CASE STUDY
MOVIE APP

WAS

Eine Webanwendung, die Informationen über Filme, Genres und Regisseure bereitstellt. User können einen Account erstellen, in einer Filmdatenbank suchen, eine Liste mit Favoriten anlegen und persönliche Informationen ändern.
Server- und clientseitiger Teil der Anwendung wurden mithilfe des MERN-Stacks erstellt.

Link zur live App

Code Server-Side

WARUM

Das Projekt habe ich im Rahmen meiner Ausbildung “Full Stack Web Development” bei CareerFoundry erstellt. Dabei mussten genaue Zielvorgaben eingehalten werden.

Code Client-Side

WIE

Das Projekt wurde umgesetzt mit modernen, leistungsfähigen Frameworks. Dadurch konnten die vielfältigen Aufgaben von der Erstellung eines Webservers mit Datenbankzugriff über die sichere Userauthentifizierung bis hin zu responsiven, zeitgemäßen User Interfaces auf effiziente Weise implementiert werden.

Bild: midjourney

Serverseitiger Teil

Zu Beginn musste ich mich mit den Grundlagen von Node.js vertraut machen, mit Node Version Management und dem Node Package Manager. Mit Express wurden der Webserver und die Endpunkte der REST API realisiert. Die Anwendung greift auf eine MongoDB Datenbank zu. Für das Testen der Endpunkte kam Postman zum Einsatz.

Eine Herausforderung war es, die zugrundeliegenden Mechanismen der User-Authentifizierung mit Passport zu verstehen und z.B. die Tokens richtig zu konfigurieren. Weitere Maßnahmen in Bezug auf Sicherheit waren Hashing des Userpassworts, serverseitige Input-Validierung sowie die CORS-Konfigurierung des Servers.

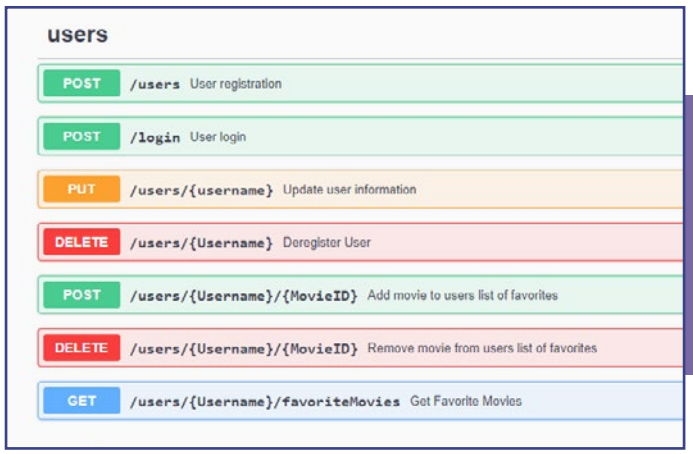
```
  async (req, res) => {
  });

// Add a movie to a user's list of favorites
app.post('/users/:Username/:MovieID', passport.aut

  // check if username matches the user saved in t
  if (req.user.username !== req.params.Username) {
    return res.status(400).send('Permission denied
  }

  await Users.findOneAndUpdate({ username: req.par
    {
      $addToSet: { favoriteMovies: req.params.Movi
    },
    { new: true })
    .then((updatedUser) => {
      res.status(200).json(updatedUser);
    })
    .catch((err) => {
      console.error(err);
      res.status(500).send('Error: ' + err);
    });
});

// Delete a movie from a user's list of favorites
app.delete('/users/:Username/:MovieID', passport.a
```



Die Dokumentation der Endpunkte mit Swagger war zeitaufwendig.
Bei der Implementierung der clientseitigen Anwendung konnte ich jedoch selbst sehr davon profitieren.

Code Server-Side

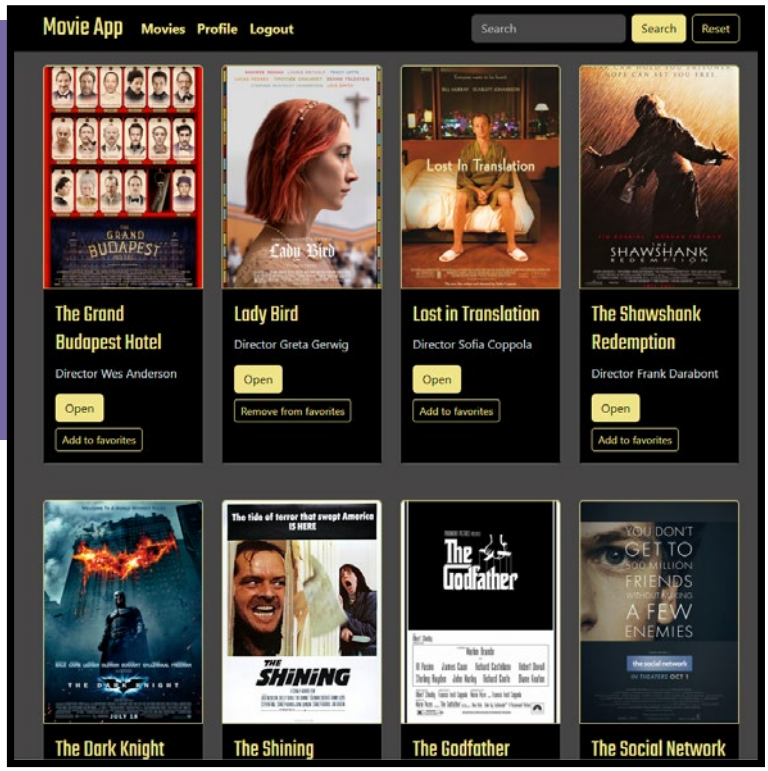
API Endpunkt Dokumentation mit Swagger

Clientseitiger Teil

Auch hier musste ich mich im ersten Schritt mit der Entwicklungsumgebung und mit den Grundprinzipien von React vertraut machen. Die von React verwendeten .jsx Dateien werden vor dem Release in einem Build-Prozess in JavaScript übersetzt.
Die Herangehensweise beim Entwickeln mit React erforderte etwas Eingewöhnung, erschien mir aber schon bald logisch und schlüssig.

```
return (
  <BrowserRouter>
    <NavigationBar
      user={user}
      onLoggedOut={() => {
        setUser(null); setToken(null); localStor
      }}
      onSearch={onSearch}
    />
    <Row className="justify-content-md-center">
      <Routes>
        <Route
          path="/signup"
          element={
            <>
              {user ? (
                <Navigate to="/" />
              ) : (
                <Col lg={5} md={8} sm={10}>
                  <SignupView />
                </Col>
              )}
            </>
          }
        />
        <Route
```

Im Rahmen des Projekts wurden Ajax - Abfragen an die serverseitige API realisiert, die Auswertung der Responses inklusive Fehlerbehandlung, User-Authentifizierung und clientseitiges Routing. Das responsive Layout wurde mit Hilfe von React Bootstrap erstellt.



Link zur App

Code Client-Side

Fazit

Über das ‘klassische’ Programmieren mit JavaScript hinaus habe ich mich in diesem Projekt mit neuen Konzepten und Herangehensweisen vertraut gemacht und Erfahrungen mit den Frameworks Express und React gesammelt.

Immer wieder war ich beeindruckt, wie mächtig diese Bibliotheken sind und wie elegant damit Probleme gelöst werden. Kurze Befehle reichen aus, um komplexe Aufgaben zu bewältigen, moderne User Interfaces lassen sich mit wenigen Seiten Code realisieren. Dabei steht immer auch die gute Strukturierung und Lesbarkeit des Codes im Vordergrund.