Dokumentation Teachingbee

Software-Praktikum Gruppe 5

Prof. Thies und Prof. Kunz

Namen

Contents

[1 Anforderungsspezifikation 2](#_Toc75772531)

[1.1 Zielsystem 2](#_Toc75772532)

[2 Vorgehen und Meilensteine 3](#_Toc75772533)

[3 Use-Case-Diagram 3](#_Toc75772534)

[Klassendiagramm 4](#_Toc75772535)

[Person 4](#_Toc75772536)

[Profile 5](#_Toc75772537)

[Group 5](#_Toc75772538)

[Message 5](#_Toc75772539)

[GroupMessage 5](#_Toc75772540)

[ER-Diagramm 6](#_Toc75772541)

[Installation der fertigen Anwendung 6](#_Toc75772542)

[Vorraussetzungen 6](#_Toc75772543)

[Lokal 6](#_Toc75772544)

[Cloud 6](#_Toc75772545)

[Nutzerhandbuch 6](#_Toc75772546)

[Aufbau der Anwendung 7](#_Toc75772547)

[Nutzung der Anwendung 7](#_Toc75772548)

# 1 Anforderungsspezifikation

## 1.1 Zielsystem

Die zu erstellende Anwendung soll Studenten die Möglichkeit bieten, Lernpartner zu finden. Dazu muss jede Person einen Account anlegen und ihre Lernvorlieben angeben. Über diese Lernvorlieben kann ein entsprechender Algorithmus dann passende Lernpartner vorschlagen. Diese Vorschläge umfassen sowhol Personen als auch Lerngruppen. Diesen Vorschlägen können dann Kontaktanfragen gesendet werden. Werden diese akzeptiert, können die Nutzer daraufhin über einen Chat miteinander kommunizieren. Anfragen, die an Gruppen gestellt werden, können von jedem Gruppenmitglied angenommen oder abgelehnt werden. Weiterhin können die Accountdaten bearbeitet und der gesamte Account gelöscht werden.

Die Authentifizierung erfolgt über Google Firebase und die Anwendung soll über Google Cloud gehostet werden. Die App verfügt über eine Clientseite, geschrieben mit dem Framework React und eine Serverseite, die das Python Webframework Flask nutzt. Client und Server kommunizieren über eine REST-API. Die Daten werden in einer MySQL-Datenbank gespeichert und von dort aus abgerufen.

# 2 Vorgehen und Meilensteine

Im Folgenden werden die Meilensteine kurz erläutert.

Für den ersten Meilenstein galt es, sich in einer Gruppe aus 5-6 Personen zu finden und diese Zusammenstellung anzugeben. Die vorliegende Anwendung wurde von sechs Personen erstellt, deren Namen auf dem Deckblatt genannt sind.

Der zweite Meilenstein umfasste die Erstellung des Use-Case-Diagramms, das die verschiedenen Möglichkeiten abbildet, wie die Anwendung genutzt werden kann. Das aktuelle und angepasste Use-Case-Diagramm befindet sich unter 3.

Im Rahmen des dritten Meilensteins wurde dann das Klassen- und das Entiny-Relationship-Diagramm erstellt. Diese repäsentieren die verwendeten Klassen der Anwendung und den Aufbau der Datenbank. Im laufe der Entwicklung wurden beide Diagramme mehrmals angepasst und überarbeitet. Die akuellen Versionen sind unter 4., beziehungsweise 5. Abgebildet.

Der vierte Meilensteil besteht aus der Abgabe des Peer-Feedbacks, das dazu dient, die eigenen Gruppenmitglieder hinsichtlich Mitwirkung um Team einzuschätzen. Es dient vor allem dazu, den Gruppenmitgliedern Rückmeldung zu ihrer Performance zu geben und Probleme frühzeitig zu erkennen.

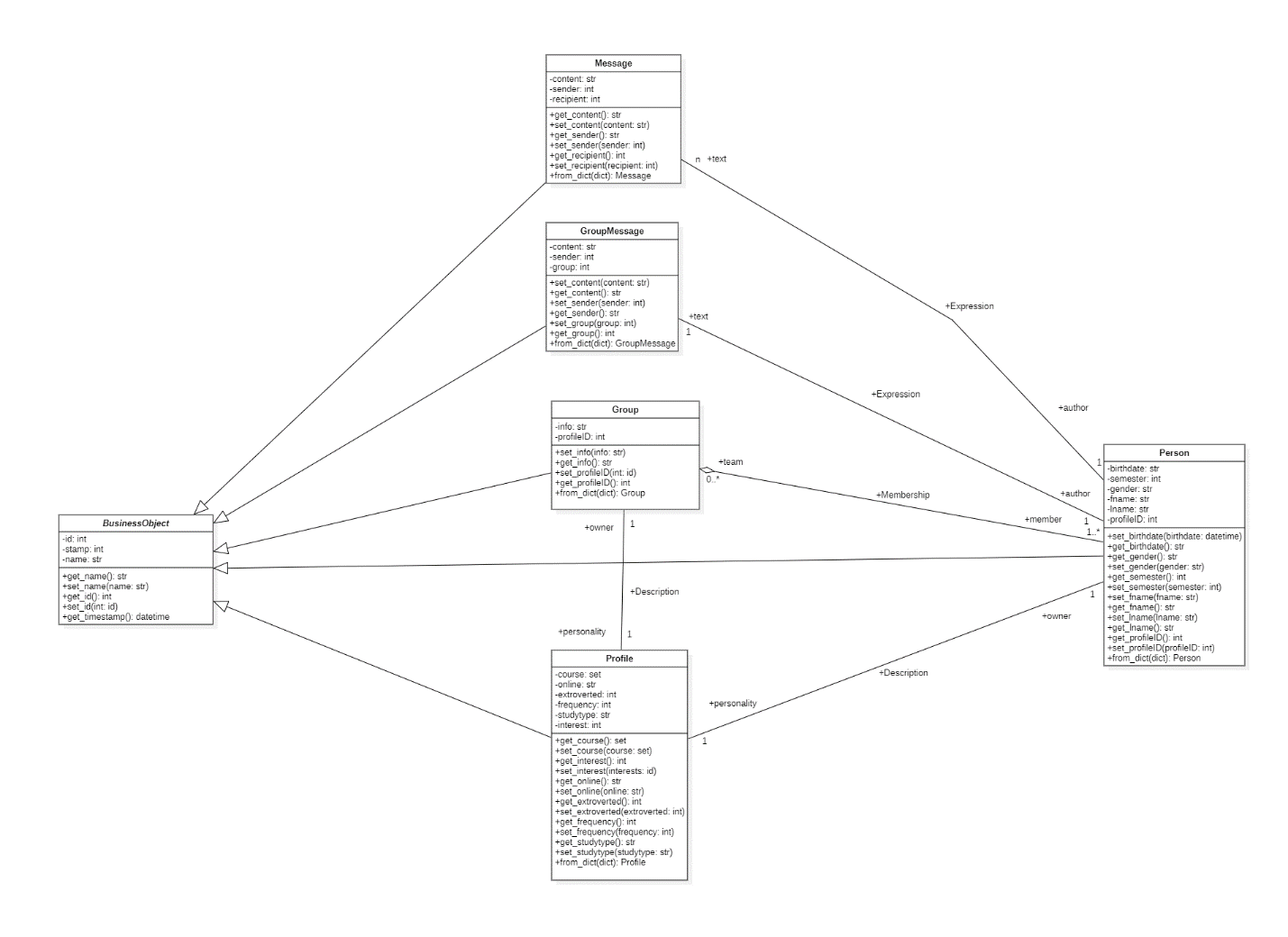
Der fünfte Meilenstein ist die Abgabe des vollständigen Source-Code der Anwendung, sowie diese Dokumentation. Der Source-Code soll mit Kommentaren entsprechend umfangreich dokumentiert sein.

Die beiden letzten Meilensteine umfassen die Präsentation fertigen Zielsystems in der Cloud und das individuelle Fachgespräch. Diese werden hier in der Dokumentation nicht behandelt.

# 3 Use-Case-Diagram

Noch anpassen

# 4 Klassendiagramm

Eine hochauflösende Version des Bildes befindet sich unter <https://github.com/sihensel/teachingbee/blob/main/docs/ClassDiagram.png>

Das Klassendiagramm besteht aus der Basisklasse BusinessObject mit den Attributen id, timestamp und name, sowie den fünf weiteren Klassen Person, Profile, Group, Message und Groupmessage, die alle von BusinessObject erben. Die einzelnen Klassen werden im folgenen kurz erläutert. Alle Klassen besitzen Getter- und Setter-Methoden, mit denen die Attribute der Klasse ausgelesen, beziehungsweise neu gesetzt werden können. Weiterhin verfügt jede Klasse über die Methode from\_dict, mit der aus einem JSON-Object ein Klassenobjekt erstellt werden kann. Da die Methoden aller Klassen gleicher Natur sind, wird im Folgenden nicht mehr näher auf die Mehtoden eingegangen.

### Person

Diese Klasse beschreibt alle Personenobjekte und ist dient der Erfassung von personenbezogenen Daten der Nutzer. Sie besteht aus den folgenden Attributen:

* fname (Vorname)
* lname (Nachname)
* birthdate (Geburtsdatum)
* semester (aktuelles Semester)
* gender (Geschlecht)
* profileID(verknüpft das Personenobjekt eines Nutzers mit dessen Profil

### Profile

Profile repräsentiert das Lernprofil eines Nutzers oder einer Gruppe und umfasst die Attribute:

* course (Studiengang)
* online (ob man online oder offline lernen möchte)
* extroverted (die Extrovertierheit einer Person auf einer Skala von 1-3)
* frequency (die gewünschte Lernhäufigkeit auf einer Skala von 1-3)
* studytype (der Lerntyp, visuell, kommunikativ, motorisch oder auditiv)
* interest (eine der Interessen aus der Datenbanktabelle Interests (vgl. X.Y)

Die Lernprofile werden für den Matching-Algorithmus verwendet, indem die Profile miteinander verglichen werden um deren Ähnlichkeit festzustellen. Ähnliche Profile werden gematcht und dem Nutzer als Vorschlag angezeigt.

### Group

Gruppen repräsentieren eine Menge von Personen, die ein bestimmtes Thema lernen wollen. Gruppenobjekte haben nur zwei Attribute:

* info (kurze Beschreibung der Gruppe)
* profileID (das Profil einer Gruppe)

Gruppen können von jedem Nutzer erstellt werden und werden dann für das Matching berücksichtigt. Die Zuorfdnung von Personen zu einer Gruppe erfolgt úber die Tabelle R\_person\_group (vgl. X.Y).

### Message

Chatnachrichten sind logisch getrennt in Nachrichten zwischen Personen und Nachrichten innerhalb von Gruppen. Nachrichten zwischen Personen haben die Attribute

* content (textueller Inhalt der Nachricht)
* sender (Absender)
* recipient (Empfänder),

wobei sender und recipient jeweils Personen sind.

### GroupMessage

Gruppennachrichten sind vom Aufbau deckungsgleich zu Personennachrichten, mit dem einzigen Unterschied, dass der Empfänger einer solchen Nachricht immer ein Gruppenobjekt ist. Daher wurde das Attribut recipient durch group ersetzt.

# 5 ER-Diagramm

Eine hochauflösende Version des Bildes befindet sich unter <https://github.com/sihensel/teachingbee/blob/main/docs/ER-diagram.png>

Das ER-Diagramm gibt im wesentlichen die Struktur des Klassendiagramms wieder, da jede Klasse durch eine Tabelle repräsentiert wird. Darüberhinaus kommen noch folgende Tabellen hinzu:

* Interests (beinhaltet alle zur Auswahl stehdenden Interessen, wird mit Profile verknüpft)
* Request (für Kontaktanfragen zwischen Personen)
* Grouprequest (Kontaktanfragen von Personen an Gruppen)

Diese Tabellen haben keine entsprechenden Klassen.

Jede Tabelle besitzt einen Primärschlüssel, der von der Datenbank automatisch vergeben und hochgezählt wird und einen Timestamp, der den Erstellungszeitpunkt des Datensatzes wiedergibt. Innerhalb der Datenbank wurden alle n:m-Beziehungen aufgelöst und alle Relationen von Fremdschlüssel-Beziehungen repräsentiert.

Die Tabellennamen starten mit einem Großbuchstaben, damit sie leichter von Tabellenfeldern zu unterscheiden sind. Einige Tabellen und Felder haben spezielle Namen um Konflikte mit reservierten Schlüsselbegriffen seitens MySQL zu vermeiden. SO heißt der Timestamp nur ‘stamp’ oder die Tabelle zu Lerngruppen ‘studygroup’ anstatt nur ‘group’.

Die Erstellung der Datenbank erfolgt über die Datei ‘dump.sql’ im Ordner ‘database’.

# 6 Installationshinweise

Das Github-Repository mit dem vollständigen Source Code befindet sich unter <https://github.com/sihensel/teachingbee>.

## Vorraussetzungen

Um die Anwendung lokal ausführen zu können werden folgende Programme beziehungweise Laufzeitumgebungen benötigt:

* Python 3.9
* Node.js
* Terminal (wie es etwa in VSCode oder Pycharm eingebunden ist)
* Web-Browser

Darüber hinaus werden einige Python-Module benötigt, die über das Programm pip installiert werden können. Das Anlegen eines Virtual Environments ist empfohlen.

* Flask
* Flask-restx
* Flask-Cors
* mysql-connector-python

Um die benötigten Node-Packages zu installieren, im Terminal in den Ordner ‘/reactfrontend’ navigieren und **npm install** ausführen. Welche Module installiert werden, kann in der Datei ‘/reactfrontend/package.json’ eingesehen werden.

## Lokal

Um die App lokal zu installieren muss zuerst die MySQL-Datenbank lokal angelegt werden. Dies kann über ein Tool wie MySQL Workbench oder auch das CLI von MySQL geschehen. Für genauere Hiwneise ist die die Dokumentation des Herstellers zu konsultieren. Anschließend müssen in der Datei ‘Mapper.py’ im Ordner ‘/backend/server/db/’ noch die entsprechenden Login-Informationen angegeben werden.

Als nächster Schritt muss die Authentifizierung mit Google Firebase eingerichtet werden. Dazu in Firebase ein Projekt für eine Web-App anlegen. In diesem Zug wird automatisch eine Konfiguration erstellt, die dann in die Datei ‘/reactfrontend/firebaseconfig.js’ kopiert werden muss. Bei Schwierigkeiten kann die Dokumentation seitens Firebase konsultiert werden.

Danach kannd er React-Client kompiliert werden. Dazu im Terminal in den Ordner ‘reactclient’ navigieren und den Befehl **npm run build** ausführen. Das fertige Kompilat steht dann im neuen Order ‘build’ zur Verfügung. Der Inhalt dieses Ordners muss nun nach ‘backend/static/reactclient’ kopiert oder verschoben werden.

Anschließend kann der Flask-Server gestartet werden, indem die Datei ‘main.py’ ausgeführt wird. Die Anwendung steht dann im Browser unter <http://localhost:5000/static/reactclient/index.html> zur Verfügung. Die REST-API befindet sich unter <http://localhost:5000>.

## Cloud

Um die App auf Google Cloud bereit zu stellen, muss zuerst über die Google Cloud Console ein neues Projekt angelegt werden. Danach in Google Cloud eine MySQL-Datenbank erstellen und über die Datei ‘dump.sql’ befüllen. In der Datei /backend/server/db/Mapper.py wieder die entsprechenden Login-Informationen angeben.

Als nächstes muss die Google App Engine aktiviert und auf dem lokalen Rechner die Google Cloud SDK installiert werden. Hier müssen noch die beiden Komponenten **app-engine-python** und **app-engine-python-extras** installiert und das gerade erstellte Projekt ausgewählt werden. Es wird ausdrücklich auf die Dokumentation des Herstellers verwiesen. Anschließend die Firebase-Konfiguration erstellen und den React-Client kompilieren (siehe oben). Dann kann die Anwendung mit der Google Cloud SDK deployed werden. Dazu in den Ordner backend navigieren und den Befehl gcloud app deply ausführen.

Ist der Prozess abgeschlossen, kann die Anwendung mit gcloud app browse genutzt werden.

# 7 Nutzerhandbuch

Die App kann über die folgenden Links online aufgerufen werden:

* <https://teachingbee-179c2.ew.r.appspot.com/>
* <https://teachingbee-179c2.ew.r.appspot.com/static/reactclient/index.html>

Noch erklären was die einzelnen menüs machen