
TIMA

# Abschlussarbeit

## TIMA

Nathanael Philipp, Felix Rauchfuß, Kai Trott

12. August 2015

# Inhaltsverzeichnis

# 1 Einleitung

# 2 Assoziationsdatenbank und Website

Der Hauptteil von TIMA ist die Datenbank in denen die Assoziation gespeichert werden. Diese ist direkt verknüpft mir dem Webfrontend, dass sowohl der Hauptanlaufpunkt für die Benutzer ist als auch für die Apps durch die Bereitstellung einer umfassenden API.

Im nächsten Abschnitt wird das Backend und die Datenbank genauer beschrieben. Dabei wird genauer auf den Aufbau der einzelnen Tabellen eingegangen, sowie die verschiedenen Designentscheidungen.

Anschließend wird die API und die dahinter stehenden Designentscheidung genauer erläutert.

## 2.1 Backend und Datenbank

Für das Backend der Website haben für Django als grundlegende Technologie entschieden. Bei Django handelt es sich um ein in Python geschriebenes Webframework, das dem Model-View-Controller-Schema folgt. Django bietet unter anderem einen sehr komplexen objektrelationalen Mapper, der es ermöglicht auch komplexe Objektstrukturen abzubilden ohne die verwendete Datenbank explizit zu kennen.

### 2.1.1 Datenmodell

In Abbildung 2.1 ist das komplette Datenmodell von TIMA dargestellt. Dabei bildet das Modul `associations.models` den Kern.

Das Modell `Language` repräsentiert die verfügbaren Sprachen der Wörter, das Modell `Word` die einzelnen Wörter, wobei ein Wort, wenn es in mehreren Sprachen existiert, für jede Sprache einen eigenen Eintrag hat und das Modell `Association` die Assoziation zwischen zwei Wörtern. Zusätzlich wird bei einem Wort gespeichert, wie oft dieses gefragt wurde und bei der Assoziation wir häufig diese gegeben wurde.

Die Modelle des Moduls `app.models` behandeln grundlegende Funktionen des Benutzermanagements. So zum Beispiel das Modell `Profile`, dass den Kulturellen Hintergrund, die Punktzahl und die Sprachen für die ein Benutzer assoziiert hat,

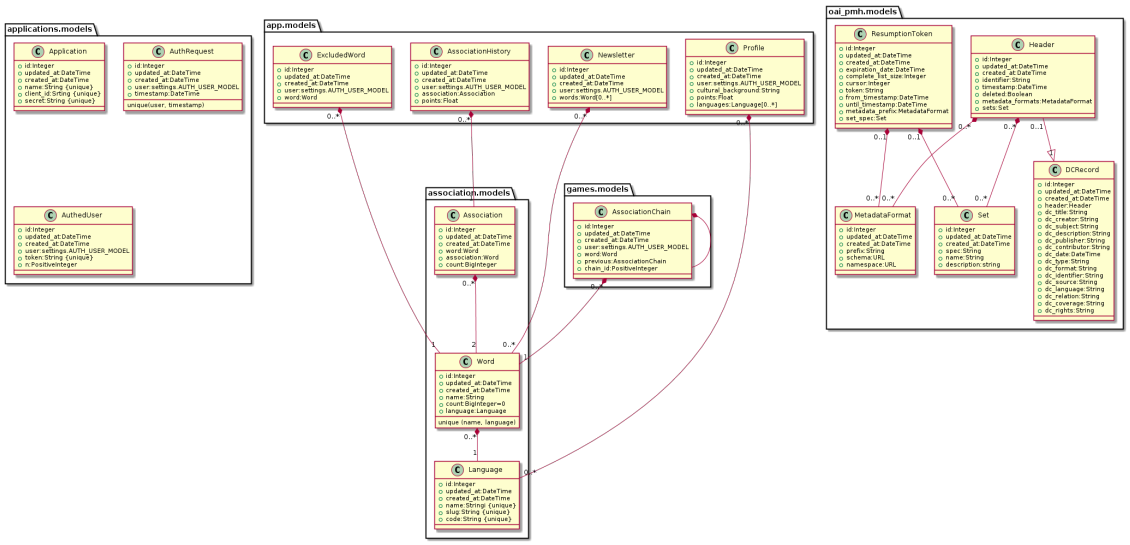


Abbildung 2.1: UML des TIMA Datenmodells

speichert. In dem Modell `AssociationHistory` wird die gesamte Assoziationsgeschichte eines Benutzers gespeichert, mit den jeweils für eine Assoziation erhaltenen Punkte. Das Modell `ExcludeWord` enthält für jeden Benutzer die Wörter, die er übersprungen hat (siehe TODO ref), diese werden automatisch nach sieben Tagen gelöscht. Das letzte Modell in diesem Modul speichert für jeden Benutzer welche Worte er in seinem Newsletter empfangen möchte.

Das Modul `games.models` enthält Modelle die für verschiedenen Spiele wichtig sind. Dies ist im Moment nur AssoziationsKette (vgl. TODO ref), hierfür werden in dem Modul `AssociationChain` die letzte/aktuelle Assoziationskette eines Benutzer gespeichert. Diese wird bei jedem Start eines Spieles gelöscht.

Für die Kommunikation zwischen App und Backend, insbesondere die schreib Zugriffe (vgl. TODO ref), zu autorisieren und dem speichert der nötigen Informationen dient das Modul `applications.models`. Das Modell `Applicaion` speichert die Apps, die Autorisiert sind, mit den nötigen Daten für die Autorisierung (vgl TODO ref). Die beiden anderen Modell in diesem Modul `AuthRequest` und `AuthedUser` speichern die nötigen Information für einen Benutzer der sich authentifizieren möchte und hat.

Das letzte Modul und die beinhalteten Modell sind für das OAI-PMH erforderlich siehe dafür (TODO ref).

## 2.2 Webfrontend

Das Webfrontend ist die Hauptanlaufstelle für Benutzer. Hierüber kann sowohl anonym als auch angemeldet Assoziationen eingegeben, Wörter und deren Assoziationen angesehen und weitere Funktionen (Rangliste, Statistik) aufgerufen werden.

Das Webfrontend basiert auf Django, wurde zusätzlich zu HTML mit Bootstrap und JQuery erstellt, sowie zur Visualisierung D3.

## 2.3 API

Damit die verschiedenen Apps mit der TIMA Datenbank kommunizieren können, haben wir uns entschieden eine umfangreiche API zu implementieren. Diese lässt sich grob in drei Teile. Zum einen gibt es die Anfragen, die keiner Autorisierung bedürfen, zweitens jeden die einer Autorisierung erfordern und drittens eine OAI-PMH Schnittstelle.

Eine komplette Dokumentation der API ist in der Datei API.md[1] im git zu finden.

Im folgenden Abschnitt werden die einzelnen API Anfragen die keiner Autorisierung bedürfen näher erläutern, im darauffolgenden Abschnitt die, die eine Autorisierung benötigen, dort wird ebenfalls der Autorisationsprozess näher erläutert. Zum Schluss wird dann noch ein Abschnitt zu OAI-PMH folgen.

### 2.3.1 Nicht autorisierte Anfragen

### 2.3.2 Autorisierte Anfragen

Zum einem soll man sich über die Apps auf die Anmelden können, zum anderen soll nicht jede beliebige App schreib zugriff auf die TIMA Datenbank haben. Aus diesem war er erforderlich, dass einige API Anfragen einer Autorisation bedürfen.

Um dies zu realisieren haben wir uns zunächst bestehende Frameworks wie zum Beispiel OAuth2 angeschaut und getestet in wie weit diese unseren Anforderungen genügen. Dies hat allerdings zu keinen zufriedenstellendem Ergebnis geführt, weswegen wir entschieden haben dies selbstständig zu implementieren.

Die grundlegenden Anforderungen die wir dabei hatten sind wie folgt:

1. sichere Authentisierung einer App
2. sichere Authentisierung eines Benutzers
3. sicherstellen das spätere Anfragen von einem Authentisierten Benutzer kommen

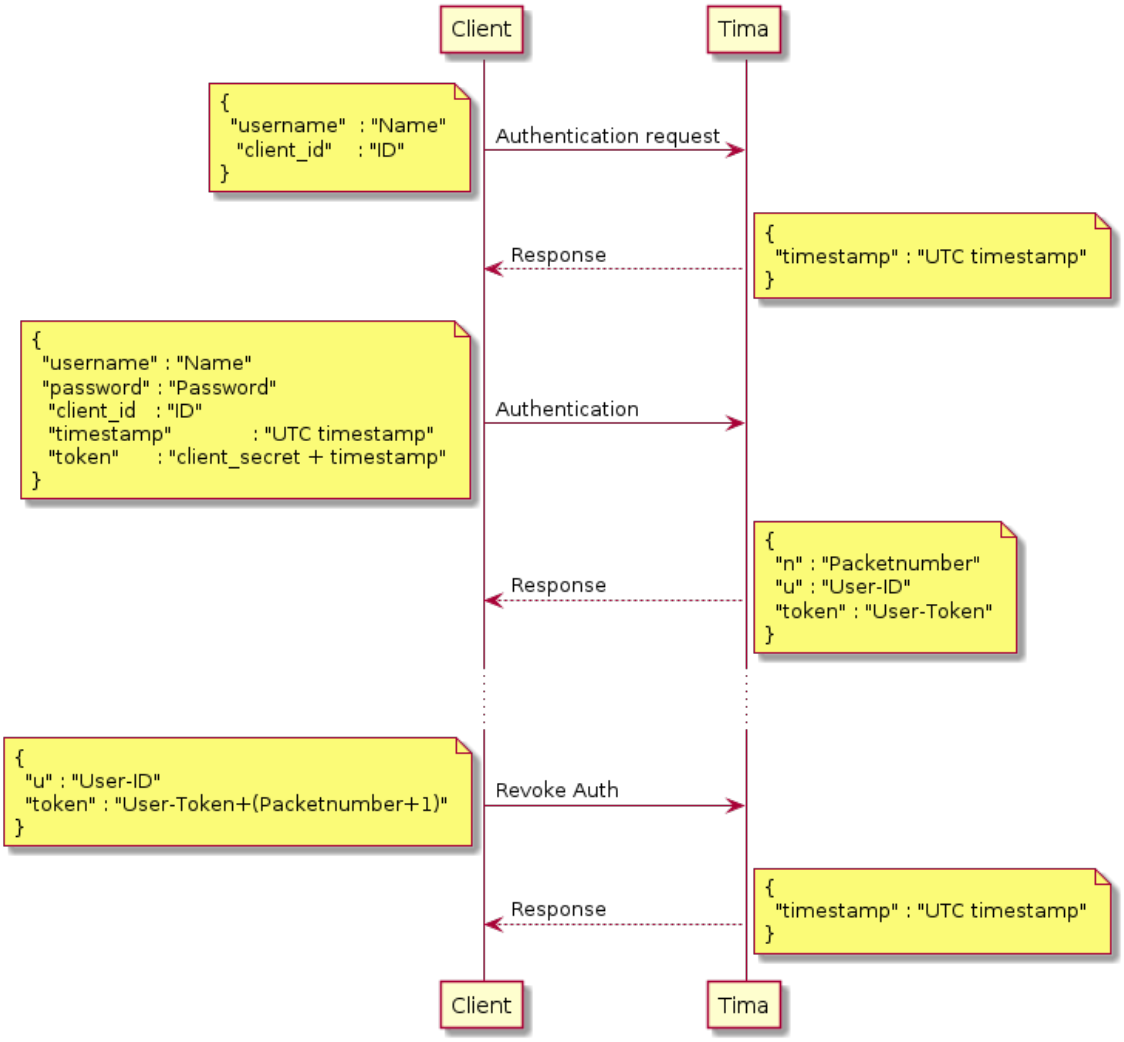


Abbildung 2.2: Authentisierungsprozess

In Abbildung 2.2 ist der Authentisierungsprozess schematisch Dargestellt. Der Client ist dabei eine App, ber die sich ein Nutzer authentisieren möchte. Die App verfügt zum einen über eine `client_id` und über ein `secret`, beides von TIMA vergebene eindeutige zufällige Strings. Der Authentisierungsprozess läuft wie folgt ab.

1. Eine App sendet eine Anfrage an TIMA mit dem `username` des Benutzers und der `client_id`. TIMA prüft diese beiden Werte auf Existenz und antwortet entweder mit **200** (HTTP Response Code) und dem aktuellen Zeitstempel oder mit **404**.

2. Als nächstes sendet die App die eigentliche Authentisierungsanfrage. Mit `username` und `password` des Benutzers, `client_id` der App, dem Zeitstempel der Antwort der letzten Anfrage und einem `token` das aus dem `secret` der App und dem Zeitstempel geniert wird (SHA512).

3. TIMA antwortet wenn die Authentisierung erfolgreich war mit **200** und den folgenden drei Werte:

   n Paketnummer jeden Anfrage einer App muss diese um eins nach oben zählen. Als Wertebereich ist uint32 zubenutzen.

   u eine eindeutige Benutzer-ID, die bei jeder Anfrage mit zusenden ist

   token einem zufälligen String der bei jeder Anfrage zusammen mit `n`, der Paketnummer, in einem SHA512 Hash zu senden ist

### 2.3.3 OAI-PMH

bisschen OAI-PMH

[1] `https://github.com/Tima-Is-My-Association/TIMA/blob/master/API.md`

# 3 App

grundlegende APP mit Qt/Qt-quick/QML

# 4 ausblick

# 5 Zusammenfassung