# Webforum

1. Beskrivning:

Huvudansvarig för projektet: Patrik Stern med kontaktuppgift [patrik\_stern14@hotmail.com](mailto:patrik_stern14@hotmail.com)

Webforum är ett webbaserat diskussionsforum öppet för allmänheten. Applikationen är strukturerad på ett sådant sätt att ägaren av applikationen får ett automatgenererat admin inlogg att publicera huvud och subkategorier på sidan med. Nya användare kan registrera sig på sidan och sedan läsa, skapa, kommentera, gilla och anmäla poster samt samtalstrådar.

Icke registrerade användare kan fortfarande läsa vad som postats på forumet men inte skapa egna inlägg för andra.

På sidan finns även möjligheten att skapa kontakt med andra registrerade användare, genom att kunna kommentera och skapa sub-diskussioner av vardera post, samt besöka folks profiler för att lära känna dem bättre och skicka meddelanden via hemsidans chatt-funktion. I utvecklandet av produkten har inga specifika avvikelser skett utifrån vad som var planerat från start, det finns förbättringsområden för framtida vidare utveckling, men all funktionalitet för applikationens grundfunktionalitet utifrån uppdragsbeskrivningen finns implementerad.

1. Tidsöversikt:

Jag har själv programmerat och byggt alla delar för projektet och har inte haft någon arbetspartner med. Tidsmässigt har jag haft ganska exakt en månad på mig sedan den 17/5-17/6. Tidsmässigt har projektet fördelats tillsammans med annan pågående undervisning under förloppets gång och jag tror att ungefär 2/3 av en heltids arbetsmånad har lagts på projektet under denna tid. Med viss förhindring p.g.a. personlig sjukdom efter operation ca 10/6 – 15/6 vilket drog ner arbetstempot en viss del. Bortsett från sjuk perioden tycker jag att tiden har räckt till för att skapa den sortens forum som beskrevs i uppdragsbeskrivningen.

1. Tekniska krav:

Applikationen är byggd och lösningen ligger i ett VisualStudio .NET Core 5 RazorPage projekt vilket är lämpligt för att fortsätta vidare utveckla produkten just nu. Projektet bör vara driftbart på de största datoroperativsystem i och med att det är byggt inom .NET Core. Projektet innehåller C#, HTML, CSS och JavaScript samt EntityFramework för modulering och administration av MS SQL-databas kontexten. Databasen har byggts med code-first approach. Microsoft identity och applikationens entiteter är sammansatt till en gemensam databas för enkelhetens skull och är sammankopplad med FK för att kunna fördela innehållet enkelt över hemsidan. Projektets lösningar och databasen har körts lokalt via IIS under produktens utveckling men är samtliga idag publicerade på Azure för demonstration av produkten.

Nuget/bibliotek i projekten:

(WebForum)

* microsoft.aspnetcore.identity.entityframeworkcore\5.0.6\
* microsoft.aspnetcore.identity.ui\5.0.6\
* microsoft.entityframeworkcore.sqlserver\5.0.6\
* microsoft.entityframeworkcore.tools\5.0.6\
* microsoft.visualstudio.web.codegeneration.design\5.0.2\

(ConvoCollector)

* microsoft.entityframeworkcore\5.0.6\
* microsoft.entityframeworkcore.sqlserver\5.0.6\
* microsoft.entityframeworkcore.tools\5.0.6\
* swashbuckle.aspnetcore\5.6.3\

1. Applikationsstruktur:

Byggt främst i mvvm struktur genom RazorPages, innehållande ett par extra komponenter.

Beroenden

Applikationen har några beroenden som används och/eller injiceras in i lösningen via Startup.cs. Det första beroendet är till en SQL databas som kan migreras och administreras via EF i lösningen, mer om det under rubriken installation. Det andra beroendet är till ett publicerat API Endpoint och dess URI så att metoderna inom APIGateway kan användas i koden. De beroenden som är uppsatta som en direkt service i Startup.cs är, APIGateway och DBGateway klasserna, för att kunna använda injicering direkt till klasserna där dessa sedan vill användas.

Components

I koden används två så kallade Components(html komponenter), där DBGateway klassen som är uppsatt som en service i applikationen används direkt i vyn för att kunna använda klassens metoder direkt i html koden. De komponenter som finns är SiteStatistics.razor som används för forumstatistik och är implementerad på hemsidans alla publika sidor samt Boardoverview.razor, vilket är forumets startsida och endast används där.

Views(.cshtml files) och Models(.cshtml.cs files)

Applikationens Views och tillhörande PageModels är infördelade i två separata mappsystem. Den första samlingen ligger under mappen Areas/Identity/Pages och är de filer vi fått med för att redigera själva då vi bett om att få skriva över de vyer vi får med på köpet utav att använda Microsoft Identity i lösningen för registrering/inlogg.

Det andra mappsystem med vyer/modeller ligger som vanligt under Pages mappen som följer med per default av att använda RazorPages i .Net Core 5.

Mönstret för sammansättningen mellan back/front-end i vyerna följer ett generellt klassiskt mönster, där våra metoder för DB/API anrop har lagts i separata klasser för bättre renhet och översikt och vyernas modeller används för att köra ”code-behind” i server renderingen, för att sedan populera vyernas html innehåll. I vissa fall injiceras även klasserna innehållande metoderna direkt i vyerna och ofta blandas sätten.

Areas

Detta är huvudmappen som följt med då vi scaffholdat Identity till lösningen, inom den finns filerna för registrering/inlogg, Contexten som ärver av IdentityDbContext för att populera vår databas med rätt tabeller och ger oss möjlighet till olika konfigurationer/OnModeling inställningar. Sedan ligger där även vår User klass som vi också kan modifiera efter tycke. Under samma huvudmapp ligger även lösningens entiteter, alltså de klasser som kommer avbilda en modell för entityframework att bygga våra databastabeller utifrån.

Gateways

Som nämnt tidigare ligger databas och API anropen i separata klasser som är lagt under mappen Gateways, dem är självförklarande och är byggda som metoder för att anropa från andra platser i lösningen.

Microservices

I projekt mappen ligger en mapp döpt Microserv som innehåller API lösningsfilen för endpoints till databasen som används vid hämtning av poster i forumet och också skulle kunna byggas ut.

1. Installation:

För att konfigurera och installera appen efter nedladdning behöver du minst förhålla dig till dessa steg.

1. Lösningen innehåller EF migrationsfil att köra mot databasserver för en konstruering av databas och databas kontexten att använda inom lösningen. Om du vill ändra något i entitets klasserna eller användarklassen kan du göra det och sedan köra din egen migration, så länge rätt Nugets är installerade skall det funka, ibland kan man behöva ta bort den gamla migrations mappen för att migrationerna inte ska krocka med varandra på något sätt. Givet att du vill använda den migrationen som redan finns behöver du skapa en SQL databasserver som fungerar att köra inom EF, antingen lokalt eller molnbaserad och placera dess connectionString i appsettings.json filen. I filen står även kommenterat vart kontaktsträngen behövs placeras.
2. Update-database. Använd package manager consolen och ange kommandot update-database för att uppdatera din befintliga databasserver med den EF genererade databasen.
3. När detta har gjorts behöver du antingen publicera microservicesen ConvoCollector online eller köra lösningen lokalt. Detta för att kunna få en URI till ditt API endpoint som används vid hämtning av poster från databasen till hemsidan. När API lösningen körs och du därifrån kan få den behövda URI:n behöver du placera URI:n i metoden GetPosts inom APIGateway.cs filen, även där finns kommenterat vart det är menat att URI strängen skall placeras.
4. Vid skapandet av en ny databas utifrån lösningen så auto genereras ett admin för huvudadministration på sidan. Admin kan inte bli utlåst från hemsidan, vilket andra användare kan för att ge denna kontroll ifall något skulle missbrukas. Settings för admin blir satta under OnModeling() metoden i WebforumContext klassen vilket är DbContexten för hemsidan. Om lösenord, adminnamn eller annat behöver korrigeras görs detta därifrån innan uppdateringen av den nya databasen till servern görs. Även administratörens UserRoll samt övrigt, nyregistrerade användares UserRolls blir satta här. Administratören får en specifik roll som Admin i databasen och övriga användare som Users. Vilket både syns i skillnad på Id och i Userrolls av användarna för att kunna särskilja tillgång på hemsidan.
5. När dessa steg är gjorda, Microservicesen körs någonstans och är kontaktbar, en kontaktsträng till en kontaktbar databasserver med ny uppdaterad databas i finns tillhands så kan denna lösning köras.
6. För att publicera använd VisualStudios publiceringsverktyg. Den hittar du under Build i huvudmenyn, skapa sedan tex en webapplikation på Azure molnservice som är kopplat till ditt personliga konto, om du loggar in på VS med samma konto så har du den webapplikationen tillgänglig att publicera projektet i.
7. EndPoints

Som sagt finns det ett API som är byggt inom ett microservice projekt. Endpointen för denna API efter det har publicerats är <https://adressensomblirgiven.net/api/FetchConvo> . Endpointen är alltså /api/FetchConvo det är en Postmetod och vill ha en ThreadID alltså samtalstråds id i string format postat i kroppen för att köras. I lösningen tas detta hand om men se hur metoden är byggd i API Controllern + APIGateway klassen för att skriva eget JSON till API:et om ni vill provköra genom tex Postman.

1. Funktionalitet

* Registrering: Inom applikationen finns funktionalitet för användarhantering där .Net Identitys klassbibliotek används i lösningen. Den ger möjlighet för nya besökare att registrera sig på sidan, redigera sin användarinformation och logga in för att få full access för normal användare på sidan och kunna skapa samtalsämnen, trådar, poster, kommentarer och tilldela reaktioner på poster samt använda sin chatt.
* Postfunktioner: Administratören som auto-genereras vid skapande av databasen är den användaren som har huvudansvaret över hemsidans övergripande diskussion forum samt dess under ämnen vilket denna kan lägga till/redigera via tillhörande administratörs sida.

Resterande samtalsrum och tillhörande samtalstrådar kan skapas utav alla registrerade användare och funktion för detta hittas via knapp val på hemsidan. I vardera samtalstråd kan alla anslutna användare posta inlägg, kommentera på inlägg, gilla inlägg och anmäla inlägg.

* Bildfunktioner: I lösningen finns en bildhanterare, denna för att användare skall kunna lägga till en profilbild som visar sig på deras personliga profil, bredvid postinlägg i forumet samt bredvid chattmeddelanden inom chattfunktionen. Bildhanteraren i lösningen är en fileStream som kopierar ett exemplar av den lokalt valda bilden av användaren och placerar den i mappen img under mappen wwwroot med samma filnamn som den haft tidigare på användarens lokala dator. Detta filnamn kan sedan användas som URL sträng tillsammans med sökvägen till wwwroot/img/ för att placera bilderna där de skall vara inom html elementen. Samma upplägg av fileStreamer används vid registrering av användare samt vid infogande av bilder till postinlägg inom samtalstrådar.
* Chattfunktion: Varje registrerad användare får även en chattfunktion för att kunna skicka privata meddelanden till andra anslutna användare på hemsidan. Chattfunktionen innehåller inte några hjälpmedel för att hitta användare på hemsidan utan personen behöver ha det rätta och fullständiga användarnamnet till den som denne vill skicka ett meddelande till. Har man rätt användarnamn så kan man skapa en konversation genom att trycka på plustecknet i gränssnittet för tillägg av ny konversation och får sedan en konversation tilldelad med rätt person tillhörande chatten.
* Svärordskontroll: I lösningen finns en metod i klassen InternalFunctions. Metoden heter FaultWordChecker() och innehåller en metod som använder en klass med registrerade svärord i ett listformat att kontroller mot innehållet som antingen postas eller kommenteras inom samtalstrådarna. Om något ord i en sträng innehåller svärorden Jävlar, Fan eller Helvete så kommenteras dessa ord bort med \*\*\* för censurering. Ägare av systemet kan själva lägga till fler ord i listan efter önskemål.
* Reaktioner: Som nämnt ovan kan registrerade användare reagera på postade inlägg, antingen med att gilla inlägget, eller att anmäla det. Gillas ett inlägg visas det i gränssnitten med info om detta, anmäls ett inlägg byts posten ut mot informationstexten inlägget granskas efter anmälning… En anmäld post dyker sedan upp i en lista nåbar utav administratören på dennes administrationssida. Där kan administratören välja att ta bort inlägget om denne vill.
* Profil: Varje registrerad medlem får en personlig profil utifrån den bild/personuppgifts information de hängivit vid registreringen.
* API: Hemsidans poster hämtas som nämnt tidigare via en microservice innehållande ett API. API:et hämtar hem alla poster för den specificerade samtalstråden. Inom vardera samtalstråd anges även en dropdown meny med möjligheten att bläddra mellan vardera samtalstråd inom det specifika samtalsämnet.
* Sidstatistik: Medföljande på vardera sida finns komponenten SiteStatistics.razor i botten, vilket visar generell statistik för forumets utveckling. Medföljande relevant aktivitetsstatistik om postning, författare och datum följer även med på vardera tabell i användargränssnittet för snabb överblick av utvecklingen.

1. Bristfälligheter:

Utifrån mitt eget användande har jag upplevt lösningen som funktionell, bugg fri utifrån vad som kan ses inom spannet av mina egna tester och med ett relativt förståeligt UI & UX. Koden är utifrån min kunskap robust och välfungerande, men har säkerligen en del säkerhets och andra icke upptäckta bristfälligheter som någon senior utvecklare skulle kunna ha hittat. En uppenbar övergripande bristfällighet som jag dock inte hunnit lösa är tillgången till administrationssidan för hemsidan. Där anges inga inställningar för att endast ge åtkomst åt rätt sorts profil, vilket skulle kunna leda till att sidan missbrukas om den hittas av utomstående användare. Även en grundligare säkerhetsgenomgång borde gås igenom för säkerheten av hemsidan och dess användare.

1. Förbättringsmöjligheter:

Självklart finns många förbättringsmöjligheter med lösningen, några jag kan komma att tänka på är:

* Möjlighet att ta bort/ändra sina poster eller kommentarer som blivit publicerade på sidan.
* Möjligheten för administratören att ta bort en rapportering där någon anmält ett inlägg och inte bara ta bort inlägget.
* Tydligare UX för bildtillägg i registrering av användare + poster.
* Möjligheten att byta ut även sin profilbild i editeringen av ens personuppgifter.
* Chattfunktionen kan till viss del upplevas förvirrande, ett bättre UX där till att börja med för tillägg av en ny konversation med ny användare, endast ett plustecken signalerar nu för vart den funktionen är lagd och kan vara svår att förstå för en första gångs användare. Även det att man måste klicka på varje konversation man vill fortsätta skriva i inom konversationslistan på sidan, även om man precis lagt till en ny, då skulle man kanske kunna tro att det är ok att skriva direkt.
* Att använda alla hyperlänkar på hemsidan att länka vidare till rätt samtal/samtalstråd som visas i förhandsstatistiken. Idag används endast hyperlänken för användare inom spannet för att komma åt andra användares profiler.
* Sökfunktion för sökning av andra användare på sajten både i huvud gränssnittet och i chattfunktionen.
* Att Javascript funktionen för Visa mer.. Visa mindre… i kommentarsspalten listar den senast postade kommentaren i den ihop vecklade vyn och inte bara det som är överst.
* Fler förbättringsmöjligheter kan och bör absolut tas fram i framtiden för ett än funktionellare forum med gott UI/UX.

10.

Summering:

Webforum är ett internet baserat samtalsforum för allmänheten. För att få tillgång till hemsidans helhet behöver nya användare ansluta sig genom registrering och sedan inloggning. När detta är gjort kan denne användare skapa samtalsrum baserat på samtalsämnen som utges från huvudsidan vilka blivit tillagda av huvud admin. I samtalsrummen kan inloggade användare skapa samtalstrådar, posta text och bilder, kommentera och reagera på andra användares poster. Varje användare får även en egen profil att använda utifrån den registrerade personuppgiften och blir även tilldelade en chattfunktion som de kan använda för att skicka och ta emot privata meddelanden från andra forums användare.

Lösningen är byggd i ett .Net Core 5 RazorPage projekt vilket är lämpligt för fortsatt utveckling av produkten och hemsidan, API samt SQL databas är publicerade på internet via VisualStudios publiceringsverktyg.

För egen installation av produkten behövs en åtkomlig SQL databas server, att API:et i projektet vid namn microserv/CollectConvo är publicerat och att kontaktsträng till DB server samt URI till api:et är placerat på rätt ställen i koden (appservice.json för DB kontaktsträng och APIGateway klassen för API URI). I lösningen finns Nuget-paketen för Entityframework installerade, så för att konstruera en databas utifrån databas modellen i projektet migrationsmapp är allt som behövs att ange kommandot update-database i package-manager-consolen inom tex VS.

Givet att DB och API strängarna nu är på plats och har uppdaterats rätt så kan lösningen köras, antingen lokalt eller publicerad på annan server.