Fachhochschule Nordwestschweiz
Hochschule für Technik

Google Docs Light

Workshop Web - FS22

21. Mai 2022

Studenten	P. Schmucki, J. Villing, K. Zellweger
Dozenten	D. König, S. Meichtry, J. Luthiger
Studiengang	Informatik
Hochschule	Hochschule für Technik

Inhaltsverzeichnis

1 Summary

2 Systemübersicht

2.1 Services

2.2 Sequenz

2.3 Applikationsprotokoll

2.4 Benutzerverwaltung

Es ist keine persistente Benutzerverwaltung mit Registrationsprozess implementiert. Nach erstmaligem Anmelden in der Applikation mit einem globalen Benutzer, wird ein zufälliger Author erstellt. Die Daten des Authors werden im Local Storage des Browsers gespeichert, sodass bei erneutem Öffnen der Applikation der gleiche Author wiederverwendet wird.

3 Frontend

Das Fronted der TeamDocument Applikation ist als React SPA entwickelt. Einmal angemeldet kann ein Benutzer an der kollaborativen Bearbeitung des Dokumentes teilnehmen.

Folgende Interaktionen sind möglich:

- Ändern des eigenen Namens
- Hinzufügen eines neuen Paragrafen
- Bearbeitung bestehender Paragrafen
- Sperren des Paragrafen an dem gerade gearbeitet wird (implizit)
- Verschieben von Paragrafen innerhalb des Dokuments
- Löschen eines bestehenden Paragrafen
- Wiederherstellen des zuletzt gelöschten Paragrafen (Hidden Feature)

Des Weiteren werden folgende Informationen auf dem UI dargestellt:

- Name des ursprünglichen Authors eines Paragrafen
- Name des Authors welcher aktiv einen Paragrafen bearbeitet.
- Highlight des eigenen aktuellen Paragrafen
- Liste mit allen Dokumentupdates in chronologischer Reihenfolge
- Avatare aller aktiven Benutzer

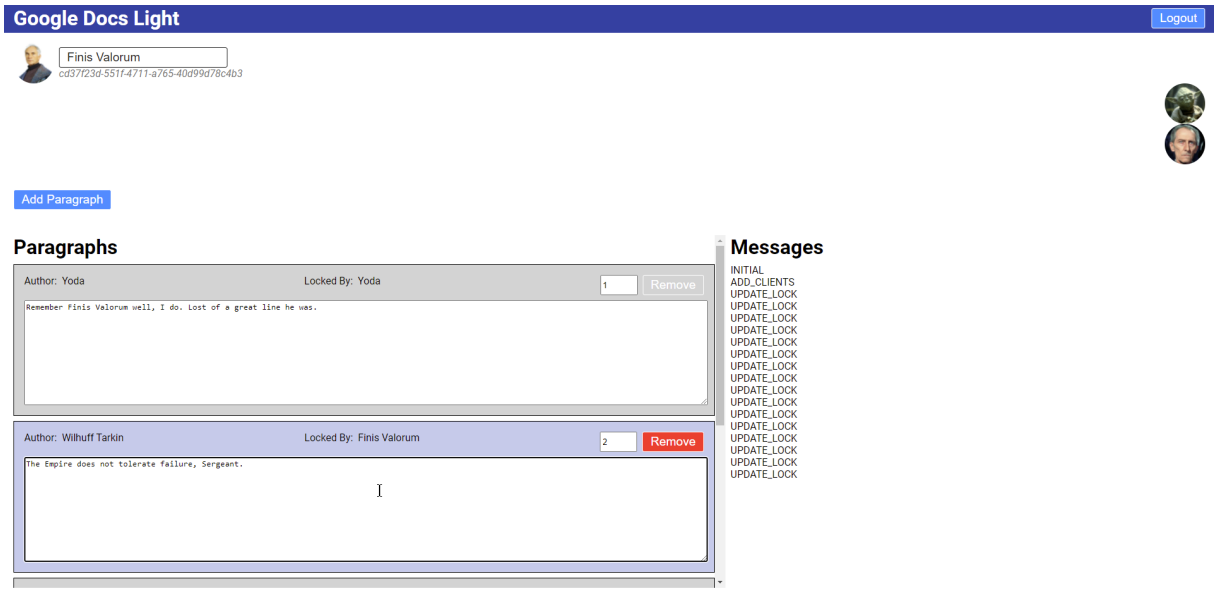


Abbildung 3.1: Team Document User Interface

3.1 Aufbau

Die UI-Elemente sind als React-Komponenten umgesetzt und hierarchisch gegliedert. Einzelne Komponenten nutzen zusätzliche Funktionalität, die in kleine Service Module ausgelagert ist.

Die Anbindung ans Backend ist mit zwei unidirektionalen Kanälen realisiert.

Der State der gesamten Applikation wird vom Redux Store bewirtschaftet.

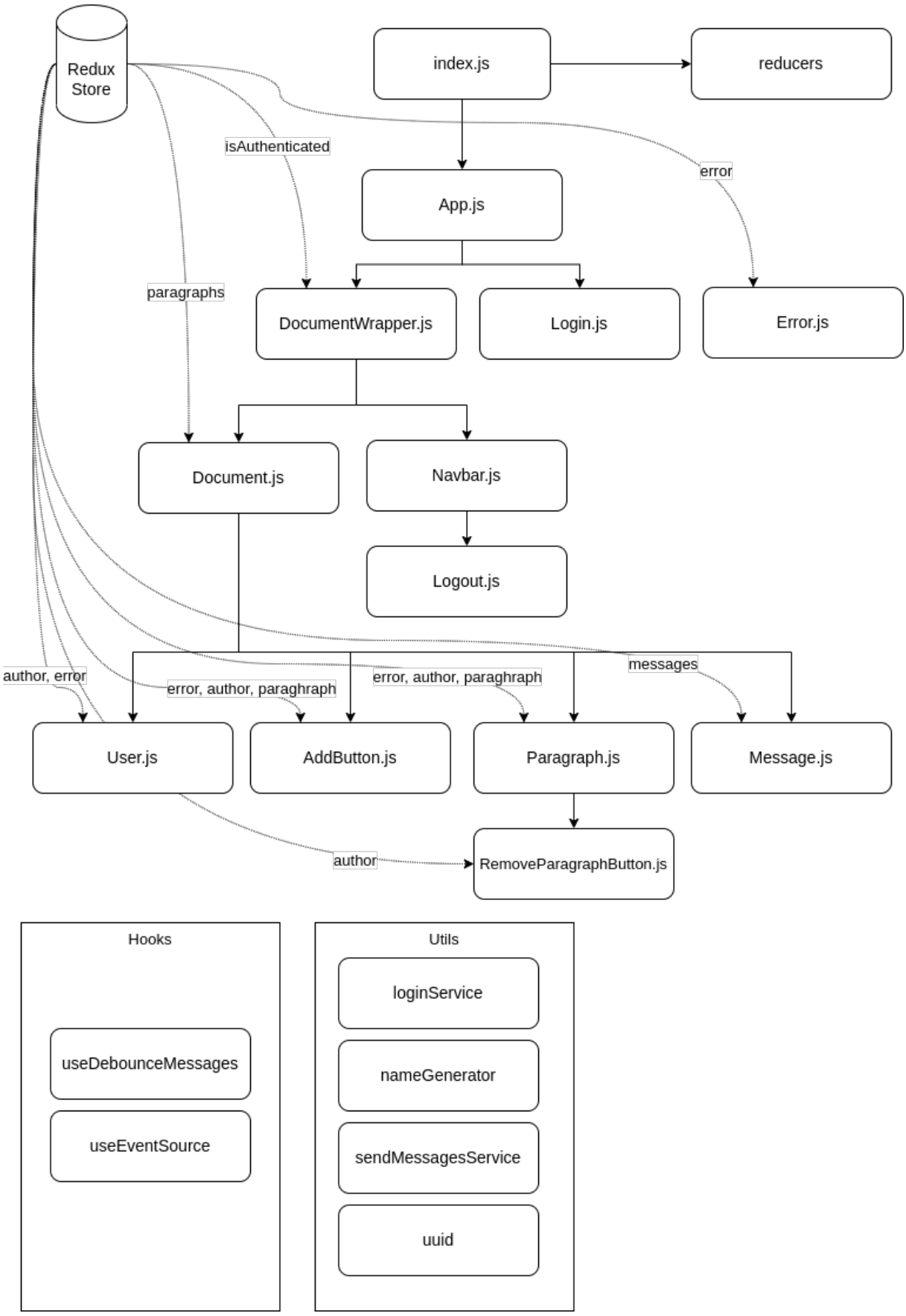


Abbildung 3.2: Komponenten Struktur

index.js

Das Index File ist der Eintrittspunkt für den Browser. Beim Laden der Applikation wird der Redux Store erstellt und initialisiert. Ebenfalls wird im Local Storage des Browsers geprüft, ob bereits ein User registriert ist. Ist dies nicht der Fall, so wird ein zufälliger Benutzer generiert.

App.js

Die App Komponente ist das äusserste Element, welches alle anderen Elemente hält. Wir verwenden einen BrowserRouter um zwischen dem eigentlichen Dokument und der Login-Seite zu navigieren.

DocumentWrapper.js

Wrapper um das Dokument zu schützen. Solange sich ein User noch nicht ordentlich am Backend authentifiziert hat, leitet diese Komponente den Benutzer stetig auf die Login-Page weiter.

Login.js

Login Formular, welches den Login Service verwendet. Das Formular übersetzt die eingegebenen Credentials in einen Basic Auth Header und sendet damit einen GET Request ans Backend. Bei erfolgreicher Authentifizierung wird das User Principal im Local Storage abgelegt.

Error.js

Generische Fehlermeldung, welche als modales PopUp angezeigt wir, im Falle eines fehlgeschlagenen Requests.

Document.js

Document ist der Parent des eigentlichen Dokuments. Hauptsächlich ist sie dafür verantwortlich alle Paragrafen sortiert darzustellen.

Navbar.js

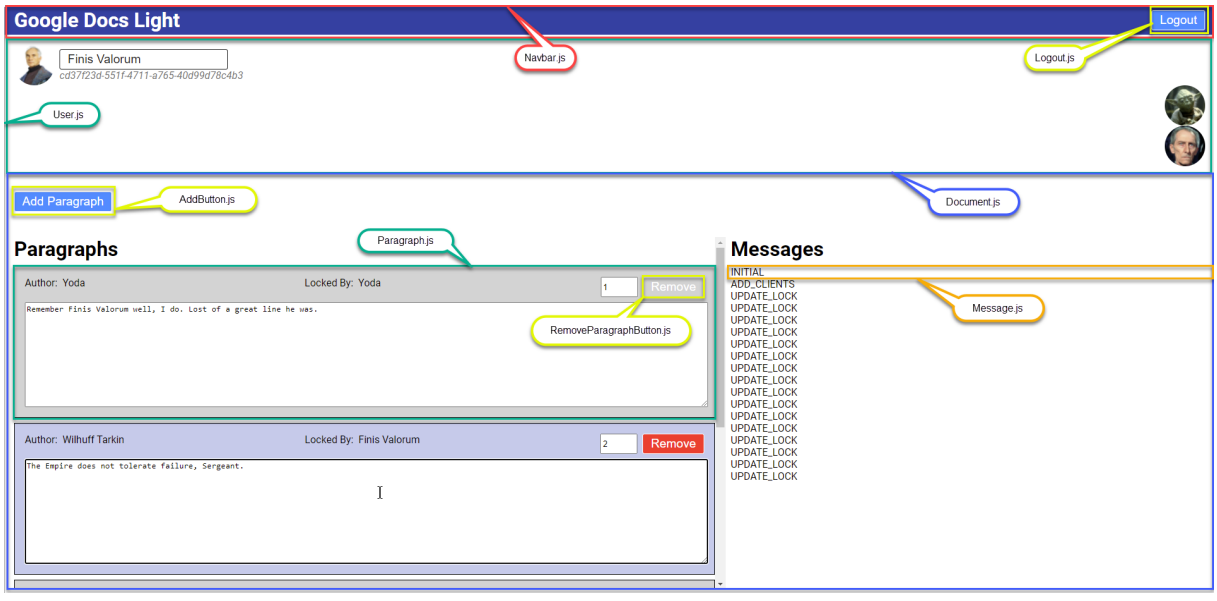
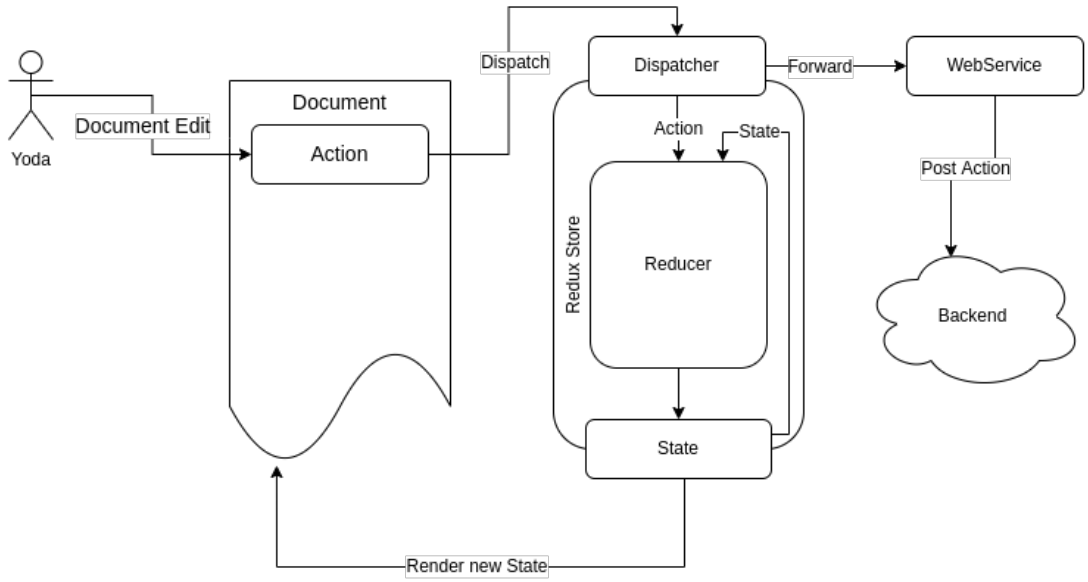


Abbildung 3.3: UI-Components

3.2 Ablaufdiagram



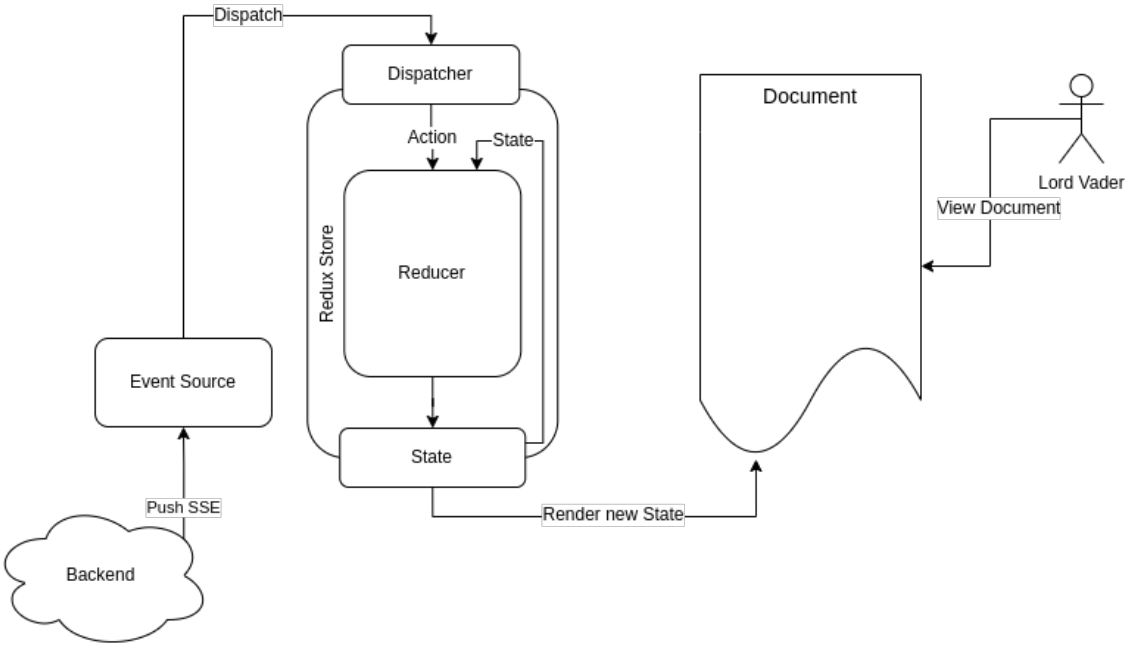


Abbildung 3.4: Datenfluss

3.3 State- und Konfliktmanagment

3.4 Fehler Behandlung

4 Backend

4.1 Aufbau

Der Aufbau der Serverapplikation orientiert sich am Konzept der Onion-Architecture. In Onion Architecture wird die Applikation in Layer aufgeteilt.

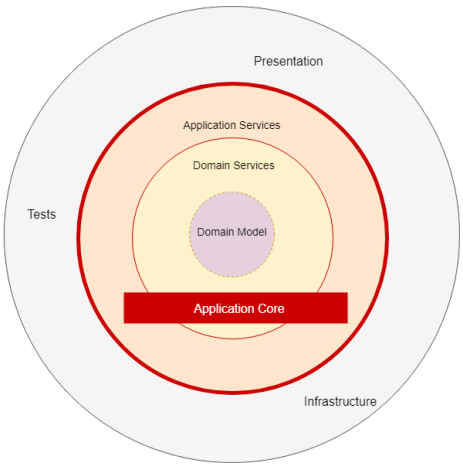


Abbildung 4.1: Onion Architecture

Um zu garantieren, dass keine ungewollten Abhängigkeiten zwischen Layern bestehen, können die Layers in eigene Module verpackt werden. Dies erhöht jedoch die interne Komplexität der Applikation. Die Umsetzung wird aufgrund der geringen Projektgrösse deshalb nicht in unabhängigen Modulen realisiert, sondern über die Package Struktur angedeutet.

Bei der Implementation wird dabei konsequent darauf geachtet, die einzelnen Layer so zu halten das diese als eigenständige Module extrahiert werden können. Für die Verwaltung der Komponenten der Serverapplikation wird folgende Packagestruktur definiert:

```
ch.fhnw.woweb.teamdocumentserver
├── api
├── config
├── domain
├── persistence
├── service
└── web
```

Abbildung 4.2: Package Struktur Cloud Service

Im Zentrum des Modells steht der Domain Layer. Dieser beinhaltet die Domänenobjekte und darf nur Abhängigkeiten auf sich selbst haben. Umgekehrt dürfen aber alle anderen Layers Abhängigkeiten auf den Domain Layer haben. Der Domain Layer wird mit dem Package domain abgebildet.

Die nächste Schicht im Modell ist der Domain Service Layer. Dieser bietet die fachliche Logik und definiert die Verhaltensweise des Domain Layers. Der Layer Application Services bildet die Brücke zwischen externer Infrastruktur und Domain Services. Dies beinhaltet Repository Services für die Schnittstelle zu persistentem Speicher und Rest Controllers für Schnittstellen zu anderen Applikationen. In der äussersten Schicht steht der Infrastructure Layer. Dieser beinhaltet externe Systeme, welche von der Applikation benötigt werden wie Datenbank und Benutzeroberfläche.

Im Zentrum steht das Package Domäne. Es beinhaltet alle Domänenobjekte und stellt alleine den Domäne Layer dar.

Der Domain Service Layer wird durch das Package Service abgebildet. Hier werden sämtliche Domain Services implementiert. Die Packages persistence und web beinhalten schliesslich den Application Service Layer. Dabei definiert das Package persistence Services welche für Interaktion mit der Datenbank verwendet werden. Das Package web definiert die HTTP-Endpunkte, welche für die Kommunikation mit dem Frontend des Systems verwendet werden. Diese werden im Package services implementiert und im package Web verwendet. Letztlich beinhaltet das Package config die technische Konfiguration der Applikation.

4.2 API

4.3 Komponenten

4.4 Sequenz

4.5 State- und Konfliktmanagment

5 Testing

5.1 Frontend

5.2 Backend

5.3 End to End Test

6 Ausblick

7 Fazit