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

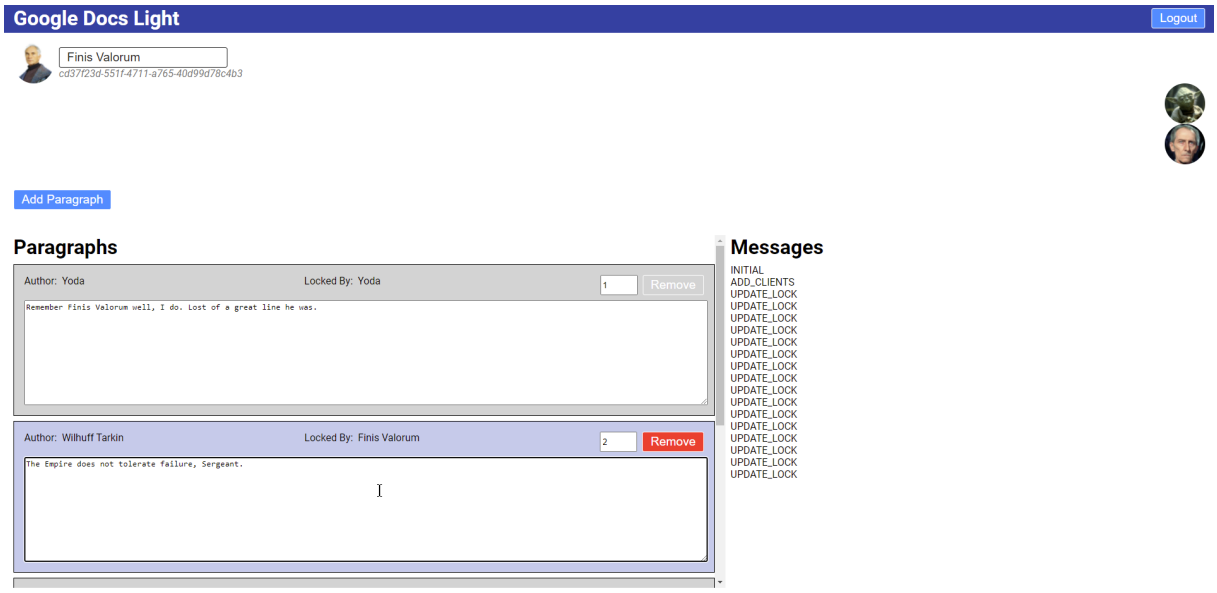


Abbildung 3.1: Team Document User Interface

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.

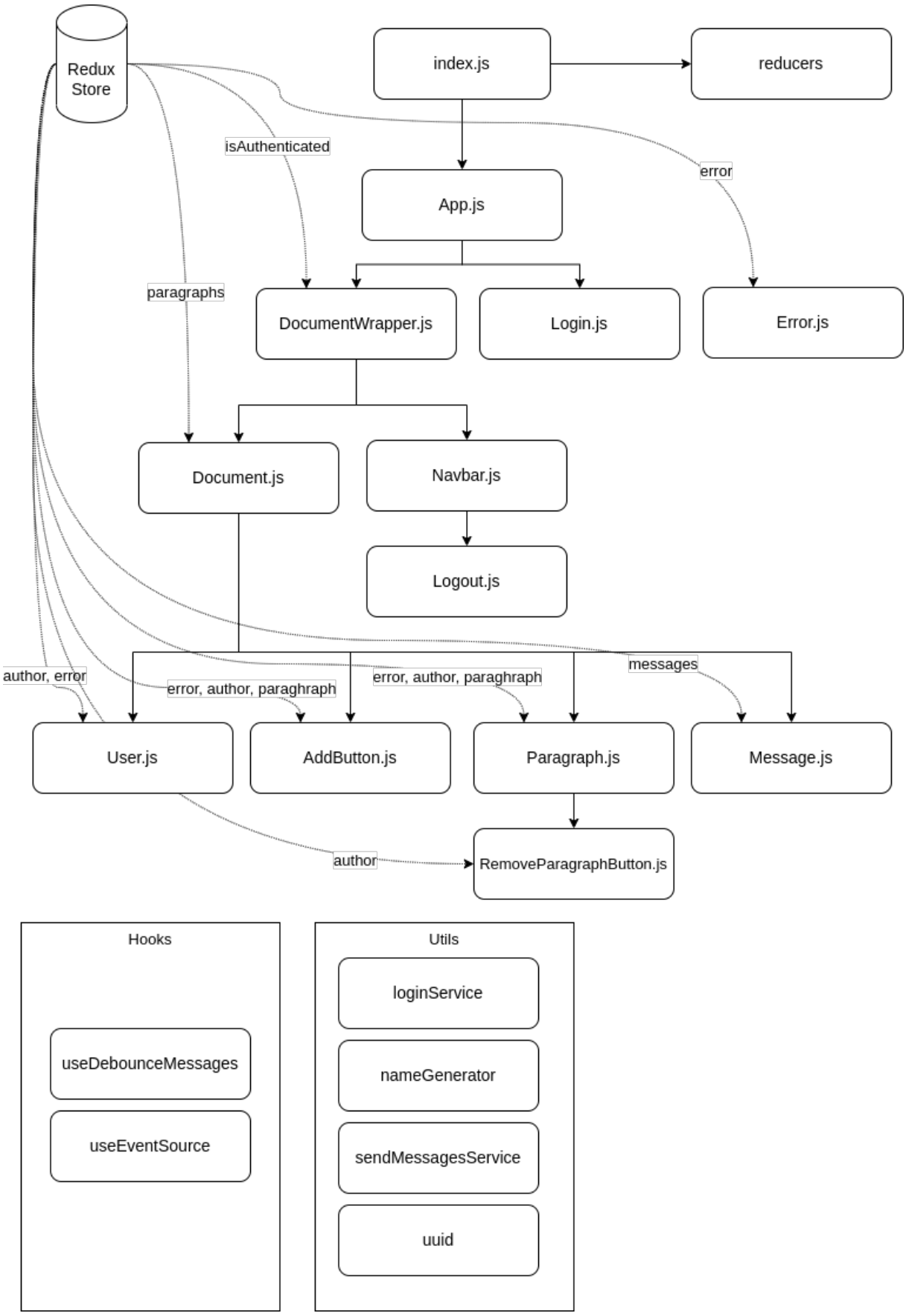


Abbildung 3.2: Komponenten Struktur

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

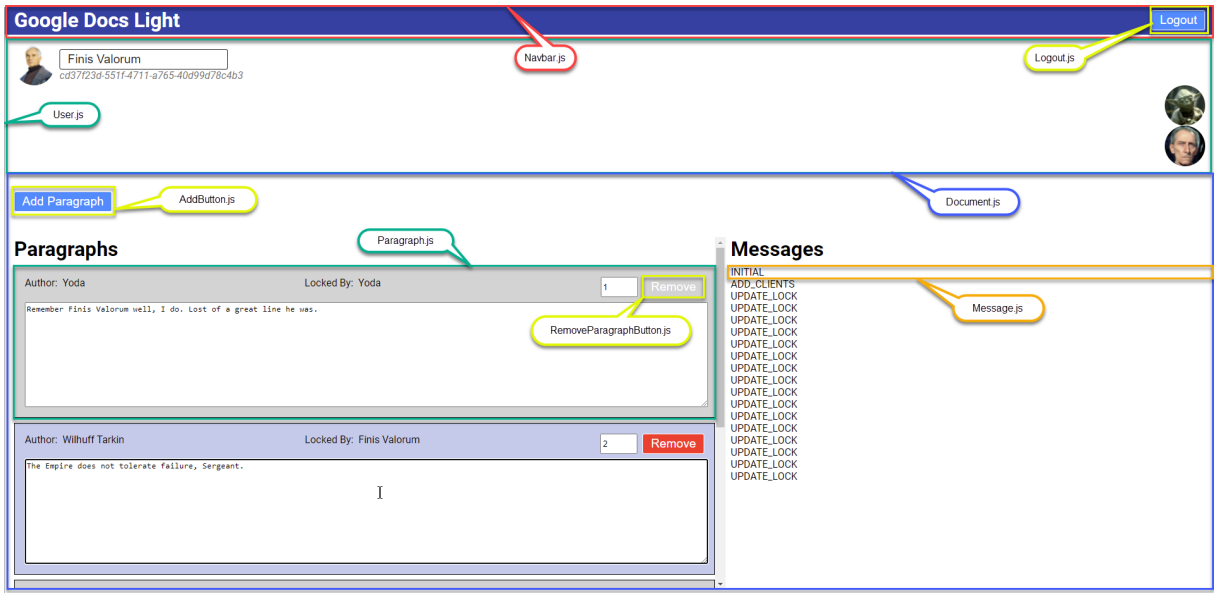
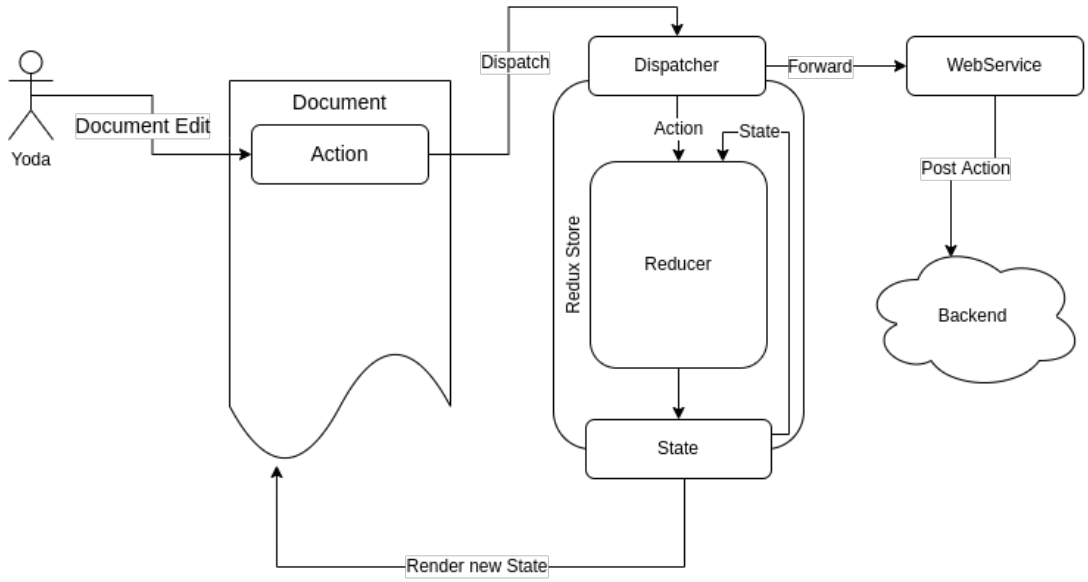


Abbildung 3.3: UI-Components

3.2 Ablaufdiagram



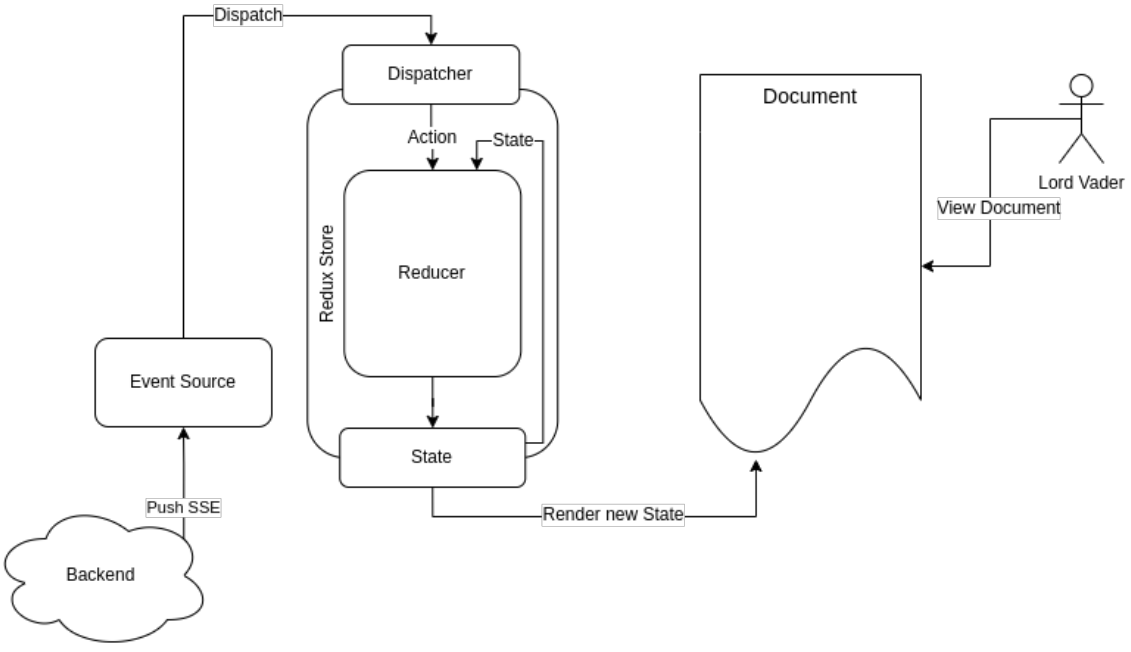


Abbildung 3.4: Datenfluss

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.

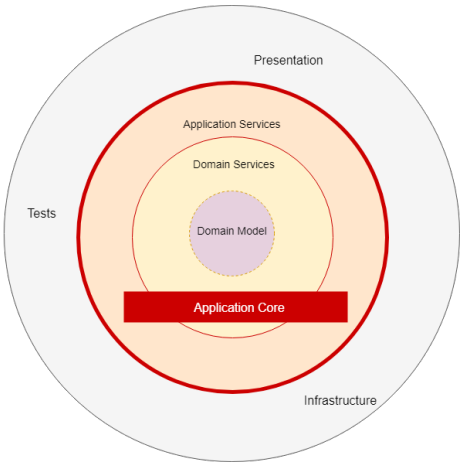


Abbildung 4.1: Onion Architecture

Um zu garantieren, dass keine ungewollten Abhängigkeiten zwischen Layern bestehen, können die Layer in eigene Module verpackt und Abhängigkeiten über Interfaces abstrahiert 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 gelöst. 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
├── config
├── domain
├── persistence
├── service
└── web
```

Abbildung 4.2: Package Struktur Cloud Service

Der Domain Layer wird durch das Package domain abgebildet. Dieses beinhaltet die Domänenobjekte und darf keine Abhängigkeiten auf andere Module oder Frameworks beinhalten. Umgekehrt dürfen alle anderen Layer Abhängigkeiten auf den Domain Layer haben. Die Fachlogik der Applikation wird im Domain Service Layer implementiert. Dieser wird durch das Package service abgebildet. Das Package Service beinhaltet alle Komponenten welche die Domänenobjekte verwalten oder den internen Zustand der Applikation führen. Der Layer Application Services bildet die Brücke zwischen externer Infrastruktur und Domain Services. Er ist in den Packages persistence und web beinhaltenagebildet. 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. Letztlich beinhaltet das Package config die technische Konfiguration der Applikation.

4.2 API

Die Backendapplikation bietet eine HTTP-Schnittstelle, welche von Frontendapplikationen verwendet werden kann. Die Schnittstelle ermöglich es, sich im System anzumelden, Dokumente zu laden und Änderungen an Dokumenten zu laden und speichern. Um diese Funktionalität zu ermöglichen, bietet die Schnittstelle die drei Bereiche ”Authentication”, ”Document” und ”Message”.

4.2.1 API Authentication

Macht Authentifizierung

Endpunkt:	/api/v1/authentication
Methode	GET
Headers:	Authentication: Basic
Response Code:	200, 401 oder 500
Response Body:	application/json

4.2.2 API Document

Liefert den Initialen Status und Updates

Endpunkt:	api/v1/document
Methode	GET
Headers:	Authentication: Basic
	X-ClientId: text
Response Code:	200, 401 oder 500
Response Body:	text/event-stream

4.2.3 API Message

Verarbeitet Updates

Endpunkt:	/api/v1/message
Methode	POST
Headers:	Authentication: Basic
	Content-Type: application/json
Body:	DocumentCommand
Response Code:	200, 401 oder 500

Stellt den zuletzt gelöschten Paragraphen wieder her.

Endpunkt:	/api/v1/message/restore
Methode	POST
Headers:	Authentication: Basic
Response Code:	200, 401 oder 500

4.3 Komponenten

4.3.1 Domäne

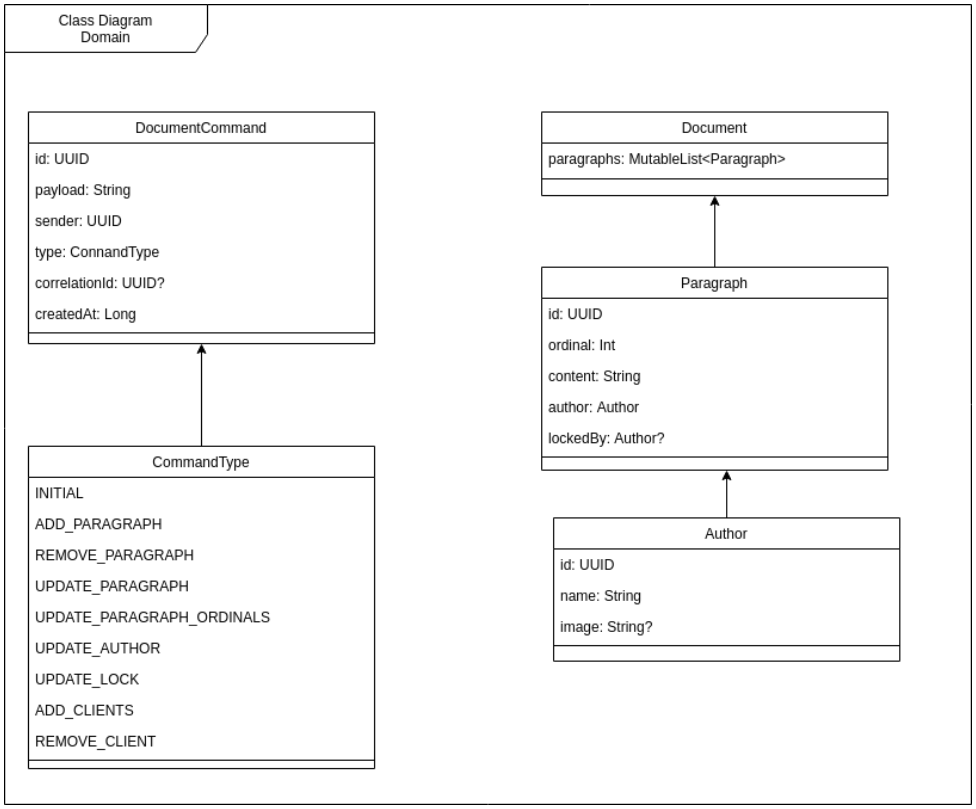


Abbildung 4.3: Klassendiagramm Domain

4.3.2 Services

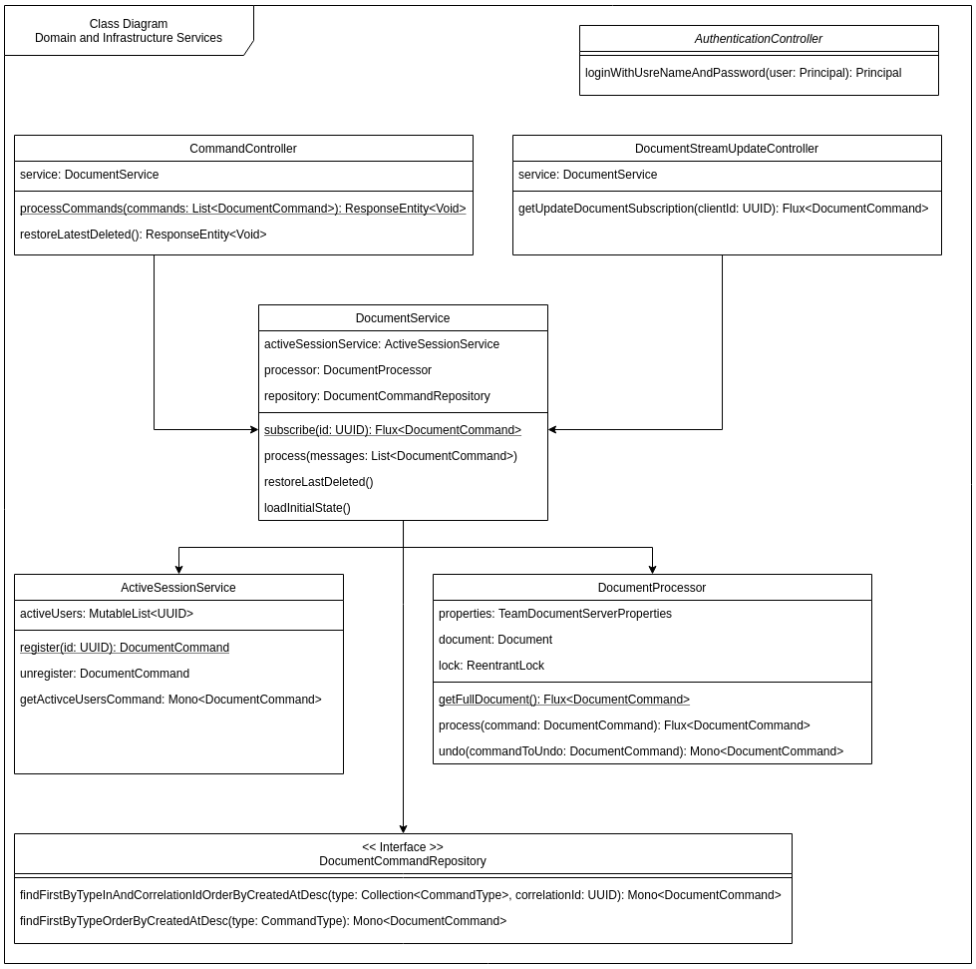


Abbildung 4.4: Klassendiagramm Services

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