

TIMA

The logo features the word "TIMA" in a bold, dark blue, sans-serif font. Two bright blue arrows are integrated into the design: one starts above the letter 'M', extends horizontally to the right, and then curves downward to point at the top of the letter 'A'; the other starts below the letter 'A', extends horizontally to the left, and then curves upward to point at the bottom of the letter 'M', creating a continuous clockwise cycle.

Abschlussarbeit

TIMA

Nathanael Philipp, Felix Rauchfuß, Kai Trott

15. September 2015



Inhaltsverzeichnis

1	Einleitung	1
2	Assoziationsdatenbank und Website	2
2.1	Backend und Datenbank	2
2.1.1	Datenmodell	2
2.2	Webfrontend	4
2.2.1	Wortselektierungsalgorithmus	4
2.2.2	ExcludeWords	5
2.2.3	Spiele	5
2.2.4	Newsletter	5
2.3	API	5
2.3.1	Nicht autorisierte Anfragen	5
2.3.2	Autorisierte Anfragen	6
2.3.3	OAI-PMH	8
3	Applikationen	9
3.1	Framework	9
3.2	Aufbau	9
3.3	Sicherheit	9
3.3.1	Authentisierung	11
3.4	Spiele	11
4	Ausblick	13
5	Zusammenfassung	14

1 Einleitung

2 Assoziationsdatenbank und Website

Der Hauptteil von TIMA ist die Datenbank in denen die Assoziation gespeichert werden. Diese ist direkt verknüpft mit dem Webfrontend, das 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 wir uns 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 wie häufig diese gegeben wurde.

Die Modelle des Moduls `app.models` behandeln grundlegende Funktionen des Benutzermanagements. So zum Beispiel das Modell `Profile`, das den Kulturellen Hintergrund, die Punktzahl und die Sprachen für die ein Benutzer assoziiert hat, speichert. In dem Modell `AssociationHistory` wird die gesamte Assoziationsgeschichte

2 Assoziationsdatenbank und Website

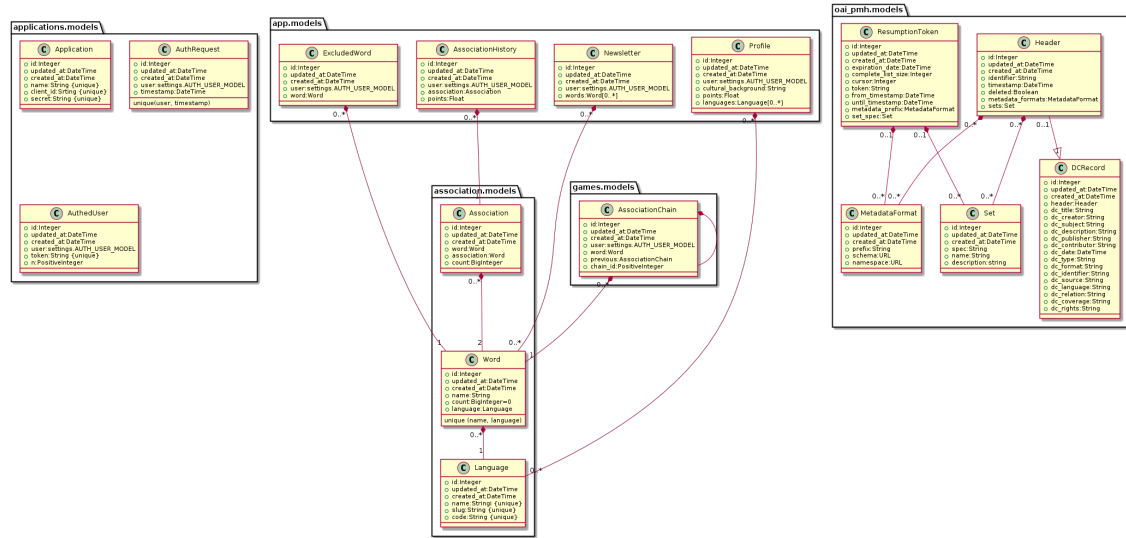


Abbildung 2.1: UML des TIMA Datenmodells

eines Benutzers gespeichert, mit den jeweils für eine Assoziation erhaltenen Punkte. Das Modell `ExcludeWord` enthält für jeden Benutzer die Wörter, die er übersprungen hat (vgl. Abschnitt 2.2.2), 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 (vgl. Abschnitt 2.2.4).

Das Modul `games.models` enthält Modelle die für die verschiedenen Spiele wichtig sind. Dies ist im Moment nur `AssoziationsKette` (vgl. Abschnitt 2.2.3), 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. Abschnitt 2.3), zu autorisieren und dem speichert der nötigen Informationen dient das Modul `applications.models`. Das Modell `Application` speichert die Apps, die Autorisiert sind, mit den nötigen Daten für die Autorisierung (vgl. Abschnitt 2.3.2). 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 (vgl. Abschnitt 2.3.3).

2.2 Webfrontend

Das Webfrontend ist die Hauptanlaufstelle für Benutzer. Hierüber kann er sowohl anonym als auch angemeldet Assoziationen eingeben, Wörter und deren Assoziationen ansehen und weitere Funktionen (Rangliste, Statistik) aufrufen.

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

Im folgenden werden nun einige Funktionalitäten noch einmal einzeln kurz erläutert und beschrieben.

2.2.1 Wortselektierungsalgorithmus

In diesem Abschnitt wird die Funktionsweise des Wortselektierungsalgorithmus genauer beschrieben. Der Wortselektierungsalgorithmus ist zuständig für das auswählen der Worte die assoziiert werden sollen. Folgende Anforderungen sollen dabei erfüllt werden:

- Jedes Wort soll möglichst mindestens eine Assoziation haben.
- Jeder Benutzer soll möglichst zu jedem Wort mindestens eine Assoziation geben.
- Wörter, die wenig assoziiert wurden, entweder insgesamt oder von einem einzelnen Benutzer, bevorzugen.
- Wörter sollen ausgeschlossen werden können, um z.B. zu verhindern, dass das selbe Wort mehrmals hintereinander kommt.

Der Algorithmus der diese Anforderungen erfüllt sieht folgendermaßen aus:

Listing 2.1: Wortselektierungsalgorithmus

```
1 w = [alle Wörter einer Sprache]
2 w = w - [alle auszuschließenden Wörter]
3 if ( Anonymer Benutzer )
4     w = [15 Wörter mit niedrigstem Häufigkeit aus w]
5 else
6     w = w - [alle auszuschließenden Wörter des Benutzers]
7     w = [15 Wörter mit niedrigstem Auftreten in der
           AssociationHistory des Benutzers1 aus w]
8 return [zufälliges Wort aus w]
```

2.2.2 ExcludeWords

Falls ein Benutzer zu einem Wort keine Assoziation einfällt, hat er die Möglichkeit das Wort zu überspringen. Damit er nicht ständig danach gefragt wird (vgl. Abschnitt 2.2.1), wird das Wort auf eine Ausschlussliste gesetzt. Einträge die älter als 7 Tage sind werden automatisch von der Liste gelöscht. Zu diesem Zeitpunkt, wird der Benutzer dann wieder gefragt dieses Wort zu assoziieren.

2.2.3 Spiele

Die Idee mit den Spielen kam hauptsächlich durch die Apps (vgl. Abschnitt 3.4). Allerdings haben wir uns dazu entschieden einzelne Spiele auch auf der Website anzubieten. Derzeit ist nur das Spiel AssoziationsKette verfügbar.

2.2.4 Newsletter

Ein weiteres Feature über das TIMA verfügt ist der Newsletter. Hier kann jeder Benutzer Wörter auswählen, die ihn interessieren und wöchentlich darüber einen Newsletter erhalten, in dem zu jedem Wort die Assoziationen mit deren Häufigkeit enthalten sind.

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 gliedern. Zum einen gibt es die Anfragen, die keiner Autorisierung bedürfen, zweitens jene 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, welche eine Autorisierung benötigen. Dort wird ebenfalls der Autorisierungsprozess näher erläutert. Zum Schluss wird dann noch ein Abschnitt zu OAI-PMH folgen.

2.3.1 Nicht autorisierte Anfragen

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

¹<https://github.com/Tima-Is-My-Association/TIMA/blob/master/API.md>

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.

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 Grund war es erforderlich, dass einige API Anfragen einer Autorisierung 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, über 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.

2 Assoziationsdatenbank und Website

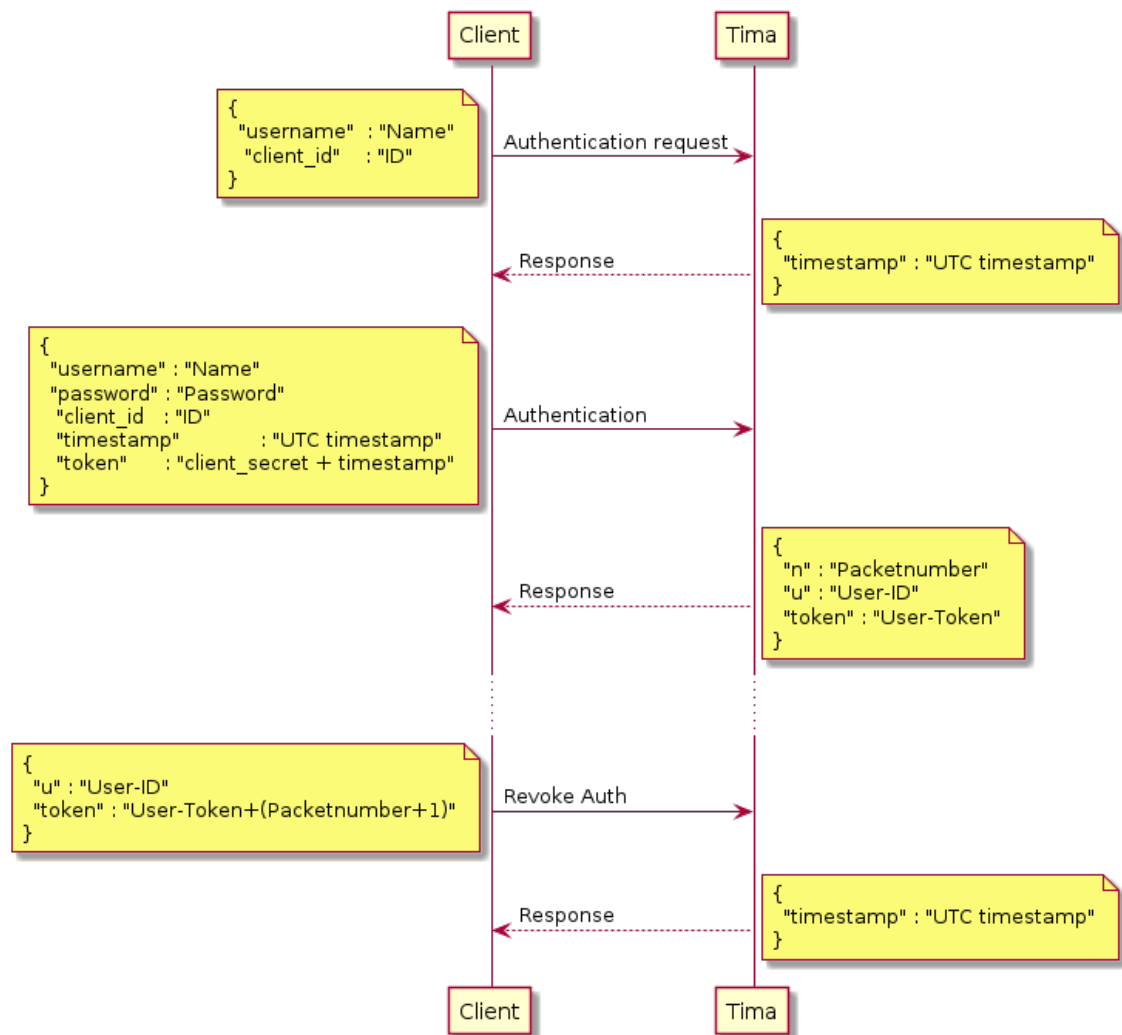


Abbildung 2.2: Authentisierungsprozess

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 Werten:

- n Paketnummer jeder Anfrage einer App muss diese um eins nach oben zählen. Als Wertebereich ist uint32 zu benutzen.
- u eine eindeutige Benutzer-ID, die bei jeder Anfrage mitzusenden 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. Es wird dabei unterschieden zwischen Data Providern und Service Providern. Ein Data Provider betreibt ein oder mehrere Repositories, die OAI-PMH unterstützen um Metadaten bereitzustellen. Service Provider sammeln die Metadaten der Data Providern und bieten Mehrwertdienste an.

Wir haben das OAI-PMH Protokoll implementiert um Metadaten zu den gesammelten Assoziationen bereitzustellen. Wir stellen dabei Metadaten zu den Wörtern und im geringen Maße zu den Benutzern bereit.

3 Applikationen

Die Applikationen sollen 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 der API, um etwaige Fehler im Protokoll oder Verbesserungen dessen aufzuzeigen und zu beheben.

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 Fehlerkorrektur zu ermöglichen. In Abbildung 3.1 wird der Zusammenhang der einzelnen Zustände angezeigt. Das wechseln der Zustände wird 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 Applikationen Schreibrechte für die Datenbank zu geben. Das Auslesen der Informationen bleibt jedoch davon unangetastet, sofern es sich nicht um Benutzer spezifische Daten handelt, und ist nach wie vor für jeden offen.

3 Applikationen

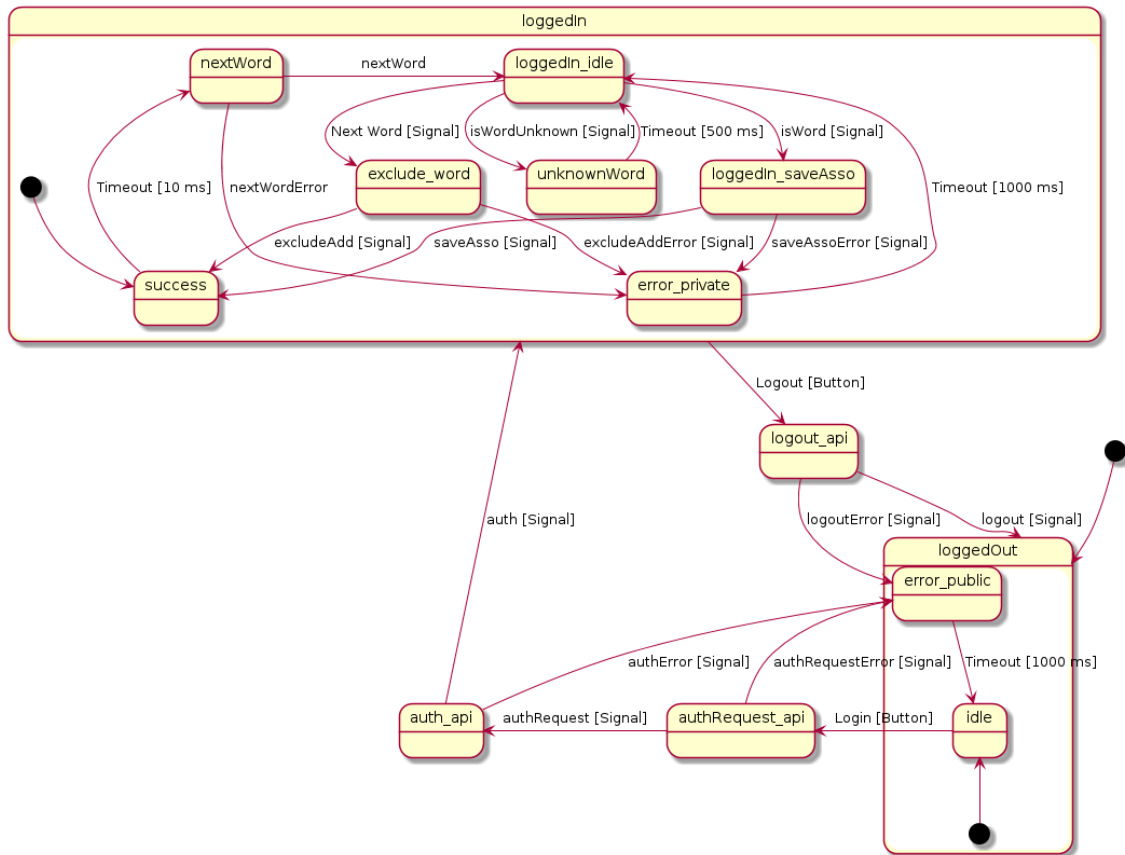


Abbildung 3.1: UML State Diagramm des Applikationszustands-automaten

Sicherheitsplanung Zuerst 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.

Als zweites wurde eine Implementierung vom OAuth2-Standard versucht. Die Anforderungen bestanden hierbei an einem geheimen Schlüssel in der Applikation, dem Ausschluss einer Replay-Attacke und dem Verwenden von User eigenen Passwörtern.

Im Standard sind 4 verschiedene "Authorization Grants" definiert² jedoch erfüllt keine davon jede Anforderung an das gewünschte Protokoll.

Zuletzt wurde sich für eine Eigenentwicklung entschieden um sowohl den Arbeitsaufwand klein zu halten, als auch alle Anforderungen ausreichend zu erfüllen. Eine kleine Übersicht ist in Abbildung 2.1 zu sehen.

²siehe <https://tools.ietf.org/html/rfc6749#section-1.3>

3 Applikationen

Die Übertragung erfolgt mit JSON Objekten um das Parsen zu erleichtern und einen üblichen Standard zu verwenden. Der Rechenaufwand des Servers wurde zudem versucht so gering wie möglich zu halten, um mögliche DDOS Angriffe abzuschwächen.

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 Nutzernamen 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 konkatenierten Applikations-Geheimnis besteht. Als Hashmethode wurde hierbei SHA-512 verwendet.

Nachdem der Server alle Angaben überprüft hat und die Signatur korrekt verifizieren konnte, erhält der Client 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 konkateniert 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 erzielen sind verschiedene Spiele geplant worden, die jedoch aus Zeitgründen nicht in den geplanten Zeitrahmen passten. Als Grundlage wird jedoch in den meisten Fällen schon eine gut gefüllte Datenbank vorausgesetzt. In den folgenden Abschnitten, werden kurz die Spiele Assoziationskette und Familienduell in ihrer Funktion beschrieben und der Nutzen für die Datenbank näher erläutert.

3 Applikationen

Assoziationskette Die Spieler assoziieren nacheinander auf die jeweilige Assoziation ihres Vorgängers. Dadurch bildet sich eine Kette mehrerer Assoziationen bis zu bestimmten Abbruchbedingungen. Ein mögliches Ende stellt das assoziieren eines schonmal verwendeten Wortes dar und die damit verbundene Schließung eines Kreises in der Kette. Das Spiel ist ebenfalls beendet, sollte einem Spieler keine weitere Assoziation zeitnah einfallen. Bei einem Computergegner wird so lange wie möglich versucht die Abschlusskriterien nicht zu erfüllen, damit der Spieler selbst in den meisten Fällen dafür verantwortlich ist.

Alle gegebenen Assoziationen des Spielers werden hierbei mit Punkten belohnt und für die Datenbank verwendet. Die Gesamtlänge der Kette wird nochmal extra als Bonus berechnet, hat jedoch keine Auswirkungen auf die Datensätze in der Datenbank.

Familienduell Der Name soll hierbei nur die Ausrichtung des Spieles näher bringen, denn wie bei der Fernsehserie³, werden dem Benutzer verschiedene verdeckte Antworten auf eine Frage gezeigt und für jede richtig gegebene Antwort erhält der Spieler Punkte. Die Fragen sind jedoch im Unterschied zum Fernsehen, ausschließlich die meist genannten Assoziationen zu einem bestimmten Wort. Zusätzlich sollte eine Zeitbegrenzung eingehalten werden, da ja alle Korrekten Antworten auf TI-MA nachgeschaut werden können und so einem möglichen Betrug entgegen gewirkt werden kann. Je nach Schwierigkeitsgrad kann ein Zeitbonus für korrekte Assoziationen gegeben werden und ein Malus bei falschen Antworten. Die Einwirkung auf das Spielvergnügen müsste entsprechend getestet werden.

Jede gegebene Antwort wird für das Füllen der Datenbank verwendet, obgleich es eine gesuchte Lösung war oder nicht. Punkte erhält der Spieler jedoch nur für richtige Lösungen und einen Bonus wenn er alle Lösungen findet.

³siehe <https://de.wikipedia.org/wiki/Familien-Duell>

4 Ausblick

5 Zusammenfassung