
TIMA

Abschlussarbeit

TIMA

Nathanael Philipp, Felix Rauchfuß, Kai Trott

26. 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,

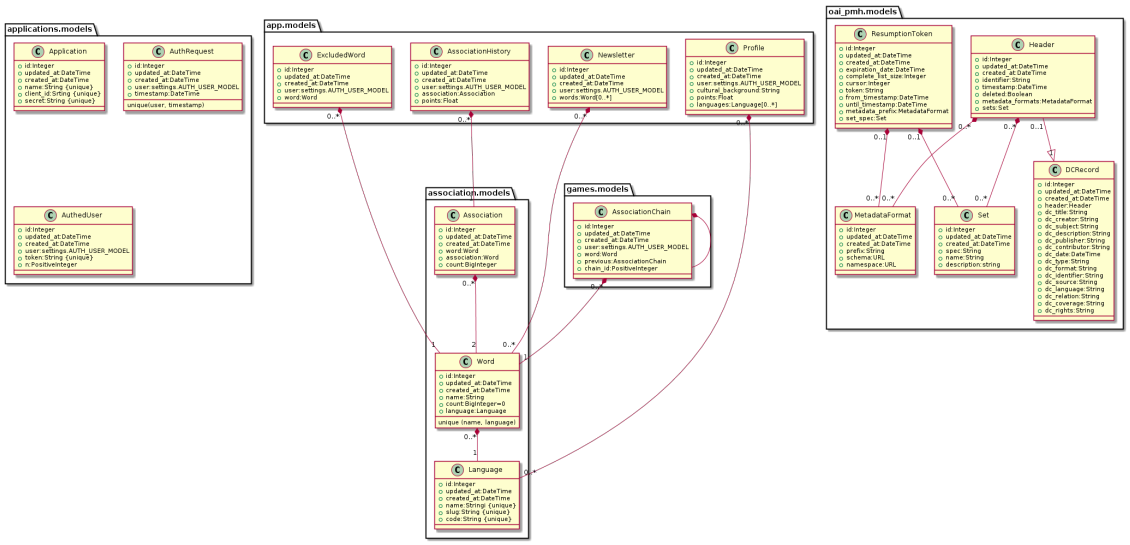

Abbildung 2.1: UML des TIMA Datenmodells

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

Die API-Anfragen, die keiner Autorisation bedürfen sind allgemeine Anfragen, an die Assoziationsdatenbank, die auch über die Webseite ohne eine Anmeldung erfolgen können.

Rangliste Eine dieser Anfragen ist die nach der Rangliste. Es werden keine weiteren Angaben benötigt und als Antwort kommt ein JSON-Object, das eine Liste der Benutzer enthält, mit den gleichen Daten wie sie über die Webseite einsehbar sind.

Statistik Ebenso ist die Statistik über die API abfragbar.

Sprachen Es kann eine Liste aller Sprachen in TIMA angefordert werden, hier ist neben dem Namen, der Sprach-Code in der Antwort enthalten, der bei vielen anderen Anfragen als Parameter angegeben werden muss.

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

Wörter Um entweder ein einzelnes Wort oder eine Liste von Wörtern ist diese Anfrage bestimmt. Es können optional Wort-IDs, Sprache oder ein Limit für die Anzahl der Assoziationen pro Wort angegeben werden. Das JSON-Objekt der Antwort enthält unter anderem zu jedem Wort einen Link zur Website des Wortes, ein Link zu dieser Anfrage mit der Auswahl auf das einzelne Wort und der OAI-PMH identifier des Wortes.

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

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:

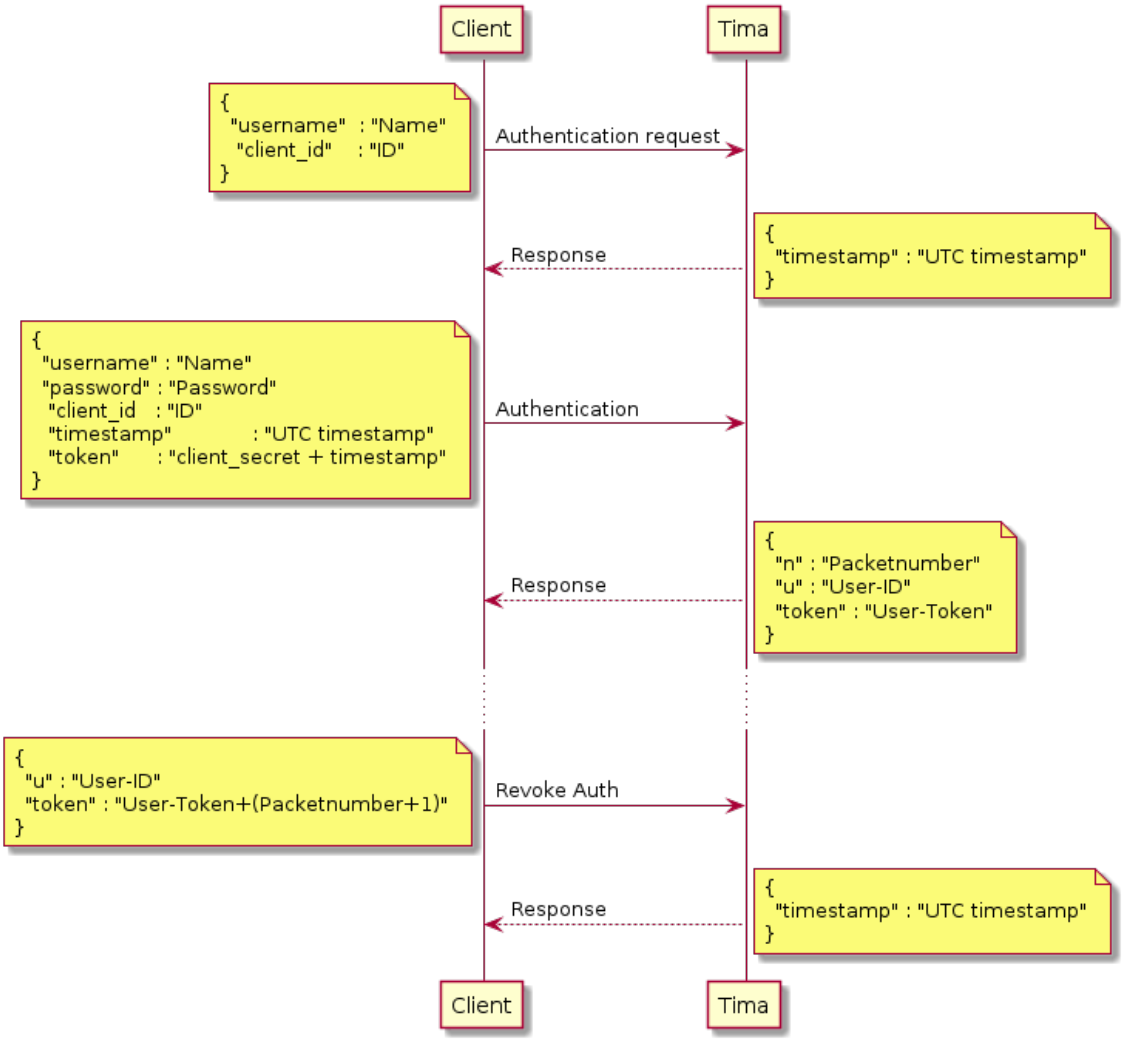

Abbildung 2.2: Authentisierungsprozess

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

Bei OAI-PMH (Open Archives Initiative - Protocol for Metadata Harvesting) handelt es sich um ein auf XML basierendes Protokoll zum sammeln von Metadaten. Diese haben wir implementiert und wir liefern darüber Metadaten zu den einzelnen Wörtern aus und in einem sehr geringem Maße über die Benutzer.

3 Applikation

Die Applikation soll die Verwendung vom TIMA ohne Webbrowser, für möglichst viele Endnutzer möglich machen. Besonders für Mobile Geräte soll die Nutzung vereinfacht werden.
Als erste Variante wurden grundlegende Funktionen der Webseite nachgebaut. Dies diente auch der Entwicklung dieser, um etwaige Fehler im Protokoll oder Verbesserungen dessen aufzuzeigen.

3.1 Framework

Als Framework wurde sich für Qt5 entschieden Aufgrund der weitreichenden Unterstützung des Frameworks von Endgeräten.
Es wurde zudem Java in betracht bezogen, da dies vorrangig auf Mobilen Geräten Verwendung findet. Allerdings wurde sich aus Sympathiegründen des Entwicklers dagegen entschieden.

3.2 Aufbau

Die innere Logik wird durch einen Zustandsautomaten dargestellt um Mehrfachanfragen zu vermeiden und eine einfache Fehlerkerrektur zu ermöglichen. In Abbildung 3.1 wird der zusammenhang der einzelnen Zustände angezeigt. Das wechseln der Zustände wir rein über die Qt eigenen Signale geregelt.

3.3 Sicherheit

Die Sicherheit hat bei der Entwicklung eine Große Rolle gespielt. Auch wenn das Projekt nicht unbedingt einer größeren Sicherheitsvorkehrung als SSL bedürfte, wurde sich für die Absicherung von Teilen der API durch unauthorisierte Dritte Abgesichert. Dies dient dazu, lediglich von TIMA akzeptierte Programme schreibrechte für die Datenbank zu geben. Das Auslesen der Informationen bliebt jedoch davon unangetastet und ist nach wie vor für jeden offen.

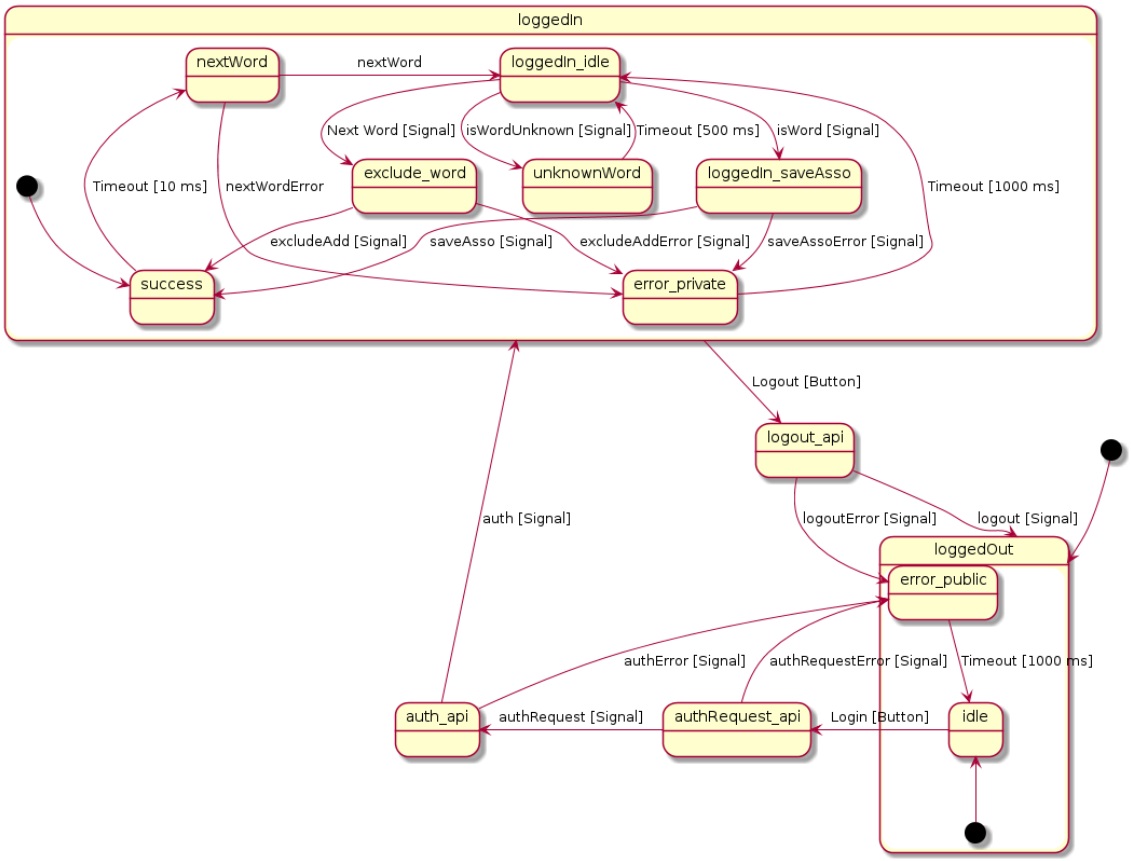

Abbildung 3.1: UML State Diagramm des Applikations Zustands-automaten

Sicherheitsplanung

In der ersten Version wurde eine Verschlüsselung mit PGP geplant um eine vollständige „Ende zu Ende“ Verbindung herzustellen und auch „Man in the Middle“ Attacken auszuschließen. Doch aufgrund von erhöhtem Programmieraufwand wurde hierauf verzichtet.

In der zweiten Version wurde eine Implementierung vom OAUTH2 Standart versucht. Die Anforderungen bestanden hierbei an einem geheimen Schlüssel in der Applikation, einem Auschluss einer Replay-Atacke und dem Verwenden von User eigenen Passwörtern.
Im Standart sind 4 verschiedene „Authorization Grants“ definiert[2] jedoch erfüllt keine davon jede Anforderung an das gewünschte Protokoll.

[2]siehe `https://tools.ietf.org/html/rfc6749#section-1.3`

Der Dritte Versuch ist eine Eigenentwicklung um sowohl den Arbeitsaufwand klein zu halten, als auch alle Anforderungen ausreichend zu erfüllen. Eine kleine Übersicht ist in Abbildung 2.1 zu sehen.
Die Übertragung erfolgt mit JSON Objekten um das Parsen zu erleichtern und einen üblichen Standart zu verwenden. Der Rechenaufwand des Servers wurde zudem versucht so gering wie möglich zu halten, um mögliche DDOS Angriffe anzuschwächen.

3.3.1 Authentisierung

Eine Authentisierung ist wie folgt aufgebaut. Der Client sendet eine Authentisierungsanfrage an den Server mit seinem Benutzernamen und einer Applikations-Identifikationsnummer. Der Nutzername dient dem Server zur Zuordnung zum Benutzerprofil und die Identifikationsnummer regelt die Zugehörigkeit der verwendeten Applikation. Es werden nur registrierte Applikationen akzeptiert.
Bei korrekten Informationen sendet der Server seine aktuelle Uhrzeit zurück und speichert diese unter dem angegebenen Nutzerprofil ab. Dieser Zeitstempel wird für die Signierung der weiteren Pakete verwendet und um nicht wiederkehrende Signaturen zu erzwingen.
Im nächsten Schritt erstellt der Client das eigentliche Authentisierungs Paket mit seinem Benutzernamen, seinem Passwort, der Identifikationsnummer, dem Zeitstempel und einem Token als Signatur, welcher aus dem Hash des Zeitstempels und dem konkatinierten Applikations-Geheimnis besteht. Als Hashmethode wurde hierbei SHA-512 verwendet.
Nachdem der Server alle Angaben überprüft hat und die Signatur korrekt verifizieren konnte, erhält er eine laufende Paketnummer, einen für ihn geltende Benutzer-Identifikationsnummer und einen Usertoken.
Für jede weitere API Abfrage muss mindestens die Benutzer-Identifikationsnummer und eine Signatur bestehend aus dem SHA-512 Hash des Usertoken konkatiniert mit der Paketnummer verschickt werden. Nach jeder erfolgreichen API Anfrage welche die Paketnummer und die Signatur verlangt, wird die Paketnummer um eins erhöht. Aus Kompatibilitätsgründen läuft die Paketnummer bis maximal 2147483646 und fängt danach wieder bei 0 an.

3.4 Spiele

Um mehr Nutzer zu erziehlen sind verschiedene Spiele geplant worden, die jedoch aus Zeitgründen nicht in den geplanten Zeitrahmen passten.

Als Grundlage wird jedoch in den meißten Fällen schon eine gut gefüllte Datenbank vorrausgesetzt.

Die einzelnen Spiele sind

- Familienduell
- Assoziationskette (Einzelspieler)
- Assoziationskette (Mehrspieler)

Familienduell

Da dieses Spiel die größte Beliebtheit verspricht, sei hier noch einmal kurz die Funktionsweise erklärt.
Der Name soll hierbei nur die Ausrichtung des Spieles näher bringen. Wie bei der Fernsehserie[3], werden Dem Benutzer verschiedene verdeckte Antworten auf eine Frage gezeigt und für jede richtig gegebene Antwort erhält der Spieler Punkte. Die Fragen sind jedoch ausschließlich die meißt genannten Assoziationen zu einem bestimmten Wort. Zusätzlich sollte eine Zeitbegrenzung eingehalten werden, da ja alle Korrekten Antworten auf TIMA nachgeschaut werden können.

[3] siehe `https://de.wikipedia.org/wiki/Familien-Duell`

4 Ausblick

5 Zusammenfassung