Diese Readme ist speziell darauf ausgelegt, dem Prüfer eine möglichst einfache Überprüfung der App zu ermöglichen. Es gibt auch 3 README Dateien innerhalb des Projekts. Diese sind eher an potenzielle Contributer gerichtet und zeigen bspw. warum das Projekt wie aufgebaut ist.

Formelles

Diese abgegebene Version dieser App liegt in GitHub im Branch „dhbw-release“. Dieser Branch beinhaltet Anpassungen, die extra für die Abgabe getätigt wurden, jedoch nicht in der produktiven Version der App genutzt werden sollen. Beispielsweise wird in der Abgabe Cordova zum Bauen der Android-App benutzt, wobei ich in der produktiven Umgebung Capacitor dafür nutze. Die eigentlichen Weiterentwicklungen der App findet man im master Branch.

Vorbereitung

Dieses Kapitel beschreibt die Schritte, die auf jeden Fall durchlaufen werden müssen. Sie sind wahrscheinlich offensichtlich, aber sicher ist sicher.

1. Öffne den Ordner lammes-assistant-ionic-app
2. npm install

Einfaches Testen

Diese Anleitung beschreibt den einfachsten Weg, die Applikation zu testen. Die Idee ist, dass die App mit dem Server auf der Produktivumgebung (auf DigitalOcean) kommuniziert und deshalb kein Server lokal deployed werden muss. Dies bedeutet jedoch auch, dass diese Methode nur funktioniert solange der produktive Server noch läuft und keine Breaking Changes beinhaltet.

Im Browser:

1. Führe innerhalb des Ordners „lammes-assistant-ionic-app“ den Befehl „npm run start:remote“ aus.
2. Die App öffnet sich automatisch im Browser.

Auf dem mobilen Endgerät:

1. Führe innerhalb des Ordners „lammes-assistant-ionic-app“ dem Befehl „ionic cordova run android“ aus.
2. Wenn ein Android Gerät gefunden wird, startet die App automatisch darauf.
3. Wenn nicht, wird in der Konsole der Pfad zur APK-Installationsdatei ausgegeben, die man dann auf einem Android Gerät installieren kann.

Lokales Testen

Diese Anleitung beschreibt einen etwas komplizierteren Weg, die App zu testen. Diese Variante hat die Vorteile, dass man die Kontrolle auf die gesamte Umgebung hat und, dass sie robuster ist und **unabhängig** von der Produktivumgebung **zuverlässig** funktioniert. Ich habe die Variante allerdings nur mit der Web-App, nicht jedoch mit der Android-App zum Laufen gebracht.

1. Stelle sicher, dass Docker installiert ist. Standartmäßig ist dann auch docker-compose installiert.
2. Stelle sicher, dass die Ports 4000, 5432 und 9000 frei sind. Meiner Einschätzung nach reicht es aus zu wissen, dass man keine Postgres Datenbank laufen hat. Erst wenn trotzdem Fehler auftreten, würde ich die Ports checken.
3. Öffne den übergeordneten Projektordner, in dem die Datei „docker-compose.yaml“ liegt.

4. Führe den Befehl „docker-compose up“ aus.
5. Die Konsole zeigt nun an, wie 3 Docker Container gestartet werden, die das Backend das App ausmachen. Dieser Start sollte problemlos verlaufen. Um sicherzugehen, dass alle benötigten Container laufen, kann in einer separaten Konsole der Befehl „docker container ls“ eingegeben werden. Dort sollten die 3 Container als laufend angezeigt werden. Man erkennt sie daran, dass ihre Namen den Prefix „lammes-assistant“ haben. Man sollte nicht den Image Namen mit dem Container Namen verwechseln, weil das zu Verwirrung führt.
6. Um im Browser zu testen, führt man im Ordner „lammes-assistant-ionic-app“ den Befehl „npm start:local“ aus.
7. Das Testen auf dem mobilem Endgerät ist etwas schwieriger, da sich das Backend und das Frontend auf unterschiedlichen Hosts befinden (PC und Android Gerät). Ich habe es nicht geschafft, die URL vom Backend so anzupassen, dass es funktioniert. Prinzipiell müsste man dieser Anleitung folgen und Variable *uriGraphQl* in „src/app/environments“ anpassen.
8. In der Konsole, in der die Container laufen, sollte man mit Strg+C oder Control+C die Container stoppen, wenn man fertig ist. Falls die Container-Prozesse detached sind, muss man sie manuell stoppen. Dazu mit „docker container ls“ alle laufenden Container anzeigen lassen und diese mit „docker stop <container-name>“ stoppen.
9. Wenn man Speicherplatz sparen möchte, kann man die Container, Images und Volumes löschen, die im Zuge dieses Prozesses erstellt wurden. Hierbei ist Vorsicht geboten, falls man wichtige Daten in der Docker-Umgebung hat. (https://docs.docker.com/config/pruning/)

Troubleshooting

Es ist unmöglich sich gegen alle Fehlersituationen abzusichern. Dieses Kapitel soll jedoch bekannte Fehler beschreiben:

- Bisher kam ich an einigen Stellen noch nicht dazu angemessenes Fehler-Handling einzubauen. Wenn bspw. die Signatur eines JWT-Tokens nicht vom Backend validiert werden kann, führt die App den Nutzer leider nicht zurück zur Login-Seite. Der Nutzer sitzt vor leeren Screens, weil er ohne Authentifizierung keine Daten laden kann. Dieser Fehler tritt dann auf, wenn man vom lokalen Testen zum remote Testen wechselt oder umgekehrt. Er lässt sich beheben, indem man alle Daten der App löscht. Auf Android deinstalliert man dazu die App. Im Chrome Browser gibt es die Möglichkeit alle App-Daten zu löschen (hier beschrieben).

Walkthrough

Die Features der App lassen sich am besten durch Ausprobieren entdecken. Es gibt jedoch einen Tipp für das „schnelle“ Erforschen der Übungsaufgaben-Funktionalität. Wenn man in der Übungsaufgaben Ansicht ist, kann man den „Ersteller-Filter“ so einstellen, dass andere User ausgewählt sind. Dann werden die Übungsaufgaben dieser User angezeigt und man kann diese direkt üben / ausprobieren.

Kontakt

Wenn Fehler auftreten kann man über das GitHub Projekt gerne auf Englisch Issues erstellen oder mich kontaktieren über ghostsgandi@gmail.com.