Praxis der Softwareentwicklung

Fridget

Entwurf

Yunjia Chen, Jasmin Jat, Min Hye Park, Alina Shah, Lisa Wang

24. Juni 2018

Betreuung: Erik Burger, Sandro Koch

IPD

Karlsruher Institut für Technologie

Inhaltsverzeichnis

1 Einleitung

Dies ist das Dokument für den Entwurf der Applikation Fridget – entstanden im Rahmen des Softwarepraktikums PSE im Sommersemester 2018.
Es wird zunächst der Grobentwurf vorgestellt, welcher sich in Systemarchitektur, Systemkomponenten und deren Beschreibung unterteilt.
Der Feinentwurf wird in Abschnitt 3 beschrieben, wobei eine Aufteilung in "Klassen des Clients", "RESTfulAPI" und "Klassen des Servers" erfolgt. Ebenfalls ist ein Klassendiagramm für die Client bzw. Serverseite und für die einzelnen Packages zu finden.
Die Verwendung der Daten wird in Abschnitt 4 erläutert.
Ebenso wichtig ist die Dynamik der Applikation, welche im nachfolgenden Abschnitt 5 in mehreren Sequenz- und Aktivitätsdiagrammen dargestellt und beschrieben wird.
Im darauf folgenden Abschnitt 6 wird auf die Änderungen zum Pflichtenheft eingegangen und die Begründung für diese Entscheidung erläutert.
Im Anhang ist die Ansicht des gesamten Klassendiagramms zu sehen.
Um Fachbegriffe zu erläutern, befindet sich im letzten Abschnitt ein Glossar.

2 Grobentwurf

2.1 Systemarchitektur

Die Applikation Fridget bedient sich der Server-Client Architektur. Es gibt also einen zentralen Server, der als Bindeglied zwischen beliebig vielen Clients fungiert.
Der Server verarbeitet alle Anfragen, die von den unterschiedlichen Clients gestellt werden. Er verwaltet die Datenbank und stellt alle Daten bereit, worauf die Clients zugreifen können.
Jeder Benutzer, welcher die Applikation Fridget benutzt, hat eine feste User-ID, die lokal auf dem Gerät gespeichert wird. Damit stellt jeder Benutzer einen Client dar, welcher mit seiner User-ID Anfragen an den Server schicken kann. Die Architektur der einzelnen Seiten wird im Folgenden beschrieben.

2.1.1 MVVM

Für die Client-Seite verwenden wir das MVVM (Model-View-View-Model) Architektur-Muster mit Data Binding. Dabei wird die Darstellung in View, die Eingabeverarbeitung in Viewmodel und die Datenhaltung in Model aufgeteilt. Zusätzlich dient der Service noch als direkter Vermittler zwischen der Client- und Server-Seite.
Dieses Muster hat zwei wesentliche Vorteile.
Erstens gibt es keine Abhängigkeit zwischen der View und dem Model. Die Daten, die für die View benötigt werden, werden nämlich im Viewmodel bereitgestellt.
Zweitens ist das ViewModel nicht von der View abhängig. Durch Data Binding kann ohne Referenz auf die View der Zustand der View im Viewmodel verwaltet werden.
Es ist also eine lose Kopplung vorhanden, welche die Testbarkeit verbessert.

2.1.2 MVC

Auf der Server-Seite wird das MVC (Model-View-Controller) Architektur-Muster eingesetzt. Dabei ist in unserem Falle die Client-Seite die View. Model ist für die Datenhaltung zuständig und der Controller steuert zwischen View, Model sowie Repository.
Sobald auf Client-Seite eine Interaktion mit dem Benutzer und der Applikation geschieht, wird dies vom Service der Client-Seite weiter an den Controller geleitet. Dieser kann die Veränderungen dann durchführen lassen.

2.2 Systemkomponenten

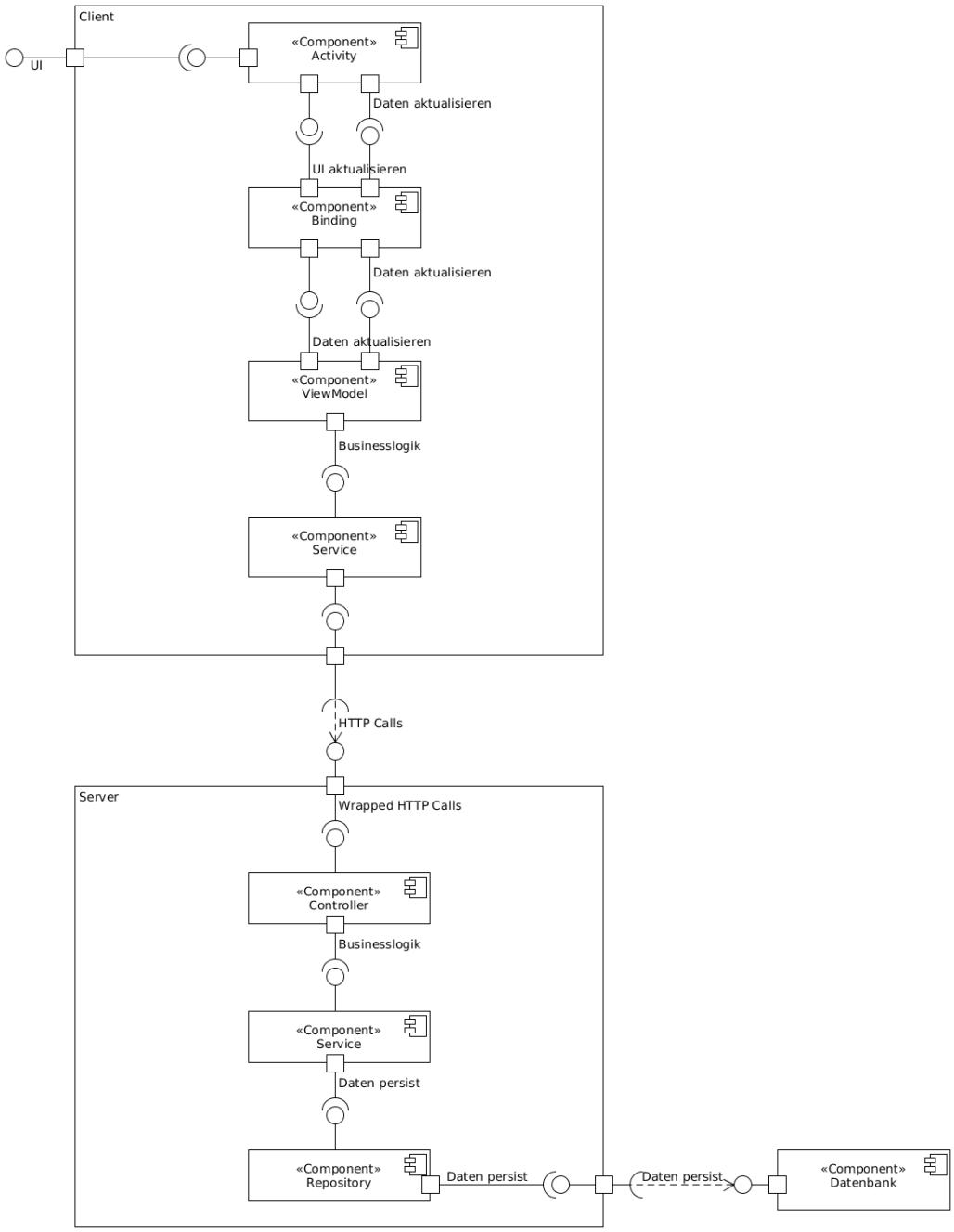

Abbildung 2.1: Systemkomponenten

2.3 Komponentenbeschreibung

2.3.1 Client

2.3.1.1 MVVM-Architektur

Model
Model-Klassen kapseln die Appdaten ab. Es handelt sich um eine Datenzugriffschicht, wo die Daten gehalten und zum Benutzen aufbereitet werden.

View
Zu jeder Activity gehört eine View, die als graphische Benutzeroberfläche dient. Sie stellt eine rechteckige Fläche auf dem Bildschirm dar und ist verantwortlich fürs Event-Handling. Die View ist das, was der Benutzer sieht und mit dem er interagieren kann (Buttons, Textfelder, etc.).

Viewmodel
Viewmodel definiert die in der View angezeigten Inhalte und dient zur Eingabeverarbeitung. Sie kümmern sich darum, Eigenschaften und Befehle zu implementieren und tauscht mit der View mittels Data Binding Daten aus. Außerdem sind Viewmodels dafür verantwortlich, die Views mittels Live Data über Zustandsänderungen zu benachrichtigen. Viewmodels kümmern sich also um die Geschäftslogik, die von den Views verwendet und angezeigt wird.

2.3.1.2 Activity

Die Activities stellen die Benutzerschnittstelle unserer App dar und kümmern sich um die Interaktionen mit unserer Benutzeroberfläche. In jeder Activity-Klasse befinden sich selbstverständlich die üblichen Methoden eines Activity-Lifecycles: onCreate() zum Erstellen, onStart() zum Starten, onPause() zum Pausieren , onResume() zum Fortsetzen, onStop() zum Stoppen und onDestroy() zum Zerstören der Activity.

2.3.1.3 Service

Ein Dienst ist eine Komponente, die ohne direkte Interaktion mit dem Benutzer im Hintergrund abläuft. Da der Service keine Benutzeroberfläche hat, ist er nicht an den Lebenszyklus einer Aktivität gebunden. Jede Methode innerhalb einer Service- Schnittstelle repräsentiert einen möglichen API-Aufruf. Es muss eine HTTP-Annotation (GET, POST usw.) haben, um den Anforderungstyp und die relative URL anzugeben. Der Rückgabewert umschließt die Antwort in einem Call-Objekt mit dem Typ des erwarteten Ergebnisses.

2.3.1.4 Live-Data

Live-Data sind beobachtbare data holder Klassen. Live-Data ist wie der Observer-Entwurfsmuster. Sie benachrichtigt den Observer-Objekt, in unserem Fall die View, wenn sich die Objekte verändern.

2.3.1.5 REST-Client Retrofit

Der REST-Client ist in unserem Fall die Retrofit-Bibliothek, die auf der Clientseite (Android) verwendet wird, um eine HTTP-Anforderung an die REST-API zu stellen.

Retrofit ist ein typsicherer HTTP-Client für Android und Java. Es ist eine Open-Source-Bibliothek, die die HTTP-Kommunikation vereinfacht, indem MRemote-APIs zu deklarativen, typsicheren Schnittstellen werden. Es macht es relativ einfach, JSON (oder andere strukturierte Daten) über einen REST-basierten Webservice abzurufen und hochzuladen. Es serialisiert die JSON-Antwort automatisch unter Verwendung eines POJO(Plain Old Java Object), das für die JSON-Struktur im Voraus definiert werden muss.

2.3.2 REST-API

Die REST-API definiert eine Reihe von Funktionen, mit denen Entwickler Anfragen ausführen und Antworten über HTTP-Protokoll wie GET und POST empfangen können.

2.3.3 Server

2.3.3.1 MVC-Architektur

Model
Das Model enthält Daten, die vom Controller gespeichert, geladen und geändert werden und vom View dargestellt werden.

Controller
Der Controller verwaltet die View und das Model. In unserem Fall implementiert der Controller eine REST-API, behandelt HTTP-Requests und gibt HTTP-Responses zurück.

Service
Die Service-Schicht trennt die Businesslogik aus dem Controller und behandelt spezifisches Transaktionsverhalten.

Repository
Die Repository-Schicht ist die unterste Schicht in unserer App. Sie reduziert den erforderlichen Code für die Implementierung von DAO (Data Access Object) für Persistenz. Unsere Repositories erweitern das JpaRepository, bieten CRUD-Funktionen und JPA-relevante Methoden an.

2.3.3.2 Framework

Für unsere RESTful Services verwenden wir Spring Boot mit JPA und Hibernate.
Spring Boot erleichtert die Entwicklung der Anwendungen per Convention over Configuration. Mit Hilfe einfacher Annotationen wird ein embedded Tomcat Webserver integriert, der REST-Services anbietet.
Die JPA (Java Persistence API) ist eine Schnittstelle für Java-Anwendungen, die die Zuordnung und die Übertragung von Objekten zu Datenbankeinträgen vereinfacht.
Hibernate ist ein Persistenz- und O-R-Mapping-Framework für Java. Es ermöglicht, POJOs in relationalen Datenbanken (MySQL in unserem Fall) zu speichern und aus entsprechenden Datensätzen wiederum Objekte zu erzeugen.

2.3.3.3 Datenbank

Wir verwenden MySQL als Datenbank. MySQL ist eine der am häufigsten benutzte, zuverlässige und relationale Datenbank. Sie beruht auf dem relationalen Datenbankmodell und speichert Daten in verschiedenen Tabellen, die untereinander verknüpft werden.

3 Feinentwurf

3.1 Klassen des Clients

3.1.1 Klassendiagramm

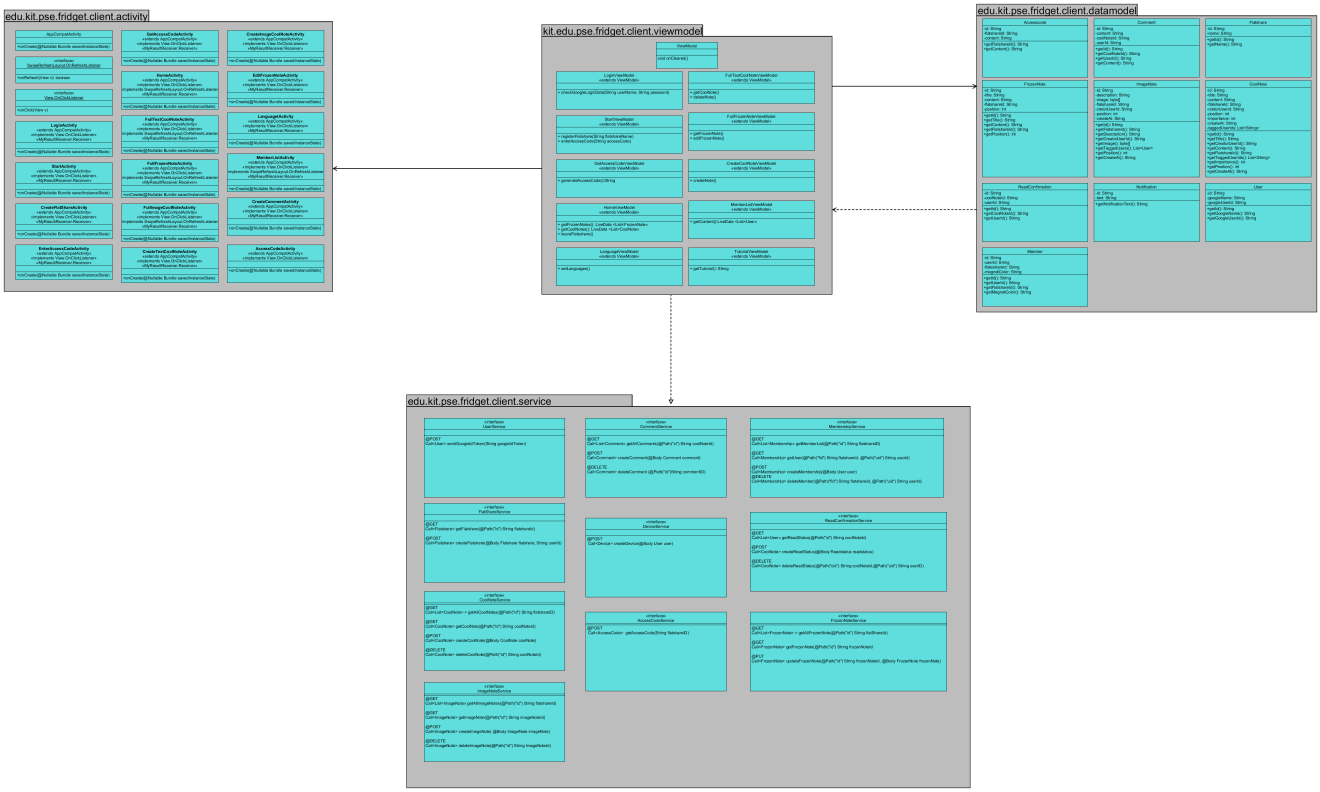

Abbildung 3.1: Klassen des Clients

3.1.2 package kit.edu.pse.fridget.client.activity

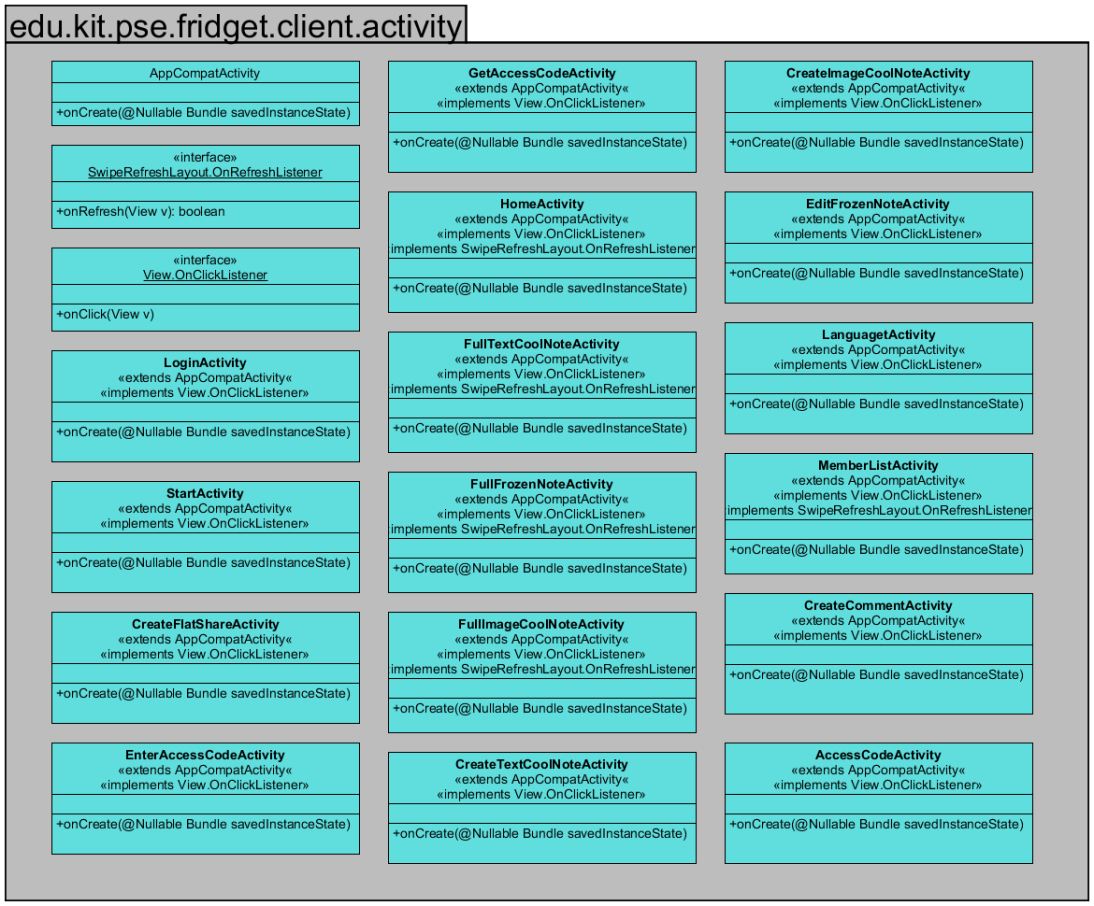

Abbildung 3.2: Klassen der Activities

3.1.2.1 `public class AppCompatActivity`

Beschreibung
Basisklasse für alle Activities

Methoden

- `public void onCreate(@Nullable Bundle savedInstanceState)`
  Hier wird das Layout der Activity erstellt.

Parameter

- `Bundle savedInstanceState`
  Die zuvor gespeicherte Instanz der Activity, die wieder hergestellt zwird, sonst NULL

3.1.2.2 `public static interface View.OnClickListener`

Beschreibung
Schnittstelle dafür, wenn auf die View geklickt wird.klickt wird.

Methoden

- `public void onClick(View v)`
  Aufruf bei einen Klick auf ein Element.

Parameter

- `View v`
  Die angeklickte View

3.1.2.3 public static interface SwipeRefreshLayout.OnRefreshListener

Beschreibung
Schnittstelle dafür, wenn durch Hinunter-Swipen eine Aktualisierung ausgeführt werden soll

Methoden

- `public boolean onRefresh()`
  Aufruf beim Hinunter-Swipen zum Aktualisieren

3.1.2.4 public class LoginActivity extends AppCompatActivity implements View.OnClickListener

Beschreibung
Diese Klasse zeigt den Login mit dem Google-Account. Man kann seinen Google-Account-Daten eingeben und sich anmelden.

Methoden

- `public void onCreate(@Nullable Bundle savedInstanceState)`
  Hier wird das Layout der Activity erstellt.

Parameter

- `Bundle savedInstanceState`
  Die zuvor gespeicherte Instanz der Activity, die wieder hergestellt zwird, sonst NULL

3.1.2.5 public class StartActivity extends AppCompatActivity implements View.OnClickListener

Beschreibung
Diese Klasse zeigt den Startbildschirm mit dem App-Logo und zwei Buttons: Ein Button zum Erstellen einer WG und einer zum Eingeben eines Zugangscodes.

Methoden

- `public void onCreate(@Nullable Bundle savedInstanceState)`
  Hier wird das Layout der Activity erstellt.

Parameter

- `Bundle savedInstanceState`
  Die zuvor gespeicherte Instanz der Activity, die wieder hergestellt zwird, sonst NULL

3.1.2.6 public class CreateFlatshareActivity extends AppCompatActivity implements View.OnClickListener

Beschreibung
In dieser Klasse kann man der WG einen Namen geben und kann mithilfe eines Buttons zur HomeActivity gelangen.

Methoden

- `public void onCreate(@Nullable Bundle savedInstanceState)`
  Hier wird das Layout der Activity erstellt.

Parameter

- `Bundle savedInstanceState`
  Die zuvor gespeicherte Instanz der Activity, die wieder hergestellt zwird, sonst NULL

3.1.2.7 public class GetAccessCodeActivity extends AppCompatActivity implements View.OnClickListener

Beschreibung
In dieser Klasse kriegt man den Zugangscode und kann mithilfe eines Buttons zur HomeActivity gelangen.

Methoden

- `public void onCreate(@Nullable Bundle savedInstanceState)`
  Hier wird das Layout der Activity erstellt.

Parameter

- `Bundle savedInstanceState`
  Die zuvor gespeicherte Instanz der Activity, die wieder hergestellt zwird, sonst NULL

3.1.2.8 public class EnterAccessCodeActivity extends AppCompatActivity implements View.OnClickListener

Beschreibung
In dieser Klasse kann man den Zugangscode eingeben und kann mithilfe eines Buttons zur HomeActivity gelangen.

Methoden

- `public void onCreate(@Nullable Bundle savedInstanceState)`
  Hier wird das Layout der Activity erstellt.

Parameter

- `Bundle savedInstanceState`
  Die zuvor gespeicherte Instanz der Activity, die wieder hergestellt zwird, sonst NULL

3.1.2.9 public class HomeActivity extends AppCompatActivity implements View.OnClickListener implements SwipeRefreshLayout.OnRefreshListener

Beschreibung
Diese Klasse zeigt das View der WG-Pinnwand, man sieht die Notes mit Überschrift und Magnet und einige Buttons. Drei Frozen Notes sind von Anfang an enthalten. Frozen Notes haben immer einen schwarzen Magneten.

Methoden

- `public void onCreate(@Nullable Bundle savedInstanceState)`
  Hier wird das Layout der Activity erstellt.

Parameter

- `Bundle savedInstanceState`
  Die zuvor gespeicherte Instanz der Activity, die wieder hergestellt zwird, sonst NULL

3.1.2.10 public class FullTextCoolNoteActivity extends AppCompatActivity implements View.OnClickListener implements SwipeRefreshLayout.OnRefreshListener

Beschreibung
Diese Klasse zeigt eine Großansicht einer Text-Cool-Note mit zugehörigem Magneten, Erstelldatum, Tags, Titel, Inhalt, Lesebestätigungen und Kommentaren. Der @All-Tag ist immer da, wenn keine Tags spezifiziert werden. Es stehen wieder einige Buttons zur Interaktion zu Verfügung.

Methoden

- `public void onCreate(@Nullable Bundle savedInstanceState)`
  Hier wird das Layout der Activity erstellt.

Parameter

- `Bundle savedInstanceState`
  Die zuvor gespeicherte Instanz der Activity, die wieder hergestellt zwird, sonst NULL

3.1.2.11 public class FullFrozenNoteActivity extends AppCompatActivity implements View.OnClickListener implements SwipeRefreshLayout.OnRefreshListener

Beschreibung
Diese Klasse zeigt eine Großansicht einer Frozen Note mit zugehörigem schwarzen Magneten, Titel und Inhalt. Es stehen wieder einige Buttons zur Interaktion zu Verfügung.

Methoden

- `public void onCreate(@Nullable Bundle savedInstanceState)`
  Hier wird das Layout der Activity erstellt.

Parameter

- `Bundle savedInstanceState`
  Die zuvor gespeicherte Instanz der Activity, die wieder hergestellt zwird, sonst NULL

3.1.2.12 public class FullImageCoolNoteActivity extends AppCompatActivity implements View.OnClickListener implements SwipeRefreshLayout.OnRefreshListener

Beschreibung
Diese Klasse zeigt eine Großansicht einer Bild-Cool-Note mit zugehörigem Magneten, Erstelldatum, Tags, Titel, Inhalt, Lesebestätigungen und Kommentaren. Der @All-Tag ist immer da, wenn keine Tags spezifiziert werden. Es stehen wieder einige Buttons zur Interaktion zu Verfügung.

Methoden

- `public void onCreate(@Nullable Bundle savedInstanceState)`
  Hier wird das Layout der Activity erstellt.

Parameter

- `Bundle savedInstanceState`
  Die zuvor gespeicherte Instanz der Activity, die wieder hergestellt zwird, sonst NULL

3.1.2.13 public class CreateTextCoolNoteActivity extends AppCompatActivity implements View.OnClickListener

Beschreibung
Diese Klasse zeigt das View für die Erstellung einer Text-Cool-Note. Dieses View wird auch für die Kommentar-Funktion benutzt, nur, dass man keinen Titel schreiben und keine Wichtigkeit auswählen kann. Das View öffnet sich auch, wenn man eine Frozen Note editieren will, wobei die Wichtigkeit wieder nicht auswählbar ist.

Methoden

- `public void onCreate(@Nullable Bundle savedInstanceState)`
  Hier wird das Layout der Activity erstellt.

Parameter

- `Bundle savedInstanceState`
  Die zuvor gespeicherte Instanz der Activity, die wieder hergestellt zwird, sonst NULL

3.1.2.14 public class CreateImageCoolNoteActivity extends AppCompatActivity implements View.OnClickListener

Beschreibung
Diese Klasse zeigt das View für die Erstellung einer Bild-Cool-Note.

Methoden

- `public void onCreate(@Nullable Bundle savedInstanceState)`
  Hier wird das Layout der Activity erstellt.

Parameter

- `Bundle savedInstanceState`
  Die zuvor gespeicherte Instanz der Activity, die wieder hergestellt zwird, sonst NULL

3.1.2.15 public class CreateCommentActivity extends AppCompatActivity implements View.OnClickListener

Beschreibung
Diese Klasse zeigt das View für die Erstellung eines Kommentars.

Methoden

- `public void onCreate(@Nullable Bundle savedInstanceState)`
  Hier wird das Layout der Activity erstellt.

Parameter

- `Bundle savedInstanceState`
  Die zuvor gespeicherte Instanz der Activity, die wieder hergestellt zwird, sonst NULL

3.1.2.16 public class EditFrozenNoteActivity extends AppCompatActivity implements View.OnClickListener

Beschreibung
Diese Klasse zeigt das View für die Bearbeitung einer Frozen Note.

Methoden

- `public void onCreate(@Nullable Bundle savedInstanceState)`
  Hier wird das Layout der Activity erstellt.

Parameter

- `Bundle savedInstanceState`
  Die zuvor gespeicherte Instanz der Activity, die wieder hergestellt zwird, sonst NULL

3.1.2.17 public class MemberListActivity extends AppCompatActivity implements View.OnClickListener implements SwipeRefreshLayout.OnRefreshListener

Beschreibung
Diese Klasse zeigt das View zum Einsehen der aktuellen Mitglieder mit mit zugehörigem Magneten.

Methoden

- `public void onCreate(@Nullable Bundle savedInstanceState)`
  Hier wird das Layout der Activity erstellt.

Parameter

- `Bundle savedInstanceState`
  Die zuvor gespeicherte Instanz der Activity, die wieder hergestellt zwird, sonst NULL

3.1.2.18 public class LanguageActivity extends AppCompatActivity implements View.OnClickListener

Beschreibung
Diese Klasse zeigt das View zum Einstellen der App-Sprache.

Methoden

- `public void onCreate(@Nullable Bundle savedInstanceState)`
  Hier wird das Layout der Activity erstellt.

Parameter

- `Bundle savedInstanceState`
  Die zuvor gespeicherte Instanz der Activity, die wieder hergestellt zwird, sonst NULL

3.1.2.19 public class AccessCodeActivity extends AppCompatActivity implements View.OnClickListener

Beschreibung
Diese Klasse zeigt das View zum Erstellen eines neuen Zugangscodes.

Methoden

- `public void onCreate(@Nullable Bundle savedInstanceState)`
  Hier wird das Layout der Activity erstellt.

Parameter

- `Bundle savedInstanceState`
  Die zuvor gespeicherte Instanz der Activity, die wieder hergestellt zwird, sonst NULL

3.1.3 package kit.edu.pse.fridget.client.viewmodel

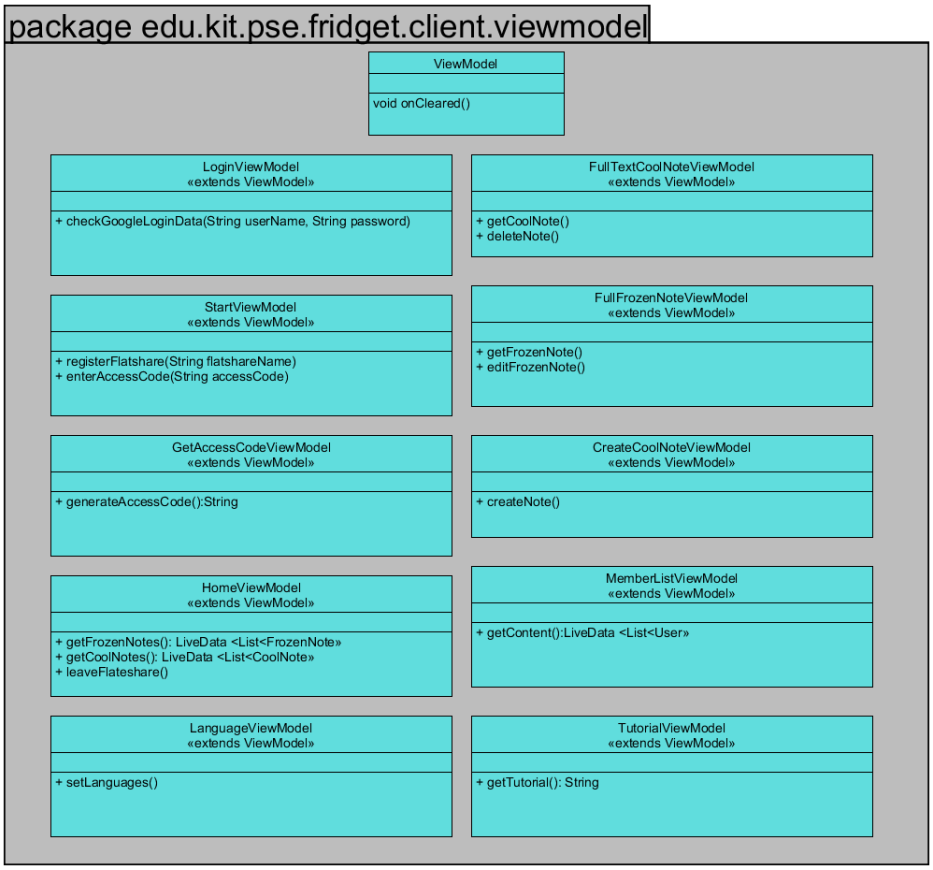

Abbildung 3.3: Klassen des ViewModels

3.1.3.1 `public class LoginViewModel extends ViewModel`

Beschreibung
LoginViewModel ist das ViewModel zur LoginActivity. In dieser Klasse wird geprüft, ob der Benutzer sich korrekt einloggt.

Methoden

- `public boolean checkGoogleLoginData(String userName, String password)`
  Diese Methode überprüft ob die eingegebenen Google-Daten richtig sind

  Parameter

  - `userName`
    Google-Account Adresse
  - `password`
    Passwort

  Rückgabewert

  - *gibt an, ob die eingebenen Werte richtig oder falsch sind*

3.1.3.2 `StartViewModel extends ViewModel`

Beschreibung
Diese Klasse ist das ViewModel zur StartActivity, CreateFlatshareActivity, EnterAccessCodeActivity. Sie ermöglicht das Einloggen in die WG und stellt alle Daten der WG zur Verfügung.

Methoden

- `public void registerFlatshare(String flatshareName)`

  Diese Methode lässt eine neue WG mit dem übergebenen Namen erstellen. Dabei erstellt sie ein Objekt des Models User und übergibt diesen dem FlatShareService. Wenn in der Datenbank eine neue Flatshare angelegt wurde, erfragt diese Methode die Flatshare-ID und speichert auf einem Local Repository die FlateShare-ID sowie die User-ID.

  Parameter

  - `flatshareName`
    Name der WG

- `public void enterAccessCode(String accessCode)`

  Diese Methode lässt den AccessCode überprüfen und stellt die passenden Daten zu der WG bereit. Sobald die FlateShare-ID bekannt ist, wird sie mit der User-ID auf einem Local Repository gespeichert.

  Parameter

  - `accessCode`
    Zugangscode

3.1.3.3 `public class GetAccessCodeViewModel extends ViewModel`

Beschreibung
Diese Klasse ist das ViewModel zur GetAccessCodeActivity. Sie dient zur Generierung des Zugangscodes.

Methoden

- `public String generateAccessCode()`

  Diese Methode lässt einen zufälligen, einzigartigen AccessCode generieren und gibt diesen zurück.

  Rückgabewert

  - *Der generierte Zugangscode wird zurückgegeben.*

3.1.3.4 `public class HomeViewModel extends ViewModel`

Beschreibung
HomeViewModel ist das ViewModel zur HomeActivity. Diese Klasse aktualisiert die Daten in der HomeActivity. Sie überprüft also, ob die Anordnung der Cool Notes verändert wurde, ob die Frozen Notes verändert wurden usw.

Methoden

- `public LiveData <List<FrozenNote>> getFrozenNotes()`

  Diese Methode übergibt die Daten aller Frozen Notes auf der Pinnwand.

  Rückgabewert

  – *Die Liste an Frozen Notes wird in Form von LiveData zurückgegeben.*

- `public LiveData <List<CoolNote>> getCoolNotes()`

  Diese Methode holt sich die Daten aller Cool Notes auf der Pinnwand sowie deren Anordnung.

  Rückgabewert

  – *Die Liste an Cool Notes wird in Form von LiveData zurückgegeben.*

- `public void leaveFlateshare()`

  Diese Methode sorgt dafür, dass alles, was derjenige, der die WG verlässt, erstellt hat, gelöscht wird. Außerdem wird derjenige aus der Mitgliederliste gelöscht.

3.1.3.5 `public class FullTextCoolNoteViewModel extends ViewModel`

Beschreibung
Diese Klasse ist das ViewModel zur FullTextCoolNoteActivity und FullImageCoolNoteActivity. Diese Klasse verwaltet alle Daten, die für die Großansicht der Cool Note benötigt wird.

Methoden

- `public void getCoolNote()`

  Diese Methode holt alle Daten der CoolNote und speichert sie in einzelne Attribute, die mit den get-Methoden geholt werden können.

- `public void deleteNote()`

  Diese Methode veranlasst das Löschen der Note in der Datenbank.

3.1.3.6 `public class FullFrozenNoteViewModel extends ViewModel`

Beschreibung
Diese Klasse ist das ViewModel zur FullTextFrozenNoteActivity. Dieses ViewModel holt alle benötigten Daten für die Großansicht der Frozen Note.

Methoden

- `public void getFrozenNote()`

  Diese Methode holt alle Daten der Frozen Note und speichert sie in einzelne Attribute, die mit den get-Methoden geholt werden können.

- `public void editFrozenNote(String title, String content)`

  Diese Methode speichert die neuen Daten.

  Parameter

  – `title`
  Überschrift

  – `content`
  Inhalt

3.1.3.7 `public class CreateCoolNoteViewModel extends ViewModel`

Beschreibung
CreateCoolNoteViewModel ist das ViewModel zur CreateTextCoolNoteActivity und CreateImageCoolNoteActivity. Es wird benötigt, um die neu erstellten Cool Notes zu speichern.

Methoden

- `public void createNote()`

  Es wird ein neues Objekt CoolNote erstellt und dem passenden Service übergeben, um die CoolNote in der Datenbank hinzuzufügen. Es wird eine zufällige Position berechnet.

3.1.3.8 `public class MemberListViewModel extends ViewModel`

Beschreibung
Diese Klasse ist das ViewModel zur MemberListActivity. Es holt alle Daten bezüglich der Mitglieder. d.h. Magnetfarben und Namen.

Methoden

- `public LiveData <List<User>> getContent()`

  Diese Methode gibt die Liste der Mitglieder zurück.

  Rückgabewert

  – *Die Liste der Mitglieder wird in Form von LiveData zurückgegeben.*

3.1.3.9 `public class LanguageViewModel extends ViewModel`

Beschreibung
Diese Klasse ist das ViewModel zur LanguageActivity.

Methoden

- `public void setLanguages()`

  Diese Methode ändert die Sprache der App.

3.1.3.10 `public class TutorialViewModel extends ViewModel`

Beschreibung
Diese Klasse ist das ViewModel zur TutorialActivity.

Methoden

- `public String getTutorial()`

  Diese Methode stellt die Daten der Textinhalte zur Verfügung.

3.1.4 package edu.kit.pse.fridget.client.datamodel

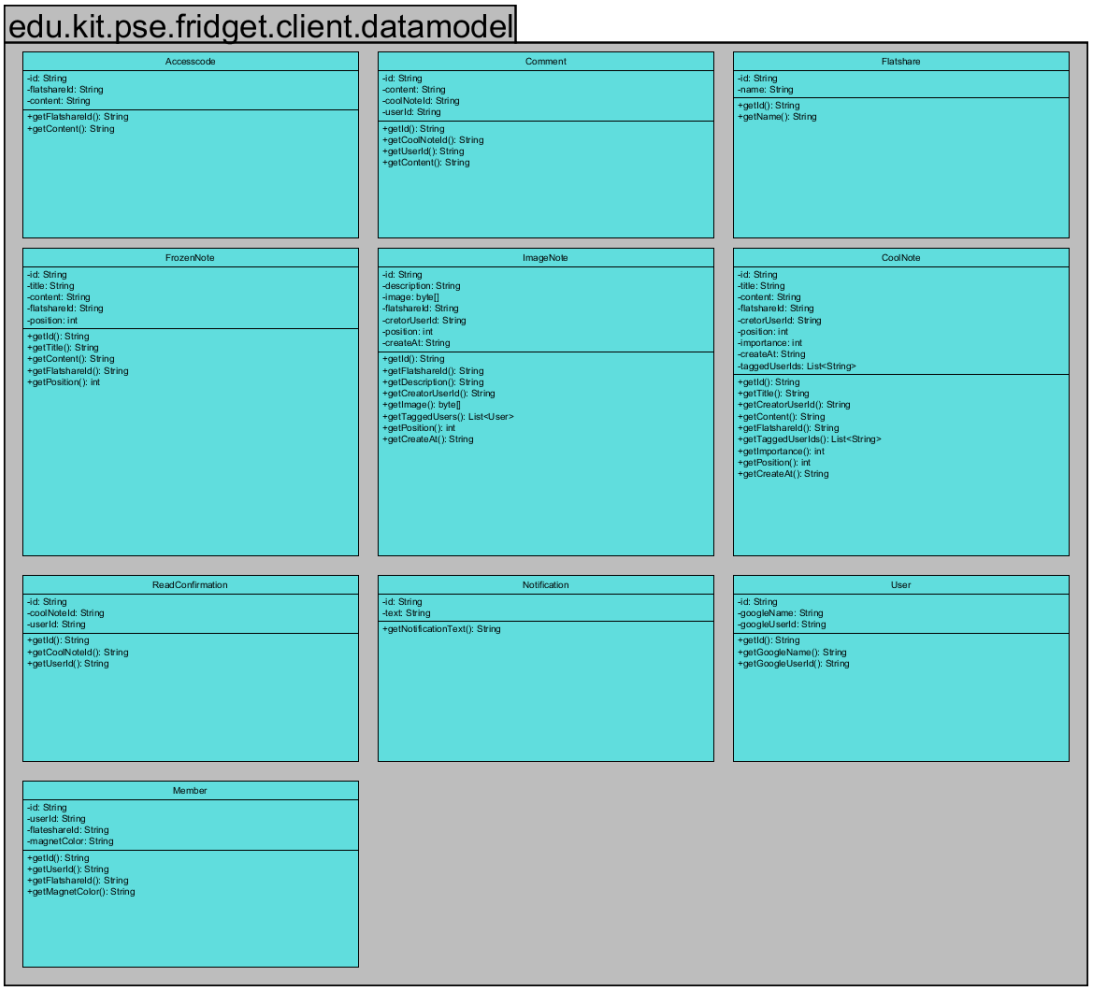

Abbildung 3.4: Klassen des Models

3.1.4.1 `public class Accesscode`

Beschreibung
Die Klasse Accesscode stellt den Accesscode oder Zugangscode einer WG dar.

Attribute

- **id**: Die ID des Accesscode.
  Typ: String
- **flatshareId:** Die ID der WG, zu der der Accesscode gehört.
  Typ: String
- **content:** Der Inhalt des Accesscodes.
  Typ: String

Methoden

- `public String getFlatshareId()`
  Gibt die ID der WG des Accesscodes zurück.

- `public String getContent()`
  Gibt den Accesscode als String zurück.

3.1.4.2 `public class User`

Beschreibung
Die Klasse User stellt einen Benutzer der App dar.

Attribute

- **id:** Die ID des Benutzers.
  Typ: String
- **googleName:** Der Google-Name des Benutzers.
  Typ: String
- **googleUserId:** Die Google-ID des Benutzers.
  Typ: String

Methoden

- `public String getId()`
  Gibt die ID des Benutzers der App zurück.
- `public String getGoogleName()`
  Gibt den Google-Namen des Benutzers zurück.
- `public String getGoogleUserId()`
  Gibt die Google-ID des Benutzers der App zurück.

3.1.4.3 `public class Member`

Beschreibung
Die Klasse Member stellt ein Mitglieder einer WG dar.

Attribute

- **id:** Die ID der Mitgliedschaft.
  Typ: String
- **userId:** Die ID des Benutzers, der Mitglied einer WG ist.
  Typ: String
- **flatshareId**: Die ID der WG, zu der das Mitglied gehört.
  Typ: String
- **magnetColor:** Die Magnetfarbe des Mitglieds.
  Typ: String

Methoden

- `public String getId()`
  Gibt ID der Mitgliedschaft der Benutzer zur WG zurück.

- `public String getUserId()`
  Gibt die ID des Benutzers der App, der Mitglied einer WG ist, zurück.

- `public String getFlatshareId()`
  Gibt ID der WG zurück.

- `public String getMagnetColor()`
  Gibt die Magnetfarbe des Benutzers zurück.

3.1.4.4 `public class Flatshare`

Beschreibung
Die Klasse Flatshare stellt eine WG dar, die in der App registriert ist.

Attribute

- **id:** Die ID der WG.
  Typ: String
- **name:** Der Name der WG.
  Typ: String

Methoden

- `public String getId()`
  Gibt die ID der WG zurück.

- `public String getName()`
  Gibt den WG-Namen zurück.

3.1.4.5 `public class CoolNote`

Beschreibung
Die Klasse CoolNote stellt die Notiz der Art "Cool Note" dar, die aus einer Überschrift und Text-Inhalt besteht. Cool Notes sind erstellbare, nicht-editierbare, löschbare Notizen.

Attribute

- **id:** Die ID der Cool Note.
  Typ: String
- **title:** Die Überschrift der Cool Note.
  Typ: String
- **content:** Der Inhalt der Cool Note.
  Typ: String
- **flatshareId:** Die ID der WG, zu der die Cool Note gehört.
  Typ: String
- **creatorUserId:** Die ID des Benutzers, der die Cool Note erstellt hat.
  Typ: String

- **position:** Die Position der Cool Note.
  Typ: int
- **importance:** Die Wichtigkeit der Cool Note.
  Typ: int
- **createAt:** Das Erstelldatum der Cool Note.
  Typ: String
- **taggedUserIds**: Die IDs der Mitglieder, die in der Cool Note getaggt wurden.
  Typ: List<String>

Methoden

- `public String getId()`
  Gibt die Cool-Note-ID zurück.

- `public String getTitle()`
  Gibt die Überschrift einer Cool Note zurück.

- `public String getCreatorUserId()`
  Gibt die ID des Mitgliedes zurück, der die Cool Note erstellt hat.

- `public String getContent()`
  Gibt den Inhalt der Cool Note zurück.

- `public String getFlatshareId()`
  Gibt die ID der WG zurück, zu der diese Cool Note gehört.

- `public List<String> getTaggedUserIds()`
  Gibt die IDs der Mitglieder zurück, die in der Cool Note getagged sind. Wenn keine Tags spezifiziert sind, sind alle Mitglieder der WG mit @All getagged.

- `public int getImportance()`
  Gibt die Wichtigkeit der Cool Note zurück.

- `public int getPosition()`
  Gibt die Position der Cool Note zurück.

- `public String getCreateAt()`
  Gibt das Erstelldatum der Cool Note zurück.

3.1.4.6 `public class FrozenNote`

Beschreibung
Die Klasse FrozenNote stellt die Notiz der Art "Frozen Note" dar, die aus einer Überschrift und Text-Inhalt besteht. Frozen Notes sind feste, editierbare, nicht-löschbare Notizen.

Attribute

- **id:** Die ID der Frozen Note.
  Typ: String
- **title:** Die Überschrift der Frozen Note.
  Typ: String
- **content:** Der Inhalt der Frozen Note.
  Typ: String
- **flatshareId:** Die ID der WG, zu der die Frozen Note gehört.
  Typ: String
- **position:** Position der Frozen Note, die schon fest ist beim Erstellen der Pinnwand.
  Typ: int

Methoden

- `public String getId()`
  Gibt die Frozen-Note-ID zurück.
- `public String getTitle()`
  Gibt die Überschrift einer Frozen Note zurück.
- `public String getContent()`
  Gibt den Inhalt der Frozen Note zurück.
- `public String getFlatshareId()`
  Gibt die ID der WG zurück, zu der diese Frozen Note gehört.
- `public int getPosition()`
  Gibt die Position der Frozen Note zurück.

3.1.4.7 `public class Notification`

Beschreibung
Die Klasse Notification stellt die Benachrichtigung an den Benutzer über eine neue Cool Note dar.

Attribute

- **notificationId:** Die ID der Benachrichtigung.
  Typ: String
- **text:** Der Inhalt der Benachrichtigung.
  Typ: String

Methoden

- `public String getNotificationText()`
  Gibt den Benachrichtigungstext zurück.

3.1.4.8 `public class Comment`

Beschreibung
Die Klasse Comment stellt einen Kommentar dar, der unter einer Cool Note geschrieben werden kann.

Attribute

- **id:** Die ID des Kommentars.
  Typ: String
- **content:** Der Inhalt des Kommentars.
  Typ: String
- **coolNoteId:** Die ID der Cool Note, zu der der Kommentar gehört.
  Typ: String
- **userId:** Die ID des Benutzers, der den Kommentar geschrieben hat.
  Typ: String

Methoden

- `public String getId()`
  Gibt die Kommentar-ID zurück

- `public String getUserID()`
  Gibt die MemberID des Mitgliedes zurück, das den Kommentar geschrieben hat.

- `public String getCoolNoteId()`
  Gibt die CoolNoteID zurück.

- `public String getContent()`
  Gibt den Inhalt des Kommentares zurück.

3.1.4.9 `public class ImageNote`

Beschreibung
Die Klasse ImageNote stellt die Notiz der Art "Cool Note" dar, die aus einer Beschreibung und einem Bild besteht. Image Notes sind wie Cool Notes erstellbar, nicht-editierbar und löschbar.

Attribute

- **imageNoteId:** Die ID der Image Note.
  Typ: String
- **description:** Die Beschreibung der Image Note.
  Typ: String
- **image:** Der Inhalt der Image Note.
  Typ: byte[]
- **flatshareId:** Die ID der WG, zu der die Image Note gehört.
  Typ: String

- **creatorUserId:** Die ID des Benutzers, der die Image Note erstellt hat.
  Typ: String
- **position:** Die Position der Image Note.
  Typ: int
- **createAt:** Das Erstelldatum der Image Note.
  Typ: String

Methoden

- `public String getID()`
  Gibt die Image-Note-ID zurück.
- `public String getFlatshareId()`
  Gibt die ID der WG zurück, zu der die Image Note gehört
- `public String getDescription()`
  Gibt die Beschreibung einer Image Note zurück.
- `public String getCreatorUserId()`
  Gibt die ID des Mitgliedes zurück, der die Image Note erstellt hat.
- `public byte[] getImage()`
  Gibt den Inhalt, also das Bild der Image Note zurück.
- `public List<User> getTaggedUsers()`
  Gibt die Mitglieder zurück, die in der Image Note getagged sind. Wenn keine Tags spezifiziert sind, sind alle Mitglieder der WG mit @All getagged.
- `public int getPosition()`
  Gibt die Position der Image Note zurück.
- `public String getCreateAt()`
  Gibt das Erstelldatum der Image Note zurück.

3.1.4.10 `public class ReadConfirmation`

Beschreibung
Die Klasse ReadConfirmation stellt die Check-Box dar, die dem Benutzer ermöglicht, für sich selbst sichtbar zu machen, ob er eine Cool Note gelesen hat.

Attribute

- **readConfirmationId:** Die ID der Check-Box.
  Typ: String
- **coolNoteId:** Die ID der Cool Note, die als gelesen oder ungelesen markiert wird.
  Typ: String

- **userId:** Die ID des Benutzers, der die Check-Box benutzt.
  Typ: String

Methoden

- `public String getId()`
  Gibt die Read-Confirmation- oder Lesebestätigungs-ID zurück.

- `public String getUserId()`
  Gibt die ID des Mitgliedes zurück, der die Check-Box benutzt.

- `public String getCoolNoteId()`
  Gibt die CoolNoteID der Cool Note zurück, die als gelesen oder ungelesen markiert wird.

3.1.5 package edu.kit.pse.fridget.client.service

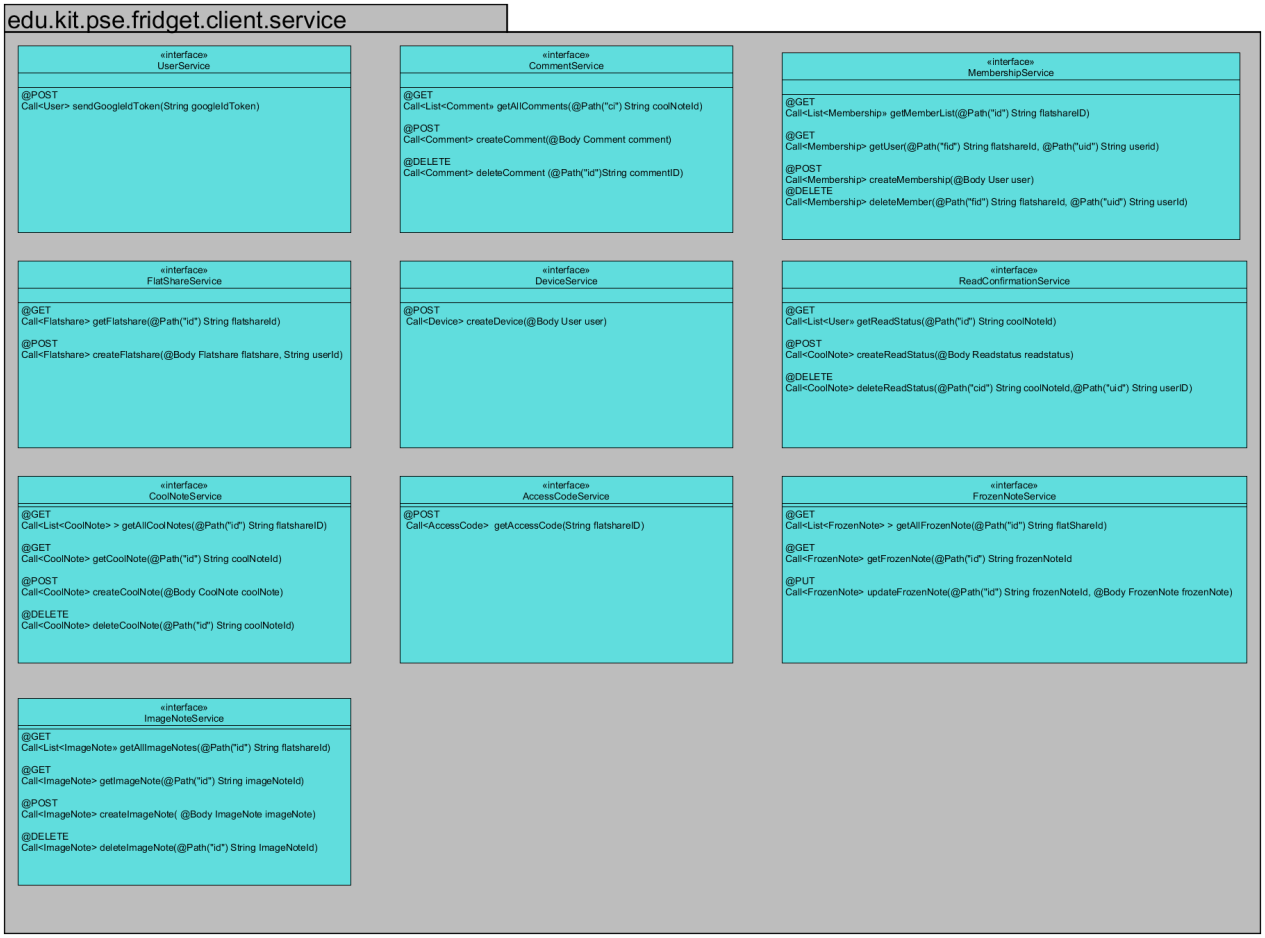

Abbildung 3.5: Klassen des Services

3.1.5.1 `public interface UserService`

Beschreibung
Dieses Interface dient dazu, den Google-Token an den Server weiterzuleiten.

Methoden

- `@POST`
  `Call<User> sendGoogleIdToken(String googleIdToken)`

  Diese Methode sendet den GoogleIdToken an den Server

  Parameter

  – `googleIdToken`
  zu sendender GoogleToken

  Rückgabewert

  – *GoogleToken*

3.1.5.2 public interface AccessCodeService

Beschreibung
Dieses Interface ist für die Synchronisation des Accesscodes mit dem Server zuständig

Methoden

- `Call<AccessCode> getAccessCode(String flatShareID)`

  Diese Methode fordert den Accesscode einer Flatshare an

  Parameter

  - `flatshareId`
    übergebene ID der Flatshare

  Rückgabewert

  - *Accesscode der Flatshare*

3.1.5.3 public interface CommentService

Beschreibung

Dieses Interface ist für die Synchronisation der Comments mit dem Server zuständig

Methoden

- `@GET("comments?cool-note={cid}")`
  `Call<List<Comment>> getAllComments(@Path("cid")String coolNoteId)`

  Diese Methode ruft alle Kommentare einer Cool Note ab

  Parameter

  - `coolNoteId`
    die ID der Cool Note

  Rückgabewert

  - *Alle Comments der zur CoolNoteId gehörenden Cool Note*

- `@POST("/comments")`
  `Call<Comment> createComment(@Body Comment comment)`

  Diese Methode schickt einen Comment an den Server

  Parameter

  - `comment`
    speichert einen Comment

  Rückgabewert

  - *Ein Comment*

- `@DELETE("/comments/id")`
  `Call<Comment> deleteComment (@Path("id")String commentID)`

  Diese Methode löscht einen Comment

  Parameter

  - `commentID`
    ID des zu löschenden Comments

3.1.5.4 public interface CoolNoteService

Beschreibung
Dieses Interface ist für die Synchronisation der Cool Notes mit dem Server zuständig

Methoden

- `@GET("/cool-notes?flatshare=id")`
  `Call<List<CoolNote>> getAllCoolNotes(@Path("id") String flatshareID)`

  Diese Methode ruft alle Cool Notes einer Flatshare ab

  Parameter

  - `flatshareId`
    Die Flatshare ID der abzurufenden Cool Notes

  Rückgabewert

  - *Alle Cool Notes*

- `@GET("/cool-notes/id")`
  `Call<CoolNote> getCoolNote(@Path("id") String coolNoteId)`

  Diese Methode ruft den Inhalt einer Cool Note ab **Parameter**

  - `coolNoteId`
    Die ID der Cool Note

  Rückgabewert

  - *Inhalt der zu der CoolNoteID gehörenden Cool Note*

- `@POST("/cool-notes")`
  `Call<CoolNote> createCoolNote(@Body CoolNote coolNote)`

  Diese Methode schickt eine neue Cool Note an den Server

  Parameter

  - `coolNote`
    Die Cool Note

  Rückgabewert

  - `Cool Note`

- `@DELETE("/cool-notes/id")`
  `Call<CoolNote> deleteCoolNote(@Path("id") String coolNoteId)`

  Diese Methode löscht eine Cool Note

  Parameter

  - `coolNoteId`
    Die Id der Cool Not

3.1.5.5 public interface FlatShareService

Beschreibung
Dieses Interface verwaltet die Synchronisation der Flatshare mit dem Server

Methoden

- `@POST("/flatshares")`
  `Call<Flatshare> createFlatshare(@Body Flatshare flatshare, String userId)`

  Diese Methode erstellt eine neue Flatshare auf dem Server **Parameter**

  - `flatshare`
    Name der zu erstellenden Flatshare
  - `userID`
    ID des Users, der die Flatshare erstellt

  Rückgabewert

  - *Flatshare*

- `@GET("/flatshares/id")`
  `Call<Flatshare> getFlatshare(@Path("id") String flatshareId)`

  Diese Methode ruft die Flatshare-Daten vom Server ab

  Parameter

  - `flatshareId`
    die ID der aufgerufenen Flatshare

  Rückgabewert

  - *Flatshare*

3.1.5.6 `public interface FrozenNoteService`

Beschreibung
Dieses Interface ist für die Synchronisation der Frozen Notes mit dem Server zuständig

Methoden

- `@GET("/frozen-notes?flatshare=id")`
  `Call<List<FrozenNote>> getAllFrozenNote(@Path("id") String flatShareId)`

  Diese Methode ruft die Frozen Notes vom Server ab

  Parameter

  - `flatshareId`
    die ID der aufgerufenen Flatshare

  Rückgabewert

  - *Frozen Note*

- `@GET("/frozen-notes/id")`
  `Call<FrozenNote> getFrozenNote(@Path("id") String frozenNoteId)`

  Diese Methode ruft den Inhalt einer Frozen Note ab

  Parameter

  - `frozenNoteId`
    die ID der aufgerufenen Frozen Note

  Rückgabewert

  - *Frozen Note*

- `@PUT("/frozen-notes/id")`
  `Call<FrozenNote> updateFrozenNote(@Path("id") String frozenNoteId, @Body FrozenNote frozenNote)`

  Diese Methode speichert Änderungen in einer Frozen Note

  Parameter

  - `frozenNoteId`
    die ID der aufgerufenen Frozen Note
  - `frozenNote`
    die geänderte Frozen Note

  Rückgabewert

  - *Frozen Note*

3.1.5.7 public interface ImageNoteService

Beschreibung
Dieses Interface dient zur Synchronisation der Image-Cool-Notes mit dem Server

Methoden

- `@GET("/image-notes?flatshare=id")`
  `Call<List<ImageNote>> getAllImageNotes(@Path("id") String flatshareId)`

  Diese Methode ruft die Image-Cool-Notes vom Server ab

  Parameter

  - `flatshareId`
    die ID der aufgerufenen Flatshare

  Rückgabewert

  - *ImageCoolNote*

- `@GET("/image-notes/id")`
  `Call<ImageNote> getImageNote(@Path(""id"") String imageNoteId)`

  Diese Methode ruft eine Image-Cool-Note ab

  Parameter

  - `imageNoteId`
    die ID der aufgerufenen Image-Cool-Note

  Rückgabewert

  - *Image-Cool-Note*

- `@POST("/image-notes")`
  `Call<ImageNote> createImageNote( @Body ImageNote imageNote)`

  Diese Methode schickt ein Image-Cool-Note an den Server

  Parameter

  - `imageNote`
    eine Image-Cool-Note

  Rückgabewert

  - *Image-Cool-Note*

- `@DELETE("/image-notes/id")`
  `Call<ImageNote> deleteImageNote(@Path(""id"") String ImageNoteId)`

  Diese Methode löscht eine Image-Cool-Note

  Parameter

  - `imageNoteId`
    Die ID der zu löschenden Image-Cool-Note

3.1.5.8 `public interface MembershipService`

Beschreibung
Dieses Interface verwaltet die Synchronisation der Members mit dem Server

Methoden

- `@GET("/memberships/users?flatshare=id")`
  `Call<List<Membership>> getMemberList(@Path(ïd") String flatshareID)`

  Diese Methode ruft die Mitglieder einer Flatshare ab

  Parameter

  - `flatshareId`
    die ID der aufgerufenen Flatshare

  Rückgabewert

  - *MemberList*

- `@GET("memberships?flatshare=fid&user =uid")`
  `Call<Membership> getUser(@Path(fid") String flatshareId, @Path("uid")`
  `String userid)`

  Diese Methode ruft die Daten eines Members ab

  Parameter

  - `flatshareId`
    die ID der aufgerufenen Flatshare
  - `userId`
    die ID des aufgerufenen Users

  Rückgabewert

  - *Daten eines Members*

- `@POST("/memberships")`
  `Call<Membership> createMembership(@Body User user)`

  Diese Methode fügt einen neuen Member in eine Flatshare ein

  Parameter

  - `user`
    Der User, der zu der Flatshare hinzugefügt wird

- `@DELETE("/memberships?flatshare=fid&user=uid")`
  `Call<Membership> deleteMember(@Path(fid") String flatshareId,`
  `@Path(üid") String userId)`

  Diese Methode löscht einen Member **Parameter**

  - `flatshareId`
    die ID der aufgerufenen Flatshare

- `userId`
  die ID des aufgerufenen Users

3.1.5.9 public interface ReadConfirmationService

Beschreibung
Dieses Interface synchronisiert den Gelesen-Status mit dem Server

Methoden

- `@GET("/read-confirmations/users?cool-note=id")`
  `Call<List<User>> getReadStatus(@Path("id") String coolNoteId)`

  Diese Methode ruft den Gelesen-Status vom Server ab

  Parameter

  - `coolNoteId`
    Die ID der betreffenden Cool Note

  Rückgabewert

  - *Read-Status*

- `@POST("/read-confirmations")`
  `Call<CoolNote> createReadStatus(@Body Readstatus readstatus)`

  Diese Methode setzt die Checkbox auf markiert

  Parameter

  - `readstatus`
    zeigt den Gelesen-Status einer Cool Note an

  Rückgabewert

  - *ReadStatus*

- `@DELETE("/read-confirmations?cool-note=cid&user=uid")`
  `Call<CoolNote> deleteReadStatus(@Path(cid") String coolNoteId, @Path("uid") String userID)`

  Diese Methode setzt die Checkbox auf unmarkiert

  Parameter

  - `coolNoteID`
    die ID der aufgerufenen Cool Note
  - `userID`
    die ID des aufgerufenen Users

3.1.5.10 public interface DeviceService

Beschreibung
Dieses Interface synchronisiert die Device-Daten mit dem Server

Methoden

- `@POST("/devices")`
  `Call<Device> createDevice(@Body User user)`

  Diese Methode fügt ein Device zu einer Flatshare hinzu

  Parameter

- `user`
  der zu dem Device gehörende User

Rückgabewert

- *Device Daten*

3.2 RESTful API

3.2.1 HTTP-Protokoll

Folgende REST Endpoints verwenden wir für die Kommunikation zwischen Client und Server:

HTTP Methode	**Endpoint**	**Beschreibung**
FlatshareController		
GET	/flatshares/{id}	WG beitreten
POST	/flatshares	WG erstellen
AccessCodeController		
POST	/access-codes	Zugangscode anfordern
User		
POST	/users	Anmelden
Device		
POST	/devices	App-Instanz-ID speichern
MembershipController		
GET	/memberships/users?flatshare={id}	Mitglieder ansehen
GET	/memberships?flatshare={fid}&user={uid}	Zugeteilte Magnetfarbe anfordern
POST	/memberships	WG beitreten (mit Zugangscode)
DELETE	/memberships?flatshare={fid}&user={uid}	WG verlassen
CoolNoteController		
GET	/cool-notes?flatshare={id}	Cool Notes aktualisieren
GET	/cool-notes/{id}	Großansicht einer Cool Note ansehen
POST	/cool-notes	Cool Note erstellen
DELETE	/cool-notes/{id}	Cool Note löschen
FrozenNoteController		
GET	/frozen-notes?flatshare={id}	Frozen Notes aktualisieren
GET	/frozen-notes/{id}	Großansicht einer Frozen Note ansehen
PUT	/frozen-notes/{id}	Frozen Note bearbeiten
ImageNoteController		
GET	/image-notes?flatshare={id}	Image Cool Notes aktualisieren
GET	/image-notes/{id}	Großansicht einer Image Cool Note ansehen
POST	/image-notes	Image Cool Note erstellen
DELETE	/image-notes/{id}	Image Cool Note löschen
ReadConfirmationController		
GET	/read-confirmations/users?cool-note={id}	Leser einer Cool Note ansehen
POST	/read-confirmations	“I have seen this”-Checkbox markieren
DELETE	/read-confirmations?cool-note={cid}&user={uid}	“I have seen this”-Checkbox unmarkieren
CommentController		
GET	/comments?cool-note={cid}	Kommentare einer Cool Note ansehen
POST	/comments	Kommentar schreiben
DELETE	/comments/{id}	Kommentar löschen

3.2.2 HTTP-Statuscodes

Folgende Statuscodes liefern wir intern bei der HTTP-Antwort auf jede HTTP-Anfrage:

Code	**Nachricht**	**Bedeutung**
Erfolgreiche Operation		
200	OK	Die Anfrage wurde erfolgreich bearbeitet und das Ergebnis der Anfrage wird in der Antwort übertragen.
201	Created	Die Anfrage wurde erfolgreich bearbeitet. Die angeforderte Ressource wurde vor dem Senden der Antwort erstellt.
204	No Content	Die Anfrage wurde erfolgreich durchgeführt, die Antwort enthält jedoch bewusst keine Daten.
Client-Fehler		
400	Bad Request	Die Anfrage-Nachricht war fehlerhaft aufgebaut.
401	Unauthorized	Die Anfrage kann nicht ohne gültige Authentifizierung durchgeführt werden.
403	Forbidden	Die Anfrage wurde mangels Berechtigung des Clients nicht durchgeführt, bspw. weil der authentifizierte Benutzer nicht berechtigt ist.
404	Not Found	Die angeforderte Ressource wurde nicht gefunden.
422	Unprocessable Entity	Die Verarbeitung der Anfrage wird z.B. wegen semantischer Fehler abgelehnt.
Server-Fehler		
500	Internal Server Error	Dies ist ein „Sammel-Statuscode“ für unerwartete Serverfehler.

3.3 Klassen des Servers

3.3.1 Klassendiagramm

Abbildung 3.6: Klassen des Servers

3.3.2 Package edu.kit.pse.fridget.server.controllers

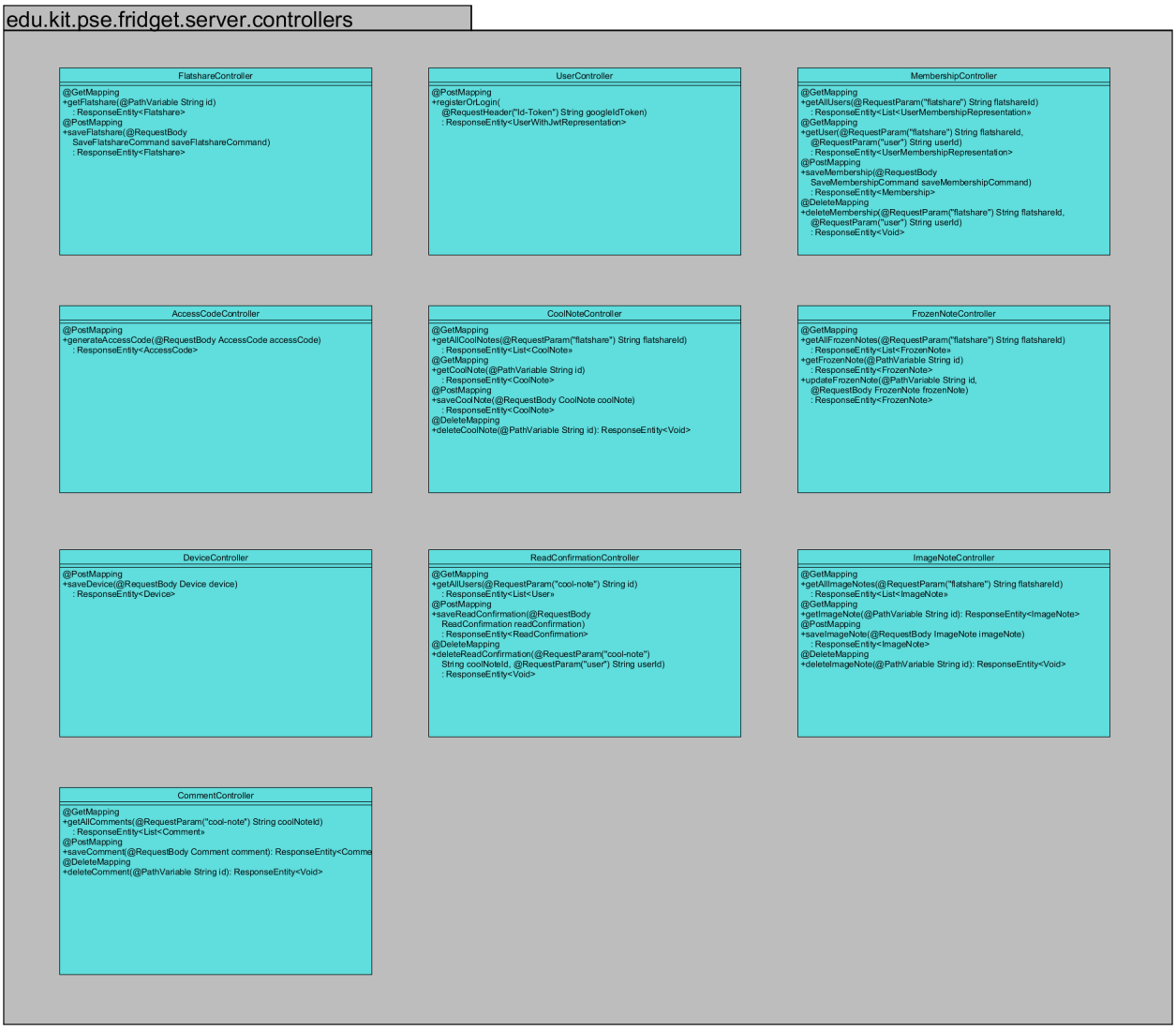

Abbildung 3.7: Klassen des Controllers

3.3.2.1 `Class AccessCodeController`

Beschreibung
Controller für Zugangscode

Konstruktor
`public AccessCodeController(AccessCodeService service)`

Methoden

- `public ResponseEntity<AccessCode> generateAccessCode(AccessCode accessCode)`

  Generiert einen Zugangscode für eine WG.

  Parameter

– `accessCode`
WG-ID mit leerem Zugangscode-Inhalt

Rückgabewert

– *Generierter Zugangscode als ResponseEntity*

3.3.2.2 `Class CommentController`

Beschreibung
Controller für Kommentar

Konstruktor
`public CommentController(CommentService service)`

Methoden

- `public ResponseEntity<List<Comment>> getAllComments(String coolNoteId)`
Findet alle Kommentare zu einer Cool Note.
Parameter
– `coolNoteId`
CoolNote-ID
Rückgabewert
– *Liste von gefundenen Kommentaren als ResponseEntity*

- `public ResponseEntity<Comment> saveComment(Comment comment)`
Speichert einen Kommentar.
Parameter
– `comment`
Kommentar zum Speichern
Rückgabewert
– *Gespeicherter Kommentar als ResponseEntity*

- `public ResponseEntity<Void> deleteComment(String id)`
Löscht einen Kommentar.
Parameter
– `id`
Kommentar-ID
Rückgabewert
– *Leere ResponseEntity*

3.3.2.3 `Class CoolNoteController`

Beschreibung
Controller für Cool Note

Konstruktor `public CoolNoteController(CoolNoteService coolNoteService, TaggedUserService taggedUserService)`

Methoden

- `public ResponseEntity<List<CoolNote>> getAllCoolNotes(String flatshareId)`

  Findet alle Cool Notes mit getaggten Benutzern in einer WG.

  Parameter

  - `flatshareId`
    WG-ID

  Rückgabewert

  - *Liste von gefundenen CoolNotes mit ID von getaggten Benutzern als ResponseEntity*

- `public ResponseEntity<CoolNote> getCoolNote(String id)`

  Findet eine Cool Note mit getaggten Benutzern.

  Parameter

  - `id`
    CoolNote-ID

  Rückgabewert

  - *Gefundene CoolNote mit ID von getaggten Benutzern als ResponseEntity*

- `public ResponseEntity<CoolNote> saveCoolNote(CoolNote coolNote)`

  Speichert eine Cool Note and die getaggten Benutzer.

  Parameter

  - `coolNote`
    CoolNote zum speichern

  Rückgabewert

  - *Gespeicherte CoolNote als ResponseEntity*

- `public ResponseEntity<Void> deleteCoolNote(String id)`

  Löscht eine Cool Note.

  Parameter

  - `id`
    CoolNote-ID

  Rückgabewert

  - *Leere ResponseEntity*

3.3.2.4 `Class DeviceController`

Beschreibung
Controller für Gerät

Konstruktor
`public DeviceController(DeviceService service)`

Methoden

- `public ResponseEntity<Device> saveDevice(Device device)`

  Speichert ein neues Gerät.

  Parameter

  – `device`
  Gerät zum Speichern

  Rückgabewert

  – *Gespeichertes Gerät als ResponseEntity*

3.3.2.5 `Class FlatshareController`

Beschreibung
Controller für WG

Konstruktor
`public FlatshareController(FlatshareService service)`

Methoden

- `public ResponseEntity<Flatshare> getFlatshare(String id)`

  Findet eine WG.

  Parameter

  – `id`
  WG-ID

  Rückgabewert

  – *Gefundene WG als ResponseEntity*

- `public ResponseEntity<Flatshare> saveFlatshare(SaveFlatshareCommand saveFlatshareCommand)`

  Speichert eine WG.

  Parameter

  – `saveFlatshareCommand`
  WG zum Speichern

  Rückgabewert

  – *Gespeicherte WG als ResponseEntity*

3.3.2.6 `Class FrozenNoteController`

Beschreibung
Controller für Frozen Note

Konstruktor
`public FrozenNoteController(FrozenNoteService service)`

Methoden

- `public ResponseEntity<List<FrozenNote>> getAllFrozenNotes(String flatshareId)`

  Findet alle Frozen Notes in einer WG.

  Parameter

  - `flatshareId`
    WG-ID

  Rückgabewert

  - *Liste von gefundenen FrozenNote als ResponseEntity*

- `public ResponseEntity<FrozenNote> getFrozenNote(String id)`

  Findet eine Frozen Note.

  Parameter

  - `id`
    FrozenNote-ID

  Rückgabewert

  - *Gefundene FrozenNote als ResponseEntity*

- `public ResponseEntity<FrozenNote> updateFrozenNote(String id, FrozenNote frozenNote)`

  Updatet eine Frozen Note.

  Parameter

  - `id`
    FrozenNote-ID
  - `frozenNote`
    FrozenNote zum Updaten

  Rückgabewert

  - *Geupdatete FrozenNote als ResponseEntity*

3.3.2.7 `Class ImageNoteController`

Beschreibung
Controller für Image-Cool-Note

Konstruktor

`public ImageNoteController(ImageNoteService service)`

Methoden

- `public ResponseEntity<List<ImageNote>> getAllImageNotes(String flatshareId)`

  Findet alle Image Cool Notes in einer WG.

  Parameter

  - `flatshareId`
    WG-ID

  Rückgabewert

- *Liste von gefundenen ImageNote als ResponseEntity*

- `public ResponseEntity<ImageNote> getImageNote(String id)`

  Findet eine Imag-Cool-Note.

  Parameter

  - `id`
    ImageNote-ID

  Rückgabewert

  - *Gefundene ImageNote als ResponseEntity*

- `public ResponseEntity<ImageNote> saveImageNote(ImageNote imageNote)`

  Speichert eine Image-Cool-Note.

  Parameter

  - `imageNote`
    ImageNote zum Speichern

  Rückgabewert

  - *Gespeicherte ImageNote als ResponseEntity*

- `public ResponseEntity<Void> deleteImageNote(String id)`

  Löscht eine Image-Cool-Note.

  Parameter

  - `id`
    ImageNote-ID

  Rückgabewert

  - *Leere ResponseEntity*

3.3.2.8 `Class MembershipController`

Beschreibung
Controller für Mitgliedschaft

Konstruktor
`public MembershipController(MembershipService service)`

Methoden

- `public ResponseEntity<List<UserMembershipRepresentation>> getAllUsers(Str`
  `flatshareId)`

  Findet alle Mitglieder in einer WG.

  Parameter

  - `flatshareId`
    WG-ID

  Rückgabewert

  - *Liste von gefundenen Benutzern mit Magenetfarben als ResponseEntity*

- `public ResponseEntity<UserMembershipRepresentation> getUser(String flatshareId, String userId)`

  Findet ein Mitglied in einer WG.

  Parameter

  - `flatshareId`
    WG-ID
  - `userId`
    Benutzer-ID

  Rückgabewert

  - *Gefundener Benutzer mit Magnetfarbe als ResponseEntity*

- `public ResponseEntity<Membership> saveMembership(SaveMembershipCommand saveMembershipCommand)`

  Speichert einen Benutzer in einer WG.

  Parameter

  - `saveMembershipCommand`
    Benutzer-ID und Zugangscode

  Rückgabewert

  - *Gespeicherte Mitgliedschaft als ResponseEntity*

- `public ResponseEntity<Void> deleteMembership(String flatshareId, String userId)`

  Löscht ein Mitglied von einer WG.

  Parameter

  - `flatshareId`
    WG-ID
  - `userId`
    Benutzer-ID

  Rückgabewert

  - *Leere ResponseEntity*

3.3.2.9 `Class ReadConfirmationController`

Beschreibung
Controller für Lesebestätigung

Konstruktor
`public ReadConfirmationController(ReadConfirmationService service)`

Methoden

- `public ResponseEntity<List<User>> getAllUsers(String id)`

  Findet alle Leser einer Cool Note.

  Parameter

  - `id`
    CoolNote-ID

Rückgabewert

– *Liste von gefundenen Benutzern als ResponseEntity*

- `public ResponseEntity<ReadConfirmation> saveReadConfirmation(ReadConfirmat readConfirmation)`

  Speichert einen Benutzer als Leser einer Cool Note.

  Parameter

  – `readConfirmation`
  Lesebestätigung zum Speichern

  Rückgabewert

  – *Gespeicherte Lesebestätigung als ResponseEntity*

- `public ResponseEntity<Void> deleteReadConfirmation(String coolNoteId, String userId)`

  Löscht einen Benutzer als Leser einer Cool Note.

  Parameter

  – `coolNoteId`
  CoolNote-ID

  – `userId`
  Benutzer-ID

  Rückgabewert

  – *Leere ResponseEntity*

3.3.2.10 `Class UserController`

Beschreibung
Controller für Benutzer

Konstruktor
`public UserController(UserService service)`

Methoden

- `public ResponseEntity<UserWithJwtRepresentation> registerOrLogin(String googleIdToken)`

  Authentifiziert einen Benutzer durch Google-ID-Token.

  Parameter

  – `googleIdToken`
  Google-ID-Token

  Rückgabewert

  – *Gespeicherter oder angemeldeter Benutzer mit JWT als ResponseEntity*

3.3.3 Package edu.kit.pse.fridget.server.models

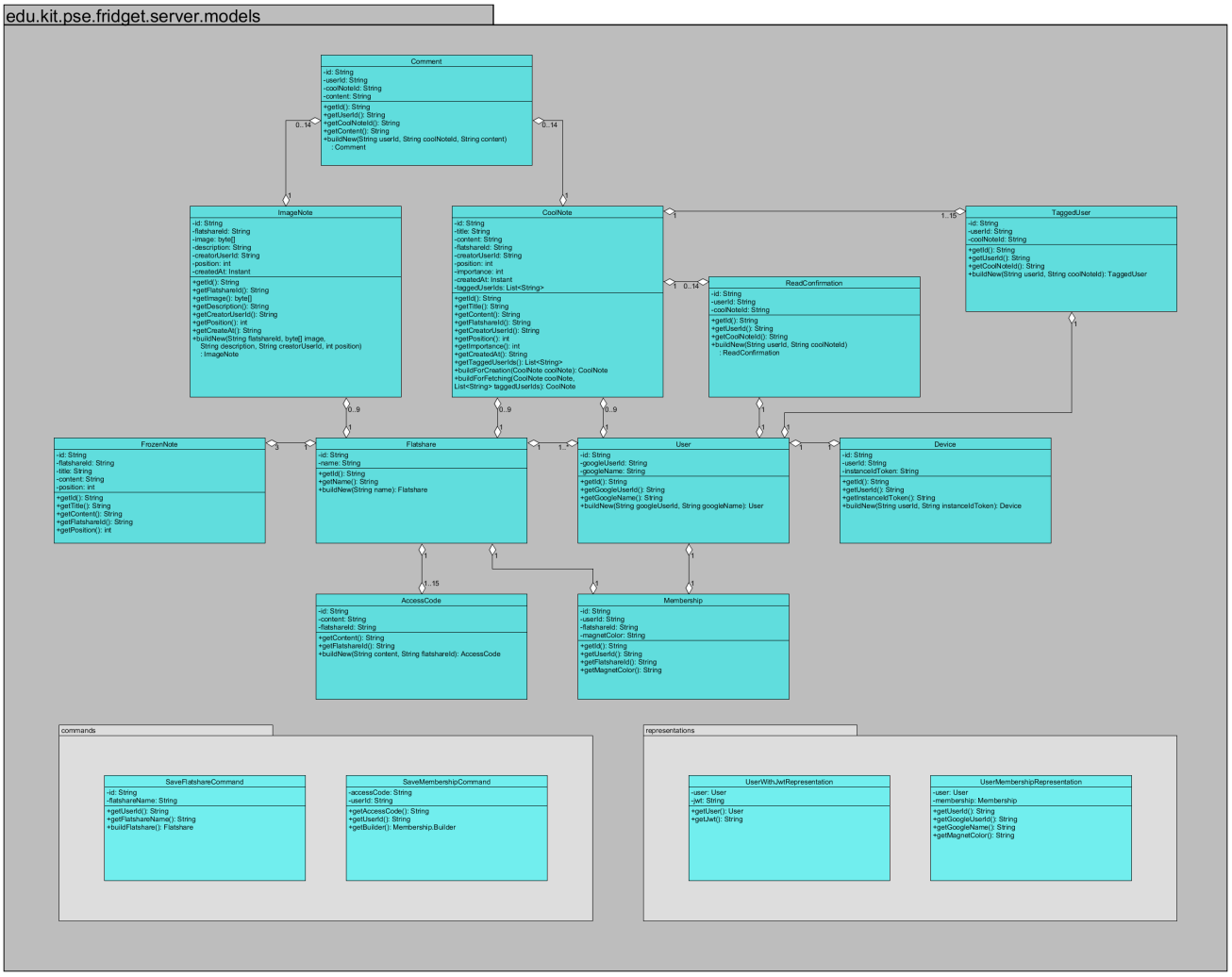

Abbildung 3.8: Klassen des Models(Server)

3.3.3.1 `Class AccessCode`

Beschreibung
Model für Zugangscode

Methoden

- `public String getId()`
  Getter für Zugangscode-ID
  Rückgabewert
  - *Zugangscode-ID*
- `public String getContent()`
  Getter für Inhalt vom Zugangscode
  Rückgabewert

– *Zugangscode-Inhalt*

- `public String getFlatshareId()`

  Getter für WG-ID

  Rückgabewert

  – *WG-ID*

- `public AccessCode buildNew(String content, String flatshareId)`

  Baut Zugangscode mit zufälliger UUID.

  Parameter

  – `content`
  Inhalt vom Zugangscode

  – `flatshareId`
  ID von der WG, zu der mit dem Zugangscode beigetreten werden kann

  Rückgabewert

  – *Gebauter Zugangscode mit zufälliger UUID*

3.3.3.2 `Class Comment`

Beschreibung
Model für Kommentar

Methoden

- `public String getId()`

  Getter für Kommentar-ID

  Rückgabewert

  – *Kommentar-ID*

- `public String getUserId()`

  Getter für ID von dem Benutzer, der den Kommentar geschrieben hat

  Rückgabewert

  – *Benutzer-ID*

- `public String getCoolNoteId()`

  Getter für ID von der Cool Note, zu der der Kommentar gehört

  Rückgabewert

  – *CoolNote-ID*

- `public String getContent()`

  Getter für Inhalt vom Kommentar

  Rückgabewert

  – *Kommentar-Inhalt*

- `public Comment buildNew(String userId, String coolNoteId, String content)`

  Baut Kommentar mit zufälliger UUID.

  Parameter

- `userId`
  ID von dem Benutzer, der den Kommentar geschrieben hat
- `coolNoteId`
  ID von der Cool Note, zu der der Kommentar gehört
- `content`
  Inhalt vom Kommentar

Rückgabewert

- *Gebauter Kommentar mit zufälliger UUID*

3.3.3.3 `Class CoolNote`

Beschreibung
Model für Cool Note

Methoden

- `public String getId()`
  Getter für Cool-Note-ID
  Rückgabewert
  - *Cool-Note-ID*
- `public String getTitle()`
  Getter für Überschrift von der Cool Note
  Rückgabewert
  - *Cool-Note-Überschrift*
- `public String getContent()`
  Getter für Inhalt von der Cool Note
  Rückgabewert
  - *Cool-Note-Inhalt*
- `public String getFlatshareId()`
  Getter für ID von der WG, zu der diese Cool Note gehört
  Rückgabewert
  - *WG-ID*
- `public String getCreatorUserId()`
  Getter für ID vom Benutzer, der die Cool Note erstellt hat
  Rückgabewert
  - *Benutzer-ID*
- `public int getPosition()`
  Getter für Position der Cool Note auf der Pinnwand
  Rückgabewert
  - *Position der Cool Note*

- `public int getImportance()`

  Getter für Wichtigkeit der Cool Note

  Rückgabewert

  – *Wichtigkeit der Cool Note*

- `public Instant getCreatedAt()`

  Getter für Erstelldatum der Cool Note

  Rückgabewert

  – *Erstelldatum der Cool Note*

- `public List<String> getTaggedUserIds()`

  Getter für IDs von den in dieser Cool Note getaggten Benutzern

  Rückgabewert

  – *Liste von IDs von den getaggten Benutzern*

3.3.3.4 `Class Device`

Beschreibung
Model für Gerät

Methoden

- `public String getId()`

  Getter für ID vom Gerät

  Rückgabewert

  – *Gerät-ID*

- `public String getUserId()`

  Getter für ID vom Benutzer, der die App auf dem Gerät benutzt

  Rückgabewert

  – *Benutzer-ID*

- `public String getInstanceIdToken()`

  Getter für ID von der App-Instanz auf dem Gerät

  Rückgabewert

  – *Instanz-ID*

- `public Device buildNew(String userId, String instanceIdToken)`

  Baut Gerät mit zufälliger UUID.

  Parameter

  – `userId`
  Benutzer-ID

  – `instanceIdToken`
  Instanz-ID-Token

  Rückgabewert

  – *Gebautes Gerät mit zufälliger UUID*

3.3.3.5 `Class Flatshare`

Beschreibung
Model für WG

Methoden

- `public String getId()`
  Getter für WG-ID
  Rückgabewert
  - *WG-ID*
- `public String getName()`
  Getter für WG-Name
  Rückgabewert
  - *WG-Name*
- `public Flatshare buildNew(String name)`
  Baut WG mit zufälliger UUID.
  Parameter
  - `name`
    Name von der WG
  Rückgabewert
  - *Gebaute WG mit zufälliger UUID*

3.3.3.6 `Class FrozenNote`

Beschreibung
Model für Frozen Note

Methoden

- `public String getId()`
  Getter für Frozen-Note-ID
  Rückgabewert
  - *Frozen-Note-ID*
- `public String getTitle()`
  Getter für Überschrift von der Frozen Note
  Rückgabewert
  - *Frozen-Note-Überschrift*
- `public String getContent()`
  Getter für Inhalt von der Frozen Note
  Rückgabewert
  - *Frozen-Note-Inhalt*

- `public String getFlatshareId()`

  Getter für ID von der WG, zu der diese Frozen Note gehört

  Rückgabewert

  – *WG-ID*

- `public int getPosition()`

  Getter für Position der Frozen Note auf der Pinnwand

  Rückgabewert

  – *Position der Frozen Note*

3.3.3.7 `Class ImageNote`

Beschreibung
Model für Image-Cool-Note

Methoden

- `public String getId()`

  Getter für ImageNote-ID

  Rückgabewert

  – *Image-Note-ID*

- `public String getFlatshareId()`

  Getter für ID von der WG, zu der diese Image-Cool-Note gehört

  Rückgabewert

  – *WG-ID*

- `public byte[] getImage()`

  Getter für Bild in der Image-Cool-Note

  Rückgabewert

  – *Bild*

- `public String getDescription()`

  Getter für Beschreibung der Image-Cool-Note

  Rückgabewert

  – *Beschreibung der Image-Cool-Note*

- `public String getCreatorUserId()`

  Getter für ID vom Benutzer, der die Image-Cool-Note erstellt hat

  Rückgabewert

  – *Benutzer-ID*

- `public int getPosition()`

  Getter für Position der Image Cool Note auf der Pinnwand

  Rückgabewert

  – *Position der Image-Cool-Note*

- `public String getCreateAt()`

  Getter für Erstelldatum der Image-Cool-Note

  Rückgabewert

  – *Erstelldatum der Image-Cool-Note*

- `public ImageNote buildNew(String flatshareId, byte[] image, String description, String creatorUserId, int position)`

  Baut Image Cool Note mit zufälliger UUID und aktuellem Datum.

  Parameter

  – `flatshareId`
  ID von der WG, zu der diese Image-Cool-Note gehört

  – `image`
  Bild in der Image-Cool-Note

  – `description`
  Beschreibung der Image-Cool-Note

  – `creatorUserId`
  ID vom Benutzer, der die Image-Cool-Note erstellt hat

  – `position`
  Position der Image-Cool-Note auf der Pinnwand

  Rückgabewert

  – *Gebaute Image-Cool-Note mit zufälliger UUID und aktuellem Datum*

3.3.3.8 `Class Membership`

Beschreibung
Model für Mitgliedschaft

Methoden

- `public String getId()`

  Getter für ID von der Mitgliedschaft

  Rückgabewert

  – *Mitgliedschafts-ID*

- `public String getUserId()`

  Getter für ID vom Mitglied

  Rückgabewert

  – *Benutzer-ID*

- `public String getFlatshareId()`

  Getter für WG-ID

  Rückgabewert

  – *WG-ID*

- `public String getMagnetColor()`

  Getter für Magnetfarbe, die dem Mitglied in der WG zugeteilt wird

  Rückgabewert

  – *Magenetfarbe*

3.3.3.9 `Class ReadConfirmation`

Beschreibung
Model für Lesebestätigung

Methoden

- `public String getId()`
  Getter für Lesebestätigung-ID
  Rückgabewert
  - *Lesebestätigungs-ID*
- `public String getUserId()`
  Getter für ID vom Benutzer, der die Cool Note gelesen hat
  Rückgabewert
  - *Benutzer-ID*
- `public String getCoolNoteId()`
  Getter für ID vom Cool Note, die gelesen wurde
  Rückgabewert
  - *Cool-Note-ID*
- `public ReadConfirmation buildNew(String userId, String coolNoteId)`
  Baut Lesebestätigung.
  Parameter
  - `userId`
    ID vom Benutzer, der die Cool Note gelesen hat
  - `coolNoteId`
    ID von der Cool Note, die gelesen wurde
  Rückgabewert
  - *Gebaute Lesebestätigung*

3.3.3.10 `Class TaggedUser`

Beschreibung
Model für getaggte Mitglieder

Methoden

- `public String getId()`
  Getter für ID vom getaggten Mitglied
  Rückgabewert
  - *ID vom getaggten Mitglied*
- `public String getUserId()`
  Getter für ID vom Benutzer, der getaggt wurde
  Rückgabewert
  - *Benutzer-ID*

- `public String getCoolNoteId()`

  Getter für ID von der Cool Note, in der der Benutzer getaggt wurde

  Rückgabewert

  – *Cool-Note-ID*

- `public TaggedUser buildNew(String userId, String coolNoteId)`

  Baut getaggtes Mitglied.

  Parameter

  – `userId`
  ID vom Benutzer, der getaggt wurde

  – `coolNoteId`
  ID von der Cool Note, in der der Benutzer getaggt wurde

  Rückgabewert

  – *Gebautes getaggtes Mitglied*

3.3.3.11 `Class User`

Beschreibung
Model für Benutzer

Methoden

- `public String getId()`

  Getter für Benutzer-ID

  Rückgabewert

  – *Benutzer-ID*

- `public String getGoogleUserId()`

  Getter für Google-ID vom Benutzer

  Rückgabewert

  – *Google-ID*

- `public String getGoogleName()`

  Getter für Google-Name vom Benutzer

  Rückgabewert

  – *Google-Name*

- `public User buildNew(String googleUserId, String googleName)`

  Baut Benutzer mit zufälliger UUID.

  Parameter

  – `googleUserId`
  Google-ID vom Benutzer

  – `googleName`
  Google-Name vom Benutzer

  Rückgabewert

  – *Gebauter Benutzer mit zufälliger UUID*

3.3.4 Package edu.kit.pse.fridget.server.models.commands

3.3.4.1 `Class SaveFlatshareCommand`

Beschreibung
Model für das Speichern der WG

Konstruktor
`public SaveFlatshareCommand(String userId, String flatshareName)`

Methoden

- `public String getUserId()`
  Getter für ID vom Benutzer, der die WG erstellt hat
  Rückgabewert
  - *Benutzer-ID*
- `public String getFlatshareName()`
  Getter für Name von der WG
  Rückgabewert
  - *WG-Name*
- `public Flatshare buildFlatshare()`
  Baut eine WG-Instanz.
  Rückgabewert
  - *WG*

3.3.4.2 `Class SaveMembershipCommand`

Beschreibung
Model für das Speichern der Mitgliedschaft

Konstruktor
`public SaveMembershipCommand(String accessCode, String userId)`

Methoden

- `public String getAccessCode()`
  Getter für Zugangscode
  Rückgabewert
  - *Zugangscode*
- `public String getUserId()`
  ID vom Benutzer, der der WG beitritt
  Rückgabewert
  - *Benutzer-ID*
- `public Membership.Builder getBuilder()`
  Getter für Builder von Mitgliedschaft
  Rückgabewert
  - *Builder von Mitgliedschaft*

3.3.5 Package edu.kit.pse.fridget.server.models.representations

3.3.5.1 `Class UserMembershipRepresentation`

Beschreibung
Model für Benutzer mit Mitgliedschaft-Info

Konstruktor
`public UserMembershipRepresentation(User user, Membership membership)`

Methoden

- `public String getUserId()`
  Getter für ID vom Benutzer
  Rückgabewert
  – *Benutzer-ID*
- `public String getGoogleUserId()`
  Getter für Google-ID vom Benutzer
  Rückgabewert
  – *Google-User-ID*
- `public String getGoogleName()`
  Getter für Google-Name von Benutzer
  Rückgabewert
  – *Google-Name*
- `public String getMagnetColor()`
  Getter für Magnetfarbe, die dem Benutzer zugeteilt wurde
  Rückgabewert
  – *Benutzer*

3.3.5.2 `Class UserWithJwtRepresentation`

Beschreibung
Model für Benutzer mit JWT (JSON Web Token)

Konstruktor
`public UserWithJwtRepresentation(User user, String jwt)`

Methoden

- `public User getUser()`
  Getter für Benutzer
  Rückgabewert
  – *Benutzer*
- `public String getJwt()`
  Getter für JWT
  Rückgabewert
  – *JWT*

3.3.6 Package edu.kit.pse.fridget.server.repositories

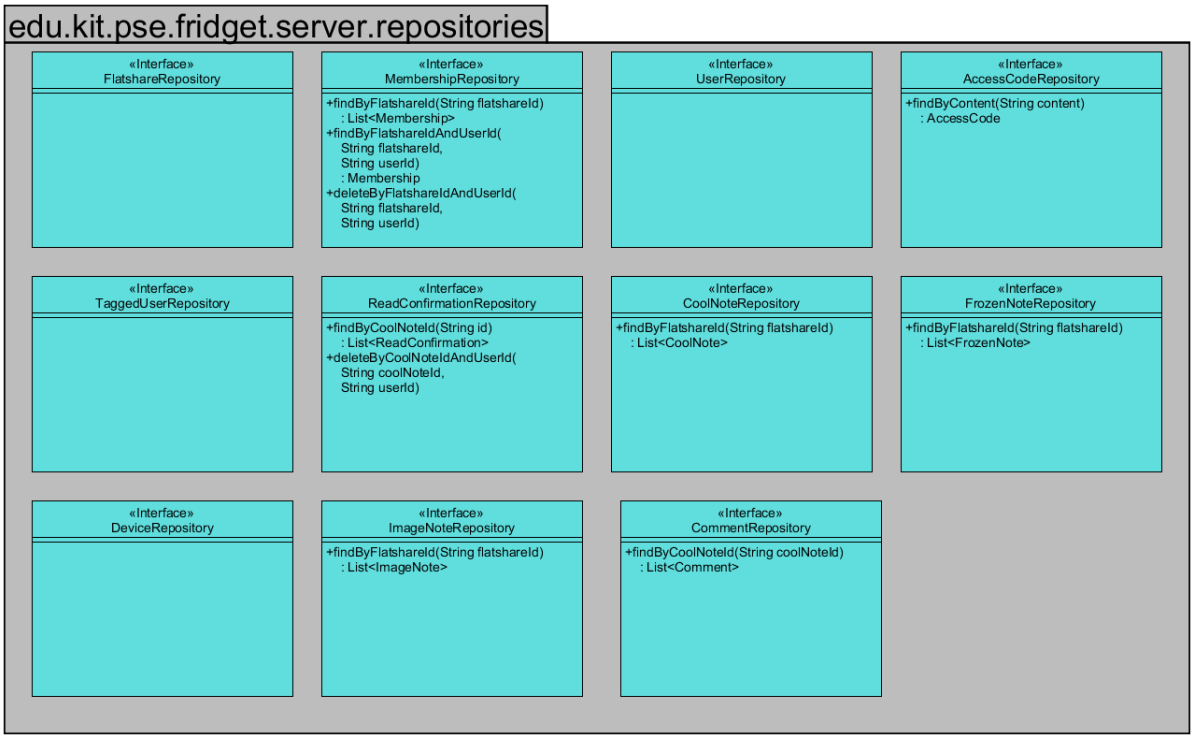

Abbildung 3.9: Klassen des Repositories

3.3.6.1 `Interface AccessCodeRepository extends JpaRepository<AccessCode,String>`

Beschreibung
Repository für Zugangscode

Methoden

- `public AccessCode findByContent(String content)`

  Findet einen Zugangscode über Inhalt.

  Parameter

  – `content`
  Inhalt des Zugangscodes

  Rückgabewert

  – *Gefundener Zugangscode*

3.3.6.2 `Interface CommentRepository extends JpaRepository<Comment,String>`

Beschreibung
Repository für Kommentar

Methoden

- `public List<Comment> findByCoolNoteId(String coolNoteId)`

  Findet alle Kommentare über Cool-Note-ID

  Parameter

  – `coolNoteId`
  Cool-Note-ID

  Rückgabewert

  – *Liste von gefundenen Kommentaren*

3.3.6.3 Interface CoolNoteRepository extends JpaRepository<CoolNote,String>

Beschreibung
Repository für Cool Note

Methoden

- `public List<CoolNote> findByFlatshareId(String flatshareId)`

  Findet alle Cool Notes über WG-ID

  Parameter

  – `flatshareId`
  WG-ID

  Rückgabewert

  – *Liste von gefundenen Cool Notes*

3.3.6.4 Interface DeviceRepository extends JpaRepository<Device,String>

Beschreibung
Repository für Geräte

3.3.6.5 Interface FlatshareRepository extends JpaRepository<Flatshare,String>

Beschreibung
Repository für WG

3.3.6.6 Interface FrozenNoteRepository extends JpaRepository<FrozenNote,String>

Beschreibung
Repository für Frozen Note

Methoden

- `public List<FrozenNote> findByFlatshareId(String flatshareId)`

  Findet alle Frozen Notes über WG-ID

  Parameter

- `flatshareId`
  WG-ID

Rückgabewert

- *Liste von gefundenen Frozen Notes*

3.3.6.7 `Interface ImageNoteRepository extends JpaRepository<ImageNote,String>`

Beschreibung
Repository für Image Cool Note

Methoden

- `public List<ImageNote> findByFlatshareId(String flatshareId)`

  Findet alle Image Cool Notes über WG-ID

  Parameter

  - `flatshareId`
    WG-ID

  Rückgabewert

  - *Liste von gefundenen Image Cool Notes*

3.3.6.8 `Interface MembershipRepository extends JpaRepository<Membership,String>`

Beschreibung
Repository für Mitgliedschaft

Methoden

- `public List<Membership> findByFlatshareId(String flatshareId)`

  Findet alle Mitgliedschaften über WG-ID

  Parameter

  - `flatshareId`
    WG-ID

  Rückgabewert

  - *Liste von gefundenen Mitgliedschaften*

- `public Membership findByFlatshareIdAndUserId(String flatshareId, String userId)`

  Findet eine Mitgliedschaft über WG-ID und Benutzer-ID

  Parameter

  - `flatshareId`
    WG-ID
  - `userId`
    Benutzer-ID

  Rückgabewert

  - *Gefundene Mitgliedschaft*

- `public void deleteByFlatshareIdAndUserId(String flatshareId, String userId)`

  Löscht eine Mitgiedschaft über WG-ID und Benutzer-ID

  Parameter

  - `flatshareId`
    WG-ID
  - `userId`
    Benutzer-ID

3.3.6.9 Interface ReadConfirmationRepository extends JpaRepository<ReadConfirmation,String>

Beschreibung
Repository für Lesebestätigung

Methoden

- `public List<ReadConfirmation> findByCoolNoteId(String id)`

  Findet alle Lesebestätigungen über CoolNote-ID

  Parameter

  - `id`
    Cool-Note-ID

  Rückgabewert

  - *Liste von gefundenen Lesebestätigungen*

- `public void deleteByCoolNoteIdAndUserId(String coolNoteId, String userId)`

  Löscht eine Lesebestätigung über Cool-Note-ID und Benutzer-ID

  Parameter

  - `coolNoteId`
    Cool-Note-ID
  - `userId`
    Benutzer-ID

3.3.6.10 Interface TaggedUserRepository extends JpaRepository<TaggedUser,String>

Beschreibung
Repository für getaggte Mitglieder

3.3.6.11 Interface UserRepository extends JpaRepository<User,String>

Beschreibung
Repository für User Entity

3.3.7 Package edu.kit.pse.fridget.server.services

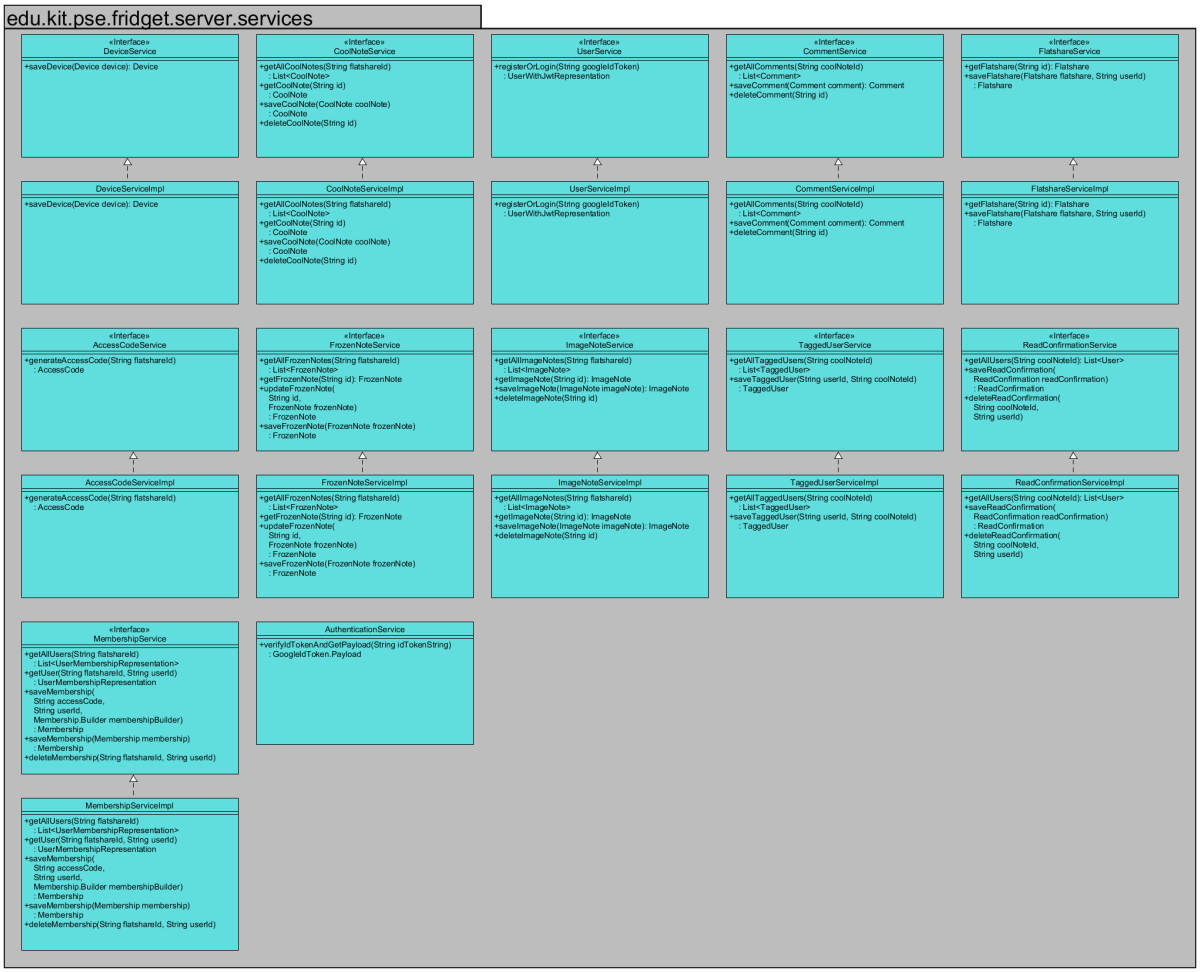

Abbildung 3.10: Klassen des Services (Server)

3.3.7.1 `Interface AccessCodeService`

Beschreibung
Interface von Service für Zugangscode

Methoden

- `public AccessCode generateAccessCode(String flatshareId)`
  Generiert einen zufälligen und eindeutigen Zugangscode für eine WG.
  Parameter
  - `flatshareId`
    WG-ID

  Rückgabewert
  - *Generierter Zugangscode*

3.3.7.2 `Class AccessCodeServiceImpl implements AccessCodeService`

Beschreibung
Service für Zugangscode

Konstruktor
`public AccessCodeServiceImpl(AccessCodeRepository repository)`

3.3.7.3 `Class AuthenticationService`

Beschreibung
Service für Authentifizierung

Methoden

- `public GoogleIdToken.Payload verifyIdTokenAndGetPayload(String idTokenString)`

  Prüft den Google-ID-Token.

  Parameter

  – `idTokenString`
  Google-ID-Token

  Rückgabewert

  – *Payload von Google-ID-Token*

3.3.7.4 `Interface CommentService`

Beschreibung
Interface von Service für Kommentar

Methoden

- `public List<Comment> getAllComments(String coolNoteId)`

  Findet alle Kommentare.

  Parameter

  – `coolNoteId`
  Cool-Note-ID

  Rückgabewert

  – *Liste von gefundenen Kommentaren*

- `public Comment saveComment(Comment comment)`

  Speichert einen Kommentar.

  Parameter

  – `comment`
  Kommentar zum Speichern

  Rückgabewert

  – *Gespeicherter Kommentar*

- `public void deleteComment(String id)`

  Löscht einen Kommentar.

  Parameter

  - `id`
    Kommentar-ID

3.3.7.5 `Class CommentServiceImpl implements CommentService`

Beschreibung
Service für Kommentar

Konstruktor
`public CommentServiceImpl(CommentRepository repository)`

3.3.7.6 `Interface CoolNoteService`

Beschreibung
Interface von Service für Cool Note

Methoden

- `public List<CoolNote> getAllCoolNotes(String flatshareId)`

  Findet alle Cool Notes mit getaggten Benutzern in einer WG.

  Parameter

  - `flatshareId`
    WG-ID

  Rückgabewert

  - *Liste von gefundenen Cool Notes mit getaggten Benutzern*

- `public CoolNote getCoolNote(String id)`

  Findet eine Cool Note mit getaggten Benutzern.

  Parameter

  - `id`
    Cool-Note-ID

  Rückgabewert

  - *Gefundene Cool Note mit getaggten Benutzern*

- `public CoolNote saveCoolNote(CoolNote coolNote)`

  Speichert eine Cool Note.

  Parameter

  - `coolNote`
    Cool Note zum Speichern

  Rückgabewert

  - *Gespeicherte Cool Note*

- `public void deleteCoolNote(String id)`

  Löscht eine Cool Note.

  Parameter

- `id`
  Cool-Note-ID

3.3.7.7 Class CoolNoteServiceImpl implements CoolNoteService

Beschreibung
Service für Cool Note

Konstruktor
`public CoolNoteServiceImpl(CoolNoteRepository coolNoteRepository, TaggedUserRepository taggedUserRepository)`

3.3.7.8 Interface DeviceService

Beschreibung
Interface von Service für Gerät

Methoden

- `public Device saveDevice(Device device)`

  Speichert ein neues Gerät.

  Parameter

  - `device`
    Gerät zum Speichern

  Rückgabewert

  - *Gespeichertes Gerät*

3.3.7.9 Class DeviceServiceImpl implements DeviceService

Beschreibung
Service für Gerät

Konstruktor
`public DeviceServiceImpl(DeviceRepository repository)`

3.3.7.10 Interface FlatshareService

Beschreibung
Interface von Service für WG

Methoden

- `public Flatshare getFlatshare(String id)`

  Findet eine WG.

  Parameter

  - `id`
    WG-ID

  Rückgabewert

  - *Gefundene WG*

- `public Flatshare saveFlatshare(Flatshare flatshare, String userId)`

  Speichert eine WG mit einem Namen.

  Parameter

  - `flatshare`
    WG zum Speichern
  - `userId`
    Benutzer-ID

  Rückgabewert

  - *Gespeicherte WG*

3.3.7.11 `Class FlatshareServiceImpl implements FlatshareService`

Beschreibung
Service für WG

Konstruktor
`public FlatshareServiceImpl(FlatshareRepository flatshareRepository, MembershipService membershipService, FrozenNoteService frozenNoteService)`

3.3.7.12 `Interface FrozenNoteService`

Beschreibung
Interface von Service für Frozen Note

Methoden

- `public List<FrozenNote> getAllFrozenNotes(String flatshareId)`

  Findet alle Frozen Notes in einer WG.

  Parameter

  - `flatshareId`
    WG-ID

  Rückgabewert

  - *Liste von gefundenen Frozen Notes*

- `public FrozenNote getFrozenNote(String id)`

  Findet eine Frozen Note.

  Parameter

  - `id`
    Frozen-Note-ID

  Rückgabewert

  - *Gefundene Frozen Note*

- `public FrozenNote updateFrozenNote(String id, FrozenNote frozenNote)`

  Updatet eine Frozen Note.

  Parameter

  - `id`
    Frozen-Note-ID

- `frozenNote`
  Frozen Note zum Updaten

Rückgabewert

- *Geupdatete Frozen Note*

- `public FrozenNote saveFrozenNote(FrozenNote frozenNote)`

  Speichert eine Frozen Note.

  Parameter

  - `frozenNote`
    Frozen Note zum Speichern

  Rückgabewert

  - *Gespeicherte Frozen Note*

3.3.7.13 `Class FrozenNoteServiceImpl implements FrozenNoteService`

Beschreibung
Service für Frozen Note

Konstruktor
`public FrozenNoteServiceImpl(FrozenNoteRepository repository)`

3.3.7.14 `Interface ImageNoteService`

Beschreibung
Interface von Service für Image Cool Note

Methoden

- `public List<ImageNote> getAllImageNotes(String flatshareId)`

  Findet alle Image Cool Notes.

  Parameter

  - `flatshareId`
    WG-ID

  Rückgabewert

  - *Liste von gefundenen Image Notes*

- `public ImageNote getImageNote(String id)`

  Findet eine Image Cool Note.

  Parameter

  - `id`
    Image-Note-ID

  Rückgabewert

  - *Gefundene Image Note*

- `public ImageNote saveImageNote(ImageNote imageNote)`

  Speichert eine Image Cool Note.

  Parameter

- `imageNote`
  Image Note zum Speichern

Rückgabewert

- *Gespeicherte Image Note*

- `public void deleteImageNote(String id)`

  Löscht eine Image Cool Note.

  Parameter

  - `id`
    Image-Note-ID

3.3.7.15 `Class ImageNoteServiceImpl implements ImageNoteService`

Beschreibung
Service für Image Cool Note

Konstruktor
`public ImageNoteServiceImpl(ImageNoteRepository repository)`

3.3.7.16 `Interface MembershipService`

Beschreibung
Interface von Service für Mitgliedschaft

Methoden

- `public List<UserMembershipRepresentation> getAllUsers(String flatshareId)`

  Findet alle Mitglieder in einer WG.

  Parameter

  - `flatshareId`
    WG-ID

  Rückgabewert

  - *Liste von gefundenen UserMembershipRepresentation*

- `public UserMembershipRepresentation getUser(String flatshareId, String userId)`

  Findet ein Mitglied in einer WG.

  Parameter

  - `flatshareId`
    WG-ID
  - `userId`
    Benutzer-ID

  Rückgabewert

  - *Gefundene UserMembershipRepresentation*

- `public Membership saveMembership(String accessCode, String userId, Membership.Builder membershipBuilder)`

  Speichert einen Benutzer mit gültigem Zugangscode in einer WG.

  Parameter

- `accessCode`
  Zugangscode
- `userId`
  Benutzer-ID
- `membershipBuilder`
  membershipBuilder

Rückgabewert

- *Gespeicherte Mitgliedschaft*

- `public Membership saveMembership(Membership membership)`

  Speichert eine Mitgliedschaft.

  Parameter

  - `membership`
    Mitgliedschaft zum Speichern

  Rückgabewert

  - *Gespeicherte Mitgliedschaft*

- `public void deleteMembership(String flatshareId, String userId)`

  Löscht ein Mitglied von einer WG.

  Parameter

  - `flatshareId`
    WG-ID
  - `userId`
    Benutzer-ID

3.3.7.17 `Class MembershipServiceImpl implements MembershipService`

Beschreibung
Service für Mitgliedschaft

Konstruktor
`public MembershipServiceImpl(MembershipRepository membershipRepository, UserRepository userRepository, AccessCodeRepository accessCodeRepository)`

3.3.7.18 `Interface ReadConfirmationService`

Beschreibung
Interface von Service für Lesebestätigung

Methoden

- `public List<User> getAllUsers(String coolNoteId)`

  Findet alle Leser einer Cool Note.

  Parameter

  - `coolNoteId`
    Cool-Note-ID

  Rückgabewert

- *Liste von gefundenen Benutzern*

- `public ReadConfirmation saveReadConfirmation(ReadConfirmation readConfirmation)`

  Speichert einen Benutzer als Leser einer Cool Note.

  Parameter

  - `readConfirmation`
    Lesebestätigung zum Speichern

  Rückgabewert

  - *Gespeicherte Lesebestätigung*

- `public void deleteReadConfirmation(String coolNoteId, String userId)`

  Löscht einen Benutzer als Leser einer Cool Note.

  Parameter

  - `coolNoteId`
    Cool-Note-ID
  - `userId`
    Benutzer-ID

3.3.7.19 Class ReadConfirmationServiceImpl implements ReadConfirmationService

Beschreibung
Service für Lesebestätigung

Konstruktor
`public ReadConfirmationServiceImpl(ReadConfirmationRepository readConfirmation UserRepository userRepository)`

3.3.7.20 Interface TaggedUserService

Beschreibung
Interface von Service für getaggte Mitglieder

Methoden

- `public List<TaggedUser> getAllTaggedUsers(String coolNoteId)`

  Findet alle getaggte Mitglieder.

  Parameter

  - `coolNoteId`
    Cool-Note-ID

  Rückgabewert

  - *Liste von gefundenen getaggten Mitgliedern*

- `public TaggedUser saveTaggedUser(String userId, String coolNoteId)`

  Speichert ein getaggtes Mitglied.

  Parameter

- `userId`
  Benutzer-ID
- `coolNoteId`
  Cool-Note-ID

Rückgabewert

- *Gespeichertes getaggtes Mitglied*

3.3.7.21 `Class TaggedUserServiceImpl implements TaggedUserService`

Beschreibung
Service für getaggte Mitglieder

Konstruktor
`public TaggedUserServiceImpl(TaggedUserRepository repository)`

3.3.7.22 `Interface UserService`

Beschreibung
Interface von Service für Benutzer

Methoden

- `public UserWithJwtRepresentation registerOrLogin(String googleIdToken)`
  Authentifiziert einen Benutzer durch Google-ID-Token.

  Parameter

  - `googleIdToken`
    Google-ID-Token

  Rückgabewert

  - *Gespeicherter oder angemeldeter Benutzer mit JWT*

3.3.7.23 `Class UserServiceImpl implements UserService`

Beschreibung
Service für Benutzer

Konstruktor
`public UserServiceImpl(UserRepository repository)`

4 Datenstrukturen

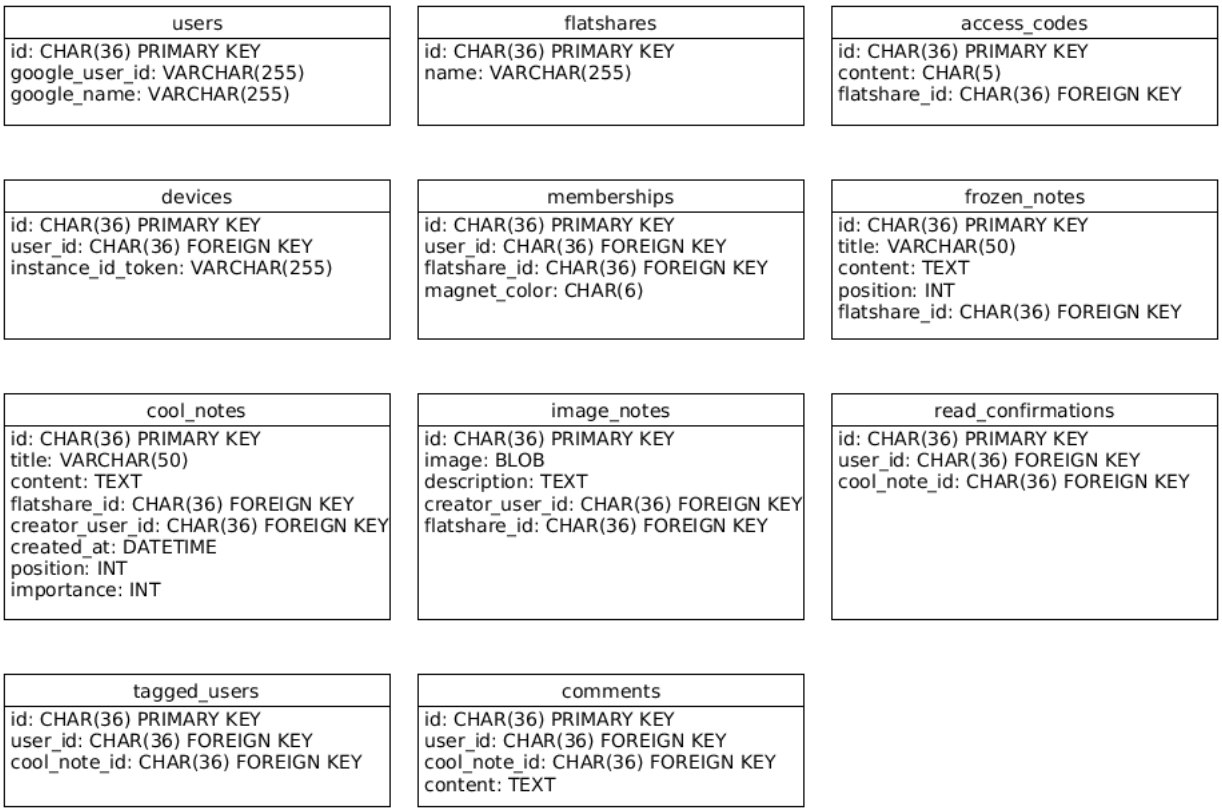

Abbildung 4.1: Datenstrukturen

In der Abbildung 4.1 sieht man die Datenstruktur der Datenbank, also die Tabellen, die es in der Datenbank geben wird.

5 Dynamische Modelle

5.1 Aktivitätsdiagramm

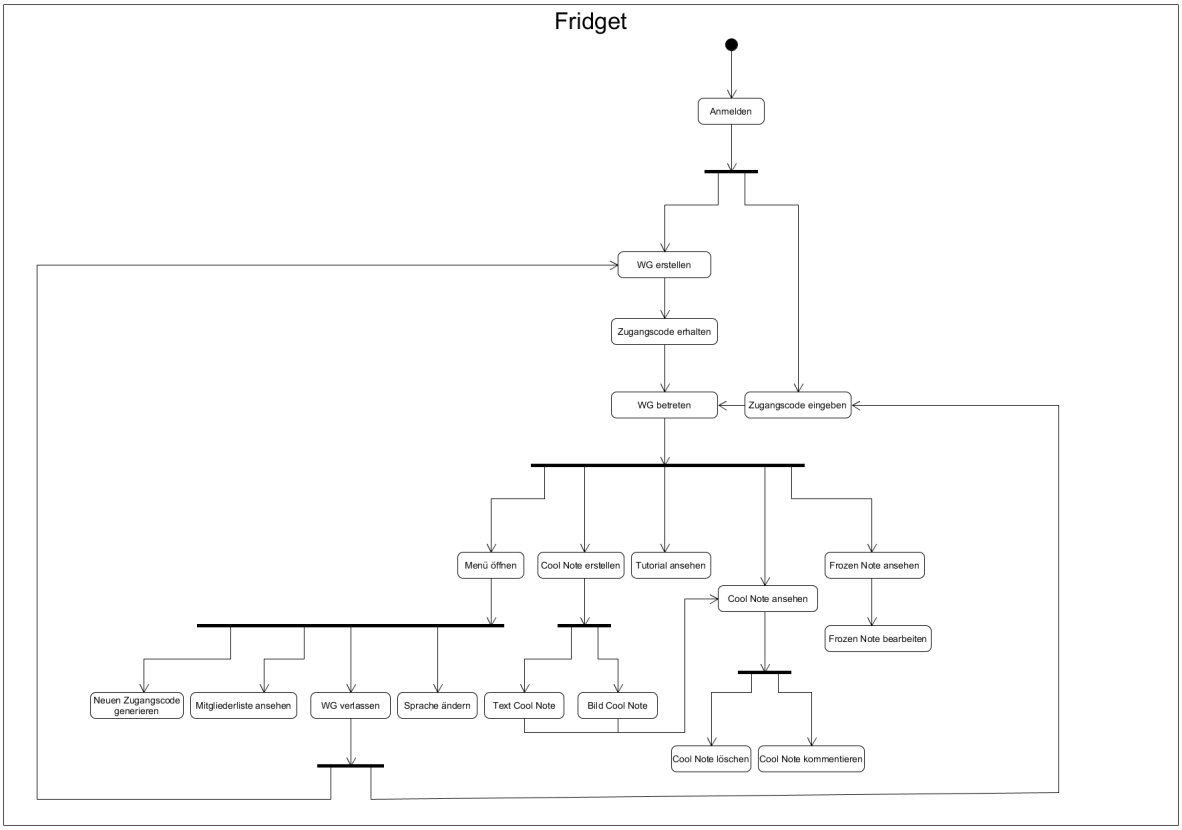

Abbildung 5.1: Aktivitätsdiagramm

In der Abbildung 5.1 sieht man ein Aktivitätsdiagramm für unsere App, welches alle Aktivitäten zeigt, die ausgeführt werden können. Man sieht auch die Beziehungen zwischen den Aktivitäten.

5.2 Login

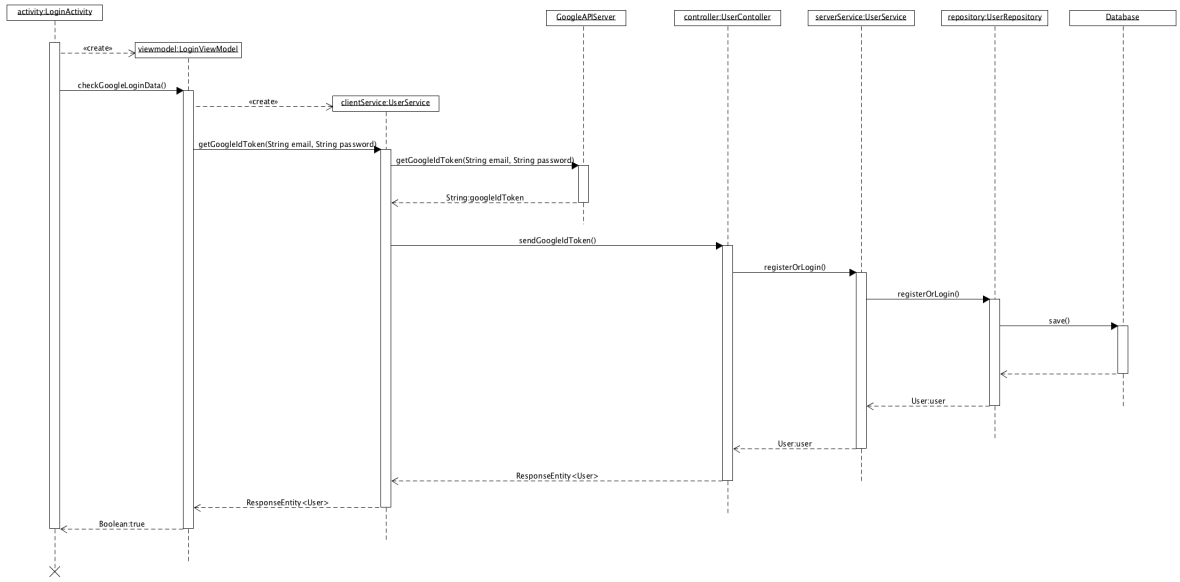

Abbildung 5.2: Login

5.2.1 Ablauf der Methode checkGoogleLoginData()

Die Methode checkGoogleLoginData() prüft die eingegebenen Google-Login-Daten des Users. Sie wird in der Activity LoginActivity ausgeführt. Als Erstes wird das zugehörige Viewmodel zur Activity, nämlich das LoginViewModel, verwendet. Darauffolgend wird die Methode checkGoogleLoginData(), die in der View-Model-Klasse steht, aufgerufen. In dieser Methode wird der zugehörige Service des Clients, UserService, erstellt, der dann die Verknüpfung zur Server-Seite herstellt. Die Methode getGoogleIdToken des Services wird aufgerufen und als Parameter werden die vom User eingegebenen Login-Daten (E-Mail, Passwort) übergeben. Es wird ein HTTP GET Request an den Google-API-Server gesendet, welcher dann den Google-ID-Token zurücksendet. Der Token wird nun mithilfe der Methode sendGoogleIdToken() an den Server bzw. den zugehörigen UserController gesendet. Im UserController wird die Methode registerOrLogin() aufgerufen. Im UserService des Servers wird ebenfalls eine Methode registerOrLogin() aufgerufen. Das UserRepository ist verantwortlich für die Datenpersistenz und speichert nun also die User-Daten in der Datenbank ab. Auf dem Rückweg zur Activity wird schlussendlich der User übergeben und eine Response gesendet.

5.3 Erstellen einer Cool Note

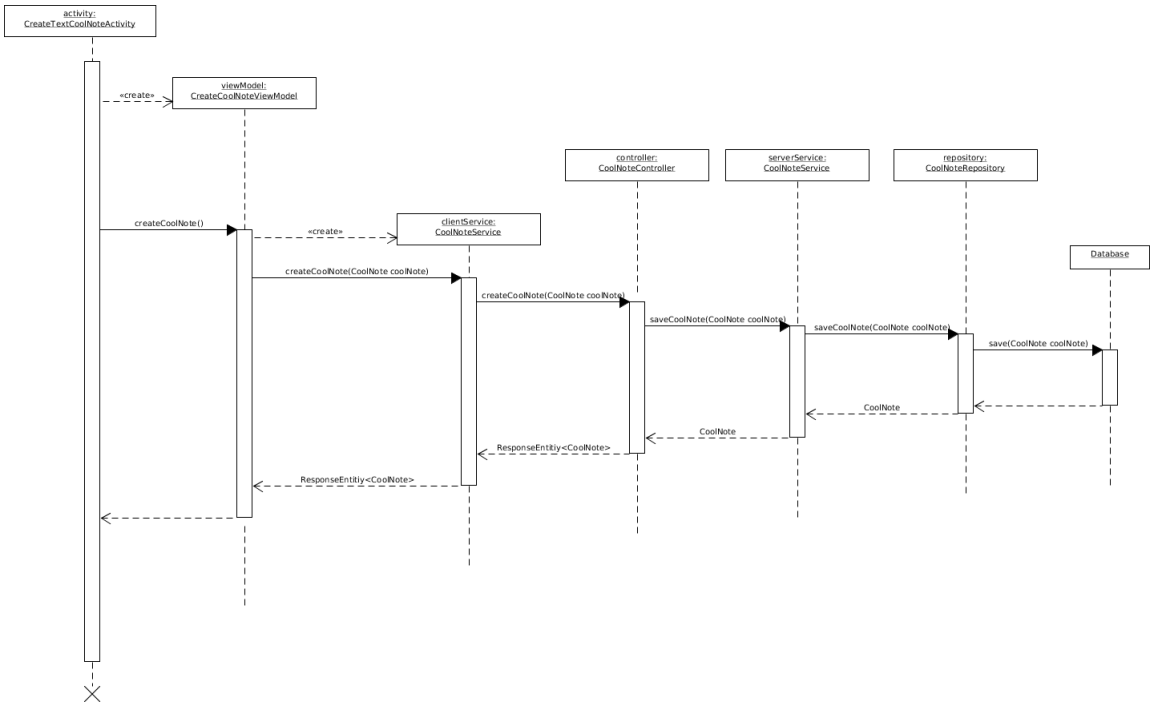

Abbildung 5.3: CoolNote erstellen

5.3.1 Ablauf der Methode createCoolNote()

Die Methode createCoolNote() erstellt eine Cool Note und platziert sie auf der WG-Pinnwand. Sie wird in diesem Fall in der Activity CreateTextCoolNoteActivity ausgeführt. Als Erstes wird das zugehörige Viewmodel zur Activity, nämlich das CreateCoolNoteViewModel, verwendet. Darauffolgend wird die Methode createCoolNote(), die in der View-Model-Klasse steht, aufgerufen. In dieser Methode wird der zugehörige Service des Clients, CoolNoteService, erstellt, der dann die Verknüpfung zur Server-Seite herstellt. Die Methode createCoolNote() des Services wird aufgerufen und als Parameter wird eine Instanz des Objektes CoolNote übergeben, welches die vom Benutzer eingetippten Attribute enthält (Titel, Text-Inhalt, gegebenenfalls Wichtigkeit, Tags, etc.). Es wird ein HTTP POST Request an den Server gesendet bzw. an den zugehörigen CoolNoteController. Im CoolNoteController wird die Methode saveCoolNote() aufgerufen. Im CoolNoteService des Servers wird ebenfalls eine Methode saveCoolNote aufgerufen. Das CoolNoteRepository ist verantwortlich für die Datenpersistenz und speichert nun also die Cool-Note-Daten in der Datenbank ab. Auf dem Rückweg zur Activity wird schlussendlich die erstellte Cool Note übergeben und eine Response gesendet.

5.4 Frozen Note bearbeiten

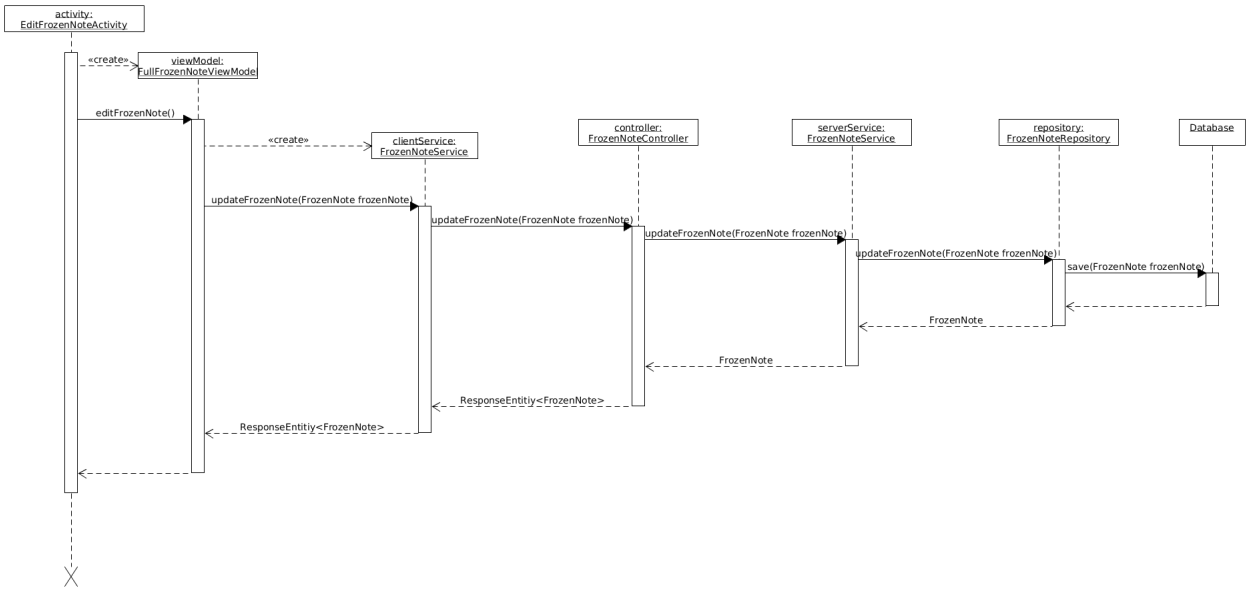

Abbildung 5.4: FrozenNote bearbeiten

5.4.1 Ablauf der Methode editFrozenNote()

Die Methode editFrozenNote() aktualisiert den Inhalt und den Titel einer Frozen Note. Sie wird in der Activity EditFrozenNoteActivity ausgeführt. Als Erstes wird das zugehörige Viewmodel zur Activity, nämlich das FullFrozenNoteViewModel, verwendet. Darauffolgend wird die Methode editFrozenNote(), die in der View-Model-Klasse steht, aufgerufen. In dieser Methode wird der zugehörige Service des Clients, FrozenNoteService, erstellt, der dann die Verknüpfung zur Server-Seite herstellt. Die Methode updateFrozenNote() des Services wird aufgerufen und als Parameter wird die bereits vorhandene Instanz des Objektes FrozenNote übergeben, welches die vom Benutzer eingetippten Attribute enthält (Titel, Text-Inhalt). Es wird ein HTTP PUT Request mithilfe an den Server gesendet bzw. an den zugehörigen FrozenNoteController. Im FrozenNoteController wird die Methode updateFrozenNote() aufgerufen. Im FrozenNoteService des Servers wird ebenfalls eine Methode updateFrozenNote() aufgerufen. Das FrozenNoteRepository ist verantwortlich für die Datenpersistenz und speichert nun also die neuen Frozen-Note-Daten in der Datenbank ab. Auf dem Rückweg zur Activity wird schlussendlich die editierte Frozen Note übergeben und eine Response gesendet.

5.5 WG verlassen

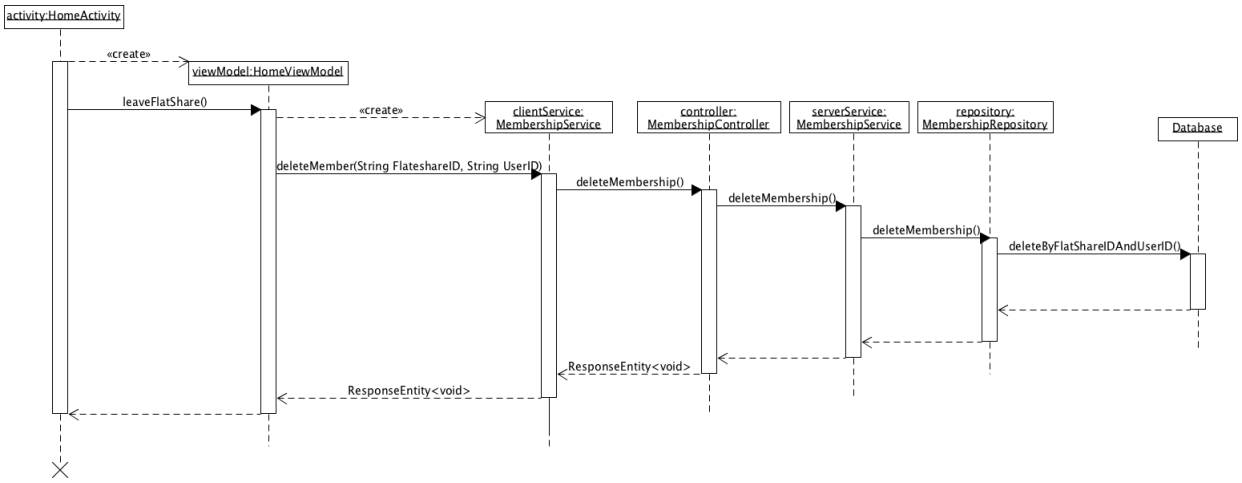

Abbildung 5.5: WG verlassen

5.5.1 Ablauf der Methode leaveFlatshare()

Die Methode leaveFlatshare() dient zum Verlassen einer WG. Sie wird in der Activity HomeActivity ausgeführt. Als Erstes wird das zugehörige Viewmodel zur Activity, nämlich das HomeViewModel, verwendet. Darauffolgend wird die Methode leaveFlatShare(), die in der View-Model-Klasse steht, aufgerufen. In dieser Methode wird der zugehörige Service des Clients, MembershipService, erstellt, der dann die Verknüpfung zur Server-Seite herstellt. Die Methode deleteMember() des Services wird aufgerufen und als Parameter werden die FlatshareId und die UserId übergeben. Es wird ein HTTP DELETE Request an den Server gesendet bzw. an den zugehörigen Controller. Im MembershipController wird die Methode deleteMembership() aufgerufen. Im MembershipService des Servers wird ebenfalls eine Methode deleteMembership() aufgerufen. Das MembershipRepository ist verantwortlich für die Datenpersistenz und löscht nun also die User-Daten aus der Datenbank. Auf dem Rückweg zur Activity wird schlussendlich eine Response gesendet.

6 Änderungen zum Pflichtenheft

6.1 Kein Archiv

Wir haben uns dazu entschieden, das Archiv nicht umzusetzen, weil wir andere Wunschkriterien als wichtiger sehen. Außerdem glauben wir, dass Zeitprobleme anfallen würden.

6.2 Zugangscode-Regelung

Damit das Zugangscode-Verfahren sicherer ist, haben wir uns dazu entschieden, dass man für jede Einladung einen neuen Zugangscode manuell generieren muss. Dies geschieht dann in der AccessCodeActivity mithilfe eines Buttons. Davor gab es einen dauerhaften Zugangscode für die gesamte WG, nun sind die Zugangscodes einmalig und werden nach der Benutzung gelöscht.

7 Glossar

Activity	Stellt eine Aktivität in einer App dar
Activity-Lifecycle	Stufen, die eine Activity während ihres Lebens durchschreitet
API	Application Programming Interface - Schnittstelle, die ein Softwaresystem bereitstellt um dieses in andere Programme einzubinden
Button	Knopfelement einer Benutzeroberfläche
Check Box	Eine Box, die abgehakt werden kann
Client	Eine App auf einem Endgerät bzw. das Endgerät selber
Cool Note	Notizen, die löschbar und nicht-bearbeitbar sind und vom Benutzer erstellt werden
Data Binding	Automatische Datenweitergabe zwischen Objekten
Framework	Ein Programmiergerüst
Frozen Note	Notizen, die fest, nicht-löschbar und bearbeitbar sind und beim Erstellen der WG generiert werden.
Geschäftslogik	Mittelschicht einer mehrschichtigen Anwendung
HTTP	Zustandsloses Protokoll zur Übertragung von Daten auf der Anwendungsschicht über ein Rechnernetz
ID	Identifikation
Instanz	Objekt
Interface	Schnittstelle, Übergangsstelle zwischen verschiedenen Komponenten
JSON	JavaScript Object Notation - Datenaustauschformat, das für Menschen einfach zu lesen und für Maschinen einfach zu analysieren und generieren ist
JWT	JSON Web Token - Ein auf JSON basiertes und nach RFC 7519 genormtes Access-Token
Layout	Text- und Bildgestaltung
Methoden	Unterprogramme, in der Form von Funktionen, die das Verhalten von Objekten beschreiben und implementieren
MVC	Model View Controller
MVVM	Model View Viewmodel
Open-Source	Quelltext, welcher öffentlich und von dritten eingesehen, geändert und genutzt werden kann
Parameter	Übergabewert
POJO	Plain Old Java Object - Java-Klasse ohne komplexe Strukturen
Repository	Verwaltetes Verzeichnis zur Speicherung und Beschreibung von Objekten

Server	Ein Programm, das auf die Kontaktaufnahme eines Clients wartet, um eine bestimmte Dienstleistung für ihn zu erfüllen.
Tag / Taggen	Markierung und namentliche Erwähnung von Mitgliedern auf Notizzetteln
Token	Hardwarekomponente zur Identifizierung und Authentifizierung von Benutzern
Tutorial	Einführung in die Funktionen der App
URL	Uniform Resource Locator - Identifiziert und lokalisiert eine Ressource
UUID	Universally Unique Identifier - Ein Standard für Identifikatoren, der aus 16 Bytes besteht

8 Anhang

8.1 Klassendiagramm Client-Server

Abbildung 8.1: Client-Server Klassendiagramm