# **Snackis**

Anna Sjöberg, [asb4615@gmail.com](mailto:asb4615@gmail.com), 070 714 72 70

Rapporten innehåller fler sidor än önskat men det är mest tabeller och bilder och inte så mycket löptext.

### **Beskrivning**

Applikationen används för att skapa ett forum där medlemmar kan diskutera ämnen relaterade till forumets inriktning. Inlägg kan göras öppet för alla medlemmar, i grupper eller genom privata meddelanden. Applikationen är beskriven i beställarens kravspecifikation, se Appendix 1.

### **Tidsöversikt**

Utvecklingsarbetet pågick under perioden 20 maj 2021 – 16 juni 2021. Slutredovisning skedde 17 juni 2021.

### **Tekniska krav**

* Windows plattform
* Serverlösning: IIS
* Ramverk: .NET CORE 5 valdes eftersom det är senaste versionen av .NET ramverket och innehåller klassbibliotek vilka gör utvecklingsarbetet effektivt, till exempel Identity verktyget som även underlättar efterlevnad av GDPR.
* Databaslösning: MS SQL lämpar sig för de väldefinierade parametrar som används i databasens tabeller.
* Servermiljö: Azure erbjuder färdiga lösningar för att publicera web-applikationer.

|  |  |  |
| --- | --- | --- |
| **Nugetpaket** | **Funktion** | **Projekt** |
| Swashbuckle.AspNetCore | Swagger för dokumentering av API:er | PostsAPI, OffensiveWordsAPI |
| Microsoft.EntityFrameworkCore.  Tools | Verktyg för Nuget Package Manager i Visual Studio | SnackisApp, PostsAPI, OffensiveWordsAPI |
| Microsoft.EntityFrameworkCore.  SqlServer | Ger tillgång till SQL Server databas | SnackisApp, PostsAPI, OffensiveWordsAPI |
| Microsoft.AspNetCore.Identity.  EntityFrameworkCore | Funktionalitet för Microsoft Identity | SnackisApp |
| Microsoft.AspNetCore.  Identity.UI | Funktionalitet för Microsoft Identity | SnackisApp |

### **Applikationsstruktur**

Projektfiler

Web-applikation: SnackisSolutionVG.sln

Microservices: DiscussionsAPIVG.sln - API för Forum, OffensiveWords.sln - API för förbjudna ord

Projekten använder Dependency Injection. Beroenden inom och mellan projekten framgår i respektive projekts Startup.cs-fil samt IdentityHostingStartup.cs under SnackisApp.Areas.Identity i SnackisSolutionVG.sln. Notera att namnet på Connectionstring för databasen skall anges i både startup.cs och IdentityHostingStartup.cs i SnackisSolutionVG.sln.

SnackisAppSolutionVG är skrivet enligt programmeringsmönstret Razor Pages

### **Installation**

1. Spara Connectionstring för databasen i appsettings.json i de tre projekten.
2. Spara URL:er för Gateways i appsettings i SnackisSolutionVG.sln.
3. Kör kommandot “Update-Database” i Visual Studios Package Manager Console i alla tre projekten.
4. Publicera de tre projekten med hjälp av funktionen Publish i Visual Studio.
5. Starta projektet SnackisApp i SnackisSolutionVG.sln. Första gången projektet startas skapas användarrollerna “Admin” och “Användare” automatiskt. Ett ämne “Gruppmeddelande” som behövs för att ge gruppmeddelanden ett ämne skapas också automatiskt.
6. Skapa en adminanvändare från formuläret som visas.
7. Gå till admin-sidan och ge forumet ett namn.

### **Endpoints**

Här visas API:ernas GET-anrop, se Appendix 2 för komplett lista av anrop.

**Forum API**

|  |  |
| --- | --- |
| **Endpoint** | **Funktion** |
| GET /api/Forum | Hämtar alla forum med ämnen och poster |
| GET /api/Forum/{id} | Hämtar forum med angivet ID inklusive dess ämnen och poster |
| GET /api/Subjects | Hämtar alla ämnen med poster |
| GET /api/Subjects/{id} | Hämtar ämne med angivet ID |
| GET /api/Posts | Hämtar alla poster med svarsposter |
| GET /api/Posts/{id} | Hämtar post med angivet ID inklusive svarsposter |
| GET /api /Posts /answers /{parentId} | Hämtar alla svar för angivet postID |
| GET /api /Posts /subjectname /name | Hämtar alla poster för ämne med angivet namn |
| GET /api/PostImages | Hämtar alla postbilder |
| GET /api/PostImages/{id} | Hämtar postbild med angivet ID |

**Förbjudna ord API**

|  |  |
| --- | --- |
| **Endpoint** | **Funktion** |
| GET /api/OffensiveWords | Hämtar alla förbjudna ord |
| GET /api/OffensiveWords/{id} | Hämtar förbjudet ord med angivet ID |

### **Funktionalitet**

* Inloggning

- Hanteras med Microsoft identity

* Registrering

- Användarnamn, e-post, förnamn och efternamn är obligatoriska uppgifter. En personlig bild kan väljas.

* Admin

- Skapa Forum och diskussionsämnen

- Ändra behörighet hos användare

- Hantera anmälda inlägg , radera eller godkänna

- Skapa, ändra och radera förbjudna ord

* Användare

- Ladda upp en bild som kopplas till användarnamnet

- Skapa en personlig infosida

* Inlägg

- Skapa ny tråd i ett diskussionsämne

- Svara på inlägg

- Bifoga bilder

- Ge inlägg tumme upp eller hjärta

- Förbjudna ord ersätts med \*\*\*\*

- Anmäla inlägg med olämpligt innehåll

* Privata meddelanden

- Skicka privat meddelande

- Svara på privat meddelande

* Gruppinlägg

- Skapa grupp

- Bjuda in medlemmar till gruppen

- Ta bort medlemmar från grupp man skapat

- Acceptera aller avvisa gruppinbjudningar

- Skicka gruppmeddelande

- Svara på gruppmeddelanden

- Gå ur grupp

### **Brister**

Trädstrukturen för inlägg skapas med upprepning av kod för varje nivå vilket gör att antalet nivåer i trädstrukturen begränsas. Jag tror man borde använda recursion och/eller TreeView klassen i .NET men hade tyvärr inte tid att sätta mig in i det.

En del länkar från HTML-koden blir ibland fel när programmet körs. Jag skulle behöva lägga mer tid på att förstå hur relativa länkar fungerar.

Ämnet “Gruppmeddelande” är en nödlösning som beror på att kolumnen SubjectId för poster inte är nullable. Detta definierades tidigt i projektet och jag ville inte ta risken att ändra till nullable när jag lade till funktionen att skicka gruppmeddelanden som inte tillhör något ämne.

Vem som helst kan spara, ändra eller radera data i databasen via API:erna utan att vara autentisierad.

### **Förbättringsförslag**

Applikationen är skriven för att innehålla ett forum men eftersom databasen kan hantera fler samtidigt skulle en vidareutveckling för att hantera fler forum i samma applikation vara relativt enkel.

Gå igenom anropen till databasen för att se om de kan effektiviseras. Det skulle kunna minska svarstiden från Azure.

När ett inlägg tas bort raderas även alla svar nedströms i tråden. Man skulle istället kunna dölja texten i det raderade inlägget och låta resten av diskussionen finnas kvar.

### **Summering**

Snackisapplikationen följer de grundläggande kraven för uppgiften såsom att använda designmönstret Razor Pages, dependency injection, implementera Microservices och publicera projektet på Azure.

Arbetet har flutit på bra och det var roligt att bygga upp ett projekt med hjälp av vad vi lärt oss och förstå hur det fungerar.

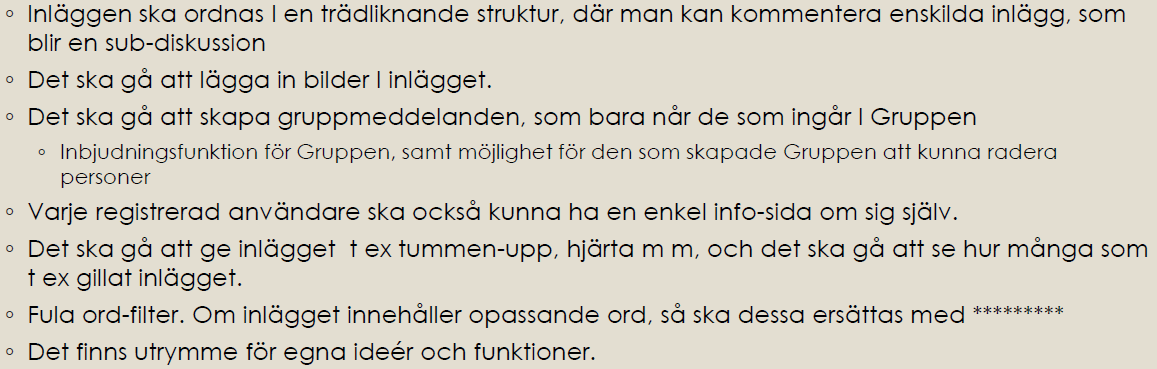
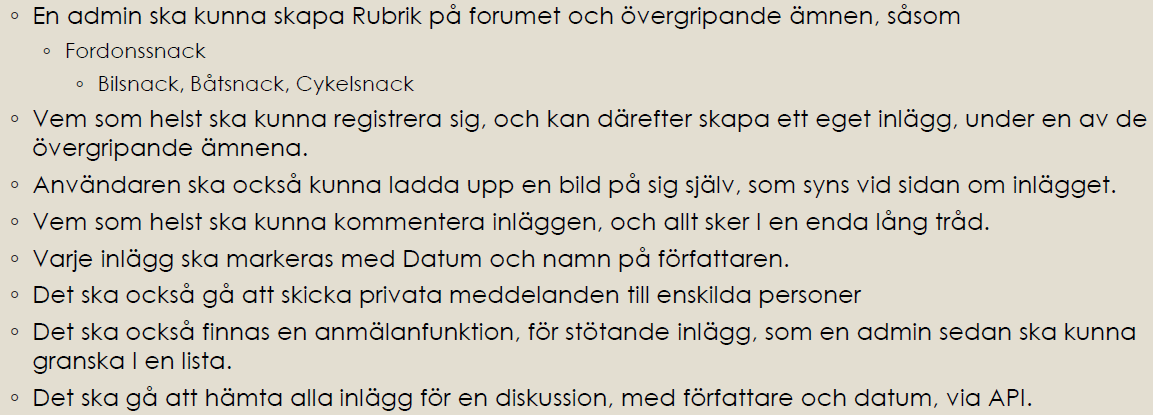
En svårighet var att inte kunna vara på plats och träffa kurskamrater och höra hur de tänker om uppgiften. Man går miste om tips som kommer fram när man pratar med varandra. Däremot har det varit en stor fördel att alla lektioner spelats in så man kunnat gå tillbaka och repetera och göra code along med möjlighet att pausa för att tänka.

Ibland har det varit svårt att göra en avvägning mellan att försöka skriva enligt good practice eller bara få till något som funkar.

Jag ser fram emot att använda mina kunskaper på LIA i höst.

**Appendix 1, s1**

# **Kravspecifikation för diskussionforum “Snackis”**

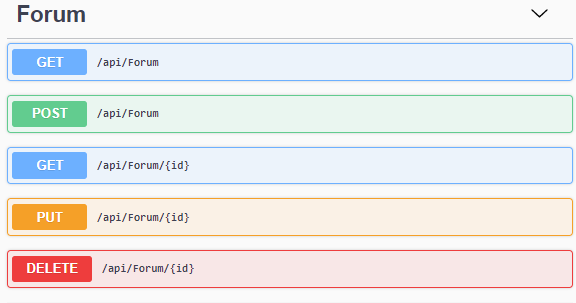


**Appendix 2, s1**

## **Forum API**

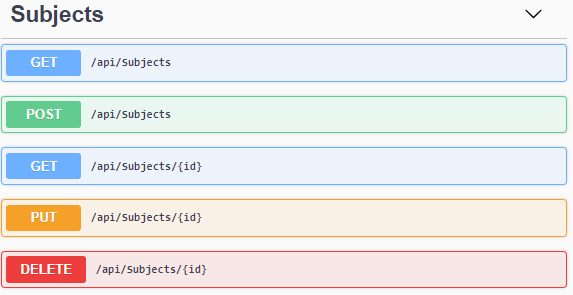
**Forum**

Parameter för POST och PUT: name



**Subject**

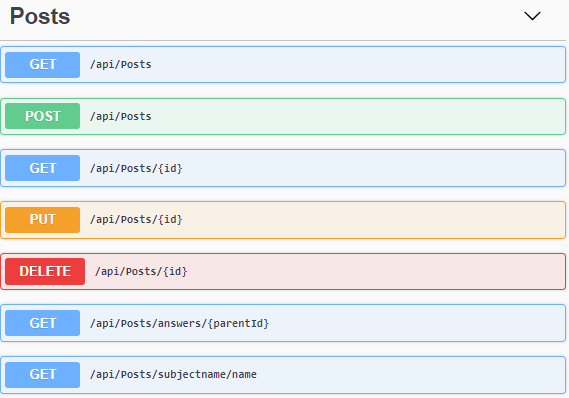
Parametrar för POST och PUT: forumId, name



**Appendix 2, s2**

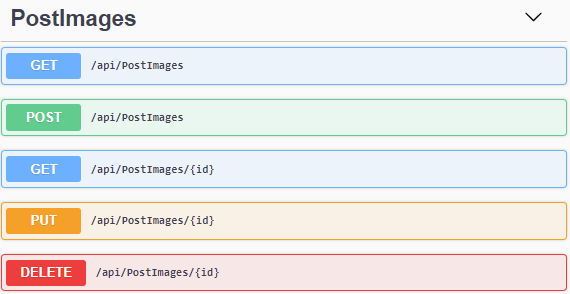
**Post**

Parametrar för POST och PUT: userId, subjectId, postId (ej obl), groupId (ej obl), date, title, text, numberOfLike, numberOfLove, isOffensive



**PostImage**

Parametrar för POST och PUT: postId, fileName



**Appendix 2, s3**

## **Förbjudna ord API**

Parametrer för POST och PUT: forbiddenWord

