# Ziel des Sprints (17.12)

Hauptaufgabe war es, den Client auf den Web Service umzubauen und diesen zu implementieren. Nebenaufgabe war das vorbereiten für die Auswertung von Events.

# Events

In der Datenbank wurde das Bauen von Gebäuden implementiert. Die Funktion wird nun als PL/SQL Prozedur angeboten. So kann im nächsten Sprint sehr einfach ein Webservice mit dieser Funktion erstellt werden.

Die Prozedur behandelt die verschiedenen Fälle von aufleveln, unbekannten und nichtexistierenden Gebäuden.

Erstellt wurden die folgenden Prozeduren und Funktionen:

* BAUE\_GEBÄUDE\_IN\_DORF erstellt ein Gebäude von angegebenem Typ in angegebenem Dorf, besteht es bereits wird es aufgelevelt.
* CREATE\_GEBÄUDE\_BAU erstellt ein Event. In diesem werden alle Daten gespeichert, die BAUE\_GEBÄUDE\_IN\_DORF benötigt um aufgerufen zu werden. Sie errechnet außerdem automatisch die Dauer des Events, abhängig von Gebäude und Level.
* Event Prozedur: liest alle Events aus; Wertet Art des Events aus & ruft Folgeprozeduren auf die die Daten der DB je nach Event verändern und löscht Events dann aus der DB

Außerdem wurde die Engine erweitert um die Event-Prozedur in gewissen Abständen aufzurufen.

# Webservice

Der Webservice soll vorerst nur den Client mit den nötigen Informationen versorgen. Im nächsten Sprint werden Calls für die bereits, implementierten Events angefertigt.

## Credentials

Hier werden per PUT neue Nutzer angelegt und mit GET angemeldet.

Bei anmelden müssen User Name und ein Hash des Passworts übertragen werden. In einem zukünftigen Sprint wird der Webservice beim Anmelden eine SessionID zurückgeben. Diese muss dann zur Autorisierung bei DorfDetailFull mitgesendet werden.

GET liefert ein sogenanntes LoginStatus Objekt. Darin befinden sich:

* Status
* Session ID (zukünftig: generiert, zurzeit noch ein Testwert)
* LastLogin (zukünftig: aktuelles Datum zurzeit des Calls, zurzeit noch ein Testwert)

## DorfDetailFull

Hier werden die Daten eines Dorfes gesammelt zurückgegeben um die Dorfübersicht mit Daten zu versorgen. Dieser Service soll nur dem User zur Verfügung stehen. Dies wird aber erst in einem späteren Sprint eingebaut.

Folgende Daten werden geliefert:

* ID
* Name
* Liste aller Gebäude, samt Level
* Rohstoffe
* Truppen des Dorfbesitzers im Dorf
* Truppen die als Unterstützung von anderen Spielern dort stationiert wurden.

## Map

Hier erhält man die Daten die man benötigt um alle Dörfer am Client anzuzeigen um die Übersichtskarte mit Informationen u versorgen. Dieser Webservice wurde angefertigt, da bei einem GET-Call von DorfDetailFull sehr viele, für die Übersichtskarte unnötige, Informationen geschickt werden. Außerdem soll während Betreib des Spieles alle vorhanden Dörfer auf der Karte angezeigt werden, deshalb wurde ist DorfMap eine sehr stark abgespekte Version von DorfDetailFull, welche nur die Koordinaten sowie die Id, für den späteren Aufruf für DorfDetailMap liefert. Alle Dörfer werden zurückgeliefert.

Folgende Daten werden geliefert:

* ID
* Position (X/Y)

## DorfDetailMap

Da bei Click auf ein Dorf auf der Übersichtskarte, grundlege, für alle ersichtliche Informationen des Dorfes, angezeigt werden. Dies ist eine kürzere Version von DorfDetailFull. Hierbei werden wieder nur die nötigen Informationen zur Verfügung gestellt.

Folgende Daten werden geliefert:

* ID
* Name
* Liste aller Gebäude, samt Level
* Rohstoffe
* Truppen des Dorfbesitzers im Dorf
* Truppen die als Unterstützung von anderen Spielern dort stationiert wurden.

# Problem

Es war uns nicht möglich den Client, von direkter Datenbankverbindung, zu Webservice verbindungumzubauen. Wir gingen am Anfang des Sprints davon aus das wir vom Webservice XML zum Client schicken. Dies ging bei der Überprüfung der Credentials noch gut, da wir nur eine Zusätzliches Tag für den, in C# benötigte, Namespace hinzufügen mussten. Daraufhin bauten wir unsere anderen Webservice ebenfalls um, damit sie den Namespace zum XML addierten. Nach dem Login ist es nötig die Dorf-Übersicht mit Daten zu versorgen gingen die Probleme los.

Der Spieler besitzt eine Collection von Dörfern, welche in Java folgend serialisiert wird:

<Namespace>

<Element>

Properties...

</Element>

</Namespace>

Für die korrekte deserialisierung von XML in C# wird ein zusätzlicher Collection-Tag benötigt. Durch eine zusätzliche Klasse in Java war es uns möglich einen solchen Tag zum XML hinzuzufügen. Da jedoch das Umbauen des gesamten Webservice zu aufwendig ist, entschieden wir uns nur die „Produces“-Anotaions der betroffenen Services auf JSON serialisierung zu ändern, da deserialisierung von JSON in C# weitaus komfortabler ist als jenen von XML. Am Anfang des nächsten Sprints ist, steht als erstes der Umbau unserer C#-Deserialisierer-Klasse auf dem Programm.

JAVA Webservice sowie C# Client (inklusive XML-Deserialiesirungs-Klasse) stehen auf dem GitHub zur Verfügung.