Adminpanel für ChronicleDB

Mina Gaid, Walid El Kassem
Wintersemster 2021

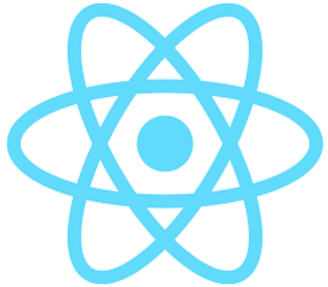
Zusammenfassung

In diesem Dokument wird über die Erstellung eines Adminpanel mit React für ChronicleDB berichtet.

Einleitung

Dieser Bericht erfolgte im Rahmen des fortgeschrittenenpraktikums „**Adminpanel für ChronicleDB**“ mit Amir El-shaikh , Arbeitsgruppe Datenbanksysteme , und Prof. Dr. Bernhard Seeger an der Philipps-Universität Marburg. In dem Praktikum wurden Eine FrontEnd Adminpanel in mehrere Frontend webframeworks entwickelt: React, Angular und Vue. In diesem Bericht geht es um **React**, das Framework was wir genommen haben.

Warum React ?



React ist eine JavaScript-Bibliothek zur Erstellung moderner Benutzeroberflächen (UI's) für Webanwendungen
Das Github-Repo von React hat momentan **110.000** Sternen, Ausserdem zeigen die riesige Beliebtheit und der wachsende Marktanteil, dass es sich React gut etabliert hat.

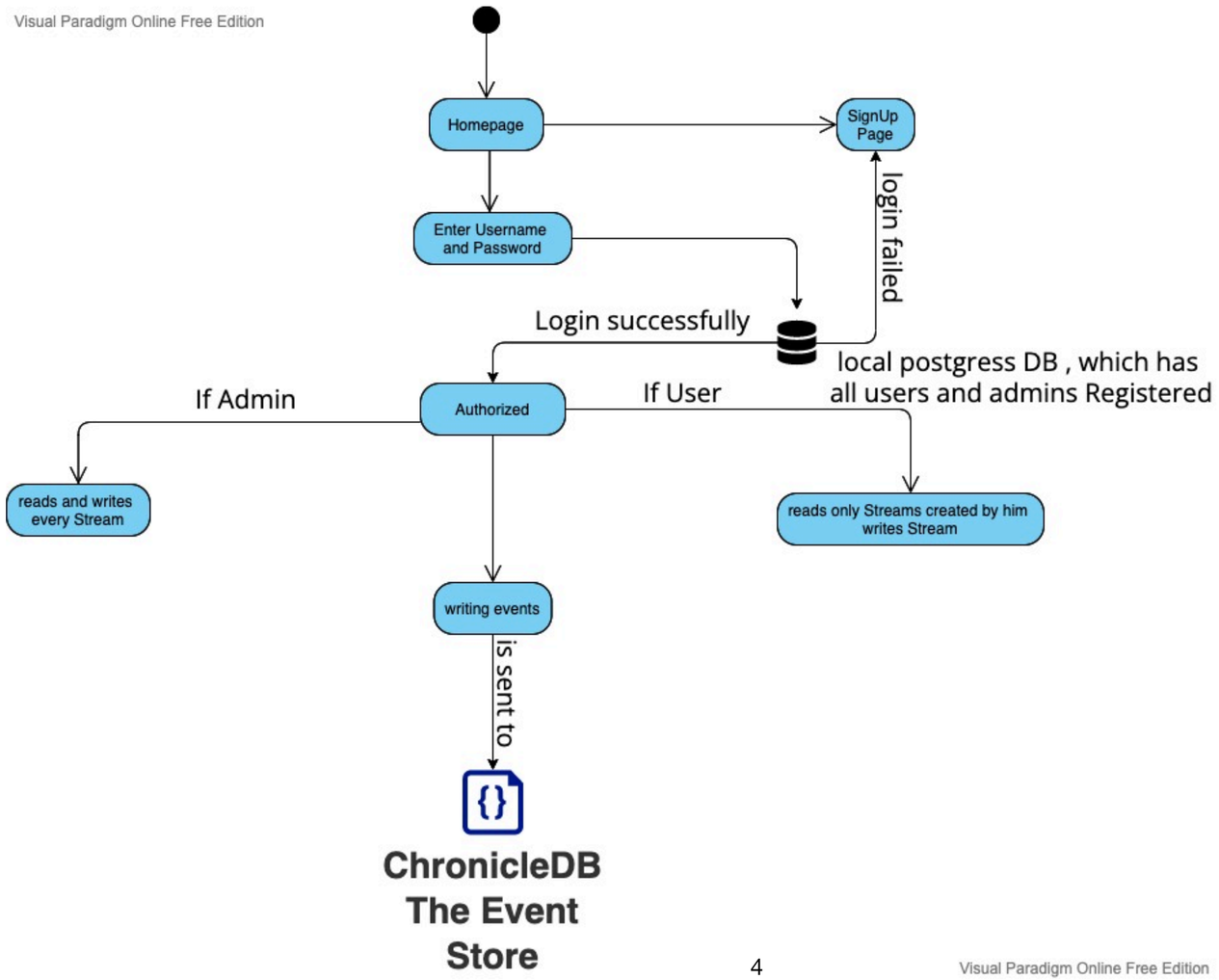
- **Einfachheit**
  ReactJS ist einfach zu verstehen. Der komponentenbasierte Ansatz, der klar definierte Lebenszyklus und die Verwendung von einfachem JavaScript machen es sehr einfach, React zu lernen, ein professionelles Web (und mobile Anwendungen) zu erstellen und es zu unterstützen. React verwendet eine spezielle Syntax namens JSX, mit der man HTML mit JavaScript mischen kann. Dies ist aber keine Voraussetzung , Entwickler können immer noch in einfachem JavaScript schreiben, aber JSX ist viel einfacher zu verwenden.

- **Performance und Leistung**
  React bietet kein Konzept eines eingebauten Containers für Abhängigkeiten(Dependencies) an. wir können Browserify, Require JS, EcmaScript 6-Module verwenden, die wir über Babel, ReactJS-di verwenden können, um Abhängigkeiten automatisch einzufügen.

- **Native Approach**
  React kann verwendet werden, um mobile Anwendungen zu erstellen (React Native). Und React ist ein eingefleischter Fan der Wiederverwendbarkeit, was bedeutet, dass eine umfassende Wiederverwendbarkeit von Code unterstützt wird. So können wir gleichzeitig IOS-, Android- und Webanwendungen erstellen.

Wie Sieht unser Projekt aus ?

High level Architektur



Technologien

Eingesetzte Technologien und Werkzeuge

- **NodeJS** für Das Backend.
- **postgresSQL** für Die Speicherung der Nutzerdaten (Login und Registering).
- **ReactJS** für als webframework.
- **Visualstudiocode** als Editor.
- **Github** als Versioncontroll.
- **Postman** für Verwendung und Bauen der API’s.
- **Google Chrome** für das Spielen der Frontend.

Benötigte Abhängigkeiten

Die Bibliotheken , die wir für das Projekt verwendet haben.

- MUI Material (https://mui.com/components/material-icons/)
- React-router-dom (https://v5.reactrouter.com/web/guides/quick-start)
- Axios (https://www.npmjs.com/package/axios)
- React-Cookie (https://www.npmjs.com/package/react-cookie)
- React Toastify (https://www.npmjs.com/package/react-toastify)
- React Bootstrap (https://react-bootstrap.github.io/components/dropdowns/)

Coding Style

- Wir haben **StandardJS** verwendet , als Styling Kontrolle
  Aus dem Grund , dass es mit React sehr kompatibel ist.

Warum eine PostgressDB für Authentikation und Authentifizierung ?

Unser erster Approach war FireBase zu verwenden , da wir damit schon Erfahrung gesammelt haben.
da weder ChronicleDB noch unsere Webapp irgendwo deployed sind , haben wir uns auch für eine Lokale Lösung für das Speichern der Users entschieden
Ausserdem haben alle andere Teams sich für eine Lokale Datenbank entschieden, dann hätten wir etwa Unterstützung bei dem Thema ,falls wir an Irgendeiner Stelle nicht weiterkommen könnten.
Der Ansatz ist recht Einfach :
Eine Connection wird aufgebaut , Nutzerdaten in eine Tabelle reingeschrieben
Passwörter selbstverständlich gehasht gespeichert.

Installationsanleitung:

- ChronicleDB Instanz laufen lassen
- Das Githubrepo von uns(Admin-Panel-ChronicleDB) Clonen und **npm i** ausführen ,damit alle Bibliotheken heruntergeladen werden.
- Danach **npm run start** ausführen.
- Das **BackendRepo**(Authentication BackEnd) Clonen (Amir hat Zugriff drauf) ,auch danach ist ein **npm i** , dann **npm run start** erforderlich.
- Das **Schema.sql**(für die Erzeugung der Datenbank) dann **init.sql**(für die Erstellung der Usertabelle) in Pgadmin ausführen (In Backendrepo zu finden).

Erfahrungsbericht

Entscheidungen , die wir gemacht haben und die Bewertung davon
und Tipps für zukünftige Entwickler

- Lange Zeit genommen um ein Framework zu wählen . Das war eine schlechte Entscheidung und total kontraproduktiv und es spielt keine riesige Rolle ,welche Framework man verwendet , besonders bei einem Mittelgrossen Projekt (wie unseres) . Damit man wirklich die Unterschiede zwischen den Frameworks sieht , muss man das gleiche Projekt , 3 Mal zu entwicklen ,mit den 3 grossen Frameworks.
  besser wäre es : make it work , make it right , then make it fast !

- Wir hatten ein paar Probleme , da wir zwei verschiedene Maschinen haben bzw. Verschiedene Umgebungen . Probleme wie Status der Datenbank , also wenn ich einen User addiere , ist der User nicht in der Tabelle auf Walid’s Rechner .
  so besser wäre wenn alle Arbeitskomponeten irgendwo **deployed** sind , dann wäre es viel besser bei der Entwicklung und schafft Einheit .

Vorgehen

Entwicklungsprozess

Da das Team nur aus 2 Personen besteht , haben wir nicht so komplizierte Prozesse gehabt.Ausserdem wohnen wir im selben Haus , so haben wir fast jede Woche getroffen und alles durchgegangen und pairprogrammiert .

Rollenvergabe war rotierend.

Eingesetzte Programmiersprachen war eigentlich nur Javascript ,Serverside(für das Backend Authentication) war es nodejs und für das Frontend war es React.js

Als Editor haben wir beiden Visuallstudio Code verwendet.

Projektmanagementwerkzeuge waren in unserem Fall nicht nötig.

Zeitplan

- 1 mal der Woche war unser Treffen mit Amir
- Wir haben uns je nach der Tasks ,die wir haben , getroffen . Meistens auch 1 mal der Woche .

  in den Protokollen sind unsere Meetings in Detail beschrieben.

Team

Vorstellen der Teammitgliedern

1. Walid El-kassem
   Entwickeln , Dokumentieren und Testen

2. Mina Gaid
   Entwickeln , Dokumentieren und Testen

Testen

Unit Tests waren weniger geeignet , daher sind wir auf Systemtests übergegangen .

mit der Abgabe finden Sie ein Folder mit den Screenshots von allen Möglichen Szenarien bei der App.

Beispielanwendung für das Adminpanel Step By Step:

In Repo(also in der Abgabe) gibt es ein Folder mit den Screenshots von allen möglichen szenarien .

Zusammenfassung

- In der Zeit von the Internet of Things (IoT) ,braucht man das schnelle und fluktuierende Schreiben von Ereignissen (Events), und da eignet sich **ChronicleDB** sehr , weil **ChronicleDB** eine hervorragende write und query Leistung hat.

- Im Allgemein hat sich **Javascript** in den letzten Jahren sehr verstärkt , und dominiert momentan die **Web**. Ausserdem ist JS auch auf der Serverseite einsetzbar.

- **ReactJS** ist auch super Stark , wird auch ständig von **FACEBOOK** gepusht und erweitert. **React** kann auch auf den Server mithilfe von **NodeJS** gerendert. Sogar für die Mobile-Apps kann man heutzutage React als **Reactnative** verwenden.Momentan ist es auch in den **Jobmarkt sehr gefragt** , daher ist React ein Tolles Werkzeug und wir haben die Zeit genossen ,uns React zu aneignen.