Inlämningsuppgift 4

Avancerad JavaScript med React (60p)

- Beskrivning
  - Förkunskap
  - Begränsningar
  - Inlämning
- Instruktioner
  - Backend
    - GET /api/scores
      - Request
      - Response
    - POST /api/scores
      - Request
      - Respons
    - GET /api/games
      - Request
      - Respons
  - Frontend

Beskrivning

Detta är inlämningsuppgift 4 för kursen "Avancerad JavaScript med React" där du bygger vidare på det arbete du utfört för inlämningsuppgift 1, 2 och 3.
I denna sista inlämningsuppgift kommer du bygga en backend för applikationen som gör det möjligt att hämta och spara highscores. Du kommer även använda Redux för att lagra och hantera applikationens tillstånd.

Förkunskap

För att lösa uppgiften behöver du följande kunskap:

- Grundläggande kunskap inom JavaScript.
- Hur man bygger en enkel backend med ett API (Application Programming Interface) som låter frontend hämta och lagra information.
- Hur man använder Redux tillsammans med React, inklusive hur man hanterar sidoeffekter (såsom exempelvis att hämta och skicka data till backend) med hjälp av "thunk"-funktioner ("redux-thunk").

Begränsningar

- React måste användas.
- JavaScript eller TypeScript måste användas.
- Backend ska finnas med API för hämtning av highscores och spel, samt lagring av highscores.
- Redux ska användas för att lagra och hantera applikationens tillstånd.
- Det ska gå att verifiera funktionalitet i webbläsaren.

Inlämning

Radera node_modules (du kan enkelt återställa den sen igen med "npm install" och komprimera sen hela projektet till en Zip- eller RAR-fil och ladda upp denna till Ping Pong under Examinering > Inlämningsuppgifter > Inlämningsuppgift 4, **senast fredag 16/4, 23:55**.
Innan du lämnar in, försäkra dig om att applikationen uppfyller samtliga punkter listade under "Begränsningar" ovan.
För snabbare rättning, boka tid med läraren för redovisning (1-on-1) i samband med att du lämnar in uppgiften.

Instruktioner

Backend

Vi lägger på en backend till applikationen, som låter frontend hämta highscores och spel, samt lagra nya highscores. För detta behöver vi ett API med tre endpoints:

- GET /api/scores (hämta highscores)
- POST /api/scores (skapa highscore)
- GET /api/games (hämta spel)

Detaljer för varje finner du nedan.

GET /api/scores

Vi behöver kunna hämta ut highscores från applikationen. I en riktigt applikationen är det osannolikt att vi vill hämtat ut samtliga highscores på en gång. Istället hade vi mest sannolikt enbart hämtat de vi behöver just nu. Men för att hålla exemplet enkelt väljer vi här att hämta ut samtliga highscores direkt.

Request

GET /api/scores	

Response

Oavsett om det finns highscores eller ej, kommer vi alltid returnera statuskod "200 OK" och en array, som möjligtvis är tom om det nu inte finns några highscores att returnera.

Status	200 OK
Body	[{ "id": 1, { game: { "id": 1, "title": "Tetris" }, "score": 12345, "date": "2020-01-01 20:15" }, ...]

POST /api/scores

Det ska gå att registrera nytt highscore.

Request

POST /api/scores	
Headers	Content-Type: application/json
Body	{ "gameId": 1, "player": "Jane Doe", "score": "190000", date: "2021-01-01 20:15" }

Respons

Om spelet finns, registrerar vi highscore. Vi hade normalt velat ta höjd här för ett scenario där vi får in ogiltig data - men vi håller det enkelt och utgår ifrån att vi aldrig kommer få det.

Status	201 Created
Body	{ "id": 10, "game": { id": 1, "player": "title": "Tetris" }, "player": "Jane Doe", "score": "190000", date: "2021-01-01 20:15" }

Notera att vi här skickar med information om spelet, eller åtminstone de delar av spelet vi kommer behöva för att visa ett highscore. Detta är ett designval vi gjort för denna applikationen.

GET /api/games

När man registrerar ett nytt highscore behöver man lista tillgängliga spel man kan registrera highscore för. Exempelvis har man en <select> där man väljer spel, eller en kontroll precis innan man skickar data till backend, för att säkerställa att spelet finns.

Oavsett vad behöver vi kunna få ut en lista av spel.

Request

GET /api/games

Respons

Oavsett om det finns spel eller ej, kommer vi alltid returnera statuskod "200 OK" och en array, som möjligtvis är tom om det nu inte finns några spel att returnera.

Status	200 OK
Body	[{ "id": 1, "title": "Tetris", "description": "Lorem ipsum dolor", "imageUrl": "http://domain/image.png" }, ...]

Frontend

För tillfället laddas data relaterad till highscores och spel från statiska filer inledningsvis när applikationen startar upp. Vi ska nu istället ladda informationen från en riktig backend - den som beskrivs i instruktionerna ovan.

Applikationens ska dessutom använda Redux för att lagra och hantera tillstånd i applikationen. Eftersom vi har asynkrona operationer (ladda data, registrera highscore) behöver applikationen även använda "thunk"-funktioner ("react-thunk").

Du behöver alltså göra följande:

- Installera följande paket:
    - redux
    - react-redux
    - redux-thunk
- Flytta applikationens tillstånd till Redux - vilket kommer innebära att du tar bort merparten av useState (kommer dock låta dessa ligga kvar i formuläret som används för att registrera highscores).
- Använd API exponerat av backend för att hämta/lagra data för applikationen.

FAQ

Får man använda Redux Toolkit?

Det går bra.