FURPS+

Krav fra opgavebeskrivelsen fortolkes herunder i forhold til FURPS+ modellen.
De formulerede krav i opgavebeskrivelsen indsættes under modellens punkter.

Functionality

- REST API implementeret med Node.js og Express.js
- Routes med endpoint for HTTP-metoderne GET, POST, PUT/PATCH, DELETE.
- CRUD-operationer der læser og skriver til JSON-fil.
- Det skal være muligt at kunne få både alle objekter og et objekt på baggrund af specificeret id.
- JSON-fil som datakilde
- Artist-objekterne i JSON-listen skal bestå af minimum følgende properties:
    - Name
    - Birthdate
    - activeSince
    - genres
    - labels
    - website
    - image
    - shortDescription
- User Interface implementeret med HTML, CSS og JavaScript
- Brugeren skal kunne oprette, læse, opdatere og slette data (CRUD)
- Filtrering og sortering på udvalgte parametere (props)
- Du skal kunne markere en kunstner som favorit og vise en liste over alle favoritkunstnere. Du bestemmer selv, hvordan denne liste gemmes (backend, localStorage, variabel eller lignende).
- Anvendelse af CSS Grid, CSS Flex og/eller HTML Table samt relaterede HTML-elementer
- Kode opdelt i modules

Usability

- Brugere skal kunne administrere og udforske information om musikkunstnere.
- Brugergrænsefladen skal være brugervenlig og nem at navigere.
- Alle CRUD-operationer skal være let tilgængelige
- filtrerings/sorteringsfunktionerne skal være intuitive.

Reliability

- Applikationen skal være stabil og pålidelig.
- Dataintegritet og korrekt håndtering af CRUD-operationer er afgørende.

Performance

- Applikationen skal have en acceptabel ydeevne, herunder hurtig datahentning og responsivitet i brugergrænsefladen.

Supportability

- Koden skal være velstruktureret og veldokumenteret, så det er nemt for fremtidige udviklere at vedligeholde og udvide applikationen.

+

Både frontend og backend opbygges efter generelle principper som:

- **Separation of Concerns** – så ting er adskilt så meget som muligt, I bruger forskellige funktioner/moduler til at manipulere data og vise data.

- **Loose Coupling** – så funktioner er så uafhængige af hinanden som muligt, eller i det mindste kun har afhængigheder en vej.

- **High Cohesion** – så funktioner der arbejder med det samme er samlet så tæt som muligt, enten i closures eller i modules.