Entwurf

Fangzhou Bian, Kathrin Blum, Matthias Bruns,
Leonhard Duda, Tan Grumser, Yuguang Lin

2. Juli 2019

Inhaltsverzeichnis

1 Einleitung

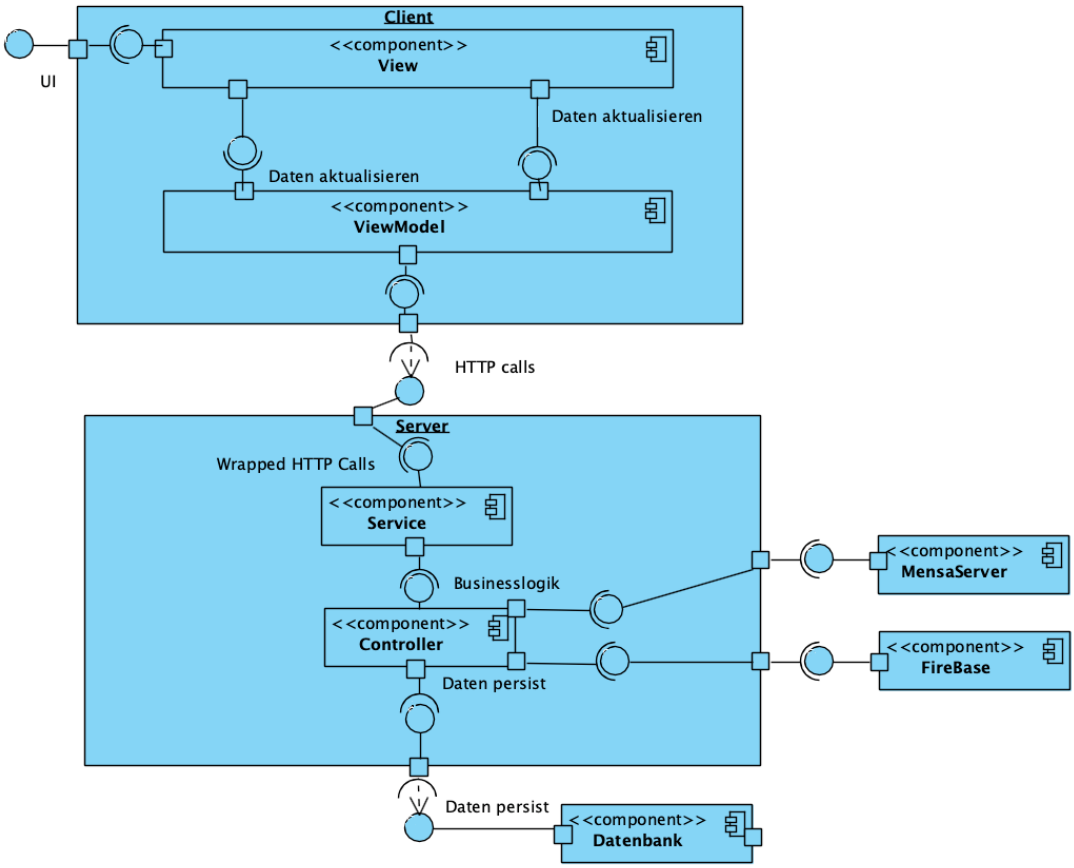
Dies ist das Entwurfsdokument für die Applikation ”MensaMeet”. Es ist das Artefakt der Entwurfsphase, die direkt an die Definitionsphase anschließt, aus welcher das Pflichtenheft hervorging. In den folgenden Kapitel wird auf die Systemarchitektur (”Grobentwurf”) eingegangen. Es folgt die Erläutereung der Systemkomponente, sowie der verwendeten Frameworks und Schnittstellen. Im Kapitel ”Feinentwurf”werden die Klassendiagramme der einzelnen Pakete im Detail vorgestellt und das verwendete HTTP Protokoll aufgeührt. Unter ”Datenstrukturen”findet sich die Strukturiereung unserer Datenbank wieder. Im Abschnitt ”dynamische Modelle”werden an Hand von Sequenzdiagrammen einige Abläufe von MensaMeet anschaulich dargestellt. Schließlich Begründen wir in Kapitel 6 die Änderungen gegenüber dem Pflichtenheft. Auf den letzten Seiten dieses Dokuments findet sich ein Glossar und ein vollständiges Klassendiagramm für die gesamte Applikation MensaMeet.

2 Grobentwurf

2.1 Systemarchitektur

Die App MensaMeet wird als Client-Server Anwendung umgesetzt. Clients fordern Dienste an, welche vom zentralen Server bereitgestellt werden. Dazu erhält jeder User der sich registriert ein eindeutiges UserToken, das auf seinem Endgerät gespeichert wird. Der Server verwaltet eine Datenbank, in welcher die Mensadaten und User- und Gruppeninformationen gehalten werden. Die Client-Seite der Anwendung wird mithilfe der MVVM (Model-View-Viewmodel) Architekur umgesetzt. Die Server-Seite mithilfe der MVC (Model-View-Control) Architektur.

2.2 Systemkomponenten



2.3 Komponentenbeschreibung

2.3.1 Client

Die Client Seite bedient sich der MVVM Architektur. Die View dient zum anzeigen von Daten, sie beinhaltet alle grafischen Elemente, stellt also die GUI dar. Im Model werden die Daten gehalten und Funktionen zur Manipulation der Daten bereitgestellt. Das Viewmodel enthält die Darstellungslogik für die View und kapselt sie von der Anwendungslogik, die sich im Model befindet, ab. View und Viewmodel sind durch Databinding nur lose gekoppelt, dadurch kann die View problemlos ausgetauscht werden bzw. können mehrere verscheidene Views (z.b. angepasst an Tablet, Desktop,...) zum selben Viewmodel existieren. Das Datainding kann als bidirektionaler Beobachter verstanden werden: View und Viewmodel kennen die interne Struktur voneinander nicht, aber Benachrichtigen einander, wenn es Änderungen gibt. Diese lose Kopplung durch Databinding verbessert die Testbarkeit, da so View und Viewmodel unabhängig voneiander getestet werden können.

2.3.2 Server

Für die Server Seite wird die MVC Architektur verwendet. Das Model ist für die Datenhaltung zuständig und beinhaltet/verwaltet eine Datenbank. In unserem Fall wird die Relationale Datenbank MySQL verwendet. Als Kommunikationsschnittstelle zwischen der Datenbank und den Controllern des Servers wird JPA (Java Persistance API) mit Hibernate genutzt. View ist durch eine Schnittstelle gegeben, über die der Server Anfragen des Client empfängt und an den Controller delegiert. Der Controller führt diese Anfragen durch, indem er angefordere Daten aus dem Model holt oder neue Daten einfügt bzw Daten anpasst.

2.3.3 Frameworks und Interfaces

Hibernate und JPA

Objektrelationales Mapping (ORM) ist eine Technik der Softwareentwicklung, mit der ein (in objektorientierten Programmiersprache geschriebenes) Programm seine Objekte in einer relationalen Datenbank ablegen kann. Dem Programm wird dazu eine objektorientierte Sicht auf die Tabellen und Beziehungen im Datenbankmanagementsystem (DBSM) geboten.

Wir benutzen für ORM das Java Persistence Application Programming Interface (JPA). JPA ist eine API für Datenbankzugriffe und objektrelationales Mapping und muss durch einen sogenannten JPA Provider implemetiert werden. Wir verwenden hierzu das Framework Hibernate.

Hibernate ermöglicht es normale Objekte (Plain Old Java Object, POJO) in relationalen Datenbanken zu speichern und aus den Datensätzen der Datenbank Objekte zu erzeugen.
Hibernate bietet Kompatibilität mit verschiedenen Datenbanken, sodass die von uns verwendete Datenbank MySQL leicht gegen eine andere relationale Datenbank ausgetauscht werden kann.

Spring Boot

Spring Boot ist ein Framework, welches es erleichtern soll Java und Java EE Anwendungen zu schreiben und gute Programmierpraktiken zu fördern. Es ist eine vereinfachte Form des Spring Frameworks und bietet viele dessen Funktionalitäten auf schnell zu integierbarem Wege an.

Wir benutzen Spring Boot, um schnell eine exportierbare direkt einsatzfähige Anwendung schreiben zu können und um Funktionalitäten wie Dependency Injection nutzen zu können.

Firebase

Firebase ist ein Framework, das viele Services bietet um die Entwicklung und die Handhabung der Infrastruktur von mobilen und Webanwendungen zu vereinfachen. Es übernimmt für uns die Registrierung, Speicherung und Organisation von Nutzerdaten, die wir für die Verwaltung von Accounts benötigen.

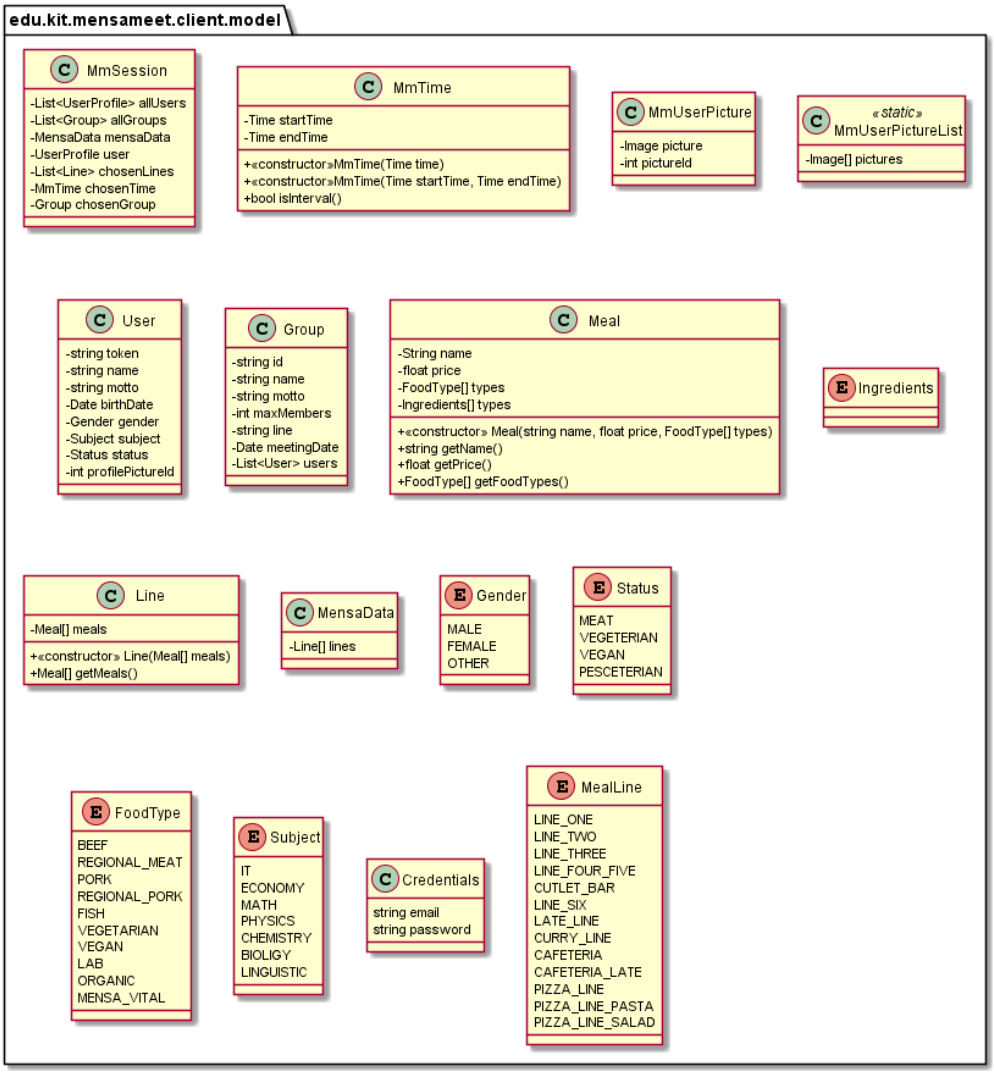
Wir nutzen von den Services die Firebase bietet die Registrierung und Anmeldung mit einer Email-Adresse und einem Passwort, sowie die Möglichkeit zur Wiederherstellung des Passworts. Firebase bietet viele weitere Funktionalitäten, wie Push Notifications, an, die wir, bei Bedarf, in Zukunft nutzen können, um Wunschkriterien zu implementieren.

3 Feinentwurf

3.1 Klassen des Clients

3.1.1 package.edu.kit.mensameet.client.model

Client Model - Paketdiagramm



3.1.1.1 Class MmSession

Beschreibung

In dieser Klasse können die Daten einer Sitzung wie Benutzer, ausgewählte Zeit, ausgewählte Linien und die ausgewählte Gruppe gespeichert werden. Bei ihrer Erzeugung werden zusätzlich noch Listen aller Benutzer bzw. Gruppen sowie die Mensadaten hineingeladen.

3.1.1.2 Class MmTime

Beschreibung

Diese Klasse kann eine Zeit oder einen Zeitraum speichern.

Konstruktoren

- `MmTime(Time time)`
  Dieser Konstruktor legt eine Zeit an.

  ###### Parameter

  - time
    Zeit.

- `MmTime(Time startTime, Time endTime)`
  Dieser Konstruktor legt einen Zeitraum an.

  ###### Parameter

  - startTime
    Startzeit.
  - endTime
    Endzeit.

Methoden

- `bool isInterval()`
  Diese Methode gibt zurück, ob es sich um ein Zeitintervall handelt.

  ###### Rückgabewert

  Ob Zeitintervall.

3.1.1.3 Class MmUserPicture

Beschreibung

Diese Klasse enthält eine Profilbilddatei mit zugehöriger Id.

3.1.1.4 Static class MmUserPictureList

Beschreibung

Diese Klasse enthält die Liste aller möglichen Profilbilder.

3.1.1.5 Class Group

Die folgenden Klassen enthalten Standardkonstruktoren, die nicht explizit aufgeführt werden: User, Group.

Beschreibung

Dies ist eine Datenhalterklasse für eine Gruppe.

3.1.2 Class User

Beschreibung

Dies ist eine Datenhalterklasse für einen User.

3.1.3 Class Credentials

Beschreibung

Diese Klasse beschreibt die Zugangsdaten (engl. Crendentials) eines Nutzers.

3.1.4 Class MensaData

Beschreibung

Diese Klasse beinhaltet den Speiseplan der Mensa des aktuellen Tages.

3.1.5 Class Line

Beschreibung

Diese Klasse beschreibt den Speiseplan der einzelnen Linien bzw. Werke an der Mensa.

3.1.6 Class Meal

Beschreibung

Diese Klasse beschreibt die einzelnen Speisen die an der Mensa angeboten werden. Dazu gehören die Zutaten (enum Ingredients) und der Typ (enum Foodtype).

3.1.7 Class Preferences

Beschreibung

In dieser Klasse sind die Preferenzen des Nutzers zur Gruppensuche.

3.1.7.1 Enum Gender

Beschreibung

Die Geschlechter, die zu einem User gehören können.
MALE - das männliche Geschlecht
FEMALE - das weibliche Geschlecht
OTHER - weder männlicht, noch weiblich

3.1.7.2 Enum Subject

Beschreibung

Die Fachrichtungen die ein User haben kann:

Angewandte Geowissenschaften
Architektur
Bauingenieurwese
Bioingenieurwesen
Biologie
Chemie
Chemieingenieurwesen und Verfahrenstechnik
Chemische Biologie
Deutsch
Elektro- und Informationstechnik
Energy Engineering and Management
Europäische Kultur und Ideengeschichte
Financial Engineering
Funktionaler und Konstruktiver Ingenieurbau-Engineering Structures
Geodäsie und Geoinformatik
Geographie
Geoökologie
Geophysik

Germanistik
Information Systems Engineering and Management
Informatik
Ingenieurpädagogik
Kunstgeschichte
Lebensmittelchemie
Management of Product Development
Maschinenbau
Materialwissenschaft und Werkstofftechnik
Mathematik
Mechanical Engineering
Mechatronik und Informationstechnik
Meteorologie
Mobilität und Infrastruktur
Mobility Systems Engineering and Management
Naturwissenschaft und Technik
Optics and Photonics
Pädagogik
Philosophie/Ethik
Physik
Production and Operations Management
Regionalwissenschaft
Remote Sensing and Geoinformatics
Sport
Sportwissenschaft
Technische Volkswirtschaftslehre
Technomathematik
Water Science and Engineering
Wirtschaftsinformatik
Wirtschaftsingenieurwesen
Wirtschaftsmathematik
Wissenschaft-Medien-Kommunikation

3.1.7.3 Enum Status

Beschreibung

Der Stauts den ein User haben kann:

Professor - Ein/e Professor/in oder Dozent/in
CollegeStudent - Ein/e Schüler-Student/in
Apprentice - Ein/e Auszubildende/r
Student - Ein/e reguläre/r Student/in

Other - Sonstiges

3.1.7.4 Enum FoodType

Beschreibung

Die Sorten von Essen die es gibt und die man Gerichten zuordnen kann:

BEEF
REGIONAL_MEAT
PORK
REGIONAL_PORK
FISH
VEGETARIAN
VEGAN
LAB
ORGANIC
MENSA_VITAL

3.1.7.5 Enum Ingredient

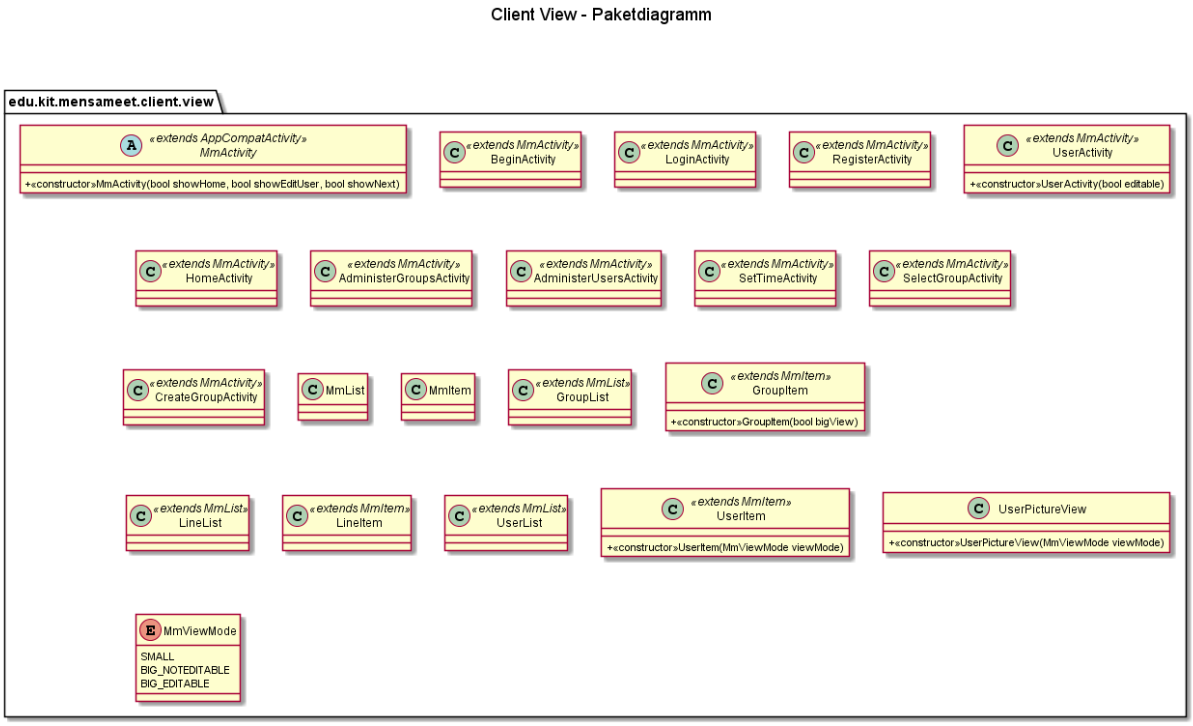
Beschreibung

Die Inhaltsstoffe, die in einem Gericht enthalten sein können:

(1) mit Farbstoff
(2) mit Konservierungsstoff
(3) mit Antioxidationsmittel
(4) mit Geschmacksverstärker
(5) mit Phosphat
(6) Oberfläche gewachst
(7) geschwefelt
(8) Oliven geschwärzt
(9) mit Süßungsmittel
(10) kann bei übermäßigem Verzehr abführend wirken
(11) enthält eine Phenylalaninquelle
(12) kann Restalkohol enthalten
(14) aus Fleischstücken zusammengefügt
(15) mit kakaohaltiger Fettglasur
(27) aus Fischstücken zusammengefügt
(R) enthält Rindfleisch
(RAT) enthält regionales Rindfleisch aus artgerechter Tierhaltung
(S) enthält Schweinefleisch
(SAT) enthält regionales Schweinefleisch aus artgerechter Tierhaltung
(VEG) vegetarisches Gericht
(VG) veganes Gericht (ohne Fleischzusatz)

(B) kontrolliert biologischer Anbau / DE007 Öko Kontrollstelle
(MSC) MSC-zertifizierter Fisch
(MV) MensaVital
(LAB) mit tierischem Lab
(GER) mit Gelatine
(Gl) Glutenhaltiges Getreide
(We) Weizen
(Ro) Roggen
(Ge) Gerste
(Ha) Hafer
(Di) Dinkel
(Ka) Kamut
(Nu) Schalenfrüchte / Nüsse
(Ma) Mandeln
(Ha) Haselnüsse)
(Wa) Walnüsse
(Ca) Cashewnüsse
(Pe) Pekanüsse
(Pa) Paranüsse
(Pi) Pistazie
(Qu) Queenslandnüsse/Macadamianüsse
(Ei) Eier
(Er) Erdnüsse
(So) Soja
(Sn) Senf
(Kr) Krebstiere
(Fi) Fisch
(ML) Milch / Laktose
(Se) Sellerie
(Sf) Schwefeldioxid / Sulfit
(Sa) Sesam
(Lu) Lupine
(We) Weichtiere

3.1.8 package.edu.kit.mensameet.client.view



3.1.8.1 Abstract class MmActivity extends AppCompatActivity

Beschreibung

Diese abstrakte Klasse bildet das Grundgerüst für die Activitys.

Konstruktoren

- `MmActivity(bool showHome, bool showEditUser, bool showNext)`

  Mit diesem Konstruktor lässt sich einstellen, welche Buttons angezeigt werden sollen.

  ###### Parameter

  - showHome
    Zeige Home-Button an.
  - showEditUser
    Zeige Profil-Button an.

– showNext
Zeige Weiter-Button an.

3.1.8.2 Class BeginActivity extends MmActivity

Beschreibung

Diese Oberfläche empfängt den Benutzer, wenn dieser die App das erste Mal startet. Er kann sich dann mit seinem bereits vorhandenem Account anmelden, falls er zum Beispiel nur ein neues Smartphone hat. Alternativ hat er die Möglichkeit sich zu registrieren und einen neuen Account anzulegen.

3.1.8.3 Class LoginActivity extends MmActivity

Beschreibung

Diese Klasse bietet die Oberfläche der App, in der sich der Benutzer mit seinen Accountdaten anmelden kann.

3.1.8.4 Class UserActivity extends MmActivity

Beschreibung

Diese Klasse zeigt ein Benutzerprofil auf einer ganzen Seite an.

Konstruktoren

- `UserActivity(bool editable)`

  Mit diesem Konstruktor lässt sich einstellen, ob das Profil editierbar sein soll.

  Parameter

  – editable
  Profil soll editierbar sein.

3.1.8.5 Class HomeActivity extends MmActivity

Beschreibung

Diese Klasse bietet die Oberfläche der App, die als Homescreen für den Benutzer agiert. Der Benutzer kann hier seine bevorzugten Linien auswählen, an denen er gerne essen gehen würde.

3.1.8.6 Class AdministerGroupsActivity extends MmActivity

Beschreibung

Diese Klasse zeigt für Administratoren eine Übersicht aller Gruppen an und stellt eine Löschtaste zur Verfügung.

3.1.8.7 Class AdministerUsersActivity extends MmActivity

Beschreibung

Diese Klasse zeigt für Administratoren eine Übersicht aller Benutzer an und stellt eine Löschtaste zur Verfügung.

3.1.8.8 Class SelectGroupActivity extends MmActivity

Beschreibung

Diese Klasse bietet die Oberfläche der App, die dem Benutzer verschiedene Gruppen anzeigt, die anhand seiner davor eingegebenen Präferenzen ausgewählt wurden. Er kann hier die Gruppen antippen, um sich weitere Details über diese anzeigen zu lassen.

3.1.8.9 Class CreateGroupActivity extends MmActivity

Beschreibung

Diese Klasse bietet die Oberfläche der App, die es dem Benutzer ermöglicht eine neue Gruppe zu gründen.

3.1.8.10 Class MmList

Beschreibung

Diese Klasse dient als Grundgerüst für Listen in den Activitys.

3.1.8.11 Class MmItem

Beschreibung

Diese Klasse dient als Grundgerüst für Listenelemente in den Activitys.

3.1.8.12 Class GroupList extends MmList

Beschreibung

Diese Klasse bietet eine Oberfläche für eine Gruppenliste.

3.1.8.13 Class GroupItem extends MmItem

Beschreibung

Diese Klasse bietet die Oberfläche der App, die dem Benutzer eine Gruppe und seine Teilnehmer anzeigt.

Konstruktoren

- `GroupItem(bool bigView)`
  Mit diesem Konstruktor lässt sich einstellen, ob die Gruppenansicht in groß für eine Einzelseite oder, für Listen geeignet, in klein erfolgen soll.

  Parameter

  - bigView
    Ob die Gruppenansicht groß sein soll.

3.1.8.14 Class LineList extends MmList

Beschreibung

Diese Klasse bietet eine Oberfläche für eine Liste von Mensalinien.

3.1.8.15 Class LineItem extends MmItem

Beschreibung

Diese Klasse bietet eine Oberfläche für ein Listenelement einer Mensalinien-Liste.

3.1.8.16 Class UserList extends MmList

Beschreibung

Diese Klasse bietet eine Oberfläche für eine Benutzerliste.

3.1.8.17 Class UserItem extends MmList

Beschreibung

Diese Klasse bietet die Oberfläche der App, die einen Benutzer anzeigt.

Konstruktoren

- `UserItem(MmViewMode viewMode)`
  Mit diesem Konstruktor lässt sich einstellen, ob die Benutzeransicht in groß für eine Einzelseite, dabei bearbeitbar oder, für Listen geeignet, in klein erfolgen soll.

Parameter

- viewMode
  Anzeigeart: groß nur lesen, groß bearbeitbar oder klein.

3.1.8.18 Class UserPictureView

Beschreibung

Diese Klasse kapselt die Darstellung eines Benutzerbildes.

Konstruktoren

- `UserPictureView(MmViewMode viewMode)`
  Mit diesem Konstruktor lässt sich einstellen, ob das Benutzerbild in groß für eine Einzelseite, dabei bearbeitbar oder, für Listen geeignet, in klein erfolgen soll.

Parameter

- viewMode
  Anzeigeart: groß nur lesen, groß bearbeitbar oder klein.

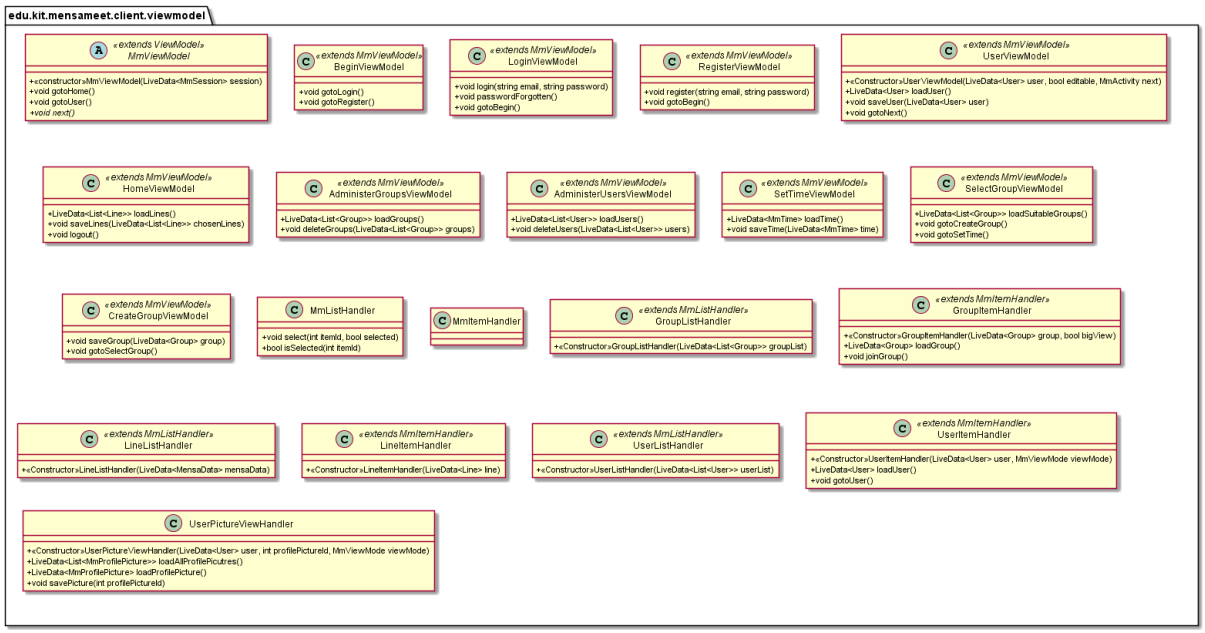
3.1.8.19 enum MmViewMode

Beschreibung

Anzeigearten *SMALL* - klein
BIGNOTEDITABLE - gross und nicht bearbeitbar
BIGEDITABLE - gross und bearbeitbar

3.1.9 package.edu.kit.mensameet.client.viewmodel

Client ViewModel - Paketdiagramm



3.1.9.1 Abstract class MmViewModel extends ViewModel

Beschreibung

Diese abstrakte Klasse verwaltet alle Grundfunktionen, die die Klasse MmActivity ihren Unterklassen bereitstellt.

Konstruktoren

- `MmViewModel(LiveData<MmSession> session)`

  Dieser Konstruktor nimmt ein MmSession-Element auf, in der die Daten der Sitzung gespeichert werden.

  ###### Parameter

  – session
  Sitzungs-Klasse. Falls leer, wir eine neue erzeugt.

Methoden

- `void gotoHome()`
  Diese Methode startet die Klasse HomeActivity.
- `void gotoUser()`
  Diese Methode startet die Klasse UserActivity.
- `abstract void next()`
  Diese Schablonenmethode kann von jeder abgeleiteten Klasse individuell für die folgende Activity angepasst werden.

3.1.9.2 Class BeginViewModel extends MmViewModel

Beschreibung

Diese Klasse verwaltet alle Funktionen, die die Klasse BeginActivity dem Benutzer bietet.

Methoden

- `void gotoLogin()`
  Diese Methode startet die LoginActivity Klasse, damit der Benutzer sich mit seinem Account einloggen kann.
- `void gotoRegister())`
  Diese Methode startet die RegisterActivity Klasse, damit der Benutzer sich mit seinen Daten registrieren und somit einen Account anlegen kann.

3.1.9.3 Class LoginViewModel extends MmViewModel

Beschreibung

Diese Klasse verwaltet alle Funktionen, die die Klasse LoginActivity dem Benutzer bietet.

Methoden

- `login(string email, string password)`
  Diese Methode leitet die Anmeldedaten an FireBase zur Überprüfung weiter und startet die Klasse HomeActivity, falls die Überprüfung erfolgreich war. Falls nicht wird eine Fehlermeldung in der LoginActivity erzeugt.

  ##### Parameter

  - email
    FireBase-E-Mail-Adresse.
  - password
    FireBase-Passwort.

- `passwordForgotten()`
  Diese Methode führt auf die FireBase-Seite bei Vergessen des Passworts.

- `gotoBegin()`
  Diese Methode startet die Klasse BeginActivity wieder.

3.1.9.4 Class RegisterViewModel extends MmViewModel

Beschreibung

Diese Klasse verwaltet alle Funktionen, die die Klasse RegisterActivity dem Benutzer bietet.

Methoden

- `register(string email, string password)`
  Diese Methode leitet Registrierungsdaten an FireBase zur Überprüfung weiter und erzeugt neue Profil, fallss diese erfolgreich war. Dann wird die Klasse UserActivity zur Profilvervollständigung gestartet. Falls nicht wird eine Fehlermeldung erzeugt.

  Parameter

  - email
    FireBase-E-Mail-Adresse.
  - password
    FireBase-Passwort.

3.1.9.5 UserViewModel extends MmViewModel

Beschreibung

Diese Klasse verwaltet alle Funktionen, die die Klasse UserActivity dem Benutzer bietet.

Konstruktoren

- `UserViewModel(LiveData<User> user, bool editable, MmActivity next)`
  Mit diesem Konstruktor lässt sich anzeigen, ob das Profil eine Bearbeitungsansicht sein soll. Außerdem lässt sich einstellen, welche Folgeseite aufgerufen wird.

  Parameter

  - LiveData<User> user
    Der bearbeitete Benutzer.

- bool editable
  Ob Bearbeitungsansicht aktiviert ist.
- MmActivity next
  Folgeseite.

Methoden

- `LiveData<User> loadUser()`
  Diese Methode gibt den Benutzer zurück.

  Rückgabewert

  Benutzer.

- `void saveUser(LiveData<User> user)`
  Diese Methode speichert die Benutzerdaten.

  Parameter

  - user
    Benutzer.

- `void gotoNext()`
  Diese Methode startet die Folgeseite.

3.1.9.6 Class HomeViewModel extends MmViewModel

Beschreibung

Diese Klasse verwaltet alle Funktionen, die die Klasse HomeActivity dem Benutzer bietet.

Methoden

- `LiveData<List<Line> > loadLines()`
  Diese Methode lädt eine Liste mit dem Essensangebot der Mensalinien.

  Rückgabewert

  Liste des Essensangebots der Mensalinien.

- `void saveLines(LiveData<List<Line> > chosenLines)`
  Diese Methode speichert die Linien, die der Benutzer als Favoriten angegeben hat und startet die Klasse SetTimeActivity.

Parameter

- chosenLines
  Liste der als Favoriten angegebenen Linien.

- `+void logout()`
  Diese Methode startet Klasse BeginActivity, wobei die MmSession-Klasse mit einer neuen ersetzt wird.

3.1.9.7 Class AdministerGroupsViewModel extends MmViewModel

Beschreibung

Diese Klasse verwaltet alle Funktionen, die die Klasse AdministerGroupsActivity dem Benutzer bietet.

Methoden

- `LiveData<List<Group> > loadGroups()`
  Diese Methode lädt eine Liste aller Gruppen.

  Rückgabewert

  Liste aller Gruppen.

- `deleteGroups(LiveData<List<Group> > groups)`
  Diese Methode löscht eine Menge von Gruppen.

  Parameter

  - groups
    Zu löschende Menge von Gruppen.

3.1.9.8 Class AdministerUsersViewModel extends MmViewModel

Beschreibung

Diese Klasse verwaltet alle Funktionen, die die Klasse AdministerUsersActivity dem Benutzer bietet.

Methoden

- `LiveData<List<User> > loadUsers()`
  Diese Methode lädt eine Liste aller Benutzer.

Rückgabewert

Liste aller Benutzer.

- `void deleteUsers(LiveData<List<User> > users)`
  Diese Methode löscht eine Menge von Benutzern.

  #### Parameter

  – users
  Zu löschende Menge von Benutzern.

3.1.9.9 Class SetTimeViewModel extends MmViewModel

Beschreibung

Diese Klasse verwaltet alle Funktionen, die die Klasse SetTimeActivity dem Benutzer bietet.

Methoden

- `LiveData<MmTime> loadTime()`
  Diese Methode lädt die bereits eingestellte Zeit/Zeitraum.

  #### Rückgabewert

  Bereits eingestellte Zeit/Zeitraum.

- `void saveTime(LiveData<MmTime> time)`
  Diese Methode speichert die eingegebene Zeit/Zeitraum zur weiteren Verwendung für die nachfolgenden Funktionen in der App ab.

  #### Parameter

  – time
  Zu speichernde Zeit/Zeitraum

3.1.9.10 Class SelectGroupViewModel extends MmViewModel

Beschreibung

Diese Klasse verwaltet alle Funktionen, die die Klasse SelectGroupActivity dem Benutzer bietet.

Methoden

- `LiveData<List<Group> > loadSuitableGroups()`
  Diese Methode lädt alle Gruppen, die zu den gespeicherten Kriterien in der MmSession-Klasse passen.

  ##### Rückgabewert

  Alle passenden Gruppen.

- `void gotoCreateGroup()`
  Diese Methode startet die Klasse CreateNewGroupActivity.

- `void gotoSetTime()`
  Diese Methode startet die Klasse SetTimeActivity.

3.1.9.11 Class CreateGroupViewModel extends MmViewModel

Beschreibung

Diese Klasse verwaltet alle Funktionen, die die Klasse CreateGroupActivity dem Benutzer bietet.

Methoden

- `void saveGroup(LiveData<Group> group)`
  Diese Methode speichert die Daten der neu angelegten Gruppe.

  ##### Parameter

  – group
  Neue Gruppenklasse.

- `void gotoSelectGroup()`
  Diese Methode startet die Klasse SelectGroupActivity.

3.1.9.12 Class MmListHandler

Beschreibung

Diese Klasse verwaltet alle Funktionen, die die Klasse MmList dem Benutzer bietet.

Methoden

- `void select(int itemId, bool selected)`
  Diese Methode setzt ein Element in der Liste auf (nicht) ausgewählt.

Parameter

- itemId
  Id des Elements der Liste.
- selected
  Ob das Element ausgewählt sein soll.

- `bool isSelected(int itemId)`
  Diese Methode gibt zurück, ob ein Element der Liste ausgewählt ist.

Parameter

- itemId
  Id des Elements der Liste

,

Rückgabewert

Ob das Element ausgewählt ist.

3.1.9.13 Class MmItemHandler

Beschreibung

Diese Klasse verwaltet alle Funktionen, die die Klasse MmItem dem Benutzer bietet.

3.1.9.14 Class GroupListHandler extends MmListHandler

Beschreibung

Diese Klasse verwaltet alle Funktionen, die die Klasse GroupList dem Benutzer bietet.

Konstruktoren

- `GroupListHandler(LiveData<List<Group> > groupList)`
  Dieser Konstruktor erzeugt aus einer Gruppenliste die Klasse.

Parameter

- groupList
  Gruppenliste.

3.1.9.15 Class GroupItemHandler extends MmItemHandler

Beschreibung

Diese Klasse verwaltet alle Funktionen, die die Klasse GroupItem dem Benutzer bietet.

Konstruktoren

- `GroupItemHandler(LiveData<Group> group, bool bigView)`
  Dieser Konstruktor erzeugt aus einer Gruppe die Klasse und gibt dabei an, ob die Anzeige im zugehörigen GroupItem groß oder klein ist, was Unterschiede in der Funktionalität zur Folge haben können soll.

  Parameter

  - group
    Gruppe.
  - bigView
    Ob die Anzeige im zugehörigen GroupItem groß ist.

Methoden

- `LiveData<Group> loadGroup`
  Lädt die gespeicherte Gruppe.

  Rückgabewert

  Gespeicherte Gruppe.

- `void joinGroup()`
  Diese Methode fügt den Benutzer in die Gruppe hinzu.

3.1.9.16 Class LineListHandler extends MmListHandler

Beschreibung

Diese Klasse verwaltet alle Funktionen, die die Klasse LineList dem Benutzer bietet.

Konstruktoren

- `LineListHandler(LiveData<MensaData> mensaData)`
  Dieser Konstruktor erzeugt aus den Mensadaten die Klasse.

  Parameter

  - mensaData
    Mensadaten.

3.1.9.17 Class LineItemHandler extends MmItemHandler

Beschreibung

Diese Klasse verwaltet alle Funktionen, die die Klasse LineItem dem Benutzer bietet.

Konstruktoren

- `LineItemHandler(LiveData<Line> line)`
  Dieser Konstruktor erzeugt aus den Daten einer Mensalinie die Klasse.

  Parameter

  - line
    Daten einer Mensalinie.

3.1.9.18 Class UserListHandler extends MmListHandler

Beschreibung

Diese Klasse verwaltet alle Funktionen, die die Klasse UserList dem Benutzer bietet.

Konstruktoren

- `UserListHandler(LiveData<List<User> > userList)`
  Dieser Konstruktor erzeugt aus einer Benutzerliste die Klasse.

  Parameter

  - userList
    Benutzerliste.

3.1.9.19 Class UserItemHandler extends MmItemHandler

Beschreibung

Diese Klasse verwaltet alle Funktionen, die die Klasse UserItem dem Benutzer bietet.

Konstruktoren

- `UserItemHandler(LiveData<User> user, MmViewMode viewMode)`
  Dieser Konstruktor erzeugt aus einem Benutzer und dem Anzeigemodus (groß nur lesen, groß bearbeitbar oder klein) die Klasse. Der Anzeigemodus soll Änderungen in der Funktionalität zur Folge haben können.

Parameter

- user
  Benutzer.
- viewMode
  Anzeigemodus (groß nur lesen, groß bearbeitbar oder klein).

Methoden

- `LiveData<User> loadUser()`
  Lädt den gespeicherten Benutzer.

Rückgabewert

Gespeicherter Benutzer.

3.1.9.20 Class UserPictureViewHandler

Beschreibung

Diese Klasse verwaltet alle Funktionen, die die Klasse UserPictureView dem Benutzer bietet.

Konstruktoren

- `UserPictureViewHandler(LiveData<User> user, int profilePictureId, MmViewMode viewMode)`
  Dieser Konstruktor erzeugt die Klasse aus dem zum Bild gehörigen Benutzer, der Id des Bildes und dem Anzeigemodus (groß nur lesen, groß bearbeitbar oder klein). Der Anzeigemodus soll Änderungen in der Funktionalität zur Folge haben können.

Parameter

- user
  Benutzer.
- profilePictureId
  Id des Bildes.
- viewMode
  Anzeigemodus (groß nur lesen, groß bearbeitbar oder klein).

Methoden

- `LiveData<List<MmProfilePicture> > loadAllProfilePicutres()`
  Lädt eine Liste aller möglichen Profilbilder, damit der Benutzer in der Klasse UserPictureView eins auswählen kann.

  #### Rückgabewert

  Liste aller möglichen Profilbilder.

- `LiveData<MmProfilePicture> loadProfilePicture()`
  Lädt das gespeicherte Profilbild.

  #### Rückgabewert

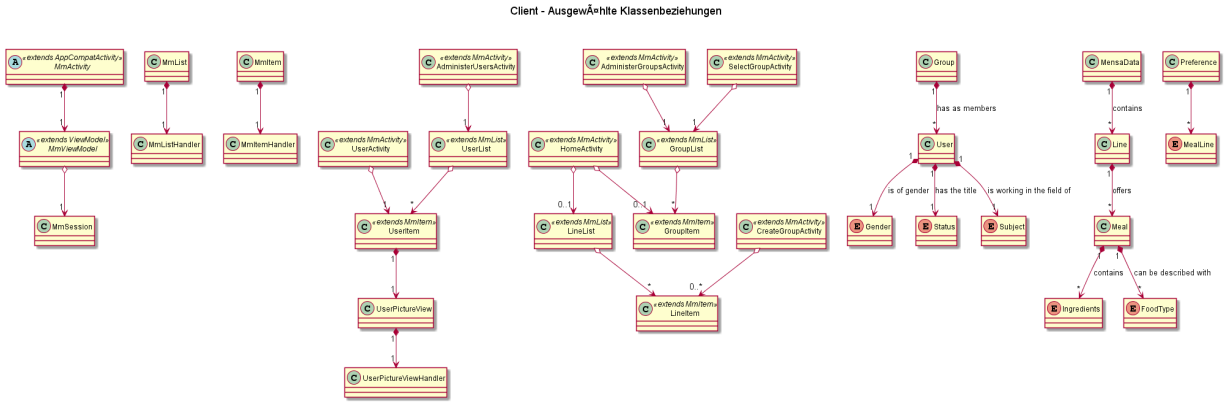
  Das gespeicherte Profilbild.

- `void savePicture(int profilePictureId)`
  Speichert ein ausgewähltes Profilbild.

  #### Parameter

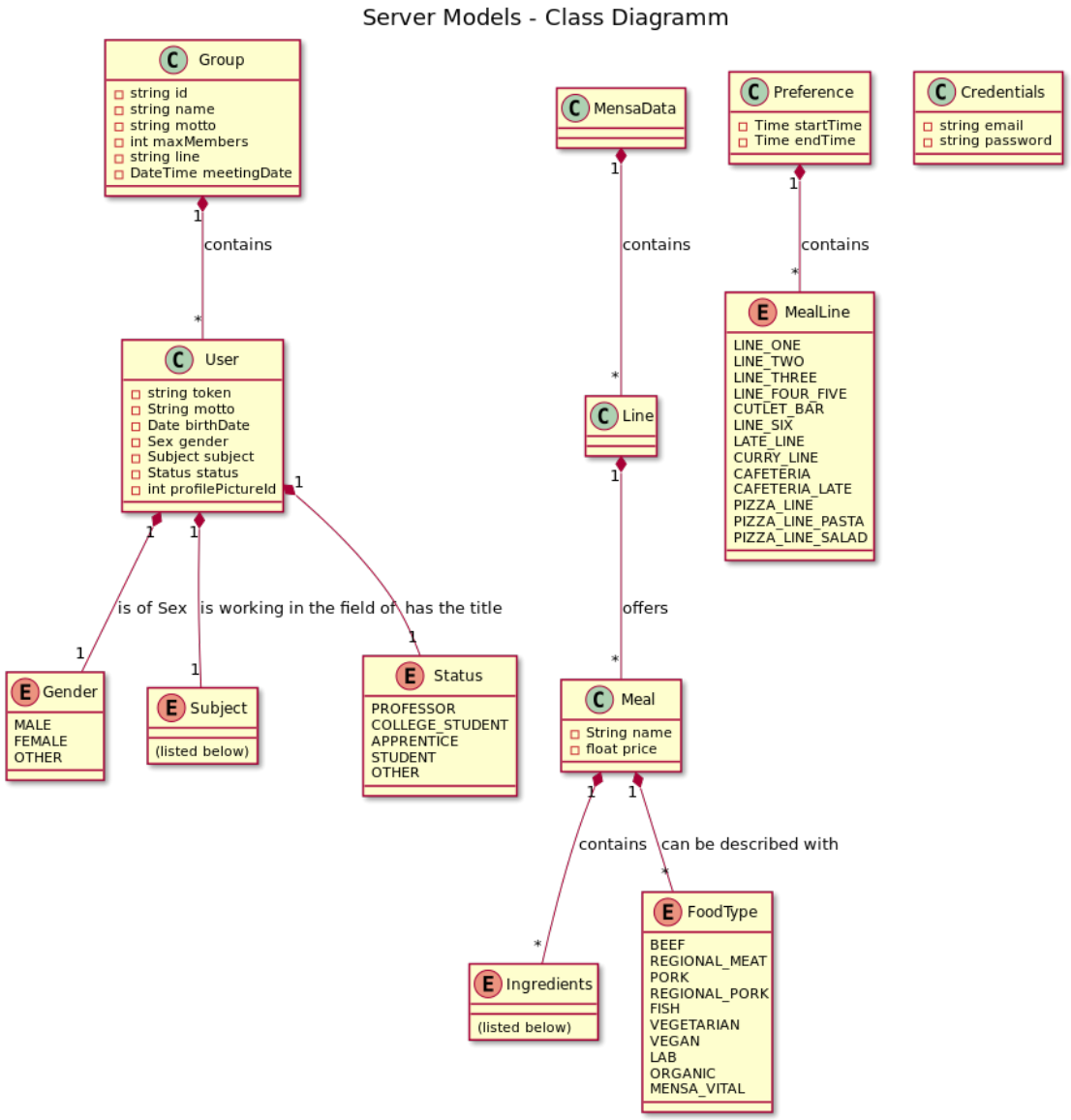
  – profilePictureId
  Id des ausgewählten Bildes.

3.1.10 Ausgewählte Klassenbeziehungen des Clients



3.2 Klassen des Server

3.2.1 Klassendiagramm



3.2.2 package.edu.kit.mensameet.server.model

3.2.2.1 Class Group

Die folgenden Klassen enthalten Standardkonstruktoren, die nicht explizit aufgeführt werden: User, Group.

Beschreibung

Dies ist eine Datenhalterklasse für eine Gruppe.

3.2.2.2 Class User

Beschreibung

Dies ist eine Datenhalterklasse für einen User.

3.2.2.3 Class Crendentials

Beschreibung

Diese Klasse beschreibt die Zugangsdaten (engl. Crendentials) eines Nutzers.

3.2.2.4 Class MensaData

Beschreibung

Diese Klasse beinhaltet den Speiseplan der Mensa des aktuellen Tages.

3.2.2.5 Class Line

Beschreibung

Diese Klasse beschreibt den Speiseplan der einzelnen Linien bzw. Werke an der Mensa.

3.2.2.6 Class Meal

Beschreibung

Diese Klasse beschreibt die einzelnen Speisen die an der Mensa angeboten werden. Dazu gehören die Zutaten (enum Ingredients) und der Typ (enum Foodtype).

3.2.2.7 Class Preferences

Beschreibung

In dieser Klasse sind die Preferenzen des Nutzers zur Gruppensuche.

3.2.2.8 Enum Gender

Beschreibung

Die Geschlechter, die zu einem User gehören können.
MALE - das männliche Geschlecht
FEMALE - das weibliche Geschlecht
OTHER - weder männlicht, noch weiblich

3.2.2.9 Enum Subject

Beschreibung

Die Fachrichtungen die ein User haben kann:

Angewandte Geowissenschaften
Architektur
Bauingenieurwese
Bioingenieurwesen
Biologie
Chemie
Chemieingenieurwesen und Verfahrenstechnik
Chemische Biologie
Deutsch
Elektro- und Informationstechnik
Energy Engineering and Management
Europäische Kultur und Ideengeschichte
Financial Engineering
Funktionaler und Konstruktiver Ingenieurbau-Engineering Structures
Geodäsie und Geoinformatik
Geographie
Geoökologie
Geophysik
Germanistik
Information Systems Engineering and Management
Informatik
Ingenieurpädagogik
Kunstgeschichte
Lebensmittelchemie
Management of Product Development
Maschinenbau
Materialwissenschaft und Werkstofftechnik
Mathematik
Mechanical Engineering

Mechatronik und Informationstechnik
Meteorologie
Mobilität und Infrastruktur
Mobility Systems Engineering and Management
Naturwissenschaft und Technik
Optics and Photonics
Pädagogik
Philosophie/Ethik
Physik
Production and Operations Management
Regionalwissenschaft
Remote Sensing and Geoinformatics
Sport
Sportwissenschaft
Technische Volkswirtschaftslehre
Technomathematik
Water Science and Engineering
Wirtschaftsinformatik
Wirtschaftsingenieurwesen
Wirtschaftsmathematik
Wissenschaft-Medien-Kommunikation

3.2.2.10 Enum Status

Beschreibung

Der Stauts den ein User haben kann:

Professor - Ein/e Professor/in oder Dozent/in
CollegeStudent - Ein/e Schüler-Student/in
Apprentice - Ein/e Auszubildende/r
Student - Ein/e reguläre/r Student/in
Other - Sonstiges

3.2.2.11 Enum FoodType

Beschreibung

Die Sorten von Essen die es gibt und die man Gerichten zuordnen kann:

BEEF
REGIONAL_MEAT
PORK
REGIONAL_PORK
FISH
VEGETARIAN
VEGAN
LAB
ORGANIC
MENSA_VITAL

3.2.2.12 Enum Ingredient

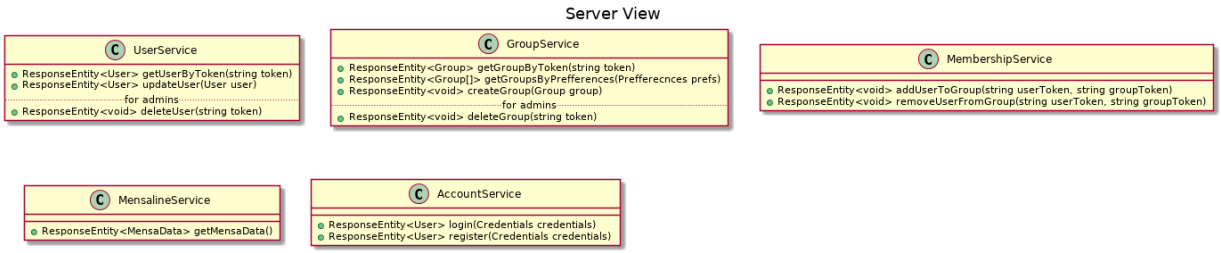
Beschreibung

Die Inhaltsstoffe, die in einem Gericht enthalten sein können:

(1) mit Farbstoff
(2) mit Konservierungsstoff
(3) mit Antioxidationsmittel
(4) mit Geschmacksverstärker
(5) mit Phosphat
(6) Oberfläche gewachst
(7) geschwefelt
(8) Oliven geschwärzt
(9) mit Süßungsmittel
(10) kann bei übermäßigem Verzehr abführend wirken
(11) enthält eine Phenylalaninquelle
(12) kann Restalkohol enthalten
(14) aus Fleischstücken zusammengefügt
(15) mit kakaohaltiger Fettglasur
(27) aus Fischstücken zusammengefügt
(R) enthält Rindfleisch
(RAT) enthält regionales Rindfleisch aus artgerechter Tierhaltung
(S) enthält Schweinefleisch
(SAT) enthält regionales Schweinefleisch aus artgerechter Tierhaltung
(VEG) vegetarisches Gericht
(VG) veganes Gericht (ohne Fleischzusatz)

(B) kontrolliert biologischer Anbau / DE007 Öko Kontrollstelle
(MSC) MSC-zertifizierter Fisch
(MV) MensaVital
(LAB) mit tierischem Lab
(GER) mit Gelatine
(Gl) Glutenhaltiges Getreide
(We) Weizen
(Ro) Roggen
(Ge) Gerste
(Ha) Hafer
(Di) Dinkel
(Ka) Kamut
(Nu) Schalenfrüchte / Nüsse
(Ma) Mandeln
(Ha) Haselnüsse)
(Wa) Walnüsse
(Ca) Cashewnüsse
(Pe) Pekanüsse
(Pa) Paranüsse
(Pi) Pistazie
(Qu) Queenslandnüsse/Macadamianüsse
(Ei) Eier
(Er) Erdnüsse
(So) Soja
(Sn) Senf
(Kr) Krebstiere
(Fi) Fisch
(ML) Milch / Laktose
(Se) Sellerie
(Sf) Schwefeldioxid / Sulfit
(Sa) Sesam
(Lu) Lupine
(We) Weichtiere

3.2.3 package.edu.kit.mensameet.server.view



Alle folgenden Klassen haben keine Konstruktoren, da sie keine Instanzen besitzen sollen,sondern lediglich statische Funktionen beinhalten, die die Schnittstelle des Servers bilden. Da alle Rückgabetypen vom Typ ResponseEntity sind werden auch alle http status codes die auftreten können beschrieben.

3.2.3.1 Class UserService

Beschreibung

Diese Klasse stellt die für den Client benötigten Funktionen bereit um User Daten abzufragen und zu bearbeiten.

Methoden

- `ResponseEntity<User> getUserByToken(string token)`
  Liefert den User mit dem übergebenen Token.

  ###### Parameter

  – token
  Der Token, mit dem der User eindeutig identifizierbar ist.

  ###### Rückgabewert

  Der gesuchte User als ResponseEntity mit status code 200. Status code 404, falls der User mit dem Token nicht existiert.

- `ResponseEntity<User> updateUser(User user)`
  Updated den User auf dem Server mit dem übereinstimmenden Token.

  ###### Parameter

  – user
  Der User der geupdated werden soll.

Rückgabewert

Der geupdatete User als ResponseEntity mit status code 200. Status code 404, falls der User mit dem Token nicht existiert.

- `ResponseEntity<void> deleteUser(string token)`
  Löscht den User auf dem Server mit dem übereinstimmenden Token.

Parameter

  - token
    Der Token des User, der gelöscht werden soll.

Rückgabewert

Status code 200, falls der User gefunden und gelöscht werden konnte. Status code 404, falls der User mit dem Token nicht existiert.

3.2.3.2 Class GroupService

Beschreibung

Diese Klasse stellt die für den Client benötigten Funktionen bereit um Gruppen Daten abzufragen und zu bearbeiten.

Methoden

- `ResponseEntity<Group> getGroupByToken(string token)`
  Liefert die Gruppe mit dem übergebenen Token.

Parameter

  - token
    Der Token, mit dem die Gruppe eindeutig identifizierbar ist.

Rückgabewert

Die gesuchte Gruppe als ResponseEntity mit status code 200. Status code 404, falls die Gruppe mit dem Token nicht existiert.

- `ResponseEntity<Group[]> getGroupsByPrefferences(Prefferecnces prefs)`
  Liefert alle Gruppen, die zu den übergebenen Präferenzen passen.

Parameter

- prefs
  Die Präferenzen, mit denen die Gruppen gesucht werden sollen.

Rückgabewert

Ein Array aller passenden Gruppen als ResponseEntity mit status code 200.

- `ResponseEntity<void> createGroup(Group group)`
  Erstellt eine Gruppe auf dem Server.

Parameter

- group
  Die Gruppe, die erstellt werden soll.

Rückgabewert

Die erstellte Gruppe mit einen generierten token als ResponseEntity, mit status code 200.

- `ResponseEntity<void> deleteGroup(string token)`
  Löscht die Gruppe auf dem Server mit dem übereinstimmenden Token.

Parameter

- token
  Der Token der Gruppe, die gelöscht werden soll.

Rückgabewert

Status code 200, falls die Gruppe gefunden und gelöscht werden konnte. Status code 404, falls die Gruppe mit dem Token nicht existiert.

3.2.3.3 Class MembershipService

Beschreibung

Diese Klasse stellt die für den Client benötigten Funktionen bereit um die Mitgliedschaft von Usern bei Gruppen zu bearbeiten.

Methoden

- `ResponseEntity<void> addUserToGroup(string userToken, string groupToken)`
  Fügt einen User einer Gruppe bei.

  ##### Parameter

  - userToken
    Der Token, mit der User eindeutig identifizierbar ist.
  - groupToken
    Der Token, mit dem die Gruppe eindeutig identifizierbar ist.

  ##### Rückgabewert

  Status code 404, falls der User oder die Gruppe nicht gefunden werden konnte.
  Status code 200, sonst.

- `ResponseEntity<void> removeUserFromGroup(string userToken, string groupToken)`
  Entfernt einen User von einer Gruppe.

  ##### Parameter

  - userToken
    Der Token, mit der User eindeutig identifizierbar ist.
  - groupToken
    Der Token, mit dem die Gruppe eindeutig identifizierbar ist.

  ##### Rückgabewert

  Status code 404, falls der User oder die Gruppe nicht gefunden werden konnte.
  Status code 200, sonst.

3.2.3.4 Class MensalineService

Beschreibung

Diese Klasse stellt die für den Client benötigten Funktionen bereit um die Daten der Mensalinien abzufragen.

Methoden

- `ResponseEntity<MensaData> getMensaData()`
  Liefert die aktuellen Daten der Mensalinien.

Rückgabewert

Die aktuellen Angebote der Mensa.

3.2.3.5 Class AccountService

Beschreibung

Diese Klasse stellt die für den Client benötigten Funktionen bereit um sich zu registrieren und anzumelden.

Methoden

- `ResponseEntity<User> login(Credentials credentials)`
  Loggt einen User ein.

  #### Parameter

  – credentials
  Die Email-Adresse und das Passwort des Users

  #### Rückgabewert

  Der User als ResponseEntity mit status code 200. Status code 404, die email nicht vorhanden ist. Status code 401, falls das Passwort falsch ist.

- `ResponseEntity<User> register(Credentials credentials)`
  Registriert einen neuen User und generiert ein Token.

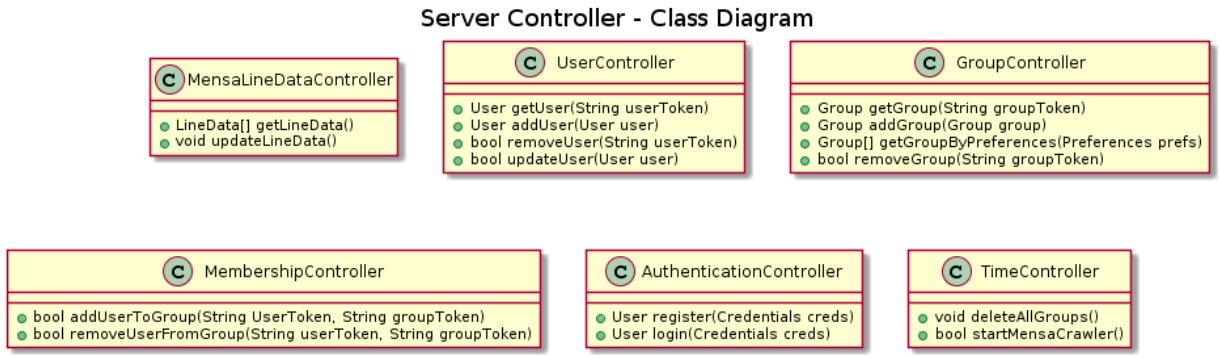
  #### Parameter

  – credentials
  Die Email-Adresse und das Passwort für den neuen User

  #### Rückgabewert

  Der neue User, mit dem Token als ResponseEntity mit status code 200. Status code 406, falls die Email-Adresse schon benutzt wird. Status code 400, falls die Form der Anfrage falsch ist. Bsw. die Email-Adresse eine falsche Form hat.

3.2.4 package.edu.kit.mensameet.server.controller



3.2.4.1 Class UserController

Beschreibung

Diese Klasse ist der User Controller und beinhaltet die für den User zuständigen Methoden

Methoden

- `public User getUser(String userToken)`
  Mit dieser Methode wird eine Anfrage nach einem Userprofil gestellt. Es wird nach dem eindeutigen User Token gesucht

  Parameter

  - userToken
    Eindeutiger Token eines Userprofils

  Rückgabewert

  - Userprofil

- `public User addUser(User user)`
  Diese Methode fügt einen User zur Datenbank hinzu

  Parameter

  - user
    Eindeutiges Userprofil

Rückgabewert

- Userprofil

- `public bool removeUser(String userToken)`
  Diese Methode löscht ein Userprofil aus der Datenbank

  Parameter

  - userToken
    Eindeutiger Token eines Userprofils

  Rückgabewert

  - Wird das Userprofil erfolgreich gelöscht, so wird eine Bestätigung zurückgegeben. Wenn der Löschvorgang fehlschlägt wird eine Fehlermeldung zurückgegeben

- `public bool updateUser(User user)`
  Diese Methode überschreibt ein bestehendes Userprofil mit einem Userprofil des selben Tokens, aber geänderten Parametern

  Parameter

  - user
    Eindeutiges Userprofil mit geänderten Parametern (bis auf Token)

  Rückgabewert

  - Wird das Userprofil erfolgreich überschrieben, so wird eine Bestätigung zurückgegeben. Wenn das Überschreiben fehlschlägt wird eine Fehlermeldung zurückgegeben

3.2.4.2 Class GroupController

Beschreibung

Diese Klasse ist der Group Controller und beinhaltet die für die Gruppen zuständigen Methoden

Methoden

- `public Group getGroup(String groupToken)`
  Mit dieser Methode wird eine Anfrage nach einer Gruppe gestellt. Es wird nach dem eindeutigen Group Token gesucht

Parameter

- groupToken
  Eindeutiger Gruppen Token

Rückgabewert

- Gruppe

- `public Group addGroup(Group group)`
  Diese Methode fügt eine Gruppe zur Datenbank hinzu

Parameter

- group
  Gruppe mit korrekten Parametern

Rückgabewert

- Die hinzugefügte Gruppe wird zurückgegeben

- `public Group[] getGroupByPreferences(Preferences prefs)`
  Diese Methode sucht nach Gruppen mit den passenden Präferenzen

Parameter

- prefs
  Nötige Präferenzen zur Findung einer passenden Gruppe (Linie und Uhrzeit)

Rückgabewert

- Es werden alle Gruppen mit übereinstimmenden Präferenzen zurückgegeben

- `public bool removeGroup(String groupToken)`
  Diese Methode löscht eine Gruppe aus der Datenbank

Parameter

- groupToken
  Eindeutiger Group Token der zu löschenden Gruppe

Rückgabewert

- Wird die Gruppe erfolgreich gelöscht, so wird eine Bestätigung zurückgegeben. Wenn der Löschvorgang fehlschlägt wird eine Fehlermeldung zurückgegeben

3.2.4.3 Class MensaDataController

Beschreibung

Diese Klasse ist der Mensadaten Controller und enthält die Methoden, die für die Verwaltung der Mensadaten zuständig sind

Methoden

- `public LineData[] getLineData()`
  Diese Methode fragt die Datenbank nach dem tagesaktuellen Speiseplan ab

  #### Rückgabewert

  - Es werden alle Essenslinien mit tagesaktuellem Menü zurückgegeben

- `public void updateLineData()`
  Diese Methode beinhaltet einen Crawler der den tagesaktuellen Speiseplan der Mensa vom Studentenwerk bezieht und anschließend in der Datenbank speichert

3.2.4.4 Class MembershipController

Beschreibung

Diese Klasse ist der Membership Controller und beinhaltet die Methoden, die für die Gruppenmitgliedschaft der einzelnen User zuständig sind

Methoden

- `public bool addUserToGroup(String UserToken, String groupToken)`
  Diese Methode fügt einen User zu einer Gruppe hinzu

  #### Parameter

  - userToken
    Eindeutiger Token des Users der zur Gruppe hinzugefügt werden soll
  - groupToken
    Eindeutiger Token der Gruppe, zu der der User hinzugefügt werden soll

Rückgabewert

- Falls das Hinzufügen zur Gruppe erfolgreich war, wird eine Bestätigung zurückgegeben. Eine Fehlermeldung falls nicht.

- `public bool removeUserFromGroup(String userToken, String groupToken)`
  Diese Methode entfernt einen User aus einer Gruppe

  #### Parameter

  - userToken
    Eindeutiger Token des Users, der aus der Gruppe entfernt werden soll
  - groupToken
    Eindeutiger Token der Gruppe, aus der der User entfernt werden soll

  #### Rückgabewert

  - Falls das Entfernen aus der Gruppe erfolgreich war, wird eine Bestätigung zurückgegeben. Eine Fehlermeldung falls nicht.

3.2.4.5 Class AuthenticationController

Beschreibung

Diese Klasse ist der Authentication Controller und beinhaltet die Methoden, die für die Registrierung und das Anmelden eines Users zuständig sind

Methoden

- `public User register(Credentials creds)`
  Diese Methode ist für die Registrierung eines Users zuständig

  #### Parameter

  - creds
    Die vom User festgelegten Daten zur Registrierung, bestehend aus E-Mail und Passwort

  #### Rückgabewert

  - Es wird ein leeres Userprofil zurückgegeben

- `public User login(Credentials creds)`
  Diese Methode ist für die Ameldung eines Users zuständig

Parameter

- creds
  Die vom User bei der Registrierung festgelegten Daten, bestehend aus E-Mail und Passwort

Rückgabewert

- Es wird das UserProfil des Users zurückgegeben

3.2.4.6 Class TimeController

Beschreibung

Diese Klasse ist der Time Controller und beinhaltet die Methoden, die für die zeitlich festgelegte Abläufe zuständig ist

Methoden

- `public void deleteAllGroups()`
  Diese Methode löscht am Ende des Tages alle Gruppen
- `public bool startMensaCrawler()`
  Diese Methode startet jeden Morgen den Aufruf zum Abrufen der Mensadaten

Rückgabewert

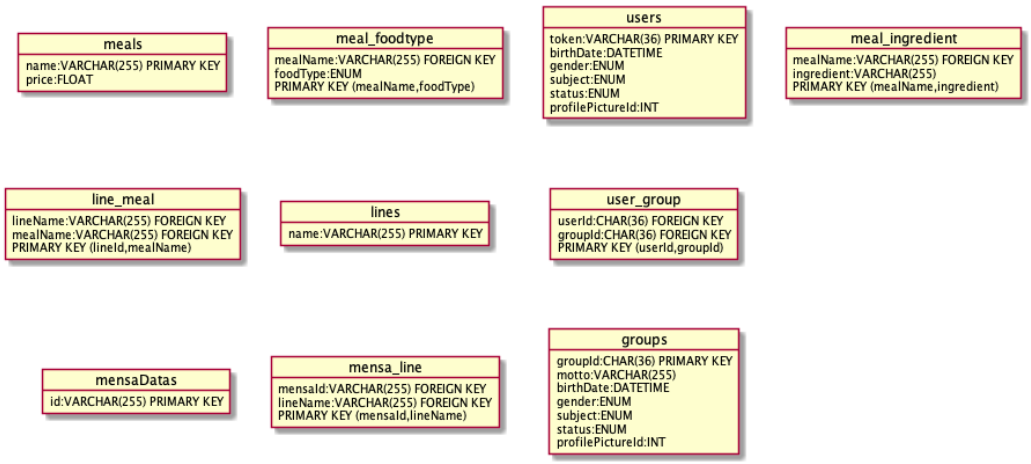
- Wurden die Mensadaten erfolgreich abgegriffen, wird eine Bestätigung zurückgegeben. Eine Fehlermeldung falls nicht.

3.3 HTTP Protokoll

Die REST Endpoints die in der Tabelle aufgelistet sind werden für die Kommunikation zwischen Client und Server verwendet.

Request Type	Endpoint	Payload	Beschreibung
GET	/user/{token}		Liefert den User mit dem übergebenem Token.
POST	/user	User	Updated den User (Token liegt in User)
GET	/group/token		Liefert die Gruppe mit dem übergebenem Token.
POST	/group-prefferences	Preferrences	Liefert alle Gruppen die zu den übergebenen Prefferenzen passen.
POST	/create-group/	Group	Erstellt die übergebene Gruppe.
DELETE	/group/{token}		Löscht die Gruppe mit dem übergebenen Token.
POST	/add-user-to-group?user={userToken}&group=groupToken		Fügt den User mit dem userToken zu der Gruppe mit dem groupToken.
POST	/remove-user-from-group?user={userToken}&group=groupToken		Entfernt den User mit dem userToken von der Gruppe mit dem groupToken.
POST	/login	email, password	Meldet den Client an dem Account mit den übergebenen Anmeldedaten an.
POST	/register	email, password	Erstellt einen Account mit den übergebenen Anmeldedaten.
GET	/mensadata		Liefert die aktuellen Mensadaten.

4 Datenstrukturen



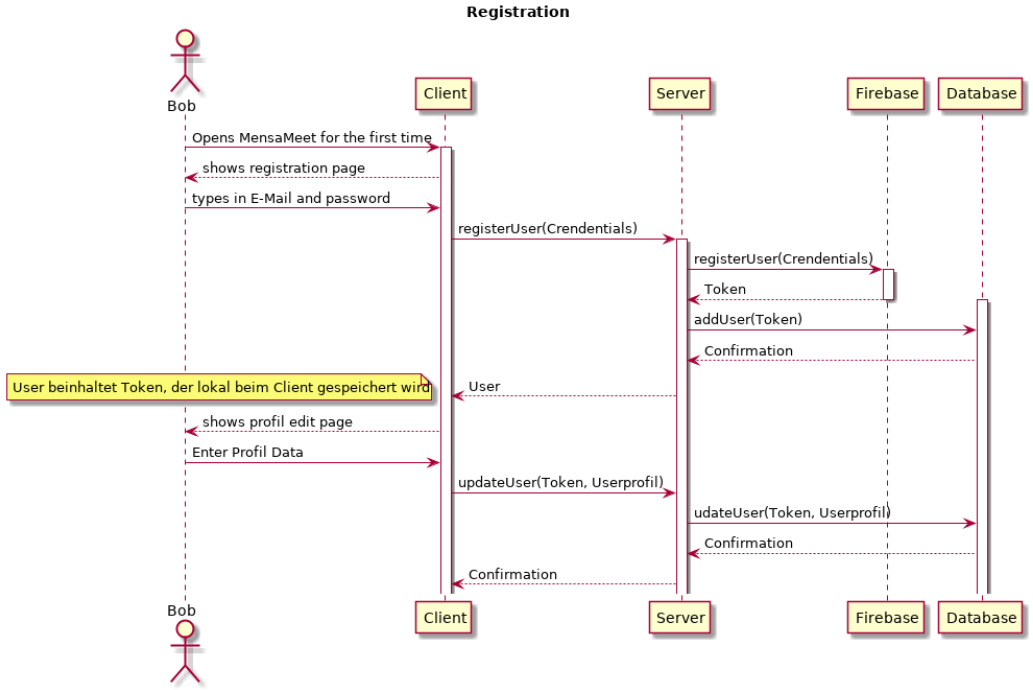
In der Abbildung sieht man die Datenstruktur der Datenbank, also die Tabellen, die es in der Datenbank geben wird. Weil wir mySQL verwenden und mySQL nicht List-Type als Datentype unterstützt, speichern wir Liste in Form von extra Tabellen.

5 Dynamische Modelle

5.1 Sequenzdiagramme

Es werden nun einige Abläufe innerhalb der App MensaMeet durch Sequenzdiagramme veranschaulicht.

5.1.1 Registrieren

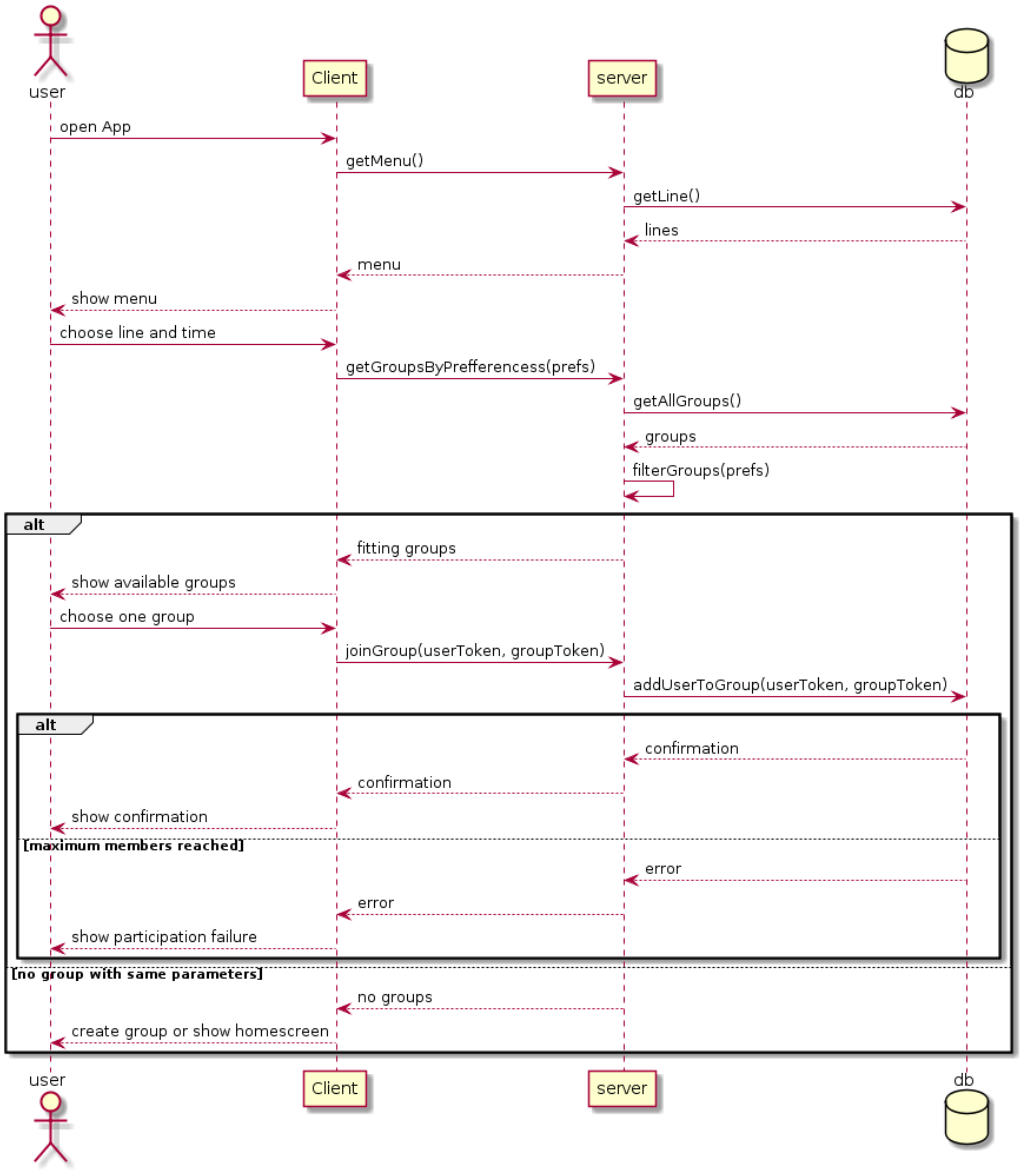


Beschreibung

Bob öffnet zum ersten mal die Applikation MensaMeet und sieht den Registrierungsbildschrim. Dort soll er seine E-Mailadresse und ein Passwort eingeben. Sobald er das getan hat, werden die Daten an der Server geschickt, welcher sie an Firebase weiterleitet. Firebase legt auf seinem Server den User an und generiert ein Token zur UserIdentifikation. Das Token wird an den Server zurückgegeben, der damit einen User in der Datenbank anlegt und das Userobjekt an den Client schickt. Auf dem Client (Bobs Smartphone) wird der User/Token gespeichert und bei jeder zukünftigen Anfrage an den Server mitgegeben. Bob sieht nun den Bildschirm zur Profilbearbeitung. Die Eingabe der Profildaten

ist Pflicht, vorher kann er nicht weiter zu einer anderen Seite gelangen. Nachdem er seinen Nutzernamen, Motto, Alter, Geschlecht, Status, Fachrichtung eingegeben hat, werden diese Daten an den Server gesendet, der damit das Userprofil in der Datenbank aktualisiert.

5.1.2 Mensalinien wählen

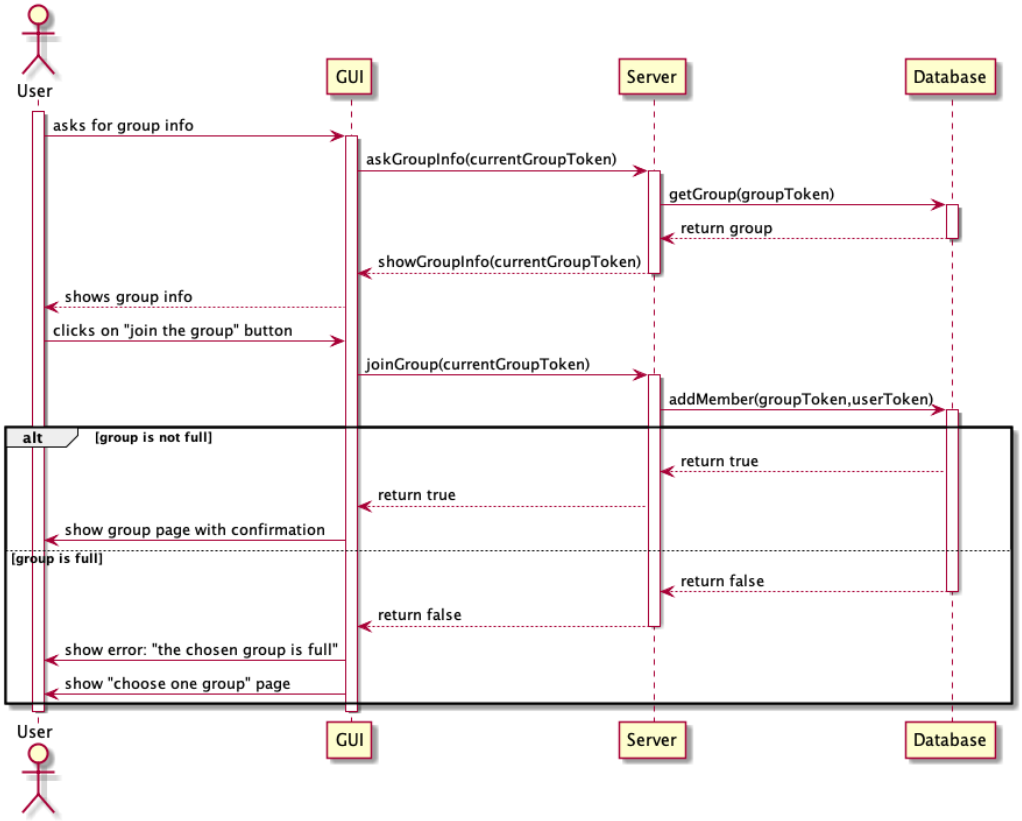


Beschreibung

Der Nutzer öffnet die Anwendung (die Registrierung ist bereits erfolgt). Nun soll er als Scrolldown-menü die Mensalinien mit ihrem jeweiligen Angebot zu sehen kriegen (Dies ist der ”HomeScreen”). Dazu werden die Mensadaten über den Server von der Datenbank angefragt, aufbereitet und dem Client übermittelt. Dieser kann nun die Linien anklicken an denen er gern essen möchte. Danach stellt er das für ihn geeignete Zeitintervall ein. Diese Daten werden dem Server übermittelt, welcher sich die Gruppen aus der Datenbank holt, nach den angegebenen Daten filtert und an den Client übergibt. Der Nutzer

kann sich die Gruppen anschauen und einer davon beitreten. Beim Versuch beizutreten wird dem Server der User- und der GroupToken übermittelt. Es wird überprüft ob die Gruppe bereits voll ist, ist dies der Fall, erhält der Client eine Fehlermeldung und gelangt wieder zur Gruppenübersicht. Ist noch Platz in der Gruppe, wird die Gruppe in der Datenbank aktualisiert, indem der User hinzugefügt wird. Dann wird über den Server eine Bestätigung an den Client geschickt und der Nutzer gelangt zur Gruppenseite. Falls keine Gruppen passend zu den Angaben des Clients gefunden wurde, erhält der Client eine passende Fehlermeldung und kann entweder zurück zum HomeScreen oder selbst eine Gruppe erstellen.

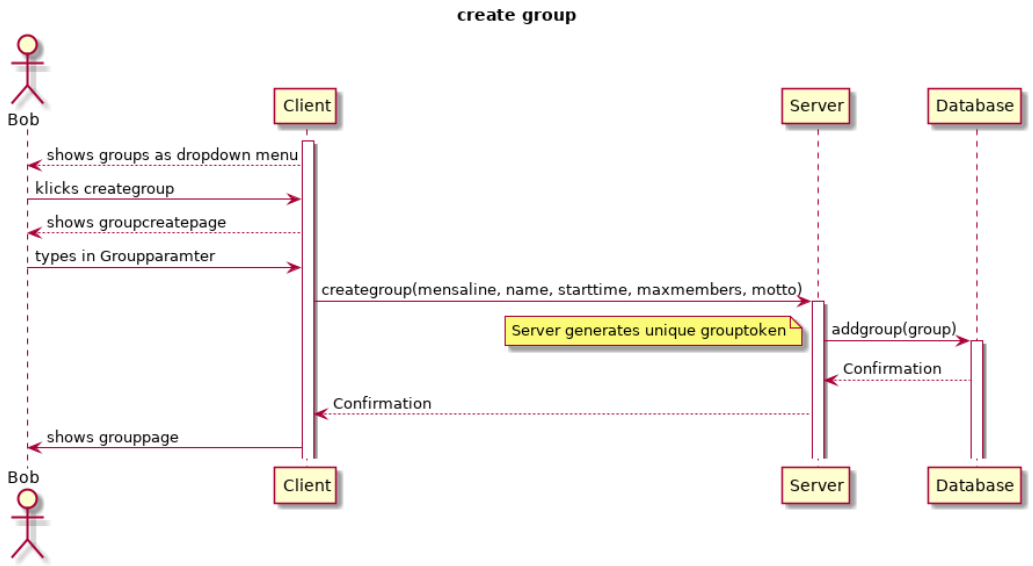
5.1.3 Gruppe beitreten



Beschreibung

Nachdem der Nutzer, passend zu seinen ausgewählten Linien und seinem eingestellten Zeitintervall, Gruppen in einem Scrolldown-menü angezeigt bekam, klickt er nun eine Gruppe an, um mehr Informationen zu ihr zu sehen. Dazu wird an den Server eine Anfrage mit dem GroupToken gesendet. Der Server sucht damit nach der Gruppe in der Datenbank und übermittelt die Informationen dem Client. Beim User wird die angeklickte Gruppe nach unten aufgklappt, so dass nun zusätzlich zur Linie, Startzeit und Motto auch die Mitgliederliste und den ”BeitretenButton sieht. Er drückt auf ”Beitreten”. Nun wird dem Server die UserID und GroupID übermittelt. Er prüft in der Datenbank ob die Gruppe inzwischen voll ist. Ist dies der Fall, erhält der Client eine Fehlermeldung : ”Deine gewählte Gruppe ist bereits vollünd gelangt wieder zur Gruppenübersicht. Ist noch Platz in der Gruppe, wird die Datenbank aktualisiert und dem Client der Beitritt bestätigt. Der Nutzer gelangt zur Detailansicht seiner Gruppe. Er steht nun ebenfalls in der Mitgliederliste und sieht den Button ”Gruppe Verlassen”.

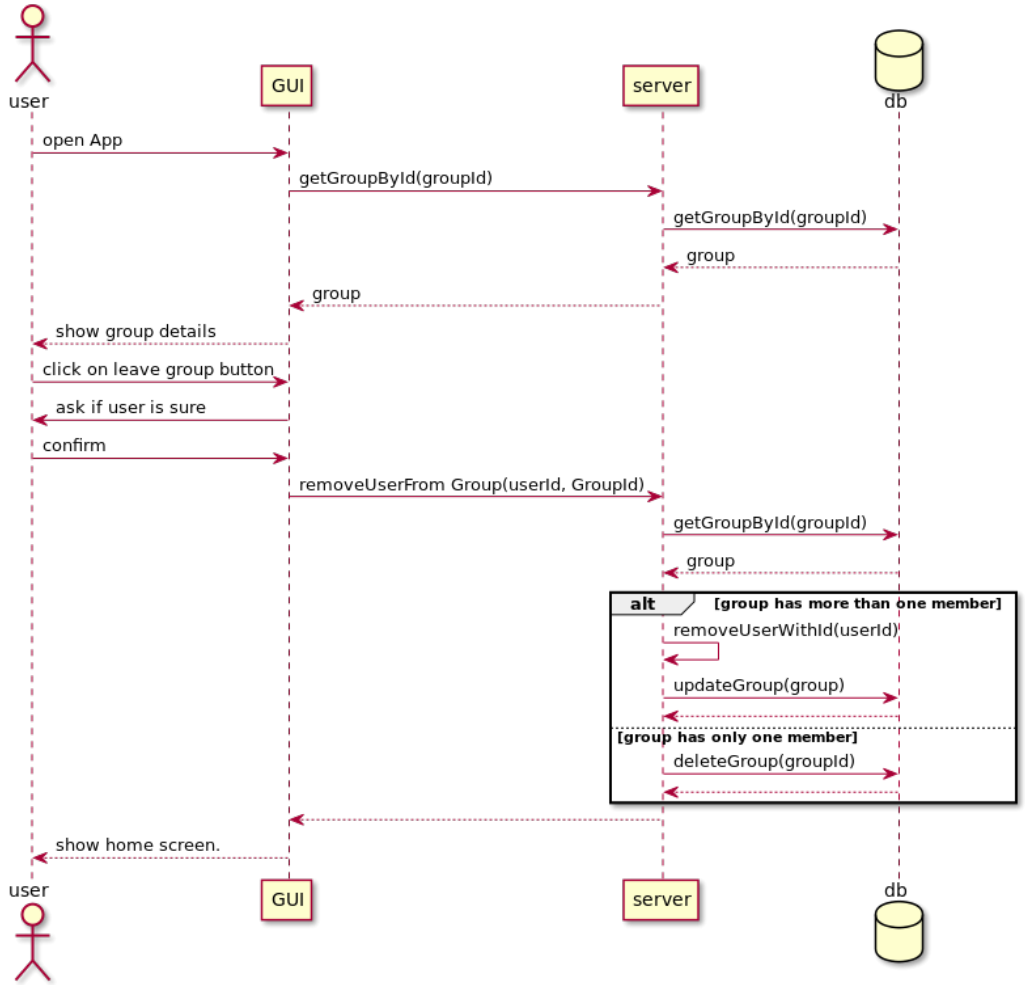
5.1.4 Gruppe erstellen



Beschreibung

Nachdem der User die zu ihm passenden Gruppen angefragt hat, aber keine gefunden wurde entscheidet er sich eine Gruppe zu erstellen. Er stellt dazu die erforderlichen Gruppenparameter(Gruppenname, Mensalinie, Startzeit, Motto, Maximale Mitgliederzahl) ein. Der Server erhält die Anfrage mit diesen Paramtern eine neue Gruppe anzulegen. Er generiert (aus dem Namen?) ein eindeutiges groupToken und legt die Gruppe in der Datenbank an. Dem Client wird die erfolgreiche Gruppengründung bestätigt und ihm wird als nächstes die Detailansicht seiner Gruppe angezeigt.

5.1.5 Gruppe verlassen



Beschreiung

Der Nutzer öffnet die App MensaMeet. Da er bereits Mitglied einer Gruppe ist, wird ihm die Detailansicht seiner Gruppe angezeigt. In dieser Ansicht ist ein Button ”Gruppe verlassen”. Er drückt diesen Button und muss seine Entscheidung bestätigen. Für den Austritt wird dem Server der User- und GroupToken übermittelt. Der Server sucht anhand des GroupToken die Gruppe in der Datenbank um sie zu aktualisieren. Der User wird aus der Gruppe entfernt und die Mitgliederzahl der Gruppe überprüft. Ist diese nun bei 0, so wird die Gruppe aus der Datenbank gelöscht. Der Client erhält in jedem Fall die Bestätigung zum Austritt aus der Gruppe und wird wieder zum HomeScreen überführt.

6 Änderungen zum Pflichtenheft

- Nutzername und Gruppennamen müssen nicht mehr eindeutig sein
  stattdessen wird für den Nutzer von Firebase ein eindeutiges Token generiert, das als Identifikator dient. Für Gruppen wird ebenfalls ein eindeutiges Token durch den Server generiert.

- Zum Registrieren wird statt Nutzername und Password nun Email und Passwort benötigt.
  Diese Änderung ist bedingt durch die Nutzung von Firebase und erfüllt damit eines unserer Wunschkriterien: Dass eine Email Verifikation beim Registrieren stattfindet.

Glossar

Anwendungslogik Die Anwendungslogik einer Anwendung beschreibt die konkrete Verknüpfung von Bausteinen zu dieser Anwendung. 7

Applikation Software für einen Benutzer, die mindestens eine Funktion erfüllt. 55

Beobachter Der Beobachter ist ein Entwurfsmuster und dient der Weitergabe von Änderungen an einem Objekt an von diesem Objekt abhängige Strukturen. 7

Client Computerpogramm, das auf einem Endgerät ausgeführt wird und mit einem Server kommuniziert. 8

Datenbankmanagementsystem Ist ein Bestandteil der Datenbank und übernimmt die Aufgabe der Organisation und Strukturierung der Daten. 8

Dependency Injection Als Dependency Injection wird in der objektorientierten Programmierung ein Entwurfsmuster bezeichnet, welches die Abhängigkeiten eines Objekts zur Laufzeit reglementiert: Benötigt ein Objekt beispielsweise bei seiner Initialisierung ein anderes Objekt, ist diese Abhängigkeit an einem zentralen Ort hinterlegt – es wird also nicht vom initialisierten Objekt selbst erzeugt. 8

Endgerät Internetfähige Computerhardware, wie Smartphones, Tablets, Laptops usw.. 6

Framework Modernes Rahmenwerk, das dem Programmierer den Entwicklungsrahmen für seine Anwendungsprogrammierung zur Verfügung stellt und damit die Software-Architektur der Anwendungsprogramme bestimmt. 8

GUI Graphical User Interface, beschreibt die grafische Benutzeroberfläche von Computersystemen, um die Bedienung zu erleichtern. 7

Homescreen Hauptbildschirm der beim Öffnen einer Applikation erscheint. 19

Mensadaten Zu den Mensadaten gehören die Essenslinien und Essenswerke und deren Speisepläne. 6, 11

MySQL Eines der weltweit verbreitetsten relationalen Datenbankverwaltungssysteme. 8

POJO Plain Old Java Object, einfache Java-Klasse ohne komplexe Strukturen. 8

relationale Datenbank Sammlung von Tabellen(Relationen), in welchen Datensätze abgespeichert sind. 8

Server Ein Programm, das auf die Kontaktaufnahme eines Clients wartet, um eine bestimmte Dienstleistung für ihn zu erfüllen.. 8

Systemarchitektur Beschreibt die innere Struktur eines Softwaresystems. 5

Systemkomponente Einzelne Bestandteile eines Systems. 5

Token Komponente die zur Identifizierung und Authentifizierung von Benutzern und Gruppen dient. 62

7 Anhang

7.1 Klassendiagramm Client-Server