Konzeption und Implementierung einer REST-basierten Schnittstelle zum Kopieren von Jira-Projekten

Praxisbericht 2 - T2000

des Studiengangs Informatik

an der Dualen Hochschule Baden-Württemberg Stuttgart

von

Luca Stanger

Februar 2020

Bearbeitungszeitraum	12 Wochen
Matrikelnummer, Kurs	7474265, TINF-18B
Ausbildungsfirma	camos Software und Beratung GmbH, Stuttgart
Betreuer	M.Sc. Markus Riegger

Inhaltsverzeichnis

Abkürzungsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1 Einleitung

1.1 Projektteam

1.2 Aufgabenverteilung

1.3 Programmablauf

2 Entwicklung

Für die Entwicklung der Software wurde von Florian die IDE Visual Studio Code verwendet und von Luca die IDE Jetbrains PhpStorm. Zum Verwalten der Datenbank wurde die Software Jetbrains DataGrip.

2.1 Toolchain

2.2 Bibliotheken/Frameworks

2.2.1 Bower

2.2.2 Composer

2.2.3 phpunit

2.3 MVC

2.3.1 .htaccess

2.3.2 Model

2.3.3 View

2.3.4 Controller

2.4 Funktionalität

2.4.1 Login/Register

passwort hashing confirm ohne email wegen email server in aws. logik der registrierung und anmeldung

2.4.2 Lobby

Scrum Master

Spieler

3 Versionsverwaltung

Zur Versionsverwaltung wurde ein Github Repository eingerichtet damit mehrere Entwickler an dem Projekt arbeiten können.
Das Projekt wurde von jedem Entwickler geklont um lokal entwickeln zu können.
In dem Github Repository ist das Projekt, die Dokumentation sowie verschiedene Konfigurationsdateien zufinden.

4 Hosting

Die Planning Poker Anwendung wird auf einem AWS Elastik Beanstalk 64bit Amazon Linux/2.9.4 Server mit der PHP Version 7.3 gehostet.

4.1 Anbieter

Zuerst wollten wir die Datenbank und den PHP Server auf Microsoft Azure hosten. Als Student erhält man, über das Github Student Developer Pack, ein jährliches Budget von 100$ ohne Angabe einer Kreditkarte bzw. Bezahlmethode. Wir wollten Azure verwenden, da wir bereits im dritten Semester eine PHP Anwendung auf Azure gehostet haben und eine einfache Einbindung mit Github funktioniert. Jedoch mit diesem Abonnement könnten keine Server und Datenbanken in Europa erstellt werden.
Danach haben wir Amazon Web Services verwendet. Ebenso erhält man, über das Github Student Developer Pack, ein Budget von 100$ (damals 150$).

5 Datenbank

5.1 Hosting

Zum Hosten der Datenbank wird die Amazon RDS for MariaDB mit der MariaDB Version 10.2.21 verwendet.
„Amazon Relational Database Service (Amazon RDS) ist ein Webservice, der das Einrichten, Betreiben und Skalieren einer relationalen Datenbank in der AWS Cloud vereinfacht.“[**AmazonRDS**] Amazon RDS unterstützt bis zu sechs verschiedene Datenbank-Engines, z.B. MySQL, PostgreSQL oder auch MariaDB. [**AmazonRDS**]

Der Vorteil einer Cloud-basierten Datenbank ist, dass jeder Entwickler problemlos darauf zugreifen kann.
Um auf diese Datenbank zugreifen zu können, unterstützt AWS sogenannte Security Groups. Diese Gruppen dienen als virtuelle Firewall. Man bindet sie einer Ressource wie z.B. der Datenbank oder dem Elastic Beanstalk Server zu. In einer Security Group wird spezifiziert welche IPs und sonstige Ressourcen Zugriff haben.
Da während der Entwicklungszeit die IP Adresse von Luca sich ständig geändert hat, haben wir von überall Zugriff erlaubt. Dies stellt natürlich ein erhöhtes Sicherheitsrisiko dar, dennoch braucht man Anmeldedaten für die Datenbank. Nach der Entwicklungsphase wurde dieser Zugang entfernt.

5.2 Verwendung von $_SERVER

Damit die Anmeldedaten für die Datenbank nicht blank im Quellcode zu finden sind, wurden diese in dem $_SERVER Array des Webservers gespeichert. Dieses Verfahren wurde gewählt, da ein öffentliches Github Repository erstellt wurde und somit sonst für jeden sichtbar wäre.

Um die Daten auf dem lokalen Xampp Server verwenden zu können, werden diese in der `httpd-xampp.conf` gespeichert. Im folgenden Codeblock werden die Anmeldedaten im Modul `env\_module` definiert.

```
1 #
2 # XAMPP settings
3 #
```

```
4
5  <IfModule env_module>
6      ...
7      SetEnv RDS_DB_NAME "planning_poker"
8      SetEnv RDS_HOSTNAME "planning-poker.cztpemauxamy.eu-central-1.rds.
           amazonaws.com"
9      SetEnv RDS_PASSWORD "<password>"
10     SetEnv RDS_PORT "3306"
11     SetEnv RDS_USERNAME "admin"
12 </IfModule>
```

Nun kann man innerhalb des Programmcodes auf das $_SERVER Array zugreifen um die zuvor definierten Werte aufzurufen. Dieser Codeblock, aus der Datei Model/ModelBase.php, zeigt die Funktion zur Erstellung der Datenbankverbindung.

```
1  /**
2  * GetPDO: returns a PDO Connection
3  * @author Luca Stanger
4  * @return Database
5  */
6  public function getPdo()
7  {
8    if (self::$_db === null) {
9      self::$_db = new \PlanningPoker\Model\Database(
10       $_SERVER['RDS_HOSTNAME'],
11       $_SERVER['RDS_PORT'],
12       $_SERVER['RDS_DB_NAME'],
13       $_SERVER['RDS_USERNAME'],
14       $_SERVER['RDS_PASSWORD'],
15       'utf8'
16     );
17   }
18
19   return self::$_db;
20 }
```

Damit der AWS Elastic Beanstalk Server ebenfalls auf die Daten über $_SERVER zugreifen kann, werden diese unter den Konfigurationseinstellung in AWS als Umgebungseigenschaften definiert.

5.3 Schema

In der Abbildung 5.1 ist das Schema der Datenbank zu sehen.

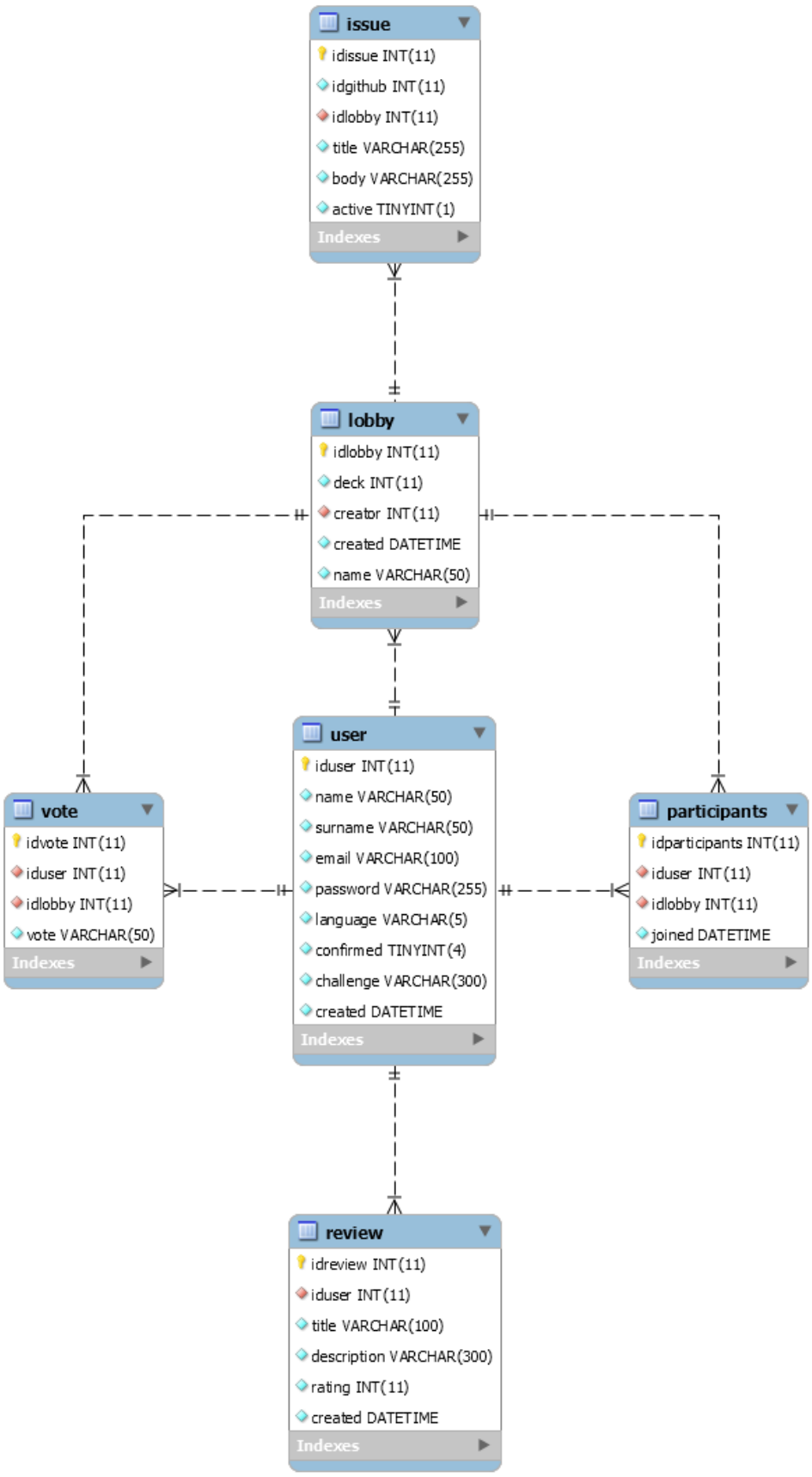


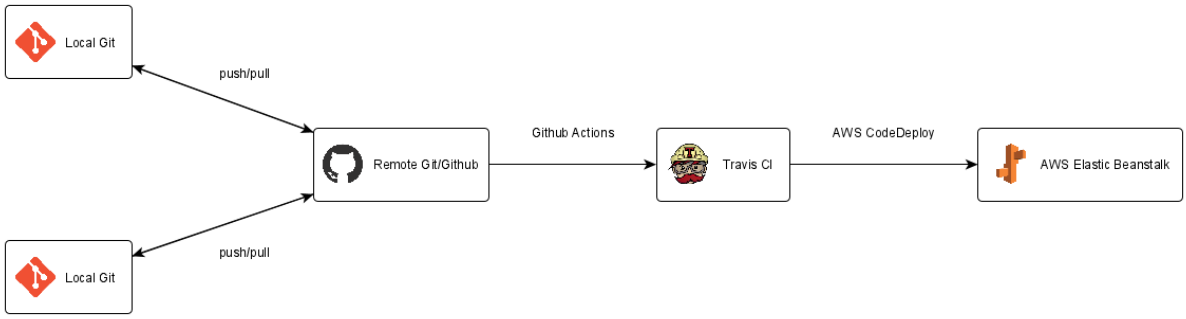
Abbildung 5.1: Visualisierung der Datenbank

Zu sehen sind die einzelnen Tabellen der Datenbank mit Verknüpfungen zu anderen Tabellen. In einer Tabelle sind die einzelnen Spalten beschrieben. Primärschlüssel haben einen goldenen Schlüssel, Fremdschlüssel eine rote Raute und normale Spalten eine blaue Raute. Außerdem werden die Typen der Spalten gezeigt. Alle Verbindungen beschreiben eine 1:N Beziehung und die Cascade Regel.
Dadurch, dass alle Fremdschlüssel mit der Option ON DELETE CASCADE beschrieben werden, Ein Beispiel wäre: Ein User kann mehrere Lobbies als Ersteller besitzten, ein Vote in einer Lobby besitzten, aber mehrere Votes in mehreren Lobbies, er kann durch die Tabelle participants an mehrere Lobbies teilnehmen, aber jeweils nur einmal in einer Lobby teilnehmen und er kann mehrere Reviews schreiben.

Der Vorteil durch dieses Schema ist, wenn z.B. ein Benutzer gelöscht wird werden automatisch alle Daten dieses Benutzers aus den Verbunden Tabellen gelöscht. D.h. die Reviews, Lobbies die der Benutzer erstellt hat, die Teilnahme aus anderen Lobbies, sowie alle Votes aus den Lobbies werden gelöscht. Außerdem, weil die Lobbies des Benutzers gelöscht werden, werden auch die Issues dieser Lobby gelöscht.

6 Continuous Integration

Als CI wurde am Anfang AWS CodePipeline verwendet. Dies bot eine einfach Einbindung zwischen Github und AWS Elastikbeanstalk. Sobald Der Master des Github Repositories einen neuen Push erhalten hat, wurde automatisch der neue Code auf den Server geladen. Jedoch als später Tests zum Projekt hinzugefügt wurden, wollten wir, dass automatisch die Tests vor dem Deploy ausgeführt werden. Dazu wäre AWS CodeBuild eine Möglichkeit gewesen. Doch persönlich wurde bereits mit Travis-CI gearbeitet, was eine kostenlose und einfach implementierte Version darstellt. Für Travis CI muss man nur dem Programm Berechtigungen für das Github Repository geben, sowie eine .travis.yml Datei im Root des Repositories vorhanden sein. In dieser yml-Datei spezifiziert man Befehle die in einer bestimmten Reihenfolge ablaufen. Außerdem wurde AWS CodePipeline nicht mehr benötigt, da Travis CI eine eigene kostenlose Deploy Möglichkeit bereitstellt. Dadurch wird nach jedem Push auf den Master des Github Repositories bei Travis CI eine virtuelle Umgebung aufgebaut die die definierten Befehle ausführt. Schlägt ein Build fehl wird es gestoppt. Den Status des Builds erkennt man an der build Badge in der ReadMe Datei. Sehen Sie hier den letzten Build des Projektes: https://travis-ci.com/github/Drinkler/Planning-Poker.



travis ci - aws codedeploy -> encrypted keys github app / actions

7 Fazit

7.1 Gewonnene Erkenntnisse

7.2 Ausblick

Anhang

A.