Dokumentation

der App Code Academy

Engineer-Trio:
Daniel Drescher
Taylan Özdabak
Serhat Güler

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. Use Cases

4.1. Register

Name: Register

Kurze Beschreibung: Der Nutzer kann sich registrieren.

Vorbedingungen: Eine valide E-Mail Adresse, ein valides Passwort sowie ein valider Nutzername

Typischer Ablauf:

1. Nutzer gibt seine Daten auf der Startseite an
2. Nutzer drückt auf den Knopf „Register“

Nachbedingungen:

- Der Nutzer ist registriert
- Der Nutzer ist automatisch auch eingeloggt
- Der Nutzer wird auf die personalisierte Startseite weitergeleitet

4.2. Login

Name: Login

Kurze Beschreibung: Der Nutzer kann sich einloggen.

Vorbedingungen: Ein Account

Typischer Ablauf:

3. Nutzer kann auf der Startseite oben-rechts auf das Account-Icon clicken.
4. Nutzer gibt seine E-Mail Adresse im entsprechenden Feld ein
5. Nutzer gibt sein Passwort im entsprechenden Feld ein
6. Nutzer drückt auf den Knopf Sign in
7. Der Nutzer wird eingeloggt
8. Der Nutzer wird auf die personalisierte Startseite für Nutzer weitergeleitet

Alternative Abläufe:

1. Der Nutzer gibt falsche Daten ein
2. Ein sog. Alert erscheint mit der Nachricht „Invalid credentinals!“

Nachbedingungen:

- Der Nutzer ist eingeloggt
- Der Nutzer kann den Dienst vollständig nutzen

4.3. Lesson bearbeiten

Name: Lesson bearbeiten

Beschreibung: Der Nutzer bearbeitet eine Lesson

Akteure: Nutzer, Autoren (passiv)

Vorbedingungen: Nutzer hat einen Account erstellt und befindet sich auf der Home-Seite, es sind Lessons vorhanden

Typischer Ablauf:

1. Der Nutzer wählt den Kurs aus, den er bearbeiten möchte
2. Der Nutzer wählt eine Lesson des Kurses
3. Der Nutzer liest die Anleitung durch
4. Der Nutzer gibt den richtigen Code in dem Editor ein
5. Der Nutzer drückt auf den Run-Button
6. Die Statusbar wird grün, die Ausgabe erscheint im Output-Feld und ein Popup, das dem Nutzer gratuliert, erscheint

Alternative Abläufe:

1. Der Nutzer gibt Code mit einem syntaktischen Fehler ein
2. Der Nutzer drückt auf den Run-Button
3. Die Statusbar wird rot und der Fehler erscheint im Output-Feld

1. Der Nutzer gibt Code mit einem logischen Fehler ein
2. Der Nutzer drückt auf den Run-Button
3. Die Statusbar wird blau und das Output erscheint im Output-Feld zusammen mit dem Hinweis, dass es fehlerhaft ist

Nachbedingungen:

- Die Lesson wird für den Nutzer als abgeschlossen gespeichert

4.4. Kurs beginnen

Name: Kurs beginnen

Beschreibung: Der Nutzer beginnt einen Kurs

Akteure: Nutzer, Autoren (passiv)

Vorbedingungen: Der Nutzer hat sich registriert und ist auf der Home-Seite

Typischer Ablauf:

1. Der Nutzer wählt einen Kurs aus
2. Der Nutzer bearbeitet eine Lesson und schließt diese ab

Nachbedingungen:

- Der Kurs wird auf der User-Seite unter „started courses“ angezeigt

4.5. Kurs beenden

Name: Kurs beenden

Kurze Beschreibung: Der Nutzer beendet erfolgreich einen Kurs

Vorbedingungen: Der Nutzer muss einen validen Account haben, angemeldet sein und einen Kurs gestartet haben

Typischer Ablauf:

1. Der Nutzer befindet sich in der Kursansicht
2. Der Nutzer klickt auf einen beliebigen Kurs
3. Der Nutzer startet den Kurs, indem er eine Lerneinheit erfolgreich absolviert
4. Der Nutzer erreicht 80% der zu erreichbaren XP des jeweiligen Kurses

Nachbedingungen:

- Der beendete Kurs wird nun in dem Nutzer Profil angezeigt als „Completed Course“
- Der Nutzer erhält eine Errungenschaft für das Beenden des Kurses

4.6. Logout

Name: Logout

Kurze Beschreibung: Der Nutzer meldet sich ab

Vorbedingungen: Der Nutzer muss einen validen Account haben und angemeldet sein.

Typischer Ablauf:

1. Der Nutzer klickt auf den „Logout“ Knopf, welcher sich oben rechts befindet
2. Der Nutzer wird abgemeldet

Nachbedingungen:

- Das aktuelle Token des Nutzers wurde vom Server gelöscht

5. Prozessdokumentation

5.1. 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.

5.2. 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.

5.3. Verwendetet 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.

6. Produktdokumentation

6.1. Frontend

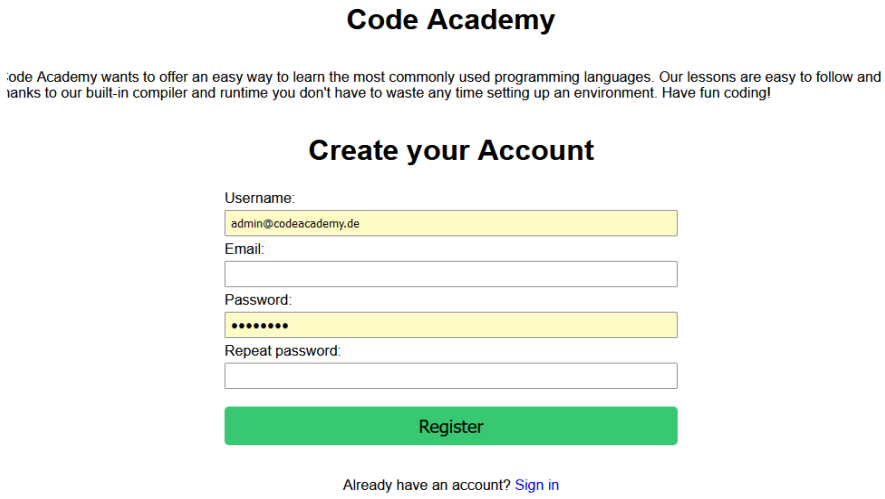
Zur Darstellung unserer Web-Anwendung benutzen wir „React“.
React ist eine Javascript-Softwarebibliothek, welche von Facebook entwickelt wurde. Heutzutage ist es eines der am häufigsten benutzten Frameworks für das Frontend im Bereich Webentwicklung.

Im folgenden werden die einzelnen Seiten unserer Webanwendung präsentiert:

6.1.1. Layout

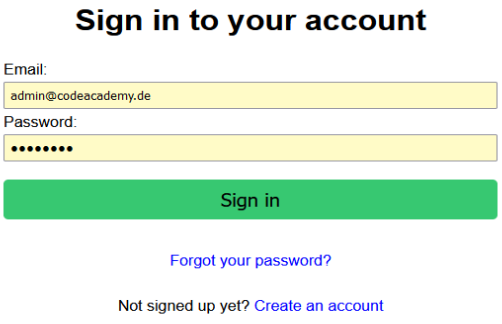
Das generelle Layout unserer Anwendung enthält einen einheitlichen Header und Footer. Der Header besteht aus einem Home Button, einem Login/Logout Button und einem User-Icon, um auf die User Page zu navigieren. Unser Header wird in der Farbe grün angezeigt. Unser Footer ist ein recht standarmäßiger Copyright Footer, welcher in der Farbe Grau dargestellt wird.

6.1.2. Startseite



Auf der ersten Seite die der Nutzer sieht, wird das Konzept unserer CodeAcademy erklärt und das Erstellen eines Accounts ermöglicht.
Wie gewohnt muss der Nutzer bei der Registrierung einen Nutzernamen, eine Email und ein Passwort in die jeweiligen Felder eingeben.
Falls der Nutzer schon einen Account erstellt hat, kann er auf „Sign in“ drücken, um sich einzuloggen.

6.1.3. Login-Seite



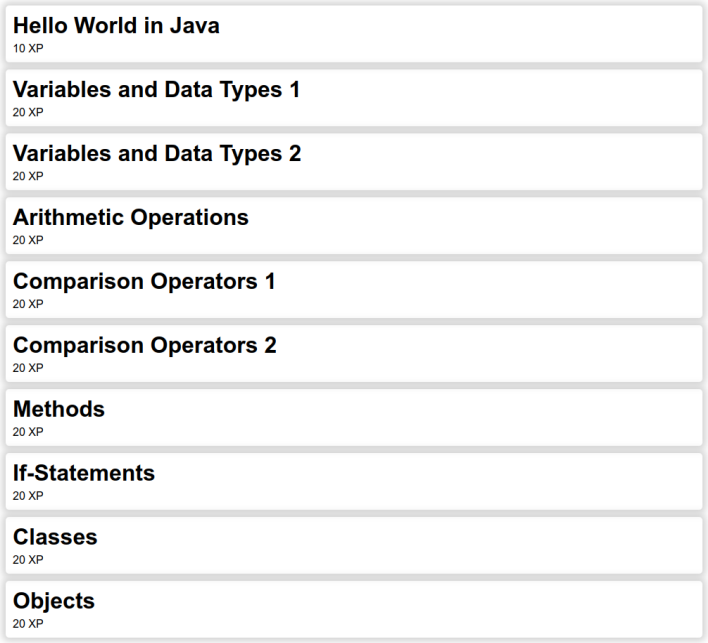
Wie oben schon erwähnt, erreicht man diese Seite, wenn man auf „Sign in“ drückt. Aber es ist auch möglich, auf den „Login“ Knopf oben rechts zu klicken, um auf diese Seite zu gelangen.

Hier gibt der Nutzer seine E-Mail-Adresse und sein Passwort für das Konto, welches er schon erstellt hat, ein.
Auf dieser Seite ist wieder ein Link, um auf die Registrierung zurückzukehren, falls der Nutzer doch noch nicht einen Account erstellt haben sollte.
Nach einer erfolgreichen Anmeldung, erreicht der Nutzer die Kurs Seite.

6.1.4. Kurs-Seite

In der Kursansicht, werden oben die zuletzt besuchten Kurse und unten alle angebotenen Kurse angezeigt.
Jeder Kurs hat eine XP-Leiste unter sich. Diese Leiste zeigt den Fortschritt des Nutzers im jeweiligen Kurs an. XP steht hier für „Erfahrungspunkte“ und sobald 80% der XP erreicht wurden, gilt der Kurs als beendet.
Außerdem kann man oben rechts sehen, dass sich der „Login“ Knopf zu „Logout“ geändert hat, da sich der Nutzer erfolgreich eingeloggt hat.

6.1.5. Lesson-Liste

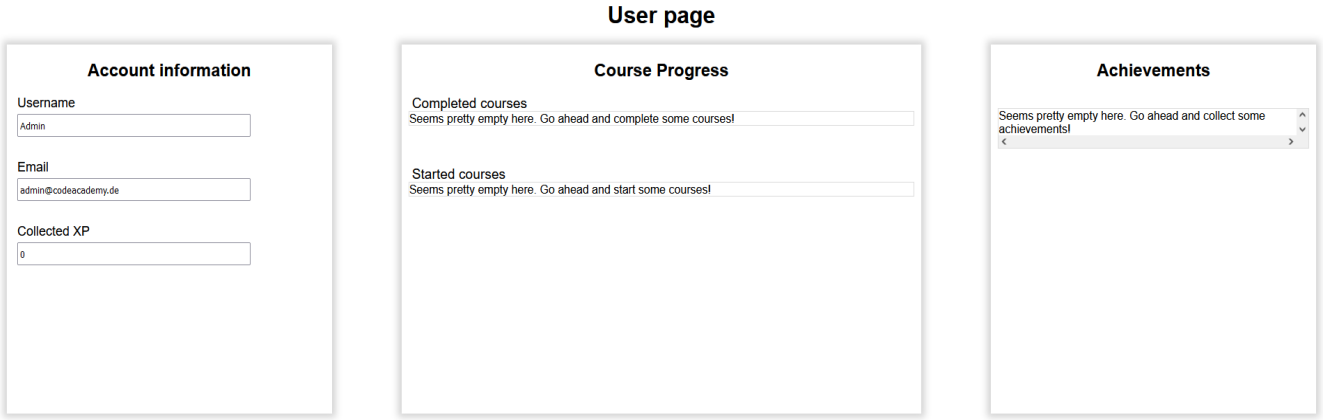


Diese Seite ist zu sehen, nachdem der jeweilige Kurs angeklickt wurde. Jeder Kurs hat 10 Lerneinheiten, welche die Grundlagen der jeweiligen Programmiersprache behandeln. Unterhalb der Titel von einer Lerneinheit stehen die XP, die der Nutzer kriegen kann. Die Liste der Lerneinheiten ähneln sich in allen 3 Kursen, da alle Kurse nur die Grundlagen thematisieren. Sobald der Nutzer eine Lerneinheit erfolgreich abgeschlossen haben sollte, erscheint ein Haken neben der Lerneinheit um zu verdeutlichen, dass die Lesson abgeschlossen ist.

6.1.6. Lesson-Seite

Hier ist die erste Lerneinheit vom Java Kurs zu sehen. Es beginnt mit einer Aufgabenstellung, in der dem Nutzer gesagt wird was er zu tun hat. Direkt darunter ist ein vorgegebener Code Ausschnitt, welchen der Nutzer ausfüllen muss um die Lerneinheit zu bestehen. Durch das Drücken von „Run“, wird der Code ausgeführt und im unteren Feld wird die Ausgabe vom Code ausgegeben.
Sollte die Ausgabe richtig sein, erhält der Nutzer eine Pop-up Nachricht und kann zur nächsten Lerneinheit gehen.
Bei einer falschen Ausgabe, wird der Nutzer darauf hingewiesen, dass seine Eingabe falsch war und die Aufgabe erneut versuchen soll.

6.1.7. User-Seite



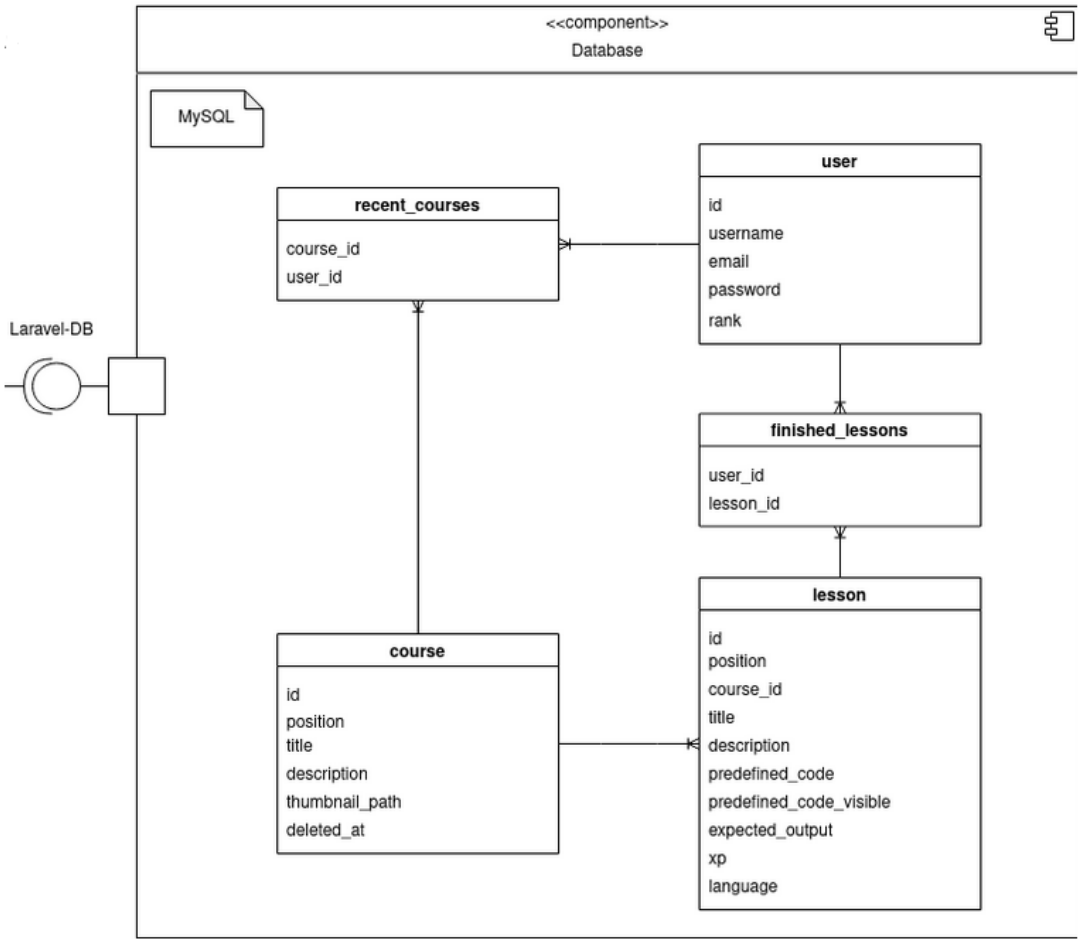
Auf der Nutzerseite werden links die Account Informationen gespeichert und wiedergegeben.
In der Mitte ist der Kurs Fortschritt zu sehen. Hier werden beendete und gestartete Kurse angezeigt. Ein Kurs gilt dann als gestartet, wenn mindestens eine der Lerneinheiten erfolgreich abgeschlossen wurde. Kurse sind erst dann beendet, wenn der Nutzer mehr als 80% der XP erreicht hat.
Auf der rechten Seite werden Errungenschaften, wie zum Beispiel „Complete Java Course“ oder „Complete Python Course“ angezeigt.

6.1.8. Dependencies

- Code Editor:
  Für den eingebauten Code Editor im Frontend, welcher Syntaxhighlighting zur Verfügung stellt wird die npm Abhängigkeit react-simple-code-editor (https://www.npmjs.com/package/react-simple-code-editor) verwendet. Da diese von Haus aus keine Zeilenangaben bei Code unterstützt, wurde zusätzlicher selbstgeschriebener Code eingebaut im Frontend. Zeilenangaben ist für CodeAcademy unerlässlich, da Anwender sonst nur schwer erkennen könnten, auf welcher Zeile jetzt der jeweilig besagte Fehler tatsächlich im Code liegt.

- Loading Bar:
  Für die Loading Bar, die verwendet wird um den Status der Ausführung des Codes anzuzeigen, wird eine Komponente einer npm Abhängigkeit benutzt. Diese stammt von @mui/material (https://www.npmjs.com/package/@mui/material) und ist die Komponente LinearProgress. Mit dieser geben wir mithilfe von hex codes zwei Farben an, ein mal die primäre und sekundäre um einen schönen visuellen Effekt zu erzielen.
- Html-react-parser:
  Der Text von Lektionen ist eine einfache Zeichenkette, die HTML-Elemente enthalten kann. Deshalb bedarf es der npm Abhängigkeit html-react-parser (https://www.npmjs.com/package/html-react-parser). Diese nimmt die Zeichenkette und setzt an entsprechende Stellen die jeweilig definierten HTML-Elemente für eine schöne Formatierung.

6.2. Datenbankschema



In unserer MySQL Datenbank speichern wir folgende 5 Tabellen: user, lessons, courses, recent_courses und finished_courses.
Alle der 5 Tabellen haben verschiedene Attribute, die einen bestimmten Datentypen erwarten.

Die Tabellen recent_courses und finished_lessons sind ähnlich aufgebaut und werden deshalb hier gleich behandelt. Es wird einfach jeweils die ID des Nutzers und die ID des besuchten Kurses/der abgeschlossenen Lesson gespeichert.

Die users Tabelle hat neben den Standard id, username, email und password auch ein Feld rank, in dem gespeichert wird, ob es sich um einen Standardnutzer oder Admin handelt.

Die courses Tabelle hat ebenfalls eine ID. Die position gibt an, an welcher Stelle im Frontend der Kurs angezeigt wird. Ein Kurs hat außerdem noch einen Titel und eine Beschreibung. thumbnail_path gibt den Dateipfad/-namen an. deleted_at speichert den Zeitpunkt, an dem der Kurs gelöscht wurde. Aus der Datenbank wird er nicht entfernt, dass Nutzer, die den Kurs bereits begonnen haben, diesen auch beenden können.

Die lessons Tabelle besitzt auch eine ID. Genau wie die Kurse gibt es auch hier position, title und description. Zusätzlich wird die ID des zugehörigen Kurses gespeichert. In predefined_code und predefined_code_visible wird für die Stunde vorgegebener Code gespeichert. In expected_output ist das Output, welches für die Validierung verwendet wird. Außerdem wird gespeichert wie viel XP der Nutzer für die Lesson bekommt und welche Programmiersprache verwendet wird.

Damit der Server problemlos mit der Datenbank interagieren kann, haben wir den eingebauten Datenbank Service „Eloquent“ von Laravel benutzt.

6.3. Routing

6.3.1. Routen

Im Laravel-Framework wird sehr viel mit Routen bearbeitet sowie gearbeitet.

Routen werden ihren Namen gerecht: Mithilfe von Routen kann man die Web-Adresse/Route seiner REST Schnittstelle definieren und was bei Aufruf ausgeführt werden soll.

Ein gängiges Beispiel in CodeAcademy ist, die Kurse zu laden. Das Frontend (Application) sendet dem Backend (Server) auf die Adresse https://code-academy.online/api/run eine Anfrage mit dem JSON Körper, der den code vom Nutzer beinhaltet sowie eine eindeutige Identifikationsnummer der Lektion, die er gerade im Code bearbeitet. Durch die Identifikationsnummer der Lektion weiß der Server, dass es sich bspw. um Java handelt und nicht um Python oder JavaScript. Dann startet er die docker container, wo der Code vom jeweiligen Nutzer ausgeführt wird. Nach Abschluss bekommt validiert der Server die Ausgabe des Codes und überprüft ob die Lektion fertiggestellt wurde und antwortet dem Frontend. In der Route aus dem Beispiel, besteht auch noch eine sog. Middleware und eine Angabe, in welcher Klasse welche Funktion aufgerufen werden soll.

6.3.2. Middleware

Eine Middleware bietet einen praktischen Mechanismus, der nicht viel Schreibarbeit erfordert. Diese kann man ganz einfach in den Routen mithilfe einer Funktion definieren oder sogar zu Routen-Gruppen hinzufügen. Middlewares werden zum Filtern von HTTP-Anforderungen verwendet, die in unsere Anwendung eingehen. Zum Beispiel enthält Laravel eine Middleware, die überprüft, ob der Benutzer Ihrer Anwendung authentifiziert ist. Dadurch kann man nicht-eingeloggten Nutzern Zugriff auf bestimmte Ressourcen des Webservices verweigern. In unserem Fall wäre das nahezu die gesamte Applikation, da Nutzer einen Account anlegen müssen um unsere Dienste nutzen zu können.

6.3.3. Gruppen

In den Routen von Laravel besteht ebenfalls die Möglichkeit sog. Gruppen zu definieren. Eine Gruppe kommt immer dann zur Anwendung, wenn einige Routen Gemeinsamkeiten haben und man nicht viel Code schreiben will und dadurch auch nicht muss. Hier kommt das Stichwort less verbose zum Einsatz, welches weniger geschwätzig übersetzt bedeutet. Wenig Geschwätzigkeit bedeutet, dass der Entwickler nicht viel tippen muss um einfache Sachen zu implementieren. Gemeinsamkeiten von Routen wären bspw. dass diese sich einen Präfix teilen. Beispielsweise statt zwanzig mal für user routen ein user/ manuell anzusetzen, packt man diese einfach in die Gruppe und muss das gar nicht mehr

machen. Wenn man dann eine Route mit /info definiert, wird diese automatisch zu /user/info. In den Gruppen können auch Middlewares definiert werden. In CodeAcademy verwenden wir beispielweise die Login-Check Middleware, um sicherzustellen, dass Nutzer die auf geschützte Bereiche zugreifen, auch wirklich eingeloggt sind.

6.4. Controller

Der Controller ist ein Teil des MVC-Architekturmusters. Das MVC Muster wird benutzt um die Software einzuteilen in model (Modell) , view (Präsentation) und controller (Steuerung). Das model beinhaltet die Daten, die in der Anwendung präsentiert werden.
Die view kümmert sich um Art der Darstellung der Daten des models, jedoch verarbeitet es nicht die Daten vom model.
Der controller fungiert als eine Verwaltungsinstanz zwischen dem model und der view. Benutzerinteraktionen in der view benachrichtigen den controller, welcher daraufhin Änderungen an der view und an den Daten vom model vornimmt.

6.4.1. UserController

Aufgaben vom UserController :

- Registrierung
  - Validierung der Eingabedaten
  - Überprüfung ob Schimpfwörter im Nutzernamen enthalten sind
  - Erstellen eines Nutzers bei erfolgreicher Validierung
  - Erstellen eines Tokens für die Authentifizierung
- Login
  - Validierung der Eingabedaten
  - Überprüfung ob Nutzer existiert
- User Profil
  - Nutzerdaten
  - Erreichte XP
  - Kursfortschritt
  - Errungenschaften
- Logout

6.4.2. CourseController

Aufgaben vom CourseController:

- Kurse anzeigen
- Kurse sortieren
- Zuletzt besuchte Kurse anzeigen
- XP anzeigen

6.4.3. Lesson Controller

Aufgaben vom LessonController:

- Lerneinheiten anzeigen
- Lerneinheiten sortieren
- Wechsel zur nächsten Lerneinheit
  - Letzte Lerneinheit hat keine nächste Lerneinheit → Kurs beendet
- Überprüfung ob Lerneinheit beendet wurde

6.4.4. 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.

6.5. 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.

6.5.1. 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.

6.5.2. 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.

6.5.3. 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.

6.6. Authentifizierung

Die Authentifizierung erfolgt durch den PHP-Server mithilfe von dem Framework Laravel. Ein Framework ist ein Hilfsmittel für Entwickler, in dem oft gängige Konzepte bereits ausgearbeitet wurden und reibungslos funktionieren. In Laravel wurden ebenfalls einige Konzepte von anderen open-source Entwicklern ausgearbeitet. Die Konzepte beziehen sich bei Laravel darauf, was oft bei vielen Webseiten gebraucht wird. Das wären dann zum Beispiel die Authentifizierung der Nutzer des Webservices, verschiedene Datenbankschemen, Web Services mit REST APIs mit Controllern und vielen weiteren Helfern bei gängigen Problemstellungen.

6.6.1. Token

Ein sog. Token wird bei der Authentifizierung zwischen Nutzer und Server genutzt.
Mit dem Token gibt der Nutzer seine Identität an, die der Server kennt. Angenommen „ABCDEFGH123456789“ ist ein Token für den Nutzer Max Mustermann. Wenn dem Server eine Anfrage mit dem Token „ABCDEFGH123456789“ gesendet wird, weiß er dass der Benutzer Max Mustermann ist. Dies funktioniert auch, wenn ein falscher Nutzer den Token sendet. Ein Token ist der Schlüssel zu einem Account. Wenn jemand fremdes den Schlüssel zu der eigenen Haustür hat, kann er auch einfach die Haustür öffnen. Das selbe Prinzip gilt auch hier bei der Token-Authentifizierung.

6.6.2. Laravel Passport und Laravel Sanctum

In Laravel gibt es zwei Authentifizierungmöglichkeiten:

Die eine heißt Laravel Sanctum (fortfolgend Sanctum genannt) und die andere Laravel Passport (fortfolgend Passport genannt). In unserer Anwendung nutzen wir Passport, da unsere Anwendung eine höhere Komplexität besitzt und auch einiger Funktionen von Passport bedarf.

Der signifikante Unterschied zwischen den beiden ist, dass Passport vollständige OAuth2-Authentifizierung unterstützt, während Sanctum API Tokens generiert und diese in den

built-in Laravel Cookies speichert. Diese sind sog. Session-basierte Token, das bedeutet in jedem neuen Tabs müsste sich der Nutzer erneut anmelden. Ebenfalls müsste der Nutzer sich erneut anmelden, wenn nur der Browser geschlossen und wieder geöffnet wird.

Durch die Nutzung von Sanctum braucht der Entwickler keine Kenntnisse von OAuth, sondern kann ein sehr einfaches Login schnell und ohne hohe Komplexität programmieren.

Anders ist das bei Passport: Da Passport OAuth verwendet, sollte man einige Kenntnisse davon haben. Der große Vorteil bei OAuth ist, dass man die Token vom Server im Browser speichern kann und auch Verfallsdaten für die Token festlegen kann.

Beispielweise ist es möglich, ein Verfallsdatum auf genau fünf Tage festzulegen ab dem Zeitpunkt des Logins. Wenn ich innerhalb der fünf Tage erneut ein Tab aufmache, bin ich immer noch eingeloggt, außer ich melde mich innerhalb der Zeit von meinem Nutzer-Account ab. Nach den fünf Tagen wird der Token aus Sicherheitsgründen zerstört und es muss erneut ein Anmeldeversuch durchgeführt werden.

Nach gelungener Anmeldung in den jeweiligen Account bekommt man wieder einen Token vom Server ausgestellt, der fünf Tage gültig ist.

6.6.3. OAuth2

OAuth ist ein offenes Sicherheitsprotokoll für eine tokenbasierte Autorisierung und Authentifizierung im Internet auf verschiedenen Webservices. Bei OAuth2 können Drittanbieters-Webservices einfach auf externe Ressourcen zugreifen, ohne dass Benutzernamen und Passwort zu offenbaren sind und auch nicht in dem lokalen Speicher (fortfolgend local storage genannt) gespeichert werden müssen.

Vor OAuth2 gab es OAuth1. Es wurde aber durch OAuth2 abgelöst, da OAuth1 ein sog. complete rewrite (bei einem rewrite wird von null angefangen und es werden keine alten Komponenten aus dem Vorgänger weiter verwendet) war und neue, wichtige Funktionalitäten mit sich brachte.

OAuth1 hatte noch einige Schwachstellen bzw. fehlende Funktionen.

Beispielsweise mussten OAuth1-Clients noch kryptologische Technik unterstützen. Bei OAuth2 ist das nicht mehr der Fall. Das stellte vor allem bei IoT-Geräten, die minimalistisch sein wollten ein Problem dar, weil Kryptologie nur deswegen eingebaut werden müsste. OAuth2 Tokens haben einen kurzen Lebenszyklus vor der Zerstörung, das war bei OAuth1 ganz anders.

Noch ein Problem war, dass man bei OAuth1 mit dem selben Token über Jahre hinweg eingeloggt bleiben konnte. Das war auf Twitter lange Zeit der Fall. Das stellte ein sicherheitstechnisches Problem dar. Wenn jemand den Token des Accounts erlangt hatte, hatte er nämlich Zugriff auf den Account auf meist unbestimmte Zeit.

6.6.4. XSS-Scripting

Das Speichern von sensiblen Daten (bspw. Passwort vom Nutzer) stellt bei XSS-Scripting ein großes Problem dar.

XSS-Scripting ist auch bekannt unter dem Begriff Cross-Site-Scripting, wo der Angreifer den Nutzer woanders hinleitet ohne seine Kenntnis und der Nutzer zwar visuell meist das selbe wie immer sieht, aber im Hintergrund zusätzlich schädlicher Code des Angreifers ausgeführt wird, die der Angreifer im Vorfeld einbaut hat. Das wird meist dazu genutzt, um aus dem local storage Daten zu klauen. Dank OAuth kann der Angreifer höchstens nur den Token klauen und sich nur bei der Webseite einloggen, von der der Token ausgestellt wurde und nur dort etwas ändern. Das war auch nur auf beschränkte Zeit möglich, da der Token irgendwann wieder abläuft. Da aber keine sensiblen Daten wie Passwörter sich im

local storage befunden haben, kann er mit einem Token nicht so viel anfangen wie mit einem Passwort. Wenn ein Passwort im local storage gewesen wäre, hätte der Angreifer deutlich mehr Schaden anrichten können. Hätte der Angreifer das Passwort gekapert und der Nutzer nutzt überall das selbe Passwort, könnte der Angreifer beispielsweise bei einer Bank mit unsicherem Online-Banking, mit ein paar Tricks das Konto vom Opfer leer räumen.

Der Tokendiebstahl ist bei wichtigen Online-Finanzdienstleistungsanbieter wie PayPal immer noch ein großes Problem. Doch durch die sog. Two-Factor-Authorization (fortfolgend 2FA genannt), die meist als SMS-Code oder TAN für Online-Banking genutzt wird, müsste der Angreifer noch vor schädlichen Aktionen den jeweiligen Code des 2FA-Prozesses bekommen bzw. die Verifizierung von 2FA erlangen.

Hier macht aber 2FA einen Strich durch die Rechnung, denn spätestens jetzt wird der eigentliche Account-Inhaber benachrichtigt, sei es durch eine SMS oder durch einen TAN-Code. Es kann auch ein Knopf in einer App sein, der einfach „Freigeben“ heißt und somit der 2FA-Prozess verifiziert wird. In der Regel werden hier aber die meisten Nutzer stutzig, da sie nichts auf PayPal oder ihrer Online-Bank machen wollte und lassen Zugänge sperren und Passwörter ändern, bevor irgendwelche Betrüger hohen Schaden hätten anrichten können.

Durch 2FA wird dem Angreifer sein Handwerk deutlich erschwert. Außerdem kennen die Finanzdienstleister solche Sicherheitsprobleme und haben deshalb viele Sicherheitsexperten engagiert, die aufpassen, dass kein XSS-Scripting möglich ist auf deren Webseiten.

Bei unserer Anwendung wurden einige XSS-Attacken durchgeführt und es wurden keine Schwachstellen im eigenen Quellcode gefunden. Es kann allerdings so sein, dass Abhängigkeiten welche beinhalten und diese in kommenden Versionen beheben. Dieser Prozess ist jedoch unabhängig von CodeAcademys Quellcode. Da unsere Anwendung

jedoch sowieso keine sensiblen Daten beinhaltet und bearbeitet, wie bspw. Zahlungsarten oder ähnliches, wären XSS-Attacken auch einfach gesagt nutzlos.

6.6.5. Login

Das Login funktioniert in CodeAcademy im Frontend mithilfe von axios, einem einfachen HTTP client womit REST Requests versendet werden kann. Nach dem der Nutzer die E-Mail und das Passwort eingegeben hat und den "Sign in" button betätigt, sendet axios eine Request mit einem JSON body, welcher sowohl die E-Mail als auch das Passwort an den Server übermittelt. Dann wird die Anfrage von Laravel aufgenommen und an den entsprechenden UserController mithilfe der definierten Routen weitergeleitet. Im Controller werden die Daten verifiziert und bei Korrektheit erhält der Client eine Antwort vom Server, die axios mit der eigenen "then"-Funktion bearbeitet. Axios prüft nun die Antwort vom Server, bei Korrektheit wird dann der Token im Local Storage vom Webbrowser gespeichert. Bei falscher E-Mail oder Passwort wird im Frontend visuell mithilfe eines alerts angezeigt, dass die eingegeben Daten falsch sind.

6.6.6. Register

Das registrieren bei CodeAcademy funktioniert genau so wie beim Login mithilfe von axios. Nach dem der Nutzer alle Felder aussgefüllt hat, wird dem Server eine Request übermittelt, womit sich das Frontend registrieren will. Die Anfrage wird genau wie beim Login an die entsprechende Stelle weitergeleitet durch Laravel, dem UserController. Dort wird überprüft, ob der Username ein anstößiger Name ist, ob die E-Mail korrekt ist vom Format her und ob die Passwörter übereinstimmen, da der User das Passwort zwei mal eingeben muss. Ein mal als Angabe und ein mal als Verifizierung für Tippfehler. Falls etwas nicht passt, wird ans Frontend eine Nachricht versendet, wo die entsprechende Stelle (bspw. inkorrektes E-Mail Format) bemängelt wird und der Nutzer bekommt die Nachricht anhand visueller Effekte zu Gesicht. Es entsteht ein rotes Kästchen mit der entsprechenden Nachricht. Falls die Registrierung richtig abgelaufen ist, bekommt der Client genau wie beim Login ein Token, den er im Local Storage speichert.