Slutrapport Snackis : ForumWeb

Kaan Gölcuk .NET20
kaan.golcuk@iths.se

Ett globalt forum som alla kan läsa eller registrera sig på, syftet med detta forum är att man fritt ska kunna diskutera olika typer av frågor. Användarna behöver inte fylla i för/efter-namn eller personnummer, utan räcker endast med mailadress och det synliga användarnamnet. Målet med projektet var att få fram ett forum som alla enkelt kan använda utan att behöva tänka på krångliga funktioner som kan finnas på olika forum. Under covid pandemin så har det framgått från “Internetstiftelsen” att människor har använt internet mer än vad de gjort tidigare och detta forum ska underlätta för individer att få svar på sina frågor och diskutera olika ämnen.

Projektet startade 17 maj 2021 och avslutades 16 juni 2021, inom den planerade tidsramen. Första projektveckan gick åt till att planera och strukturera upp arbetet. En av fel bedömningarna som gjordes under planeringen var databasen, den va mer komplex än vad jag trodde. Därav behövde jag lägga ner extra tid åt den, och gjorde om ett par gånger.

De tekniska kraven som krävs för att kunna köra applikationen är:
-Visual studio, helst .NET Core 5. Använder man de tidigare versionerna kan man behöva ändra lite i koden.
-Windows, OSX eller Linus.
- Nugget-paket:
    - Microsoft.AspNetCore.Indentity.EntitFrameWorkCore.
    - Microsoft.AspNetCore.Identity.UI.
    - Microsoft.EntityFrameWorkCore.
    - Microsoft.EntityFrameWorkCore.SqlServer.
    - Microsoft.EntityFrameWorkCore.Tools.
    - Microsoft.VisualStudio.Web.CodeGeneration.Design.

-Projektet finns på Azures egna servrar, där jag lagt in som Azure SQL Database.

Jag har 2 projekt med 2 databaser. I huvudprojektet ingår identity user, det vill säga att jag lagrar användarna i en databas som ligger i huvudprojektet. Det andra projektet är mitt API, där jag lagrar all innehåll i forumet tillsammans med ett UserId som finns i API’t. Så man kan se vilken användare som har gjort ett inlägg. API’t innehåller ett par Controllers för varje modell som är skapat i projektet. API’t är alltså ett Micro service till huvudprojektet. Huvudprojektet är skapad genom Razor Pages.

Dependency injection har använts ibland annat i huvudprojektet i startup.cs:
-CategoryGateWay
-SubCategoryGateway
-PostGateway

-CommentGateway
-ReportGateway
-MessageGateway

För att installera forumet behöver man hämta in det genom min Github profil. Jag har ett repository för API och ett för Huvudprojektet. Bägge dessa behöver laddas ner och därefter göra en “update-database” på bägge projekten. API’t behöver startas först för att kunna få tillgång på innehållet innan man startar huvudprojektet.
Om det är så att man vill köra programmet via Azure så behöver man först i Azure portal skapa en resource grupp, sedan en App service plan med två app services. I samma resource grupp behöver man skapa två Sql databaser. Ena databasen för huvudprojektet, och den andra för API’t.
I bägge projekten finns ett “appsettings-json” fil och där i kan man ändra Connection stringen om det så behövs. Kör man programmet på exempelvis Azure, så behöver man byta ut Connection stringen som man hämta från azure sql databas.

Endpoints

I mitt API har jag 6 modeller (classer), Categories, SubCategories, Posts, Comments, Messages och Reports.

För samtliga classer har jag controllers med endpoints. Alla controllers har fyra gemensamma endpoints, detta är Get, Post, Get{Id}, Put{Id}, Delete{id}. Dessa endpoints är densamma även för SubCategories, Posts, Comments, Messages och Reports.
Skriver exempel här utifrån Category, men dessa gäller för alla klasser som är nämnda ovan.
*Get hämtar ut alla Categories
*Post skapar en Category
*Get{id} hämtar en Category via id
*Put{id} ändrar en Category via id
*Delete{id} tar bort en Category via id

Utöver dessa har följande controller endpointsen:

-SubCategories har en endpoint som tar ut en Subcategory av en Category genom CategoryId: /api/SubCategories/CategoryId/{categoryId}.

-Post har en endpoints som tar ut en Post av en Subcategory genom subCategoryId: /api/Posts/SubCategoryId/{subCategoryId}

-Comments har en endpoint som tar ut en Comment av en Post genom PostId: /api/Comments/PostId/{postid}.

-Messages har en endpoint som tar ut en Message genom en UserId: /api/MEssages/UserId/{userId}

Funktionalitet

Vid inloggning använder jag mig utav Microsoft Identity. Detta gör så att en användare kan ändra sin mail, nickname, lägga till telefonnummer, ändra sitt lösenord och ta bort sitt konto.

Vid registrering utav ett konto så behöver man skriva in Email, Password och att jag lagt till så att man har ett “Nick name”. Efter att ett konto är skapat så kan man därefter lägga in en bild i sin profil. Bilden går även att ändra på samma sida. Har man en bild så visas detta i forumet.

Har man inte loggat in på hemsidan så kan man fortfarande se alla inlägg, men man kommer inte kunna skriva något.

Vid start så finns även cookies gällande GDPR samt en sida som heter “Privacy” där GDPR och cookies förklaras.

Har skapat 2 roller i projektet, “Admin” och “Basic”, Admin rollen skapas när man skapar ett konto med mailadress: admin@test.com. Basic rollen skapas på alla resterande konton automatiskt, oavsett vilken mail man använder (förutom admin mailen).

Admins behörigheter:
-Skapa kategorier och sub kategorier
-Ta bort kategorier och sub kategorier
-Kommentera poster och inlägg.
-Ta bort alla poster och inlägg.
- Skicka meddelande till alla användare
-Meddelanden visas i en separat sida
-Hantera rapporter som kommer in från användarna

Basic rollens behörigheter (alltså användarna):
-Kommentera poster och inlägg
-Ta bort endast sina poster och inlägg
-Skicka meddelande till alla användare
-Meddelanden visas i en separat sida
-Rapportera stötande inlägg för alla användare

Bristerna i programmet är bland annat att man inte direkt kan svara på meddelanden i meddelande fliken. Utan man behöver gå specifikt till en user i en post eller comment och skicka meddelandet.
Jag skulle kanske använda en annan hostingmiljö än Azure då gratis användarna har begränsad med använding. När tester gjordes för hemsidan så låste Azure sig och man behövde vänta mellan 12-24 timmar för att få igång sidan igen.

Frontenden kan förbättras, både på admin/user sidan själva forumets layout i de olika sidorna.

Arbetet med projektet har gått bra och varit väldigt lärolikt, det var lite brist på undervisning när det kommer till databasen. Det tog lite tid att lösa databasen för man var tvungen att kolla på videor och läsa på själv på nätet. Applicationen har potential att förbättras och därefter kunna användas på riktigt. Skulle man lägga lite mer tid på frontend och lägga till lite fler funktioner så skulle hemsidan vara utmärkt att lansera på riktigt.