Dokumentation

Sprechstunden Internet Verwaltung

cdeinlein, tdubovik, spahl, ndiadowitz

Inhaltsverzeichnis

1 Vorwort

In dieser Studienarbeit geht es um die Erstellung einer Website zur Sprechstundenverwaltung an der Hochschule Hof. Als Projektname wurde `Schiv` gewählt das eine Abkürzung von (S)pre(ch)stunden (I)nternet (V)erwaltung ist. Für die Implementierung werden folgende Hilfsmittel verwendet:

- HTML 5
- CSS 3
- Angular 1.6.3
- Bootstrap 3.3.7
- jQuery 1.9.1
- PHP 7.0
- Laravel 5.4
- MySQL
- Sqlite3 (Lokale Entwicklung)

Dateinamen beziehen sich immer auf das Wurzel-Verzeichnis des Projekts und sind in `Proportionalschrift` geschrieben.

Es wird davon ausgegangen das die Anwendung lokal läuft. Alle angegeben Urls beziehen sich auf `localhost`.

2 Funktionale Anforderungen

Eine Liste von Funktionalen Anforderungen die während der Gespräche mit dem Auftraggeber beschlossen wurden + weiterer Funktionalitäten:

1. Konten sind nur registrierbar mit E-Mail-Adressen (`@hof-university.de`) der Hochschule Hof.
2. Zur Registrierung wird nur die E-Mail-Adresse und ein Passwort benötigt.
3. Konten werden nach der Registrierung als „Studenten“ behandelt.
4. Beim Vergessen eines Passworts soll der Nutzer in der Lage, sein durch seine Email, das Konto wiederherzustellen.
5. Eine Liste der aktiven Dozenten wird über die Schnittstelle der iOS-Stundenplan-App[1] abgefragt. Hierbei muss aus den Namen des Dozenten die E-Mail-Adresse abgeleitet werden, da die diese nicht direkt abfragbar ist. Für jeden Dozenten wird ein deaktiviertes Konto angelegt.
6. Konten können nur „Dozent“ werden, wenn bereits ein deaktiviertes Konto diesen Typs angelegt ist.
7. Jedes Konto ist beliebig vielen Fakultäten zugeordnet.
8. Aktivierung eines Kontos erfolgt über eine E-Mail, die, nach der Registrierung, mit einem Aktivierungslink geschickt wird.
9. Die Anwendung soll mindestens auf normalen Desktop-PCs mit den weitverbreitetsten Browsern laufen.
10. Es werden generell keine Datensätze gelöscht, sondern nur weich gelöscht („unsichtbar gemacht“).
11. Am Ende eines Semesters werden Studenten aus dem System gelöscht und müssen sich dann wieder erneut registrieren
12. Aktivierungslink wird über einen GET abgesetzt, d.h. der Aktivierungslink muss im Frontend implementiert sein.
13. Ein Dozent soll in der Lage sein Einzeltermine, sowie wöchentlich wiederkehrende Termine anzulegen.
14. Ein Student soll in der Lage sein Anfragen, zu einen beliebigen Termin zu stellen.
15. Ein Dozent soll in der Lage sein Anfragen von Studenten anzunehmen und abzulehnen.
16. Einem Dozent soll die Möglchkeit gegeben werden Anfragen von bestimmten Studenten zu blockieren bzw. diese Blockierung wieder aufzuheben.
17. Ein Student darf nicht in der Lage sein willkürlich Spam Anfragen zu stellen.
18. Ein Dozent soll in der Lage sein terminfreie Tage für sich zu definieren die auch von wiederkehrenden Terminen berücksichtigt werden.
19. Ein Nutzer(Student oder Dozent) soll in der Lage sein seine Email zu ändern, jedoch nicht

[1] https://github.com/HochschuleHofStundenplanapp/iOS-App/wiki/Schnittstellen-zum-Server

in eine die bereits von einem anderen Nutzer belegt ist.

3 Style-Guide

3.1 Farbschema

Das Farbschema orientiert sich an den drei Farben der Hochschule mit eigener Interpretation.

Tabelle 3.1: Grundlegende Farben

HEX	Name	Beschreibung
`ff7043`	orange	Buttons und Eingabe Felder
`03a9f4`	blau	Login Fenster und Termine, so wie Terminanfragen bei Studenten.
`ffb74d`	orange-gelb	Terminanfragen beim Dozenten.
`ffeb3b`	gelb	Liste der Dozenten
`989898`	hell grau	Abgelehnte Terminanfragen

Tabelle 3.2: Weitere Farben

HEX	Name	Beschreibung
`808080`	grau	box-shadow
`a94442`	rot	Fehlerhafte Validierung in Eingabe Feldern background-color: rgba(168, 68, 66, 0.05)
`3c763d`	grün	Erfolgreiche Validierung in Eingabe Feldern background-color: rgba(60, 118, 61, 0.05)
`ffffff`	weiß	Hintergrundfarbe eines Event Fensters
`000000`	transparentes schwarz	Hintergrund eines Event Fensters background-color: rgba(0, 0, 0, 0.33)

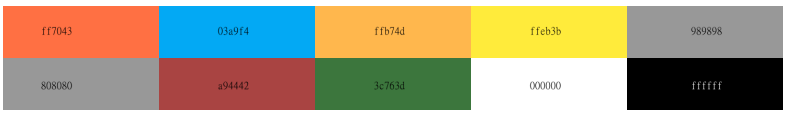

Abbildung 3.1: Oben: Grundlegende Farben. Unten: Weitere Farben

3.2 Elemente

Als Grundlage wird Bootstrap in Version 3 verwendet. Alle Elemente haben keine Text Effekte und abgerundete Ecken mit 2px Größe.

3.2.1 Eingabe Elemente

Der Aufbau eines Eingabefeldes ist sehr simpel gehalten. Am unteren Rand ist eine 2px hohe durchgehende Linie in orange.

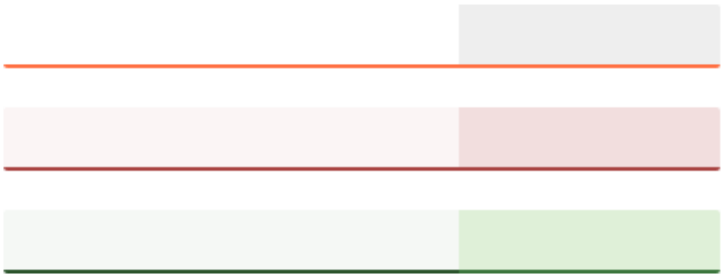

Abbildung 3.2: Eingabefeld mit addon in Standard und mit fehlerhafter und erfolgreicher Validierung

Ein addon eines Eingabefeld hat ebenfalls unten ein orangen Rahmen.

Für die Anzeige einer fehlerhaften oder erfolgreichen Validierung werden die Farben rot und grün verwendet.

3.2.2 Buttons

Knopfe sind einfarbig, flach in orange. Für activ und hover wird ein nach inset box-shadow Effekt verwendet.

3.2.3 Termine und Anfragen

Anfragen von Studenten auf Termine, werden in den Terminen angezeigt. An der rechten Seite sind jeweils Knöpfe zum Annehmen, ablehnen oder löschen des Termins oder der Anfrage. Der Dozent kann einen Studenten Sperren. Dafür ist der Knopf mit dem Daumen nach unten.

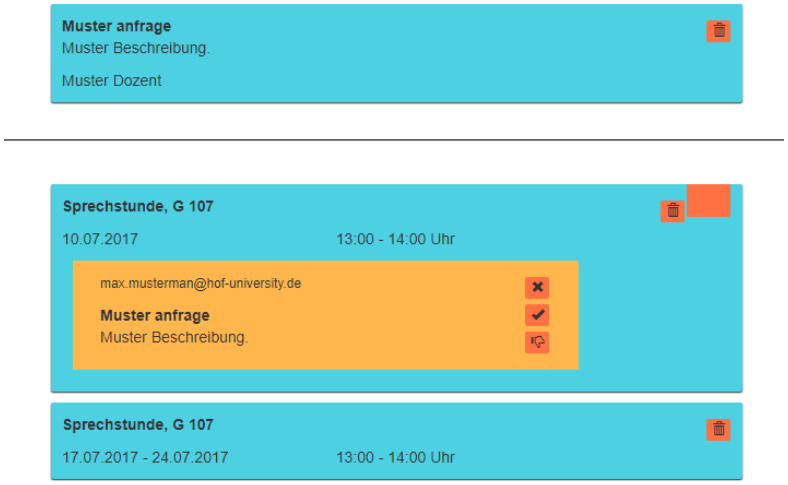

Abbildung 3.3: Oben: Terminanfrage in Sicht eines Studenten. Unten Termin mit und ohne Anfrage in sicht eines Dozenten

3.3 Typographie

(TODO)

Als Schriftart wird Roboto[1] verwendet, da diese klar lesbar ist.

3.4 Layout

Beim Layout muss zwischen Student und Dozent unterschieden werden.

3.4.1 Student

Als Student sieht man auf der Startseite neben den Terminanfragen auf der linken Seite, eine Liste von Dozenten auf der anderen Seite.

In den Einstellungen sind nur die Möglichkeiten der Änderung der Email Adresse und des Passwords. So wie die Angabe der Fakultät.

[1] https://fonts.google.com/specimen/Roboto?selection.family=Roboto

3.4.2 Dozent

Das Layout einer Dozenten Seite ähnelt der einer Studenten Seite. Auf der rechten Seite fehlt jedoch die Liste der Dozenten und die Ansicht der Termine ist auf ganzer breite zu sehen.

Auch die Einstellungen sind grundlegend gleich mit den Studenten. Hinzu kommt noch die Möglichkeit eine Sperrung eines Studenten wieder aufzuheben. Und das ein und austragen eines individuellen Feiertags.

3.5 Navigation

Die Navigationsleiste am oberen Rand ist immer sichtbar. Alle Funktionen sind darüber mit wenigen Klicks erreichbar.

4 Datenbankmodel

4.1 Überblick

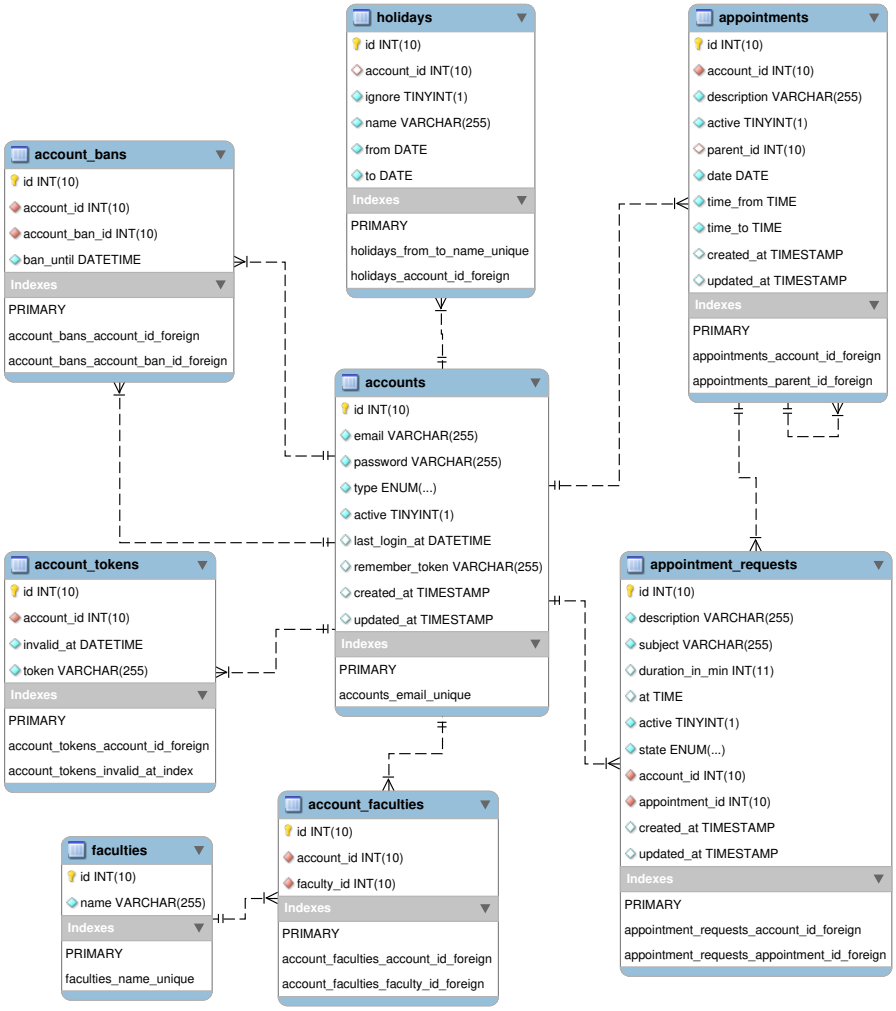

Abbildung 4.1: Datenbankmodel von schiv (Erstellt mit mysql-workbench)

4.2 Tabelle account_bans

Ein Dozent (account_id) kann Benutzer (account_ban_id) bis zu einem bestimmten Datum (ban_until) aussperren. Der betroffene Benutzer kann während dieser Zeit keine Anfragen stehlen.

4.3 Tabelle holidays

Diese Tabelle sollte in events umbenannt werden.

Enthält Zeiträume (from bis to) bzw. Zeitpunkte (from == to) von wichtigen Ereignissen (name). Diese Ereignisse können auch Ferien sein (ignore == false). Beispiele:

- Ostern 2017-04-13 bis 2017-04-08 (Feiertag)
- Vorlesungsbeginn 2017-10-02

Allgemeine Termine können nicht durch Benutzer geändert werden (account_id == null). Dozenten können ihre eigenen „Ferien“ eintragen. Wird bei der Erstellung von Terminserien beachtet.

4.4 Tabelle appointments

Enthält Termine (date, time_from und time_to) mit einer Beschreibung (description) von Dozenten (account_id). Die Termine können aktiv (active), d.h. es kann sich auf ihnen eingetragen werden, oder sie sind nicht aktiv bzw. gelöscht. Terminserien werden über die parent_id abgebildet.

4.5 Tabelle appointment_requests

Enthält Anfragen von Studenten (account_id) an Termine (appointment_id) von Dozenten. Die Anfrage kann sich einem von drei Zuständen (state) befinden: Accepted, Idle oder Declined. Für die Anfrage kann ein Betreff (subject) und eine detailliertere Beschreibung (description) angegebene werden. Zusätzlich wird noch der Startzeitpunkt (at) innerhalb des Termin und die Dauer (duration_in_min) gespeichert.

4.6 Tabelle accounts

Enthält einen Benutzer der über seine E-Mail-Adresse (email) identifiziert wird und vom Typ (type) entweder Dozent oder Student ist. Benutzer die nicht aktiv (active) sind, können sich

nicht anmelden und müssen sich erst neu registrieren. Ein Passwort (`password`) dient zur Authentifizierung des Benutzers. Um zu erkennen welche Benutzer noch mit dem System arbeiten, wird der Zeitpunkt der letzten Anmeldung (`last_login_at`) gespeichert.

4.7 Tabelle account_tokens

Zum Aktivieren von Konten nach der Registrierung eines neuen Benutzers (`account_id`). Der Token (`token`) wird per E-Mail verschickt, verpackt in einem Aktivierungslink, und ist nur ein bestimmte Zeit gültig (`invalid_at`). Vom Programm wird sichergestellt das jeder Benutzer nur einen Satz in der Token-Tabelle hat.

4.8 Tabelle account_faculties und faculties

Enthalten die Zuordnungen von den Fakultäten zu den Benutzerkonten.

5 REST-Schnittstelle

5.1 Überblick

Tabelle 5.1: Zeigt den Aufbau der REST-Schnittstelle

Beschreibung	[A|D] Method:Url (Body)
Login	put:login (email, password)
Logout	put:logout (email, password)
Reset Passwort	put:reset (email)
Registrieren	post:register (email, password)
	put:register (token)
Suche: Dozenten	get:docents
Informationen bzw. Termine von Dozenten	get:docents/{docent_id}
Terminanfragen abrufen	**A** get:appointment_requests
Vergangene Terminanfragen	**A** get:appointment_requests/past
Termin Einschreibung	**A** post:appointment_requests (…)
Terminanfrage annehmen bzw. ablehen	**D** put:appointment_requests (id, state, duration_in_min)
Terminanfrage zurücknehmen	**A** delete:appointment_requests/{id}
Termine Dozenten	**D** get:appointments
	D get:appointments/past
	D post:appointments (…)
	D delete:appointments/{appointment_id}
Feiertage	**A** get:holidays/{from}/{to}
	D post:holidays (from, to, name)
	D put:holidays (id, from, to, name)
	D delete:holidays/{id}
Einstellungen	**A** get:settings
	A put:settings (email, password)
Ausgeschlossene Konten	**D** get:account_bans
	D post:account_bans (account_ban_id)
	D delete:account_bans/{id}

Anmerkungen:

- `(...)` bedeutet das alle Felder, wie im Model definiert, verwendet werden können.
- **A**: Zugriff nur wenn man als Student oder Dozent angemeldet ist.
- **D**: Zugriff nur wenn man als Dozent angemeldet ist.

5.2 Details

5.2.1 Allgemein

Die Nachrichten sind im JSON-Format. Bei Erfolg kommt der HTTP-Status **200** zurück, im Fehlerfall kann **401**, **404**, **409**, **422**, **429**, **500** oder **503** zurückgegeben werden:

- **200**: Es kommen die angeforderten Daten zurück oder eine leere Nachricht falls die kein Rückgabewert nötig bzw. vorhanden ist.
- **401**: Falls der Login fehlschlägt, der Benutzer nicht angemeldet ist oder der Benutzer ein Dozent sein muss um auf die Route zugreifen zu können. Als Antwort kommt eine Fehlermeldung mit dem Grund zurück. Beispiel:

```
{"message":"login unsuccessful"}
{"message":"account isn't of type 'Docent'"}
{"message":"login required"}
```

- **404**: Die angefragten Objekte sind nicht in der Datenbank vorhanden. Als Antwort kommt eine leere Nachricht zurück. Beispiel:

```
[]
```

- **409**: Wird zurückgegeben wenn ein Konto bereits registriert ist.
- **422**: Die Anfrage enthält nicht alle nötigen Felder, oder ein Feld ist falsch formatiert. Die Antwort enthält als Attribute die Feldnamen bei denen ein Fehler festgestellt wurde. Als Wert der Attribute wird ein Array mit Fehlermeldungen zurückgegeben. Beispiel:

  Anfrage an `post:register`:

```
{"email": "alice","password": "bob"}
```

  Antwort von `post:register`:

```
{"email":["validation.required"],"password":["validation.min.string"]}
```

- **429**: Wird beim erstellen von Anfragen gesendet, wenn bereits eine Anfrage an einen Dozenten gestellt wurde. Oder wenn mehr als 60 Anfragen pro Minute von einer Session gestellt wurden.
- **500**: Ein allgemeiner Fehler (Syntaxfehler oder nicht behandelter Fehler). Als Antwort wird eine Fehlermeldung im Attribut `message` zurückgegeben. *Hinweis:* Ist die Anwendung im Debugmodus (`app.debug == true`), dann wird die Meldung aus der auslösenden

Exception zurückgegeben. Es kann im Log unter `storage/logs/laravel.log` die genaue Fehlermeldung nachgeschaut werden. Beispiel:

```
{"message": "fatal error"}
```

- **503**: Kommt nur beim erstellen von Terminen vor. Zeigt an dass das „Vorlesungsende“ nicht in der Ereignistabelle gefunden wurde.

5.2.2 Register, Login, Logout und Reset Password

Der Login funktioniert über eine Session, die serverseitig gespeichert wird. Auf Clientseite muss in einem Cookie die geschickte Session-Id mitgeführt werden. Da die Anwendung sowieso in einem Browser läuft, kümmert sich der Browser um die Cookie Verwaltung. Auf dem Server kümmert sich Laravel um die Session, die für jede Verbindung eindeutig ist und sein muss. Der Webserver muss später über das HTTPS-Protkoll angesprochen werden, sonst könnte ein Angreifer die Session „hijacken“. Wenn man die Implementierung von Laravel für die Session verwendet, wird bei jeder Anfrage die Session-ID geändert, aus Sicherheitsgründen. Als Konsequenz müssen die Anfragen per Javascript an die Schnittstelle synchron erfolgen. Ansonsten könnte es passieren das eine alte Session-ID aus dem Cookie gelesen wird, bevor die neue Session-ID abgespeichert wird.

Register: Beim Registrieren wird ein `post:register` geschickt mit E-Mail und Passwort des Benutzers. Wenn dies erfolgreich ist, dann wird eine E-Mail an den Benutzer mit einem Aktivierungslink geschickt (Momentan: wird einfach zum Testen der Token mit in der Antwort zurückgeben). Ein Aktivierungslink ist zwei Stunden lang gültig. Beispiel:

```
{"email":"alice@wonder.land","password":"rabbithole"}
```

Momentane Antwort:

```
{"token":"twon-ha"}
```

Login: Wenn der Login nicht erfolgreich ist wird ein **401** zurückgeben, sonst ein **200** mit leerer Nachricht. Beispiel: siehe **Register**.

Logout: Löscht die Session auf dem Server und meldet den Benutzer ab. Falls ein nicht angemeldeter Benutzer versucht sich abzumelden wird ein **401** zurückgeliefert.

Reset Password: Benötigt nur die E-Mail-Adresse und schickt eine E-Mail an den Benutzer mit einem neu generierten Passwort und einem Aktivierungslink. (Momentan: wird einfach zum Testen der Token und das neue Passwort mit in der Antwort `{"token":"<token>","password":"<password>"}` zurückgeben). Nach einem Klick auf den Aktivierungslink ist das Konto des Benutzer wieder aktiviert. Anmelden kann sich der Benutzer dann mit dem generierten Passwort. Das Passwort ist fest und dem Benutzer ist es freigestellt es wieder zu ändern. Beispiel:

```
{"email":"alice@wonder.land"}
```

Momentane Antwort:

```
{"token":"twon-ha","password":"rabbithole2"}
```

Laut Vorgabe (siehe Kapitel 2) muss der Aktivierungslink im Frontend implementiert sein. Da der Link über ein GET abgesetzt wird, aber das Backend ein PUT benötigt. Das Frontend muss dann ein `put:register` mit dem Token in der Nachricht absetzen, um das Konto wieder zu aktivieren.

5.2.3 Suche: Dozenten

Liefert alle aktiven Dozenten zurück, um im Frontend eine Suche implementieren zu können.

Bei `get:docents`: Liefert alle Dozenten zurück (`id`, `email`). Zusätzlich noch die Fakultät des Dozenten. Beispiel:

```
[
  {
    "id": 3,
    "email": "helmut.kohl@hof-university.de",
    "faculties": [{"id": 2, "name": "Ingenieur"}]
  },
  { "id": 4, "email": "apfel.mus@hof-university.de", "faculties": [] }
]
```

Bei `get:docents/{id}`: Wird zu allen obigen Informationen noch die Termine des Dozenten zurückgegeben. Beispiel:

```
{
  "id": 3,
  "email": "helmut.kohl@hof-university.de",
  "appointments": [
    {
      "id": 3,
      "account_id": "3",
      "description": "Klausureinsicht",
      "active": "1",
      "weekday": null,
      "date": "2017-05-17",
      "time_from": "11:00:00",
      "time_to": "12:00:00",
    }
  ],
```

```
  "faculties": [
    { "id": 2, "name": "Ingenieur" }
  ]
}
```

5.2.4 Termine

Terminanfragen:

Bei `get:appointment_requests`: Liefert bei einem Studenten die eigenen Anfragen zurück. Bei einem Dozenten die Anfragen zu den eigenen Terminen.

Bei `get:appointment_requests/past`: Liefert bei einem Studenten alle Anfragen des Studenten zurück, die sich auf vergangene Termine beziehen. Bei einem Dozenten alle Anfragen zu den eigenen abgelaufenen Terminen.

Bei `post:appointment_requests`: Legt eine neue Terminanfrage in der Datenbank an. Zu übergebene Parameter im *JSON-Objekt*: `appointment_id` (Id des Termins an den angefragt wird), `description` (Beschreibung was besprochen werden soll), `subject` (Betreff der Anfrage).

Bei `put:appointment_requests`: Damit wird der Status der Anfrage geändert sowie die Dauer der Besprechung. Zu übergebende Parameter im *JSON-Objekt*: `id` (Id der zu bearbeitenden Anfrage), `state` (Neuer Status der Anfrage: `Accepted`, `Declined`), `duration_in_min` (Voraussichtliche Dauer der Besprechung)

Bei `delete:appointment_requests/{id}`: Damit wird die in `id` angegebene Anfrage gelöscht.

Termine:

Bei `get:appointments`: Liefert alle Termine des Aufrufers zurück.

Bei `get:appointments/past`: Liefert alle inaktiven Termine des Aufrufers zurück.

Bei `post:appointments`: Legt einen neuen Termin in der Datenbank an. Bei wiederholenden Terminen werden alle anfallenden Wiederholungen bis zum Ende des Semesters mit angelegt. Zu übergebende Parameter im *JSON-Objekt*: `weekday` (Identifiziert einen wiederholenden Termin. Gültige eingaben sind: `MON TUE WED THU FRI SAT SUN NULL`), `date` (Identifiziert einen Einzeltermin. Falls `weekday` nicht den Wert `NULL` hat wird `date` nicht beachtet. Angabe wie folgt: `YYYY-MM-DD`), `time_from` (Beginn des Termins: `HH:MM:SS`), `time_to` (Ende des Termins:`HH:MM:SS`), `description` (Beschreibung des Termins).

Bei `delete:appointments/{id}`: Setzt den Status des mit `id` angegeben Termins auf `Inactive`.

Versucht ein gesperrter Student eine Anfrage zu stellen, wird **200** zurückgegeben, aber nur so getan als ob die Anfrage durchgegangen ist.

5.2.5 Feiertage

Allgemeine Feiertage (sprich: Feiertage die nicht mit einem Konto verknüpft sind) können nicht direkt bearbeitet werden, sondern nur abgefragt werden.

Sollte versucht werden einen allgemeinen Feiertag zu bearbeiten kommt ein **404** als Fehler zurück.

Das Format von `{from}` und `{to}` ist im Format `YYYY-MM-DD`.

Bei `post:holidays`: Liefert die Id des eingefügten Satzes zurück. Falls bereits ein Feiertag mit `from`, `to` und `name` eingetragen wurde, wird die Id dieses Satzes zurückgegeben.

5.2.6 Einstellungen

Liefert die Einstellungen des angemeldeten Benutzers zurück.

Bei `get:settings`: Liefert die Einstellungen des Kontos zurück, also eine Untermenge der Attribute von der `accounts`-Tabelle. Zusätzlich enthält die Antwort noch die Fakultäten des Kontos und alle möglichen Fakultäten. Beispiel:

```
{
  "email": "max.musterman@hof-university.de",
  "account_faculties": [
    { "id": 3, "name": "Wirtschaft" }
  ],
  "faculties": [
    { "id": 1, "name": "Informatik" },
    { "id": 2, "name": "Ingenieur" },
    { "id": 3, "name": "Wirtschaft" }
  ]
}
```

Bei `put:settings`: Kann E-Mail, Passwort und Fakultäten übergeben werden. Die Fakultäten werden als Array von Ids übergeben: `[1, 2, 3]`.

5.2.7 Konten Sperren

Liefert Kontosperrungen des angemeldeten Dozenten zurück. Alle gesperrten Benutzer können keine Anfragen mehr stellen. Bis sie aus der Tabelle gelöscht werden oder die Zeit für die Sperrung abgelaufen ist (Die Sperrung ist standardmäßig bis zum Beginn des nächsten Semester).

Wenn eine Sperrung eingetragen wird, wird der Datensatz mit aktuelle Id zurückgeben.

6 Implementierung

6.1 Allgemein

Die REST-Schnittstelle kann manuell über `test.html` ausprobiert bzw. getestet werden (Kurze Beschreibung ist unter Abschnitt 8.4 zu finden). Implementierungsdetails für die REST-Schnittstelle werden bereits in Abschnitt 5.2 beschrieben.

6.2 Verschicken von E-Mails

Die E-Mails werden nur in die Logdatei `storage/logs/laravel.log` geschrieben und nicht verschickt. Falls E-Mails verschickt werden sollen, kann dies in der Konfiguration `config/email.php` geändert werden. Die Vorlagen für die E-Mail Nachrichten befinden sich unter: `resources/views/emails`. Diese haben nur den nötigsten Inhalt, welcher für den Rahmen des Projekts ausreicht.

6.3 Login

Bereits in Abschnitt 5.2.2 ausführlich beschrieben.

(TODO Weitere Details der Implementierung hinzufügen)

7 Testfälle

7.1 Backend

Es sind für jeden Controller Testfälle im Verzeichnis `tests/Feature` angelegt.

7.2 Fontend

(TODO)

8 Benutzerhandbuch

8.1 Installation für Entwicklung unter Ubuntu 17.04

Es wird von einer neuen Installation ausgegangen.

```
$ sudo apt update
$ sudo apt install git composer unzip php php-mbstring php-xml
$ sudo apt install php-sqlite3 sqlitebrowser
$ cd $HOME
$ git clone https://github.com/studentcstn/schiv
$ cd schiv
$ composer install
$ composer dumpautoload
$ cp .env.sqlite .env
$ touch /tmp/db
$ php artisan migrate:refresh --seed
$ php artisan retrieve:docents
$ php artisan retrieve:holidays
$ php artisan serve
$ xdg-open http://localhost:8000
```

8.2 Dokumentation erstellen

Es wird davon ausgegangen das der vorherige Schritt in Abschnitt 8.1 ausgeführt wurde und das Arbeitsverzeichnis unverändert ist.

```
$ sudo apt install make latexmk
$ sudo apt install texlive-latex-recommended texlive-fonts-recommended
$ sudo apt install texlive-latex-extra texlive-fonts-extra
$ sudo apt install texlive-luatex texlive-lang-german
```

Ubuntu hat kein Paket `pandoc-crossref`, deshalb muss dieses erst mit `cabal` heruntergeladen und kompiliert werden. Bitte auch den nächsten Absatz beachten!

```
$ sudo apt install cabal-install
$ cabal update
$ cabal install pandoc pandoc-crossref
$ export PATH=~/.cabal/bin/pandoc:$PATH
```

Der vorherige Schritt kann weggelassen werden und das Standardpaket installiert werden, allerdings funktionieren dann keine Referenzen (d.h. die Dokuemtentation lässt sich komplett erstellen, aber Verweise auf andere Kapitel oder Unterkapitel funktionieren nicht). Als Folge daraus muss noch die Zeile `--filter pandoc-crossref` aus dem `doc/Makefile` entfernt werden.

```
$ apt install pandoc
```

Nach dem pandoc installiert ist, kann die Dokumentation wie folgt erstellt werden:

```
$ cd doc
$ make
$ xdg-open pdf/paper.pdf
```

8.3 Befehle

8.3.1 Allgemein

Die folgenden Befehle können manuell aufgerufen werden, aber auch automatisch. Um sie automatisch auszuführen kann das Task-Scheduling aktiviert werden mit dieser Anleitung https://laravel.com/docs/5.4/scheduling. Dann werden die Befehle am Anfang jedes Semesters aufgerufen.

8.3.2 Dozenten aktualisieren

Am Ende des Semester können die Dozenten-Konten aktualisiert werden. Dazu dient der Befehl `php artisan retrieve:docents`. Alle Dozenten die über die Schnittstelle nicht mehr vorhanden sind, werden deaktiviert. Deaktivierte Dozenten die noch Zugriff auf ihr E-Mail-Konto haben können sich erneut registrieren und anmelden. Die Zugangsdaten für die Schnittstelle der iOS-Stundenplan-App müssen in der `.env`-Datei eingetragen werden (Schlüssel: `IOSAPP_USERNAME` und `IOSAPP_PASSWORD`).

Zum Testen kann die Option `--from-cache` verwendet werden. Hier werden die Daten nur einmal vom Server geholt und dann in einer lokalen Datei `/tmp/docents` zwischengespeichert.

8.3.3 Feiertage aktualisieren

Die Feiertage können über den Befehl `php artisan retrieve:holidays` aktualisiert werden. Die Daten werden von der Hochschule Homepage unter http://www.hof-university.de/studierende/studienbuero/termine.html geholt.

8.3.4 Wartung

Zur Wartung kann der Befehl `php artisan maintance` benutzt werden. Es ist sinnvoll den Befehl am Ende des Semester auszuführen. Der Befehl führt folgendes aus:

- Alle Kontosperrungen werden gelöscht
- Alle Studentenkonten werden auf inaktiv gesetzt
- Alle Konten bei denen ein Login das letzte Mal vor 10 Jahren stattgefunden hat, werden gelöscht
- Alle Termine und Terminanfragen die älter als 10 Jahren sind werden gelöscht

8.4 Manuelles Testen der REST-Schnittstelle

Über http://localhost/test.html kann die REST-Schnittstelle manuelle getestet werden.

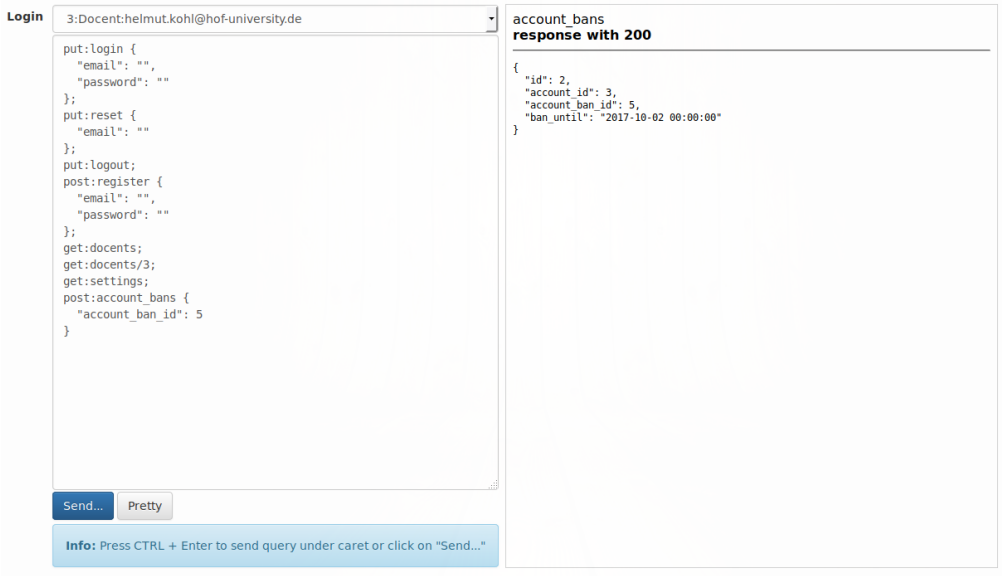

Abbildung 8.1: Seite test.html im Browser

Über die Auswahl `Login` kann man sich als jeder Benutzer anmelden, solange das Passwort `clearTextPassword` ist. Im darunter liegenden Feld kann man seine Anfragen formulieren. Ein Anfrage halt folgende Syntax:

`(delete|post|put|get):<url> <json>;`

Es können mehrere Anfragen in diesem Feld stehen, diese müssen aber mit Semikolon (;) getrennt werden. Um eine bestimmte Anfrage auszuführen klickt man mit der Maus oder positioniert das Caret auf irgendein Zeichen vor dem Semikolon. Ein Anfrage kann entweder mit klicken auf `Send...` oder mit der Tastenkombination `CTRL + Enter` ausgeführt werden.

Im rechten Bereich sieht man das Ergebnis der Anfrage.

8.5 Front-End

(TODO)