Lab 4: Fullstack med Vue och Express

Översikt
I den här labben ska du bygga ett labbokningssystem för att kunna boka in labbtider hos specifika handledare. Om du tänker göra bonusuppgiften är det rekommenderat att göra den samtidigt som grunduppgiften. För att komma igång med labben är det bra att kika i README:n i git-repot ni fått tillgång till.

Systemet ska bestå av 3 sidor:

- En bokningssida där studenter kan se handledarnas namn och vilka tider som finns för en handledare
- En bekräftelsesida där studenter fullbordar en bokning efter att ha valt en tid genom att ange sitt namn och trycka ok
- Adminsida där en handledare kan ange sitt namn och därefter redigera sina tider (ta bort/lägga till), på denna sida syns även vilka som bokat en viss tid

Webbsidan hanterar två sorters objekt. Assistant och TimeSlot.
Ett **förslag** på struktur på datan är som följer:

- Assistant:
    - id: Int
    - name: String
- TimeSlot:
    - assistant_id: Int // Foreign key
    - id: Int
    - time: String // Tiden som visas för studenten
    - booked_by: String //Studentens namn

Bokningsflödet ska fungera som följande:

1. En student går in på bokningssidan och får se alla assistenter och alla lediga tider.
2. Studenten klickar på en tid som hen vill boka varpå två grejer sker:
    a. Studenten blir omdirigerad till bekräftelsesidan
    b. Med hjälp av websockets så blir övriga studenter som kollar på bokningssidan notifierade om att tiden är **reserverad** och just den tiden går inte att boka längre i deras UI:n.
3. Studenten på bekräftelsesidan har nu tre val:
    a. Fylla i namn och trycka på OK varpå man skickar till servern att tiden är **bokad** och användaren blir omdirigerad till bokningssidan.
    b. Vänta i 20 sekunder på en timeout varpå tiden blir fri igen och studenten omdirigerad till bokningssidan.
    c. Trycka på cancel varpå servern skickar ut till alla anslutna att tiden är ledig igen med hjälp av websockets.

Som hjälp så har vi producerat en kort mockup-video som ni finner på canvas.

Specifika krav:

- Ni måste använda ett SPA ramverk för er front-end. Exempel på ramverk är Vue, React, Angular, Aurelia, elm. *(Tips: Skelett koden är skriven i Vue så det är helt OK att utgå från skelettet.)*
- MVC (MVW, MV*, …) används ordentligt.
- Du har en förståelse för hur komponenterna hänger ihop och varför de är uppdelade på det sättet de är.
- När man navigerar till bokningssidan så hämtas datan om lediga tider med en AJAX-request och därefter så uppdateras listorna med nya/borttagna tider och vilka tider som är bokade via websockets.
- Vid redovisning ska ni ha skapat åtminstone 2 handledare med 4 tider vardera innan redovisningen, detta för att det ska gå snabbare att redovisa.
- Timeouten för en tid som är reserverad men ej bokad ska ske både på server och på den klient som har reserverat tiden. På klienten för att man enkelt ska kunna omdirigera studenten till bokningssidan om studenten av någon anledning inte hinner slutföra bokningen inom de 20 sekunderna. På servern för att se till att tiden blir ledig igen även om klienten inte hinner slutföra bokningen inom de 20 sekunderna.

Tips:

- Börja i tid, ni har fått 2 veckor för denna labb då vi vet att den kommer ta längre tid.
- Börja med att experimentera med det givna exemplet innan ni börjar med själva uppgiften. 50% av arbetet är att förstå hur saker hänger ihop.
- Utgå från exemplet och gör små inkrementella ändringar för att göra om exemplet till något som löser uppgiften. Det kan vara ganska svårt att debugga i början om man gjort stora ändringar.
- Dem flesta stora SPA ramverk har sina egna “dev-tools”, dessa kan vara väldigt hjälpsamma under utvecklingen. Om ni väljer att utgå från skelettet så kan ni hämta dev-tools för chrome eller för firefox (det finns även en standalone i fall ni avskyr chrome och firefox :P).
- Skumma igenom “Docs & Resources” i README.md i repot innan ni börjar.

Testsekvens:

1. Öppna 2 webbläsare (vanlig + inkognitoläge exempelvis)
2. Det under **a** görs i ena webbläsaren, det under **b** görs i andra
    a. Öppna bokningssidan
    b. Gå in i adminläget och lägg till en ny tid
3.
    a. Se att tiden som lades till dyker upp med hjälp av WS(websockets)
    b. Gå till bokningssidan
4.
    a. Börja boka en tid, du bör bli omdirigerad till bekräftelsesidan
    b. Se att tiden blir reserverad
5.
    a. Tryck på cancel
    b. Se att tiden är ledig igen
6.
    a. Boka en tid hela vägen
    b. Se att tiden är tagen
7.
    a. Gå till adminsidan, kontrollera att namnet på den som bokade är korrekt och ta bort den bokade tiden
    b. Se att den bokade tiden försvinner
8.
    a. Börja boka en tid, men låt bekräftelsesidan timea ut
    b. Se att tiden blir reserverad och därefter ledig igen
9.
    a. Börja boka en tid, stäng därefter webbläsaren utan att bekräfta tiden
    b. Se att tiden blir reserverad och därefter ledig igen

Anteckningar:
Ni får bygga mer funktionalitet än kraven i labben om ni vill (det resulterar dock inte i ett högre betyg, men kan vara kul och lärorikt och resultera i en glad assistent).

Det är så klart okej att sätta upp er egen miljö för att bygga både back-end och front-end. Skelettet ni får har inga byggsteg av anledningen att vi ansåg att det skulle öka tröskeln för ovana/nya js utvecklare. *(För den nyfikna https://cli.vuejs.org/)*

Frågor:

- Vad är ett promise? Varför används promises? Har ni använt några promises i labben?
- Vad är dependency injection? Varför används dependency injection? Har ni använt dependency injection i labben?
- Varför vill man använda websockets? Vad löser det för problem?
- Vad kan man göra om websockets inte finns (för gammal webbläsare eller något liknande)?

Bonus
Använd dig av en ORM/ODM och spara all data i en databas. Ni får med ett db.sql script i repot.

Frågor (Bonus):

- Lägg fram för- och nackdelar med en ORM mot “rå SQL”. När ska man använda den ena eller den andra?