

Software Architecture Document für die App “Crush It”

von

Lukas Pietzschmann (76010),

Johannes Wimmer (65884),

Vincent Ugrai (76389),

Leon Schugk (74420),

Johannes Schenker (75982)

Contents

1 Einführung und Ziele (JS)	4
1.1 Aufgabenstellung	4
1.2 Qualitätsziele	4
1.3 Stakeholder	4
2 Randbedingungen (JS)	4
2.1 Technisch	5
2.2 Organisatorisch	5
2.3 Konventionen	6
3 Kontextabgrenzung(LP)	7
4 Lösungsstrategie (LS)	7
5 Bausteinsicht (LP, LS)	7
5.1 Übersicht	8
5.2 Backend Übersicht	9
5.3 Datenbank Struktur	10
6 Laufzeitsicht (LP, VU)	10
6.1 Externe Interaktion mit dem System	11
6.2 Interaktionen innerhalb des Systems	12
6.3 Fehler- und Ausnahmeszenarien	13
7 Verteilungssicht	13
7.1 User Experience und User Interface (VU, LP)	13
7.2 Sicherheit (JS)	18
8 Entwurfsentscheidungen (JS)	18
8.1 Frontend	18
8.2 Backend	18
9 Qualitätsanforderungen	18
10 Risiken & tech. Schulden (JS)	19
10.1 Kein Zugriff auf Amazon API	19
10.2 Unerwartete Veränderungen der Anforderungen	19
11 Quellen	19

List of Figures

1	System Übersicht	8
2	Backend Übersicht	9
3	ER Diagramm	10
4	Externe Interaktion - Sequenzdiagramm	11
5	Interne Interaktion - Sequenzdiagramm	12
6	Fehlerszenario - Sequenzdiagramm	13
7	Registrierungs-Bildschirm	14
8	Startseite	14
9	Profil-Bildschirm	15
10	Plan-Bildschirm	16
11	Gruppen-Bildschirm	17
12	Such-Bildschirm	17

1 Einführung und Ziele (JS)

Dieser Abschnitt dient dazu, die entwickelte Software kurz vorzustellen und die Anforderungen an diese aufzulisten. Die Software wurde in Rahmen der Vorlesung "Cloud and Distributed Computing" an der Hochschule Aalen erstellt.

1.1 Aufgabenstellung

Was ist Crush It?

- Crush It ist eine Web-Applikation zum Erstellen und Ausführen von Trainingsplänen ind Gruppen oder alleine
- Sie soll es den Nutzern erleichtern regelmäßig Sport zu treiben

Wesentliche Features:

- Auswahl verschiedener, bereits erstellter Trainingspläne, die je nach trainierten Muskelgruppen sortiert sind
- Kopieren und anpassen der Trainingspläne an individuelle Ansprüche
- Tracking des Fortschritts innerhalb eines Plans
- Bilden von Gruppen, um zusammen zu trainieren. Trainingspläne können sowohl alleine oder in einer Gruppe bearbeitet werden
- Einfaches Finden von Erklärungen der Trainingseinheiten und von Händlern, bei denen benötigtes Equipment gekauft werden kann

1.2 Qualitätsziele

<Entfällt aufgrund der Aufgabestellung>

1.3 Stakeholder

<Entfällt aufgrund der Aufgabestellung>

2 Randbedingungen (JS)

Bei der Entwicklung der Lösung mussten von Beginn an und während der Durchführung verschiedene Randbedingungen beachtet werden. In diesem Abschnitt werden diese erklärt und, falls angemessen, Motivationen beschrieben.

2.1 Technisch

Randbedingung	Erläuterungen, Hintergrund
Zeilplattform	Die Lösung soll als Web-Applikation implementiert werden.
User Interface	Für die Benutzeroberfläche im Browser standen einige Optionen zur Verfügung. Unabhängig von der Wahl muss die Oberfläche sowohl in Desktop Browsern, als auch auf mobilen Endgeräten nutzbar sein.
Architektur	Mikroservice Architecture Pattern, in der jeder Service in einen eigenen Docker Container ausgeführt werden soll.
Datenbank	Es soll eine NoSQL Datenbank (rein oder als hybrid) in mindestens einen Teil der Anwendung benutzt werden.
Backend	Das Backend soll mit NodeJS/Express, LoopBack, Python/Falcon, Flask, oder Django REST Framework, Spring Boot, ktor, oder Go implementiert werden.
Frontend	Das Frontend soll mit react, Angular, Vue.js, Svelte, oder Ionic implementiert werden.
Pub/Sub Real-Time-API	Es soll eine Pub/Sub Api mit Echtzeitdaten benutzt werden. Es ist unerheblich, ob diese Daten echt oder künstlich erzeugt werden.

2.2 Organisatorisch

Randbedingung	Erläuterungen, Hintergrund
Team	Lukas Pietzschmann (76010), Johannes Wimmer (65884), Vincent Ugrai (76389), Leon Schugk (74420), Johannes Schenker (75982)
Zeitplan	Beginn der Entwicklung März 2021. Erste Architektur und Technologie-Entscheidungen (Exploratory Sprint). Erster lauffähiger Prototyp Mai 2021 (Alpha Sprint), Fertigstellung aller Funktionen Juni 2021 (Beta Sprint), Abnahme der Software Juli 2021 (Zertifizierung und Abgabe)
Vorgehensmodell	Iterative Entwicklung, Aufgaben unter Teammitgliedern verteilt. Die Dokumentation wird mit arc42 und Postman erstellt. Ein Software Architecture Document (SWAD) ist zentraler Bestandteil der Projektleistung im Rahmen der Vorlesung.
Entwicklungswerkzeuge	Entwicklung des Quellcodes in Visual Studio Code. Verschiedene Online-Dienste für die Erstellung von Diagrammen.
Versionsverwaltung	Git und Bitbucket

2.3 Konventionen

Randbedingung	Erläuterungen, Hintergrund
SWAD	Entsprechend dem deutschen arc42-Template in der Version 7.0
Sprache (Englisch vs Deutsch)	Klassen, Funktionen, Kommentare, Variablen und ähnliches im Code in Englisch. Dokumentation außerhalb des Quellcodes in Deutsch. Hintergrund: Aufgrund des internationalen Charakters von Software Development ist es üblich, Code in Englisch zu schreiben. Das SWAD ist Teil der Abgabe für den Kurs Cloud and Distributed Development und wird daher in Deutsch verfasst, um das Verständnis für Studenten und Professoren zu erleichtern.
Vorgehensmodell	Iterative Entwicklung, Aufgaben unter Teammitgliedern verteilt. Die Dokumentation wird mit arc42 und Swagger erstellt. Ein Software Architecture Document (SWAD) ist zentraler Bestandteil der Projektleistung im Rahmen der Vorlesung.

3 Kontextabgrenzung(LP)

Im Folgenden werden die verschiedenen Schnittstellen und die Interaktion mit diesen Beschrieben

Benutzer - System Der Benutzer kann mit dem System auf zwei verschiedene Arten interagieren. Zum einen über das User Interface und zum anderen über die API direkt. In beiden Fällen ist der Kommunikationskanal HTTP.

Frontend - Backend Das Frontend kommuniziert mit dem Backend über zwei Protokolle. Für eher statische Daten, wird HTTP verwendet. Für eher dynamische Daten, die frequent geändert werden, werden Websockets verwendet. Wie in Figure ?? zu sehen, besteht das Backend nicht nur aus einer monolithischen Anwendung, sondern aus mehreren kleineren Services. Das Frontend kann allerdings nicht direkt mit diesen zu interagieren. Stattdessen werden alle Anfragen über den Proxy geleitet, der die Anfrage dann an den jeweiligen Service weiterleitet. Die Kommunikation innerhalb des Backends findet ebenfalls über HTTP statt.

Backend - Datenbank Das Backend greift auf die Datenbank mit Hilfe der für die Datenbank spezifischen Abfrage-Sprache zu.

4 Lösungsstrategie (LS)

Um Code-Bloat zu vermeiden und auf Grund der erwarteten etwas kleineren Größe des Projektes, haben wir uns für das Backend für eine RESTful API mit Python Flask entschieden. Aus den gleichen Gründen haben wir uns für MongoDB als Datenbank entschieden auch weil es sehr leicht zu implementieren ist und die Web-Oberfläche von MongoDB ebenfalls sehr angenehm zu benutzen ist. Für das Frontend haben wir uns, schlichtweg wegen vorangegangener Erfahrung mit der Technologie, für React entschieden, damit auch bei Fehlern ein neuer Lösungsansatz leichter zu finden ist und mit weniger Recherche verbunden ist.

5 Bausteinsicht (LP, LS)

Dieses Kapitel behandelt die Struktur des Projektes näher.

5.1 Übersicht

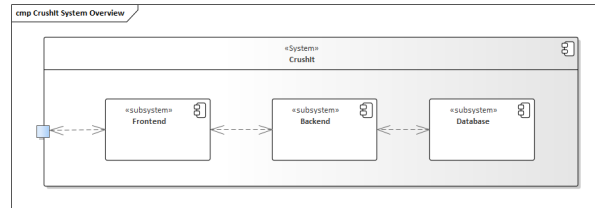


Figure 1: System Übersicht

Diese Übersicht zeigt die grobe Struktur des Systems. So besteht die Applikation CrushIt aus insgesamt drei Elementen. Dem Frontend, dem Backend und der Datenbank. Das Frontend stellt die Benutzer-Schnittstelle nach außen dar. Das Frontend kommuniziert lediglich mit dem Backend und niemals direkt mit der Datenbank. Das Backend hingegen enthält die komplette Logik der Applikation. Es stellt alle Ressourcen zur Verfügung und kann direkt auf die Datenbank zugreifen. Die Datenbank enthält nur minimale Logik, sie stellt primär nur Daten zur Verfügung.

5.2 Backend Übersicht

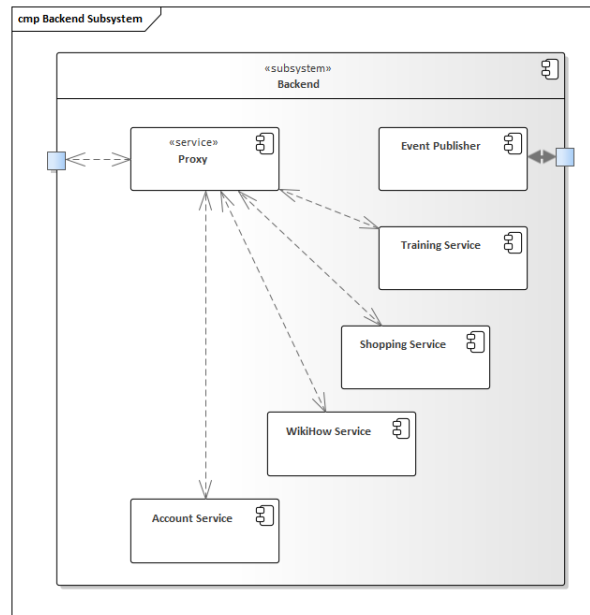


Figure 2: Backend Übersicht

Dieses Diagramm bringt die Struktur des Backends näher. Die zentralste Rolle des Backends nimmt der Proxy ein. Er ist die Haupt-Schnittstelle in das Backend. Alle Anfragen werden an den Proxy gestellt, der diese dann an den korrekten Service weiterleitet. Der Proxy ist allerdings nicht dafür zuständig das meistens angegebene Authentifizierungs-Token zu validieren, das ist die Aufgabe jedes einzelnen Services.

Die zweite Schnittstelle nach außen ist der Event-Publisher. Er nimmt WebSocket Verbindungen an, und verteilt aktuelle Events an verbundene Clients.

Weitere im Backend vorhandene Services sind:

Account-Service Der Account-Service ist der Größte Service. Hier werden neben Benutzern auch Gruppen verwaltet. Auch Anfragen `/login` und `/register` werden hier behandelt.

WikiHow-Service Der WikiHow-Service ist lediglich ein Wrapper um die Daten die die Website `www.wikihow.com` bereit stellt. Dieser Service dient dazu den Zugriff auf WikiHow zu

vereinfachen.

Shopping-Service Der Shopping-Service stellt eine Möglichkeit zur Verfügung nach Trainings-Equipments zu suchen. Dies dient als Ersatz für die Amazon API, die wie in Kapitel 10.1 beschrieben nicht verwendet werden konnte.

Training-Service Der Trainings-Service stellt Trainingspläne und deren einzelne Trainingseinheiten bereit.

5.3 Datenbank Struktur

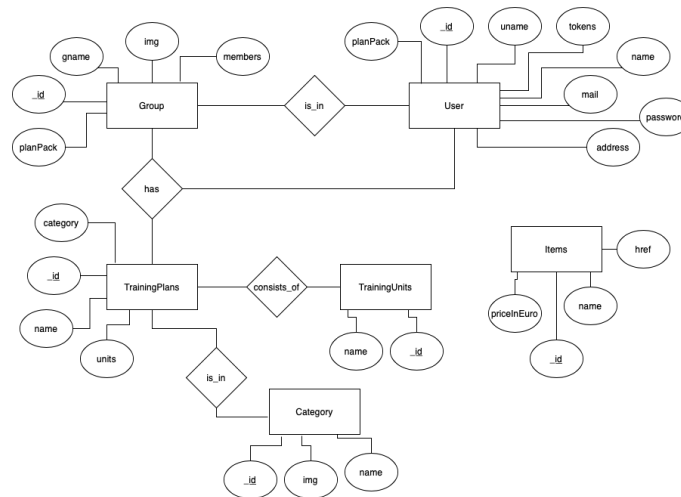


Figure 3: ER Diagramm

6 Laufzeitsicht (LP, VU)

Dieser Abschnitt erklärt konkrete Abläufe und Beziehungen zwischen einzelnen Bausteinen in Form von Szenarien aus den folgenden drei Bereichen:

1. Externe Interaktion mit dem System
2. Interaktionen innerhalb des Systems
3. Fehler- und Ausnahmeszenarien

Mit System ist im Folgenden das Backend gemeint.

6.1 Externe Interaktion mit dem System

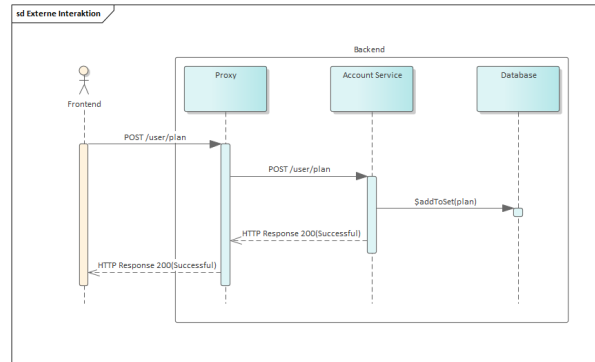


Figure 4: Externe Interaktion - Sequenzdiagramm

Dieses Diagramm zeigt vereinfacht, wie eine Anfrage vom Backend angenommen, verarbeitet und beantwortet wird. In diesem Beispiel soll ein Trainingsplan zu dem Profil des Benutzers hinzugefügt werden. Dazu sendet das Frontend eine HTTP POST Anfrage an die URL `/user/plan`. Im Body dieser Anfrage befindet sich die ID des gewünschten Plans. So wie alle Anfragen, geht auch diese zuerst an den Proxy. Dieser hat lediglich die Aufgabe diese Anfrage an den korrekten Service weiterzuleiten, was in diesem Fall der Account Service ist. Dieser bearbeitet die Anfrage dann, indem er den konkreten Plan zur übermittelten ID abfragt (diese Aktion wird konkreter in Abschnitt 6.2 erläutert). Daraufhin nimmt der Account Service Kontakt zur Datenbank auf, in der er dann den Benutzerdaten den gewünschten Plan hinzufügt. Ist diese Aktion abgeschlossen gibt der Service den HTTP Code 200 (Successful) zurück, welcher durch den Proxy dann letztendlich zum Frontend weitergeleitet wird.

6.2 Interaktionen innerhalb des Systems

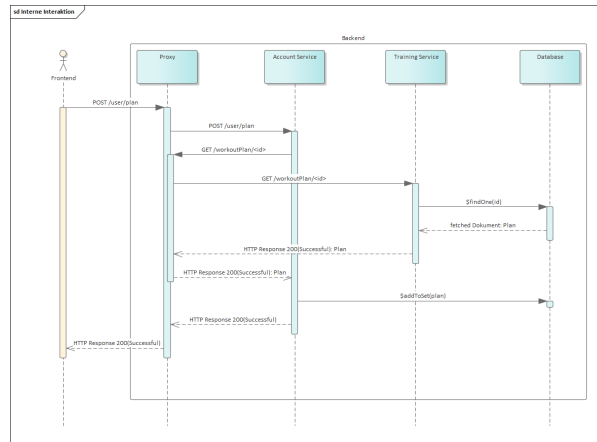


Figure 5: Interne Interaktion - Sequenzdiagramm

Dieses Diagramm greift das vorherige Beispiel auf und zeigt, wie die einzelnen Services miteinander kommunizieren. Bis inklusive der ersten Anfrage an den Account Service ist der Ablauf äquivalent zum vorigen Beispiel. Nun benötigt der Account Service anstelle der ID den konkreten Plan. Dazu stellt er eine HTTP GET Anfrage an `/workoutPlan/<id>`. Diese Anfrage geht wieder zuerst an den Proxy, der sie dann den Training Service weiterleitet. Dieser stellt nun eine Anfrage an die Datenbank um die konkreten Daten zu bekommen. Terminiert diese Anfrage, wird die Antwort an den Proxy zurückgegeben, der sie dann zum Account Service weiterleitet. Eine interne Anfrage funktioniert prinzipiell also genau wie eine von extern. Ab diesem Punkt entspricht der Verlauf nun wieder dem aus dem ersten Beispiel.

6.3 Fehler- und Ausnahmeszenarien

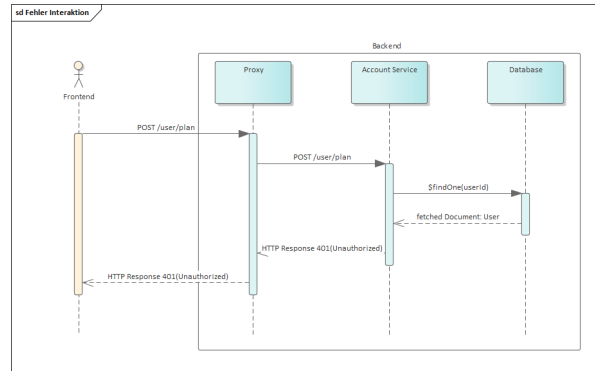


Figure 6: Fehlerszenario - Sequenzdiagramm

Dieses Diagramm beschreibt einen simplen Fehlerfall. Auch hier möchte der Benutzer wieder einen Plan zu seinem Profil hinzufügen. Um diese Aktion ausführen zu dürfen, muss der Benutzer sich authentifizieren, das tut er durch das Hinzufügen seiner UserID und eines validen Tokens in den Header der Anfrage. In diesem Beispiel wird davon ausgegangen, dass der Nutzer kein valides Token angegeben hat. Die Anfrage geht nun wieder zuerst in den Proxy. Dessen Aufgabe ist es nicht die Authentifizierung zu validieren, ergo leitet er die Anfrage an den Account Service weiter. Dieser überprüft nun die UID und das Token und stellt dabei fest, dass das gesendete Token nicht existiert. Um das Frontend auf diesen Fehler aufmerksam zu machen, kategorisiert das Backend jeden Fehler in HTTP Status Codes ein. Der Account Service schickt nun also nicht mehr 200 (Successful) zurück, sondern 401 (Unauthorized). Kommt diese Antwort am Frontend an, kann dieses dann entsprechend reagieren.

7 Verteilungssicht

7.1 User Experience und User Interface (VU, LP)

Beim erstellen der Website haben wir uns anhand eines Modelles überlegt wie unsere Web-Applikation sich bedienen lassen sollte.

Zum Nutzen unsere Web-Applikation benötigt der Nutzer einen Account. Wenn er nicht angemeldet ist oder kein Account besitzt, startet die Applikation auf der Login Seite, von der aus man sich auch registrieren kann.

Registriert der Benutzer sich, wird ihm visuell durch ein Sternchen visuell hervorgehoben, welche Felder ausgefüllt werden müssen.

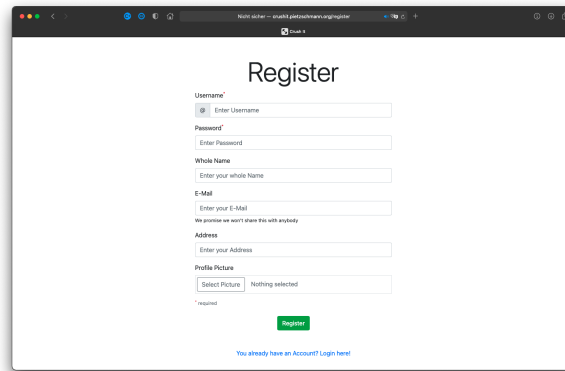


Figure 7: Registrierungs-Bildschirm

Besitzt der Nutzer dann einen Account, meldet er sich mit Hilfe seines Benutzernamens und seines Passworts an.

Nun wird dem Nutzer die Startseite präsentiert.

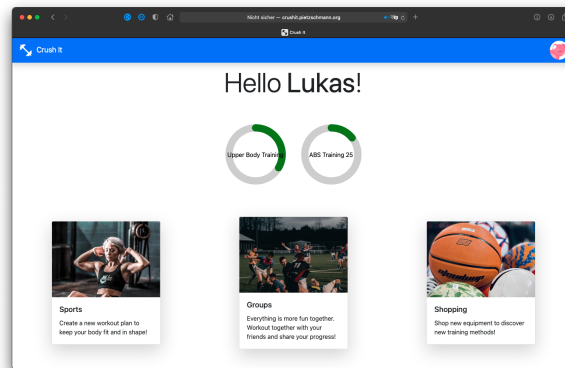


Figure 8: Startseite

Der oberhalb der Startseite angezeigte Header wird auf jeder Seite angezeigt. Hier findet der Nutzer neben dem Applikationsnamen und Titel auch schnellen Zugriff auf seinen Account, indem er auf das Profilbild klickt.

Navigiert der Nutzer also dorthin, kann er alle angegebenen Informationen einsehen und abändern. Dabei hat er allerdings zwei Einschränkungen: So kann der Benutzername nicht verändert werden und das Profilbild darf, genau wie bei der Registrierung, nicht größer als 1MB sein.

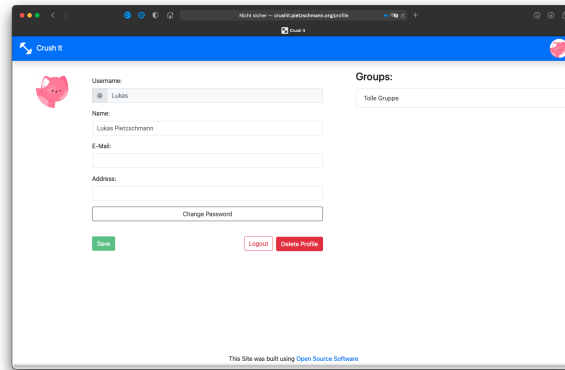


Figure 9: Profil-Bildschirm

Die Startseite zeigt ansonsten alle wichtigen Navigationspunkte. In der unteren Hälfte lassen sich unter *Sports* alle Kategorien nach relevanten Plänen durchsuchen, unter *Groups* kann der Nutzer dann seine Gruppen einsehen, neue erstellen oder beitreten. *Shopping* ermöglicht es dem Nutzer nach Equipment zu suchen, welches er für Trainingseinheiten benötigt. Die obere Hälfte dieser Seite zeigt eine kurze Zusammenfassung aller Trainingspläne die der Nutzer aktuell bearbeitet. Hier wird neben dem Namen auch der Fortschritt angezeigt. Klickt der Nutzer auf einen dieser Pläne, wird er auf eine neue Seite weitergeleitet, auf der eine detaillierte Sicht aller seiner Pläne angezeigt. Hier sieht er die einzelnen Schritte und wie er diese ausführen muss und kann diese auch als geschafft markieren.

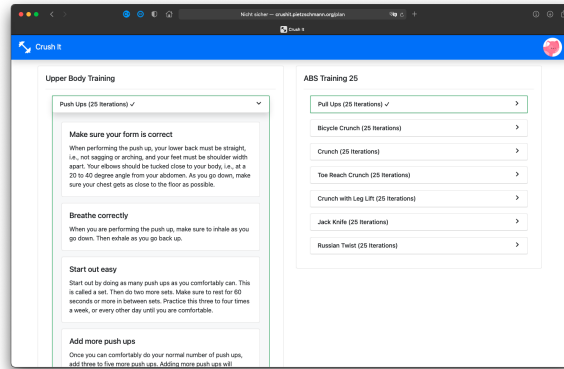


Figure 10: Plan-Bildschirm

Auf der Sports-Seite werden dem Nutzer verschiedene Kategorien geboten. Wählt er dann eine aus, werden ihm alle in der Kategorie enthaltenen Trainingspläne angezeigt. Dort sieht man neben dem Namen auch alle Einheiten und die Anzahl deren Wiederholungen. Nun hat der Nutzer die Möglichkeit einen Plan entweder seinem Profil, oder einer Gruppe in der er ist hinzuzufügen.

Die Groups-Ansicht zeigt neben allen Gruppen in denen der Spieler ist, auch die Möglichkeit einer Gruppe per Link beizutreten, oder eine neue zu erstellen. Möchte der Nutzer eine Gruppe erstellen, muss er einen Namen eingeben und kann ein Bild setzen. Auch hier darf das Bild wieder nicht größer als 1MB sein. Nach dem Erstellen wird der Nutzer automatisch zu der Gruppe hinzugefügt.

Lässt sich der Nutzer eine Gruppe anzeigen, kommt er auf folgende Ansicht:

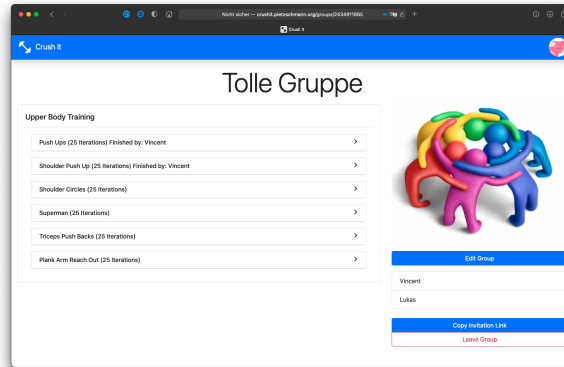


Figure 11: Gruppen-Bildschirm

Hier werden alle Informationen der Gruppe in zwei Spalten angezeigt. Auf der linken Seite findet der Nutzer alle in der Gruppe enthaltenen Pläne. Diese Ansicht verhält sich äquivalent zu der Plan Ansicht des Benutzers. Auf der rechten Seite befinden sich generelle Informationen wie das Gruppenbild und die Teilnehmerliste. Hier hat der Nutzer auch die Möglichkeit einen Einladungslink für andere Benutzer zu kopieren, das Gruppenbild zu editieren (auch hier gibt es wieder die 1MB Einschränkung) oder die Gruppe zu verlassen.

Die Shopping-Ansicht ermöglicht es über eine Suchleiste nach Equipment zu suchen.

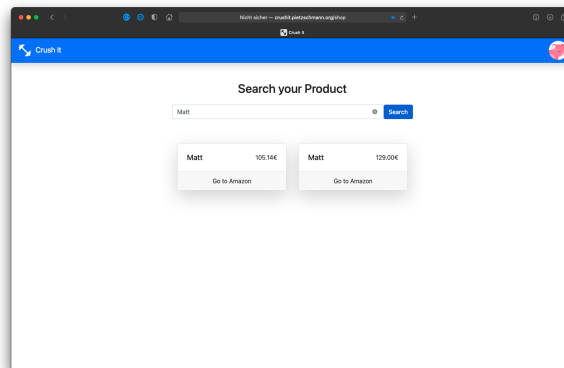


Figure 12: Such-Bildschirm

Werden Elemente zu dem eingegebenen Suchbegriff gefunden, kann der Nutzer sofort den Preis einsehen und bekommt die Möglichkeit direkt zu Amazon zu gehen und das Produkt zu kaufen.

7.2 Sicherheit (JS)

Passwörter werden im Frontend gehasht und dann zum Backend geschickt und in gehashter Form in der Datenbank gespeichert. Bei der Anmeldung wird auch ein Token generiert, welcher in der Datenbank mit dem Benutzer gespeichert und im Frontend als Cookie gespeichert wird. Dieses Token wird dann für alle weiteren Anfragen zur Authentifizierung verwendet, so muss nicht jedes mal das Passwort übertragen werden.

8 Entwurfsentscheidungen (JS)

In diesem Abschnitt werden wichtige Entwurfsentscheidungen erläutert.

8.1 Frontend

Ein wichtiger Aspekt des Projekts ist es, die entwickelten Dienstleistungen als eine Web App zur Verfügung zu stellen, die sowohl auf Desktop PCs als auch auf mobilen Endgeräten wie Smartphones gut nutzbar sein soll. Aufgrund der Aufgabenstellung wurden uns dafür einige Frameworks zur Auswahl gestellt. React wurde gewählt, da das Team damit bereits einige Erfahrung hatte und es sich deshalb für die Entwicklung anbot.

8.2 Backend

Die Entscheidung, welches Framework für das Backend genutzt wird, wurde über Umwege getroffen. Ursprünglich wurde C# in Betracht gezogen, um vorherige Erfahrungen nutzen zu können. Aber nach Rücksprache mit dem Product Owner wurde entschieden, Python Flask zu verwenden, um neue Fähigkeiten zu erwerben.

9 Qualitätsanforderungen

<Entfällt aufgrund der Aufgabestellung>

10 Risiken & tech. Schulden (JS)

10.1 Kein Zugriff auf Amazon API

Eine Anforderung an das Projekt ist es, dass man direkt aus der App auf einen Shop zugreifen kann, um für Übungen notwendige Geräte oder anderes Zubehör kaufen zu können. Dafür soll die Amazon API genutzt werden. Da zu Beginn des Projekts nicht bekannt ist, wie die Nutzungsbedingungen für diese API sind, besteht das Risiko, dass sie für dieses Projekt nicht nutzbar ist.

Eventualfallplanung: Sollte das Risiko eintreten, soll eine Fake-API erstellt und an eine kleine Datenbank angeschlossen werden, über die Platzhalterdaten an die App geschickt werden, die einen Shop simulieren.

Risikominderung: Die Verfügbarkeit und Nutzbarkeit der Amazon API sollte schnellstmöglich überprüft werden, um im Falle des Eintretens des Risikos genug Zeit für die Ergreifung der Gegenmaßnahme zu haben.

10.2 Unerwartete Veränderungen der Anforderungen

Es kann passieren, dass sich die Anforderungen des Product Owners an das Projekt unerwartet ändert beziehungsweise Anforderungen durch Probleme in der Kommunikation falsch verstanden wurden. Wenn dies erst kurz vor Projektende passiert beziehungsweise auffällt, könnte das erfüllen der Anforderungen mit großem Stress oder sogar Crunch verbunden sein.

Eventualfallplanung: Für dieses Projekt ist eine Verlängerung nicht möglich, daher ist die einzige Notlösung, einzelne Funktionen zu streichen, um einen Crunch und die damit verbundene mentale Belastung für die Mitarbeiter zu reduzieren.

Risikominderung: Das Risiko kann durch regelmäßige Sprint Reviews mit dem Product Owner reduziert werden.

11 Quellen

<Keine externen Quellen benutzt>