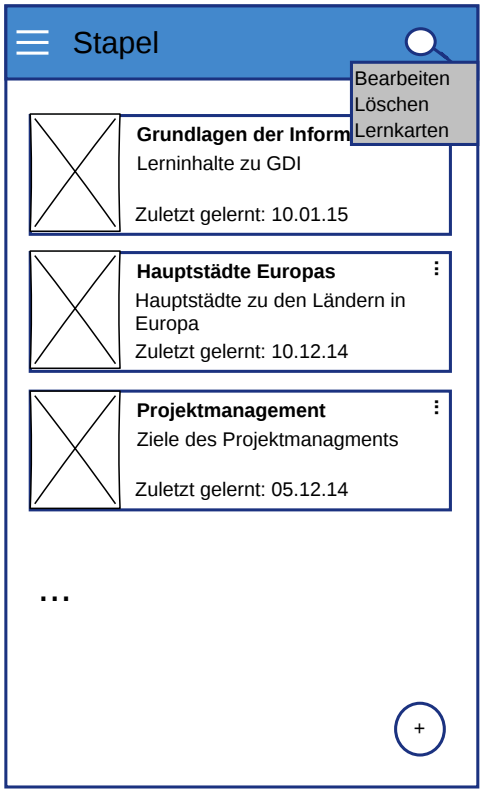

Abbildung 2: Screen: Home ohne Menü

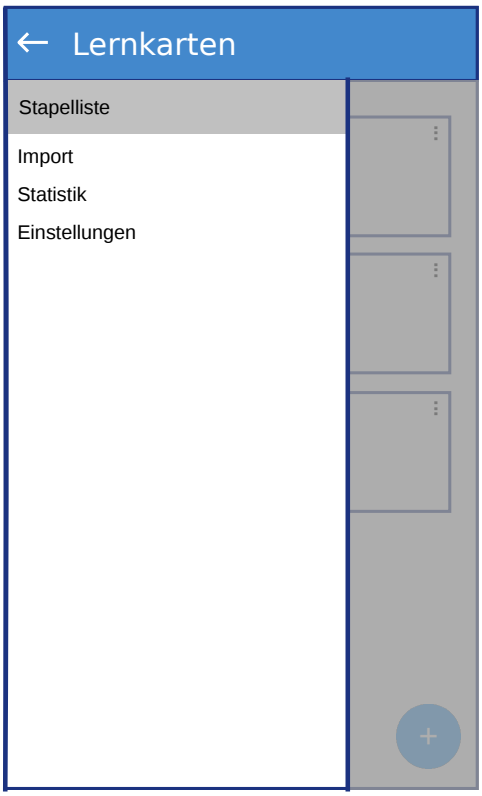

Abbildung 3: Screen: Home mit Menü

Anlegen von einem Stapel

Vom Home-Screen aus ist es möglich zunächst einen neuen Stapel über das Menü anzulegen. Hierzu erscheint ein zusätzliches Fenster, in dem die Eingaben gemacht werden können.

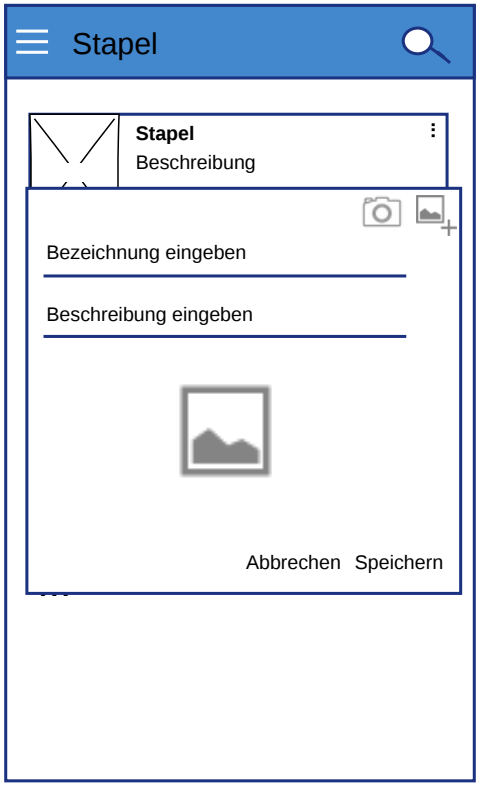

Abbildung 4: Screen: Anlegen eines Stapels

In einem Stapel kann dann eine neue Karte angelegt werden. Hierzu ist es dann auch erforderlich eine Antwort zu definieren. Diese kann aus einer Lösung bestehen oder auch eine Multiple-Choice Auswahl für mehrere Lösungen bereitstellen.

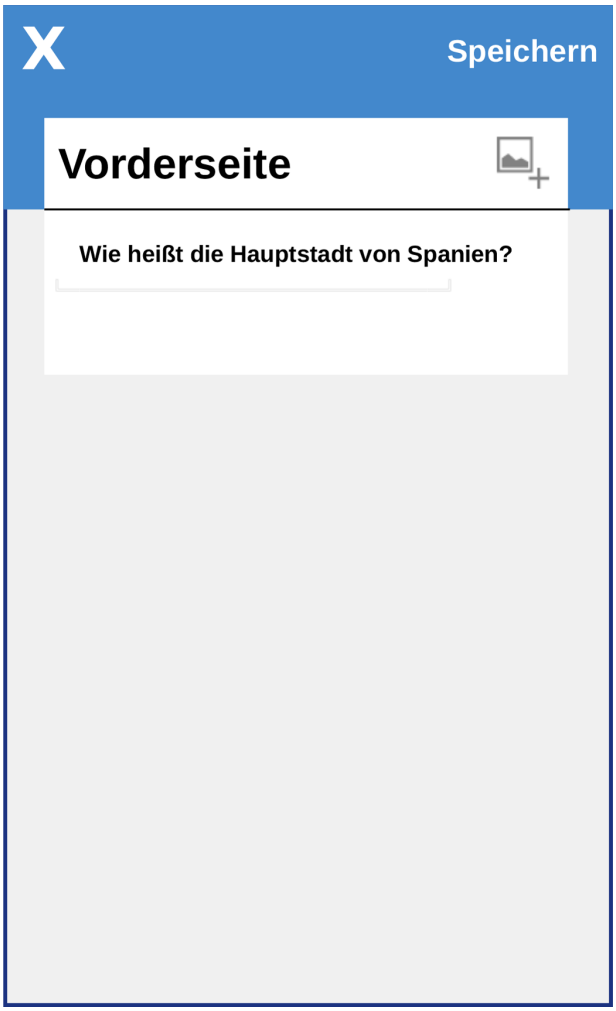

Abbildung 5: Anlegen einer Karte

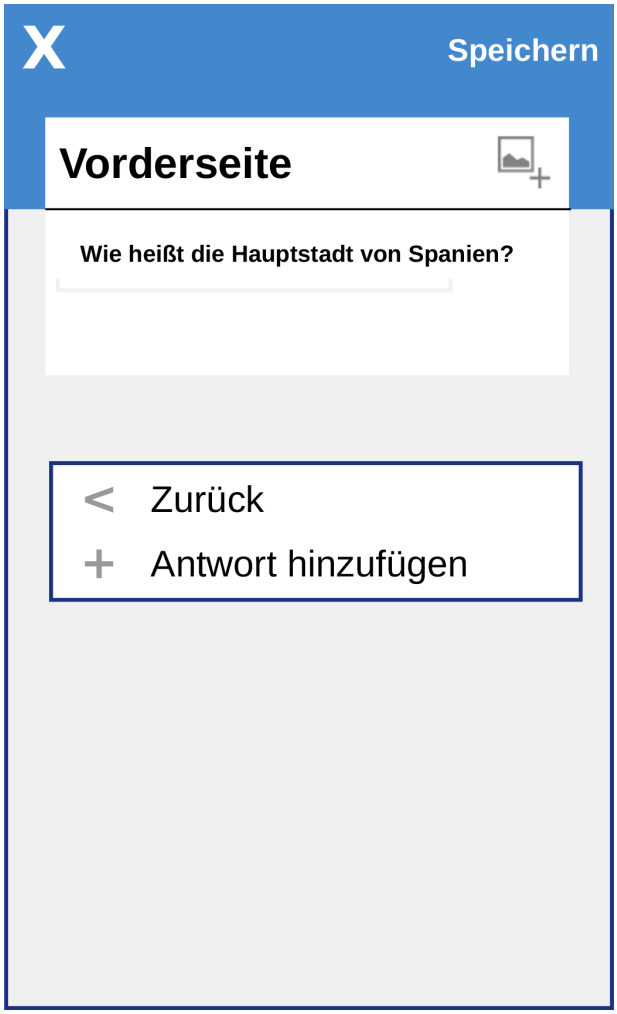

Abbildung 6: Speichern Dialog

Bei dem Speichern einer neuen Karte können neue Antworten hinzugefügt werden. Speichert man die neue Antwort, wird der Benutzer durch einen Dialog aufgefordert eine weitere Antwort hinzufügen oder er kann zu der Frage zurück kehren.

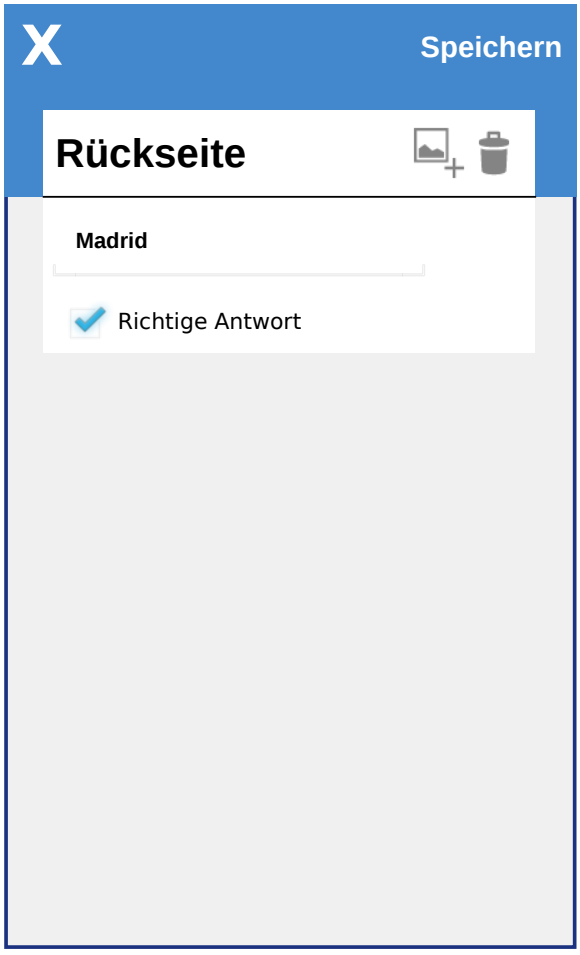

Abbildung 7: Anlegen einer Antwort

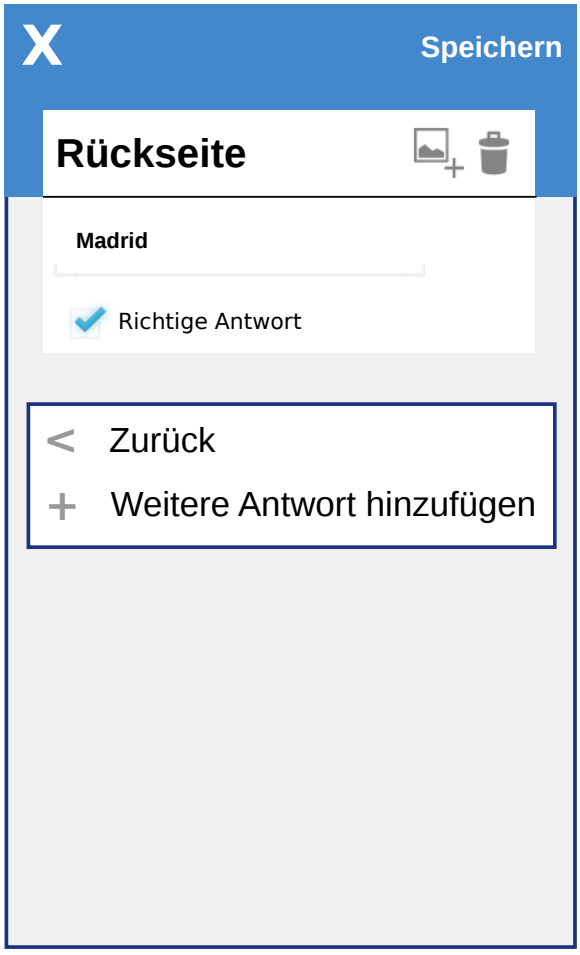

Abbildung 8: Speichern Dialog

Lernmodus

Im Lernmodus gibt es zwei verschiedene Arten, wie ein Benutzer die Frage beantworten kann. Im einfachen Modus kann man sich die Antwort über einen Button anzeigen lassen und dann auswählen, ob man die Lösung gewusst hat. Im erweiterten Modus ist eine manuelle Eingabe der Antwort möglich. Bei einer Karte mit einer Multiple-Choice-Lösung ist die korrekte Antwort auszuwählen.

Abbildung 9: Anzeige einer Frage im Lernmodus

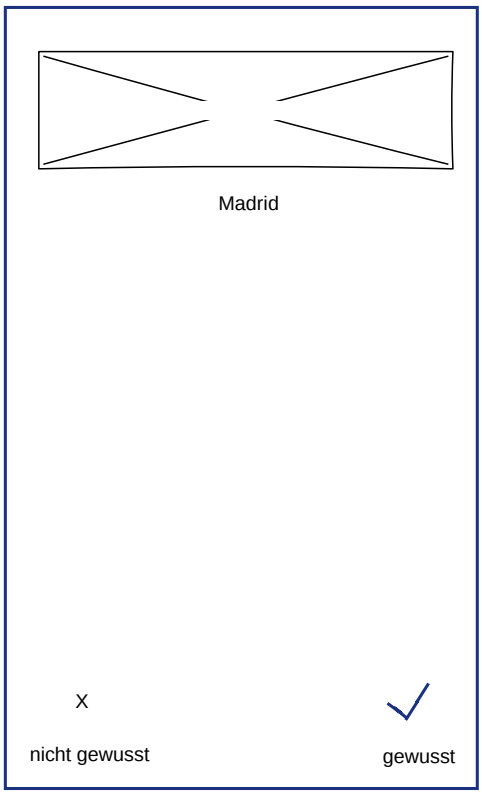

Abbildung 10: Anzeige einer Antwort

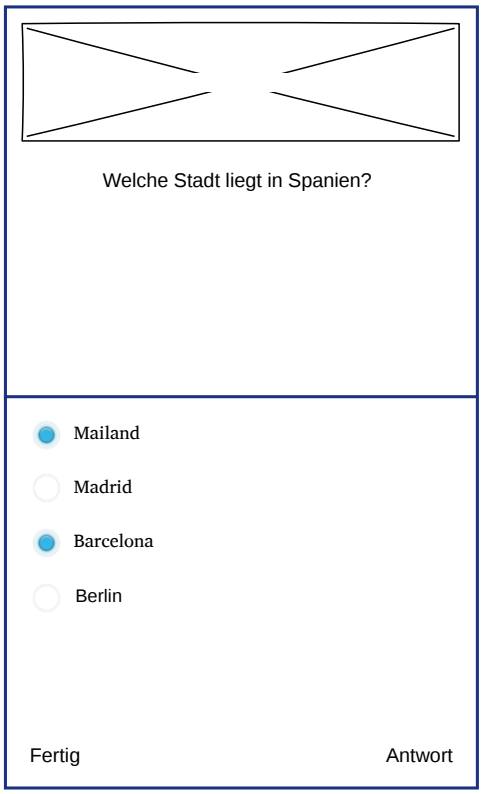

Abbildung 11: Anzeige einer Frage im Lernmodus - Multiple Choice

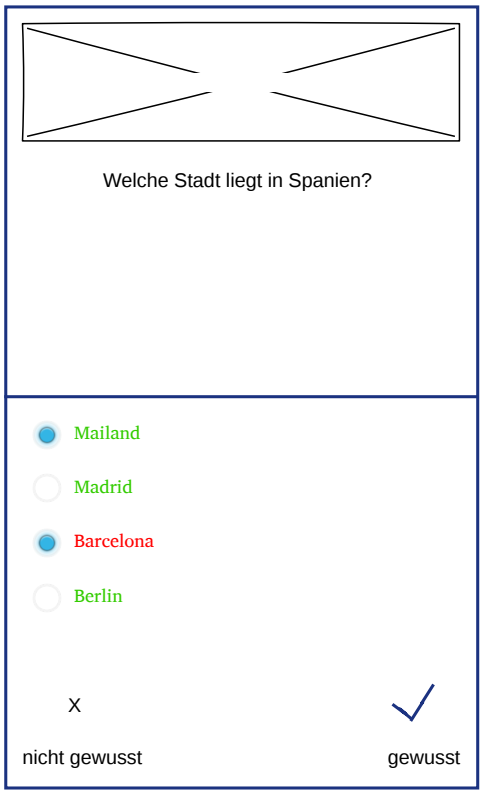

Abbildung 12: Anzeige einer Antwort - Multiple Choice

Statistik

Während des Lernens werden Statistiken zu gewussten und nicht gewussten Fragen geführt. Diese Statistiken kann der Benutzer sich Stapelweise ansehen.

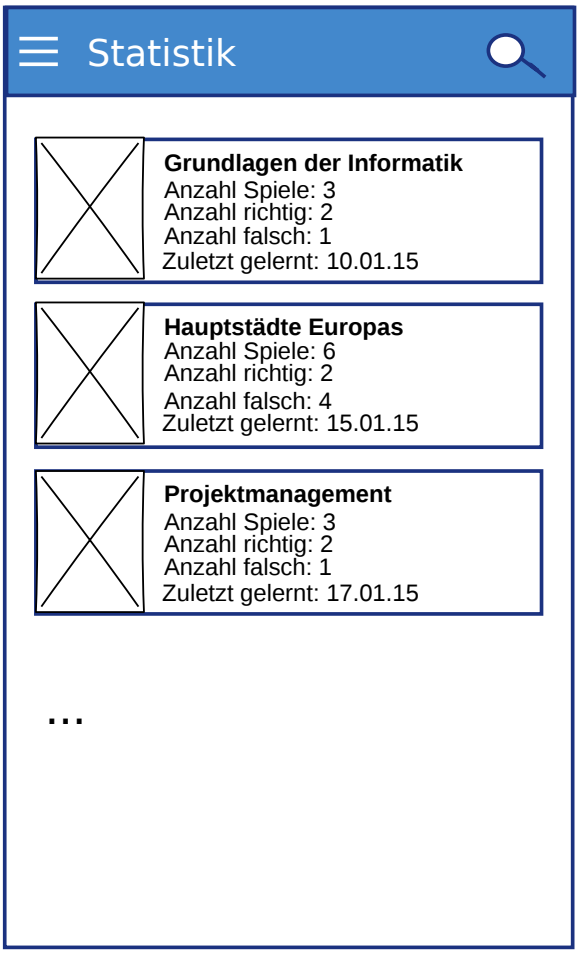

Abbildung 13: Statistik Darstellung

Einstellungen

In den Einstellungen kann der Benutzer die gewünschte Spielweise auswählen. Die Applikation bietet zwei Spielweisen an, zum einen das Spielen nach einer zufälligen Sortierung. Bei der zweiten Spielweise werden Fragen, welche falsch beantwortet wurden häufiger im Spiel vorkommen.

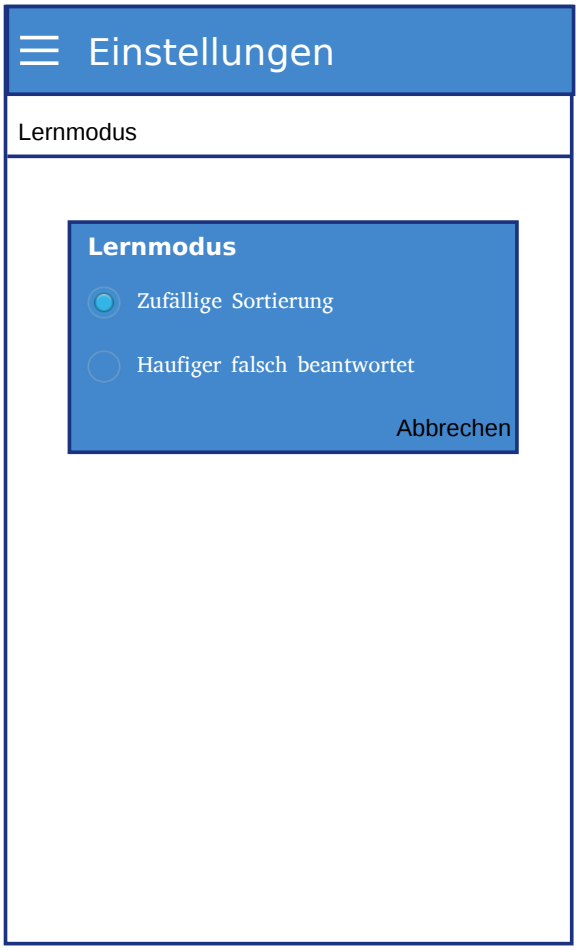

Abbildung 14: Auswahl der Einstellungen

1.4.2 Tablet

Beim Einsatz eines Tablets wird das Layout automatisch an die größere Anzeige angepasst. Es sind die gleichen Optionen und Möglichkeiten wie bei einem kleineren Display vorhanden. Dabei soll allerdings der größere Bildschirm besser genutzt werden. Die folgenden Abbildungen zeigen beispielhaft wie dies umgesetzt werden soll. Ein Beispiel hierfür ist, das die Karten in einer mehrspaltigen Ansicht dargestellt werden.

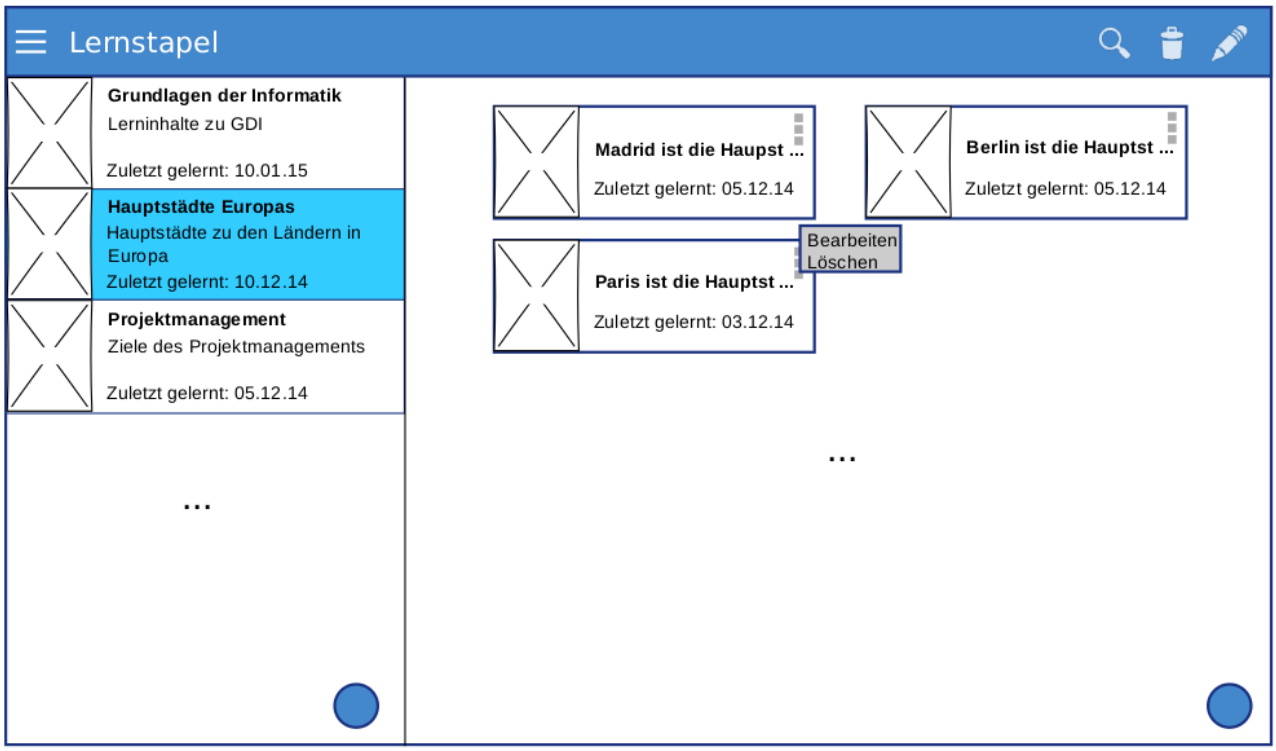

Abbildung 15: Mehrspaltige Ansicht auf einem Tablet

Im Lernmodus wird der Fokus auf die einzelne Karte gelegt, indem diese in der Mitte des Bildschirms platziert wird.

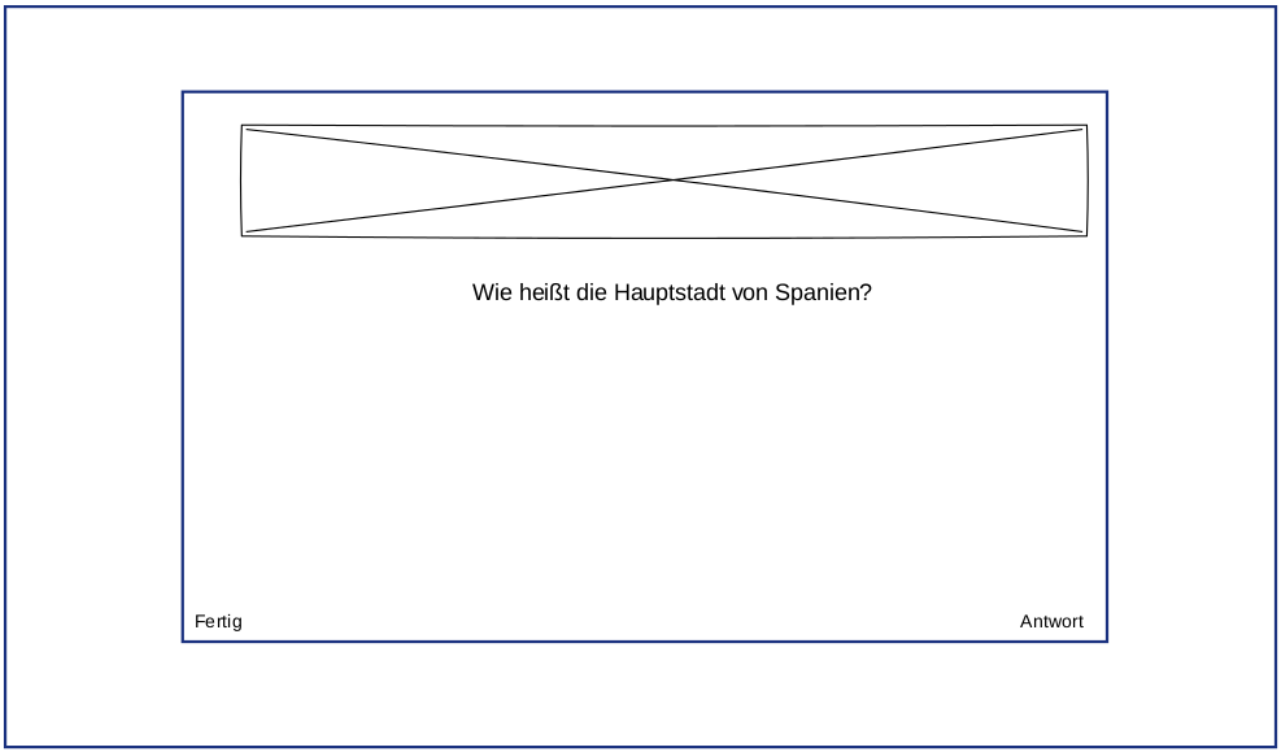

Abbildung 16: Lernmodus auf einem Tablet

1.5 Anwendungsfälle

Die verschiedenen Funktionen der App sollen nun in konkreten Anwendungsfällen beschrieben werden.

UC001 Einen neuen Stapel anlegen

Ziel	Einen neuen Stapel anlegen.
Ablauf	1. Der Benutzer klickt auf den Menü-Button. 2. Klick auf „Suche“ (Neuer Stapel). 3. Eingabe der benötigten Daten (Bezeichnung und Beschreibung). 4. Stapel wird in Liste angezeigt.
Alternative Ablaufschritte	4a. Fehlermeldung zu ungültigen Eingaben oder zu fehlenden Daten in Pflichtfeldern.

UC002 Eine neue Karte anlegen

Ziel	Eine neue Karte anlegen.
Ablauf	1. Der Benutzer klickt auf den Menü-Button des Stapels und wählt Liste aus. 2. Anzeige der Karten zu diesem Stapel. 3. Benutzer klickt auf Menü-Button und wählt neue Karte aus. 4. Eingabe der Daten zu dieser Karte. 5. Speichern der Daten.
Alternative Ablaufschritte	4a. Hinzufügen von Bildern. 5a. Fehlermeldung zu ungültigen Eingaben oder zu fehlenden Daten in Pflichtfeldern.

UC003 Eine Karte bearbeiten

Ziel	Eine neue Karte anlegen.
Ablauf	1. Der Benutzer klickt auf den Menü-Button des Stapels und wählt Liste aus. 2. Anzeige der Karten zu diesem Stapel. 3. Benutzer klickt auf Menü-Button der einzelnen Karte und wählt Karte bearbeiten aus. 4. Eingabe der Daten zu dieser Karte. 5. Speichern der Daten.
Alternative Ablaufschritte	4a. Hinzufügen von Bildern. 5a. Fehlermeldung zu ungültigen Eingaben oder zu fehlenden Daten in Pflichtfeldern.

UC004 Eine Karte löschen

Ziel	Eine Karte löschen.
Ablauf	1. Der Benutzer klickt auf den Menü-Button des Stapels und wählt Liste aus. 2. Anzeige der Karten zu diesem Stapel. 3. Benutzer klickt auf Menü-Button und wählt Karte löschen aus. 4. Eingabe der Daten zu dieser Karte. 5. Benutzer beantwortet Sicherheitsabfrage positiv.
Alternative Ablaufschritte	

UC005 Lernen

Ziel	Einen Stapel lernen.
Ablauf	1. Der Benutzer wählt den gewünschten Stapel und der Lernmodus startet. 2. Nach Anzeige der Karte klickt Benutzer auf die jeweilige Antwort (Multiple Choice) oder auf Zeige Antwort. 3. Benutzer wählt, ob seine Antwort richtig war. 4. Die nächste Karte erscheint, weiter bei 2. bis alle Karten durchgegangen sind. 5. Anzeige der Statistik zu diesem Stapel.
Alternative Ablaufschritte	2a. Eingabe der Antwort und Prüfung, ob diese korrekt ist. 2b. Einseitige Karte: Auswahl, ob die Antwort korrekt war oder nicht.

UC006 Statistik

Ziel	Anzeige der Statistiken zu den einzelnen Stapeln.
Ablauf	1. Der Benutzer klickt auf den Menü-Button und wählt Statistik. 2. Anzeige einer Liste mit Statistiken zu jedem Stapel.
Alternative Ablaufschritte	

UC007 Import

Ziel	Importieren eines Stapels mit Karten aus einer Datei.
Ablauf	1. Der Benutzer klickt auf den Menü-Button und wählt Import. 2. Benutzer wählt den Stapel im Dateisystem aus.
	3. Anzeige, ob Import erfolgreich.
Alternative Ablaufschritte	

1.6 Weitere Anforderungen

Der Benutzer kann im Einstellungsmenü die Reihenfolge der Lernkarten im Lernmodus bestimmen. Es gibt die Möglichkeit der zufälligen Sortierung oder das Karten, die häufiger falsch beantwortet wurden, öfter angezeigt werden.

Die Beschreibungen zu weiteren technischen Anforderungen sind im Entwurfsdokument im Kapitel technische Infrastruktur zu finden.

Entwurf

Dieses Entwurfsdokument enthält den technischen Lösungsentwurf für die im Konzept spezifizierte App. Einerseits wird beschrieben, welche Entwurfsentscheidungen durch Android vorgegeben sind, andererseits werden die Datenhaltung, die Paketstruktur und einige besondere Aspekte erläutert, die als Basis für die spätere Implementierung dienen.

1 Technische Infrastruktur

1.1 Anforderungen

Aus der Projektbeschreibung lassen sich folgende technische Anforderungen ableiten:

1. Speicherung und Verwaltung der einzelnen Stapel und Lernkarten.
2. Nachladen der Lernkarten von einem Server.
3. Anderes Layout für Tablets, um den größeren Bildschirm besser auszunutzen.

Diese Anforderungen werden bereits größtenteils durch Android abgedeckt. Das System stellt hier zur Speicherung der Daten eine SQLite Datenbank zur Verfügung. Das Nachladen von weiteren Karten und Stapeln erfolgt mit Hilfe einer JSON-Datei, die von einem Server heruntergeladen wird.

1.2 Realisierung

1.2.1 Android SDK

Für die Realisierung der Anwendung wird das Android SDK benötigt. Dies ist eine Sammlung von verschiedenen Werkzeugen, welche die Softwareentwicklung mit Android erlauben.

1.2.2 Android Support Library

Die Android Support Library ist eine Sammlung von Bibliotheken, welche die Abwärtskompatibilität von neuen Features ermöglichen[1]. In diesem Projekt wird unter anderem das Material Design eingesetzt. Da dieses Design erst ab Android 5 - Lollipop eingeführt wurde, ist es für vorherige Android Versionen notwendig diese zusätzliche Bibliothek einzusetzen.

[1] `http://developer.android.com/tools/support-library/index.html`

1.2.3 Gradle

Gradle[2] ist ein Build-Management-Automatisierungs-Tool, welches in der Entwicklungsumgebung Android Studio integriert ist. Hiermit ist es möglich die fertige APK für die Installation auf einem Android Gerät zu bauen.

1.2.4 Datenbankzugriff

Unter Android wird für die Persistierung von Daten in der Regel eine SQLite Datenbank verwendet. Für den Zugriff darauf enthält die API des Android SDK eine Klasse mit dem Namen SQLiteOpenHelper[3]. Diese Klasse kümmert sich um das Öffnen und falls notwendig um das Erzeugen einer Datenbank. Außerdem können Transaktionen genutzt werden, um sicherzustellen, dass sich die Datenbank immer in einem konsistenten Zustand befindet.

1.2.5 Erforderliche Infrastruktur

Zum Bereitstellen von weiteren Lernkartenstapeln im JSON-Format über HTTP wird ein einfacher Webserver benötigt.

[2] `http://developer.android.com/tools/building/building-cmdline.html`
[3] `http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html`

2 Projektstruktur

Bevor auf die Architektur der Anwendung eingegangen wird, soll die Organisation der Dateien, Ordner und Klassen des Projektes kurz vorgestellt werden.

2.1 Grundlegende Organisation

Die grundlegende Ordnerstruktur des Projektes ist durch Android und Gradle folgendermaßen festgelegt:

- src
  Das Quellverzeichnis des Projektes
  - main
    Die eigentliche Anwendung
    - java
      Der Java-Code
    - res
      Bilder und zusätzliche Ressourcen
    - AndroidManifest.xml
      Grundlegende Angaben zum Projekt
  - androidTest
    Enthält Tests

2.2 Paketstruktur

Der Code des Projektes ist in folgende Pakete unterteilt:

Tabelle 1: Die Aufteilung der Klassen in Pakete

Paket	**Inhalt**
usg	Die Activites der Anwendung sind hier enthalten
usg.data	Enthält Entitätsklassen, sowie Androidspezifische Klassen zum Zugriff auf eine SQLite-Datenbank
usg.util	Beinhaltet Hilfsklassen der Anwendung

3 Architektur

Die Architektur der Anwendung ist im wesentlichen durch Android vorgegeben. In diesem Abschnitt soll daher auf die für das Verständnis unserer Anwendung nötigen Konzepte von Android eingegangen werden und geschildert werden, wie diese eingesetzt wurden.

3.1 Grundlegende Funktionsweise von Android

Android Anwendungen werden in Java geschrieben und der Code wird zusammen mit allen Daten und Resourcen in ein APK-Archiv gepackt. Dieses Archiv lässt sich dann auf einem Gerät, welches mit Android läuft, installieren. Aus Sicherheitsgründen läuft jede App in ihrer eigenen vom System verwalteten Sandbox, sodass jede Anwendung so wenig Rechte wie möglich besitzt. Sind bestimmte Rechte, wie zum Beispiel Netzwerkkonnektivität erforderlich, können sie explizit angefragt werden und müssen dann beim Installieren vom Benutzer erlaubt werden.

3.2 Anwendungskomponenten

Eine Android-Anwendung besteht aus einer Reihe von Anwendungskomponenten, die im Folgenden kurz erläutert werden sollen, um ein genaueres Verständnis von der Funktionsweise zu bekommen und später aufgeführte Implementierungsdetails besser verstehen zu können.

3.2.1 Activity

Eine Activity ist eine Komponente, die eine einzelne Seite der Anwendung darstellt. Es wird die Kontrolle über das User Interface und den User Input übernommen. Eine Anwendung enthält in der Regel mehrere Activities, die zusammenarbeiten, aber voneinander unabhängig sind.

Ein Beispiel für eine Activity kann die Liste von Stapeln in einer Lernkarten-App sein.

Jede Activity besitzt einen Lebenszyklus und kann sich daher in verschiedenen Zuständen befinden. Die wichtigsten Zustände sind Resumed, Paused und Stopped und sollen nun kurz erläutert werden.

- **Resumed:** Die Activity befindet sich im Vordergrund und der Benutzer hat den Fokus darauf. Dieser Zustand kann auch als **Running** bezeichnet werden.
- **Paused:** Eine andere Activity ist im Vordergrund und hat den Fokus des Benutzers. Diese Activity ist aber immer noch sichtbar. Es kann zum Beispiel sein, dass eine andere Activity darüber liegt und das durch Transparenz noch eine Sichtbarkeit dieser Activity gegeben ist. Pausierte Activities befinden sich noch komplett im Speicher, alle Zustandsinformationen werden immer noch behalten und sie ist immer noch an den Fenstermanager angehängt. Das System beendet eine pausierte Activity nur, wenn extrem wenig Speicher zu Verfügung steht.
- **Stopped:** Die Activity befindet sich im Hintergrund und wird vollständig von einer anderen bedeckt. Sie befindet sich noch komplett im Speicher und die Zustandsinformationen werden ebenfalls noch

behalten. Eine gestoppte Activity ist allerdings nicht mehr an den Fenstermanager angehängt und kann vom System beendet werden, wenn irgendwo anders Speicher benötigt wird.

Für jeden Zustand dieses Lebenszyklus existiert eine Callback-Methode, die aufgerufen wird, wenn das System den Zustand wechselt. So kann man die Liste aktualisieren wenn die Activity sichtbar wird, indem man die entsprechende Callback-Methode überschreibt und den eigenen Code ausführt.

Abbildung 1 auf der nächsten Seite zeigt den vollständigen Überblick über den Lebenszyklus einer Activity und stellt die verschiedenen Callback-Methoden dar.

3.2.2 Service

Ein Service ist eine Komponente, die im Hintergrund läuft, um lang laufende Operationen durchzuführen. Hierbei existiert keine Benutzerschnittstelle. Somit werden Dienste für andere Komponenten bereitgestellt. Ein Beispiel für einen Service kann das Abspielen von Musik oder das Herunterladen einer größeren Datei im Hintergrund sein.

3.2.3 Content Provider

Content Provider regeln den Zugriff auf Daten zwischen Anwendungen. Hierzu können diese Daten über URIs zum Beispiel nur lesend, aber auch schreibend zur Verfügung gestellt werden.
Eine App kann über einen Content Provider auf die Kontaktdaten im Adressbuch zugreifen, somit kann eine andere App zum Beispiel die Daten zu einer bestimmten Person anzeigen.

3.2.4 Broadcast Receiver

Ein Broadcast Receiver empfängt systemweite Nachrichten und verarbeitet diese. Beispiele für diese Nachrichten können das Ausschalten des Bildschirms oder die Nachricht, dass der Akkustand niedrig ist, sein. Diese Komponente besitzt keine Benutzerschnittstelle, allerdings können hieraus Benachrichtigungen in der Statusbar erstellt werden.

3.3 Design

Seit Android Version 5 soll das sogenannte Material Design bei der Entwicklung von Apps eingesetzt werden. Diese Forderung zu realisieren ist wichtig, um dem Benutzer auf dem gesamten System ein einheitliches Bedienungs- und Designsystem zu bieten. Hierdurch kann die Akzeptanz einer Anwendung erheblich gesteigert werden, wenn der Anwender eine „gewohnte“ Umgebung vorfindet.

Google gibt hierzu genaue Regeln und Ziele vor, wie das Design einer Anwendung aussehen soll. Dazu wird eine 3D-Welt als Material Umgebung definiert. Jedes Objekt in dieser Welt besitzt eine X, Y und Z-Dimension. Dabei steht die Z-Achse senkrecht auf der Ebene des Displays und zeigt somit auf den

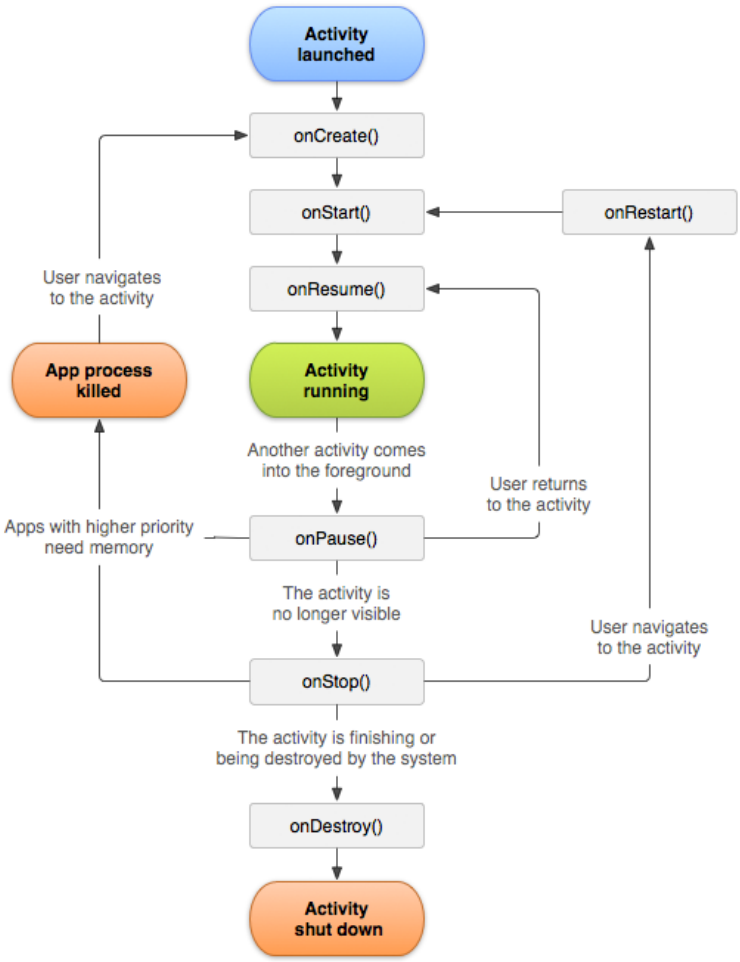

Abbildung 1: Lebenszyklus einer Activity[5]

Betrachter. Zusätzlich hierzu werden Regeln für Licht und Schatten, Bewegungen und Größen von Objekten gegeben. Die detaillierte Auflistung und Beschreibung ist auf der Website[4] zu Material Design zu finden.

3.4 Listen

Innerhalb der Anwendung benötigen wir verschiedene Listen. Dies sind unter anderem die Auflistung aller Stapel und innerhalb der Stapel die Anzeige der jeweiligen Lernkarten. Hierzu benutzt Android ein eigenes Konzept. Dieses besteht aus drei Komponenten. Auf der einen Seite steht die Datenquelle, aus dieser werden die anzuzeigenden Daten abgerufen. Auf der anderen Seite befindet sich eine View-Komponente, die für die Anzeige der Daten verantwortlich ist. Diese View arbeitet mit einem Adapter zusammen, der die Daten

[5] Quelle: `http://developer.android.com/guide/components/activities.html`
[4] `https://www.google.com/design/spec/material-design/introduction.html`

aus der Datenquelle nimmt und diese in die Liste hineinfüllt. Dieses Konzept ist in Abbildung 2 auf der nächsten Seite aufgezeigt.

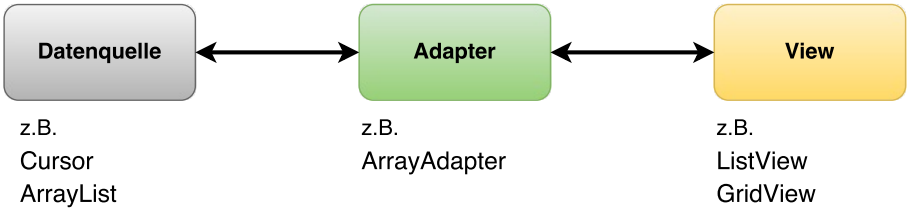

Abbildung 2: Zusammenspiel von Datenquelle, ArrayAdapter und ListView

Die Implementierung richtet sich nach diesem Konzept. Um einen Eindruck von dessen Verwendung zu geben, soll es hier am Beispiel der Liste aller Decks vorgestellt und erläutert werden. Eine vereinfachte Implementierung ist in Listing 1 dargestellt.

Die hier dargestellte Klasse ist von Fragment abgeleitet. Dies erlaubt es das Fragment einerseits in einer Ansicht bildschirmfüllend auf einem Smartphone anzuzeigen und andererseits es in einer zweispaltigen Ansicht auf einem Tablet zu integrieren.

In der dargestellten Methode `onActivityCreated` werden die einzelnen Variablen jeweils für die Benutzung initialisiert. Innerhalb der Anwendung wird häufig `findViewById` benutzt. Die Initialisierung der Variable `mListView` zeigt die Benutzung dieser Methode (Zeile 16). Um deren Funktionsweise zu verstehen, muss man wissen, dass man innerhalb des XML-Layouts für ein Fragment oder eine Activity jeder einzelnen Komponente mit einer ID kennzeichnen kann. Mit Hilfe dieser ID kann eine Komponente aus dem Layout dann innerhalb einer Activity gefunden werden. Da `findViewById` nur eine View zurückliefert, ist es noch noch notwendig einen Cast auf die jeweilige Klasse durchzuführen.

Die Datenquelle wird durch die Variable `mDecks` repräsentiert. Das Abrufen der Daten aus der Datenbank geschieht innerhalb der Methode `ddao.findAll`. Anschließend werden alle gefundenen Daten als ArrayList zurückgegeben.

Für den notwendigen Adapter wird ein ArrayAdapter verwendet, dessen Methode `getView` überschrieben wird. Ein ArrayAdapter ist ein konkreter Adapter innerhalb der API des Android SDK und kann für Listen und Arrays verwendet werden. Innerhalb von `getView` erfolgt dann das Befüllen der einzelnen TextView-Komponenten eines Decks. In Zeile 30 wird beispielsweise der Titel eines Decks gesetzt. Hierzu wird über die Variable `position` das aktuelle Element der ArrayList ausgewählt. An dieser Stelle ergibt sich also, dass die Methode `getView` für jedes Element der Kollektion aufgerufen wird.

In Zeile 45 erfolgt abschließend eine Zuweisung des Adapters an die ListView.

```
1
2  public class DeckListFragment extends Fragment implements SearchView.OnQueryTextListener {
3
4      private View rootView;
5      private ListView mListView;
6      private ArrayAdapter<Deck> mAdapter;
7      private List<Deck> mDecks;
8
9      // [...]
10
11     @Override
12     public void onActivityCreated(Bundle savedInstanceState) {
13         super.onActivityCreated(savedInstanceState);
14
15         DeckDAO ddao = AppFactory.get(getActivity().getApplicationContext()).getDeckDAO();
16         mListView = (ListView) rootView.findViewById(R.id.deck_listview);
17         mDecks = ddao.findAll();
18
19         // [...]
20
21         mAdapter = new ArrayAdapter<Deck>(getActivity(), R.layout.deck_list_item, mDecks) {
22
23             @Override
24             public View getView(final int position, View convertView, ViewGroup parent) {
25                 LayoutInflater inflater = (LayoutInflater) mContext
26                         .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
27
28                 View rowView = mHasTwoPanes ? inflater.inflate(R.layout.deck_list_item_multipane,
                       parent, false) : inflater.inflate(R.layout.deck_list_item, parent, false);
29
30                 TextView textViewTitle = (TextView) rowView.findViewById(R.id.list_item_title);
31                 textViewTitle.setText(mDecks.get(position).getName());
32
33                 TextView textViewDesc = (TextView) rowView.findViewById(R.id.list_item_desc);
34                 textViewDesc.setText(mDecks.get(position).getDescription());
35
36         // [...]
37
38                 return rowView;
39
40             }
41
42
43         };
44
45         mListView.setAdapter(mAdapter);
46         mListView.setTextFilterEnabled(true);
47
48         // [...]
49
50     }
51 }
```

Listing 1: Verwendung eines ArrayAdapters am Beispiel der Klasse DeckListFragment

Die Darstellung der hier implementierten Liste aller Lernstapel ist in Abbildung 3 auf der nächsten Seite zu sehen. Es gibt hier zwei Elemente der Liste, die mit den jeweiligen Daten aus der Datenbank gefüllt wurden.

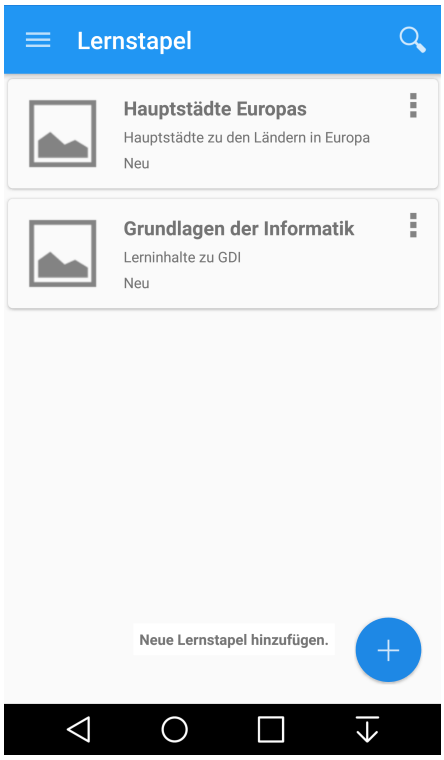

Abbildung 3: Liste der Lernstapel

3.4.1 Fragments

Ein Fragment repräsentiert ein Verhalten oder einen bestimmten Teil der Benutzeroberfläche. Es ist möglich mehrere Fragments in einer Activity zu kombinieren. Der Ansatz von Fragments erlaubt die Wiederverwendbarkeit in mehreren Activities. Dies hat den Vorteil, dass man beispielsweise ein Fragment für die Ansicht einer Lernkarte erstellen kann. Dieses kann in der Smartphone-Ansicht den gesamten Screen füllen. Für die Tablet-Ansicht kann es neben einem anderen Fragment eingesetzt werden, wenn man zum Beispiel ein Zwei-Spalten-Layout verwendet. Aus diesem Grund werden in dieser Anwendung Fragments benutzt, um sowohl die Smartphone-Ansicht als auch die Tablet-Ansicht darzustellen und gleichzeitig den Code möglichst wartbar zu gestalten.

3.5 Datenschicht

Die Daten der Anwendung werden in einer Datenbank gespeichert. Wir setzen hier SQLite ein, da es bereits in Android enthalten ist.

3.5.1 Repräsentation der Daten

Aus dem in der Spezifikation vorgestelltem Fachmodell ist ein Datenbankschema entwickelt worden. Hierzu war es notwendig, die benötigten Datentypen auszuwählen und die genauen Beziehungen zwischen den

Tabellen zu modellieren. Das Schema der Datenbank ist in Abbildung 4 dargestellt.

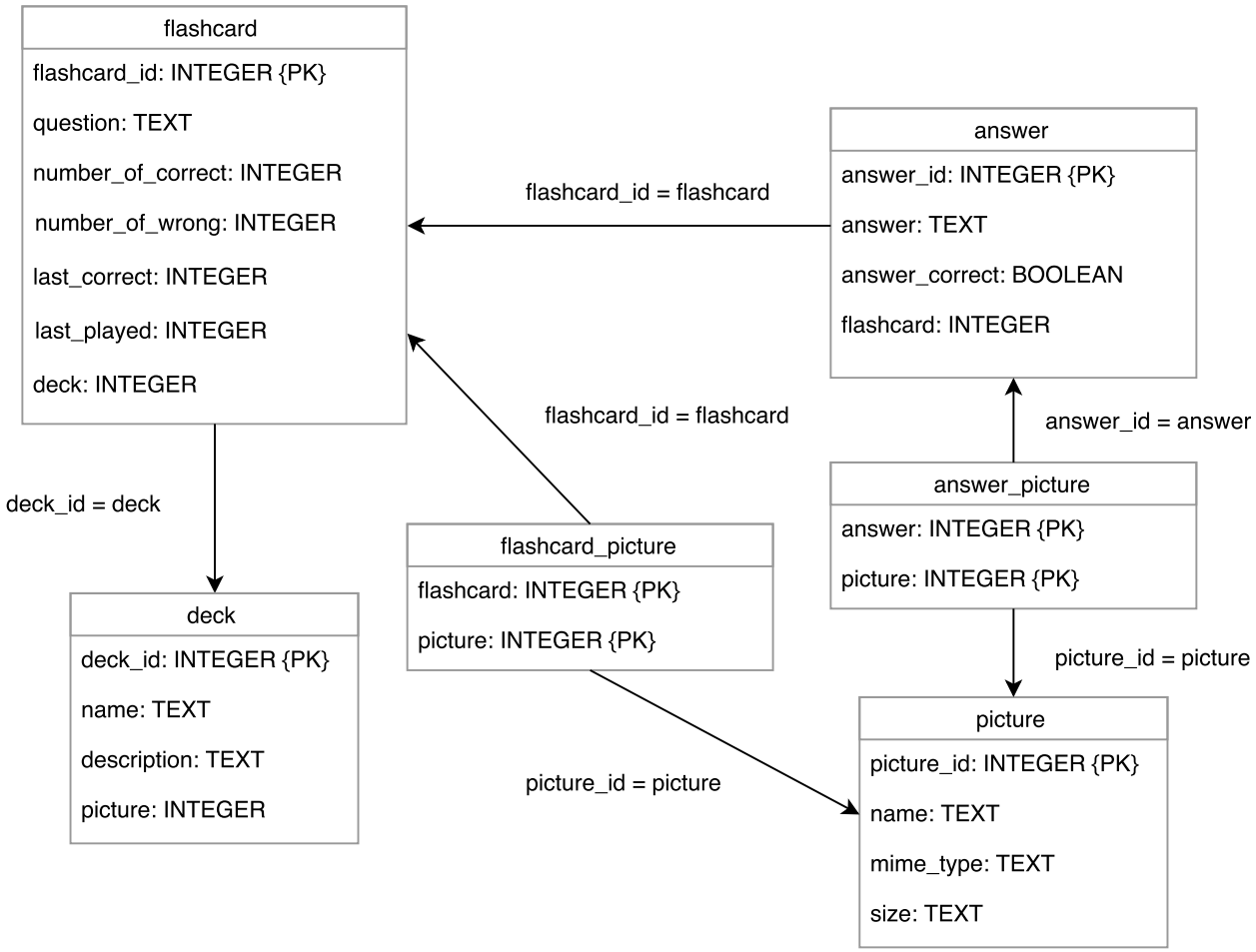

Abbildung 4: Diagramm des Datenbankschemas

3.5.2 Abbildung der Struktur auf Klassen

Innerhalb der Anwendung arbeiten wir mit Objekten, die die verschiedenen Datensätze der Datenbank widerspiegeln. Die Klassen dieser Objekte enthalten jeweils die Attribute, die in der Datenbank gespeichert sind. Die Attribute können jeweils mit einer Getter-Methode abgerufen und mit einer Setter-Methode geändert werden. Eine Übersicht über diese Klassen ist in Abbildung 5 auf der nächsten Seite zu sehen.

3.5.3 Zugriff auf die Datenbank

Für den Zugriff auf die Datenbank wir die Klasse MySQLiteOpenHelper verwendet. Diese ist abgeleitet von SQLiteOpenHelper und überschreibt die Methode `onCreate`. Hierüber ist das Erstellen und Anpassen des Tabellenmodells der Datenbank möglich. Listing 2 zeigt unsere Implementierung dieser Klassen. Aufgrund einer besseren Übersichtlichkeit erfolgt die Darstellung verkürzt.

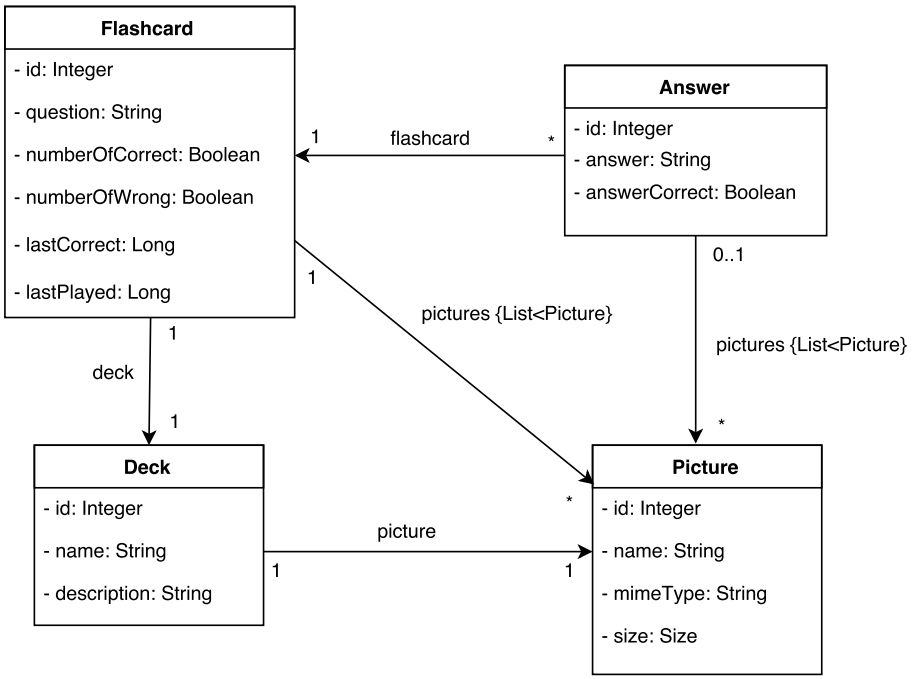

Abbildung 5: Klassenmodell

Mit Hilfe der Variablen `DATABASE_VERSION` (Zeile 2) kann eine Versionierung der Datenbank vorgenommen werden. Ist es nun beispielsweise nach einem Update der App notwendig, eine Änderung am Schema durchzuführen, kann so eine neue Version definiert werden. Nun kann man abfragen, welche Version die Datenbank hat und ggf. die notwendigen Änderungen am Schema durchführen. Dieses Konzept unterstützt den Entwickler bei Versionsupdates der App und erlaubt das Beibehalten der Daten eines Benutzers.

In der Methode `onCreate` definiert man das Schema der Datenbank, erzeugt alle Tabellen und fügt die initialen Daten in die Datenbank ein. Im folgenden Listing werden beispielsweise die Tabellen für Bilder und Decks erzeugt (ab Zeile 13).

Zusätzlich kommt das Entwurfsmuster Data Access Objekt (DAO) zum Einsatz. Hiermit ist es möglich den Zugriff auf die Datenquelle zu abstrahieren. Dies erleichtert den Wechsel der Datenquelle, ohne weitreichende Änderungen durchführen zu müssen. Es ist aber vor allem sinnvoll, um wiederkehrende Operationen, wie das Finden eines Datensatzes anhand der ID, an einer Stelle zu konzentrieren, sodass bei möglichen Änderungen an der Abfrage an die Datenbank auch nur diese eine Stelle geändert werden muss.

Zur Umsetzung dieses Entwurfsmusters definieren wir für jeden Entitätstypen ein Interface mit den jeweils benötigten Methoden. Anschließend leiten wir von diesem Interface eine Klasse ab, die diese Methoden für die eingesetzte SQLite-Datenbank implementiert. Ein Beispiel dafür ist in Abbildung 6 dargestellt.

```java
public class MySQLiteOpenHelper extends SQLiteOpenHelper {

    private static final Integer DATABASE_VERSION = 1;
    private static final String DATABASE_NAME = "flashcards.sqlite";

    public MySQLiteOpenHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }


    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("CREATE TABLE " + PictureDB.TABLE_NAME + " ("
                + PictureDB.ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
                + PictureDB.PICTURE_NAME + " TEXT,"
                + PictureDB.PICTURE_MIME_TYPE + " TEXT,"
                + PictureDB.PICTURE_SIZE + " TEXT);");

        db.execSQL("CREATE TABLE " + DeckDB.TABLE_NAME + " ("
                + DeckDB.ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
                + DeckDB.DECK_NAME + " TEXT, "
                + DeckDB.DECK_DESCRIPTION + " TEXT,"
                + DeckDB.DECK_PICTURE_ID + " INTEGER REFERENCES " + PictureDB.TABLE_NAME + ");");

        // [...]
    }

}
```

Listing 2: Klasse MySQLiteOpenHelper

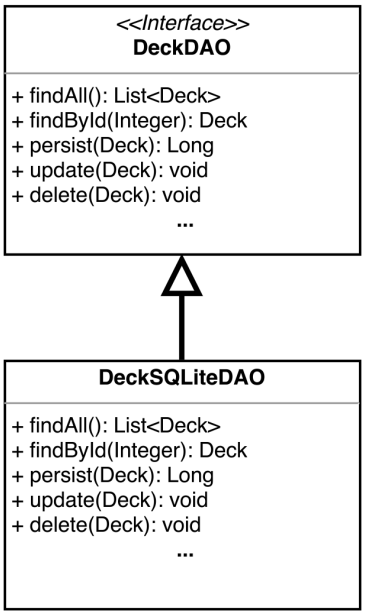

Abbildung 6: Data Access Object am Beispiel der Klasse Deck