Avancerad JavaScript

Gruppuppgift

NORTH LINK

Gruppuppgift

- De sista veckorna i den här kursen ska ni jobba med ett grupp-projekt
- Målet med projektet är att bygga en klient för Dropbox som är en populär molnlagringstjänst
- Ni kommer att använda Dropbox officiella SDK för att bygga applikationen

https://github.com/dropbox/dropbox-sdk-js

Gruppindelning

- Jag vill att ni delar in er i grupper om 3 till 4 personer
- Ni som inte redan har bildat en grupp bör göra det senast idag

Redovisning

- Ni kommer att bygga applikationen tillsammans i grupper men projektet ska redovisas individuellt
- Ni behöver därför ha bra koll på hur hela applikationen fungerar och kunna svara på frågor om den
- Jag kommer att återkomma med mer information om hur redovisningen kommer att gå till

Dropbox

- Dropbox är en väldigt populär molnlagringstjänst som är gratis att använda (upp till 2GB)
- Om ni inte redan använt Dropbox rekommenderar jag att ni börjar med att registrera ett konto och testar tjänsten så att ni får en bra bild av hur den fungerar och vad man kan göra

https://www.dropbox.com

Vad ni ska implementera

- Ni ska implementera en webapplikation med React som använder sig av Dropbox officiella SDK
- Applikationen ska implementera viss (långt ifrån all) funktionalitet som finns i på Dropbox egen webapplikation

Betygssättning

- Det finns olika krav på applikationen för betygen G och VG
- Även redovisningen är betygsgrundande
- Efter den här sliden kommer alla kraven för både G och VG att presenteras

Krav (G) - Autentisering

- Ni behöver implementera autentisering med hjälp av OAuth https://www.dropbox.com/developers/reference/oauth-guide
- När en användare loggar in på sitt konto på Dropbox kommer de få en fråga om de vill godkänna att er applikation får tillgång till hans eller hennes konto
- Resultatet av OAuth-flödet är en token som ni behöver spara i localStorage
- Denna token ska tas bort från localStorage när användaren loggar ut

Krav (G) - UI

- Applikationens UI måste innehålla
  - Ett huvudinnehåll som visar innehållet (filer och kataloger) i en katalog
  - Sökvägen till katalogen ska visas längst upp på sidan. Varje del av sökvägen måste vara klickbar så att användaren kan navigera högre upp i hierarkin.
  - Om en fil är en bild ska en thumbnail visas istället för en vanlig ikon
  - Metadata för filerna ska visas och den måste inkludera: filnamn, filstorlek (i ett format som människor kan förstå), tiden filen senast modifierats
- När en användare klickar på en fil ska en nedladdning starta
- En användare måste kunna ladda upp filer till en katalog
- En användare måste kunna skapa en ny katalog
- En användare måste kunna ta bort filer och kataloger. Innan en fil/katalog tas bort ska applikationen fråga användaren om han eller hon verkligen vill ta bort filen eller katalogen
- Det ska finnas en knapp för att få till föräldern till nuvarande katalog

Krav (G) - Favoriter

- En användare måste kunna lägga in en fil eller en katalog bland sina “favoriter”
- Gränssnittet ska ha en vy där användaren kan se alla sina favoritkataloger och filer
- Den här funktionaliteten ska implementeras på klienten. Spara favoriter i localStorage
- Om en favorit tas bort eller flyttas utifrån applikationen kommer den inte längre gå att komma åt genom applikationen. Det här felet måste hanteras korrekt
- Kom ihåg att ta bort favoriter från localStorage när användaren loggar ut

Krav (VG) - Sökning

- En användare ska kunna söka efter filer och kataloger
- Visa sökresultat istället för huvudinnehållet
- Om en användare klickar på en katalog ska applikationen navigera till den katalogen
- Om en användare klickar på en fil ska filen laddas ner

Krav (VG) - UI

- En användare ska kunna kopiera filer och kataloger. Ni kan antingen implementera det så att kopian hamnar i samma katalog med ett annat namn eller att användaren får välja målkatalog
- En användare ska kunna byta namn på filer och kataloger
- En användare ska kunna flytta filer och kataloger. Applikationen ska visa en dialogruta där användaren kan välja målkatalog

Krav (VG) - Realtidsuppdateringar

- Om en ändring sker utanför applikationen måste innehållet uppdateras automatiskt
- Ni kan antingen lösa det här med pollning (göra anrop med SDK:et i ett intervall) eller använda webhooks (mer om det senare)

Krav (VG) - Tester

- För VG krävs att ni skriver tester för minst en av era komponenter
- Skriv gärna fler tester så att säkerställa att er applikation är stabil

Webhooks

- För VG krävs att ni uppdaterar innehållet i en katalog automatiskt om någon ändring sker utanför applikationen
- Till exempel kan ni göra ändringar i den officiella webapplikationen eller göra ändringar på er egen dator om ni har synkning aktiverat
- Ni kan antingen lösa detta genom att pollning men det finns ett bättre sätt eftersom Dropbox erbjuder webhooks
- För att använda webhooks krävs att ni har någon typ av backend som hanterar anrop från Dropbox

https://www.dropbox.com/developers/reference/webhooks

Implementation

- Applikationen ska implementeras som en SPA med React
- Ni ska implementera korrekt routing
- För att kommunicera med Dropbox API ska ni använda deras officiella SDK http://dropbox.github.io/dropbox-sdk-js/
- En referens över alla metoder finns här https://dropbox.github.io/dropbox-sdk-js/Dropbox.html
- För att implementera UI-komponenterna kan ni använda ett CSS-bibliotek eller skriva CSS själva

Deployment

- När ni är färdiga ska ni publicera er applikation så att den är tillgänglig på en publik URL
- Ni kan publicera applikationen hur ni vill, t ex genom att använda GitHub Pages

Tips

- Bestäm er layout och struktur innan ni börjar programmera applikationen
- Försök dela upp implementationen i oberoende delar så att alla medlemmar i gruppen kan arbeta med olika delar samtidigt
- Hjälp gärna varandra och fråga om ni kör fast
- Lycka till!