Writeup for cross Development ionic project

Introduction

Like the previous project in Qt, I made a movie rater/tracker app, but this time it's developed in ionic. It will still contain the same features as the previous one, except file I/O. It also includes one additional feature, when a movie is added, a short toast message will appear.

Project requirements

The app had to meet the following requirements:

- standard ionic components
- at least one custom component
- at least one capacitor plugin
- minimum of 2 pages
- data transfer between pages using a service
- must run on a real device or emulator

I made use of ionic's standard components as well as a selfmade custom star component to display the ratings, this star component was made using normal HTML elements and ion-icons. As capacitor plugin I used Toast to show a short message when adding a movie to the list, for the web version I mocked this using a div that shows up for a couple of seconds. Page 1 is the main page where you can see your movie list and add new movies using a modal, page 2 is a settings page where you can change the color of the stars. Data transfer between the pages is done using a service called star-color, this stores the selected color and gives it to page 1 where the star component uses it to set the color of the stars, the color of the stars in the add movie modal also use this service to change its color accordingly.

after some tweaks to my pc and adding android to the project, the app finally managed to run on my virtual android device using ionic capacitor.

This means that all app requirements were met.

Setting up the development environment

To set up the development environment for ionic, I first had to install node.js and npm, luckily these were already installed from the course Web Development where we learn angular.

Next I installed the ionic CLI globally using the following command:

```
npm install -g @ionic/cli
```

I also installed the recommended vs Code extension: WebNative.
Angular Language Service was already installed.

To create my project I used the WebNative extension, this comes with an interface to create projects easily. I chose for an ionic angular project with target web and the template tabs.

Now that I have my project I can start developping.

Development of the app

I started by making my main page, here I made the movielist part and modal to add movies. First the app looked very crude, but thanks to ionic's standard components I was able to quickly make it look much better. After both my movielist and modal were done, I removed the dummy data (used to test the movielist) and linked the movielist and modal together, this is also where I implemented the capacitor plugin Toast. Then I started on my custom star component to use inside the movielist. And lastly I used tab 2 to create a settings page where you can change the color of the stars, this color is then communicated to page 1 using a service.

Running ionic on a real device or emulator

To run the app on an emulator, I first needed an emulator. From the course App Dev Advanced I already had android studio installed with a virtual device set up.

Now I needed to install capacitor android in my project, to enable capacitor for this project, I used the WebNative extension, by just clicking the add android project button capacitor was enabled.

In order for the emulator to work I needed to add a local.properties file and set the sdk.dir to the location of my android sdk, which I found in android studio settings.

I also had to set the JAVA_HOME environment variable to the location of jbr in android studio, this is needed for gradle to work properly.

After this I was able to run the app on the android emulator using the WebNative extension.

Installing capacitor plugins

Installing the capacitor plugin was one of the easiest parts of this project, using npm I just installed the toast plugin:

```
npm install @capacitor/toast
npx cap sync
```

https://capacitorjs.com/docs/apis/toast

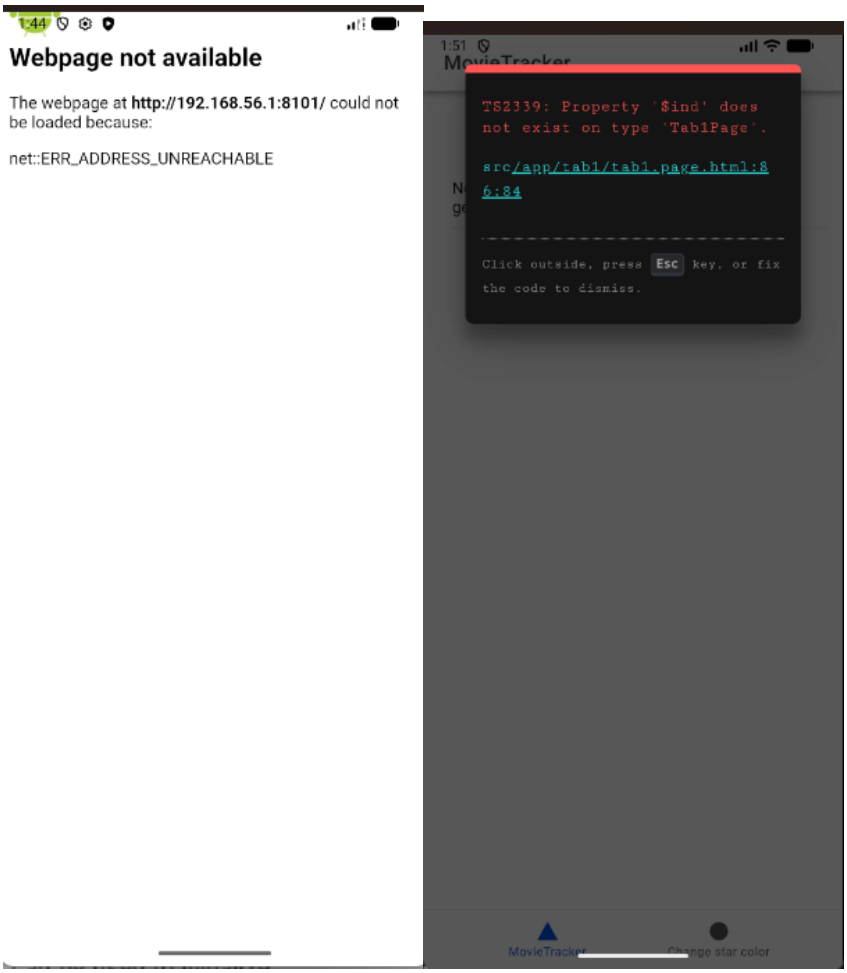
After installing the plugin I imported it on the right page and put the simple toast code to show a toast message when adding a movie.

I noticed that these capacitor plugins don't work on web, so I made a simple div that shows up for a couple of seconds to mock the toast message on web.

Link to the theory

In the theory we talked about 3 types of custom components, default HTML, canvas and svg. I made use of default HTML elements and ion-icons to create my custom star component. This component has an input, the rating, and displays the corresponding number of stars.

ionic is a hybrid framework meaning that it runs its applications on android in webviews, this can sometimes be seen when certain errors occur or when internet connection is slow/no connection.



Capacitor needs to be installed in order for ionic to run on real devices or emulators, it acts as a bridge between the web code and the native code of the device. Capacitor plugins are used to access native device features such as camera and toast messages. In my project I used the toast plugin to show a short message when adding a movie.

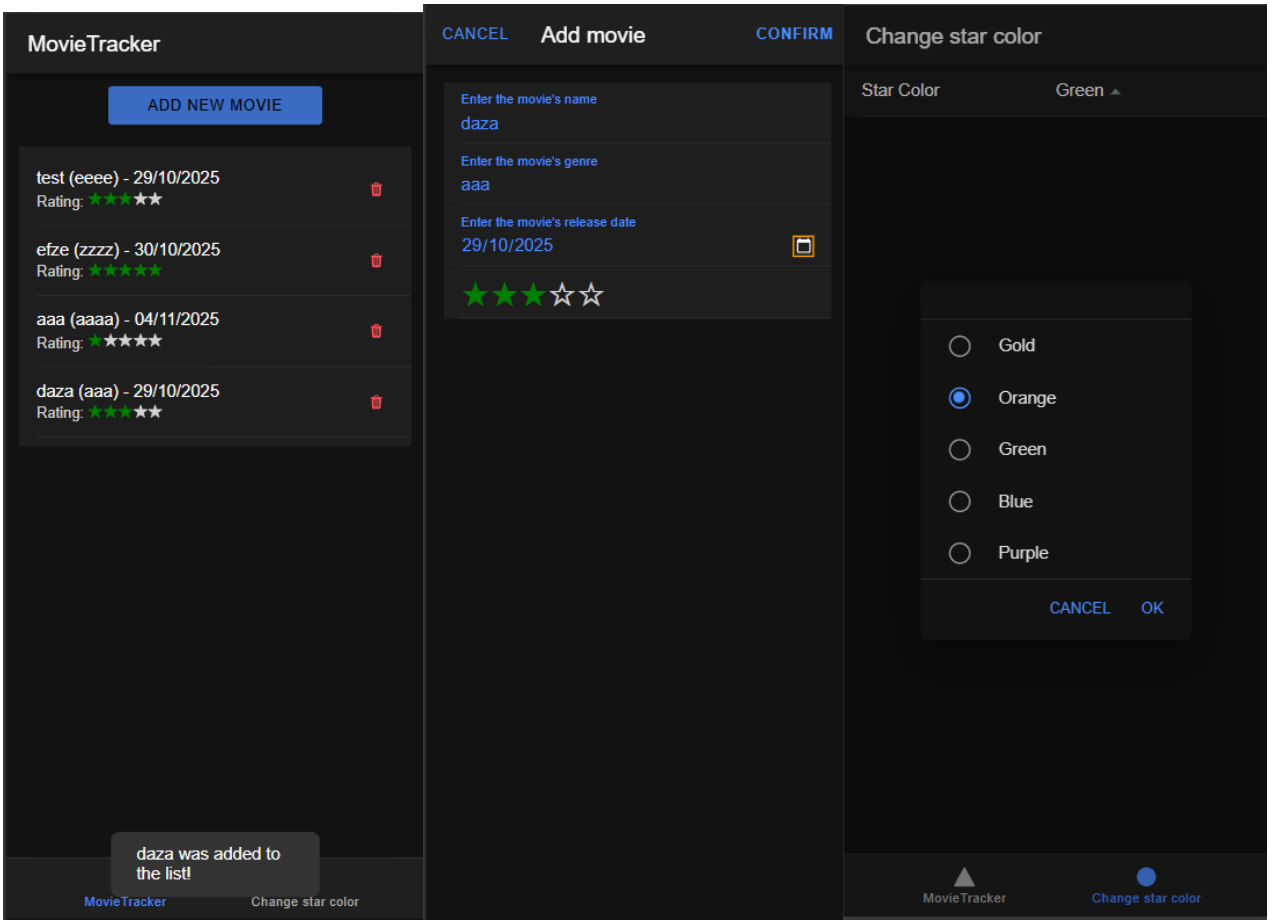
changes to the concept of the app as a result of the used technology

No changes were made to the concept of the app, I just adjusted/added some features to fit the requirements of the project. Adding the toast message, use of a modal for adding movies.

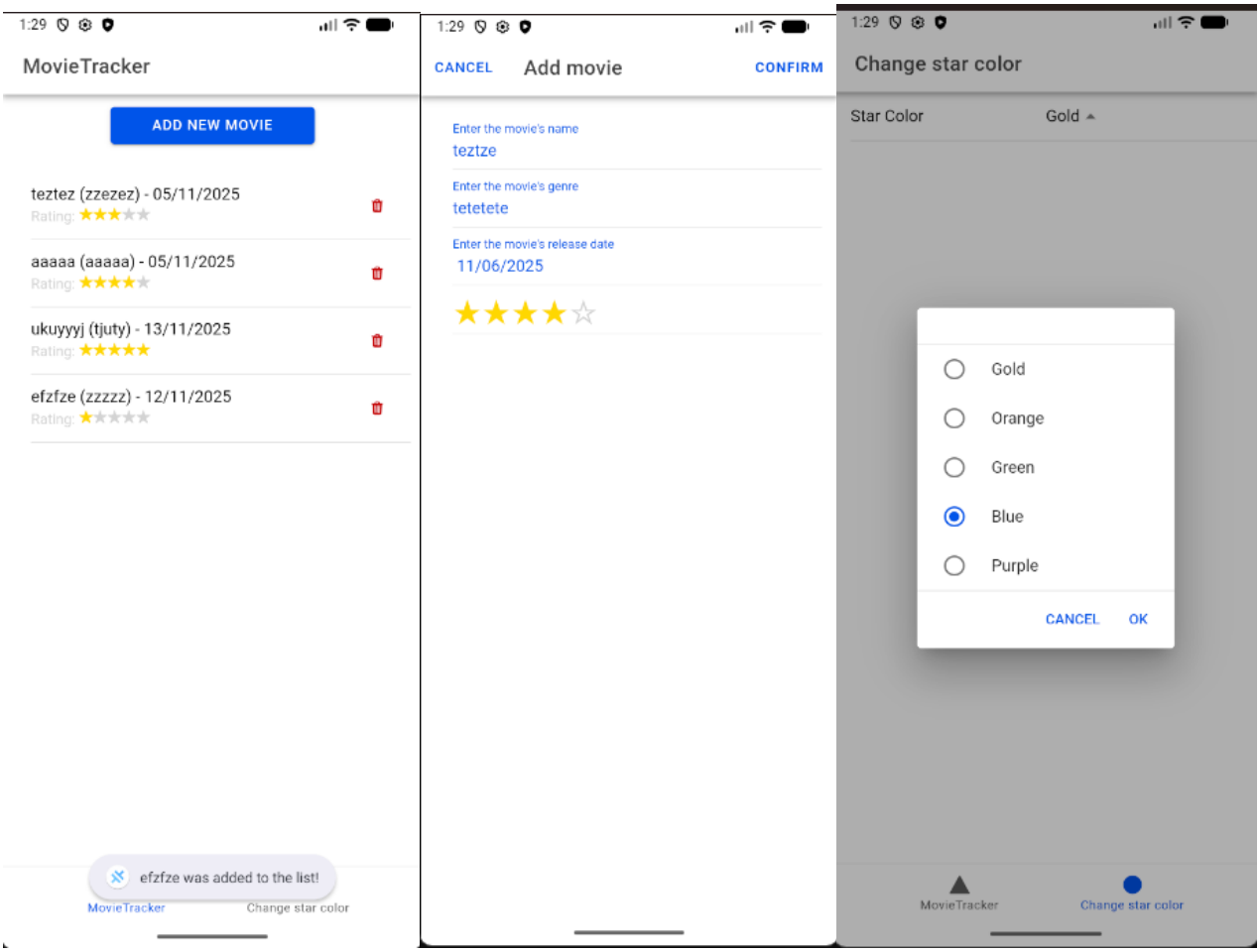
The end result

My app became a fully functional movie rater/tracker that uses ionic to run both on web and android.

Running on web in dark mode:



Running on android emulator:



Conclusion

The ionic framework is great for quickly and easily developing cross-platform applications. The use of standard components and capacitor plugins makes it easy to create a functional and visually appealing app. The development process was fast and smooth, and I was able to meet all the project requirements without any major issues. Overall, I am satisfied with the final result and the experience of using ionic for cross-development.

Comparison with project in Qt

The ionic version of the app is more visually appealing and has a more modern look compared to the Qt version. The use of standard ionic components makes it easy to create a consistent and user-friendly interface. The toast message when adding a movie is such a small thing but adds a lot of value in my opinion. Developping in ionic was also much faster and easier compared to Qt. However, the Qt version has better performance and more advanced features because it's a native application compared to ionic's hybrid approach.