**Dokumentation - projekt Geocaching!**

Projektet gick ut på att skapa en GUI-app med WPF som använder sig av EF Core för att hantera skapandet av tabeller och åtkomster till databasen där data angående Geocachen och Personerna lagras och hämtas.

Klasserna:

Vi började med att skapa klasser i VS vilka skulle hanterar skapandet och konfigurationen av tabellerna i databasen med hjälp av EF Core. Klasserna vi skapade är **Person, Geocache** och **FoundGeocache** (som är en jointabell) och som skulle bli till tabeller i databasen. Dessutom skapade vi klassen **AppDbContext.**

I **Person-klassen** la vi till alla variabler som behövs till den databas-tabellen. Vi la också in en one-to-many koppling till Geocache- och FoundGeocache-tabellerna. Dessutom använder vi oss av den inbyggda GeoCoordinate-klassen genom att deklarera den i både Person och Geocache-klasserna.

I **Goecache-klassen** la vi också till dom variabler som behövs till den tabellen och en one-to-many koppling till FoundGeocache-tabellen. Dessutom skapade vi en referens till Geocache-klassen som skall motsvara en FK i databasens Geocache-tabell.

I **FoundGeocache-klassen** (som blir en jointabell) la vi till en referens till Person- och en referens till Geocache-klasserna samt PersonID och GeocacheID som vi med Fluent API (i AppDbContext-klassen) gör till en composite key.

I både Person och Geocache-klasserna så gjorde vi kompletterande konfigurationer med hjälp av Data Annotations.

**AppDbContext-klassen** vi skapade ärver från DbContext och i denna klass definierade vi vilka tabeller som skall skapas av EF Core via DbSet< >. Vi skapar tabeller av Person, Geocache och FoundGeocache-klasserna. Här definierar vi även var databasen ligger och våra Fluent API’s ligger också här. Eftersom den inbyggda GeoCoordinate klassen innehåller mer än bara longitude och latitude så satte vi Ignore på övriga parametrar via Fluent API och definierade även kolumnnamnen till Longitude och Latitude.

När vi var klara med klasserna så gjorde vi Add-Migration och Update-Database via VS’s Package Manager Console för att med hjälp av EF Core skapa databasen med tillhörande tabeller.

Metoderna:

I huvudklassen MainWindow ligger sedan diverse metoder som skall hantera olika saker i programmet. Vi har försökt att strukturera det så tydligt vi kan och göra metoder som är mer generella och därför kan anropas från flera ställen i programmet. Som ett ex kan nämnas **BeginColorPersonPins** och **BeginColorGeocachePins** metoderna som sköter om färgläggningen av pinsen både vid inläsning från databasen, vid klick på olika pins, vid klick på kartan samt även vid inläsning från en fil.

Vår **UpdateMap**-metod uppdaterar kartan och pinsen utifrån det som är sparat i databasen. Denna metod anropas från flera ställen i programmet efter att ändringar har gjorts och sparats till databasen.

**OnMapLeftClick** återställer kartan och färgen på pinsen till hur det var ursprungligen när det lästes in från databasen/filen.

**BeginOnAddGeocacheClick** och **BeginOnAddPersonClick** lägger, vid högerklick på kartan, till ett nytt Geocache- respektive ett nytt Person-objekt och sparar till databasen asynkront.

Metoderna **OnLoadFromFileClick** och **OnSaveToFileClick** läser in från en text-fil respektive spara till en text-fil. OnLoadFromFileClick raderar först databasen innan filen läses in

Vi har i samtliga metoder försökt att göra alla databasoperationer (läsa och skriva) asynkrona men det finns inget sätt att testa om det fungerar eller inte. Vi har även gjort metoderna som placerar ut och färgsätter pinsen asynkrona mer för att vi själva skall kunna testa att uppdateringen av pinsen verkligen sker asynkront. Genom att ändra Delay-värdet på rad 207 samt 251 så kan man simulera en fördröjning varvid man ser att pinsen uppdateras asynkront.

Eftersom vi i projektgruppen av olika anledningar inte kunnat sitta så mycket och jobba tillsammans så delade vi upp projektet så att Novak skulle göra metoderna som lade in nya personer och geocache vid högerklick på kartan samt inläsningen från text-filen och sparandet till en text-fil. Övriga delar skulle jag göra inkl. dom båda VG-delarna. Vi hjälpte och rådfrågade varandra vid behov.

När vi läser in en fil så händer det att personerna och geocacherna inte sparas ner till databasen i exakt den ordning som dom lästes in i, men alla relationer inkl. hittade geocaches är rätt så detta påverkar inte funktionen. Vi har ”felsökt” detta och har kommit fram till att det har med ”database.SaveChanges();” att göra. När vi lägger till person- och geocache-objekten till databasen så görs detta i helt rätt ordning men när man sedan kör ”database.SaveChanges();” så kan ordningen kastas om. Detta blir mer ett ”estetiskt” problem då funktionen som sagt var inte påverkas.

Dom största utmaningarna var att jobba gentemot WPF eftersom detta är nytt för mig. Även att få till en fungerande inläsning av text-filen och fixa asynkroniteten vid alla databasoperationer var utmanande.