Code Academy

Studienarbeit über die bearbeiteten Themen
von Daniel Drescher

1. Vision

Unser Projekt heißt Code Academy und soll es neu anfangenden Programmierern möglichst einfach machen zu Lernen. Traditionell wird Programmieren häufig aus Büchern und Dokumentationen gelernt, aber vor allem bei jungen Leuten wirkt das häufig abschreckend. Deshalb wollen wir es mit kurzen Erklärungen und Code-Beispielen möglichst einfach gestalten Programmieren zu lernen.

2. Funktionale Anforderungen

Alle Stakeholder: Nutzer, Entwickler, Autoren, Projektleiter

Anforderung: Online Compiler und Runtime für einfache Benutzung.
Betroffene Stakeholder: Nutzer, Entwickler, Projektleiter
Auswirkung auf Stakeholder: Die Nutzer erwarten, dass die Webapp auf allen Geräten funktioniert und keine komplizierte Installation notwendig ist. Die Entwickler müssen den Compiler bzw. die Runtime implementieren. Der Projektleiter muss Budget für das Feature mit einplanen.

Anforderung: Lessons sind einfach und verständlich aufgebaut.
Betroffene Stakeholder: Nutzer, Autoren
Auswirkung auf Stakeholder: Die Nutzer erwarten, dass die Lessons ihnen das Lernen leicht machen und verständlich sind. Die Autoren müssen die Lessons so gestalten, dass sie den Anforderungen entsprechen.

Anforderung: Der Nutzer bekommt XP für fertige Lessons.
Betroffene Stakeholder: Nutzer, Entwickler
Auswirkung auf Stakeholder: Der Nutzer bekommt eine Belohnung dafür, Lessons abzuschließen. Die Entwickler müssen die XP implementieren.

Anforderung: Der Nutzer sieht wie weit er schon bei einem Kurs ist und bekommt ein Popup, wenn der Kurs abgeschlossen ist.
Betroffene Stakeholder: Nutzer, Entwickler
Auswirkung auf Stakeholder: Der Nutzer kann einfach sehen, wie weit er schon gekommen ist und wenn er den Kurs abgeschlossen hat.

Anforderung: Die Kurse und Lessons sind immer aktuell.
Betroffene Stakeholder: Nutzer, Autoren, Projektleiter
Auswirkung auf Stakeholder: Der Nutzer erwartet immer die neuesten, relevanten Technologien zu lernen. Die Autoren müssen sich darum kümmern, dass die Kurse immer auf dem aktuellen Stand sind. Der Projektleiter muss damit rechnen, dass das Projekt nach Fertigstellung weiter unterhalten werden muss.

3. Nichtfunktionale Anforderungen

3.1. Entwicklungsanforderungen:

- Zur Entwicklung wird React und Laravel verwendet
- Als zusätzliche Technologie wird MySQL und Docker verwendet
- Die App wird für alle Geräte mit modernen Browsern gemacht

3.2. Performanzanforderungen:

- Die Ausführung des Codes und die Rückgabe soll nicht länger als 5 Sekunden dauern
- Die Seite darf beim ersten Mal nicht länger als 3 Sekunden laden
- Das Wechseln zwischen einzelnen Seiten soll sofort passieren

3.3. Benutzbarkeitsanforderungen:

- Das Layout muss intuitiv und leicht verständlich sein
- Kurse und Lessons brauchen visuelle Indikatoren für den erreichten Fortschritt
- Nach der Ausführung des Codes muss dem Nutzer klar gemacht werden, ob er die Stunde geschafft hat, bzw. welche Art von Fehler vorliegt

3.4. Informationssicherheitsanforderungen:

- Die Passwörter der Nutzer müssen verschlüsselt in der Datenbank gespeichert werden
- Die Daten der Nutzer auf dem Server dürfen nicht von unautorisierten Personen eingesehen werden
- Code vom Nutzer stellt ein großes Sicherheitsrisiko dar und darf deshalb nur in Containern ausgeführt werden

4. Prozessdokumentation

4.1. Definition of Ready:

Ein Feature wird als „ready“ angesehen, wenn ein Liste an eindeutigen und klaren Anforderung festgelegt wurde. Außerdem muss festgelegt werden, ab wann das Feature als „done“ angesehen wird, bzw. wie man die Funktionalität des Features testen kann. Das Feature muss in so kleine Abschnitte aufgeteilt werden, dass es innerhalb eines Sprints fertiggestellt werden kann. Das Team muss in der Lage sein, die nötige Zeit für die Implementierung aus der Definition abzuschätzen.

4.2. Definition of Done:

Damit ein Feature als „done“ angesehen wird, muss es alle festgelegten Anforderungen erfüllen. Außerdem muss es mindestens in dem vorher festgelegten Maße getestet sein. Des Weiteren muss die Performanz überprüft werden und es müssen alle Standards für lesbaren und übersichtlichen Code eingehalten werden. Eine Dokumentation zu dem Feature muss entweder schriftlich in einem separaten Dokument oder als Kommentare im

Code vorhanden sein. Nur teilweise oder fehlerhaft implementierte Features werden nicht in das Sprint-Review mit aufgenommen.

4.3. Verwendete Tools:

Als Versionskontrollsystem wird GitHub verwendet, da es eine einfache Zusammenarbeit ermöglicht.

Zur Planung der Sprints wird Trello verwendet. Die interaktiven Karten machen die Planung einfach. Außerdem werden Informationen über den Status des Sprints mithilfe von farbigen Tags übersichtlich dargestellt.

Für Online-Meetings und Chatting wird Discord verwendet. Die Online-Meetings finden immer nach der Hälfte eines Sprints statt, um jedem einen Überblick über den Status des Projekts zu geben und aufkommende Fragen zu beantworten. Außerhalb dieser Meetings kann jederzeit der Chat zur Kommunikation verwendet werden.

5. Produktdokumentation

5.1. Frontend

Für das Frontend der Code Academy wurde React verwendet. React ist eine Bibliothek, die für das Erstellen von Single-Page-Applications verwendet wird. Für die Navigation zu verschiedenen Seiten innerhalb der Application wurde react-router verrwendet. Die wichtigsten Komponenten hierfür sind App und Layout.

5.1.1. App

In der App-Komponente werden alle Routen der Application festgelegt und mit der Layout-Komponente umgeben.

5.1.2. Layout

In der Layout-Komponente wird das generelle Layout der Application festgelegt. Oben auf der Seite befindet sich der Header mit dem Logo, welches ein Link auf die Startseite ist und einem Link auf die User-Seite. Außerdem gibt es einen Login/Logout-Button. Dieses Layout wurde gewählt, weil viele Seiten genau so aufgebaut sind und die Nutzer deshalb schon damit vertraut sind.

5.1.3. Startseite

Auf der Startseite wird den Nutzern zuerst das Konzept der CodeAcademy erklärt. Direkt darunter befindet sich ein Formular zur Registierung, um neue Nutzer dazu zu bewegen, sich einen Account zu erstellen und das Angebot zu nutzen.

5.1.4. Register-Seite

Auf dieser Seite kann der Nutzer, genau wie auf der Startseite, einen Account erstellen. Hierfür muss er einen Benutzernamen, eine Email-Adresse und ein Passwort angeben. Nach einer erfolgreichen Registrierung wird der Nutzer automatisch angemeldet und auf die Kurs-Übersicht weitergeleitet.

5.1.5. Login-Seite

Auf dieser Seite kann der Nutzer ganz einfach seine Email-Adresse und sein Passwort angeben und sich anmelden. Nach einer erfolgreichen Anmeldung wird der Nutzer ebenfalls automatisch auf die Seite mit der Kurs-Übersicht weitergeleitet.

5.1.6. Kurs-Übersicht

Oben in der Kurs-Übersicht werden alle zuletzt angesehenen Kurse angezeigt. Darunter gibt es eine Liste mit allen Kursen.
Die Kurse selbst werden als Karte dargestellt, auf der der Titel des Kurses, ein Profilbild und eine Progress-Bar, die anzeigt wie viele XP der Nutzer in diesem Kurs bereits gesammelt hat.

5.1.7. Kurs-Seite

Wenn man dann eine der Kurs-Karten anklickt kommt man auf die dazugehörige Kurs-Seite. Auf dieser findet man zuerst nochmal den Titel und eine kurze Beschreibung des Kurses. Darunter befindet sich eine Liste mit allen Lehreinheiten des Kurses zu sehen. Diese Lehreinheiten werden ähnlich wie die Karten für die Kurse dargestellt. Auf ihnen ist der Titel und die erreichbare Menge an XP abgebildet. Bei erfolgreich abgeschlossenen Lehreinheiten wird außerdem ein grüner Haken dargestellt.

5.1.8. Lesson

Auf der Seite einer Lehreinheit wird dem Nutzer zuerst wieder der Titel und eine kurze Beschreibung angezeigt. Direkt darunter befindet sich ein Texteingabefeld mit teilweise vorgefertigtem Code, den der Nutzer ergänzen muss. Danach kann der Nutzer den Run-Button drücken woraufhin die Ausgabe des Codes in einem darunterliegenden Textfeld angezeigt wird. Ganz am Ende der Seite gibt es noch zwei Buttons, die zur nächsten Lehreinheit und zurück zur Übersicht navigieren.

5.2. Backend

5.2.1. Web-Routen

Da es sich bei CodeAcademy um eine Single-Page-Application gibt es nur eine Route in der web.php Datei, bei der die gesamte Seite ausgeliefert wird.
Die zweite Route dient nur dem Zweck, auch dann die Seite auszuliefern, wenn ein Nutzer z.B. ein Lesezeichen auf einer Unterseite gesetzt hat.

5.2.2. API-Routen

Die Kommunikation zwischen dem Frontend und dem Backend findet über eine REST-API statt. Diese Routen werden in der api.php festgelegt und leiten die Anfragen an Controller weiter.

5.3. Controller

In Laravel erfüllen die Controller vor allem den Zweck, Zugriffe auf Daten in der Datenbank über REST-Anfragen zu ermöglichen und zu steuern, können aber auch andere Zwecke, wie z.B. die Ausführung des Codes, erfüllen. Die meisten Controller haben die gleiche Struktur, die deshalb im Anschluss zusammengefasst erklärt wird. Dieses Schema wird in dem Controller für die Kurse und Lehreinheiten verwendet.

5.3.1. Standardmethoden

Index-Methode:
Diese Methode gibt alle Elemente einer Ressource zurück. Wenn es sich um eine geschachtelte Ressource handelt, wie z.B. die Lehreinheiten innerhalb der Kurse, dann werden natürlich nur relevante Einträge für einen Kurs zurückgegeben.

Show-Methode:
Diese Methode erhält eine ID und gibt anhand dieser eine spezielle Ressource zurück.

Store-Methode:
Diese Methode erzeugt einen neuen Eintrag in der Datenbank mithilfe von den Daten, die in der gesendeten Request übergeben wurden.

Update-Methode:
Diese Methode bekommt eine ID und die Daten aus der Request und verändert einen spezifischen Eintrag in der Datenbank anhand der Daten aus der Request.

Delete-Methode:
Diese Methode löscht einen Eintrag aus der Datenbank anhand einer übergebenen ID.

5.3.2. CourseController

Der CourseController hat alle oben beschriebenen Standardmethoden implementiert. Nur die Index-Methode liefert neben einer Liste aller Kurse auch die Liste der zuletzt angesehenen Kurse eines Nutzers.

5.3.3. LessonController

Dieser Controller hat auch die Standardmethoden, aber auch hier liefert die Index-Methode zusätzlich die Daten, welche Lehreinheiten von dem Nutzer bereits abgeschlossen wurden.

5.3.4. CodeController

Die Aufgaben des CodeControllers sind komplexer und gehen über das Abrufen von Informationen aus der Datenbank hinaus. Deshalb wird die Funktionalität des Controllers im Folgenden genauer beschrieben.

Dem Controller wird in der Request die lesson_id und der Code des Nutzers übergeben. Aus der Datenbank wird der vorgegebene Code, die Sprache und die erwartete Ausgabe geholt und das <usercode>-Tag mit dem Code des Nutzers ersetzt. Anschließend wird die entsprechende private Methode für die Sprache ausgeführt.

In dieser Methode wird der Docker-Container mithilfe der Spatie/Docker library gestartet und der Code in eine Datei kopiert. Diese Datei wird dann je nach Sprache compiliert und ausgeführt. Diese Methode gibt einen return-Wert und den ausgegebenen Text zurück.

Wenn der Nutzer alles richtig gemacht hat wird die Lesson in der Datenbank als fertig gespeichert. Außerdem wird überprüft, ob der Nutzer alle Lessons fertig gemacht hat. Dann wird eine Response zurückgegeben mit den Werten „success“, der das Popup im Frontend kontrolliert, „text“ gibt das Output zurück, „status“ einen Statuscode und „courseCompleted“ wenn der Kurs beendet wurde.

5.4. Code Execution

Der Code der Nutzer wird in Docker Containern ausgeführt. Dies führt zu längeren Wartezeiten, da der Container erst gestartet werden muss, bringt aber signifikante Vorteile im Bereich Sicherheit, weil der Code nicht direkt auf dem Server, sondern isoliert in einem Container ausgeführt wird. Dass der Nutzer eine Endlosschleife in den Code schreibt und den Server so dauerhaft belastet oder sehr viele Objekte erzeugt und somit den Speicherplatz belegt, wird durch das Memory- und Time-Limit in der php.ini Datei verhindert.

5.4.1. Python Docker Container:

Nutzt das Image python:3. Zusätzlich wir das Verzeichnis /usr/src/myapp erstellt. Außerdem wird der Befehl sleep 5 ausgeführt, da der Container sonst ohne virtuelles Terminal sofort terminiert.

5.4.2. Java Docker Container:

Nutzt das Image openjdk:11. Zusätzlich wir das Verzeichnis /usr/src/myapp erstellt. Der Java Container braucht den sleep-Befehl nicht, da die JShell so lange zum starten braucht, dass der Container nicht zu schnell terminiert bevor der Code ausgeführt wird.

5.4.3. Javascript Docker Container:

Nutzt das Image node:18. Zusätzlich wir das Verzeichnis /usr/src/myapp erstellt. Außerdem wird der Befehl sleep 5 ausgeführt, da auch der Javascript Container ohne Terminal sofort terminiert.

5.5. Javascript Lessons

Die Javascript Lehreinheiten halten sich in etwa an die gleiche Struktur wie die der anderen beiden Kurse. Allerdings wurden natürlich ein paar Dinge, wie zum Beispiel die Wichtigkeit der optionalen Semikolons, mit aufgenommen bzw. ausführlicher bearbeitet.