Technische Universität Braunschweig

ISF

Software Praktikum 2015 am wi2

Lernduell

Software-Entwicklungspraktikum (SEP)
Sommersemester 2015

Technischer Entwurf

Auftraggeber
Technische Universität Braunschweig
Institut für Wirtschaftsinformatik
Abteilung Informationsmanagement
Prof. Dr. Susanne Robra-Bissantz
Mühlenpfordtstraße 23
38106 Braunschweig

Betreuer: Michael Kallookaran

Auftragnehmer:

Name	E-Mail-Adresse
Christopher Schulz	chri.schulz@tu-braunschweig.de
Ekaterina Bazhenova	e.bazhenova@tu-braunschweig.de
Fabian Sgonina	f.sgonina@tu-braunschweig.de
Frederik Book	f.book@tu-braunschweig.de
Jannis Kottke	j.kottke@tu-braunschweig.de
Jonas Korth	jonas.korth@tu-braunschweig.de
Steffen Döring	steffen.doering@tu-braunschweig.de
Timo Christophers	t.christophers@tu-braunschweig.de

Braunschweig, 1. Juli 2015

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Aufgabe des vorliegenden Dokuments ist die fachliche Modellierung des Systems. Dabei soll mit Hilfe von verschiedenen Diagrammen ein ungefährer Überblick über das Verhalten und die anfallenden sowie verwendeten Daten des Projekts *Lernduell* verschafft werden.

Lernduell ist ein Spiel, bei dem Studenten Fragen aus verschiedenen Fächern beantworten müssen und gegen reale Gegner antreten.

Der zum Spielen benötigte Client in Form einer Android Applikation arbeitet im Zusammenhang mit einem zentralen Server, welcher für die Speicherung der Spiel- und Spielerdaten, sowie das Match-Making zuständig ist.

Mit Hilfe des Aktivitätsdiagramms 1.1 wird die Struktur von *Lernduell* genauer beschrieben:

Nach dem starten der Applikation möchte der Nutzer sich anmelden. Falls die Anmeldung erfolgreich ist, ist der Benutzer eingeloggt und wird direkt auf den Homescreen geleitet. Ist die Anmeldung nicht erfolgreich (wenn der User z.B. seine Daten falsch eingegeben, oder sich noch nicht registriert hat) muss er die Anmeldung erneut durchführen. Auf dem Homescreen hat der Nutzer drei weitere Möglichkeiten: Entweder eine neue Frage zu erstellen, ein neues Spiel zu starten oder den Spielprozess zu beenden. Falls er sich dazu entscheidet eine Frage zu erstellen, muss er zunächst ein Fach auswählen und dazu eine Frage sowie vier Antwortmöglichkeiten eingeben und abschließend die neue Frage speichern. Danach wird der Benutzer zurück auf den Homescreen geleitet. Möchte der Nutzer ein neues Spiel starten, muss er als erstes auswählen, in welchem Fach er spielen möchte. Als nächstes kann er entweder gegen einen bestimmten Nutzer antreten oder „Random“ auswählen und auf eine Einladung zum Spiel gegen einen zufälligen Gegner warten. In diesem Fall sucht *Lernduell* einen Gegner aus dem Pool der Mitspieler. Danach startet das Spiel direkt mit einem dreirundigen Duell. Am Ende des Spiels sieht sowohl der User als auch sein Gegner die Ergebnisse mit den richtigen Antworten in einer Übersicht. Abschließend leitet das System den Spieler wieder zurück auf den Homescreen der Applikation.

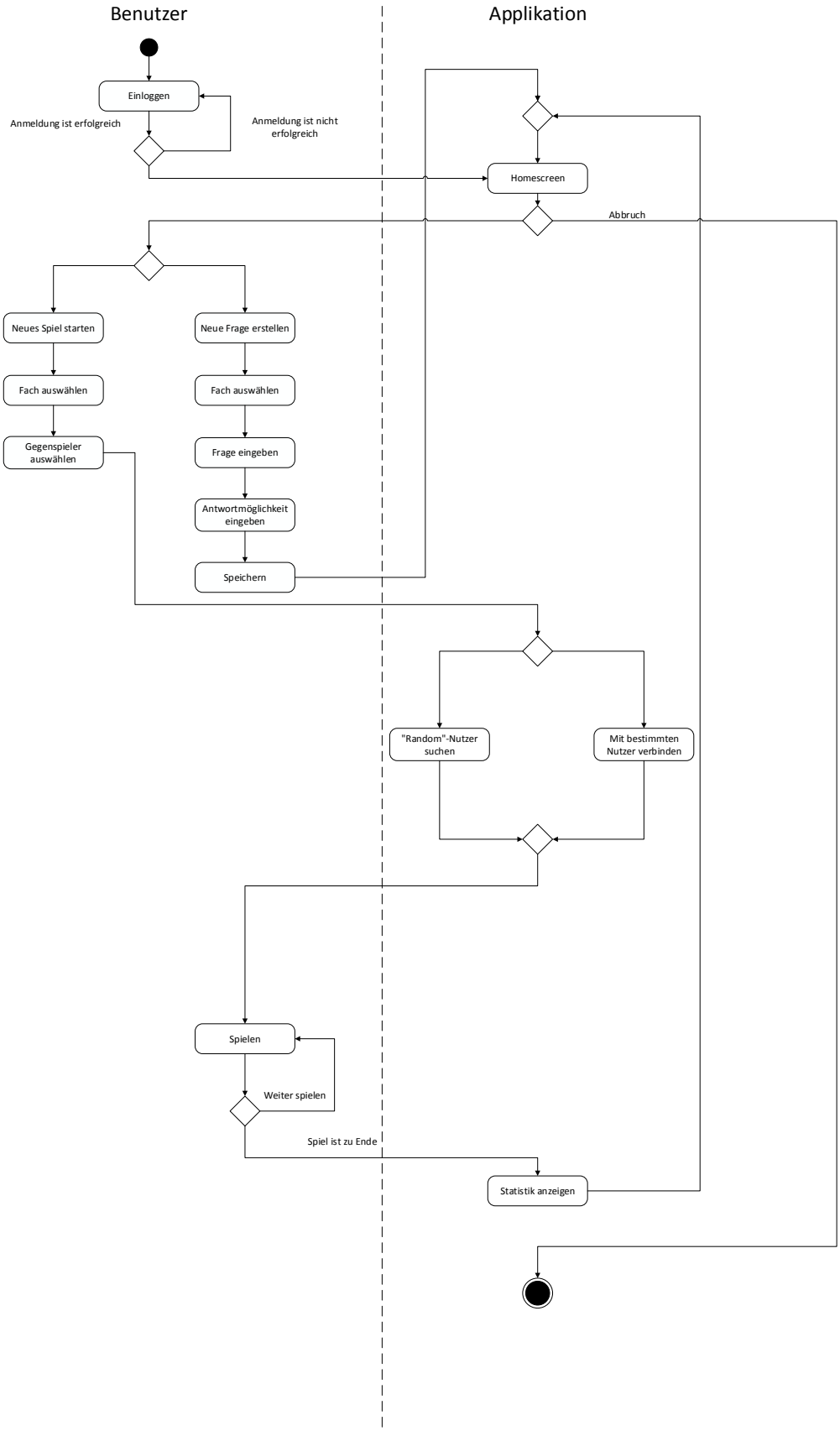

Abbildung 1.1: Aktivitätsdiagramm *User spielt Lernduell*

1.1 Projektdetails

Um die in der Einleitung gezeigte Struktur von Lernduell noch detaillierter aufzuzeigen, wird in diesem Abschnitt noch genauer auf den Spielprozess eingegangen und anhand eines Aktivitätsdiagramms visualisiert.

Als Beispiel dient die Interaktion zwischen einem Nutzer und seinem zufällig ausgewählten Gegenspieler, welche der Abbildung 1.2 entnommen werden kann:

Zu Beginn melden sich beide Benutzer an, wobei die Anmeldedaten vom Client an den Server gesendet und überprüft werden. Da die Daten beider Nutzer korrekt sind, wird der Nutzer eingeloggt auf den Homescreen geleitet. Nachdem beide Spieler ein Fach ausgewählt und den „Random“-Button gedrückt haben, wird vom Server ein neues Spiel erstellt indem die Spieler vom System als Gegner ausgewählt werden. Von der Serverseite werden Fragen verschickt, die von den Spielern beantwortet werden müssen. Wenn das Spiel zu Ende ist, werden die Spielergebnisse in der Datenbank des Servers gespeichert. Das System zeigt den Nutzern ihre Ergebnisse an. Falls der Benutzer mehr Fragen als sein Gegner richtig beantwortet hat, wird eine Siegbenachrichtigung angezeigt, falls der Gegner gewinnt, eine Niederlageninfo.

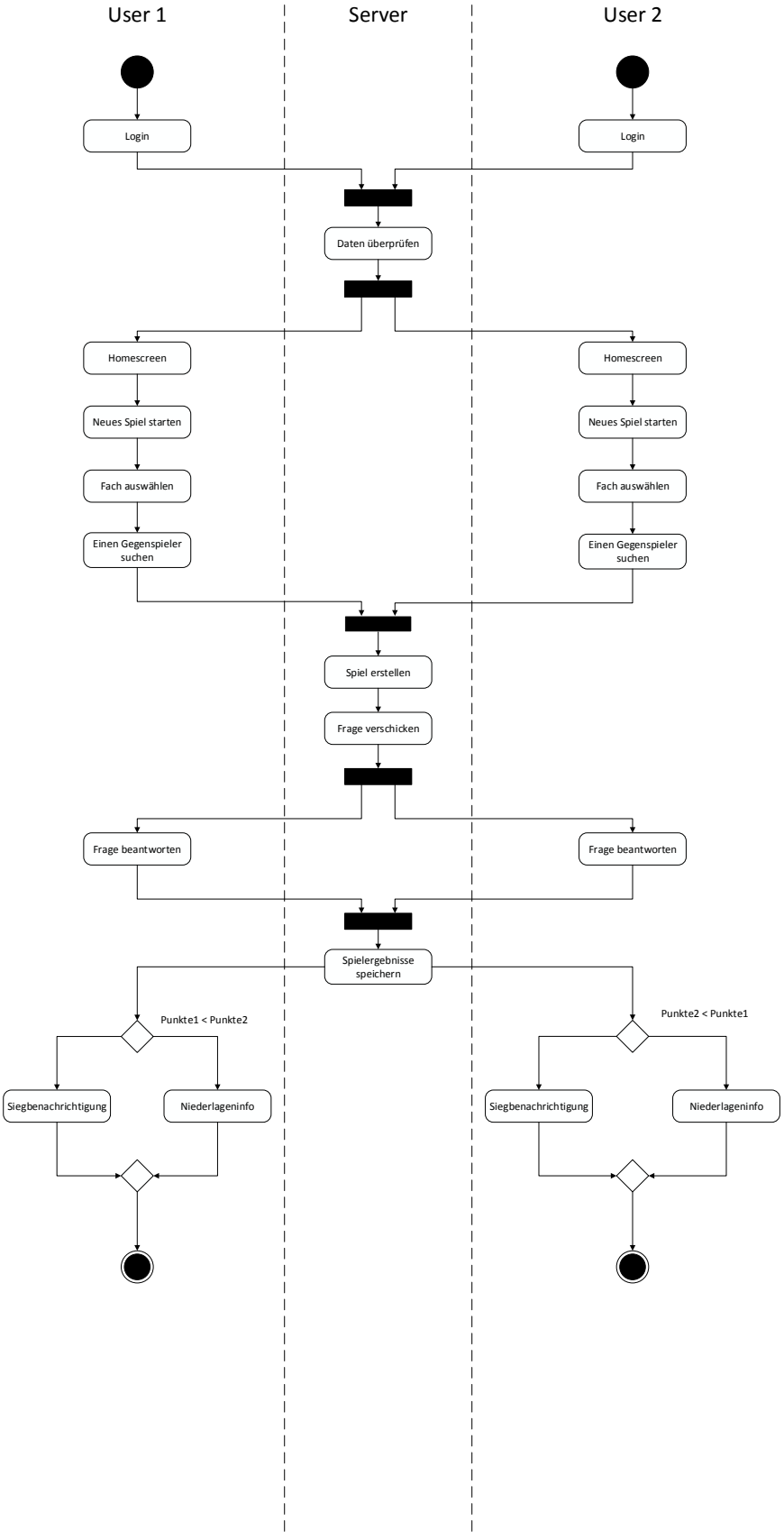

Abbildung 1.2: Aktivitätsdiagramm *Spiel zwischen zwei zufällig ausgewählten Usern*

2 Analyse der Produktfunktionen

In diesem Kapitel werden die im Pflichtenheft vorgestellten Produktfunktionen analysiert und in Sequenzdiagrammen dargestellt. So wird gut erkennbar, welche Nachrichten die einzelnen Komponenten von *Lernduell* austauschen und welche Funktionen aufgerufen werden um die jeweilige Funktion zu realisieren. Dies stellt insgesamt die Basis dar, um später eine geeignete Architektur festlegen zu können.

2.1 Analyse von Funktionalität F10: Registrierung eines Benutzers

Die Funktion *Registrieren eines Benutzers* ist notwendig um die App nutzen zu können, muss dafür aber auch nur einmal pro Benutzer durchgeführt werden. Denn mit dieser Funktion erstellt der Benutzer ein Nutzerkonto. Im folgenden Sequenzdiagramm in Abbildung 2.1 werden nun die Komponenten dargestellt, die an diesem Prozess beteiligt sind. Um sich zu registrieren, muss ein Benutzer zuerst die Applikation starten. Als nächstes gibt der Benutzer auf dem Mobilgerät einen Benutzernamen und ein Passwort an. Diese Daten, sowie aus der ComboBox ausgewählte Universität, werden dann an den Server übermittelt, welcher die Daten in die Benutzertabelle der Datenbank einträgt. Dabei kann eine ConnectionsException ausgelöst werden, wenn es Verbindungsprobleme geben sollten. Das Ergebnis des Servers kann entweder eine Erfolgsnachricht sein, oder ein Fehler, falls der Benutzername in der Datenbank vorhanden ist. Im letzteren Fall wird eine NameAlreadyUsedException erstellt.

2.2 Analyse von Funktionalität F20: Client Login

In dem Sequenzdiagramm in Abbildung 2.2 wird die Funktionalität *Client Login* näher dargestellt. Diese Funktion wird benötigt, damit der Nutzer auf sein vorher erstelltes Nutzerkonto zugreifen kann, um die Funktionen von *Lernduell* verwenden zu können. In der Abbildung gibt der Benutzer als Erstes die benötigten Daten, also seinen Benutzernamen und sein Passwort, in der Lernduell-GUI an. Diese Daten wiederum werden an den Server übermittelt und mit der Benutzerdaten in der Datenbank abgeglichen. Wenn der Benutzer bereits in der Datenbank registriert ist, loggt er sich erfolgreich ein. Wenn keine entsprechenden Benutzerdaten in der Datenbank gefunden wurden, wird ein Fehler ausgegeben und die Eingabe beginnt von vorne. Ist der Login korrekt, wird mithilfe des Session Managements eine Session ID generiert und der

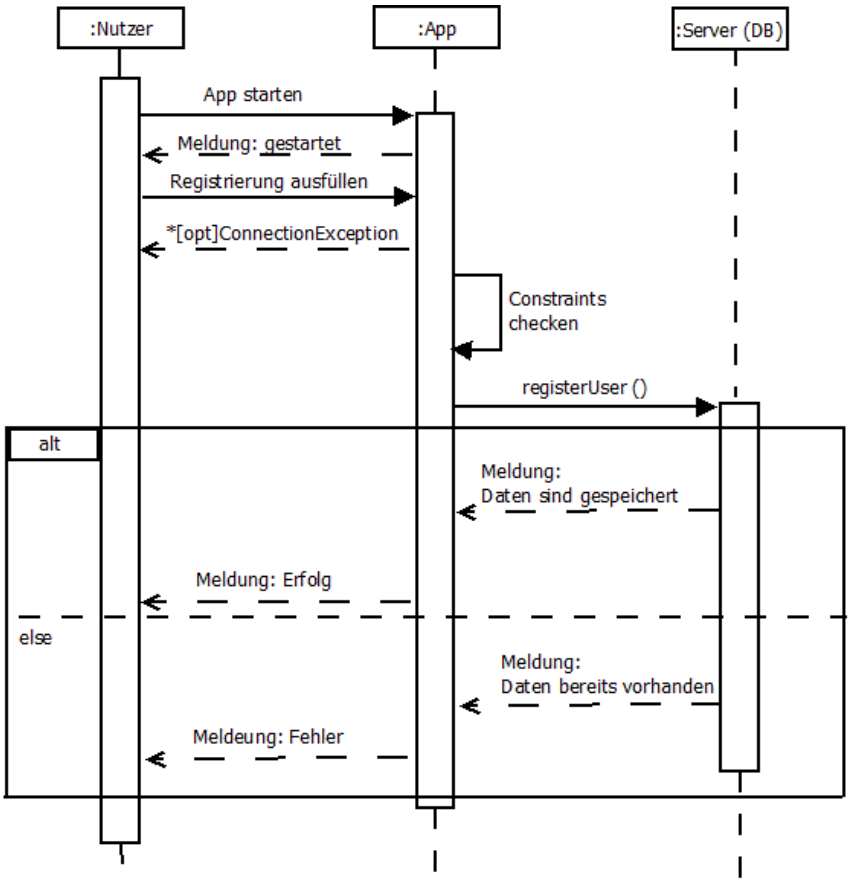

Abbildung 2.1: Sequenzdiagramm **F10** *Registrierung eines Benutzers*

App als Antwort übergeben. Ist der Login inkorrekt wird keine ID generiert und eine negative Session ID zurückgeliefert.

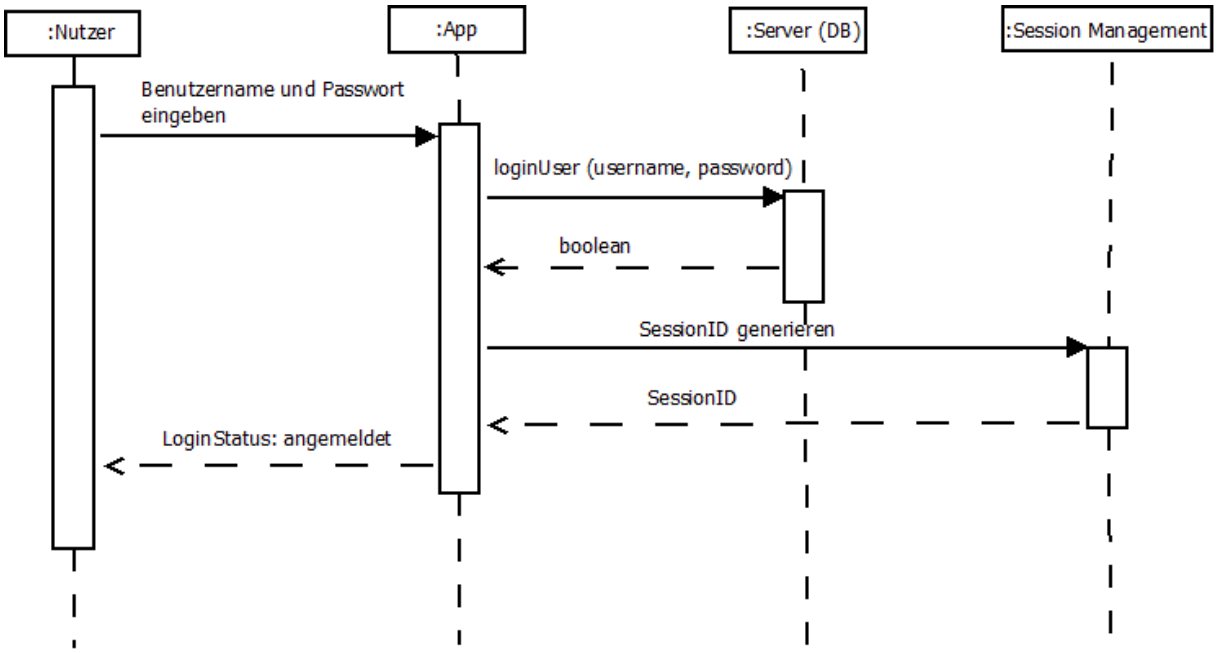

Abbildung 2.2: Sequenzdiagramm **F20** *Client Login*

2.3 Analyse von Funktionalität F30: Spiel gegen einen zufälligen Benutzer

Spiel gegen einen zufälligen Benutzer ist eine Funktionalität, die die Hauptaufgabe von Lernduell erweitert, um dem Nutzer mehr Vielfalt zu bieten. In dem folgenden Sequenzdiagramm in Abbildung 2.3 wird die Funktion bildlich dargestellt. Zunächst ist in der Abbildung dargestellt, wie der Nutzer in der Lernduell-GUI ein neues Spiel startet. Die App zeigt das Dropdown-Menü mit den Fächern. Der Benutzer wählt aus der Liste ein Fach aus und klickt auf den Button „Random-Gegner“. In der Datenbank wird das neue Spiel mit Spielerdaten erzeugt und der Server startet einen zufälligen Gegner zu suchen. Solange dem Benutzer kein Gegner zugeordnet wurde, kann er nur eine Runde spielen. Nachdem die letzte Frage der ersten Runde beantwortet wurde, soll der Benutzer auf den Gegner warten. Das System wählt den Gegenspieler aus, nachdem er sich angemeldet, das gleiche Fach ausgewählt und auf den Button „Random-Gegner“ geklickt hat. Die Lernduell-GUI wird dann vom Server benachrichtigt. Die App leitet den Benutzer in die nächste Runde und seinen Gegner – in die erste Runde weiter.

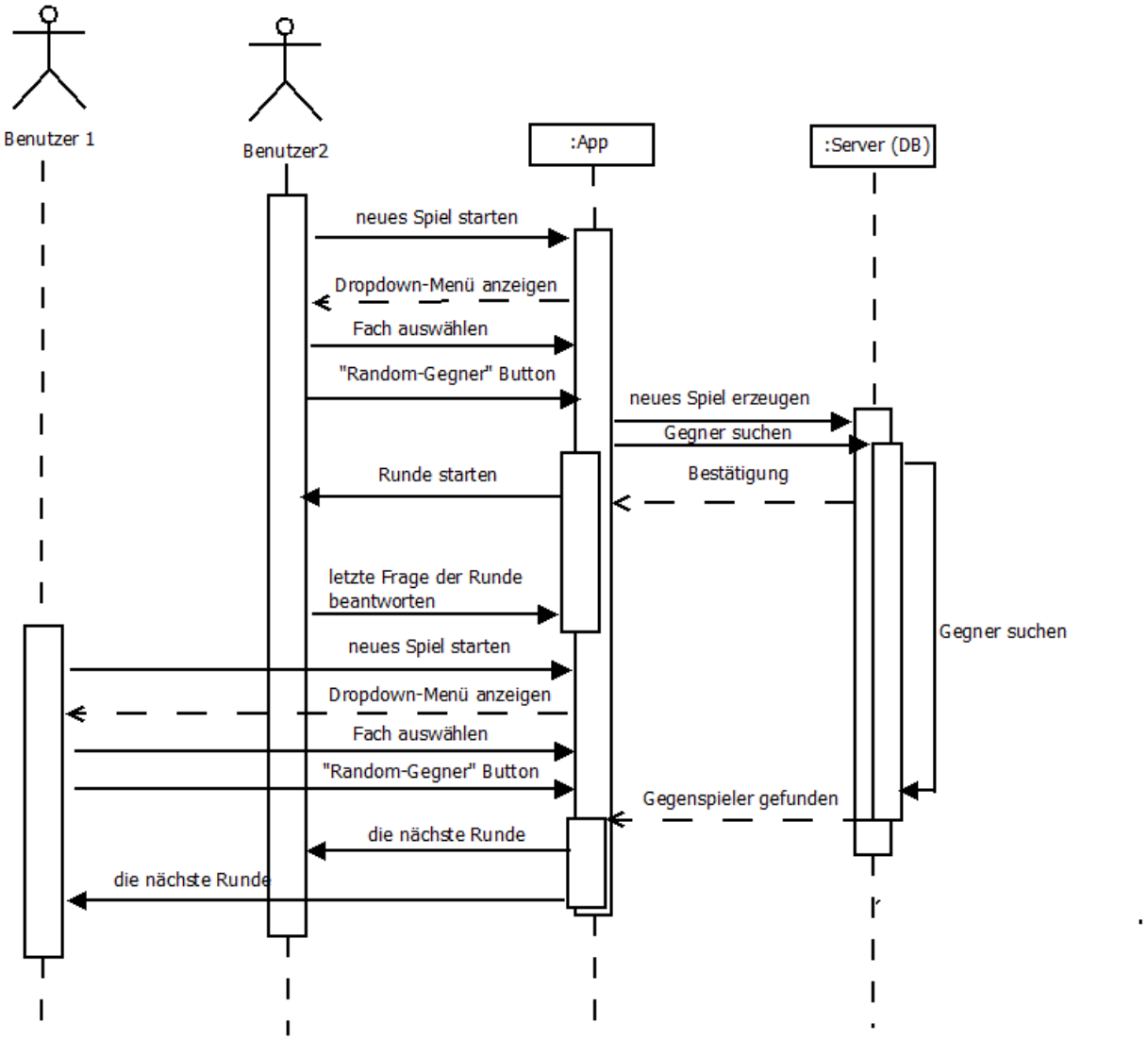

Abbildung 2.3: Sequenzdiagramm **F30** *Spiel gegen einen zufälligen Benutzer*

2.4 Analyse von Funktionalität F40: Herausfordern eines Gegners

Die Funktion *Herausfordern eines Gegners* bietet die Möglichkeit, einen bestimmten Benutzer zum Spiel einladen. Dies wird in Abbildung 2.4 in dem Sequenzdiagramm verdeutlicht. Der bereits angemeldete Benutzer startet ein neues Spiel und fordert einen bestimmten Gegner heraus. Dafür wählt er ein Fach aus, gibt den Benutzernamen des Gegners an und klickt auf den Button „Suche“. Die App sendet die Anfrage an den Server und der Server startet die Suche. Nachdem der Benutzer gefunden wurde, sendet der Server eine Bestätigung an die App, die als nächstes den Benutzer darüber informiert. Der Benutzer lädt den Gegner zum Spiel ein, die App versendet die Einladung.

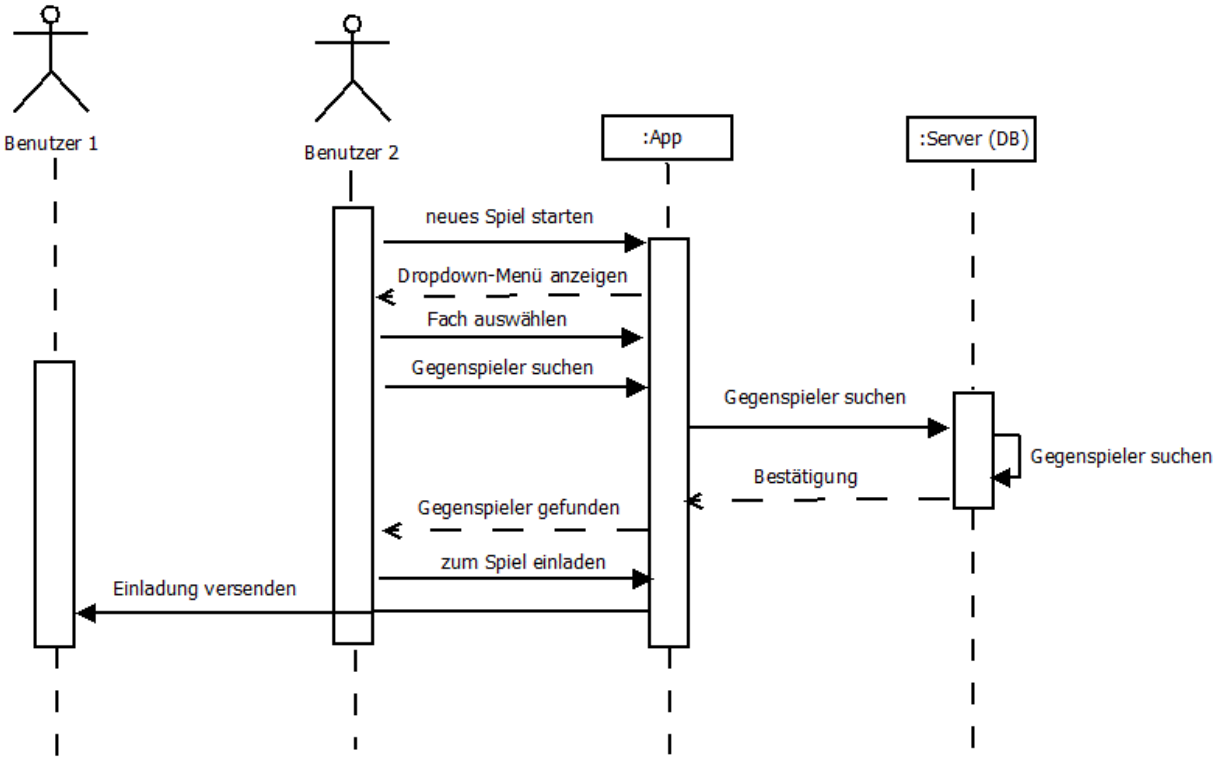

Abbildung 2.4: Sequenzdiagramm **F40** *Herausfordern eines Gegners*

2.5 Analyse von Funktionalität F50: Herausforderung eines Gegners annehmen

Das Sequenzdiagramm in der Abbildung 2.5 zeigt die Interaktion des Spielers beim Annehmen einer Herausforderung. Der Benutzer ist angemeldet und erhält von einem anderen Benutzer die Einladung zum Spiel in einem bestimmten Fach. Um das Spiel zu starten, klickt er auf den Button „Herausforderung annehmen“. Sein Gegner wird über Lernduell-GUI informiert,

dass Herausforderung angenommen wurde. In der Datenbank wird automatisch ein neues Spiel erzeugt, dabei werden Spiel ID, Benutzer ID1, Benutzer ID2 in die Tabelle „Spiele“ eingetragen. Das System startet die erste Spielrunde und informiert beide Benutzer darüber über GUI.

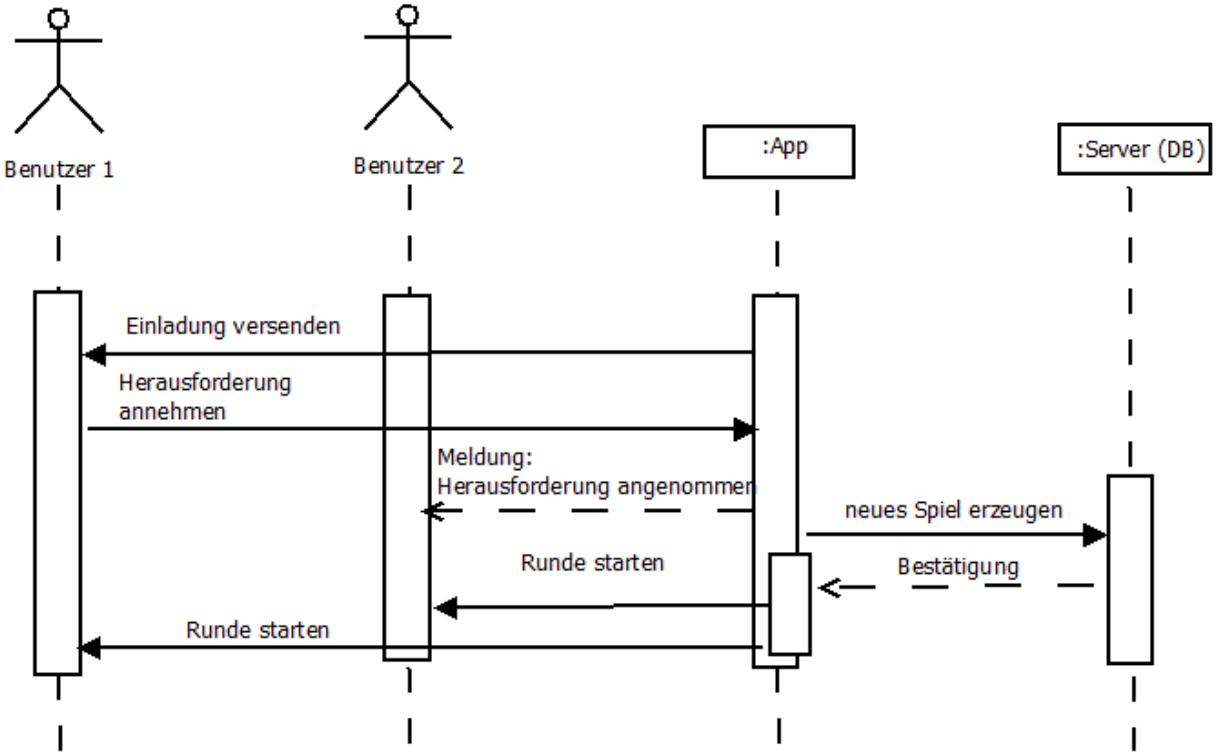

Abbildung 2.5: Sequenzdiagramm **F50** *Herausforderung eines Gegners annehmen*

2.6 Analyse von Funktionalität F60: Herausforderung eines Gegners ablehnen

Der Benutzer kann die erhaltene Herausforderung ablehnen. Im Sequenzdiagramm, in der Abbildung 2.6, wird näher auf die Funktion eingegangen. Nachdem der Benutzer eine Einladung zum Spiel in einem bestimmten Fach erhalten hat, klickt er auf den Button „Herausforderung ablehnen“. Sein Gegner wird darüber über Lernduell-GUI informiert. Anschließend wird dem Benutzer das Hauptmenü von *Lernduell* angezeigt.

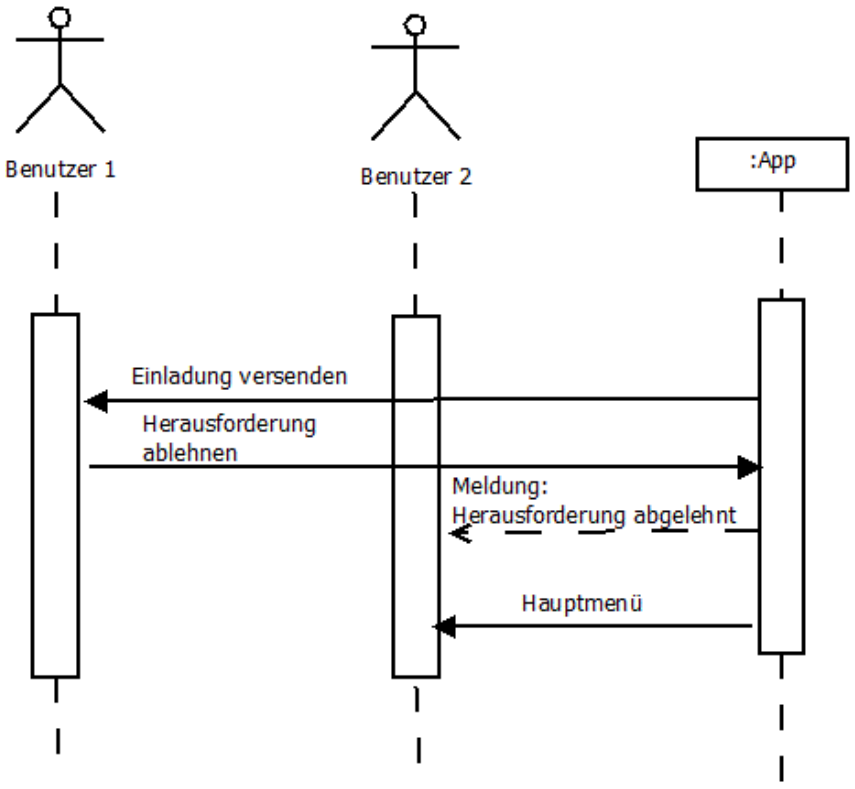

Abbildung 2.6: Sequenzdiagramm **F60** *Herausforderung eines Gegners ablehnen*

2.7 Analyse von Funktionalität F70: Spiel abbrechen

Spiel abbrechen ist eine Funktion, welche *Lernduell* erweitert. Sie soll dem Nutzer die Möglichkeit geben, das bereits angefangene Spiel gegen einen anderen Benutzer abzubrechen. Im Sequenzdiagramm in Abbildung 2.7 wird diese Funktion bildlich dargestellt. In der Abbildung wird dargestellt, dass der Benutzer auf den Button „Spiel abbrechen“ in der *Lernduell*-GUI klickt, welche anschließend dem Gegenspieler eine Nachricht anzeigt, dass das Spiel vom Gegner abgebrochen wurde. Gleichzeitigt wird der Server benachrichtigt, dass das angefangene Spiel aus der Datenbank gelöscht werden soll. Die App-GUI springt automatisch in das Hauptmenü.

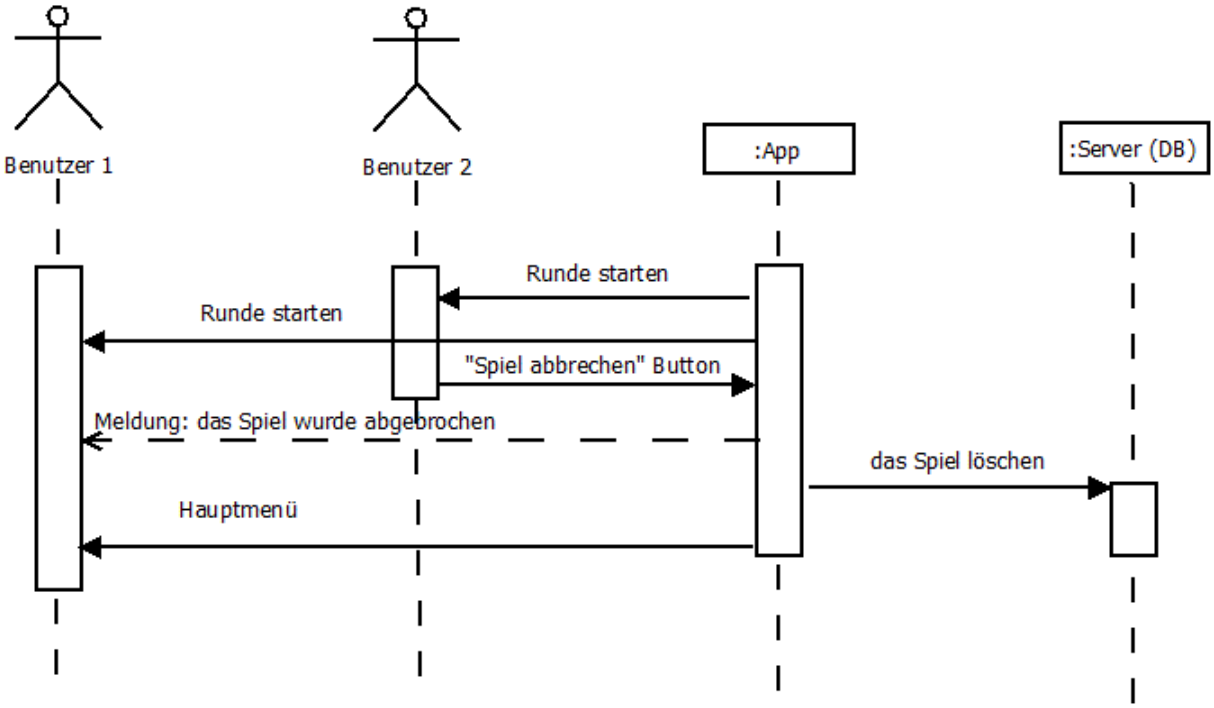

Abbildung 2.7: Sequenzdiagramm **F70** *Spiel abbrechen*

2.8 Analyse von Funktionalität F80: Abrufen der Statistik

Im Sequenzdiagramm in Abbildung 2.8 wird die Funktionalität *Abrufen der Statistik* genauer dargestellt. Diese Funktion wird benötigt, damit der Nutzer seine Spielergebnisse ansehen kann. In der Abbildung wird dargestellt, dass der Benutzer auf den „Statistik“-Button in der *Lernduell*-GUI klickt, wenn das Spiel beendet wurde. Die App sendet eine Anfrage an den Server, welcher anschließend dem Benutzer die Ergebnisse seiner letzten fünf gespielten Spiele anzeigt.

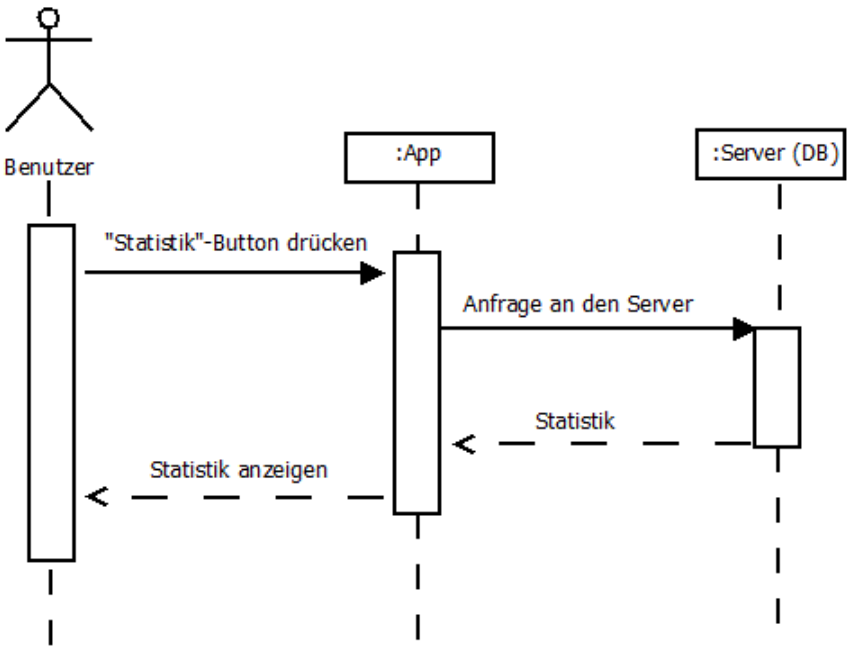

Abbildung 2.8: Sequenzdiagramm **F80** *Abrufen der Statistik*

2.9 Analyse von Funktionalität F90: Erstellen einer neuen Frage

Der Benutzer hat die Möglichkeit den Fragenkatalog durch neue Fragen zu erweitern. Diese Funktion ist in Abbildung 2.9 dargestellt. Dazu klickt er auf den Button „Neue Frage hinzufügen“. Die App zeigt die Fächer im Dropdown-Menü. Nachdem der Benutzer ein Fach aus der Liste ausgewählt hat, wird das Eingabefeld von der App angezeigt, hier kann der Benutzer seine Frage, vier Antwortmöglichkeiten dazu angeben und korrekte Antwort markieren. Die App überträgt Informationen an die Datenbank und der Server sendet eine Bestätigung, dass die Daten erfolgreich gespeichert wurden. Die App-GUI benachrichtigt den Benutzer und springt automatisch in das Hauptmenü.

2.10 Analyse von Funktionalität F100: Melden einer veralteten oder fehlerhaften Frage

Lernduell bietet dem Nutzer die Möglichkeit, veraltete oder fehlerhafte Fragen aus dem Fragenkatalog zu melden. Im folgenden Sequenzdiagramm in Abbildung 2.10 wird diese Funktion bildlich erläutert. Ein angemeldeter Spieler findet während eines laufenden Spiels eine fehlerhafte oder veraltete Frage. Im nächsten Schritt beantwortet er die Frage und klickt auf den Button „Frage markieren“. Die Applikation erhält Informationen vom Benutzer, welche Frage

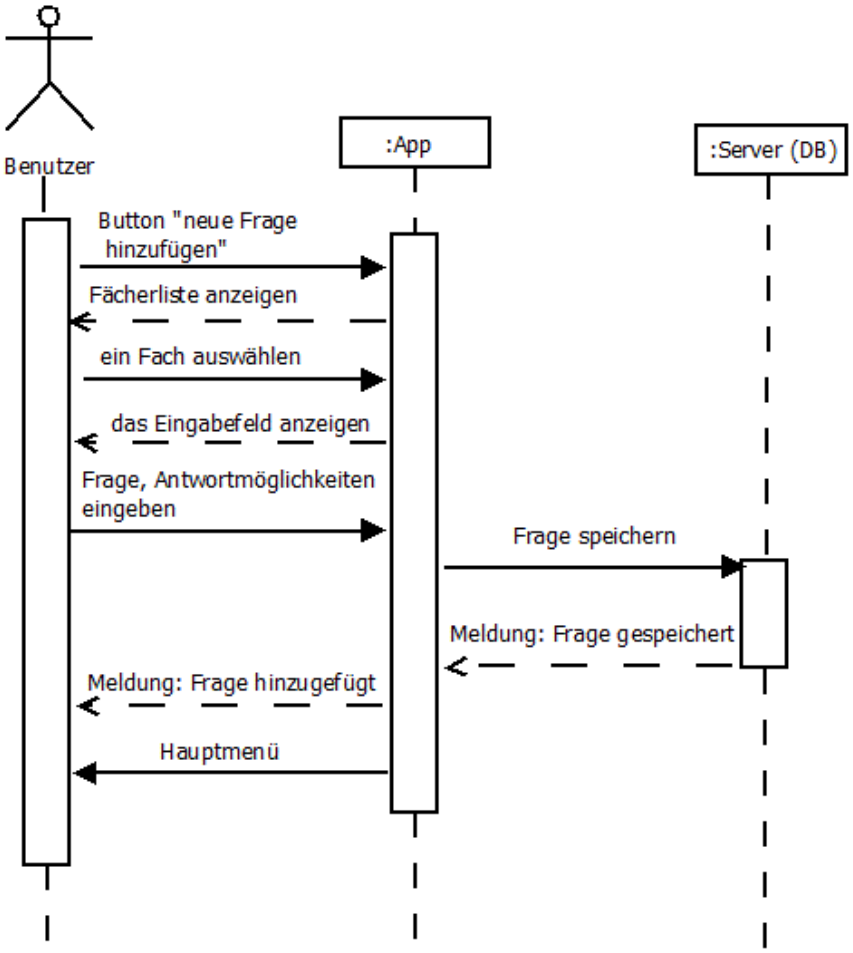

Abbildung 2.9: Sequenzdiagramm **F90** *Erstellen einer neuen Frage*

markiert wurde, und sendet dementsprechend die Frage an den Server, welcher sie in der Datenbank temporär speichert. Wenn das Spiel beendet wurde, fordert das System aus der Datenbank Informationen zu temporär gespeicherten Fragen an und leitet den Benutzer zurück zu diesen Fragen. Der Benutzer meldet die Frage, der Server speichert die gemeldete Frage. Anschließend wird dem Benutzer das Hauptmenü angezeigt. Die gemeldete Frage wird von dem Server an Administrator übermittelt. Der Administrator prüft die Frage und, falls die fehlerhaft ist, entfernt er die Frage aus dem Fragenkatalog.

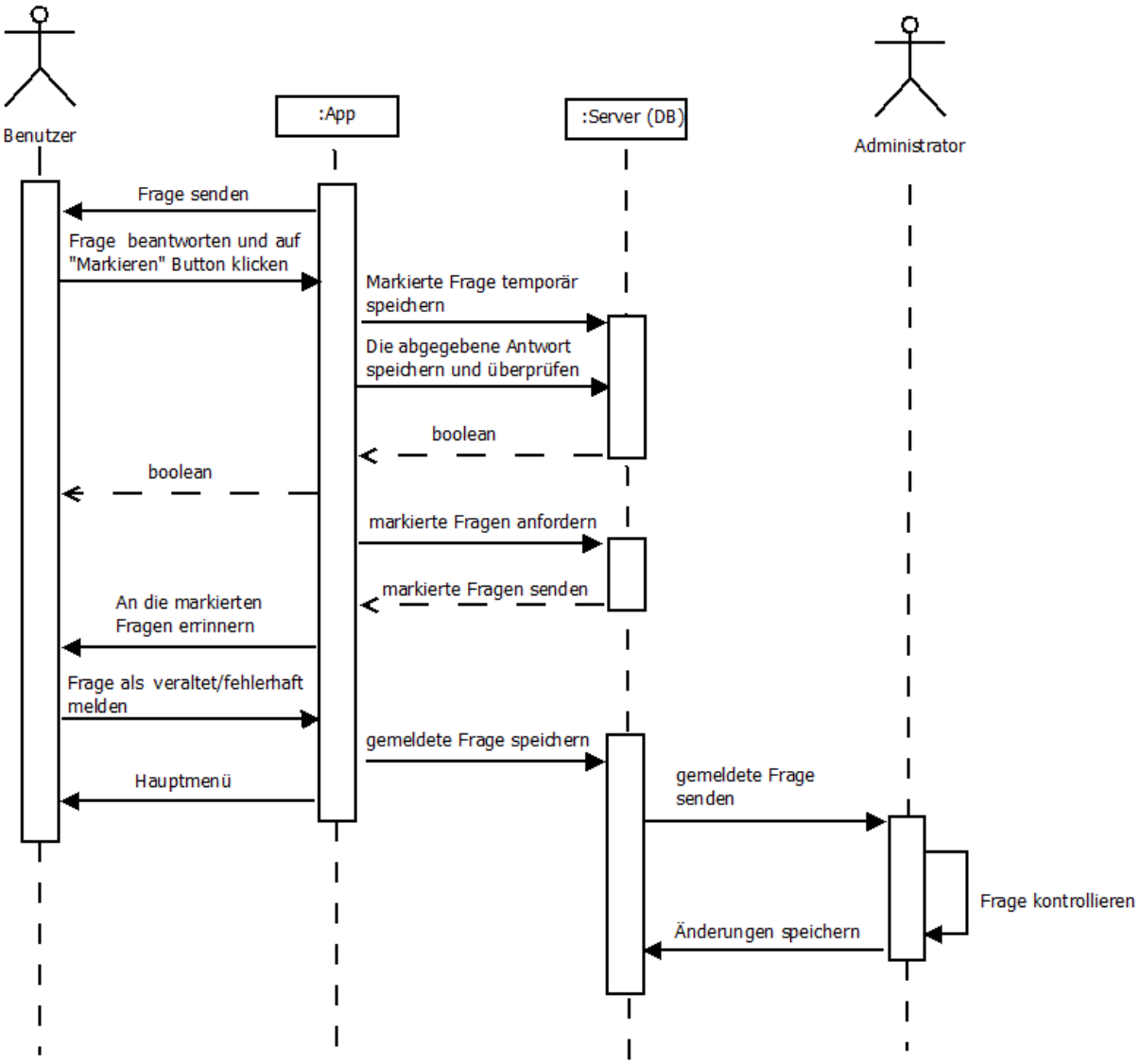

Abbildung 2.10: Sequenzdiagramm **F100** *Melden einer veralteten oder fehlerhaften Frage*

2.11 Analyse von Funktionalität F110: Aktualisieren des digitalen Fragenkataloges

Im Sequenzdiagramm in Abbildung 2.11 wird die Administratorfunktion dargestellt. Nachdem neu erstellte Fragen in der Datenbank gespeichert wurden, benachrichtige der Server den Administrator, welcher die Fragen kontrolliert. Der Administrator löscht fehlerhafte Fragen und schaltet fehlerfreie Fragen frei.

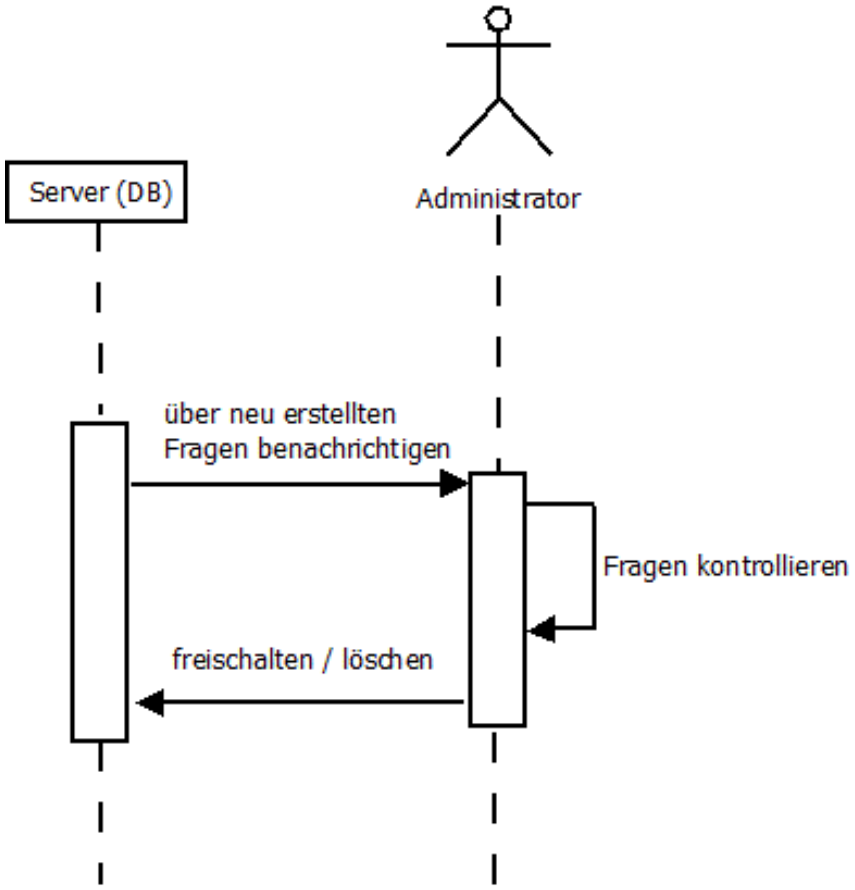

Abbildung 2.11: Sequenzdiagramm **F110** *Aktualisieren des digitalen Fragenkataloges*

2.12 Analyse von Funktionalität F120: Spielen

Spielen ist eine Kernfunktion der Anwendung *Lernduell*. Diese Funktion bietet dem Nutzer die Möglichkeit die Fragen aus verschiedenen Fächern in Spielform zu beantworten. Die Funktion und die beteiligten Komponenten werden im Sequenzdiagramm in Abbildung 2.12 dargestellt. Nachdem eine Runde gestartet wurde, fordert die App eine Frage aus der Datenbank. Nun über-

gibt das System die ausgewählte Frage an die *Lerduell*-GUI. Diese wiederum ist dafür zuständig, die Frage dem Nutzer anzuzeigen. Gleichzeitig startet Countdown Timer, der 20 Sekunden zählen soll. Beantwortet der Benutzer die Frage, soll der Timer stoppen. Die App leitet die abgegebene Antwort an den Server weiter, welcher die Antwort überprüft und in die Spieltabelle einträgt, ob die Antwort falsch oder richtig beantwortet wurde. Das Ergebnis wird an die App weitergegeben, von wo aus es dann dem Benutzer angezeigt wird.

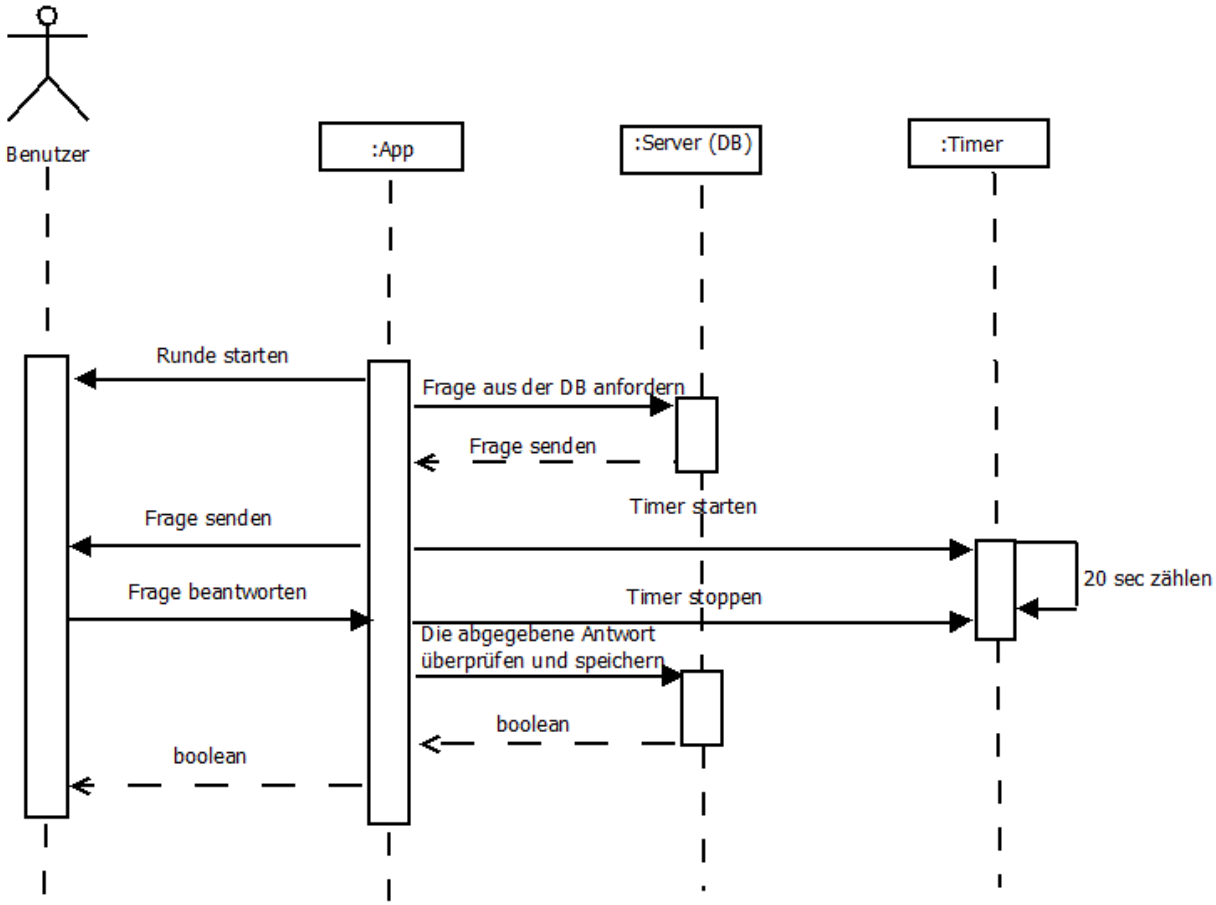

Abbildung 2.12: Sequenzdiagramm **F120** *Spielen*

3 Resultierende Softwarearchitektur

Die App *Lernduell* wird durch eine Vielzahl von Subsystemen, die durch einzelne Komponenten realisiert werden, implementiert. Die Systemarchitektur der App ist eine klassische Client-Server Architektur. Sämtliche rechenaufwändige Operationen werden dementsprechend auf dem Server ausgeführt, während der Client ermöglicht dem Nutzer die App zu bedienen. Die für den Nutzer sichtbare Komponente ist die grafische Oberfläche des Clients. Auf der GUI des Clients ist es ihm möglich, sein persönliches Nutzerprofil zu erstellen, gegen einen User zu spielen, neue Fragen zu erstellen. Die Architektur des Servers ist in drei Komponenten geteilt. Die Kernelemente für die Datenspeicherung ist der Database Handler, welcher die an den Server zu übertragenden Daten definiert, um diese dann in der Datenbank speichern zu lassen. Der Database Handler übergibt die Daten an den Background Networker. Der Server Connector soll die Verbindung herzustellen. Danach werden die Json-Daten im JsonParser verarbeitet und für die weitere Appnutzung bereitgestellt. Die Komponente DataStorage symbolisiert die Datenbank, welche User- und Spieldaten verwaltet und bereitstellt. Ein genauerer Einblick in dieses System folgt in dem nächsten Abschnitt.

3.1 Komponentenspezifikation

Im Folgenden wird die Komponentenstruktur von *Lernduell* näher beschrieben. Das Komponentendiagramm in der 3.1 liefert einen Überblick über die Komponenten des Systems: Client, Database Handler, Background Networker, Server Connector und DataStorage

Die einzelnen Komponenten werden nun kurz beschrieben:

Komponente $\langle C10 \rangle$: Client
Der Client beschreibt ein Android-Mobilgerät und hat die Aufgabe, eine Verbindung zum Nutzer herzustellen. Die von dem Client zur Verfügung gestellte grafische Oberfläche visualisiert für den Spieler den Ablauf des Spiels und ermöglicht Eingaben wie z.B. Nutzerdaten und neue Fragen für den digitalen Fragenkatalog. Der Client kommuniziert mit der Komponente $\langle C20 \rangle$, um Daten an den Server zu übertragen, und mit der Komponente $\langle C50 \rangle$, um bei der Anmeldung eingegeben Nutzerdaten zu überprüfen.

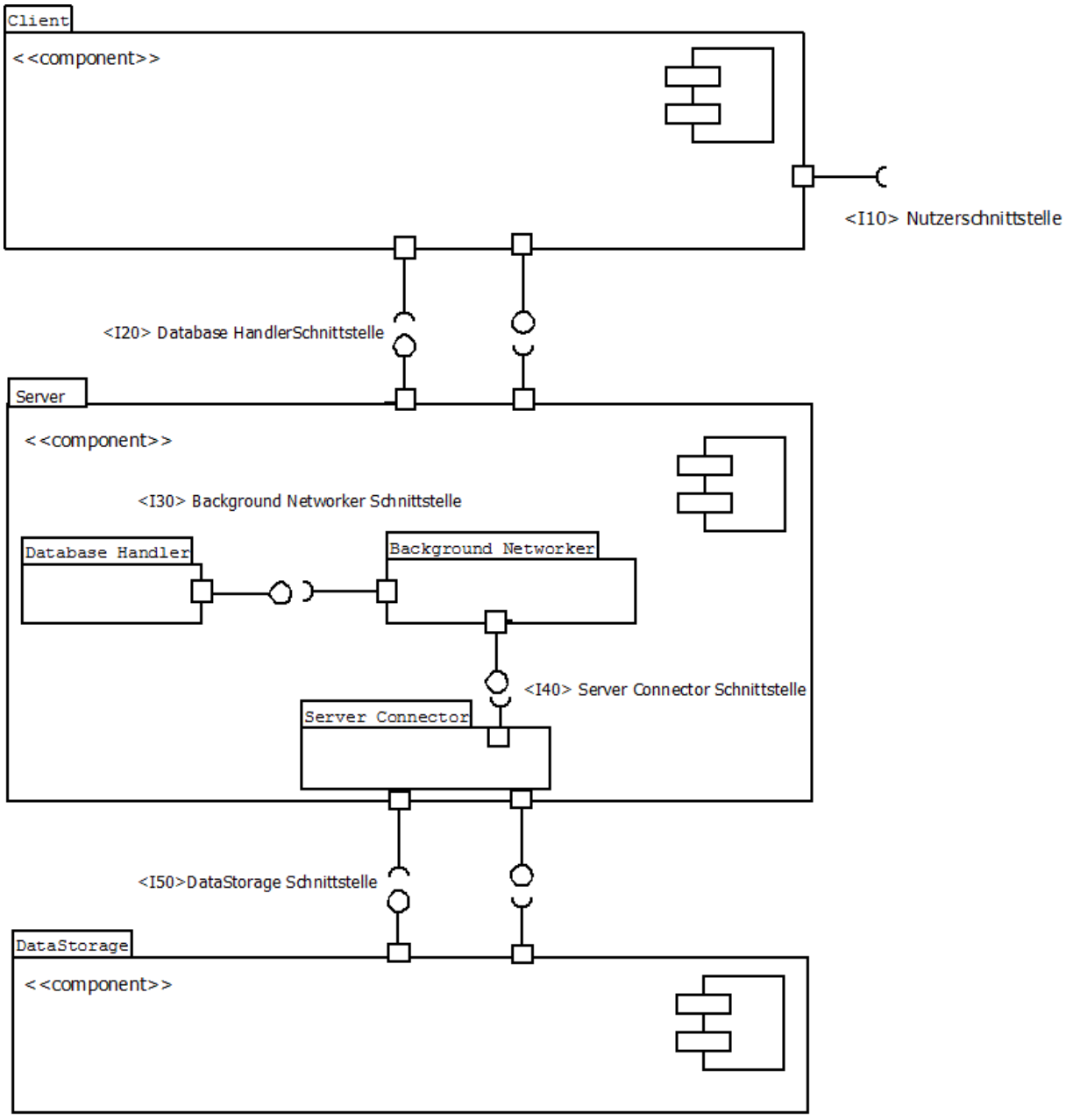

Abbildung 3.1: Komponentendiagramm

Komponente $\langle C20 \rangle$: Database Handler
Der DataBase Handler definiert dabei die an den Server zu übertragenden Daten. Anschließend werden die vorbereiteten Daten an den Background Networker übergeben.

Komponente $\langle C30 \rangle$: Background Networker
Der Background Networker dient lediglich dazu den Server Connector in einem anderen Thread auszuführen, da es in Java nicht möglich ist, Netzwerkanfragen auf dem Mainthread auszuführen.

Komponente $\langle C40 \rangle$: Server Connector
Im Server Connector findet schließlich die eigentliche Verbindungsherstellung per POST-Methode statt. Dabei wird ein vorgefertigtes php-Skript mit den gegebenen Daten auf dem Server ausgeführt. Die Daten werden im Json-Format bezogen und an den JsonParser gegeben.

Komponente $\langle C50 \rangle$: DataStorage
Der DataStorage übernimmt die gesamte Speicherverwaltung des Lernduells. Er speichert sämtliche Daten, die in den einzelnen Komponenten des Systems entstanden sind, ab und stellt diese zu einem späteren Zeitpunkt zur Weiterverarbeitung zur Verfügung. Somit werden im Storage Nutzerdaten, Statistiken, Spieldaten, Fächerdaten, der digitale Fragenkatalog verwaltet.

3.2 Schnittstellenspezifikation

Im Folgenden werden die einzelnen Schnittstellen der Komponenten aus der Komponentenspezifikation näher erläutert:

Schnittstelle $\langle I10 \rangle$: Nuzerschnittstelle

Operation	**Beschreibung**
Input ()	Der Nutzer gibt über die Tastatur Eingaben an die App-GUI. Die App arbeitet dann mit diesen Daten weiter.

Schnittstelle $\langle I20 \rangle$: Database Handler Schnittstelle

Operation	Beschreibung
RegisterRequest ()	Mobilgerät überträgt Benutzername, E-Mail-Adresse, Passwort und Universität an den Database Handler, sodass ein neuer User erstellt werden kann.
FindGameRequest ()	Die Anfrage an den Server, ein neues Spiel zu finden. Übermittelt die Nutzerdaten (Universität, Fach, Username) an Database Handler.
LoginRequest ()	Mobilgerät überträgt Login-Daten (Benutzername, Passwort) zur Authentizierungan den Database Handler.
GetCourseRequest ()	Die Anfrage an den Server, Fächer einer Universität aus der Datenbank auszulesen. Übermittelt die Daten (Universität) an Database Handler.
SaveResultRequest ()	Die Anfrage an den Server, Ergebnisse eines Spieles zu speichern. Übermittelt die Spieldaten (Spiel ID, Username, Fach, Universität, beantwortete Fragen, abgegebene Antworten) an Database Handler.
GetSemesterRequest()	Die Anfrage an den Server, Semester aus der Datenbank auszulesen.
InsertNewQuestionRequest ()	Die Anfrage an den Server, neue Frage in der Datenbank zu speichern. Übermittelt die Fragedaten (Fach, Universität, Semester, Frage, falsche und richtige Antworten) an Database Handler
GetOpenGamesRequest ()	Die Anfrage an den Server, gestartete Spiele zu zeigen. Übermittelt die Spieldaten (Username) an Database Handler
ContinueGameRequest ()	Die Anfrage an den Server, das Spiel fortzusetzen. Übermittelt die Spieldaten (Username, Universität, Spiel ID) an Database Handler
GetClosedGames ()	Die Anfrage an den Server, beendete Spiele zu zeigen. Übermittelt die Spieldaten (Username) an Database Handler

Schnittstelle $\langle I30 \rangle$: DataStorage Schnittstelle

Operation	Beschreibung
writePlayerData ()	Wenn der Nutzer erfolgreich angemeldet ist, wird er in der Datenbank als angemeldet gespeichert.

Schnittstelle $\langle I40 \rangle$: Background Networker Schnittstelle

Operation	Beschreibung
doInBackground ()	Führt den Server Connector in einem anderen Thread aus.

Schnittstelle $\langle I50 \rangle$: Server Connector Schnittstelle

Operation	Beschreibung
connectPost ()	Hier wird die Verbindung per POST-Methode hergestellt.

3.3 Protokolle für die Benutzung der Komponenten

In diesem Abschnitt wird die korrekte Verwendung der einzelnen Komponenten anhand von Statecharts dargestellt. Die Komponenten die wiederverwendet werden, werden in den dazugehörigen Teilabschnitt beschrieben.

Grundsätzlich sind viele Softwarekomponenten dieses Projekts entweder spezifisch auf die entsprechende Anwendungssituation bezogen (z.B. Spielkoordination) oder so gestaltet, dass es eine Wiederverwendung möglich sein könnte, wenn eine Adaptation auf einen neuen Anwendungsfall durchgeführt wird. Der Client kann in den anderen ähnlichen Bereichen verwendet werden, z.B. als Lernapp für Schüler genutzt werden, weil die Kernfunktion des Systems in diesem Bereich gleich bleibt.

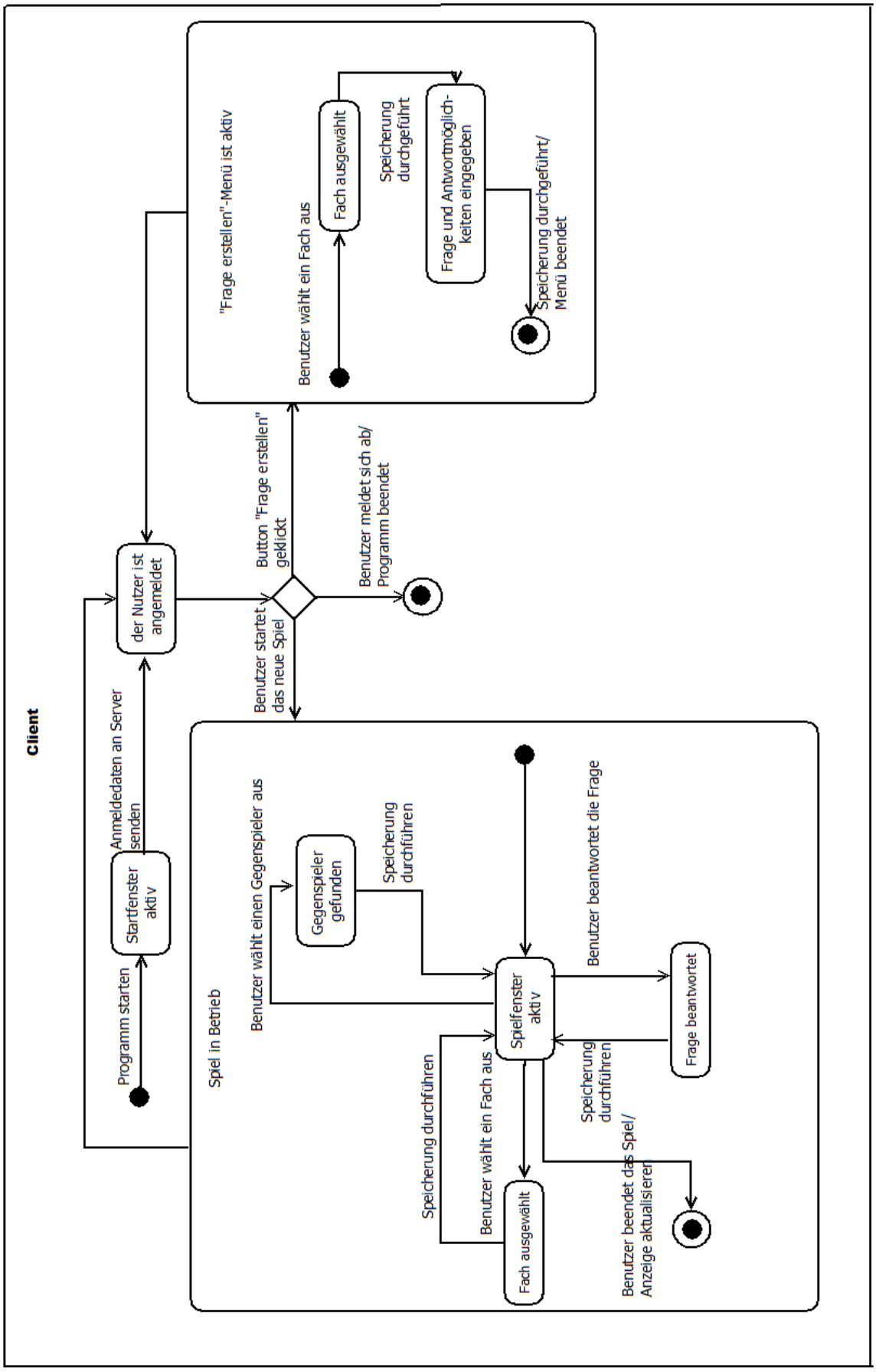

Abbildung 3.2: State Chart: Client

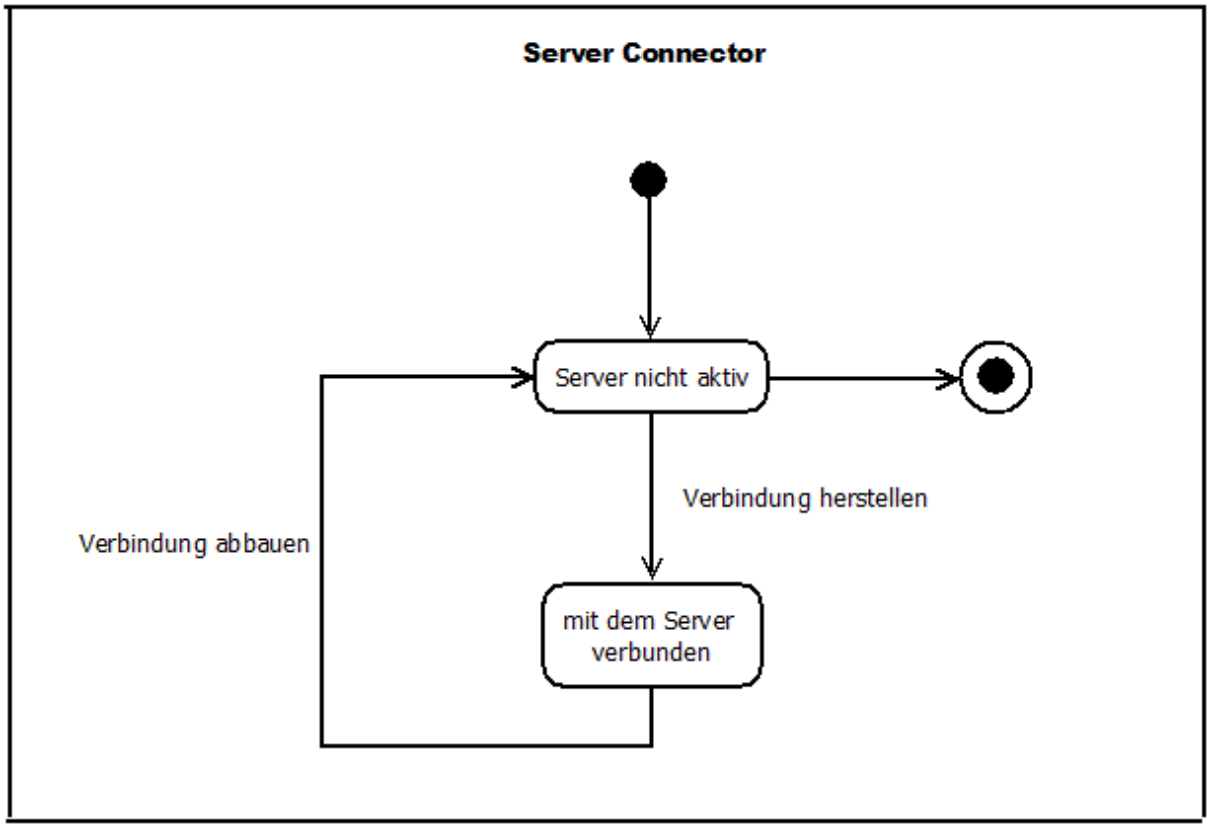

Abbildung 3.3: State Chart: Server Connector

Es ist auch denkbar, Unterkomponenten des Servers in anderen Zusammenhängen zu verwenden. Der Server Connector ließe sich z.B. auch in anderen Projekten verwenden. Server Connector ist in andere Softwareprodukte einfach einzubinden, weil er eigenständig arbeitet und keine weiteren Abhängigkeiten von dem System hat. Die Komponente besitzt nur zwei Zustände. Solange der Server nicht aktiv ist, passiert nichts in der Komponente. Um das System mit dem Server zu verbinden, soll der Server Connector die Verbindung per POST-Methode herstellen. Die Komponente Database Handler kann auch in den anderen Projekten verwendet werden. Nach dem die Daten (zum Beispiel bei der Registrierung) vom Nutzer eingegeben wurden, werden sie ausgelesen und an Database Handler übergeben. Der Database Handler erzeugt ein Feld (ArrayList<BasicNameValuePair>) für diese Eingaben. Danach werden die Daten von der Komponente an Background Networker weitergeleitet.

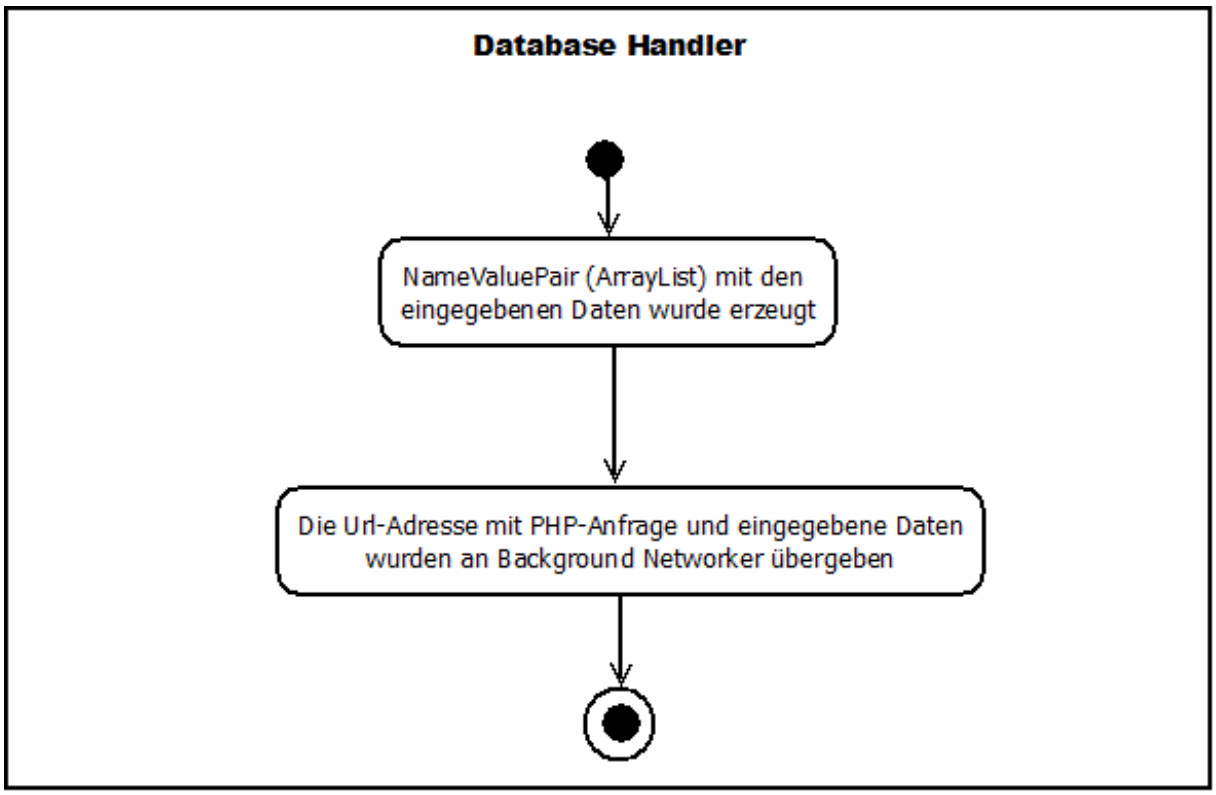

Abbildung 3.4: State Chart: Database Handler

Die anderen Komponenten des Systems sind nicht zur Weiterverwendung geeignet. Sie sind sehr stark von den anderen Komponenten abhängig und nur schwer auf andere Systeme zu übertragen.

4 Verteilungsentwurf

Dieses Kapitel beschreibt die Verteilung der einzelnen Komponenten. Zusätzlich wird ein Verteilungsdiagramm zur Veranschaulichung beigefügt. Verteilungdiagramme stellen dar, wie das System auf der Hardware, sowie der Netzwerktopologie verteilt ist. Die Applikation baut auf einer Client-Server-Arichtektur auf.

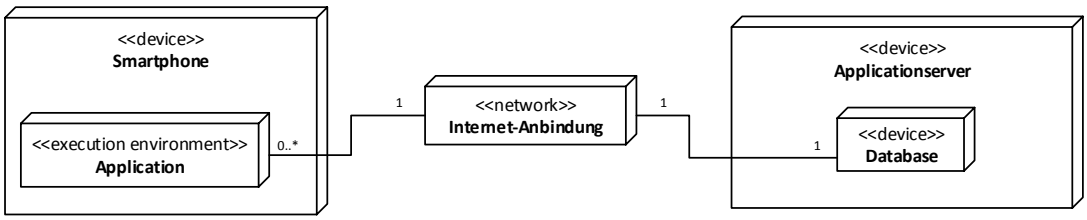

Abbildung 4.1: Verteilungsdiagramm

Die Abbildung 4.1 zeigt das Verteilungsdiagramm zur Applikation *Lernduell*, welche mit einem Server und einer Datenbank auf dem Server betrieben wird. Die Datenbank ist ein Hardware Element (device) und kommuniziert über das Internet (network) mit der Applikation, welche sich auf einem Smartphone befindet. Beim Smartphone handelt es sich ebenfalls um ein Hardware

Element (device). Die Applikation selbst ist eine Laufzeitumgebung und daher mit dem Typ execution environment gekennzeichnet.

5 Implementierungsentwurf

In diesem Kapitel werden die in Kapitel 3 aufgeführten Komponenten ein weiteres Mal betrachtet und gesondert in jeweiligen Unterkapiteln detailliert beschrieben. Dabei als Erstes auf die Implementierung der einzelnen Kompenenten eingegangen. Daraufhin werden die Klassen der verschiedenen Komponenten betrachtet und Attribute, Aufgaben und Kommunikationspartner dieser Klassen näher ausgeführt.

Für ein besseres Verständnis sollte Abbildung 3.1 erneut betrachtet werden.

5.1 Implementierung von Komponente $\langle C10 \rangle$: Client

Der Client beschreibt die gesamte Architektur der Applikation. Dazu gehört sowohl das vom Client bereitgestellte GUI, als auch die für den Spielablauf benötigten Nutzerdaten und Informationen.

Im nachfolgenden Klassendiagramm wird ein Überblick über die Architektur unseres Clients bereitgestellt.

5.1.1 Paket-/Klassendiagramm

5.1.2 Erläuterung

Der Client beschreibt die vollständige Android-Applikation. Hier geht es vom Graphic User Interface über den eigentlichen Spielablauf bis hin zur auswertenden Statistik und sonstigen Funktionen. Einen entsprechenden Umfang hat die Kompomente auch. Ebenfalls sind im Client die vom Android SDK bereitgestellten und benötigten Klassen vorhanden, um eine entsprechende Lauffähigkeit auf dem Android-Betriebssystem zu Ermöglichen.

Nachfolgend werden die Klassen der Abbildung 5.1 genauer beschrieben. Reine get- / set-Methoden oder Bibliotheksfunktionen der Übersicht halber nicht aufgeführt.

Anmelden$\langle CL110 \rangle$

Aufgabe

Ermöglicht das Anmelden in der Applikation.

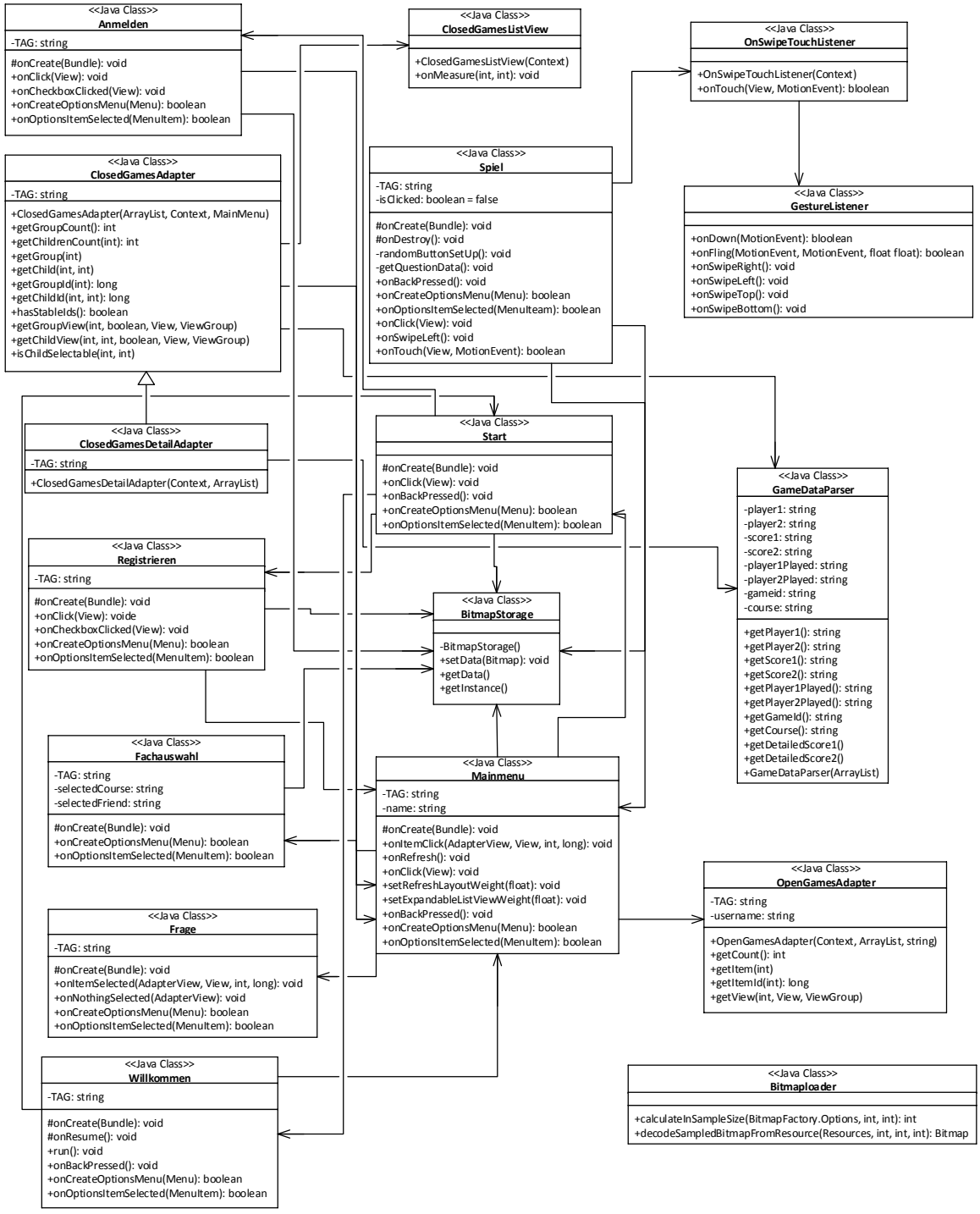

Abbildung 5.1: Klassendiagramm Komponente $\langle C10 \rangle$: Client

Attribute

TAG: Notwendig um bestimmte Inhalte in den Logs wiederzufinden

Operationen

onCreate(Bundle): Wird von Android bereitgestellt und beim Erzeugen einer Aktivität aufgerufen.

onClick(View): Zum Auslösen von Ereignissen beim Draufklicken.

onCheckboxClicked(View): Zum Auslösen von Ereignissen, wenn die Checkbox markiert wurde.

onCreateOptionsMenu(Menu): Initialisiert die Inhalte des Standardoptionsmenüs der Aktivitäten.

onOptionsItemSelected(MenuItem): Wird aufgerufen, wenn eine Auswahl im Optionsmenü getroffen wurde.

Kommunikationspartner

BitmapStorage und Mainmenu

Bitmaploader⟨$CL111$⟩

Aufgabe

Bild-Ressource umwandeln, um Sie zu Verkleinern.

Attribute

keine

Operationen

calculateInSampleSize(BitmapFactory.Options, int, int): Die Operation InSampleSize skaliert ein gegebenes Bild herunter, um Speicherplatz zu Sparen. Diese Größe wird hier berechnet.

decodeSampledBitmapFromResource(Resources, int, int, int): Wandelt eine Ressource in eine Bitmap um.

Kommunikationspartner

Keine

BitmapStorage⟨$CL112$⟩

Aufgabe

Speichert ein Bitmap für den späteren Gebrauch.

Attribute

keine

Operationen

BitmapStorage(): Der Konstruktor der Klasse

Kommunikationspartner

Keine

ClosedGamesAdapter⟨$CL113$⟩

Aufgabe

Füllt eine Liste mit bereits beendeten Spielen

Attribute

TAG: Notwendig um bestimmte Inhalte in den Logs wiederzufinden

Operationen

ClosedGamesAdapter(ArrayList, Context, Mainmenu): Der Konstruktor der Klasse
hasStableIds(): Gibt an, ob die einzelnen Elemente auch bei Änderungen der zugrunde liegenden Daten stabil bleiben.
isChildSelectable(int, int): Prüft, ob das Child-Element an der angegebenen Position Auswählbar ist.

Kommunikationspartner

ClosedGamesDetailAdapter, ClosedGamesListView und Mainmenu

ClosedGamesDetailAdapter⟨$CL114$⟩

Aufgabe

Füllt Liste von beendeten Spielen beim Auswählen mit detaillierten Spieldaten.

Attribute

Keine

Operationen

ClosedGamesDetailAdapter(Context, ArrayList): Der Konstruktor der Klasse
hasStableIds(): Gibt an, ob die einzelnen Elemente auch bei Änderungen der zugrunde liegenden Daten stabil bleiben.
isChildSelectable(int, int): Prüft, ob das Child-Element an der angegebenen Position Auswählbar ist.

Kommunikationspartner

GameDataParser

ClosedGamesListView⟨$CL115$⟩

Aufgabe

Liste zum Darstellen der detaillierten Spieldaten.

Attribute

Keine

Operationen

ClosedGamesListView(Context): Der Konstruktor der Klasse
onMeasure(int, int): Wird aufgerufen, um die Größenanforderungen für diese Ansicht und alle seine Kinder zu bestimmen.

Kommunikationspartner

Keine

Fachauswahl⟨$CL116$⟩

Aufgabe

Stellt die Möglichkeit bereit, das Fach für ein neues Spiel auszuwählen.

Attribute

TAG: Notwendig um bestimmte Inhalte in den Logs wiederzufinden

Operationen

onCreate(Bundle): Wird von Android bereitgestellt und beim Erzeugen einer Aktivität aufgerufen.

onNothingSelected(AdapterView): Callback-Methode, die aufgerufen wird, wenn die aktuelle Auswahl aus der Ansicht verschwindet.

onCreateOptionsMenu(Menu): Initialisiert die Inahlte der Aktivitäten des Optionsmenüs.

onOptionsItemSelected(MenuItem): Wird aufgerufen, wenn eine Auswahl im Optionsmenü getroffen wurde.

Kommunikationspartner

BitmapStorage

Frage⟨$CL117$⟩

Aufgabe

Stellt die zuvor hinzugefügten Fragen für ein Spiel bereit.

Attribute

TAG: Notwendig um bestimmte Inhalte in den Logs wiederzufinden

Operationen

onCreate(Bundle): Wird von Android bereitgestellt und beim Erzeugen einer Aktivität aufgerufen.

onItemSelected(AdapterView, View, int, long): Callback-Methode, die aufgerufen wird, wenn ein Element in dieser Ansicht ausgewählt wurde.

onNothingSelected(AdapterView): Callback-Methode, die aufgerufen wird, wenn die aktuelle Auswahl aus der Ansicht verschwindet.

onCreateOptionsMenu(Menu): Initialisiert die Inahlte der Aktivitäten des Optionsmenüs.

onOptionsItemSelected(MenuItem): Wird aufgerufen, wenn eine Auswahl im Optionsmenü getroffen wurde.

Kommunikationspartner

Keine

GameDataParser⟨$CL118$⟩

Aufgabe

Verarbeitet die Spieldaten, um Sie leicht zugänglich zu Machen.

Attribute

player1: Beschreibt den ersten Spieler

player2: Beschreibt den zweiten Spieler

score1: Beschreibt die Punktzahl des ersten Spielers

score2: Beschreibt die Punktzahl des zweiten Spielers

player1Played: Überprüft, ob Spieler 1 bereits gespielt hat

player2Played: Überprüft, ob Spieler 2 bereits gespielt hat

gameId: Dem Spiel wird eine feste Spiel-ID zugewiesen

course: Ruft den für das Spiel ausgewählten Kurs ab

detailedScore1: Ausführliche Statistik nach beendetem Spiel für den ersten Spieler

detailedScore2: Ausführliche Statistik nach beendetem Spiel für den zweiten Spieler

Operationen

GameDataParser(ArrayList): Der Konstruktor der Klasse

Kommunikationspartner

Keine

Mainmenu⟨*CL*119⟩

Aufgabe

Funktionen des Hauptmenüs

Attribute

TAG: Notwendig um bestimmte Inhalte in den Logs wiederzufinden

Operationen

onCreate(Bundle): Wird von Android bereitgestellt und beim Erzeugen einer Aktivität aufgerufen.

onItemClick(AdapterView, View, int, long): Callback-Methode, die aufgerufen wird, wenn ein Element angeklickt wird.

OnRefresh(): Wird aufgerufen, wenn die Seite über eine Touch-Streichgeste auf dem Bildschirm aktualisiert wird.

onClick(): Zum Auslösen von Ereignissen beim Draufklicken.

setRefreshLayoutWeight(float):

setExpandableListViewWeight(float): Eine Ansicht, die Elemente in einer vertikal scrollenden Zwei-Ebenen-Liste ausgibt.

onBackPressed(): Wird aufgerufen, wenn der Benutzer die Zurück-Taste betätigt.

calculateGameData(ArrayList):

onCreateOptionsMenu(Menu): Initialisiert die Inhalte des Standardoptionsmenüs der Aktivitäten.

onOptionsItemSelected(MenuItem): Wird aufgerufen, wenn eine Auswahl im Optionsmenü getroffen wurde.

Kommunikationspartner

BitmapStorage, Fachauswahl, Frage, OpenGamesAdapter und Start

OnSwipeTouchListener⟨$CL120$⟩

Aufgabe

Wartet auf eine Touch-Geste und führt was aus.

Attribute

Keine

Operationen

OnSwipeTouchListener(Cotext): Der Konstruktor der Klasse

onTouch(View, MotionEvent): Startet eine neue Aktivität, wenn ein Touch-Ereignis eintritt

Kommunikationspartner

GestureListener

GestureListener⟨$CL121$⟩

Aufgabe

Wartet auf eine Touch-Geste und führt was aus.

Attribute

Keine

Operationen

onDown(MotionEvent):

onFling(MotionEvent, MotionEvent, float, float):

onSwipeRight(): Startet eine neue Aktivität, wenn das Bild per Touch nach rechts geschoben wird.

onSwipeLeft(): Startet eine neue Aktivität, wenn das Bild per Touch nach links geschoben wird.

onSwipeTop(): Startet eine neue Aktivität, wenn das Bild per Touch nach oben geschoben wird.

onSwipeBottom(): Startet eine neue Aktivität, wenn das Bild per Touch nach unten geschoben wird.

Kommunikationspartner

Keine

OpenGamesAdapter⟨$CL122$⟩

Aufgabe

Füllt Liste von offenen Spielen mit Daten.

Attribute

TAG: Notwendig um bestimmte Inhalte in den Logs wiederzufinden
username: Ruft den Nutzernamen des Gegenspielers ab

Operationen

OpenGamesAdapter(Context, ArrayList, String): Der Konstruktor der Klasse

Kommunikationspartner

GameDataParser

Registrieren⟨$CL123$⟩

Aufgabe

Die Möglichkeit, einen neuen Account in der Applikation zu registrieren

Attribute

TAG: Notwendig um bestimmte Inhalte in den Logs wiederzufinden

Operationen

onCreate(Bundle): Wird von Android bereitgestellt und beim Erzeugen einer Aktivität aufgerufen.
onClick(View): Zum Auslösen von Ereignissen beim Draufklicken.
onCheckboxClicked(View): Zum Auslösen von Ereignissen, wenn die Checkbox markiert wurde.
onCreateOptionsMenu(Menu): Initialisiert die Inhalte des Standardoptionsmenüs der Aktivitäten.
onOptionsItemSelected(MenuItem): Wird aufgerufen, wenn eine Auswahl im Optionsmenü getroffen wurde.

Kommunikationspartner

BitmapStorage und Mainmenu

Spiel⟨$CL124$⟩

Aufgabe

Verarbeitet alle relevanten Spieldaten.

Attribute

TAG: Notwendig um bestimmte Inhalte in den Logs wiederzufinden
isClicked: Überprüft nach abgeschickter Lösung, ob die Antwort richtig oder falsch war.

Operationen

onCreate(Bundle): Wird von Android bereitgestellt und beim Erzeugen einer Aktivität aufgerufen.
randomButtonSetUp():

onDestroy(): Führt eine Bereinigung aus, bevor eine Aktivtät zerstört wird. Dies passiert entweder beim Beenden einer Aktivität oder wenn das System die Aktivität temporär zerstört, um Speicherplatz einzusparen.
getQuestionData(): Ruft die Daten der Spiel-Fragen ab.
onBackPressed(): Wird aufgerufen, wenn der Benutzer die Zurück-Taste betätigt.
onCreateOptionsMenu(Menu): Initialisiert die Inhalte des Standardoptionsmenüs der Aktivitäten.
onOptionsItemSelected(MenuItem): Wird aufgerufen, wenn eine Auswahl im Optionsmenü getroffen wurde.
onClick(View): Zum Auslösen von Ereignissen beim Draufklicken. onSwipeLeft(): Startet eine neue Aktivität, wenn das Bild per Touch nach links geschoben wird.
onTouch(View, MotionEvent): Startet eine neue Aktivität, wenn ein Touch-Ereignis eintritt.

Kommunikationspartner

BitmapStorage, Mainmenu und OnSwipeTouchListener

Start⟨$CL125$⟩

Aufgabe

Verarbeitet alle relevanten Daten zum Starten eines neuen Spiels.

Attribute

keine

Operationen

onCreate(Bundle): Wird von Android bereitgestellt und beim Erzeugen einer Aktivität aufgerufen.
onClick(View): Zum Auslösen von Ereignissen beim Draufklicken.
onBackPressed(): Wird aufgerufen, wenn der Benutzer die Zurück-Taste betätigt.
onCreateOptionsMenu(Menu): Initialisiert die Inhalte des Standardoptionsmenüs der Aktivitäten.
onOptionsItemSelected(MenuItem): Wird aufgerufen, wenn eine Auswahl im Optionsmenü getroffen wurde.

Kommunikationspartner

Anmelden, BitmapStorage, Registrieren und Willkommen

Willkommen⟨$CL126$⟩

Aufgabe

Stellt den Willkommens-Bildschirm bereit.

Attribute

TAG: Notwendig um bestimmte Inhalte in den Logs wiederzufinden

Operationen

onCreate(Bundle): Wird von Android bereitgestellt und beim Erzeugen einer Aktivität aufgerufen.
run():
onBackPressed(): Wird aufgerufen, wenn der Benutzer die Zurück-Taste betätigt.
onResume(): Wird aufgerunfen, um eine Aktivität mit dem Nutzer zu Starten oder fortzusetzen..
onCreateOptionsMenu(Menu): Initialisiert die Inhalte des Standardoptionsmenüs der Aktivitäten.
onOptionsItemSelected(MenuItem): Wird aufgerufen, wenn eine Auswahl im Optionsmenü getroffen wurde.

Kommunikationspartner

BitmapStorage, Mainmenu und Start

5.2 Implementierung von Komponente $\langle C20 \rangle$: Database Handler

Der DataBase Handler definiert die an den Server zu übertragenden Daten, welche er anschließend an den Background Networker übergibt. Hierbei nutzt der DataBase Handler die Google Guava Bibliothek.

5.2.1 Paket-/Klassendiagramm

5.2.2 Erläuterung

Nachfolgend werden die Klassen der Abbildung 5.2 genauer beschrieben. Reine get- / set-Methoden oder Bibliotheksfunktionen der Übersicht halber nicht aufgeführt.

Database Handler$\langle CL210 \rangle$

Aufgabe

Definiert die an der Server zu übertragenden Daten

Attribute

TAG: Notwendig um bestimmte Inhalte in den Logs wiederzufinden
pairs: ArrayList

Operationen

findGame(String, String, String): Erstellt neues Spiel gegen zufälligen Gegner
registerUser(String, String, String, String): Registriert neuen Nutzer loginUser(String, String): Loggt registrierten User ein
saveResult(ArrayList): Speichert die Antworten der Spieler in einer Runde

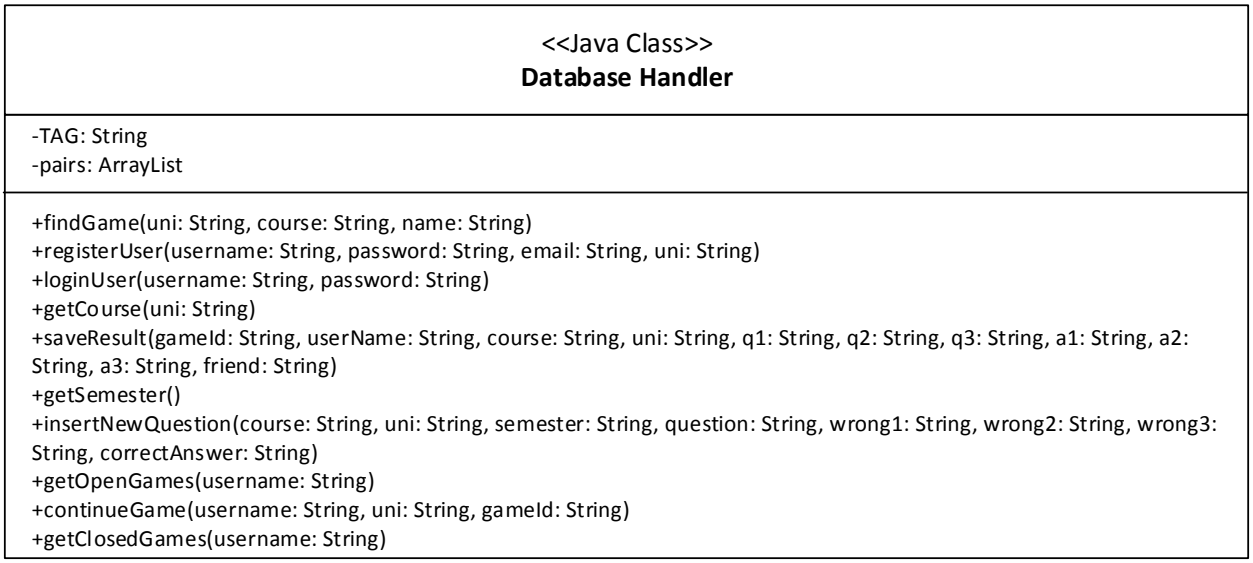

Abbildung 5.2: Klassendiagramm Komponente $\langle C20 \rangle$: Database Handler

insertNewQuestion(String, String, String, String, String, String, String, String): Fügt neue Frage hinzu

continueGame(String, String, String): „Pausiertes“ Spiel wieder aufnehmen

Kommunikationspartner

Client, Server Connector

5.3 Implementierung von Komponente $\langle C30 \rangle$: Background Networker

Da es in Java nicht möglich ist, Netzwerkanfragen auf dem Mainthread auszuführen, führt der Background Networker den Server Connector in einem anderen Thread aus. Ähnlich wie der Database Handler nutzt auch der Background Networker die Google Guava Bibliothek.

5.3.1 Paket-/Klassendiagramm

5.3.2 Erläuterung

Nachfolgend werden die Klassen der Abbildung 5.3 genauer beschrieben. Reine get- / set-Methoden oder Bibliotheksfunktionen der Übersicht halber nicht aufgeführt.

Background Networker $\langle CL310 \rangle$

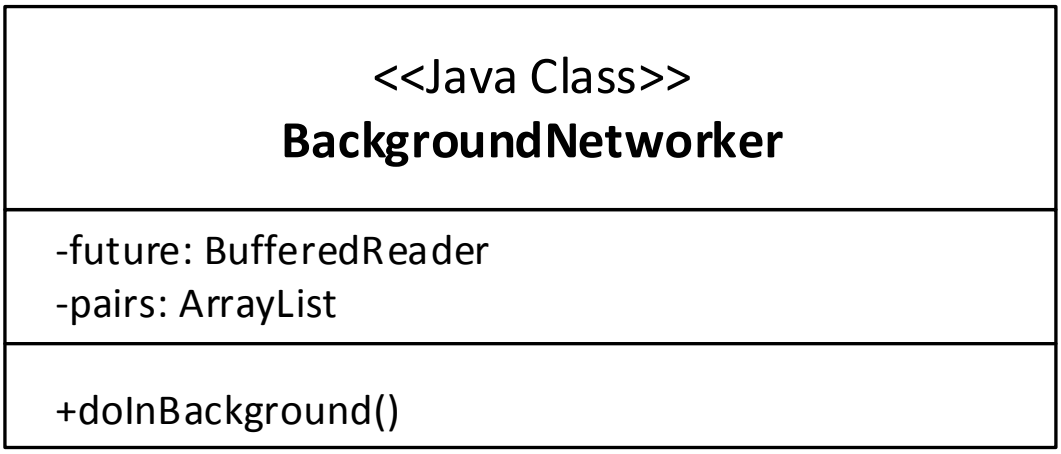

Abbildung 5.3: Klassendiagramm Komponente $\langle C30 \rangle$: Background Networker

Aufgabe

Führt den Server Connector in einem anderen Thread aus

Attribute

future: BufferedReader

pairs: ArrayList

Operationen

doInBackground(BufferedReader, ArrayList): Führt den Server Connector in einem anderen Thread aus

Kommunikationspartner

Server Connector

5.4 Implementierung von Komponente $\langle C40 \rangle$: Server Connector

Als Ergänzung zur Funktionalität der Java-Klassenbibliothek wird auch beim Server Connector die Google Guava Bibliothek eingesetzt. Im Server Connector findet die eigentliche Verbindungsherstellung statt, hierbei wird ein vorgefertigtes php-Skript mit den gegebenen Daten auf dem Server ausgeführt. Die Daten werden im Json-Format bezogen und an den JsonParser gegeben.

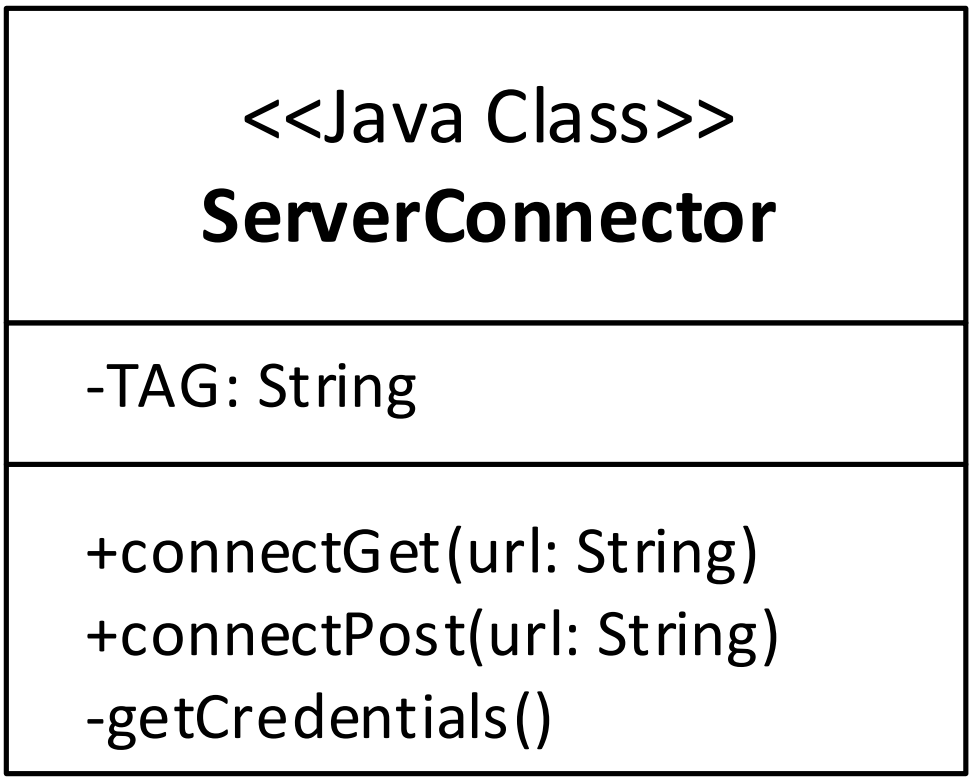

Abbildung 5.4: Klassendiagramm Komponente $\langle C40 \rangle$: Server Connector

5.4.1 Paket-/Klassendiagramm

5.4.2 Erläuterung

Nachfolgend werden die Klassen der Abbildung 5.4 genauer beschrieben. Reine get- / set-Methoden oder Bibliotheksfunktionen der Übersicht halber nicht aufgeführt.

Server Connector$\langle CL410 \rangle$

Aufgabe

Verbindungsherstellung per POST-Methode

Attribute

TAG: Notwendig um bestimmte Inhalte in den Logs wiederzufinden

Operationen

connectGet(String): Verbindet mit dem Server und gibt BufferedReader aus private String
connectPost(String, ArrayList): Verbindungsherstellung per POST-Methode

Kommunikationspartner

Database Handler, DataStorage

5.5 Implementierung von Komponente $\langle C50 \rangle$: DataStorage

Der DataStorage übernimmt ist für die Speicherverwaltung des Lernduells zuständig. Er speichert sämtliche Daten, die in den einzelnen Komponenten des Systems entstanden sind und stellt diese zu einem späteren Zeitpunkt zur Weiterverarbeitung zur Verfügung. Somit werden im Storage Nutzerdaten, Statistiken, Spieldaten, Fächerdaten, der digitale Fragenkatalog verwaltet. Die Klasse FriendStorage greift hierbei auf die Gson Bibliothek zurück.

5.5.1 Paket-/Klassendiagramm

5.5.2 Erläuterung

Nachfolgend werden die Klassen der Abbildung 5.5 genauer beschrieben. Reine get- / set-Methoden oder Bibliotheksfunktionen der Übersicht halber nicht aufgeführt.

FriendStorage$\langle CL510 \rangle$

Aufgabe

Speichert die Freundesliste

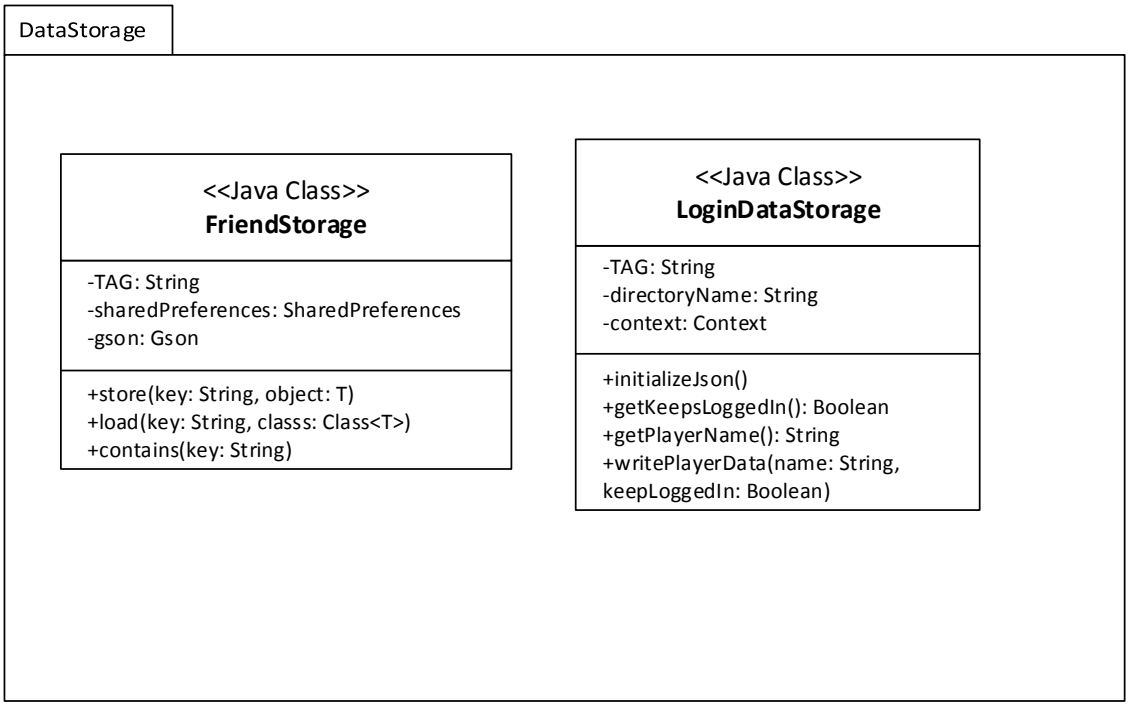

Abbildung 5.5: Klassendiagramm Komponente $\langle C50 \rangle$: DataStorage

Attribute

TAG: Notwendig um bestimmte Inhalte in den Logs wiederzufinden

sharedPreferences: Kleine Datenspeicher, die von Android für jede App bereitgestellt werden.

gson: Bibliothek um das Verarbeiten von JSON zu erleichtern.

Operationen

store(String, T): Speichert Gegenspieler(eingabe)

load(String, Class<T>): Lädt, bzw. zeigt gespeicherte (eingegebene) Gegenspieler an

Kommunikationspartner

Client, Server Connector

LoginDataStorage$\langle CL511 \rangle$

Aufgabe

Speichert loginrelevante Daten

Attribute

directoryName: Ordner, in den der JSON (die lokalen Daten) gespeichert werden.

context: Wird von allen Objekten (Views) benötigt, die aktiv angezeigt. werden sollen, oder wenn an dem aktuellen Zustand der Views etwas geändert werden muss.

Operationen

initializeJson(): Erstellt Ordner für Spieler Daten

writePlayerData(String, Boolean): Speichert Spielerdaten

Kommunikationspartner

Client, Server Connector

6 Datenmodell

In diesem Kapitel wird anhand eines Klassendiagramms dargestellt, welche Datensätze dauerhaft auf dem Server gespeichert werden. Dies umfasst neben den Nutzer- und Universätsdaten auch die Spielrelevanten Informationen vor, während und nach einer Spiel-Runde von *Lernduell*. Darüber hinaus wird auf die Beziehungen und die Kommunikation zwischen den einzelnen Klassen eingegangen.

6.1 Diagramm

Die Abbildung 6.1 zeigt das Datenmodell von *Lernduell* als Klassendiagramm, welches in Abschnitt 3.2 noch näher erläutert wird.

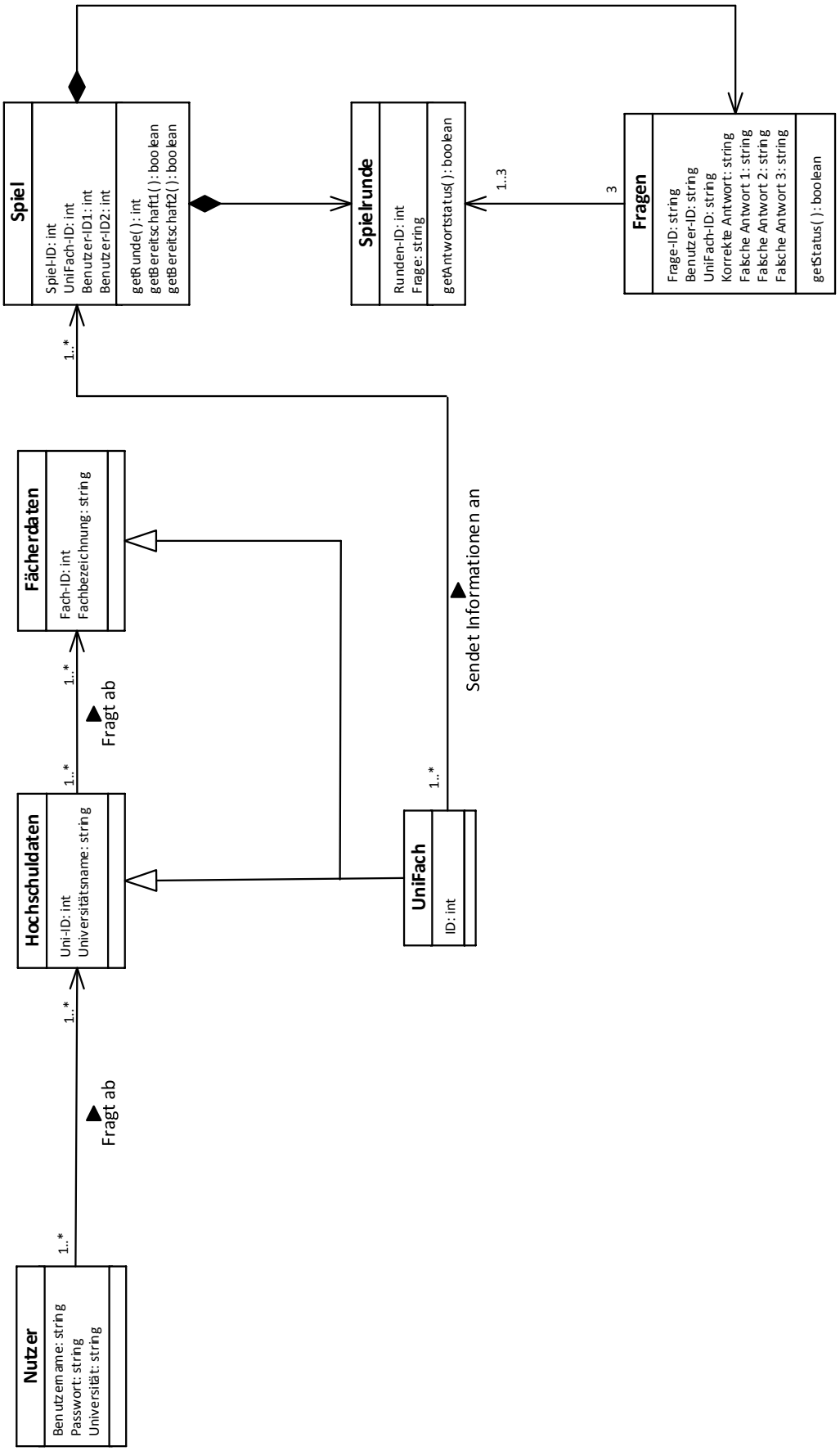

Abbildung 6.1: Klassendiagramm des Datenmodells von *Lernduell*

6.2 Erläuterung

Nutzer $\langle E10 \rangle$

Beziehung	Kardinalität	Erwartete Datenmenge	Beschreibung
Hochschuldaten $\langle E20 \rangle$	1..*	Wenige Kilobyte	Das Nutzerprofil bezieht die Daten der Hochschule und speichert Sie im Profil.

Hochschuldaten $\langle E20 \rangle$

Beziehung	Kardinalität	Erwartete Datenmenge	Beschreibung
Nutzer $\langle E10 \rangle$	1..*	Wenige Kilobyte	Sendet Daten an die Klasse Nutzer, sobald Sie abgefragt werden.
Fächerdaten $\langle E30 \rangle$	1..*	Wenige Kilobyte	Bezieht die Daten der jeweiligen angebotenen Fächer der Hochschule.
UniFach $\langle E40 \rangle$	-	Wenige Kilobyte	Vererbt seine Daten an die Klasse UniFach, welches die Verknüpfung zwischen den Hochschuldaten und den Fächerdaten darstellt.

Fächerdaten $\langle E30 \rangle$

Beziehung	Kardinalität	Erwartete Datenmenge	Beschreibung
Hochschuldaten $\langle E20 \rangle$	1..*	Wenige Kilobyte	Sendet Daten an die Klasse Hochschuldaten, sobald Sie abgefragt werden.
UniFach $\langle E40 \rangle$	-	Wenige Kilobyte	Vererbt die Daten an die Klasse UniFach, welches die Verknüpfung zwischen den Hochschuldaten und den Fächerdaten darstellt.

UniFach $\langle E40 \rangle$

Beziehung	Kardinalität	Erwartete Datenmenge	Beschreibung
Hochschuldaten $\langle E20 \rangle$	-	Wenige Kilobyte	Erbt aus der Kategorie Hochschuldaten, da diese Klasse die Verknüpfung zwischen den Hochschuldaten und den Fächerdaten darstellt.
Fächerdaten $\langle E30 \rangle$	-	Wenige Kilobyte	Erbt aus der Kategorie Fächerdaten, da diese Klasse die Verknüpfung zwischen den Hochschuldaten und den Fächerdaten darstellt.

Spiel ⟨*E*40⟩	1..*	Wenige Kilobyte	Sendet Informationen aus der Klasse UniFach an die Klasse Spiel, wodurch bestimmt wird, welcher Mitspieler geeignet wäre und wie der mögliche Fragenpool für die entsprechende Runde aussieht.

Fragen ⟨*E*50⟩

Beziehung	**Kardinalität**	**Erwartete Datenmenge**	**Beschreibung**
Spiel ⟨*E*60⟩	-	Wenige Kilobyte	Es handelt sich um eine Komposition. Ohne das Spiel gibt es die Klasse Fragen nicht. Die Fragen für die Klasse Spiel werden hier gespeichert.
Spielrunde ⟨*E*70⟩	3	Wenige Kilobyte	Die Klasse Fragen sendet die relevanten Spiel-Fragen vor jeder Spielrunde eines laufenden Spiels an die Klasse Spielrunde.

Spiel $\langle E60 \rangle$

Beziehung	Kardinalität	Erwartete Datenmenge	Beschreibung
UniFach $\langle E40 \rangle$	1..*	Wenige Kilobyte	Bekommt Informationen von der Klasse UniFach, um einen möglichen Mitspieler und die Fragen zu Bestimmen.
Fragen $\langle E50 \rangle$	1	Wenige Kilobyte	Die Klasse Fragen ist ein existenzabhängier Teil der Klasse Spiel, in welcher der gesamte Fragenpool für das Spiel gespeichert wird.
Spielrunde $\langle E70 \rangle$	1	Wenige Kilobyte	Die Klasse Spielrunde ist ein existenzabhängier Teil der Klasse Spiel, in welchem die Daten des laufenden Spiels gespeichert werden.

Spielrunde $\langle E70 \rangle$

Beziehung	Kardinalität	Erwartete Datenmenge	Beschreibung
Fragen $\langle E50 \rangle$	1..3	Wenige Kilobyte	Bezieht die Fragen für die jeweilige Spielrunde aus dieser Klasse.
Spiel $\langle E60 \rangle$	-	Wenige Kilobyte	Es handelt sich um eine Komposition. Ohne das Spiel gibt es die Klasse Spielrunde nicht. In der Klasse Spiel wird der Verlauf aller Spiele gespeichert, während in Spielrunde die jeweiligen Rundendaten gesichert werden.

7 Konfiguration

Dieses Kapitel geht genauer auf die Hard-, und Software Konfiguration von *Lernduell* ein.

Für den Spielbetrieb von *Lernduell* ist ein Web-Server nötig, welcher während der Entwicklung mit dem Betriebssystem Debian betrieben wurde. Für den Endkunden ist keine besondere Konfiguration nötig. Es muss lediglich ein reibungsloser Betrieb von MySQL als Datenbankmanagementsystem und phpMyAdmin für das Webinterface gewährleistet sein, wobei weder MySQL noch phpMyAdmin nennenswerte Anforderungen an die Hardware des Servers stellen.

Als Client ist ein Android Smartphone mit mindestens der Version 4.0.3 (IceCreamSandwich) nötig.

Zum derzeitigen Entwicklungsstand hat *Lernduell* eine config-Datei, welche sich auf dem Webserver befindet.

Config.php dient der Authentifizierung der anderen php-Skripte um auf die Datenbank zuzugreifen. Die Datei definiert also, unter welcher Adresse der Server zu erreichen ist auf dem die Datenbank liegt (MYSQL_HOST), den Benutzernamen um sich einzuloggen (MYSQL_BENUTZER), das entsprechende Passwort (MYSQL_KENNWORT) und das zu verwendende Datenbankschema (MYSQL_DATENBANK).

8 Änderungen gegenüber Fachentwurf

Zurzeit existieren folgende Änderung gegenüber dem Fachentwurf:

Abbildung 2.1 und Abbildung 2.2 enthielten Fehler, welche ausgebessert wurden.

9 Erfüllung der Kriterien

Nachfolgend wird der aktuelle Stand der Erfüllung der einzelnen Kriterien des Pflichtenheftes beschrieben.

9.1 Musskriterien

Die folgenden Kriterien sind unabdingbar und müssen durch das Produkt erfüllt werden:

RM1 *Benutzerkonten*

Dieses Kriterium wird erfüllt.

Bei erstmaligem Nutzen der Applikation auf einem Client (Komponente $\langle C10 \rangle$) muss der Nutzer sich registrieren, hierbei werden die Daten durch die Methode RegisterRequest() an den Server (Komponente $\langle C20 \rangle$) weitergegeben und ein neuer Nutzer angelegt. Mittels writePlayerData() wird der Nutzer nach erfolgreicher Anmeldung in der Datenbank als angemeldet gespeichert (Komponente $\langle C50 \rangle$).

RM2 *Spielervermittlung*

Dieses Kriterium wird erfüllt.

Mittels der Methode FindGameRequest() soll ein neues Spiel gefunden werden, dabei werden die Nutzerdaten (Universität, Fach, Username) an die Datenbank (Komponente $\langle C20 \rangle$) übermittelt.

RM3 *Eigene Fragen*

Dieses Kriterium wird erfüllt.

Durch InsertNewQuestionRequest () wird auf dem Server eine neue Frage in der Datenbank gespeichert (Komponente $\langle C50 \rangle$). Hierbei werden sämtliche vom Nutzer über den Client (Komponente $\langle C10 \rangle$) eingegebene Fragedaten (Fach, Universität, Semester, Frage, falsche und richtige Antworten) an den Server (Komponente $\langle C20 \rangle$) übermittelt.

RM4 *Partien annhemen oder ablehnen*

Dieses Kriterium wird erfüllt.

Nach erfolgreicher Spielvermittlung erhalten die Nutzer auf ihrem Client (Komponente $\langle C10 \rangle$) eine Einladung zum Spiel, die sie entweder annehmen, oder ablehnen können.

RM5 *Flüssiger Spielablauf*

Dieses Kriterium wird erfüllt.

Nach der Spielvermittlung läuft eine Partie in drei Runden mit jeweils drei Fragen ab. Mittels ContinueGameRequest () können Spiele später fortgesetzt werden. Die Nutzer spielen auf dem Client (Komponente $\langle C10 \rangle$), wobei die Antworten an den Server (Komponente $\langle C20 \rangle$) gesendet und dort abgeglichen werde.

RM6 *Statistiken*

Dieses Kriterium wird zurzeit in folgendem Maße erfüllt:

Nach jedem Spiel erhalten die Nutzer eine Statistik für die gerade gespielte Partie, mittels SaveResultRequest () werden die Ergebnisse des Spiels gespeichert, an den Server (Komponente $\langle C50 \rangle$) übertragen und dem Nutzer angezeigt.

9.2 Sollkriterien

Die Erfüllung folgender Kriterien für das abzugebende Produkt wird angestrebt:

RS1 *Übersichtliches und ansprechendes Design*

Dieses Kriterium wird zurzeit in folgendem Maße erfüllt:

Das Design der Applikation (auf dem Client, Komponente $\langle C10 \rangle$) ist übersichtlich gehalten und wird ständig verbessert um es für den Nutzer so ansprechend wie möglich zu gestalten, es ist aber noch nicht final.

9.3 Kannkriterien

Die Erfüllung folgender Kriterien für das abzugebende Produkt wird angestrebt:

RC1 *Semester*

Dieses Kriterium wird erfüllt.

Bei der Erstellung eigener Fragen wird das Semester vom Nutzer mit angegeben. Mittels GetSemesterRequest() wird es später aus der Datenbank (Komponente $\langle C50 \rangle$) ausgelesen.

RC2 *Fragen melden*

Dieses Kriterium wird voraussichtlich nicht erfüllt werden.

RC3 *Kontaktliste*

Dieses Kriterium wird zurzeit in folgendem Maße erfüllt:

Spieler gegen die man zuletzt gespielt hat, können bei der Spieleröffnung aus einer Liste ausgewählt werden.

RC4 *Aufgeben*

Dieses Kriterium wird voraussichtlich nicht erfüllt werden.

10 Glossar

Android Betriebssystem und Software-Plattform für mobile Geräte.
Server Computerprogramm oder Computer für den Zugriff auf eine zentrale Ressource oder Dienst in einem Netzwerk.
MySQL Relationales Datenbankverwaltungssystem.
PHP „PHP: Hypertext Preprocessor“, Skriptsprache mit einer an C und Perl angelehnten Syntax, die hauptsächlich zur Erstellung dynamischer Webseiten oder Webanwendungen verwendet wird.
Smartphone Mobiltelefon (umgangssprachlich Handy), das mehr Computer-Funktionalität und -konnektivität als ein herkömmliches fortschrittliches Mobiltelefon zur Verfügung stellt.
Tablet Der Tabletcomputer ist ein tragbarer, flacher Computer in besonders leichter Ausführung mit einem Touchscreen, aber, anders als bei Notebooks, ohne ausklappbare mechanische Tastatur.
App oder Applikation Anwendungssoftware
Exception Ausnahmebehandlung
Combobox Datenfeld in einer grafischen Benutzeroberfläche (GUI) einer Computer-Software, über das der Benutzer eine Auswahl aus vorgegebenen Möglichkeiten treffen oder alternativ eigene Eingaben tätigen kann.
GUI Grafische Benutzeroberfläche
Backend Das Backend bezeichnet den Teil der Software, der für die Funktionalität und Logik des Programms im Hintergrund zuständig ist. Das Backend wird vom Anwender nicht “gesehen”.
Client Client bezeichnet ein Computerprogramm, das auf dem Endgerät eines Netzwerks ausgeführt wird und mit einem Zentralrechner Server kommuniziert. Man nennt auch ein Endgerät selbst, das Dienste von einem Server abruft, Client.
Account Ein Benutzerkonto, kurz Nutzerkonto ist eine Zugangsberechtigung zu einem zugangsbeschränkten IT-System.
ID Kurzform für: Identifikator. Ein Identifikator ist ein mit einer bestimmten Identität verknüpftes Merkmal zur eindeutigen Identifizierung des tragenden Objekts.
Debian Debian ist ein seit 1993 gemeinschaftlich entwickeltes freies Betriebssystem.
phpMyAdmin phpMyAdmin ist eine freie PHP-Applikation zur Administration von MySQL-Datenbanken.
Session-ID Wird bei Anwendungen auf zustandslosen Protokollen als Identifikationsmerkmal verwendet, um mehrere zusammengehörige Anfragen eines Benutzers zu erkennen und einer Sit-

zung zuzuordnen.

Countdown die getaktete (kurz vor dem Ende im Sekundentakt) Bekanntgabe der bis zum Eintreten eines bestimmten Ereignisses noch fehlenden Zeitspanne.

JSON = JavaScript Object Notation, ist ein kompaktes Datenformat in einer einfach lesbaren Textform zum Zweck des Datenaustauschs zwischen Anwendungen.

Handler Bezeichnet in der Informatik eine Funktion, die einer anderen Funktion als Parameter übergeben und von dieser unter gewissen Bedingungen aufgerufen wird.

POST-Methode Übertragung der Daten mit einer speziell dazu vorgesehenen Anfrageart im HTTP-Nachrichtenrumpf, so dass sie in der URL nicht sichtbar sind.

Thread Ausführungsstrang oder eine Ausführungsreihenfolge in der Abarbeitung eines Programms.

Topologie (Rechnernetz) Die Topologie bezeichnet bei einem Computernetz die Struktur der Verbindungen mehrerer Geräte untereinander, um einen gemeinsamen Datenaustausch zu gewährleisten.

Klasse (Java) Unter einer Klasse (auch Objekttyp genannt) versteht man in der objektorientierten Programmierung ein abstraktes Modell bzw. einen Bauplan für eine Reihe von ähnlichen Objekten.

Klasse (Java) Gson (auch bekannt als Google Gson) ist eine open source Java Bibliothek to anordnen von Java Objekten in Verbindung mit JSON.

Google Guava Ist eine freie Sammlung von Programmbibliotheken für die Programmiersprache Java. Die Sammlung ist als Ergänzung zur Funktionalität der Java-Klassenbibliothek gedacht und erweitert diese beispielsweise bezüglich Collections, I/O-Unterstützung oder Stringmanipulationen.