Myths & Forges

Character Creation Site

Gliederung

1. Projektbeschreibung
2. Architektur
    1. Datenmodell
    2. Backend
    3. Frontend
3. Aktueller Stand
4. Fazit

Was soll die Seite können?

- Erstellen von Character-Sheets
  - Als persönliche Gedankenstütze
  - Zur Veröffentlichung von Ideen oder Modellen
  - Anwendungsfälle in Unterhaltung, Werbung/BWL Bereichen, Hobbyprojekte
- Optional public und private Sheets
- Möglichkeit Content Producern zu folgen

Architektur

Datenmodelle

Sheets

- Übliche Attribute (Name, Berufung, Urheber, etc.)
- Array aus extra Informationen
  - z.B. Genrespezifische Komponente wie Ränge, Gruppenzugehörigkeiten oder Magie
- Ein Slot für ein eventuelles Bild des Charakters

Datenmodelle

- User
  - Name
  - Mail
  - Sheet-Referenzen
- Following
  - Owner Referenz
  - Array von zu folgenden Usern

20092 - Database

sheetSchema

- name: String
- description: String
- profession: String
- creator: String
- age: String
- home: String
- isPublic: boolean
- extrainfo: [{ title: String, value: String }]
- tags: [String]
- selectedFile: String (Main picture to json)
- createdAt: Date

userSchema

- name: String
- mailAddress: String
- password: String
- sheets: [Object]
- createdAt: Date

followingSchema

- owner: ObjectId
- listOfLikeMinded: [ObjectId]

Backend

- Port 20093 für alle Zugriffe über Docker Images
  - Intern Port 5000
- Mongoose Anbindung an zur Datenbank
- Auslagerung der Routen auf Ordnerstruktur
- (Swagger in JSON)

Backend-Ordnerstruktur

Aufbau

Routen

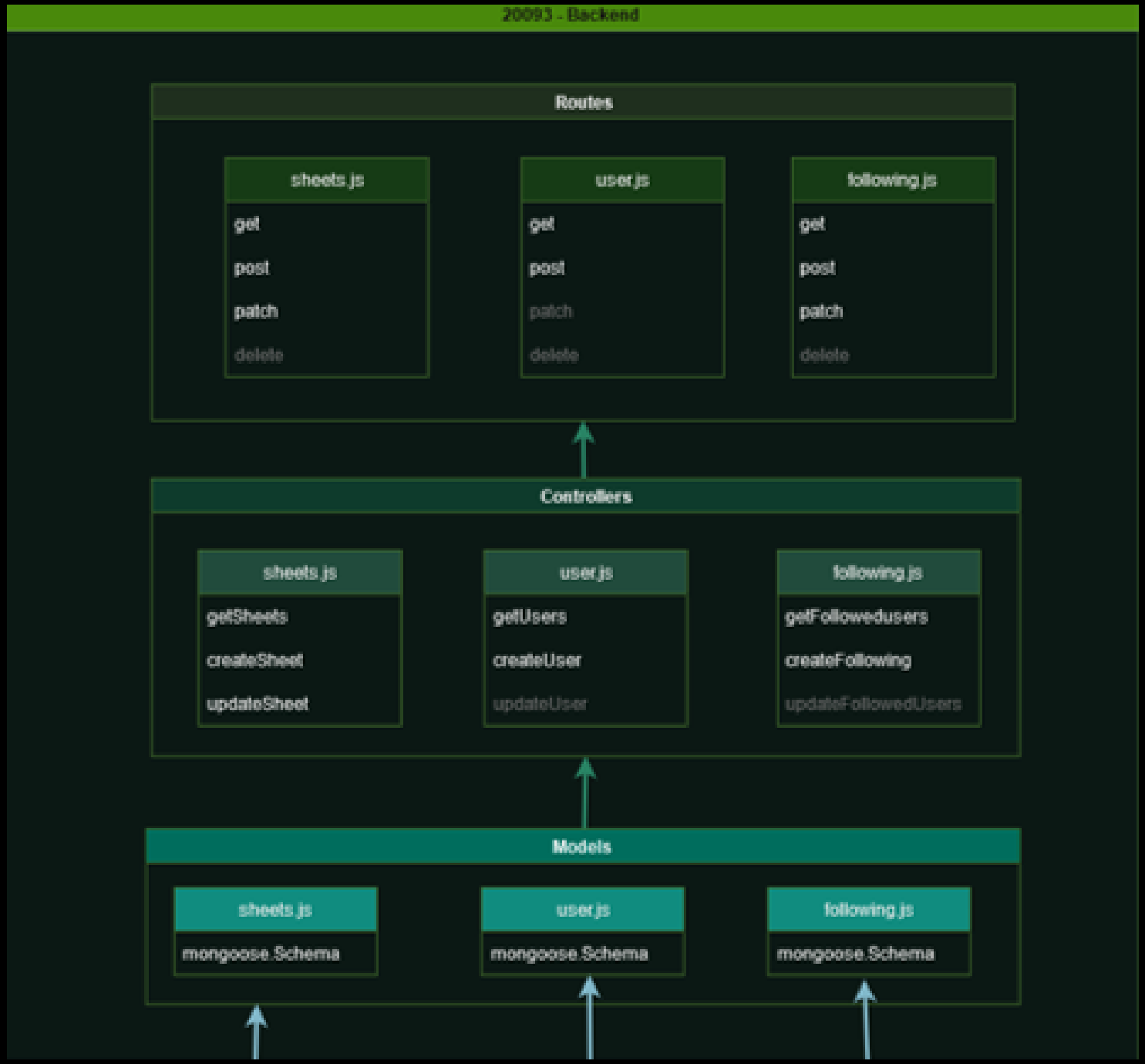
- Import von CRUD Funktionen aus Controllern
- Überleitung in express.Router()-Funktionen

Controller

- Asynchrone Datenbankzugriffe über (hauptsächlich) try-catch Blöcke
- Import von nötigen Models

Models

- Mongoose Schemata für Datenstrukture


20003 - Backend
Routes
sheets.js
get
post
patch
delete
user.js
get
post
patch
delete
following.js
get
post
patch
delete
Controllers
sheets.js
getSheets
createSheet
updateSheet
user.js
getUsers
createUser
updateUser
following.js
getFollowedusers
createFollowing
updateFollowedUsers
Models
sheets.js
mongoose.Schema
user.js
mongoose.Schema
following.js
mongoose.Schema

Frontend

- Nutzung von React-Redux
- Auslagerung der Components auf Ordnerstruktur
- Modulare Anpassung als Ziel

Frontend-Ordnerstruktur

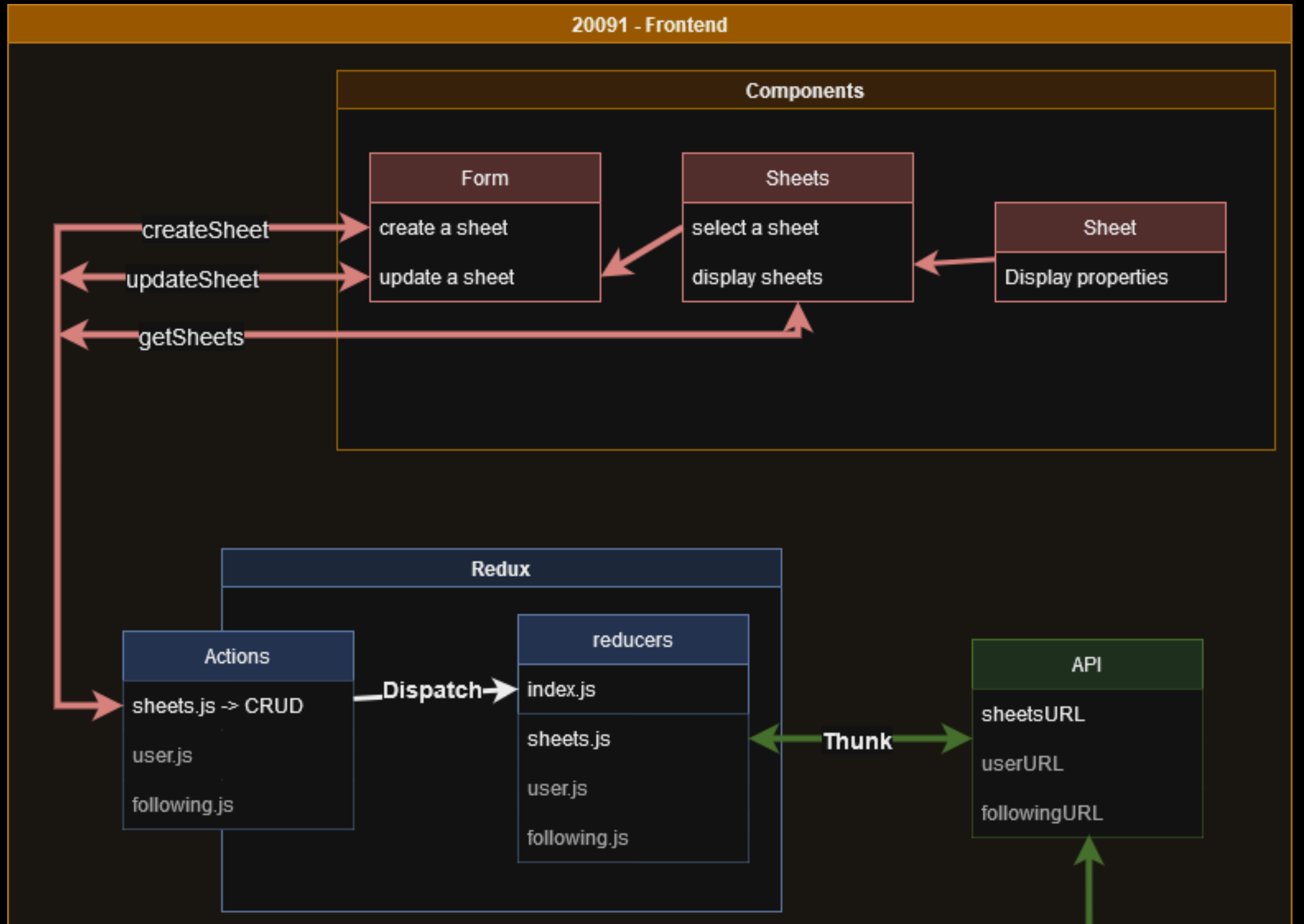
- Components
  - Import von Actions
    - Trigger durch Interaktion oder Programmabläufe
  - Form-Component zum erstellen oder Bearbeiten eines Sheets
  - Sheets-Component sammelt inital alle vorhandenen Sheets und gibt sie auf der Seite wieder
    - Sheet-Component als Vorlage von Card-Views

Frontend-Ordnerstruktur

- Actions
  - Import von API Weiterleitungen
  - Asynchrone Funktionen leiten Interaktionstyp und Payload ein
  - Per Dispatch Weitergabe an Reducer

Frontend-Ordnerstruktur

- Reducer
  - Abgleich des Action-Types
  - Payload aus Backend-Anfragen als Rückgabe
- API
  - Import von Axios
  - Angabe der URL ins Backend
    - /sheets, /user, /following
  - Operation aus Dispatch in Axios-Methoden


20091 - Frontend
Components
Form
create a sheet
update a sheet
Sheets
select a sheet
display sheets
Sheet
Display properties
createSheet
updateSheet
getSheets
Redux
Actions
sheets.js -> CRUD
user.js
following.js
Dispatch
reducers
index.js
sheets.js
user.js
following.js
Thunk
API
sheetsURL
userURL
followingURL

Aktueller Stand

- Docker Container funktionstüchtig, Backend antwortet über erste Route
- Frontend greift initial auf Sheets zu

Noch offen

- Frontend debugging
  - Probleme im store
  - Lösung vor weiteren Operationen nötig
- Swagger Fehlerfrei einbauen
- Frontendkomponenten zu User und Following aufsetzen
- Seite zusammensetzen

FAZIT