

**Technische Dokumentation**



Inhaltsverzeichnis

[Datenbank 2](#_Toc446053377)

[Model 2](#_Toc446053378)

[Events 2](#_Toc446053379)

[Statische Events 3](#_Toc446053380)

[Bewegliche Events 3](#_Toc446053381)

[Webservice 3](#_Toc446053382)

[Generelle Aufgaben 3](#_Toc446053383)

[Session Management 3](#_Toc446053384)

[Response Object 3](#_Toc446053385)

[Engine 3](#_Toc446053386)

[C# WPF Client 3](#_Toc446053387)

[Registrieren: 3](#_Toc446053388)

[Login: 4](#_Toc446053389)

[Dorf Window: 4](#_Toc446053390)

[Karte: 4](#_Toc446053391)

[Zusatzinformationen: 4](#_Toc446053392)

[Android Client 4](#_Toc446053393)

[Login 4](#_Toc446053394)

[Zusatzinformation 4](#_Toc446053395)

# Einleitung

# Datenbank

Die Datenbank ist das Gehirn unseres Projekts. Sie speichert nicht nur die Daten, welche für das Spiel wichtig sind, sondern wir nutzen auch die Fähigkeit der Oracle Datenbank um die Vorgänge im Spiel auszuführen. Diese Funktionen und Prozeduren übernehmen die Verwaltung der Dörfer, wie z.B. das Ausbauen von Gebäuden. Grund für die Serverseitige Lagerung der Funktionalität sind unteranderem das automatisch implementierte Transaktionsmanagement.

## Model

Neben den offensichtlichen Daten wie Dörfer Nutzer und Clan existieren auch die Eventtabellen. Diese Tabellen repräsentieren alle Geschehnisse, die vom Benutzer initialisiert werden können.

## Events

In der Datenbank wird zwischen Statischen und Beweglichen Events unterschieden. Diese sind eine Generalisierung eines Events welches eine Beginn Zeit und eine Dauer in Sekunden speichert.

### Statische Events

Diese Events beziehen sich nur auf ein Dorf und beinhalten entweder einen Gebäudetyp oder eine Ressource. Mit diesen Daten kann eine Prozedur sehr einfach ein Gebäude ausbauen oder auch Truppen ausbilden.

### Bewegliche Events

Im Gegensatz zu den Statischen Events beziehen sich die Bewegungsevents auf 2 Dörfer. Sie sind für Handel, Angriff oder Unterstützung zuständig.

# Webservice

Der Webservice ist die Schnittstelle zwischen den Clients (C# WPF und Android) und der Oracle Datenbank. Des Weiteren übernimmt der Webservice das Session Management.

## Generelle Aufgaben

Alle, vom Client ankommende Anfragen, rufen eine gewisse Methode auf, die entweder eine Abfrage oder eine Datenbankfunktion ausführt. Des Weiteren wird der, von der Datenbank gesendete Response an den Client weitergeleitet. Die Abfragen an die Datenbank werden natürlich, um SQL-Injektionen zu vermeiden, mit Prepared-Statements realisiert.

## Session Management

Eine weitere Aufgabe des Webservice ist, das Session Handling. Viele, vom Client gesendete Anfragen brauchen eine bestimmte Authentisierung. Um zu verhindern, dass ständig Username und Passwort mitgesendet werden müssen, wird dieser Vorgang auf den einmaligen Austausch beim Login beschränkt. Bei diesem Vorgang wird eine Session-ID generiert. Diese identifiziert den Benutzer und hat nur eine temporäre Gültigkeit. Der Webservice verzeichnet alle aktiven Sessions und deren zugehörige Spieler. Diese ID wird durch einen Hash von Nutzername, Passwort und Zeitstempel generiert und bei jeder, zu authentisierenden Anfrage mitgesendet.

## Response Object

Jeder Webservice liefert ein sogenanntes Response Object zurück. Das Response Object besteht aus einen Boolean welcher preisgibt ob der Request erfolgreich war. Wenn ja wird in einem Object das requestete Object zurückgeliefert. Wenn nein wird in einen String eine Error-Message zurückgeliefert.

# Engine

Die Engine ist dafür verantwortlich, die von der Datenbank zur Verfügung gestellten Prozeduren zur Ausführung von abgelaufenen Events sowie dem kontinuierlichen Generieren von Ressourcen für den Spieler je nach Auslastung in angemessenen Abständen aufzurufen. Dies geschieht durch dauerndes updaten bestimmter Tabellen.

# C# WPF Client

Das Juwel unseres Projekts ist der C# WPF Client. Abgesehen von seinem einmaligen Design ist auch die Funktionalität eine Augenweide.

## Registrieren

Der C# WPF Client bietet dem Benutzer die Möglichkeit sich zu registrieren. Dafür verschlüsselt der Client das Passwort und überträgt Benutzername und Passwort an den Webservice, welcher den Benutzer in der Datenbank registrieren lässt.

## Login

Beim Login wird, wie beim Registrieren das Passwort verschlüsselt und mit dem Benutzernamen an den Webservice weitergesendet. Dieser lässt die Datenbank prüfen ob es einen Benutzer mit diesem Benutzernamen und diesen Passwort gibt. Als Response erhält der Client ein Response Object welches entweder die Session ID oder die Fehlermeldung beinhaltet.

## Dorf Window

Sobald sich ein Benutzer erfolgreich angemeldet hat, holt sich der Client, über den Webservice, mithilfe der Session ID die Dorfspezifischen Daten wie Dorf Name, Ressourcen, Truppen und Gebäude und zeigt diese an. Bei Klick auf den Ausbauen Button im unteren Datagrid wird ein Call an den Webservice gesendet, welcher die Prozedur in der Datenbank auf aktiv setzt und die benötigten Ressourcen abzieht. Der Client aktualisiert die Daten und startet einen Thread welcher in der rechten unteren Liste einen Countdown startet. Dieser Countdown zeigt an wie lange die Engine noch mit der Ausführung des Events wartet. Sobald dieser Timer auf 0 gesetzt ist, führt die Engine die Prozedur aus, das Gebäude wird ausgebaut und der Client aktualisiert erneut die Oberfläche und zeigt von nun an die aktuellsten Ressourcen, Truppen und Gebäude mit den passenden Level an.

Bei Klick auf den Karte Button in der Menüleiste öffnet sich das Fenster für die Karte.

## Karte

Sobald dieses Fenster erzeugt wird, werden alle umliegenden Dörfer, mithilfe des Webservice, aus der Datenbank geladen und in eine Collection gespeichert. Nun wird die ganze Collection durchlaufen um jedes Dorf, in Form eines Pins auf der Karte anzuzeigen. Jeder Pin hat zusätzlich noch ein OnClickEvent mit welchem, über die ID des Dorfes, mithilfe des Webservice ein Call auf die Datenbank gemacht wird mit der Information zu diesem Dorf zu erhalten.

## Zusatzinformationen

Der Deserialisierungsvorgang und das Casten des Response Object einer Collection gestaltete sich in C# ziemlich kompliziert. Daher wurde zuerst das Response Object an sich deserialisiert um den Boolean auf True zu überprüfen. Danach wurde der json String so aufgeteilt das nur noch der String, welcher die Collection beinhaltet, übrigbleibt.

# Android Client

Der Android Client läuft ab Version 4.2 und ist ein Zusatzdienst zum eigentlichen Spiel. Grundsätzlich besteht der Android Client aus verschieden Boards (Themen) und dazugehörigen Kommentaren.

## Login

Das Login wurde mit dem gleichen Webservice-Call wie beim C# Client realisiert.

## Main Activity

Auf der Main Activity befindet sich zwei Fragements. Boards List und Board Detail welche abwechselnd angezeigt oder versteckt werden.

### Boards List

Auf diesem Fragement befindet sich eine Listview welche, mit Daten von dem Webserivce Call BoardList, gefüllt wird. Zusätzlich befindet sich in der oberen Ecke ein Button welcher eine Inputbox öffnet um Daten für neues Board zu erfassen. Daraufhin wird ein Webservice aufgerufen um ein neues Board zu erstellen.

### Board Detail

Bei Klick auf einen Listeneintrag der Board List wird das Fragement Boards List versteckt und das Fragment Board Detail angezeigt und mit Daten von dem dazugehörigen Webservice gefüllt. Am unteren Ende des Fragement befindet sich eine Input Box in welcher ein Kommentar zum derzeitig ausgewählten Board abgesendet werden kann.

# Zusatzinformation

Bei der Projektplanung wurde auf die Skalierbarkeit der Software viel wert gelegt. Da dieses Projekt ein sehr komplexes Online-Massiv-Multiplayer-Strategiespiel ist wurde darauf geachtet, dass es einerseits ein ansprechendes Grafikdesign als auch eine leicht zu erweiternde Funktionalität der Software implementiert wird.

Events wie zum Beispiel die Ausbildung der Truppen sind schon auf der Datenbank und teilweise am Webservice realisiert, sind jedoch noch nicht an Client umgesetzt. Jedoch wurde im finalen Sprint hauptsächlich darauf geachtet eine vernünftige Event-Infrastruktur geachtet welche sich durch alle Schichten des Projektes zieht. Um diese Infrastruktur zu testen wurde das Gebäude Ausbau Event im Client (sowie in allen anderen Schichten) implementiert