HÖGSKOLAN DALARNA

Projektarbete

Komplett webbsida med CRUD-funktionalitet

Datum

2022-12-12

PROJEKTARBETE

Komplett webbsida med CRUD-funktionalitet

HÖGSKOLAN DALARNA

Projektarbete

Komplett webbsida med CRUD-funktionalitet

Datum

2022-12-12

Förberedelser

1. Skapa en mapp på någon av era datorer.
2. Initiera ett git-repository i mappen.
    a. [`git init`]
3. Skapa ett repository på GitHub.
4. Döp det till **gik2f8-[gruppnummer]-projekt**.
5. Ange remote-adress till ditt repository på GitHub
    a. [`git remote add origin [url till ditt GitHub-repo]`]
6. Skapa en fil (vilken som helst) och gör en initial commit, samt push till ert GitHub-repo.
    a. [`git add .`]
    b. [`git commit -m "init"]`
    c. [`git push --set-upstream origin master`].
7. Se till att alla i gruppen har tillgång till GitHub-repositoryt.
8. Den av er som inte skapade git-repositoryt på sin dator i steg 1 får ladda ner det
    a. [`git clone [url till ditt GitHub-repo]`]
9. **Viktigt!** Se alltså till att alla i gruppen arbetar mot samma GitHub-repository. Ni har alltså en varsin kopia av repot lokalt, men pushar kod till ett gemensamt GitHub-repo.

HÖGSKOLAN DALARNA

Projektarbete

Komplett webbsida med CRUD-funktionalitet

Datum

2022-12-12

Instruktioner

Ni ska skapa en webbsida där man kan utföra CRUD-funktionalitet för en viss resurs. Det kan vara vad som helst –, generella produkter som t.ex. hos en webbshop. Ni väljer själva.

Webbsidan kommer när den är färdig att likna Todo-applikationen såsom den såg ut efter Lektion 6, gällande funktionalitet och komplexitet.

Krav

Webbsidan ska:

- Visa en lista av en viss resurs, ni väljer vilken. (**R i CRUD**)
  - Listan ska presenteras dynamiskt, dvs. inte finnas i HTML-koden från början.
  - Listan ska hämtas med hjälp av ett fetch()-anrop till ett API mot ett backend som ni ska skriva med hjälp av NodeJS enligt instruktioner nedan.
  - När listan har hämtats ska den konverteras från JSON till ett JavaSciptobjekt så att den kan loopas igenom och kan ritas ut i HTML.
- Ha ett formulär för att skapa en ny resurs (**C i CRUD**)
  - Hur många eller vilka fält som ska finnas bestämmer ni själva.
  - Ni får göra validering på formuläret, men ni behöver inte.
  - Data från formuläret ska hämtas och samlas till ett objekt som sedan ska skickas med hjälp av fetch()-funktionen och metoden POST till ert eget backend.
  - När den nya resursen har sparats ska ni presentera den nytillagda resursen på sidan dynamiskt genom att direkt när förfrågan fått tillbaka ett lyckat resultat uppdatera listan.
- Möjliggöra borttagning av en resurs.
  - En knapp ska visas i närheten av varje resurs i listan.
  - Vid klick på knappen ska eventet fångas upp
  - Till eventlyssnaren ska ett id skickas. Det ska vara id:t för den resurs som ta bort-knappen ligger intill.

Webbsidan kan, men måste inte:

- Möjliggöra uppdatering av en resurs (**U i CRUD**)
  - Detta kan göras genom att man i gränssnittet tillåter att användaren på något sätt ändrar en befintlig resurs.
  - Detaljer kring hur detta görs är upp till er att lista ut, men ett exempel kan vara
    - En knapp ritas ut intill en resurs i listan som läser in resursens data till formuläret igen, så att man kan ändra och sedan klicka på spara för att skicka en PUT-request till ett API mot ert backend.
    - Att någon del hos en resurs kan vara editerbar, som t.ex. checkboxen vid uppgifterna i Todo-applikationen.
    - Att man sedan fångar eventet när någon interagerar med denna del och gör en förfrågan av metoden PUT eller PATCH till ett API mot ert backend.

    - Anledningen till att U är valfritt är att ni examineras i den funktionaliteten i Labb 2.
- Ha validering på formuläret

För att möjliggöra ovanstående frontend-funktionalitet måste ni:

- Skapa ett backend med NodeJS
- Exponera passande routes så att det går att göra förfrågningar av motsvarande http-metoder
- Ha en route per http-metod; en för GET, en för POST och en för DELETE.
- Ha enroute för put eller patch behöver finnas om ni väljer att implementera funktioner för att uppdatera er resurs.
- Ha adresser hos routerna som motsvara namnet på er resurs. Man ska alltså göra API-anropet mot ***http://localhost:5000/[resursnamn]*** för exempelvis GET eller POST, samt ***http://localhost:5000/[resursnamn]/[id]*** för DELETE.
- Resursen ska skrivas till fil för lagring.

Checklista

- ✓ All logik för frontend och backend ska skrivas i JavaScript.
- ✓ Utseendet på sidan ska skapas med hjälp av ett och valfritt CSS-ramverk.
- ✓ Samtliga CRUD-operationer ska kunna skickas via API-till ert eget backend och resultatet ska sparas till fil.
- ✓ Ett CSS-ramverk ska användas för att skapa design och layout på webbsidan.
- ✓ Resursen ska skrivas till fil med hjälp av NodeJS inbyggda modul fs eller fs-promises.

Ni väljer själva

- Vad webbsidan ska handla om
    - Inom rimliga gränser, inget stötande eller olämpligt på andra sätt.
- Vilken resurs som ska finnas
    - Exempelvis filmer, böcker, användare, blogginlägg eller produkter.
    - Resursen som hanteras på sidan får inte vara stötande eller olämplig på andra sätt.
- Huruvida ni använder externa npm-paket i backend, såsom exempelvis express.
- Layout och styling av webbsidan.
- Vilket CSS-ramverk som ska användas och hur det ska användas.
- Om det ska finnas ytterligare funktionalitet, så länge grundkraven uppfylls.

HÖGSKOLAN DALARNA

Projektarbete

Komplett webbsida med CRUD-funktionalitet

Datum

2022-12-12

Inlämning

Del 1: Kod

- Lämna in länk till GitHub-repository som skapades och lades upp på GitHub under Förberedelseavsnittet.
    - Länken ska se ut något i stil med: https://github.com/[**något-av-era-github-användarnamn**]/*gik2f8*-[**gruppnummer**]-*projekt*
    - Endast en (1) länk till GitHub-repository ska lämnas in.
- **Observera**! Lämna *inte* in en zip:ad mapp med er kod!

Del 2: Inspelning

Gör en inspelad presentation där ni

1. Demonstrerar er site genom att klicka igenom flöde att skapa, visa, uppdatera och ta bort.
2. Visar och förklarar utvalda delar av koden för att belysa
    a. Hur ni genererar dynamisk HTML utifrån förändrad data
    b. Skickar data till backend
    c. Tar emot data i backend och skickar svar tillbaka
3. Gruppens erfarenheter av arbetet
    a. Svårigheter & utmaningar
    b. Reflektion av samarbetet i gruppen
    c. Om ni vill: Kod eller lösningar som ni tyckte blev extra bra och vill framhäva

Alla i gruppen ska visa delta i presentation av samtliga tre delar, genom att visa och/eller förklara.