Web- & hybride Apps mit Ionic und Cordova

In dieser Übung geht es um Web-Applikationen für Smartphones. Sie werden zuerst mit Ionic eine „mobile-friendly“ Web-App erstellen, die Informationen zu Filmen darstellt. Sie werden dazu u.a. verschiedene Seiten inkl. Navigation und Datenübergabe implementieren, asynchrone HTTP-Anfragen verschicken und die erhalten JSON-Daten darstellen.

Zuletzt wickeln Sie ihre Web-Applikation in einen Cordova-Wrapper, so dass diese wie eine native Android-Applikation auf dem Emulator, bzw. einem Gerät läuft.

Hinweis 1: Wir empfehlen für diese Übung WebStorm (für Sie kostenfrei unter https://www.jetbrains.com/student/) als Entwicklungsumgebung und Chrome als Browser zu verwenden.

Hinweis 2: Diese Übung beinhaltet verschiedenste Technologien bzw. Frameworks wie HTML, JSON, TypeScript, Ionic, Angular, Cordova – lassen Sie sich davon nicht abschrecken! Tipps: Konsultieren Sie die Folien (speziell die "How-To"-Hinweise), studieren Sie das Projekt-Template von Ionic (da sind sehr viele Dinge schon drin!), googlen Sie oder stellen Sie gerne auch Fragen im Forum. - Diese Übung läuft unter dem Motto "embrace the challenge!" ☺

1. Ionic-Projekt "MovieApp" aufsetzen

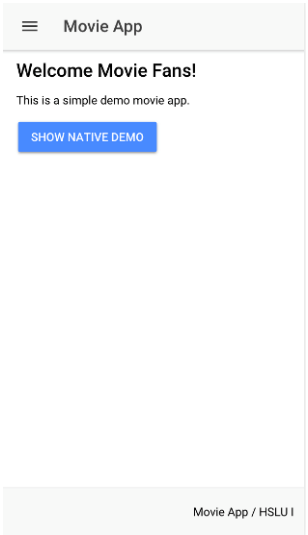
Installieren Sie wie in den Folien angegeben mit Hilfe vom Node Package Manager npm die beiden Frameworks Ionic und Cordova. Wenn Sie npm installiert haben (siehe https://nodejs.org/en/), geht diese mit folgendem Kommando:

```
$ npm install -g ionic cordova
```

Erstellen Sie dann ebenfalls in der Konsole ein neues Ionic-Projekt "MovieApp" vom Typ "sidemenu" mit dem folgenden Kommando (Sagen Sie danach Ja zu Cordova und Nein zum "Ionic Pro SDK"):

```
$ ionic start MovieApp sidemenu
```

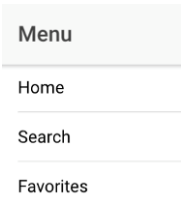
Damit wurde Ihnen ein Ionic-Projekt mit einem Side-Menu und zwei Seiten "Home" sowie "List" erstellt. Starten Sie diese App in der Konsole und testen Sie die MovieApp in Ihrem Browser. Öffnen Sie das Projekt danach in WebStrom und arbeiten Sie danach primär in dieser IDE.



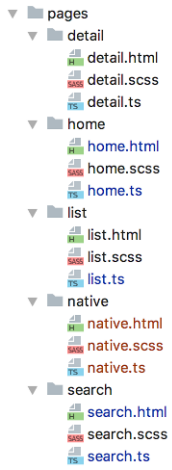
2. Start-Seite, Skelett für weitere Seiten & Sidemenu

Ergänzen Sie die Start-Seite der App (HomePage) um einen Footer und passen Sie die Texte an, so dass diese Seite wie im Screenshot rechts aussieht.
Legen Sie danach neben den unter `src/pages/` vorhandenen zwei Verzeichnissen für die zwei Seiten Home und List analog zwei weitere Seiten für Suche, Film-Details (und ggf. Native) an. Registrieren Sie diese Seiten in `src/app/app.module.ts` analog zu den vorhandenen Seiten `HomePage` und `ListPage` in `declarations:` und `entryComponents:`. Hinweis: Die App wird am Schluss aus vier Seiten `HomePage`, `ListPage`,

`SearchPage`, und `DetailPage` bestehen, optional zusätzlich `NativePage` (siehe die letzte Teilaufgabe unten dazu).



Das siehst dann in WebStorm aus wie auf dem Screenshot rechts.



Ergänzen Sie in `src/app/app.component.ts` das Property `pages` weiter um die Such-Seite, so dass das Sidemenu danach wie im Screenshot links aussieht.

Damit ist die Grundstruktur für die MovieApp bereit und Sie können sich der App-Logik zuwenden

3. Film-Suche

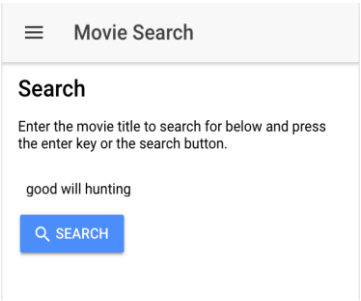
Mit dieser App soll nach Filmen gesucht werden können. Wir verwenden dazu den kostenlosen Dienst von http://www.omdbapi.com. Bestellen Sie sich dazu zuerst unter der folgenden Adresse einen kostenlosen API-KEY:

`http://www.omdbapi.com/apikey.aspx?__EVENTTARGET=freeAcct`

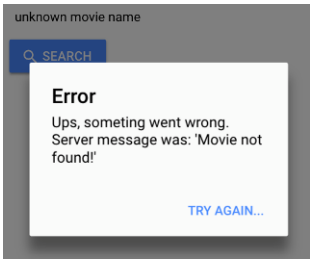
Unter der folgenden URL können Filme abgefragt werden (wobei Sie API-KEY und FILM-TITLE entsprechend ersetzen) und Sie bekommen eine JSON-Antwort:

`http://www.omdbapi.com/?apikey=API-KEY&plot=short&r=json&t=FILM-TITLE`

Passen Sie dazu die Such-Seite wie auf dem Screenshot rechts an. Insbesondere soll die Seite also ein Element `ion-input` für die Eingabe und ein Element `button` zum Auslösen der Suche haben. Implementieren Sie die entsprechende Logik, so dass auf Knopfdruck eine entsprechende Anfrage an die URL oben abgesetzt wird. Gehen Sie dabei wie auf den Folien gezeigt vor. Erstellen Sie dazu also u.a. unter `src/interfaces/Movie.ts` folgendes Interface zum typisierten Zugriff auf die erhaltenen JSON-Daten:




```
export interface Movie {
  Response: string;
  Title: string;
  Plot: string;
  Poster: string;
  Error: string;
  Country: string;
  Year: string;
  Director: string;
}
```

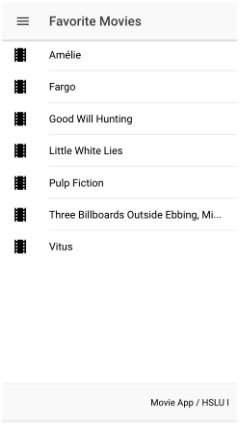


Falls die Suche erfolgreich war (`Response == 'True'`), sollen die Film-Daten der Detail-Seite übergeben werden und diese angezeigt werden (`NavController.push`), siehe nächste Teilaufgabe. Falls die Suche erfolglos war, soll ein AlertController inkl. Server-Fehlermeldung (Property `Movie.Error`) wie im Screenshot oberhalb von diesem Abschnitt rechts angezeigt werden. In der Doku zum AlertControler hat's u.a. 1:1 verwendbare Code-Beispiele, siehe https://ionicframework.com/docs/api/components/alert/AlertController/.

4. Film-Details

Stellen Sie Infos zum gefundenen Film auf der Detail-Seite dar. Verwenden Sie also die Infos zu Titel, Regisseur, Land, Jahr, Inhalt und den Link auf das Poster. Passen

Sie die Detail-Seite entsprechend an, so dass diese in etwa wie der Screenshot auf der ersten Seite von dieser Aufgabenstellung rechts oben aussieht.

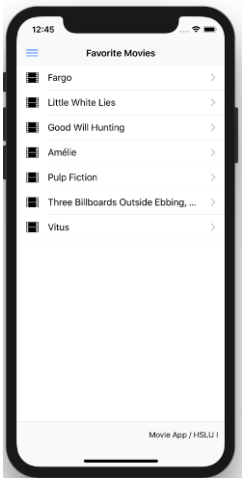


5. Favoriten-Liste

Hinterlegen Sie in der App mindestens N >= 5 Ihrer Lieblingsfilme. Legen Sie dazu unter `src/assets/movies` von diesen Filmen jeweils in JSON-Dateien `X.json` (für X von 1..N) die entsprechend Film-Daten vom Server ab. Bauen Sie dazu die vom Ionic-Template verhandene List-Seite entsprechend um, so dass sie wie im Screenshot rechts aussieht. Verwenden Sie dazu weiterhin den in `src/pages/list/list.html` vorhandenen Mechanismus mit `*ngFor`, die entsprechende Zeile soll also so aussehen:

```
<button ion-item *ngFor="let movie of movies" (click)="movieSelected(movie)">
```

Passen Sie dazu `src/pages/list/list.ts` ebenfalls entsprechend an, so dass dort die N JSON-Datein eingelesen werden und als Property `movies` zur Verfügung gestellt werden. Verwenden Sie zum Einlesen dieser "App-lokalen" JSON-Dateien `this.httpClient.get(...)` mit einem relativen Pfad der Form `'../assets/movies/1.json'` (statt einem `'http://...'` wie oben). Wird ein Film angewählt, soll die bereits erstellte Detail-Seite mit den entsprechenden Film-Infos angezeigt werden. Hinweis: Die Poster-Bilder können Sie der Einfachheit halber wie gehabt von der angegeben URL laden. Die enstprechende Bild-Datein Ihrer Lieblingsfilme müssen also nicht in der App hinterlegt werden.



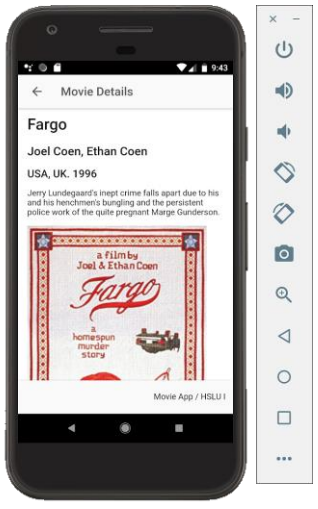
6. Native App mit Cordova

Lassen Sie Ihre App nun mittels Cordova als native Android- bzw. iOS-App laufen. Dazu wird i.A. ein installiertes AndroidStudio bzw. Xcode benötigt. Dies geht z.B. für den Android Emulator mit dem folgenden Kommando:

```
$ ionic cordova run android --emulator
```

Allenfalls müssen Sie dazu zuerst einen Android-Emulator starten, z.B. aus AndroidStudio.

Gratulation, Sie haben damit soeben eine hybride mobile WebApp mit Ionic und Cordova erstellt und deployed! ☺



7. [OPTIONAL] Native APIs benutzten mit Cordova

Verwenden Sie die beiden Cordova Plugins Battery-Status und Network, um vom unterliegenden mobilen Betriebssystem entsprechende Informationen abzufragen und darzustellen, siehe Screenshots rechts. Doku inkl. Code-Beispiele:
https://ionicframework.com/docs/native/battery-status/
https://ionicframework.com/docs/native/network/
Schauen Sie sich die Liste der in Ionic verfügbaren Cordova-Plugin an unter https://ionicframework.com/docs/native/ und experimentieren Sie ggf. gerne mit weiteren Cordova Plugins herum.