Afsluttende opgave - Dataservice

Indledning

Du har i løbet af ugerne her arbejdet med:

- **Opsætning af en backend-server** som kan håndtere data (GET POST PUT DELETE) fra Mongo-databasen – serveren opsat med Express-server. Eller noget tilsvarende (fx Firebase)
- **Single fileupload** – altså upload af 1 fil/image
- **Multiple files upload** – altså upload af flere filer på én gang
- **Multiple files upload** – med validation/validering (af filtyper og deres antal/størrelse)

Afsluttende opgave(r)

Opgaven her består i afslutningsvis at nå så mange af nedestående opgaver, som du kan.

Det skal afleveres i dag kl. 14:

- **På din GIT-konto: Upload** dit projekt (både backend og frontend)
- **På VIDOnline:** Aflever et **tekstdokument** (txt) med **GIT-link** til at hente dit projekt + skriv **hvor langt du nåede** i opgaverne herunder

Opgave 1

1. Opret en folder og kald den fx **DataserviceAfslutningsopgave**
2. Og i den folder opret en **folder** – kald den fx **backend** eller **backendscuba** el.lign.
   Husk at installere følgende pakker bakcend-mappen (**`cd backend`**):

```
npm init -y
npm install express body-parser cors mongoose
npm install -g nodemon
```

3. Opret en **server.js** (Express) med en model fx **scubaprodukt.js** i en **model**-folder eller bare **scubaprodukt.model.js**)med følgende indhold:
   a. **Produktetnavn**
   b. En kort **produktbeskrivelse**
   c. En **pris**

    d. Et **produktfoto** (her bruger du bare teksten ”foto-paa-vej.jpg” hver gang – med mindre du har godt styr på upload af foto)

4. Lav følgende API-metoder på serveren (de behøver ikke tage højde for images/billeder):
    a. **GET**: Vis alle produkter
    b. **POST**: Opret et produkt
    c. **PUT**: Ret et produkt (ud fra ID)
    d. **DELETE**: Slet et produkt (ud fra ID)
5. **Start serveren** og test af metoderne virker i **Postman**

Opgave 2

1. **Opret en Rect-app** – kald den scuba-app eller scubafrontend eller noget tilsvarende
2. Install Bootstrap, Materialize eller andet styling, hvis du foretrækker det – men brug ikke mere end max 10 min. på det.
3. **Opret en component**, hvor man kan oprette et nyt produkt (fx **OpretProdukt.js** i en component-folder) – image-feltet lader du bare være tomt
4. **Test, at opret-component virker** – tjek at der bliver puttet noget i databasen

Opgave 3

5. **Udtræk alle produkter på i App-component** eller i en selvstændig component fx **VisProdukter**-component
6. Ved produktbilledet viser du bare et foto-på-vej.jpg-billede (find et på nettet)
7. **Test, at man får vist alle produkter fra databasen**

Opgave 4

1. **Tilføj et ”Ret produkt”**-link ved alle produkter i produktoversigten
2. **Opret en component**, hvor man kan RETTE i et produkts data (fx **RetProdukt.js** i en component-folder) – igen må du gerne ignorere image-feltet
3. **Test, at ret-component virker** – tjek at der bliver rettet i databasen

Opgave 5

1. **Tilføj et ”Slet produkt”**-link ved alle produkter i oversigten
2. **Opret en component**, hvor man kan SLETTE et produkt (fx **RetProdukt.js** i en component-folder) – igen må du gerne ignorere image-feltet

3. **Test, at ret-component virker** – tjek at der bliver rettet i databasen

Opgave 6

1. Lav en if-sætning, så der KUN bliver vist et foto-paa-vej.jpg i oversigten, hvis image-feltet i databasen er tomt.
2. Udfyld direkte i database et par produkter med nogle opdigtede fotos
3. **Test, at der kun bliver vist foto-paa-vej.jpg, ved de produkter, der IKKE har et image** – alle øvrige produkter skal stadig vises med foto-paa-vej.jpg

Opgave 7

- Når ALT herover virker, så prøv kræfter med at kunne uploade et image sammen med produktet. Test at det virker, inden du evt. forsøger med næste punkt.
- Hvis du får det til at virke, så prøv evt. også (hvis der er mere tid tilbage) at forsøge at tilføje funktionen, så man kan rette et billede (vælge et andet image).