Umsetzung einer responsiven Webanwendung für Aufwandsschätzungen Agiler Projekte

Praxisprojekt

des Studiengangs Informatik

an der Dualen Hochschule Baden-Württemberg Stuttgart

von

Florian Drinkler, Luca Stanger

01. Juni 2020

Bearbeitungszeitraum	8 Wochen
Matrikelnummer, Kurs	7474265, 1234567, TINF-18B
Ausbildungsfirma	Balluff GmbH, camos GmbH, Stuttgart
Gutachter	Dr. Simon A. Frank

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

Zur Entwicklung der Anwendung wurde die folgende Toolchain verwendet:

Softwarename	**Zweck der Software**
PHPStorm	IDE zur Entwicklung fortgeschrittener Webanwendungen basierend auf PHP
Visual Studio Code	Modulare IDE für Entwickler
DataGrip	Datenbank-IDE, die auf die speziellen Bedürfnisse professioneller SQL-Entwickler zugeschnitten ist
Travis	Webbasiertes CI/CD Tool
GitHub	Versionsverwaltung
Bower	Package Manager für Webanwendungen
Composer	Dependency Manager für PHP
PHPUnit	Unit-Test Framework für PHP

Tabelle 2.1: Auflistung der verwendeten Toolchain

2.2 Bibliotheken/Frameworks

Als Frontend-CSS-Framework wurde Bootstrap in der Version 4.4.1 verwendet. Die Entscheidung hierfür ist durch bestehende Erfahrung, aus Projekten der vorherigen Studienprojekte, gefallen.

2.2.1 Bower

Um einen Fehlerfreien Build der CI/CD Pipeline zu gewährleisten, wurde der freie Paket Manager *Bower* verwendet. Hierdurch kann der Pipeline gewährleistet werden, dass benötigte Komponenten wie Bootstrap, jQuery sowie dem Font-Awesome Package jederzeit in dem finalen Build vorhanden sind.

2.2.2 Composer

Um die Abhängigkeit des Unit-Test Frameworks PHPUnit zu gewährleisten, wird das Framework via Composer nachgeladen. Darüber hinaus wird sichergestellt, dass mindestens eine PHP Version größer gleich 7.3.0 verwendet wird.

2.2.3 PHPUnit

Aufgrund der Anforderung der Testbarkeit des Projekts wird das Unit-Test Framework PHPUnit verwendet. Zur Gewährleistung der Testfunktionalität wurden zwei Unit-Test Klassen erstellt. Diese werden bei jedem Build ausgeführt und testen einen minimalen Teil der Anwendung.

2.3 MVC

Als Grundgerüst der Anwendung wurde sich für das MVC Pattern entschieden. Hierbei ist die Entscheidung auf eine komplexere Variante als dem in der Vorlesung vorgestellten MVC gefallen. Grund hierfür waren das persönliche Interesse an der Umsetzung einer etwas komplexeren Anwendung. Hierdurch konnte eine außerordentlich klare Struktur im Projekt sichergestellt werden.

2.3.1 Zusammenspiel .htaccess und index.php

Um die Umsetzung des MVC Patterns zu realisieren ist im Projekt-Root eine .htaccess Datei angelegt worden. Diese sorgt für die korrekte Umleitung aller Anfragen auf die index.php, welche mit Hilfe der Funktion spl_autoload_register alle benötigten Klassen nachlädt. In der index.php wird die URL aufgebrochen und anschließend der gewollte Controller aufgerufen.

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

Im .ebextensions Ordner findet man .config Dateien. Diese werden für die AWS Elastikbeanstalk Umgebung verwendet. Sobald eine neue Version durch Travis CI deployed wurde werden die definierten Befehle in der .config Datei ausgeführt. In diesem Projekt wurde dies verwendet um Bower nutzten zu können.

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.

5 Datenbank

aws

Zum Hosten der Datenbank wird eine AWS RDS for MariaDB mit der MariaDB Version 10.2.21 verwendet. Dadurch das die Datenbank in der Cloud gehostet ist, kann jeder Entwickler daraufzugreifen. Um jedoch Berechtigt zu sein darauf zuzugreifen unterstüzt AWS sogenannte Security Groups. Diese Gruppen bindet man einer Ressource wie z.B. der Datenbank oder dem Elastibeanstalk 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.

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.

7 Fazit

7.1 Gewonnene Erkenntnisse

7.2 Ausblick

Anhang

A.