Lern-App

Appdokumentation

Thema des Projektes: Entwicklung einer Java-Applikation in Android mit der Möglichkeit, Sprachen zu lernen

Autor: Daniel Ermilov

Inhaltsverzeichnis

1 Einleitung

1.1 Ziel und Zweck

Diese Anwendung wurde entwickelt, um Benutzer beim Erlernen verschiedener Sprachen zu unterstützen.

1.2 Arbeitsumgebung

Die genutzte Entwicklungsumgebung ist Android Studio, eine von Google entwickelte Plattform, die das Erstellen und Designen von Apps vereinfacht. Android Studio beinhaltet zahlreiche Emulatoren, welche die Testmöglichkeit auf diversen Geräten ermöglichen. In diesem Rahmen wurde ein neuer Workspace sowie einige Klassen erstellt.

2 Überblick

2.1 Systemarchitektur

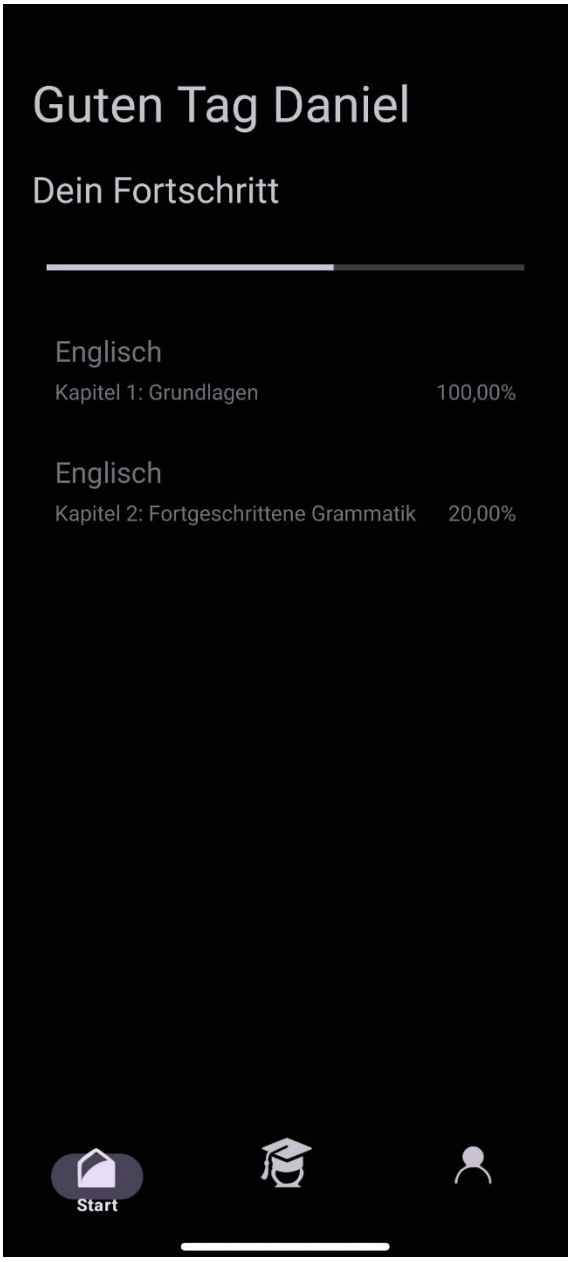
Die Lern-App nutzt die in Android integrierte Root-Datenbank (SQLite), um Aufgaben lokal auf dem Gerät des Benutzers zu speichern. Diese Datenbank ermöglicht es der App, die Daten dauerhaft zu speichern, so dass die Aufgaben auch nach dem Schließen oder Neustarten der App verfügbar bleiben.

2.2 Programmiersprachen

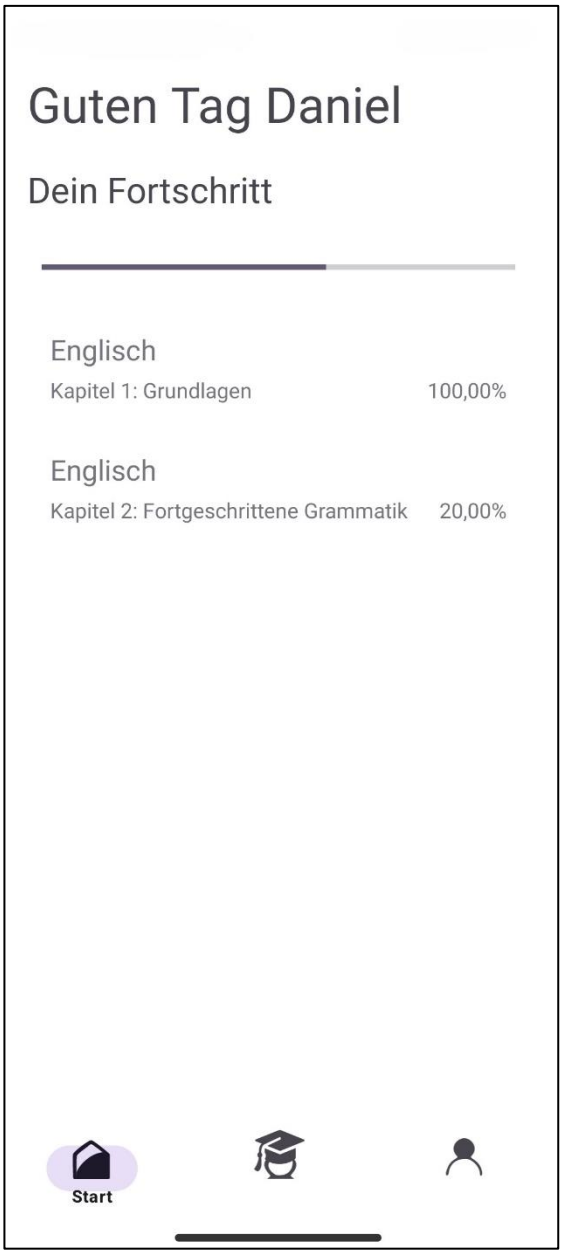
Die Lern-App wurde in Java programmiert und verwendet XML für das Design der Benutzeroberfläche. SQL wird verwendet, um mit der Datenbank zu interagieren und Daten abzufragen oder zu ändern.

2.3 Aufbau

Diese Anwendung verfügt über 2 Themes. Diese Themes wurden erstellt, um die App zu gestalten. Ein Theme wird für den Lightmode und das andere für den Darkmode verwendet. Android wählt automatisch das Theme aus, in dessen Modus sich das System befindet.



Darkmode



Lightmode

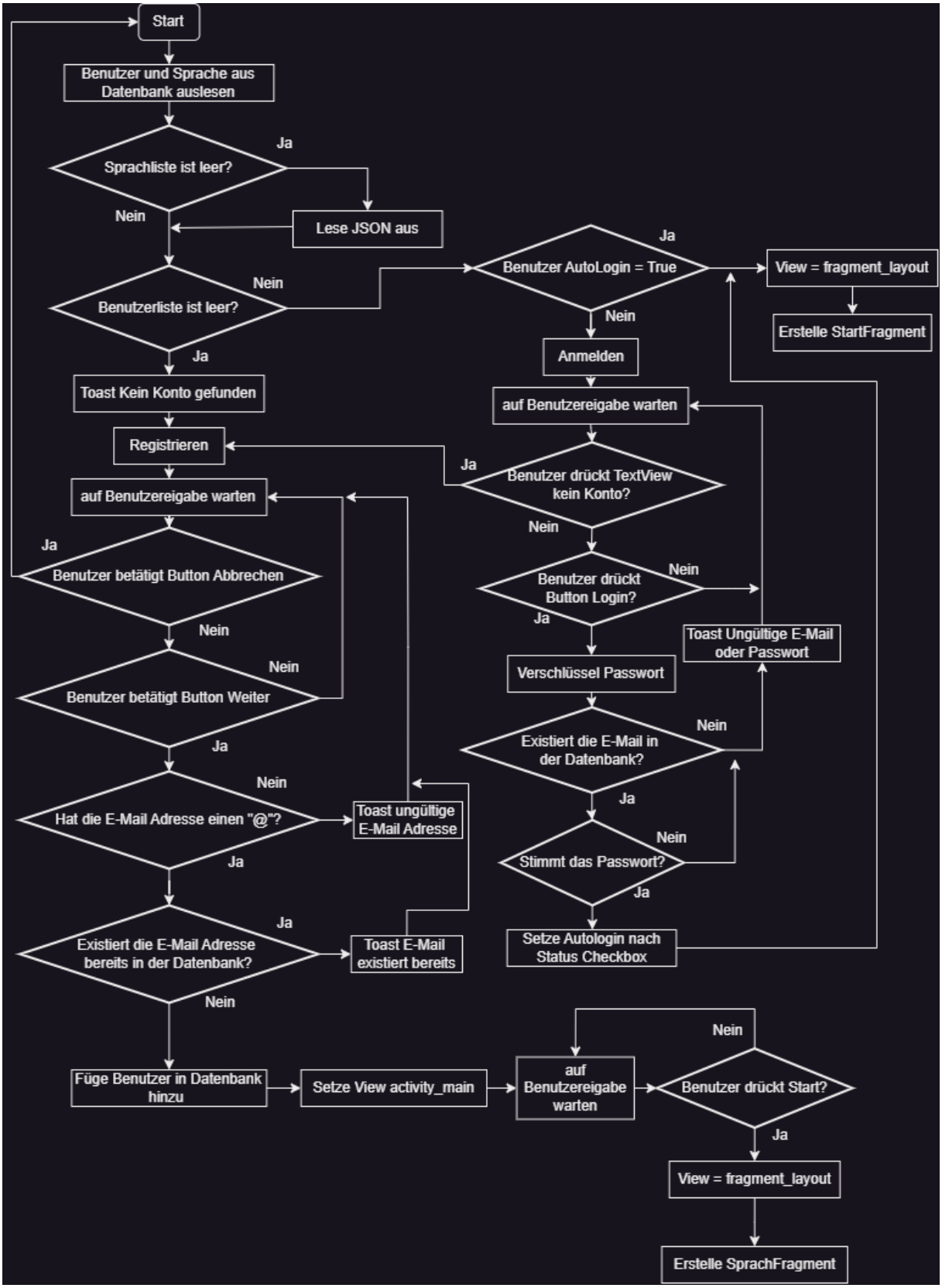
2.4 Notwendige Einstellungen

Bevor eine Room-Datenbank eingefügt werden kann, müssen einige Abhängigkeiten vorgenommen werden. Diese müssen in build.gradle (Module) unter dependencies eingetragen werden. Room ist eine Android Bibliothek, die eine zusätzliche Struktur über SQLite zur Verfügung stellt und somit die Arbeit mit Datenbanken erleichtert.

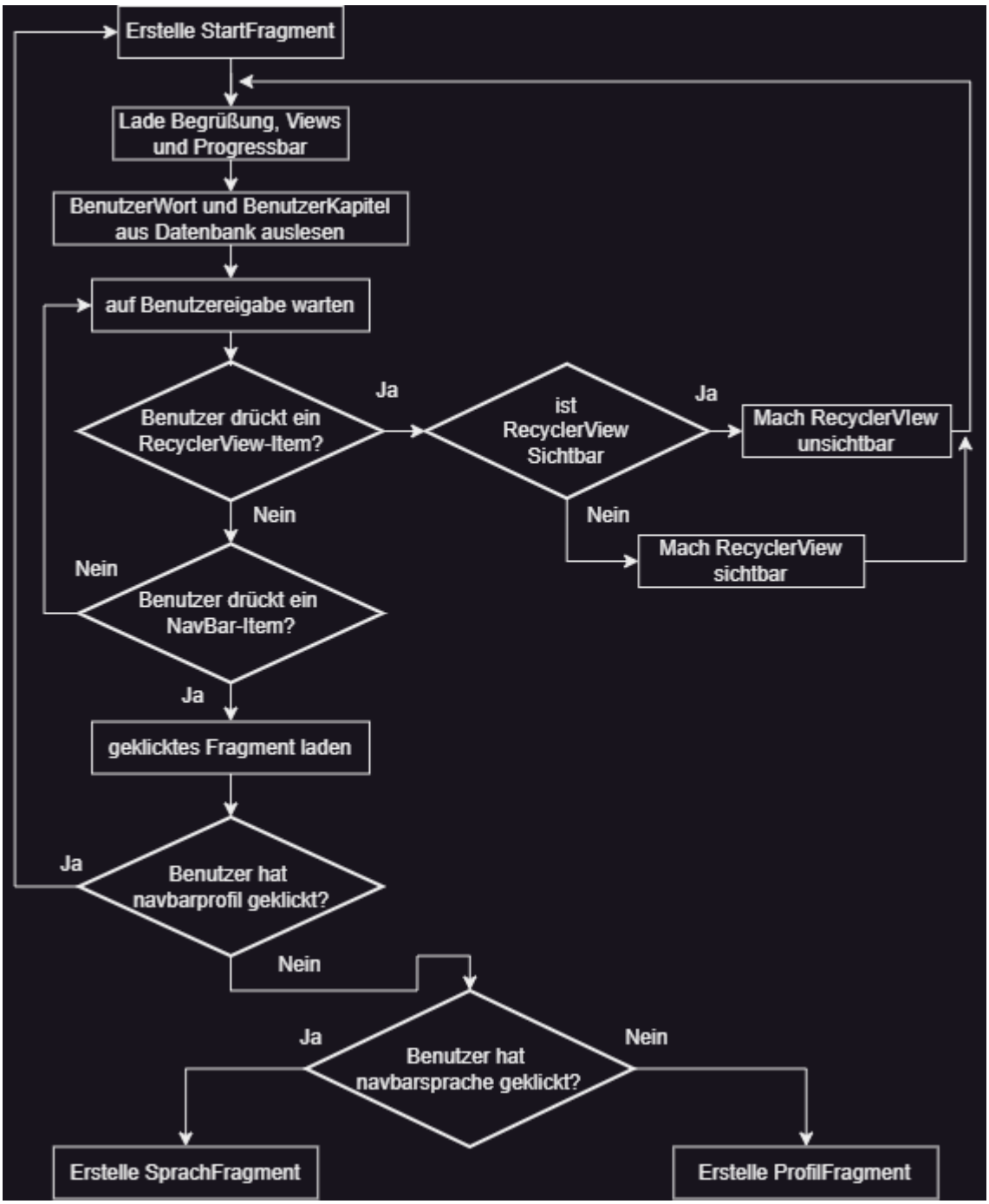
```
34    dependencies {
35        implementation("androidx.room:room-runtime:2.6.1")
36        annotationProcessor("androidx.room:room-compiler:2.6.1")
```

2.5 Ablaufdiagramm

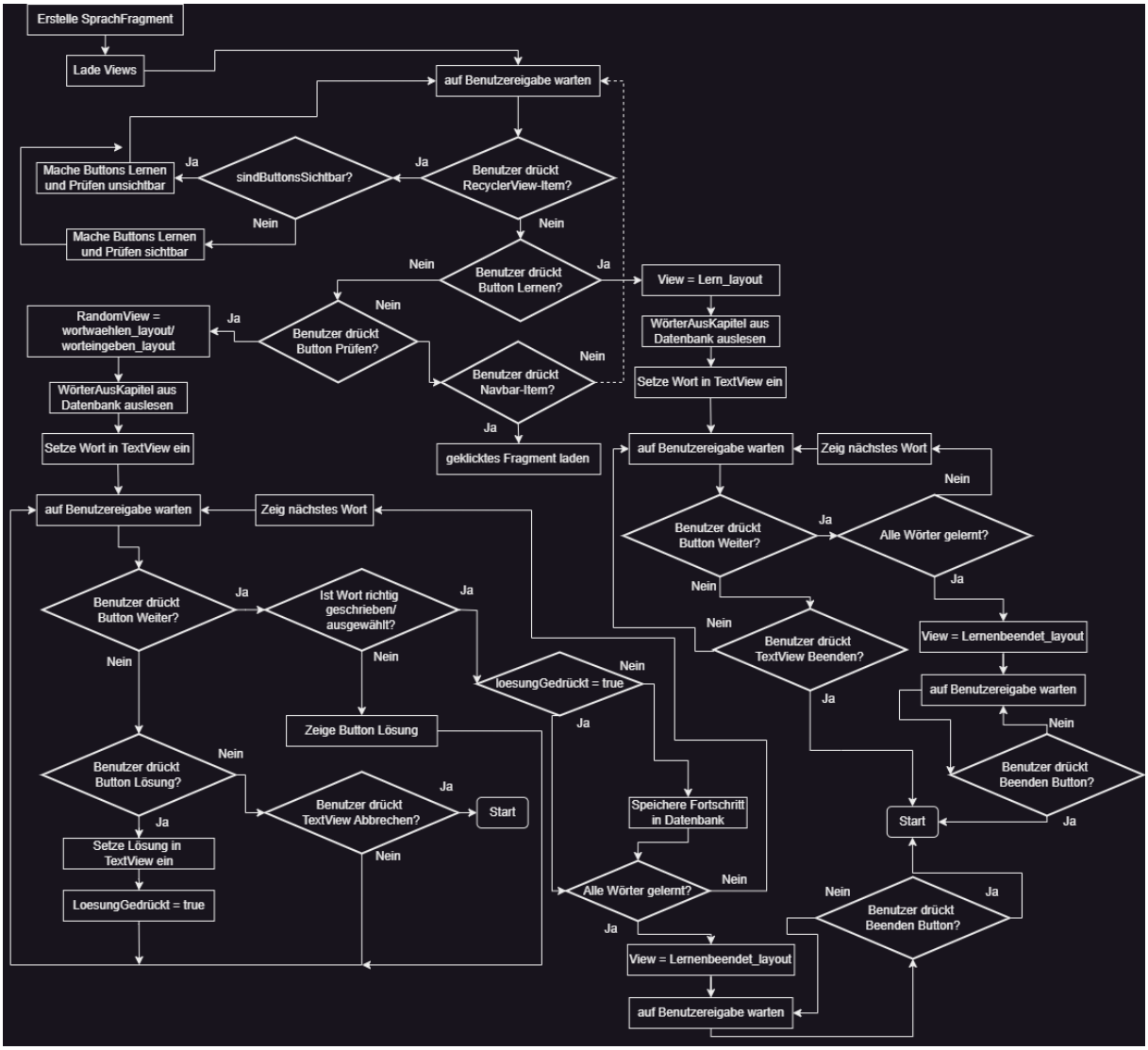
2.5.1 Ablaufdiagramm Start



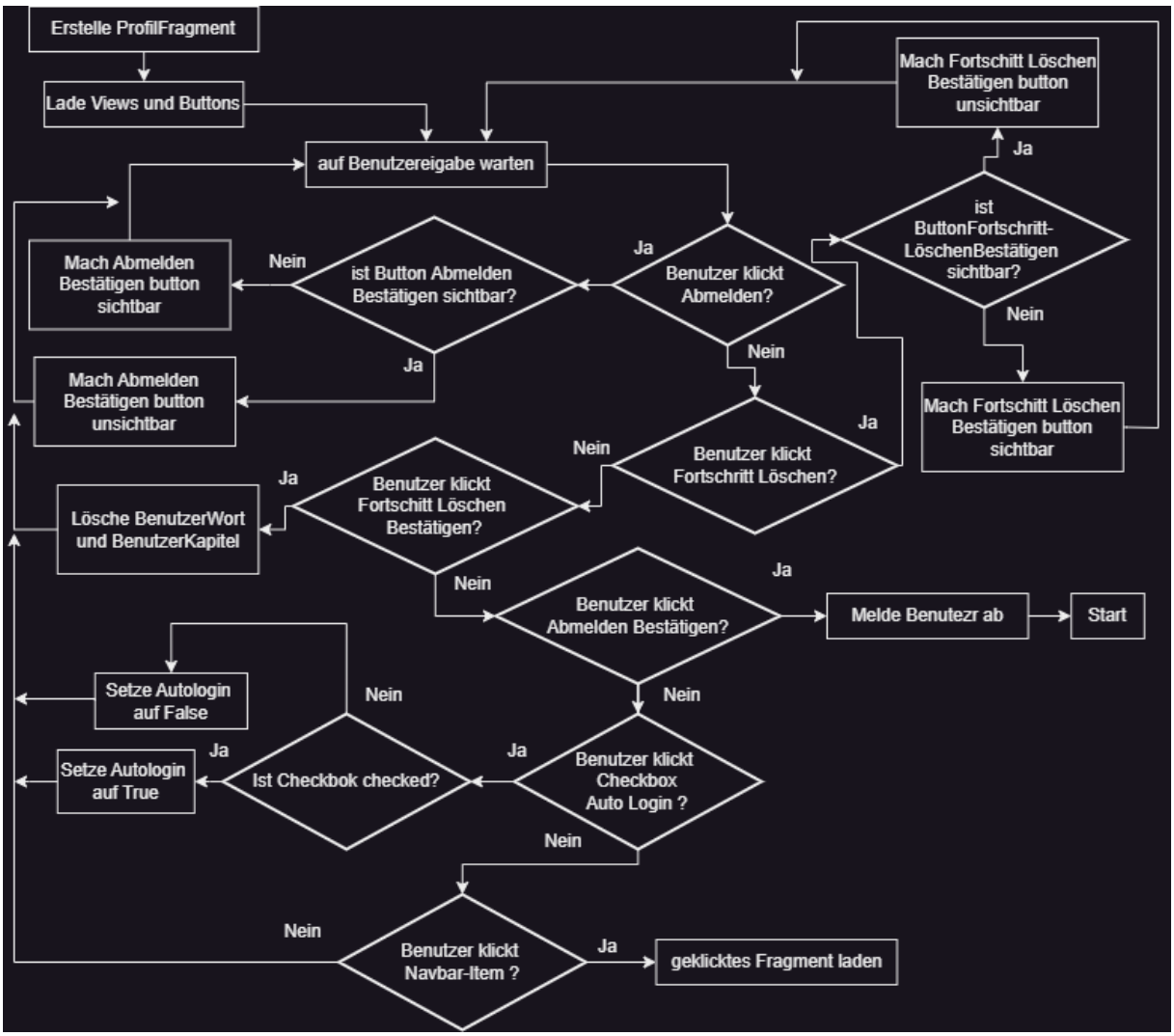
2.5.2 Ablaufdiagramm StartFragment



2.5.3 Ablaufdiagramm SprachFragment



2.5.4 Ablaufdiagramm ProfilFragment



2.6 Schnittstellen

Technische Schnittstellen

- Notebook mit Windows 11
- Android Studio als Entwicklungsumgebung
- SDK 24 (Android 7.0 / Nougat)
- Microsoft 365 zum Erstellen von Dokumentation und Datenbankstruktur
- www.draw.io zum Erstellen von Ablaufdiagrammen
- Inkscape zum Erstellen eines SVG-Hintergrundbilds

3 Implementieren von Code

Um die Eigenschaften unserer Objekte in Java zu spezifizieren, werden Klassen erstellt. Klassen ermöglichen eine logische Strukturierung und Organisation des Codes. Sie fördern die Wiederverwendbarkeit des Codes. Dadurch wird Redundanz vermieden und der Code wird wartbarer. Die Verwendung von Klassen erleichtert das Testen des Codes. Einzelne Klassen können isoliert getestet werden, um sicherzustellen, dass sie wie erwartet funktionieren.

3.1 Erstellen der Datenbank Struktur

Die Datenbank der App wurde als relationale Datenbank entworfen und besteht aus mehreren Tabellen, die durch Fremdschlüssel miteinander verknüpft sind. Dadurch wird eine effiziente Verwaltung der Benutzer-, Kapitel- und Wortdaten ermöglicht, sowie die Nachverfolgung des individuellen Lernfortschritts. Die Struktur der Datenbank wird nachfolgend beschrieben:

3.1.1 Benutzer

Diese Tabelle speichert die Benutzerinformationen. Sie enthält die Felder ID (Primärschlüssel), Name (Text), E-Mail (Text), Passwort (Text) und AutoLogin (Ja/Nein).

3.1.2 Kapitel

Jedes Kapitel ist in dieser Tabelle gespeichert und wird durch die Felder ID (Primärschlüssel), Titel (Text), Beschreibung (Text), Kapitelnummer (Zahl) und Sprache_ID (Fremdschlüssel) beschrieben.

3.1.3 Sprache

Diese Tabelle enthält alle Sprachen, die in der App zur Verfügung stehen. Sie besteht aus den Feldern ID (Primärschlüssel) und Name (Text).

3.1.4 Wort

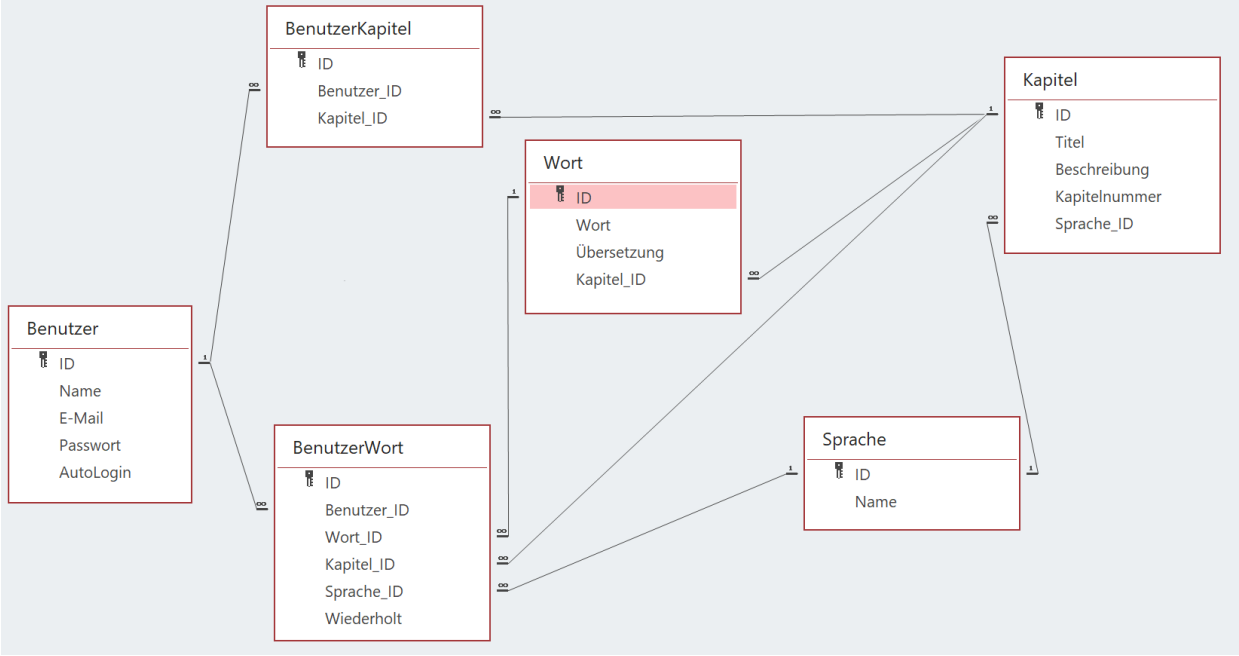
Die Wort-Tabelle speichert alle Wörter, die in einem bestimmten Kapitel vorkommen. Zu den Feldern gehören ID (Primärschlüssel), Wort (Text), Übersetzung (Text) und Kapitel_ID (Fremdschlüssel). Dabei werden die Übersetzungen als ein String gespeichert, der durch Kommas getrennte Übersetzungen enthält.

3.1.5 BenutzerKapitel

Diese Tabelle stellt eine Verbindung zwischen den Benutzern und den Kapiteln her. Sie enthält die Felder ID (Primärschlüssel), Benutzer_ID (Fremdschlüssel) und Kapitel_ID (Fremdschlüssel), die es ermöglichen, zu verfolgen, welche Kapitel ein Benutzer bereits bearbeitet hat. Hierdurch kann der Fortschritt des Lernprozesses eines Benutzers gespeichert und abgerufen werden.

3.1.6 BenutzerWort

Die BenutzerWort-Tabelle verbindet Benutzer mit Wörtern und speichert Informationen darüber, wie oft ein Wort bereits wiederholt wurde. Zu den Attributen gehören ID (Primärschlüssel), Benutzer_ID (Fremdschlüssel), Wort_ID (Fremdschlüssel), Kapitel_ID (Fremdschlüssel), Sprache_ID(Fremdschlüssel) und Wiederholt (Ja/Nein).



3.2 Erstellung der benötigten Klassen

Für die Implementierung der Lern-App wurde eine Reihe von Klassen erstellt, die die verschiedenen Entitäten und Datenbankstrukturen der Anwendung repräsentieren. Diese Klassen bilden die Grundlage für die Speicherung, Verarbeitung und Anzeige der Daten in der App.

3.2.1 MainActivity

Die MainActivity wird automatisch generiert und fungiert als Haupt-Einstiegspunkt der App. Sie steuert das Layout und die initiale Logik der App. In dieser Klasse werden die Adapter und Fragmente initialisiert sowie die grundlegenden Navigationsabläufe verwaltet.

3.2.2 Benutzer

Diese Klasse repräsentiert die Benutzer der App und speichert Informationen wie den Namen, die E-Mail und andere benutzerbezogene Daten. Sie ist notwendig, um den Fortschritt jedes Benutzers individuell nachzuverfolgen und persönliche Informationen zu speichern.

3.2.3 Sprache

Diese Klasse speichert die verschiedenen Sprachen, die in der Lern-App unterstützt werden. Sie hilft dabei, die ausgewählte Sprache eines Benutzers festzulegen und zu speichern.

3.2.4 Kapitel

Die repräsentiert einzelne Lernkapitel, die innerhalb einer Sprache zur Verfügung stehen. Jedes Kapitel enthält Informationen wie den Titel und die zugehörigen Wörter.

3.2.5 Wort

Wort speichert die Wörter, die in den Kapiteln verwendet werden. Da Wörter in verschiedenen Sprachen mehrere Übersetzungen haben können, werden alle Übersetzungen als ein einzelner String gespeichert, wobei die verschiedenen Übersetzungen durch Kommas getrennt sind. Dies ermöglicht es, die Übersetzungen kompakt zu speichern und später bei Bedarf zu verarbeiten.

3.2.6 BenutzerKapitel

Diese Klasse dient dazu, die Beziehung zwischen einem Benutzer und den Kapiteln, die er bearbeitet, darzustellen. Sie speichert, welche Kapitel von welchem Benutzer bereits bearbeitet wurden, um den Lernfortschritt festzuhalten.

3.2.7 BenutzerWort

Diese Klasse repräsentiert die Beziehung zwischen einem Benutzer und den Wörtern, die er gelernt hat. Sie speichert, welche Wörter ein Benutzer gelernt oder wiederholt hat, um personalisierte Lernerfahrungen zu ermöglichen.

3.2.8 AppDatenbank

Die AppDatenbank stellt die Datenbank der Anwendung dar. Sie enthält alle notwendigen Entitäten und ermöglicht über die DAOs den Zugriff auf die Datenbank. Diese Klasse dient als zentraler Speicherort für alle gespeicherten Daten der Anwendung.

3.2.9 Adapter und Fragmente

Die Adapter und Fragmente wurden erstellt, um die Benutzeroberfläche dynamisch mit Daten aus der Datenbank zu versorgen. Die Adapter sind dafür verantwortlich, Daten in RecyclerViews anzuzeigen, während die Fragmente für die Darstellung der unterschiedlichen Ansichten und Layouts der App verwendet werden.

3.2.10 Data Access Objects (DAO)

Die DAOs ermöglichen den Zugriff auf die Datenbank und definieren alle notwendigen Methoden für das Abrufen, Einfügen, Aktualisieren und Löschen von Daten. Jede Entität besitzt ihr eigenes DAO, um eine klare Trennung der Datenbankoperationen zu gewährleisten.

3.3 Erstellung des Hintergrundbildes

Um der App ein individuelles Design zu verleihen, wurde ein Hintergrundbild erstellt. Dafür wurde das Programm **Inkscape** verwendet, um eine **SVG-Datei** (Scalable Vector Graphics) zu erzeugen. Der Vorteil einer SVG-Datei liegt darin, dass sie verlustfrei skaliert werden kann, wodurch der Hintergrund auch auf hochauflösenden Bildschirmen nicht pixelig wird.
Obwohl dies nicht zwingend notwendig war, entschied sich der Autor dafür, das Design der App zu personalisieren, um eine hohe visuelle Qualität auf allen Geräten zu gewährleisten.

3.4 Implementierung des Logins

In der Anwendung wurde ein Login-System implementiert, um Benutzerdaten sicher zu verwalten und den Zugriff auf personalisierte Funktionen zu ermöglichen. Die Authentifizierung erfolgt über die E-Mail-Adresse und ein verschlüsseltes Passwort, das in der Datenbank gespeichert wird. Dies stellt sicher, dass die Benutzerdaten geschützt sind und jeder Benutzer nur auf seine eigenen Daten zugreifen kann.

Bei erfolgreicher Anmeldung erhält der Benutzer Zugriff auf seine personalisierten Inhalte und kann seine Fortschritte in der App verfolgen. Das Login-System sorgt somit für eine sichere und individuelle Benutzererfahrung.

3.5 Implementierung der Registrierung

Zusätzlich zum Login-System wurde ein Registrierungsfeld implementiert, das es neuen Benutzern ermöglicht, sich in der App zu registrieren. Bei der Registrierung müssen folgende Informationen eingegeben werden:

- Name: Der Name des Benutzers.
- E-Mail-Adresse: Diese wird als eindeutiger Identifikator verwendet und überprüft, ob ein "@"-Zeichen vorhanden ist.
- Passwort: Das Passwort muss zweimal eingegeben werden, um sicherzustellen, dass es korrekt ist.

Alle eingegebenen Daten werden in der Datenbank gespeichert. Das Passwort wird dabei, wie beim Login, verschlüsselt abgelegt. Nach erfolgreicher Registrierung kann der Benutzer direkt mit der App fortfahren.

3.6 Implementierung der Navigationsleiste (NavBar)

Um die Navigation zwischen den verschiedenen Fragmenten der App zu ermöglichen, wurde eine Navigationsleiste (NavBar) erstellt. Diese Leiste befindet sich am unteren Rand der App und bietet schnellen Zugriff auf die Hauptfunktionen:

- Start: Führt den Benutzer zum StartFragment, wo die Begrüßung und der Fortschritt angezeigt werden.
- Sprachen: Öffnet das SprachFragment, in dem der Benutzer zwischen den verschiedenen Sprachen wählen kann, die gelernt und geprüft werden sollen.
- Profil: Öffnet das ProfilFragment, in dem die Benutzerdaten angezeigt werden und Optionen wie Abmelden oder das Löschen des Fortschritts zur Verfügung stehen.

Um die Navigation zu verwalten, wurde ein Ordner angelegt res/menu, der die für die Navigation benötigten XML-Dateien enthält. In diesem Ordner wird die Struktur der Navigationsleiste definiert, einschließlich der Icons und Bezeichnungen der einzelnen Menüpunkte.

Die NavBar sorgt für eine intuitive Benutzerführung und ermöglicht es dem Benutzer, schnell zwischen den wichtigsten Bereichen der App zu wechseln.

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
        <item
            android:id="@+id/nav_start"
            android:title="Start"
            android:icon="@drawable/navbarhaus"/>
    <item
        android:id="@+id/nav_sprache"
        android:title="Sprachen"
        android:icon="@drawable/navbarsprache"/>
        <item
            android:id="@+id/nav_profil"
            android:title="Profil"
            android:icon="@drawable/navbarprofil"/>
    </menu>
```

3.7 Implementierung der Fragmente

In der App wurden drei Fragmente erstellt, um die Benutzeroberfläche zu strukturieren und unterschiedliche Inhalte anzuzeigen:

3.7.1 StartFragment

Dieses Fragment dient als Startbildschirm der App. Hier wird der Benutzer basierend auf der aktuellen Uhrzeit begrüßt (z.B. "Guten Morgen Benutzername" oder "Guten Abend Benutzername"). Zusätzlich wird der Lernfortschritt des Benutzers angezeigt, um ihm einen schnellen Überblick über seine Leistungen zu geben.

3.7.2 SprachFragment

In diesem Fragment werden alle verfügbaren Sprachen angezeigt, die der Benutzer lernen und prüfen kann. Sobald eine Sprache ausgewählt wird, erscheinen zwei Buttons:

- Lernen: In diesem Modus wird dem Benutzer ein Wort zusammen mit seiner Übersetzung angezeigt, um das Verständnis zu erleichtern.
- Prüfen: Hier muss der Benutzer entweder das richtige Wort eintippen oder aus einer Liste von Wörtern das korrekte Wort auswählen. Dieser Modus dient zur Überprüfung und Festigung des Gelernten.

Wenn das Wort richtig eingegeben/ausgewählt wurde, wird der Fortschritt gespeichert und im StartFragment (Startbildschirm) angezeigt.

3.7.3 ProfilFragment

Dieses Fragment zeigt die Benutzerdaten an, einschließlich der E-Mail-Adresse, des Namens und des Auto-Login-Status (Boolean-Wert). Zudem gibt es hier die Möglichkeit, sich abzumelden oder den Lernfortschritt zu löschen. Wenn einer dieser beiden Aktionen gewählt wird, erscheint unter dem entsprechenden Button ein Bestätigungsbutton, der gedrückt werden muss, um die Aktion abzuschließen.

3.8 Implementierung des JSON-Inputs

In der App wurde JSON (JavaScript Object Notation) verwendet, um die Sprachen, Kapitel und Wörter effizient zu verarbeiten. Diese Datenstruktur ermöglicht eine einfache Verwaltung und Speicherung der Inhalte in einer kompakten und leicht lesbaren Form.

Durch den Einsatz von JSON konnten die Sprachen, Kapitel und zugehörigen Wörter flexibel und strukturiert in die App integriert werden. Dies erleichtert die Erweiterbarkeit und das Laden neuer Lerninhalte ohne komplexe Anpassungen im Code.

3.9 Implementierung der Lern- und Prüfungsansicht

In der App wurden zwei Ansichten zur Unterstützung des Lernprozesses implementiert: die Lernansicht und die Prüfungsansicht.

3.9.1 Lernansicht (LernenView)

In dieser Ansicht wird dem Benutzer ein Wort zusammen mit seiner entsprechenden Übersetzung angezeigt. Diese Funktion dient dazu, die Wörter und deren Bedeutungen visuell zu lernen und zu festigen.

3.9.2 Prüfungsansicht (PrüfenView)

Die Prüfungsansicht bietet zwei verschiedene Varianten zur Überprüfung des gelernten Wissens:

- Worteingabe: Der Benutzer muss das richtige Wort manuell eintippen, um sein Wissen zu testen.
- Wortauswahl: Dem Benutzer werden drei mögliche Wörter angezeigt, von denen nur eins korrekt ist. Der Benutzer muss das richtige Wort auswählen.

Wenn das Wort falsch eingegeben/ausgewählt wurde, erscheint ein „Lösung“ Button. Dieser kann gedrückt werden, um die Lösung anzuzeigen. Wenn das Wort ohne Lösung eingegeben/ausgewählt wurde, dann wird der Fortschritt gespeichert, sonst nicht.

Diese beiden Ansichten bieten dem Benutzer eine umfassende Möglichkeit, sowohl das Lernen als auch das Überprüfen der Wörter effektiv zu gestalten.

3.10 Implementierung von Animationen

In der App wurden Animationen hinzugefügt, um die Benutzererfahrung zu verbessern und Übergänge visuell ansprechender zu gestalten. Diese Animationen kommen insbesondere beim Aufploppen von Buttons und der RecyclerView zum Einsatz, um eine dynamische Interaktion zu ermöglichen. Im Verzeichnis res/anim wurden Animationsdateien erstellt, die die gewünschten Effekte (z.B. Fade-In, Scale) beschreiben und werden in den entsprechenden Methoden der Views, wie z.B. beim Anzeigen von Buttons oder beim Laden der RecyclerView angewendet.

Durch die Verwendung dieser Animationen wird der Übergang von UI-Elementen wie Buttons oder der RecyclerView weicher und bietet dem Benutzer eine interaktivere und visuell ansprechendere Erfahrung.

Die Animation Fade In soll verwendet werden, wenn ein Element von unsichtbar auf sichtbar geht. Andersherum Fade Out.

Animation Fade In:

```
<?xml version="1.0" encoding="utf-8"?>
<alpha xmlns:android="http://schemas.android.com/apk/res/android"
    android:fromAlpha="0.0"
    android:toAlpha="1.0"
    android:duration="300" />
```

Animation Fade Out:

```xml
<?xml version="1.0" encoding="utf-8"?>
<alpha xmlns:android="http://schemas.android.com/apk/res/android"
    android:fromAlpha="1.0"
    android:toAlpha="0.0"
    android:duration="300" />
```

4 Nützliche Links

Fragments | Android Developers

Dynamische Listen mit RecyclerView erstellen | Views | Android Developers

RecyclerView.Adapter | Android Developers

RecyclerView.ViewHolder | Android Developers

Daten mit Room in einer lokalen Datenbank speichern | Android Developers

Stile und Designs | Views | Android Developers

5 Glossar

A	Adapter	Eine Schnittstelle zwischen einer Datenquelle und einer UI-Komponente (z.B. RecyclerView), die die Darstellung von Daten steuert.
D	DAO	Data Access Object = Ein Objekt, welches für den Zugriff auf Datenquellen erstellt wird. Es ist typischer Weise austauschbar.
E	Entity	In Room ist ein Entity eine Java-Klasse, die als Tabelle in der SQLite-Datenbank fungiert.
F	Fragment	Ein Fragment ist eine wiederverwendbare Komponente in Android-Apps, die als Teil einer Benutzeroberfläche fungiert.
J	Java	Java ist eine Objektorientierte Programmiersprache und eine eingetragene Marke des Unternehmens Sun Microsystems, welches 2010 von Oracle aufgekauft wurde.
R	RecyclerView	Eine RecyclerView ist ein felxibles und effizientes View-Widget in Android, das Daten in Form einer scrollbaren Liste darstellt.
	Room	Eine Abstraktionsschicht über SQLite, die das Arbeiten mit Datenbanken in Android erleichtert.

S	SQLite	Eine relationale Datenbank-Engine, die in Android zum Speichern von Daten lokal auf dem Gerät verwendet wird. Sie verwendet die Programmiersprache SQL
T	TextView	Ein TextView ist ein einfaches UI-Element in Android, das Texte anzeigt.
	Thread	Ein Thread ist eine parallele Ausführungsreihe bei der Programmausführung. Sie beeinträchtigt die Anwendung nicht.
W	Workspace	Ein Workspace bezeichnet die Arbeitsumgebung in der Entwicklungsumgebung.