Notes av Alen Gusinac

Jag valde att bygga sidan i samma ordning som en användare hade använt sidan, dvs bygga frontend, backend och databasen i den ordning som användaren klickar sig igenom. Hemsidan följer SPA-metoden och den mesta HTML koden skrevs i JavaScript för att sedan renderas ut via DOM:en.

Först skrev jag en enkel framsida som om man inte är inloggad visar en ruta där man kan logga in och skapa användare och valde då att skriva koden för att skapa uppkoppling och användare på databasen med INSERT query via en post router på backend. Då skapade jag även strukturen på databasen hur en användare skulle se ut, med id, namn, lösenord och sparar även tidsstämpeln på när användaren skapats för framtida bruk.

Med ett sätt att skapa användare skrevs nu koden för att matcha inloggningsuppgifterna mot användare i databasen genom SELECT query och vid korrekta uppgifter skickas sedan id och namn tillbaka från databasen och sparas i local storage.

Vidare kan man då logga in med sin skapade användare och hamna på en annan sida där man kan se en knapp för att logga ut, en knapp för att skapa nytt dokument och alla ens dokument man har sparat i databasen. Knappen för att logga ut är inte svårare än att den rensar användaren från local storage och kallar funktionen för att kolla om man är inloggad eller inte.

Ett tryck på “skapa dokument” knappen får två input fält för titel och beskrivning att dyka upp och även en knapp för att skicka uppgifterna, som då skapar ett dokument på databasen och svarar med att skicka tillbaka vilket id nummer det nya dokumentet har. Vid svar från servern kallar vi nu en funktion med det nya id:et som renderar det nya dokumentet i redigeringsformat.

När det var dags att inkorporera WYSIWYG redigeraren valde jag att använda TinyMCE och kollade igenom dokumentationen för att installera en basic redigerare. Ett utav mina bekymmer här var att jag skrev tinyMCE koden i roten på min javascript fil och eftersom rutan som ska redigeras inte fanns i HTML vid uppstart, fick jag till slut listat ut att jag måste flytta den koden till funktionen där jag skriver HTML för rutan, så att den initierar efter att rutan har renderats.

Efter att ha fått igång redigeraren var det dags att kunna extrahera det man har skrivit i den för att kunna spara värdet i databasen. Jag trodde först att det automatiskt sparades i value attributet på html rutan men fick bara ut undefined när

jag loggade ut det i konsolen. Efter att ha kollat i dokumentationen och plugga.tech filmen så insåg jag att jag aktivt måste skriva att den ska spara och att man kan göra det genom att kalla på spara metoden, som är inbyggd i redigeraren, varje gång en ändring skedde. Vidare sparas resultatet in i value attributet och jag kan då ta värdet och tillsammans med dokumentets id som jag sparade i ett data attribut i spara knappen kan jag nu skicka detta till databasen. Där skapade jag ett schema med id, userId, titel, beskrivning, value, datum det skapades och alla datum som det har redigerats. När jag fått ett godkänt svar från servern så anropar vi “visa dokumentet” funktionen tillsammans med id:et från spara knappen.

När vi sparar värdet lades detta in som TEXT i databasen vilket fungerade vid första anblick men det började spöka när jag skrev in olika specialtecken så jag omvandlade värdet till en BLOB och var då även tvungen att omvandla BLOB koden till ren HTML när den skulle visas eller redigeras i appen via Buffer klassen.

Jag kollade sedan på vilka funktioner man kunde lägga in i tinyMCE och letade specifikt efter hur man lade in textfärg och bakgrundsfärg för VG uppgiften. Jag insåg då ett bekymmer när värdet använde sig utav dubbel quotes och jag ändrade då queryn från dubbel quotes till singel och fick det då att funka. På onsdagens (12/4) föreläsning pratade vi om escape som jag inte hade hört om innan. Jag går då tillbaka till redigeringsläget och testar fler tecken vilket visar sig inte funkar fullständigt och får då vidare utreda djupare hur man ska använda sig utav escape. Det visade sig inte vara svårare än att kalla sin connection funktion tillsammans med metoden escape som då tar strängen man anger och lägger till bakåt snedstreck framför alla specialtecken. Här var jag även tvungen att ta bort citationstecknen från queryn för escape metoden lade till extra runt om.

Efter att ha fått till det positiva flödet i appen så loggade jag ut och började gå igenom ett negativt flöde av appen genom att ange information som inte ska funka att skriva in. Exempel på detta kan vara att skapa två användare med samma namn. Detta gav dock inget fel så då började jag med att göra namn till användare unika i databasen istället. Då fick vi ett felmeddelande från databasen som vi sedan hanterar i frontend genom att skriva ut att användarnamnet är upptaget. Samma gjorde jag sedan med inloggning fast att jag skrev in en användare som inte finns eller skickar ett tomt fält.

Mot slutet fixade jag lite styling på sidan med till exempel ny font och gör den responsiv med SaSS. Jag ville testa att kunna spara datum när man har redigerat ett dokument och sedan kunna skriva ut dessa datum i en separat flik men fick tyvärr ingen tid med detta. Jag tänkte via servern spara ett nytt datum o tid i ett JSON objekt och sedan sparat detta i databasen för varje gång spara routern triggades.

I övrigt tyckte jag att uppgiften var väldigt rolig och lärorik och ser fram emot nästa moment i kursen!