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 dürfen folgende Hilfsmittel verwendet werden:

- HTML 5
- CSS 3
- Javascript (Angular JS, Bootstrap, Ajax, jQuery)
- PHP (Laravel)
- MySQL

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

2 Funktionale Anforderungen

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

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. 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.
5. Konten können nur „Dozent“ werden, wenn bereits ein deaktiviertes Konto diesen Typs angelegt ist.
6. Jedem Konto sind ein oder mehrere Fakultäten zugeordnet.
7. Aktivierung eines Kontos erfolgt über eine E-Mail die nach der Registrierung mit einem Aktivierungslink geschickt wird.
8. Die Anwendung soll mindestens auf normalen Desktop-PCs mit den weitverbreitetsten Browsern laufen.
9. Es werden generell keine Datensätze gelöscht, sondern nur weich gelöscht („unsichtbar gemacht“). Bis auf Token, Feiertage und Sperrungen.
10. Am Ende eines Semesters werden Studenten aus dem System gelöscht und müssen sich dann wieder erneut registrieren
11. Aktivierungslink wird über einen GET abgesetzt, d.h. der Aktivierungslink muss im Frontend implementiert sein.

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

3 Style-Guide

3.1 Farbschema

(TODO)

Es wird sich am Farbschema der Hochschule Hof orientiert.

3.2 Typographie

(TODO)

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

3.3 Layout

(TODO)

Beim Layout muss zwischen Student und Dozent unterschieden werden. Das Layout der Seite soll möglichst einfach und übersichtlich sein.

3.4 Navigation

(TODO)

Es wird nur eine Navigationleiste am oberen Rand geben, die immer sichtbar ist. Alle Funktionen sind darüber mit wenigen Klicks erreichbar.

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

4 Datenbankmodel

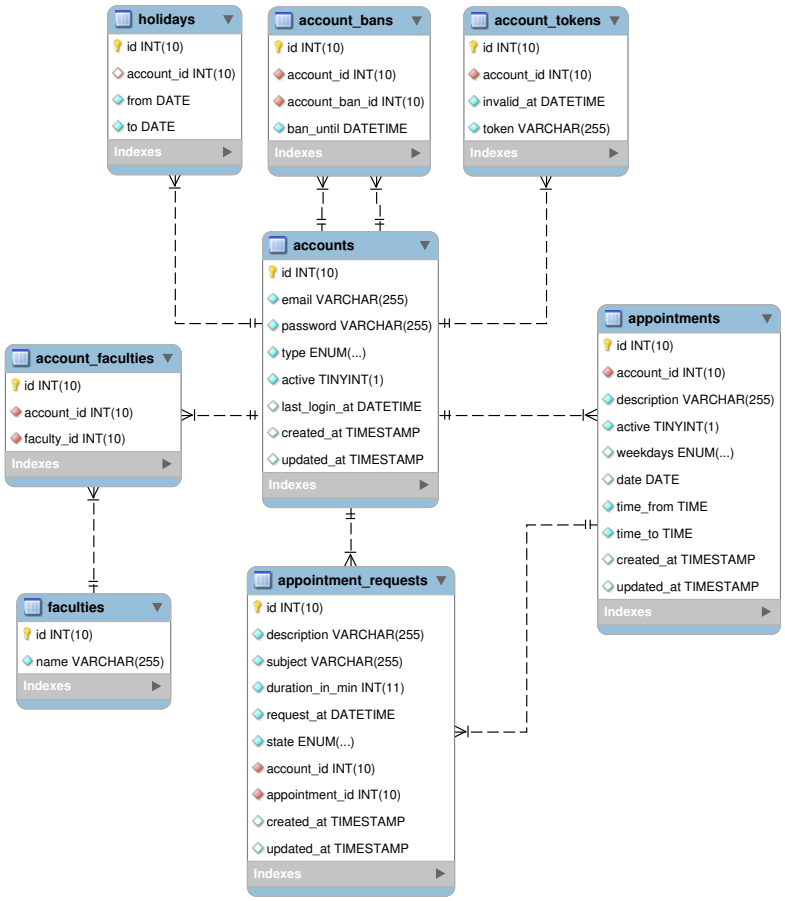

Abbildung 4.1: Datenbankmodel von schiv (Erstellt mit mysql-workbench)

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**, **422** oder **500** zurückgegeben werden:

- **200**: Es kommen die angeforderten Daten zurück oder eine leere Nachricht falls die kein Rückgabewert nötig 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:

```
[]
```

- **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"]}
```

- **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"}
```

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 mit über das HTTPS-Protkoll angesprochen werden, sonst könnte ein Angreifer die Session „hijacken“. Wenn man mit 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.

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 Ende des Semesters 15.03 oder 30.09).

6 Implementierung

6.1 Allgemein

Die REST-Schnittstelle kann manuell über `test.html` ausprobiert bzw. getestet werden. 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 den nötigsten Inhalt, welcher für den Rahmen des Projekts ausreicht.

(TODO)

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 serve
$ php artisan retrieve:docents
$ 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 diese erst mit `cabal` heruntergeladen und kompiliert werden. 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. 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 der Login das letzte mal vor 10 Jahren ist, werden gelöscht
- Alle Termine und Terminanfragen die älter als 10 Jahren sind werden gelöscht.