Projektrapport CMS

Sebastian Gustafsson

1. Vad har du gjort?

Jag arbetade ensam och har därmed programmerat både Strapi och Umbraco delarna själv, och följde punkterna i projektbeskrivningen. Både Umbraco och Strapi simulerar nyhetssidor. Umbraco har dock endast har en nyhetsartikel, medans Strapi kan innehålla många olika artiklar. Både Umbraco och Strapi använder "sebastian.gustafsson@utb.ecutbildning.se" som användarnamn/email och "Passw0rd!?" som lösenord för att komma åt adminfunktioner.

För Umbraco har jag skapat tre stycken sidor, "Home Page", "News Page" och "Newsletter Sign up Page". Alla sidor har properties som kan dynamiskt ändras av en användare via "Content" fliken på Umbraco. Jag har skapat fyra stycken Razor templates, en för varje sida samt en "Master" som innehåller allt runtomkring "Body-elementet", bland annat en navigeringsmeny till de tre olika sidorna. De andra sidornas templates renderas upp i Masterns body. Jag har skapat två stycken partial views, "ActivityAdvisor.cshtml" och "NewsletterPartial.cshtml". ActivityAdvisor hämtar information från en tredjeparts api, https://www.boredapi.com/api/activity, och visar upp resultatet i en ruta på "Home Page". NewsletterPartial innehåller ett formulär där användare kan skriva in en email för att "få uppdateringar" när dagens nyhet uppdateras. Formuläret lägger in en email i databasen om den är ny, annars meddelar sidan användaren att den inskrivna emailen redan är uppskriven för nyhetsbrev. Det finns rubriktext- och brödtext-element utspritt över alla sidor, och i navigeringsmenyn finns ett bild-element som visar en nyhetstidning.

För Strapi har jag skapat två Blazor WASM-sidor, "Index.razor" och "Submit.razor". Index är huvudsidan där användare landar först. Sidan innehåller alla nyhetsartiklar, och om man inte har rätt jwt-token kan man inte se några nyhetsartiklar. Den hämtar ut alla nyhetsartiklar via mitt Strapi-api, och visar upp dem för användaren. Submit innehåller formulär för att publicera ett grundläggande nyhetsinlägg. Båda sidorna renderar sitt innehåll på "Index.html". En navigeringsmeny, "NavMeny.razor", renderas också på Index.html och innehåller länkar till sidorna "Submit.razor" och "Index.razor". Jag har skapat två olika content-types, "Category" och "NewsArticle". NewsArticle innehåller text-fälten "title" och "text", bild-fältet "image" och relations-fältet "categories". Category innehåller text-fältet

“topic” och relations-fältet “news_articles”. Via Category och NewsArticle har programmet två olika api content-endpoints. Det existerar tre olika postinlägg av typen NewsArticle och fem postinlägg av typen Category. Jag har installerat det extra pluginet “Transformer” som gör att blazor-sidorna och Strapi kan skicka json-formatterade objekt till varandra. Sidorna inehåller rubriktext-, brödtext- och bild-element. Det finns två olika user-roller, “Author” och “Subscriber”. En Author kan skapa nya och uppdatera existerande artiklar, medans en Subscriber endast kan se artiklarna. I Strapi har jag även en användare per egenskapad roll. “Prenumeranten” har lösenordet “Passw0rd!” och rollen “Subscriber”, medans “Skribenten” har lösenordet “Passw0rd!” och rollen “Author”. Jag rekommenderar att hämta ut jwt-token för de användare som ska testas via postman och sedan klistra in den i programmet

2. **Hur funkar det tekniskt?**

Umbraco delen använder sig av Razor-templates för att visa upp sitt innehåll. Varje sidas specifika template är länkad till en Master-template som visar upp sidornas template-innehåll i body elementet via kommandot “@RenderBody()”. Den partiella vyn ActivityAdvisor visas upp på sidan “Home Page” via kommandot “@await Html.PartialAsync("ActivityAdvisor")”. ActivityAdvisor använder sig av “ActivityAdvisorClient.cs” för att hämta information från den tredjeparts api som används, https://www.boredapi.com/api/activity. Den partiella vyn NewsletterPartial visas upp på sidan “Newsletter Sign up Page” genom kommandot “@await Html.PartialAsync("NewsletterPartial", new CMSProjectUmbraco.NewsletterDTO())”. NewsletterPartial använder sig av “NewsletterMigration.cs” för att se till så databastabellen finns, annars skapar den tabellen, och “NewsletterSurfaceController” som kommunicerar med databasen så att email-adressen kan läggas in eller så att samma email-adress inte läggs in två gånger.

Strapi är uppbyggt som ett api där all information lagras, medans Blazor WASM delen fungerar som en frontend för användaren. Jag har hårdkodat in den jwt-token som identifierar användaren så att det är enklare att testa hemsidans funktionalitet. Denna token kan hittas i variabeln “jwt” i filerna “Index.razor” och “Submit.razor” och behöver ändras om en annan användares åtkomstmöjlighet ska testas. I strapi lagras alla “NewsArticles” och de kan kommas åt via api-endponten “/api/news-articles”. Varje “NewsArticle” har en titel (title), brödtext (text), bild (image) och de kan ha många olika “Categories” som beskriver nyhetsartikelns ämne, exempelvis “sports” och “world”. “Category” är den andra egenskapade content-typen och den innehåller alla olika kategorier en “NewsArticle” kan

vara. Content-typen har text (topic) och en relation till sina “NewsArticles”. Transformer pluginet möjliggör komunikation via json objekt mellan Blazor WASM och Strapi så formateringen blir rätt mellan alla delar av programmet.

3. **Varför har du kodat / gjort som du gjort?**

För Umbraco har jag till största del följt de inspelade föresläsningarna och modifierat programmet där jag behövde, men även följt guider och letat upp information på nätet. Jag delade upp så att alla sidor har sin egna template då det är enklare att hålla reda på information och göra varje sida mer unik. Att ha en Master-template underlättar också designen då exempelvis navigationsmenyn är samma för alla sidor. Att göra ActivityAdvisor till en partial tillåter mig i framtiden att flytta runt den mycket enklare, och det delar upp koden utan att behöva ge den sin egna sida. NewsletterPartial är en partial av samma anledningar, det blir helt enkelt enklare att bygga in formuläret där det behövs.

För att få Strapi att fungera har jag följt föreläsningarna och tittat på guider genom Strapis egna dokumentation. Jag valde att skapa en content-typ som innehåll alla kategorier för att enklare kunna tilldela varje nyhetsartikel flera olika kategorier. Anledningen till att användarens jwt-token är hårdkodad är på grund av att jag inte kom på ett bättre sätt att spara ner den på den tid jag hade på mig att göra projektet. Jag valde även att inte lägga ner mer tid på att skapa en bättre första sida där användaren landar på grund av att jag ville de andra delarna skulle fungera. Transformet pluginet använder jag för att det möjliggör kommunikationen på det sätt jag har satt upp mina sidor.

4. **Vad skulle du framåt sett bygga vidare om du hade mer tid och resurser?**

Jag skulle i umbraco lägga till så att ett mail skickas till användaren så fort nyhetsartikeln ändras. Jag skulle snygga till alla sidor mer och göra de mer responsiva när användaren klickar runt.

I Strapi hade jag gjort så att bilder och kategorier kan läggas till i formuläret för publicering av en ny artikel. Jag hade även gjort så att en användare utan jwt-token hamnar på en annan sida, och inte kan komma åt de sidorna “Index.razor” och “Submit.razor”. Jag skulle även vilja komma på ett bättre sätt att hantera den jwt-token som används, kanske låta användaren logga in på hemsidan och spara den token man får tillbaka i cache-minnet.