Technische Universität Braunschweig

ISF

Das Grosse SQL-Spiel

The SQL-Alchemist

Software-Entwicklungspraktikum (SEP)
Sommersemester 2015

Fachentwurf

Auftraggeber:
Technische Universität Braunschweig
Institut für Informationssysteme
Prof. Dr. Wolf-Tilo Balke
Mühlenpfordtstraße 23, 2.OG
D-38106 Braunschweig

Betreuer: Jan-Christoph Kalo

Auftragnehmer:

Name	E-Mail-Adresse
Gabriel Ahlers	g.ahlers@tu-braunschweig.de
Majid Dashtiepielehroud	m.dashtiepielehroud@tu-braunschweig.de
Ronja Friebe	r.friebe@tu-braunschweig.de
Stefan Hanisch	stefan.hanisch@tu-braunschweig.de
Fabio Luigi Mazzone	f.mazzone@tu-braunschweig.de
Nicole Naczk	n.naczk@tu-braunschweig.de
Denis Nagel	denis.nagel@tu-braunschweig.de
Luca Porcello	l.porcello@tu-braunschweig.de
Christian Reineke	c.reineke@tu-braunschweig.de
Christian Sander	christian.sander@tu-braunschweig.de
Carl Schiller	c.schiller@tu-braunschweig.de
Levent Muzaffer Üner	l.uener@tu-braunschweig.de
Sören van der Wall	s.van-der-wall@tu-braunschweig.de
Daniel Wolfram	d.wolfram@tu-braunschweig.de

Braunschweig, 3. Juni 2015

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Ziel dieses Dokuments ist es, einen tiefer gehenden Überblick über den internen Ablauf der zu entwickelnden Software zu bieten. Es werden zuerst die wichtigsten Funktionen einzeln beschrieben, wobei Sequenzdiagramme zu Hilfe genommen werden. Danach wird die Datenverwaltung näher beleuchtet.

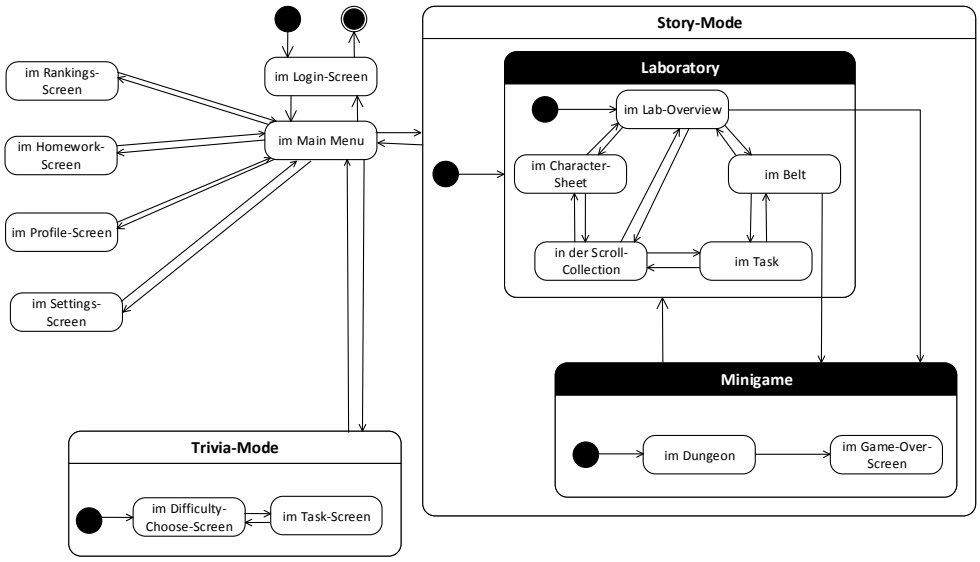

Abbildung 1.1: Zustandsdiagramm zum oberflächlichen Spielablauf

Um eine Verständnisgrundlage zu schaffen ist in Abbildung 1.1 der allgemeine Menüführung in einem Aktivitätsdiadramm dargestellt.

Nach dem Einloggen gelangt man in das Main Menu. Von dort aus gelangt man in die drei Spielmodi (Story, Trivia, Homework), die Einstellungen, das Profil und in die Ranglisten.

Der Story Mode hat wiederum eine eigene Übersicht, Laboratory genannt. Von hier aus kann der User sich seine aktuellen Playerstatistiken im Charakter Sheet ansehen, kann sich in der Scrollcollection um die Herstellung von Potions und Enchantments kümmern, diese danach in seinen Belt einfügen um sich im Anschluss dem Minispiel zu widmen. Im Minispiel werden dem User verschiedene Hindernisse vorgesetzt, die es zu überwinden gilt. Scheitert der User, so gelangt er in den Game Over Screen, in dem ihm angezeigt wird wie viele Lofi-Coins (die Spiel-Währung) und welche Scrolls er in dieser Runde eingesammelt hat. Nach Bestätigung befindet sich der User zurück im Laboratory.

Der Trivia-Mode stellt einen SQL-Trainer dar. Entscheidet der User sich dafür diesem zu verwenden, gelangt er in einen Screen in dem er sich für einen Schwierigkeitsgrad entscheiden muss. Hat der User dies getan, erhält er eine, dem ausgewählten Schwierigkeitsgrad entsprechende, Aufgabe und kann diese lösen.

Der Homework-Mode funktioniert genauso wie der Trivia-Mode, lediglich die Auswahl des Schwierigkeitsgrads entfällt in diesem Fall.

In den Rankings werden die Spieler nach verschiedenen Kriterien in Ranglisten sortiert. Darüber ist es auch möglich durch die Eingabe des Usernamens nach anderen Spielern zu suchen und sich deren Profile anzeigen zu lassen. Das eigene Profil ist über das Main Menu zu erreichen und einsehbar. Selbiges gilt für die Spieleinstellungen.

1.1 Projektdetails

Die Anzahl an Lofi-Coins, sowie die Anzahl an Scroll die der User pro Tag einsammeln kann, werden beschränkt. Dies soll verhindern, dass der User nicht den Fokus, des Übens von SQL-Statements, verliert. Auf der anderen Seite jedoch, soll so der Spieler zum Wiederkommen animiert werden.

Desweiteren wurde im Pflichtenheft erwähnt, dass Spieler sich durch Aufgabenerstellung ins Spiel mit einbringen können. Um dabei jedoch eine gewisse Qualität zu gewährleisten zu können, wird es nur beförderten Nutzern möglich sein, eigene Aufgaben zu erstellen. Wie ein User befördert wird, wird über die Spielzeit oder über Leistungen im Spiel geregelt.

2 Analyse der Produktfunktionen

Im Folgenden werden die im Pflichtenheft benannten Funktionen näher beschrieben und erklärt. Dabei wird davon ausgegangen, dass der User sich in dem Interface befindet, in dem er die erklärte Funktion initiieren kann.

Desweiteren ist zu erkennen, dass, jedes mal wenn das Back-End nach der Registration oder dem Login aktiv wird, es zuerst einen „checkSession()“-Methodenaufruf startet. Dies ist die Überprüfung des Back-Ends, ob es selbst den User kennt, ob er die gebrauchten Rechte für die Aktion hat und ob die Aktion auch auf die zum User gehörigen Daten ausgeführt wird.

2.1 Analyse von Funktionalität <F10>: <Nutzer registrieren>

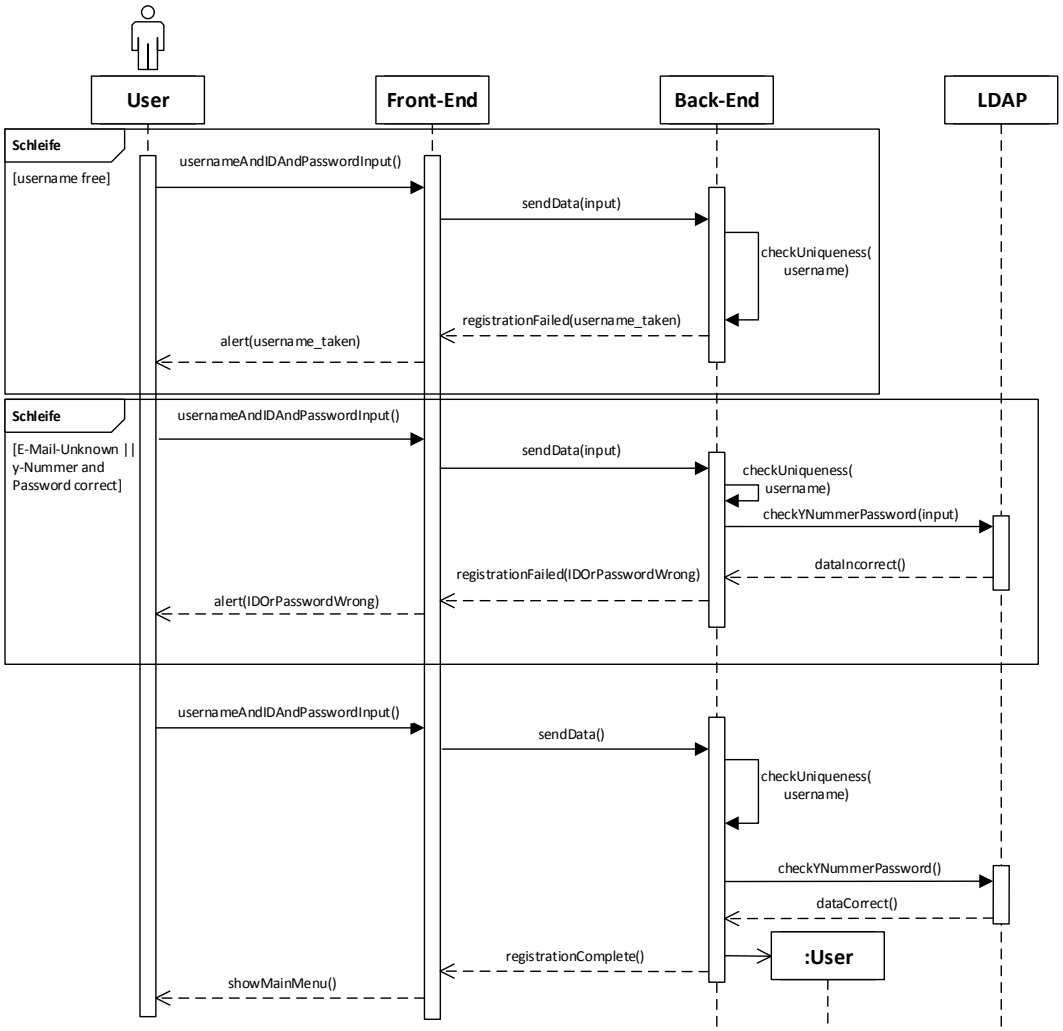

Abbildung 2.1: Sequenzdiagramm zur Registrierung

In Diagramm 2.1 wird der Vorgang der Registrierung beschrieben.

Dafür muss der User zuerst einen Username, eine E-Mail-Adresse oder seine y-Nummer (insgesamt ID genannt) und ein Passwort (im Falle einer y-Nummer das zur y-Nummer gehörende Passwort) angeben. Diese Eingaben werden dann an das Back-End übertragen. Hier wird zuerst geprüft ob der Username schon vergeben ist. Sollte das der Fall sein, wird dies dem User mitgeteilt und er muss einen neuen Username angeben. Wenn der Username noch nicht vergeben war, wird, sollte die ID eine E-Mail-Adresse sein, geprüft, ob diese schon vorhanden ist. Wenn die ID eine y-Nummer ist wird diese inklusive des eingegebenen Passworts an das „LADP“geschickt und dort überprüft. Sollte der jeweils zutreffende Schritt fehlschlagen, muss der User seine Eingaben ändern beziehungsweise korrigieren und der Prozess beginnt von vorn.

Sollte der Ablauf erfolgreich verlaufen sein, erstellt das Back-End ein neues User-Objekt, speichert dies in seiner Datenbank und gibt die Rückmeldung, dass die Registrierung erfolgreich verlaufen ist und der User wird in das Hauptmenü weitergeleitet.

2.2 Analyse von Funktionalität <F20>: <Nutzer anmelden>

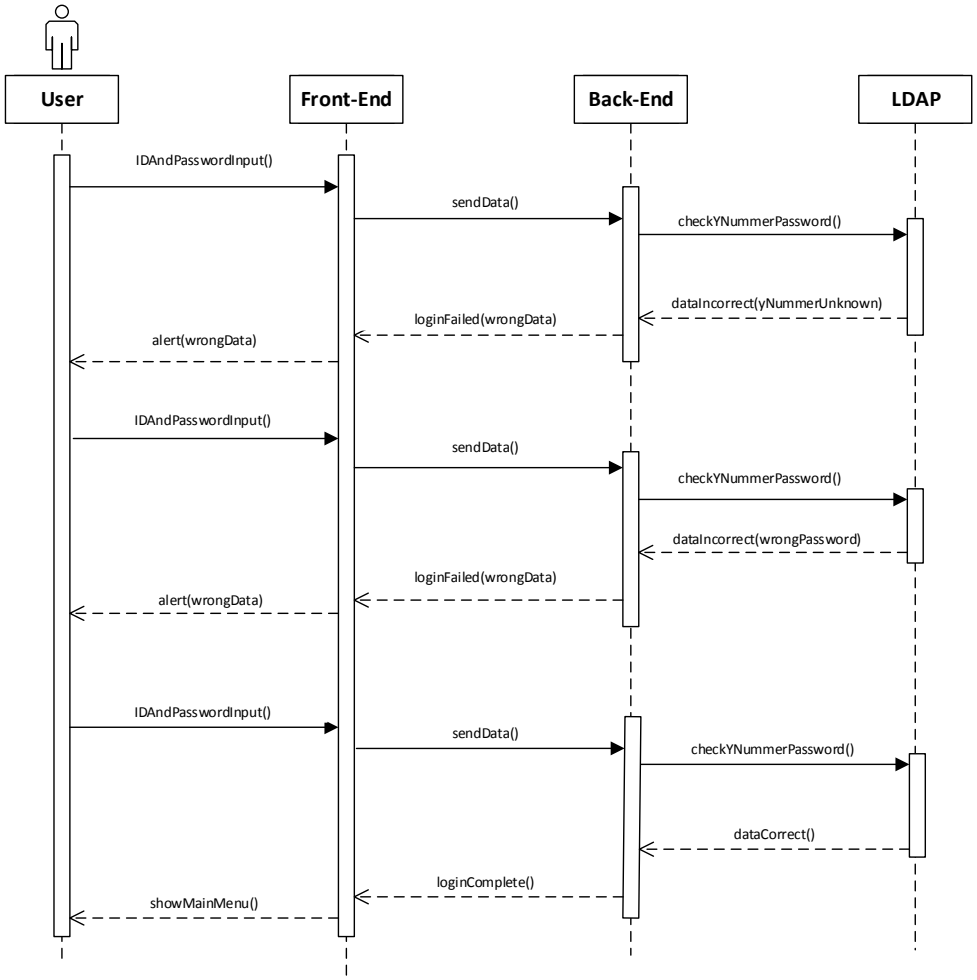

Abbildung 2.2: Sequenzdiagramm zum Login

Das Diagramm 2.2 beschreibt den Login-Vorgang.

Hierbei muss der User die von ihm registrierte ID (siehe <F10>) und sein Passwort angeben. Diese werden, je nach Typ der ID, dann entweder mit der eigenen Datenbank verglichen, oder, sollte die ID eine y-Nummer sein, an das „LDAP“geschickt und dort überprüft. Sollten die eingegeben Daten nicht korrekt sein, wir dies dem User mitgeteilt und er bekommt die Möglichkeit seine Eingaben zu korrigieren. Sind die Daten korrekt wird der User in das Hauptmenü weitergeleitet.

2.3 Analyse von Funktionalität <F30>: <Nutzer abmelden>

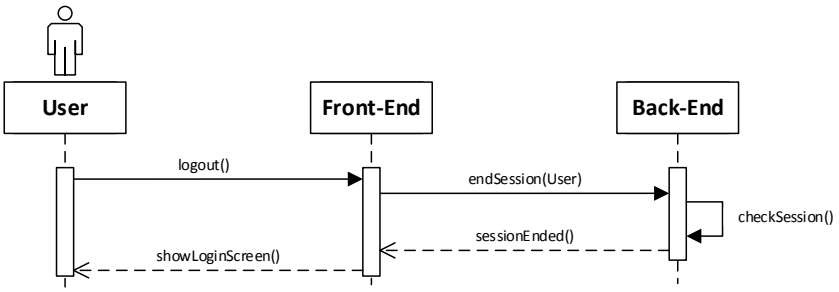

Abbildung 2.3: Sequenzdiagramm zum Login

Das Diagramm 2.3 beschreibt den Ablauf des Ausloggens.

Entscheidet sich der User sich auszuloggen, wir diese Information an das Back-End gesendet. Dieses beendet die Session des Users. Ist das geschehen wird eine Benachrichtigung an das Front-End gesendet und der User wird zurück zum Login-Screen geleitet.

2.4 Analyse von Funktionalität <F40>: <Profil einsehen>

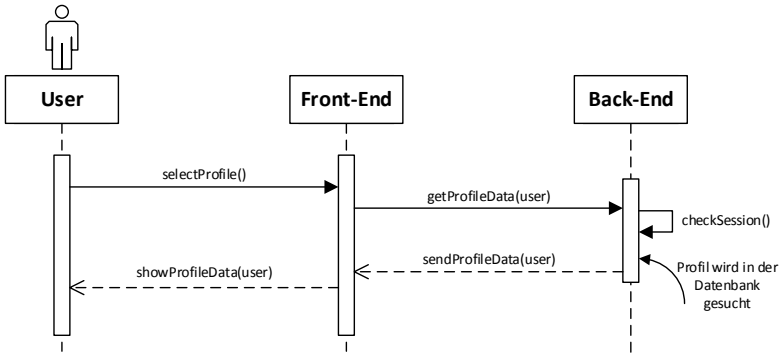

Abbildung 2.4: Sequenzdiagramm zur Darstellung der Profilübersicht

Das Diagramm 2.4 zeigt was passiert, wenn man sich das eigene Userprofil anzeigen lassen möchte.

Um das Profil des Users anzuzeigen werden zuerst die Userdaten vom Back-End angefordert. Das Back-End lädt dann die Profildaten aus den Userdaten und schickt diese zurück an das Front-End. Dort werden sie für den User einsehbar im Profil-Interface angezeigt.

2.5 Analyse von Funktionalität <F60>: <Passwort ändern>

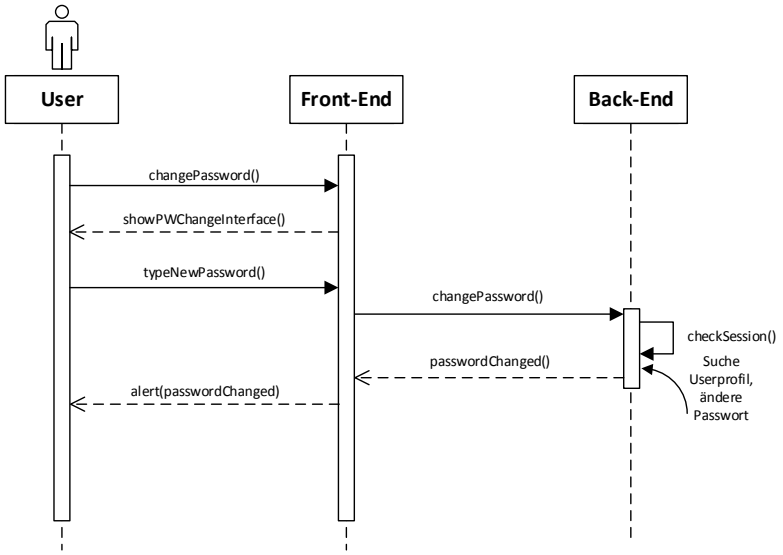

Abbildung 2.5: Sequenzdiagramm zum Ändern des Passworts

Das Diagramm 2.5 beschreibt das Ändern des Passwortes.

Das Ändern des Passwortes steht nur nicht-studentischen Nutzern zur Verfügung. Entscheidet sich der Nutzer, sein Passwort zu ändern, kann er dies mit einem Klick auf den Change Password Button tun. Zuerst muss er sein altes Passwort eingeben und im Anschluss kann er ein neues Password erstellen. Dies muss danach noch einmal bestätigen werden und wenn alles erfolgreich war, wird das geänderte Passwort an das Back-End geschickt und dort in den Nutzerdaten vermerkt. Sollte während des Vorgangs etwas schief gehen, bleibt das alte Passwort bestehen und der User wird darüber in Kenntnis gesetzt.

2.6 Analyse von Funktionalität <F70>: <Avatar ändern>

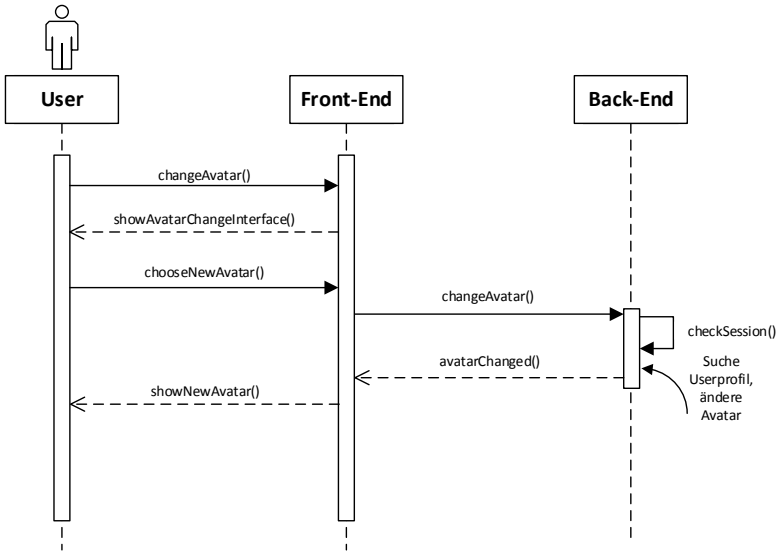

Abbildung 2.6: Sequenzdiagramm zum Ändern des Avatars

Das Diagramm 2.6 beschreibt das Ändern des Avatars.

Klickt der User auf den Change Avatar Button, werden ihm alle seine verfügbaren Avatare angezeigt und er kann sich für einen entscheiden. Hat er dies getan wird die Änderung der Einstellung vermerkt, ans Back-End geschickt, dort gespeichert und dann, wieder zurück im Front-End, aktualisiert angezeigt.

2.7 Analyse von Funktionalität <F80>: <Benutzer löschen>

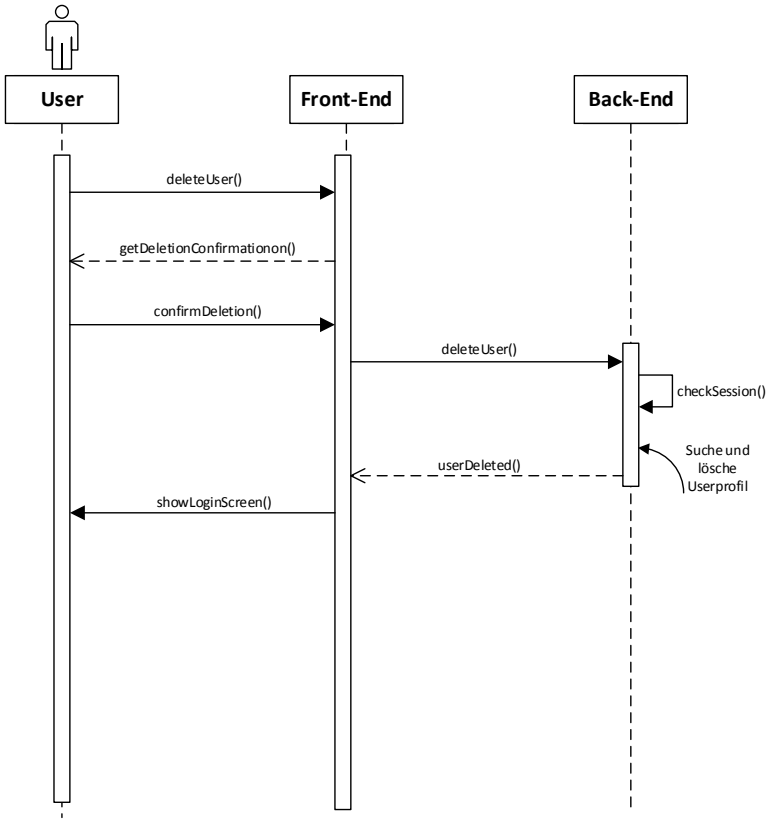

Abbildung 2.7: Sequenzdiagramm zum Löschen des Benutzers

Diagramm 2.7 zeigt, was passiert, wenn der User seinen Account löschen möchte.

Möchte der User sein Profil löschen, kann er dies im Settings-Interface tun, was dem entsprechend erst geladen werden muss. Wenn der User nun den Delete User Button drückt, wird der User erst noch einmal gefragt, ob er ein Profil wirklich löschen möchte. Beantwortet der User dies positiv geht eine Mitteilung an das Back-End, wo das Profil gelöscht wird und der User auf den Login-Screen geleitet wird wo es ihm frei steht, sich wieder zu registrieren.

2.8 Analyse von Funktionalität <F90>: <Audioeinstellungen bearbeiten>

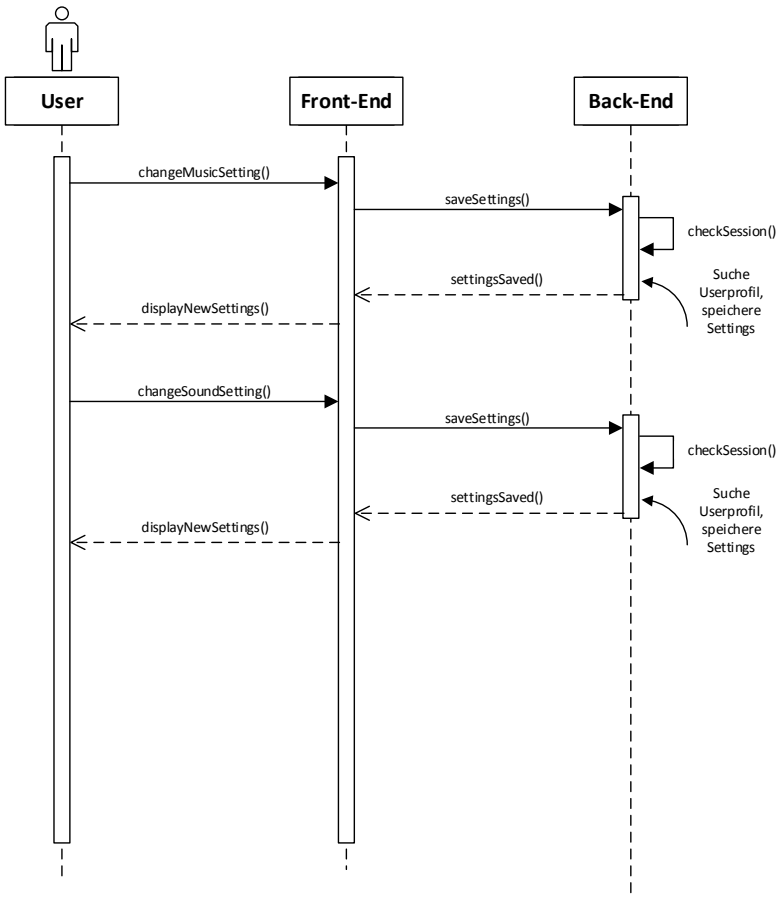

Abbildung 2.8: Sequenzdiagramm zum Ändern der Audioeinstellungen

Diagramm 2.8 zeigt, was passiert, wenn der User seine Audio-Optionen ändert.

Zum Ändern der Audiofunktionen stehen Knöpfe bereit, die sowohl die Hintergrundmusik („Music“) als auch die Soundeffekte, wie Sprungsounds und Klicksounds, aus- beziehungsweise einstellen. Jegliche Änderung wird sofort ans Back-End gesendet, dort gespeichert und dann im Profil aktualisiert.

2.9 Analyse von Funktionalität <F100>: <Spielstand zurücksetzen>

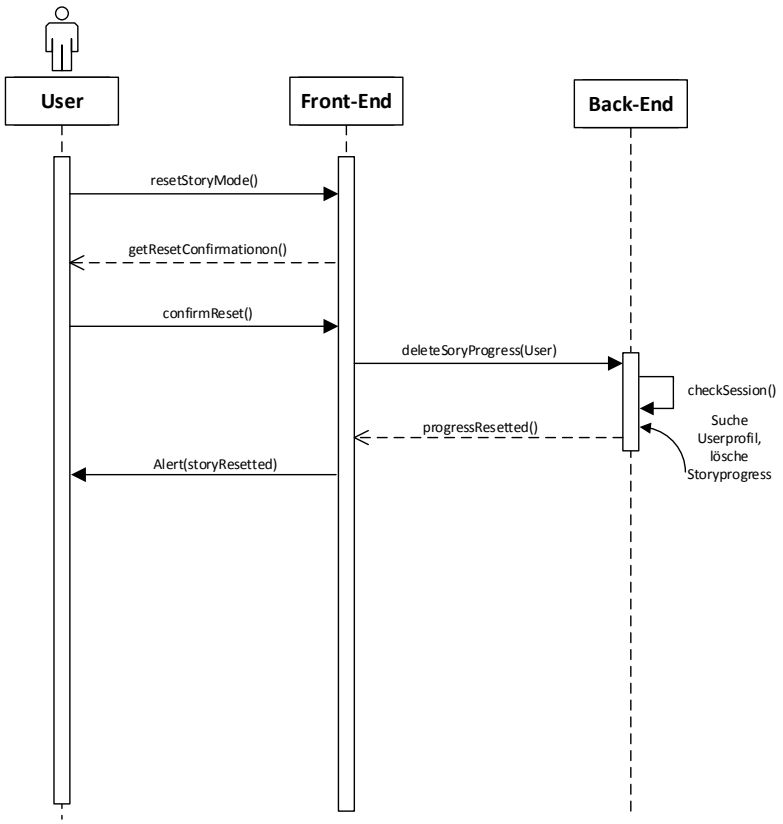

Abbildung 2.9: Sequenzdiagramm zum Zurücksetzen des Spielstands

Diagramm 2.9 beschreibt das Zurücksetzen des Storyfortschrittes.

Der User kann per Knopfdruck seinen Story-Fortschritt zurücksetzen. Tut er dies, muss er vorher noch einmal seine Zustimmung zum Löschvorgang geben. Ist auch dies geschehen, wird die Anweisung zum Löschen des Storyfortschrittes des Users an das Back-End geschickt und dort ausgeführt.

2.10 Analyse von Funktionalität <F110>: <Tutorial spielen>

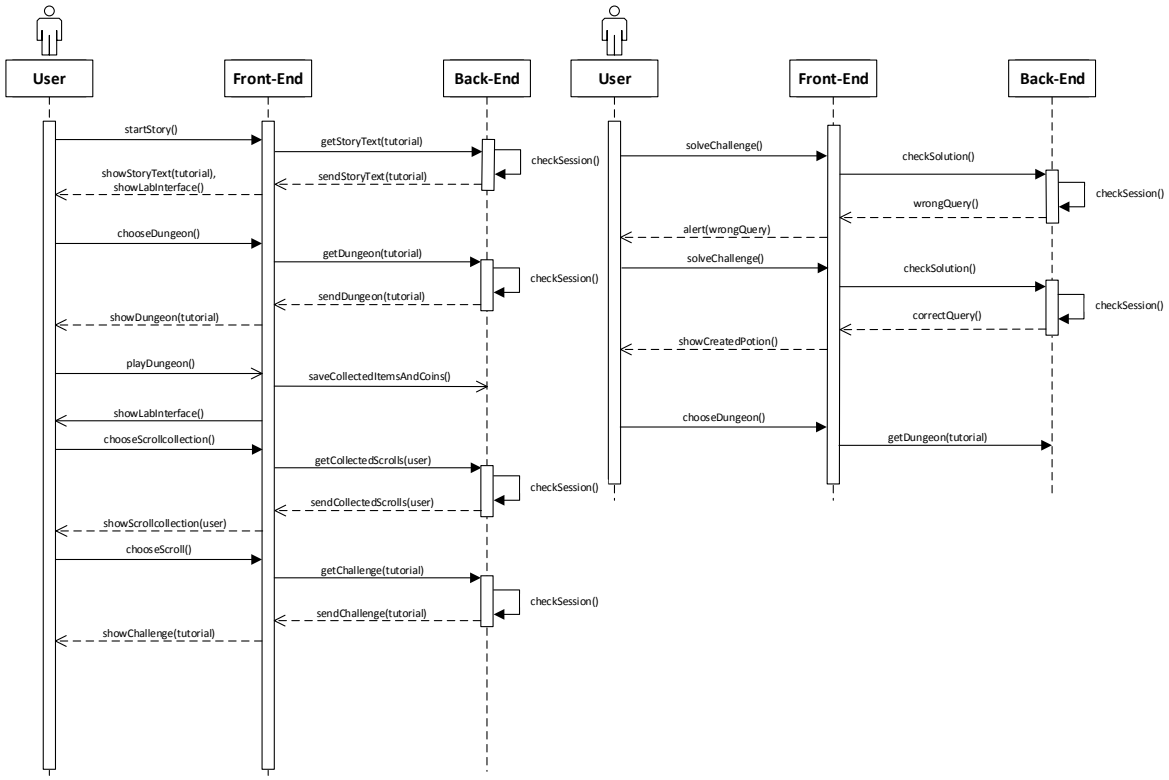

Abbildung 2.10: Sequenzdiagramm für das Tutorial

Diagramm 2.10 zeigt den Ablauf im Tutorial.

Wird das Tutorial gestartet, werden zuerst alle Tutorial-Texte vom Back-End angefordert. Diese werden dann, wenn benötigt, dem User angezeigt. Zuerst wird der User in den Dungeon geleitet. Dafür werden die Leveldaten vom Back-End geladen, sodass der User den Dungeon betreten und Spielen kann. Hierbei kann er Schriftrollen („Scrolls“) und Münzen („Lofi-Coins“) einsammeln, was jeweils sofort ans Back-End gesendet und dort im Story-Progress bzw. in den Profildaten gespeichert wird. Scheitert der User an einer Hürde im Dungeon, bekommt er zuerst einen Game-Over-Screen angezeigt, auf dem zusehen ist, was er eingesammelt hat und wird zurück in den Labor-Screen geleitet. Dort wird er per Tutorial-Text zur Scrollcollection geleitet um sich dort für ein Trank-Rezept zu entscheiden. Ist dies getan, wird aus dem Back-Ende eine für das Rezept passende Aufgabe angefordert und dem User angezeigt. Für diese Aufgabe hat der User keine Versuchsbegrenzung. Die Eingaben des Users werden dabei immer an das Back-End gesendet und dort kontrolliert. Hat der User die Aufgabe richtig gelöst, erhält er eine Potion und kann diese danach im Dungeon verwenden.

2.11 Analyse von Funktionalität <F120>: <Story spielen>

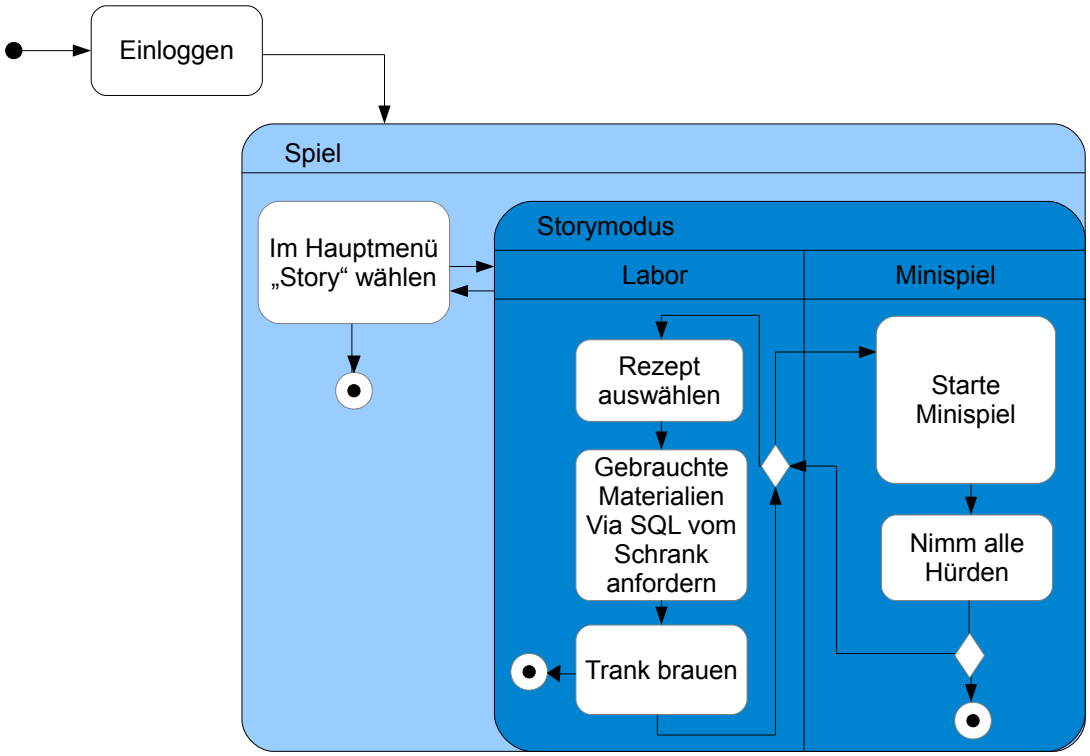

Abbildung 2.11: Aktivitätsdiagramm für den Story Mode

Der Story-Mode besteht aus sich abwechselnden Teilen aus SQL-Trainer und Minispiel. Um dies besser zu zeigen, ist in Abbildung 2.11 das Aktivitätsdiagramm für den Story Mode aus dem Pflichtenheft abgebildet.

Darin ist zusehen, dass man sich im Labor einen oder mehrere Tränke brauen kann (SQL-Trainer-Anteil) um diese dann im Dungeon zu verwenden (Minispiel-Anteil). Um die beiden Teile für sich besser beschreiben zu können, sind diese in den folgenden zwei Funktionen aufgeführt.

2.12 Analyse von Funktionalität <F130>: <SQL-Trainer spielen>

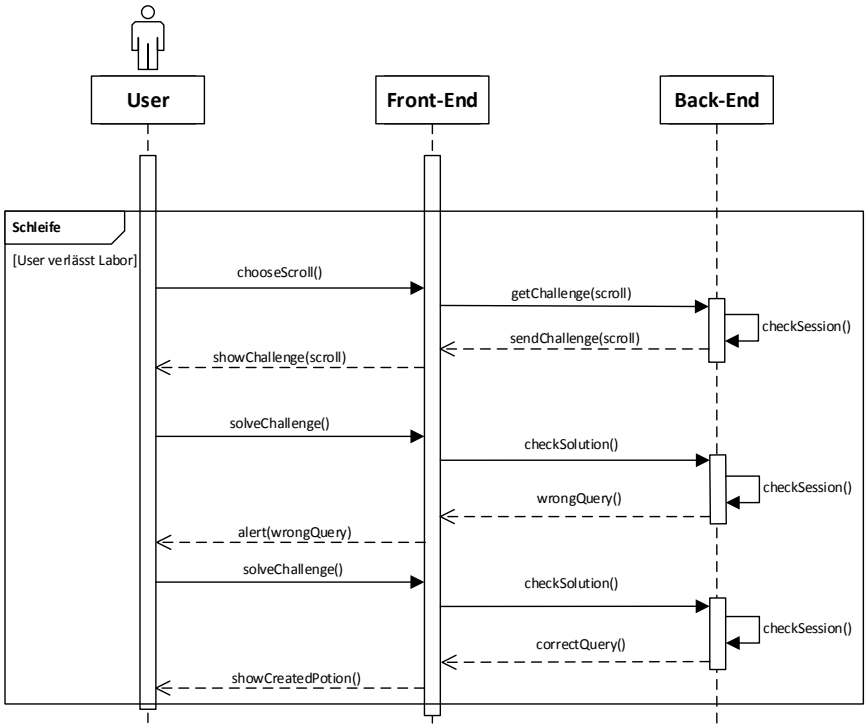

Abbildung 2.12: Sequenzdiagramm für den SQL-Trainer

Der SQL-Trainer ist Teil der drei Spielmodi (Story, Trivia, Homework) und wird einmal, wie er im Trivia-Mode verwendet wird, erklärt. Die leichten Abweichungen der anderen Spielmodi werden im Anschluss erwähnt. Diese benötigen wenig bis gar keine weitere Erklärung, da sie nur einen oberflächlichen Unterschied machen.

Zuerst werden dem User fünf Schwierigkeitsgrade angezeigt aus denen der er wählen kann. Hat er sich für einen Schwierigkeitsgrad entschieden, wird dies dem Back-End mitgeteilt, was daraufhin eine, dem gewählten Schwierigkeitsgrad entsprechende, Aufgabe bereitstellt. Diese kann dann vom User gelöst werden. Die Eingaben werden dabei immer ans Back-End geschickt und dort auf Richtigkeit geprüft.

Ist die Prüfung positiv verlaufen wird der User wieder zur Schwierigkeitsgrad-Auswahl weitergeleitet.

Abweichungen zum Story-Mode:
Es werden keine Schwierigkeitsgrade angezeigt, sondern schon eingesammelte Rezepte für Tränke. Diese beinhalten in ihrer Definition schon die Schwierigkeitsgrade für die daraus hervorgehenden Aufgaben.

Abweichungen zum Homework-Mode:
Beim Homework-Mode wird die Auswahl des Schwierigkeitsgrades in jeglicher Form übersprungen und der Aufgabentext wird direkt angezeigt.

2.13 Analyse von Funktionalität <F140>: <Minispiel spielen>

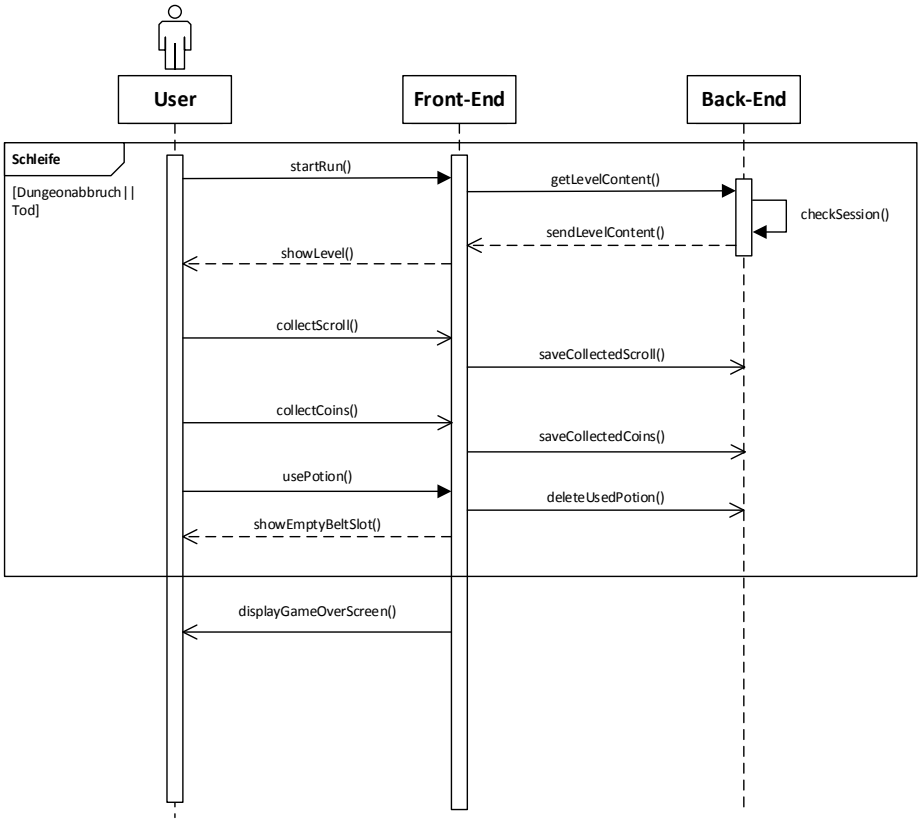

Abbildung 2.13: Sequenzdiagramm für das Minigame

Die Abbildung 2.13 zeigt die Abläufe im Dungeon.

Im Minispiel bewegt sich die Figur stetig nach rechts. Der User kann springen, wodurch er verschiedene Hindernisse überwinden kann. Desweiteren kann der User Lofi-Coins und Scrolls einsammeln. Dies wird sofort im Back-End vermerkt und im Spielerprofil und im Storyprogress gespeichert. Dem User ist zudem möglich vorher erstellte Potions zu verwenden um deren Effekte zur Überwindung von Hindernissen zu nutzen.

2.14 Analyse von Funktionalität <F150>: <Hausaufgaben bearbeiten>

Die Bearbeitung von Hausaufgaben funktioniert wie das Nutzen des SQL-Trainers. Genauere Erklärungen sind dort (F130) zufinden.

2.15 Analyse von Funktionalität <F160>: <Ranglisten einsehen>

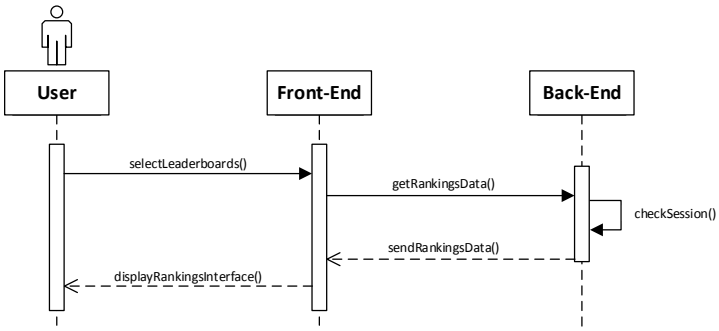

Abbildung 2.14: Sequenzdiagramm für die Ranglisten

In Abbildung 2.14 wir das Anzeigen der Ranglisten dargestellt.

Möchte der User die Ranglisten einsehen, wird eine Anfrage an das Back-End gesendet. Das Back-End stellt dann die Daten für die Ranglisten zusammen und sendet diese dann zurück ans Front-End wo diese dann für den User einsehbar präsentiert werden.

2.16 Analyse von Funktionalität <F170>: <Spieler suchen>

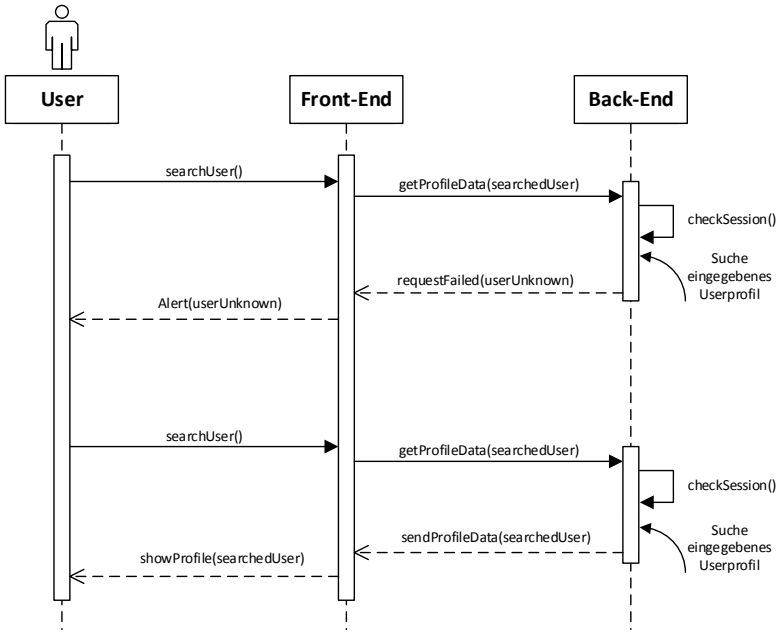

Abbildung 2.15: Sequenzdiagramm zum Suchen eines anderen Users

Im Diagramm 2.15 ist zu sehen, wie man nach einem anderen User suchen kann.

Um einen anderen User zu suchen wird ein Usersuche-Feld bereitgestellt. Der dort eingegebene Name wird an das Back-End weitergeleitet und dort in der Userdatenbank gesucht und wenn er existiert wird dessen Profil aus der Datenbank geladen dem suchenden User angezeigt. Ist der Name in der Datenbank nicht zu finden, wird dem suchenden User das mitgeteilt.

2.17 Analyse von Funktionalität <F180>: <Hausaufgabenergebnisse einsehen>

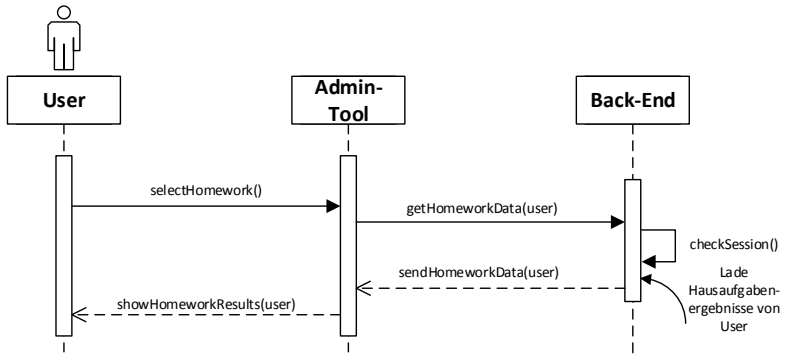

Abbildung 2.16: Sequenzdiagramm zum einsehen der eigenen Hausaufgabenergebnisse

Abbildung 2.16 beschreibt das Anzeige der Hausaufgabenergebnisse.

Nach dem Einloggen in das Admintool werden die Hausaufgabenergebnisse eines nicht-beförderten Users direkt aus der Datenbank geladen und angezeigt. Ist der User schon „befördert“muss er erst noch auf den Show Homework Results Button drücken.

2.18 Analyse von Funktionalität <F190>: <Benutzer befördern>

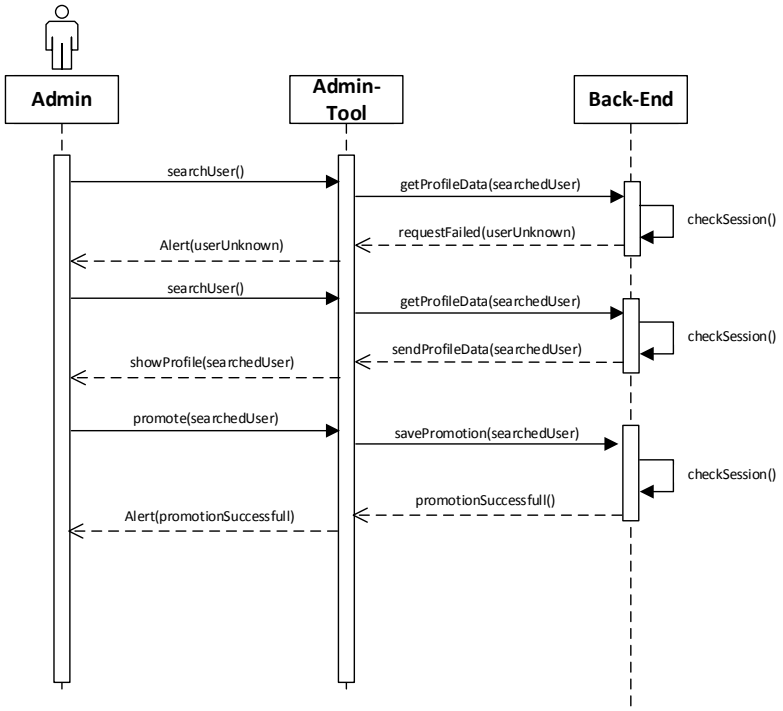

Abbildung 2.17: Sequenzdiagramm zum Befördern eines Users

Abbildung 2.17 beschreibt das Befördern eines Benutzers.

Zuerst wird für den Admin die Userdatenbank angzeigt. Hier kann er entweder manuell oder per Suchfunktion nach einem User suchen und per Knopfdruck befördern, was dann im Back-End in den jeweiligen Userdaten registriert und gespeichert wird.

Der Ablauf, einen User Adminrechte (<F200>) zu geben, gleicht dem des User Beförderns.

2.19 Analyse von Funktionalität <F210>: <Eine Trivia-Aufgabe erstellen>

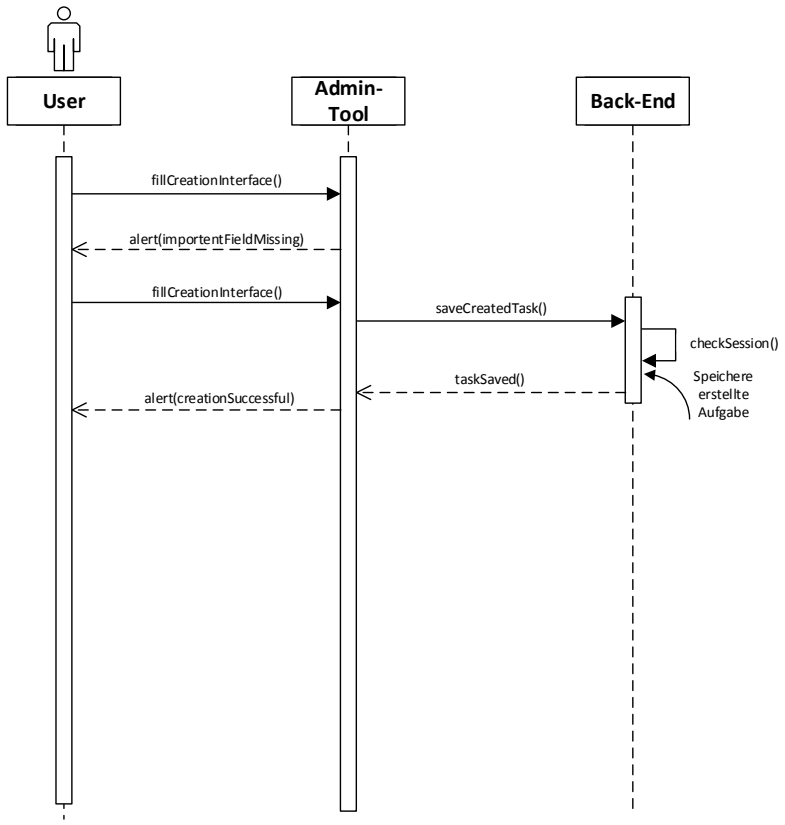

Abbildung 2.18: Sequenzdiagramm für das Erstellen von Aufgaben

Abbildung 2.18 beschreibt das Erstellen von eigenen Aufgaben.

Beförderte Nutzer können selber Aufgaben erstellen die später im Trivia Mode verwendet werden können. Dafür steht im Admin-Tool ein Interface zur Verfügung in das ganz einfach alle zur Aufgabe gehörigen Daten einsammelt und dann ans Back-End sendet, welches die Aufgabe dann speichert. Sind nicht alle wichtigen Felder ausgefüllt, wird der User alarmiert und muss dies berichtigen damit dich Aufgabe gespeichert werden kann.

2.20 Analyse von Funktionalität <F220>: <Benutzeraufgaben bewerten>

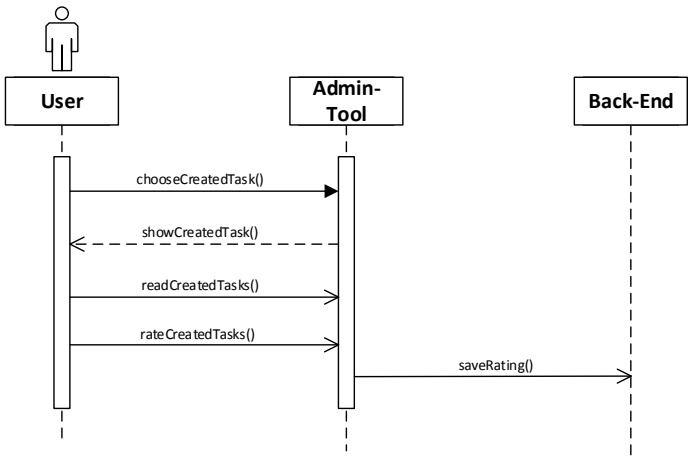

Abbildung 2.19: Sequenzdiagramm zum bewerten User-erstellter Aufgaben.

Abbildung 2.19 zeigt den Ablauf, wie man eine User-erstellte Aufgabe bewertet.

In der Liste aller User-erstellten Aufgaben kann sich der User Aufgaben aussuchen, diese Ansehen und sie bewerten und kommentieren. Die Bewertung wird dann im Back-End für die Aufgabe registriert und gespeichert.

2.21 Analyse von Funktionalität <F230>: <Hausaufgaben erstellen>

Die Erstellung von Hausaufgaben gleicht der User-seitigen Erstellung von Aufgaben, ist jedoch nur Administratoren zugänglich.

Der Unterschied zur normalen Aufgabenerstellung besteht darin, dass ein Aufgabenpaket erstellt wird. Dabei kann man entweder Aufgaben für das Paket neu erstellen, oder auf schon einmal erstellte Aufgaben, die in der Datenbank gespeichert sind, zugreifen. Diese Aufgabenpakete können dann noch mit zusätzlichen Parametern versehen werden, wie zum Beispiel einem Bearbeitungszeitraum oder einer Anzahl an Versuchen.

3 Datenmodell

Das folgende Kapitel beschreibt die Datensätze, welche der SQL-Alchemist dauerhaft oder teilweise auch nur temporär abspeichert. Dazu werden zuerst einige Erläuterungen zu den einzelnen Beziehungen angegeben und zum Ende des Kapitels ist ein Klassen-Diagramm zur Veranschaulichung dieses Sachverhaltes abgebildet.

3.1 Erläuterung

Avatar $\langle E10 \rangle$

Beziehung	Kardinalität	Erwartete Datenmenge	Beschreibung
Profile	0 ... *	Min: <...>, Max: <...>	Der Avatar wird vom Profil als Erkennungsmerkmal und Spielfigur verwendet.
Profile_Avatar	0 ... *	Min: <...>, Max: <...>	Verweis auf alle vom Nutzer gekauften Avatare.

Die Entitäten vom Typ „Avatar“ beschreiben die verschiedenen vom Spiel zur Verfügung gestellten Spielfiguren. Diese werden sowohl im Minispiel als auch als Erkennungsmerkmal der Profile der Nutzer verwendet. In der Datenbank werden unter anderem die zugehörigen Eigenschaften, sowie die Darstellungsmerkmale gespeichert. **Hinweis:** In der Relation Profile_Avatar sind die Avatare mit den Profilen verknüpft, das Profil verfügt jedoch zusätzlich über das Attribut Ävatar". Der im Profil gespeicherte Avatar ist dabei der aktuell verwendete.

Bag_Slot ⟨*E*20⟩

Beziehung	Kardinalität	Erwartete Datenmenge	Beschreibung
Potion	1	Min: <...>, Max: <...>	In Bagslots werden die bereits gesammelten Tränke gespeichert.
Profile	1	Min: <...>, Max: <...>	Ein Bagslot gehört einem Profil.

Bei den Entitäten des Typs „Bag_Slot“ handelt es sich um die einzelnen Plätze innerhalb des Inventars der Nutzer, in denen alle hergestellten Tränke abgelegt werden. Dabei nimmt jeder Slot genau einen Trank auf.

Challenge ⟨*E*30⟩

Beziehung	Kardinalität	Erwartete Datenmenge	Beschreibung
Map_In_Challenge	0 ... *	Min: <...>, Max: <...>	Eine Challenge kann mit mehreren Maps verknüpft sein.
Solve_Challenge	0 ... *	Min: <...>, Max: <...>	Bearbeitungsfortschritt der Challenge, verknüpft mit dem Profil
Text_In_Challenge	0 ... *	Min: <...>, Max: <...>	Story-Texte können Challenges zugeordnet werden, um diese in den Kontext der Handlung einzubinden.
Task_In_Challenge	1 ... *	Min: <...>, Max: <...>	Eine Challenge besteht aus mindestens einer Aufgabe.

Die „Challenge“-Objekte beschreiben die Aufgabenpakete, welche zum Beispiel im Laufe der Story oder als Hausaufgaben an die Nutzer gestellt werden. Jedes dieser Pakete besteht aus verschiedenen Aufgaben. Als Attribute werden alle Informationen, die zur Beschreibung eines solchen Paketes benötigt werden in der Datenbank gespeichert.

Map $\langle E40 \rangle$

Beziehung	**Kardinalität**	**Erwartete Datenmenge**	**Beschreibung**
Map_In_Challenge	0 ... *	Min: <...>, Max: <...>	Eine Map kann in mehreren Challenges verwendet werden.

Bei den „Map“-Entitäten handelt es sich um die verschiedenen Levels, aus denen das Minispiel besteht.

Map_In_Challenge $\langle E50 \rangle$

Beziehung	**Kardinalität**	**Erwartete Datenmenge**	**Beschreibung**
Challenge	1	Min: <...>, Max: <...>	Verweis auf die Challenge
Map	1	Min: <...>, Max: <...>	Verweis auf die Map

In diesen Objekten werden die Beziehungen der „Maps“ und der „Challenges“ festgehalten. Dies umfasst, welche Level des Minispiels für die Lösung bestimmter Aufgabenpakete abgeschlossen werden müssen und in welcher Reihenfolge diese auftreten.

Potion ⟨*E*60⟩

Beziehung	Kardinalität	Erwartete Datenmenge	Beschreibung
Bag_Slot	0 ... *	Min: <...>, Max: <...>	Die Potion wird in einem Bagslot abgelegt.
Scroll	1	Min: <...>, Max: <...>	Die Potion ist einer Scroll zugeordnet.
Task	1 ... *	Min: <...>, Max: <...>	Zum Erzeugen der Potion muss eine Aufgabe gelöst werden.

Die „Potion"-Entitäten beschreiben die Tränke, welche die Spieler im Laufe des Spiels herstellen. Dazu werden die Auswirkungen der Tränke als Attribute gespeichert.

Profile ⟨*E*70⟩

Beziehung	Kardinalität	Erwartete Datenmenge	Beschreibung
Avatar	1	Min: <...>, Max: <...>	Jedes Profil besitzt einen Avatar, der dieses repräsentiert.
Bag_Slot	0 ... *	Min: <...>, Max: <...>	Bagslots, die dem Profil gehören.
Profile_Avatar	1 ... *	Min: <...>, Max: <...>	Dem Profil zur Verfügung stehende Avatare.

Comment_Task	0 ... *	Min: <...>, Max: <...>	In dieser Relation wird auf die vom Nutzer kommentierten Aufgaben verwiesen.
Rate_Task	0 ... *	Min: <...>, Max: <...>	In dieser Relation wird auf die vom Nutzer bewerteten Aufgaben verwiesen.
Profile_Scroll	0 ... *	Min: <...>, Max: <...>	Das Profil sammelt Schriftrollen in der Scrollcollection.
Profile_Shop_Item	0 ... *	Min: <...>, Max: <...>	Relation mit gekauften Items im Shop.
Solve_Challenge	0 ... *	Min: <...>, Max: <...>	Challenges, zu denen der Nutzer Zugang hat.
Solve_Task	0 ... *	Min: <...>, Max: <...>	Tasks, zu denen der Nutzer Zugang hat.
User	1	Min: <...>, Max: <...>	Jedes Profil gehört einem Nutzer.

Die Entitäten des Typs „Profile“ stellen die zentrale Verwaltungsstelle der Nutzer dar. Über diese wird der Fortschritt der Spieler abgewickelt, deren Einstellungen gespeichert und die Eigenschaften der Spielfiguren festgehalten.

Profile_Avatar ⟨*E*80⟩

Beziehung	Kardinalität	Erwartete Datenmenge	Beschreibung
Avatar	1	Min: <...>, Max: <...>	Verweis auf ein Avatar
Profile	1	Min: <...>, Max: <...>	Verweis auf ein Profil

Diese Entitäten halten die durch ein Profil gekauften Avatare fest und speichert dazu wann die Transaktion durchgeführt wurde.

Comment_Task ⟨*E*90⟩

Beziehung	Kardinalität	Erwartete Datenmenge	Beschreibung
Profile	1	Min: <...>, Max: <...>	Verweis auf das Profil
Task	1	Min: <...>, Max: <...>	Verweis auf die Aufgabe

Über die „Comment_Task“-Objekte werden die durch die Nutzer kommentierten Aufgaben gespeichert. Als Attribut wird dazu unter anderem das jeweilige Kommentar festgehalten.

Rate_Task ⟨*E*100⟩

Beziehung	Kardinalität	Erwartete Datenmenge	Beschreibung
Profile	1	Min: <...>, Max: <...>	Verweis auf das Profil

Task	1	Min: <...>, Max: <...>	Verweis auf die Aufgabe

Über die „Rate_Task“-Objekte werden die durch die Nutzer bewerteten Aufgaben gespeichert. Als Attribut wird dazu unter anderem die jeweilige Bewertung festgehalten.

Profile_Scroll $\langle E110 \rangle$

Beziehung	**Kardinalität**	**Erwartete Datenmenge**	**Beschreibung**
Profile	1	Min: <...>, Max: <...>	Verweis auf das Profil
Scroll	1	Min: <...>, Max: <...>	Verweis auf die Scroll

Hier werden die von den Nutzern gesammelten Scrolls gespeichert.

Profile_Shop_Item $\langle E120 \rangle$

Beziehung	**Kardinalität**	**Erwartete Datenmenge**	**Beschreibung**
Profile	1	Min: <...>, Max: <...>	Verweis auf das Profil
Shop_Item	1	Min: <...>, Max: <...>	Verweis auf das Shop-Item

Die Entitäten des Typs „Profile_Shop_Item“ halten fest, welche Spielgegenstände die Spieler bisher bereits erworben haben.

Solve_Challenge $\langle E130 \rangle$

Beziehung	Kardinalität	Erwartete Datenmenge	Beschreibung
Challenge	1	Min: <...>, Max: <...>	Verweis auf die Challenge
Profile	1	Min: <...>, Max: <...>	Verweis auf das Profil

Diese Objekte halten den Bearbeitungsfortschritt der Nutzer bezüglich der Aufgabenpakete fest. Es werden dabei sowohl die bereits gelösten Aufgabenpakete, als auch die noch nicht abgeschlossenen Aufgabenpakete abgespeichert.

Solve_Task $\langle E140 \rangle$

Beziehung	Kardinalität	Erwartete Datenmenge	Beschreibung
Profile	1	Min: <...>, Max: <...>	Verweis auf das Profil
Task	1	Min: <...>, Max: <...>	Verweis auf die Aufgabe.

Diese Entitäten sorgen für die Speicherung des Fortschritts bei der Lösung der einzelnen Aufgaben durch die verschiedenen Nutzer. Dabei werden unter anderem die benötigte Zeit, sowie das Lösungsdatum als Attribute gespeichert.

Scroll ⟨$E150$⟩

Beziehung	Kardinalität	Erwartete Datenmenge	Beschreibung
Potion	0 .. 1	Min: <...>, Max: <...>	Jede Schriftrolle schaltet eine oder keine Potion frei.
Profile_Scroll	0 ... *	Min: <...>, Max: <...>	Eine gesammelte Schriftrolle wird in der Scrollcollection des Profils gespeichert.

Die „Scrolls“ beschreiben die im Spiel verteilten Schriftrollen, welche durch die Spieler eingesammelt werden um neue Tränke oder temporäre Verbesserungen der Eigenschaften der Spielfigur (Buffs) freischalten. **Hinweis:** Wird durch die Scroll keine Potion freigeschaltet, erhält der Nutzer einen Buff.

Shop_Item ⟨$E160$⟩

Beziehung	Kardinalität	Erwartete Datenmenge	Beschreibung
Profile_Shop_item	0 ... *	Min: <...>, Max: <...>	Relation zwischen Shop-Items und den Profilen der Nutzer, die das Shop-Item gekauft haben.

Die Objekte des Typs „Shop_Item“ stellen die im Shop zum Kauf zur Verfügung stehenden Gegenstände dar. Daher wird unter anderem der Kaufpreis als Attribut gespeichert.

Story_Text ⟨$E170$⟩

Beziehung	Kardinalität	Erwartete Datenmenge	Beschreibung
Text_In_Challenge	1 ... *	Min: <...>, Max: <...>	Story-Texte beschreiben Challenges handlungsbezogen.
Text	0 ... *	Min: <...>, Max: <...>	Texte beschreiben die Spielsituation.

Diese Entitäten beschreiben die verschiedenen Texte, welche in der Story aufgerufen werden. Dazu werden die Bedingungen, welche erfüllt sein müssen um den Text aufzurufen, sowie die Reihenfolge, in der diese auftreten und deren Inhalt abgespeichert.

Text_In_Challenge ⟨$E180$⟩

Beziehung	Kardinalität	Erwartete Datenmenge	Beschreibung
Challenge	1	Min: <...>, Max: <...>	Verweis auf die Challenge.
Story_Text	1	Min: <...>, Max: <...>	Verweis auf den Story-Text.

An dieser Stelle wird festgehalten, welche Texte in welchem Aufgabenpaket auftreten und in welcher Reihenfolge dies passiert.

Task ⟨$E190$⟩

Beziehung	Kardinalität	Erwartete Datenmenge	Beschreibung
Potion	1	Min: <...>, Max: <...>	Eine gelöste Aufgabe schaltet einen neuen Trank frei.
Comment_Task	0 ... *	Min: <...>, Max: <...>	Kommentare der Aufgabe.
Rate_Ttask	0 ... *	Min: <...>, Max: <...>	Bewertungen der Aufgabe.
Solve_Task	0 ... *	Min: <...>, Max: <...>	Relation mit Profilen von Nutzern, die Zugang zu dieser Aufgabe haben.
Task_In_Challenge	0 ... *	Min: <...>, Max: <...>	Aufgaben können Teil einer Challenge.

Die „Task“-Entitäten beschreiben die einzelnen Aufgaben, welche in den Aufgabenpaketen enthalten sind und von den Nutzern gelöst werden müssen um neue Tränke freizuschalten. Als Attribute werden dafür unter anderem deren Schwierigkeit, die Bewertung, die Zeitbegrenzung und der dazugehörige Trank abgespeichert.

Task_In_Challenge ⟨$E200$⟩

Beziehung	Kardinalität	Erwartete Datenmenge	Beschreibung
Task	1	Min: <...>, Max: <...>	Verweis auf die Aufgabe
Challenge	1	Min: <...>, Max: <...>	Verweis auf die Challenge.

Hier wird festgehalten in welchen Aufgabenpaketen die verschiedenen Aufgaben enthalten sind und in welcher Reihenfolge diese auftreten.

Text ⟨*E*210⟩

Beziehung	**Kardinalität**	**Erwartete Datenmenge**	**Beschreibung**
Story_Text	0 ... *	Min: <...>, Max: <...>	Texte beschreiben die Spielsituation.

Die Entitäten vom Typ "Text"beschreiben die verschiedenen Texte, welche innerhalb des SQL-Alchemist auftreten.

User ⟨*E*220⟩

Beziehung	**Kardinalität**	**Erwartete Datenmenge**	**Beschreibung**
Profile	1	Min: <...>, Max: <...>	Jeder Nutzer hat ein eigenes Profil.
User_Session	0 ... *	Min: <...>, Max: <...>	Die Anzahl der User-Sessions gibt an, wie oft der einzelne Benutzer gleichzeitig im System angemeldet ist.

Die „User“-Objekte stellen die Nutzer dar, die sich bisher für das Spiel registriert haben. Dazu werden an dieser Stelle alle Nutzerdaten, die für die Erkennung und die Anmeldung von Bedeutung sind als Attribute gespeichert.

User_Session $\langle E230 \rangle$

Beziehung	Kardinalität	Erwartete Datenmenge	Beschreibung
User	1	Min: <...>, Max: <...>	Verweis auf den entsprechenden User, der die User-Session gestartet hat.

Die „User_Session“-Objekte stellen die einzelnen durch die Nutzer gestarteten Sitzungen dar. Das bedeutet, dass jedes mal, wenn sich ein Nutzer zum Spiel anmeldet wird ein neues „User_Session“-Objekt erzeugt. Als Attribute werden die Verbindungsdaten festgehalten.

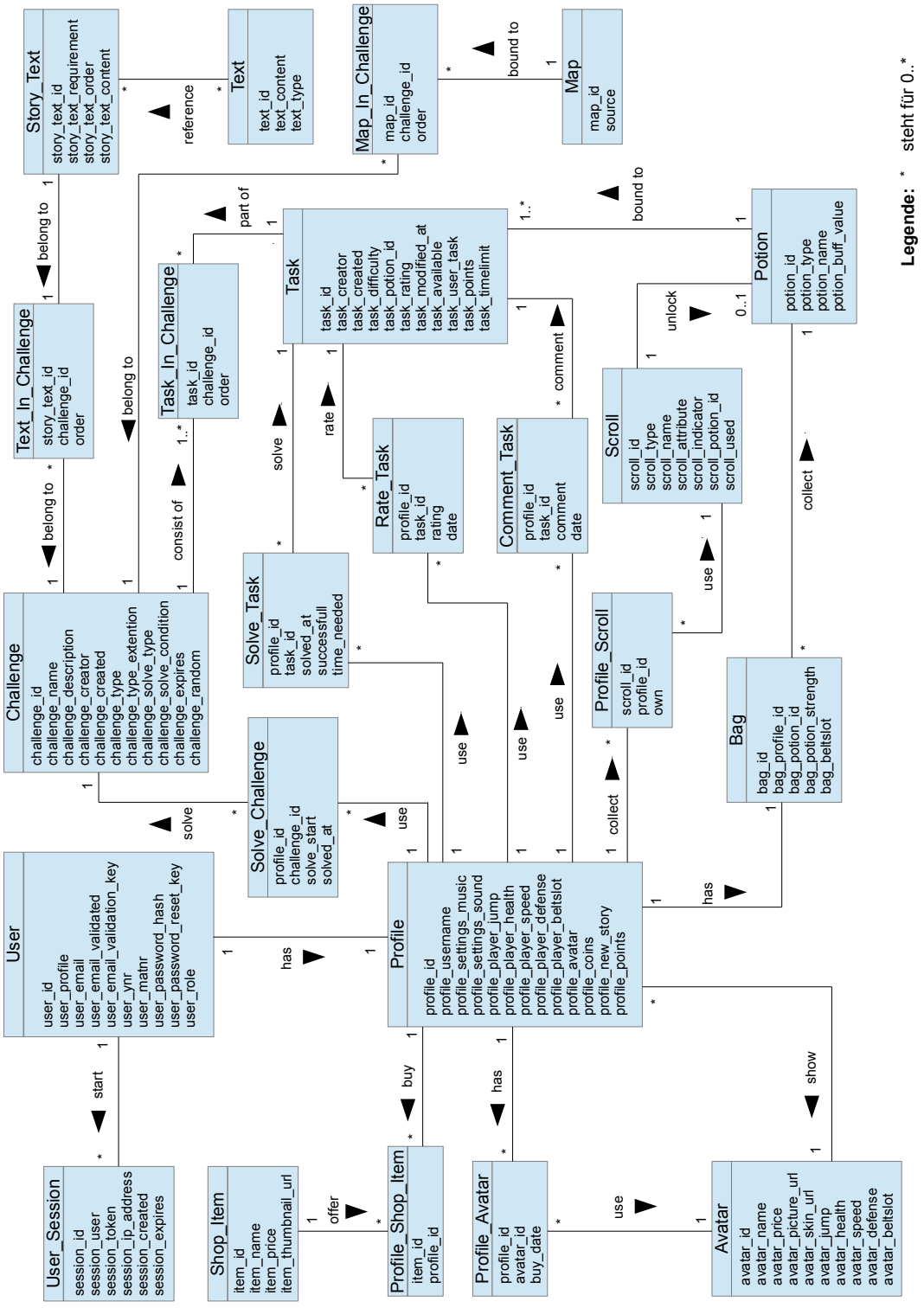

Abbildung 3.1: Klassendiagramm zum SQL-Alchemist

4 Konfiguration

Für den SQL-Alchemist benötigen wir weder einen Rechner, noch einen Server mit einer bestimmten Konfiguration. Aus diesem Grund wird an dieser Stelle nicht weiter auf diesen Punkt eingegangen.

Außerdem existieren auch keine config-Dateien auf die an dieser Stelle hingewiesen werden müsste.

5 Glossar

Lofi-Coins:
Lofi-Coins sind die Spielwährung. Diese werden im Minigame erspielt und können genutzt werden, um Avatare und andere Dinge im Ingame-Shop zuerwerben.

Playerstatistics:
Attribute des Spielers, die Auswirkungen auf das Minispiel haben (Sprunghöhe, Geschwindigkeit, Leben und Stärke).

Potions:
Die im Spiel verwendete Bezeichnung für Tränke, die dem Spieler temporäre Vorteile verschaffen. Sie werden durch das richtige Bearbeiten von SQL-Anfragen erstellt werden können.

Scrollcollection:
Beschreibt den Fortschritt des Spielers indem es die bereits gesammelten Scrolls speichert.

Scrolls:
Einmalig im Spiel im einzusammelnde Objekte. Besitzt man diese, ist man in der Lage durch das Lösen von SQL-Statements Potion zu erstellen oder die Playerstatistics dauerhaft zu erhöhen.