5.3.2017

Dokumentation

Abschlussprojekt im Fach Mobile Computing 2,
Wintersemester 2016/17

Hendrik Schmidt, Felix Wisser
Fachhochschule Erfurt

Inhalt

1 Einleitung

1.1 Das Spiel Go

Das Go Spiel ist ein asiatisches Brettspiel, welches mit über 4000 Jahren das älteste Brettspiel der Welt ist, welches noch nach den ursprünglichen Regeln gespielt wird. Es wird gewöhnlich mit schwarzen und weißen Steinen auf einem 19x19 Linien großen Spielfeld gespielt.[1] Das generelle Ziel des Spiels ist es mit den eigenen Steinen ein möglichst großes Gebiet freier Schnittpunkte abzugrenzen. Zusätzlich können gegnerische Steine durch vollständiges besetzen aller angrenzenden Schnittpunkte gefangen werden. In diesem Falle wird der Stein vom Spielfeld entfernt und der fangende Spieler erhält einen zusätzlichen Punkt.

Obwohl der Großteil der Go-Spieler weltweit im ostasiatischen Raum zu finden sind (insbesondere in Japan, Korea und China) gibt es in Deutschland über 30.000 Go-Spieler. Die Anfänge des Go Spiels in Deutschland gehen auf eine Go-Gruppe in Berlin zur Zeit der Weimarer Republik zurück. Inzwischen ist es möglich an nahezu jedem Wochenende des Jahres ein Go Turnier in Deutschland zu besuchen.[2]

1.2 Intention der App

Die Android-App, die vorläufig den Namen „GoApp“ trägt, soll gleich mehrere Funktionalitäten rund um das Spiel Go bieten. Als eine der Hauptfunktionen soll das Spielen des Spiels zwischen zwei Spielern ermöglicht werden. Das mobile Gerät dient dabei als Ersatz für das Spielbrett. Die beiden Parteien sitzen sich dabei gegenüber und setzen die virtuellen Steine auf dem abgebildeten Spielbrett in der Mitte. Funktionalitäten zum Aussetzen eines Zuges oder zum Aufgeben, sollen dabei für beide Spieler gleichermaßen gegeben sein. Weiter soll es die App ermöglichen, real gespielte Go-Spiele aufzeichnen zu können. Die Steine werden dabei parallel zum realen Spielbrett auf dem Virtuellen in der App gesetzt. Nach dem Aufzeichnen soll es dann möglich sein, die einzelnen Spielzüge nachzuvollziehen, um die eigene Spielweise und Strategie evaluieren zu können. Weiter soll es möglich sein, Go-Spiele, die über andere Medien im SGF-Format gespeichert wurden mit Hilfe der App zu laden und ggf. daran weiter zu spielen oder die einzelnen Spielzüge durchzusehen.

Im zweiten Teil des Moduls sollte die Anbindung an einen Server geschaffen werden, sodass Spiele auch über eine Netzwerkverbindung möglich sind. Dabei sollte ein Nutzer eine Spielanfrage stellen können, wobei der Server entsprechend einen Spielpartner für jenen Spieler sucht. Sollte ein anderer Spieler zur selben Zeit (oder in einem Intervall von 5 min)

1.3 Technische Voraussetzung zur Benutzung

Grundsätzlich benötigt die App keine zusätzlichen Installationsschritte um ordnungsgemäß zu funktionieren. Es sei allerdings anzumerken, dass für die „Laden“ Funktionalität, .sgf Dateien im Ordner „SGF_files“ des External Storage Public Directory des Geräts liegen sollten, da andernfalls keine Spiele zum Laden zur Auswahl stehen. Einige Beispiel .sgf Dateien sind im „doc“ Ordner des Github Repositorys hinterlegt.

Das online Spielen hingegen erfordert eine laufende Instanz der Serversoftware. Die benutzte Technologie ist ein Apache Tomcat 7.0 Server in Kombination mit der JAX-RS Implementierung Jersey unter JavaEE 7. Als Datenbank wird eine MariaDB 10.1.19 verwendet. Eine SQL-Datei zum Aufsetzen der Datenbank ist im Repository https://github.com/kampp/GoServer.git im „docs“ Ordner zu finden. Ein geeignetes Tool für das Anlegen von benötigten Tabellen ist z.B. HeidiSQL. Für die Verwendung der App in Verbindung mit einem lokalen Server, ist in der clientseitigen Klasse „NetworkController“ der String SERVERURL mit der entsprechenden lokalen IP-Adresse des den Server ausführenden Rechners

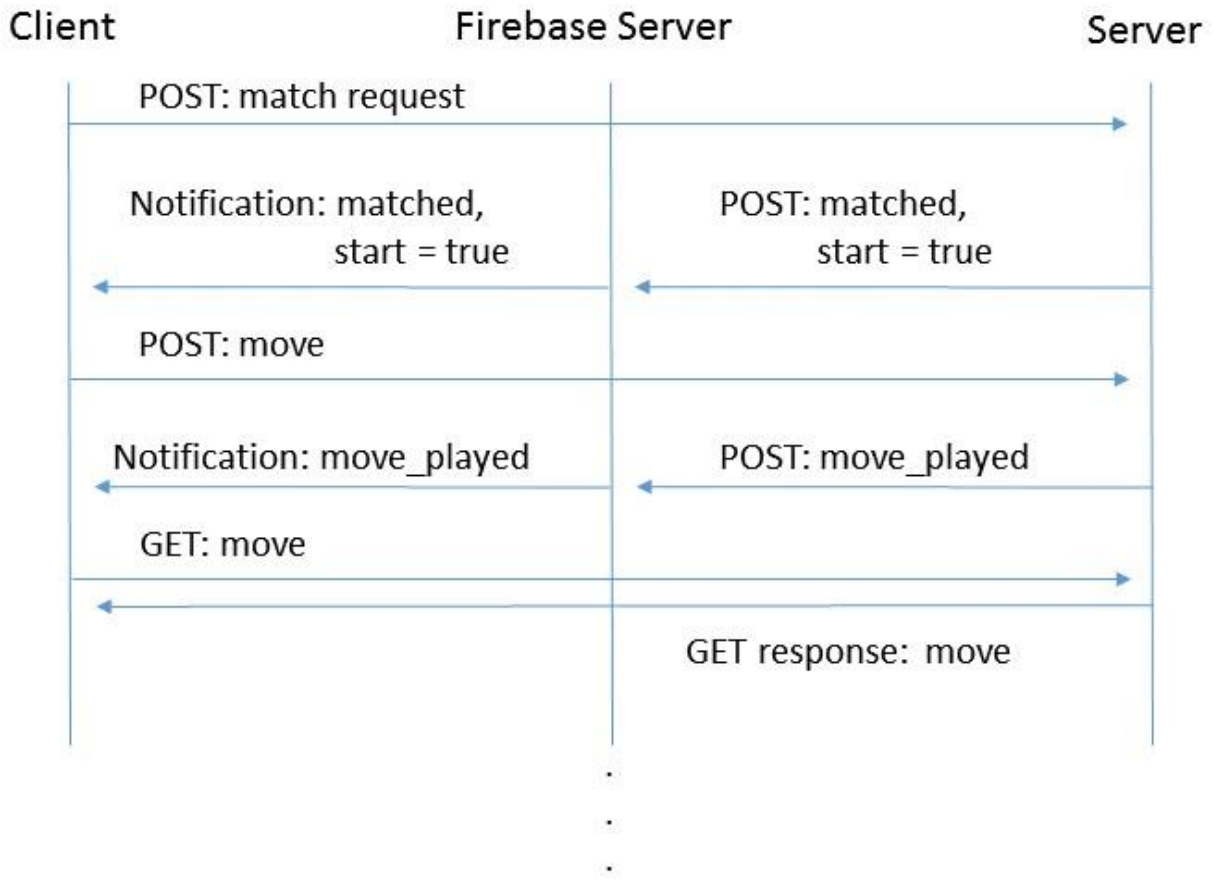
[1] 13x13 und 9x9 sind weitere Brettgrößen, auf welchen besonders Anfänger oft spielen.
[2] Quellen: www.dgob.de / www.go-baduk-weiqi.de

anzupassen. Diese kann unter Windows mit dem Powershell-Kommando „ipconfig“ in Erfahrung gebracht werden. Unter MacOSX funktioniert dies mit dem Bash-Kommando „ipconfig getifaddr en1“, wobei en1 durch das entsprechende WLAN Netzwerkinterface ersetzt werden sollte. Unter Ubuntu ist dasselbe mit dem Bash-Kommando „ifconfig wlan0 | grep inet“ zu erreichen, wobei auch hier wlan0 durch das entsprechende WLAN Netzwerkinterface ersetzt werden sollte. Die Datenbank kann durch das ausführen der Datei mysqld im bin Ordner der MariaDB Installation gestartet werden. Sobald die Komponenten Apache Tomcat und MariaDB gestartet sind und der entsprechende String in der Klasse NetworkController auf die Adresse des Servers gesetzt wurde kann die App verwendet werden.

2 Konzept

Um für den Fall einer Veröffentlichung der App ein größeres Publikum zu erreichen wurde entschieden, dass die App vollständig in Englischer Sprache verfasst werden sollte.

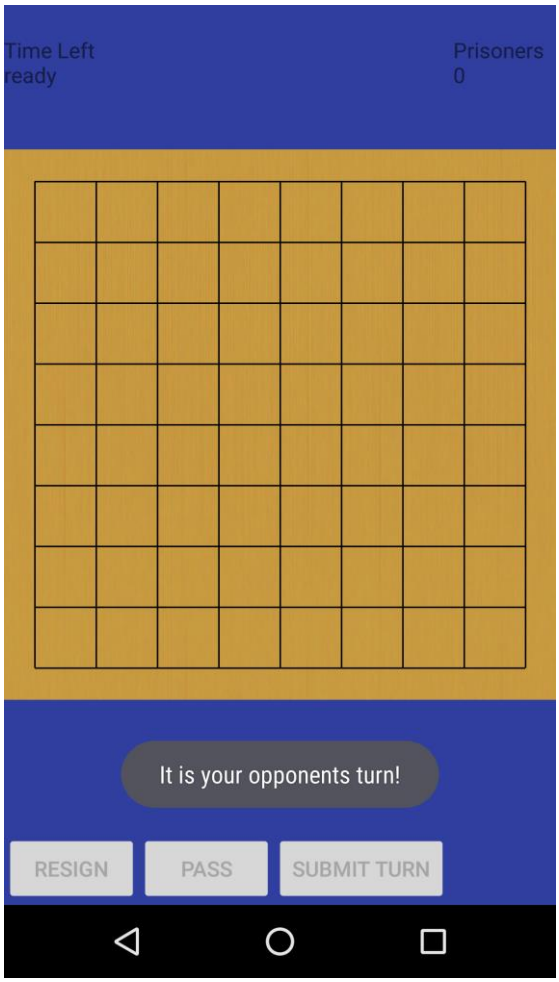
Im Falle der Netzwerkanbindung der im vorherigen Modul erstellten „GoApp“ wurde auf ein Client-Server Modell zurückgegriffen, welches im Folgenden in Form eines „Weg-Zeit-Diagramms“ dargestellt ist:



Eine zusätzliche Anbindung an Google’s Firebase Server war notwendig, da eine Benachrichtigung der an einem Spiel teilnehmenden Clients andernfalls sehr schwierig umsetzbar gewesen wäre. Um trotz der Nutzung von Firebase eine REST-Schnittstelle im Sinne des Moduls anzubieten und umzusetzen, wurde entschieden, die Datenhaltung und das Versenden von Zuginformationen nicht über Firebase umzusetzen.

2.1 Design

Das grundlegende Design der Client-Anwendung hat sich im Vergleich zu dem vorangegangenen Projekt nicht geändert. Es sind allerdings einige zusätzliche Funktionalitäten auf clientseitig hinzugekommen. Diese beinhalten vor allem die Activities „ActivityOnlineGame“ und „ActivityPlayOnline“, sowie Klassen zur Verarbeitung von Netzwerkverarbeitung. Letztere teilen sich in Firebase-bezogene Klassen, sowie in http-kommunikations-bezogene Klassen auf. Im Folgenden werden zunächst die neu erstellten Aktivities vorgestellt.



Beim Design der beiden neu hinzugekommenen Screens wurde sich größtenteils an der vorherigen Anwendung orientiert, um keine Brüche zu erhalten. So wurde die ActivityOnlineGame ebenfalls in Form von einer Auswahlmaske erstellt. Inhalte wurden insofern angepasst, als das der Nutzer nun sowohl seinen Nicknamen (unter dem er / sie für den Gegner bekannt sein wird), die Brettgröße, als auch seinen Rang angeben kann. Im Falle des Rangs besteht für den Nutzer die Möglichkeit aus Kyu(Schüler)-Rängen und Dan(Meister)-Rängen zu wählen. Das Scrollrad zur Auswahl des Ranglevels wird entsprechend angepasst, da für Kyu-Ränge die Werte 30 (absoluter Anfänger) bis 1 (moderater Spieler), sowie für Dan-Ränge die Werte 1 bis 7 (nahezu Profi) zur Verfügung stehen.

Für die Anzeige eines Spiels wird im Grunde eine Boardview (letztes Semester) benutzt, wobei das Layout angepasst wurde, um das Abschicken eines Zugs zu ermöglichen. Die Entscheidung einen Zug abschicken zu müssen wurde bewusst getroffen, um „Fehlklicks“ und den Aufwand jene zurückzunehmen zu minimieren.

2.2 Herangehensweise

Um die Arbeit im Team möglichst produktiv zu gestalten wurde ein informeller Scrum-Prozess verwendet. Dieser bestand darin, dass in wöchentlichen Treffen der aktuelle Stand des Projekts erläutert wurde und entsprechend der erfüllten Aufgaben bzw. neuen Anforderungen neue Ziele gesteckt wurden.

Im Rahmen des Projekts wurde mit dem Versionsverwaltungssystem Git gearbeitet. Dieses wurde unter anderem gewählt um das Arbeiten an gemeinsamem Code einfacher zu gestalten und Features wie zum Beispiel Branching nutzen zu können.

Die öffentlichen Git Repositories des Projekts sind unter https://github.com/kampp/GoApp.git bzw. unter https://github.com/kampp/GoServer.git zu finden.

3 Umsetzung

Für die Umsetzung des Servers wurden drei Pakete erstellt:

- Gogame
- Goserver
- Resource

In Gogame sind die Datenklassen zum Speichern von Go-Spielen enthalten. Diese unterscheiden sich nur minimal von den entsprechenden Klassen in der Clientanwendung.

3.1 Goserver

Im Paket Goserver sind generell Funktionalitäten welche ausschließlich den Server betreffen implementiert.

Die Klasse DBConnector ist eine Singleton Klasse, welche sich um die generelle Verwaltung von der Datenbank kümmert. Hier sind sowohl Methoden für die Konnektivität zur eigentlichen Datenbank als auch für das Einfügen, Löschen und Updaten von Datensätzen implementiert. Im speziellen sind Methoden vorhanden, um Match-Anfragen, Spiele und einzelne Züge einzufügen. Zusätzlich sind Methoden implementiert, um beim Auffinden eines Spiels durch seine GameID oder durch das token eines Spielers behilflich zu sein.

Die Singletonklasse GameController beinhaltet eine Methode um aus einer GameMetaInformation Instanz und den tokens der beiden Spieler ein neues online Spiel zu erzeugen. Des Weiteren sind in dieser Klasse analog zu der entsprechenden Klasse in der App Funktionalitäten vorhanden um die Validität von Zügen und eventuelle Gefangene zu bestimmen.

Die Klasse Matcher ist dafür gedacht im Hintergrund des Servers ausgeführt zu werden, um für eventuelle Match-Anfragen einen Gegner zu finden. Dies wird dadurch erreicht, dass die Klasse das Runnable-Interface implementiert und dadurch permanent ausgeführt werden kann. Match-Anfragen werden genau dann einander zugeordnet, wenn die Brettgröße übereinstimmt und keine der beiden Anfragen außerhalb der Timeout Zeit liegen (fünf Minuten).

3.2 Resource

Das Resource Paket ist der Kern des REST Webservices indem es Interfaces und Klasse bereitstellt um auf http Anfragen reagieren zu können, bzw. den aufrufenden Entitäten Ressourcen zur Verfügung zu stellen. Die Klasse teilt sich essentiell in zwei grundlegende Komponenten auf: Das Initiieren von Firebase-Benachrichtigungen, sowie den REST Webservice Endpunkten.

Die Klasse FirebaseMsgService implementiert das initiale Senden von Benachrichtigungen an den Firebase-Server. Bis auf die Benachrichtigung, dass ein Match gefunden wurde werden keine Nutzlasten mit den Benachrichtigungen übertragen.

Die Klassen MatchResource und PlayResource, sowie die dazugehörigen Interfaces IMatchResource und IPlayResource stellen die REST-Endpunkte dar. Die Spezifikation der REST-Endpunkte ist im Ordner doc des Serverrepository's in der Datei api.raml zu finden.

4 Fazit und Ausblick

4.1 Rückblick

Aufgrund eines generell sehr hohen Arbeitsaufgebots während der Projektlaufzeit hat es sich als sehr schwierig herausgestellt die vollständige Funktionalität der Server- und Clientfunktionalität umzusetzen. Rückblickend hätte eine vollständige und von vorneherein feststehende Spezifikation der API die Arbeit sicherlich vereinfacht.

4.2 Offene Punkte

Zum Zeitpunkt der Abgabe sind nicht alle vorher geplanten Features umgesetzt. Folgende Punkte sind noch in zukünftigen Versionen der App und des Servers zu gestalten:

- Anzeigen der Variationen eines Spiels innerhalb des aufzeichnen Modus
- Implementierung der Ko Regel
- Anzeigen des Endzustands gespeicherter Spiele als Bitmap
- Überprüfung der einzelnen Züge auf Serverseite
- Verschicken von Zügen auf Clientseite
    - Dabei handelt es sich um einen Fehler, wobei Zuginformationen nicht versendet werden, obwohl entsprechende Volley-Funktionen aufgerufen werden
- Möglichkeit bei einem online Spiel ein Zeitlimit zu setzen
- „Matchmaking" von Spielern anhand ihres Rangs nicht nur anhand der Brettgröße
- Nutzung von https
- Abbrechen der Suche nach einem Mitspieler