# **Projekt \_ Polygons Dagbok**

# **Dag 1-2**

## **Planering och projekt start**

* Skapade projektets grundstruktur.
* Designade en huvudmeny för konsolappen med följande alternativ:
  1. ShapeApp
  2. CalculatorApp
  3. Rock, Scissors, Paper GameApp

**Reflektion**: Inga större problem. Planerade kodstruktur för att säkerställa renhet och modularitet.

Jag beslutade att arbeta med en tydlig mappstruktur och följa DRY-principen. En lovande start.

För att sätta upp databasen installerade jag de nödvändiga nugget paketen via visual studio:

* Microsoft.EntityFrameworkCore.SqlServer
* Microsoft.EntityFrameworkCore.Tools
* Microsoft.Extensions.Configuration.Json

**Dag 3-4:**

[**Implementering av ShapeApp**](https://docs.google.com/document/d/1wYP4hq0QK28GqoPBQRdm29Ze_YvLZT3Kn-sLtkygZmk/edit?tab=t.0#heading=h.lckc7r1de69a)**:**

* Implementerade grundläggande funktioner för **Shapes**:
  + Valbara former: rektangel, parallellogram, triangel och romb.
  + Beräknade area och omkrets för varje form.

Implementerade en ShapeRepository och integrerade med AppDbContext.

* Att separera ansvar mellan handler och repository gav bättre kodstruktur.
* Lade till CRUD-operationer för **Shapes**:
  + Skapa, läsa, uppdatera och radera former.
  + Användaren kan se alla sparade former.

**Reflektion**: Utmaningar med att lagra beräkningar i databasen.

Varningar om "PendingModelChanges" vid migrationer.

Genomförde nya migrationer och uppdaterade databasen.

Fördjupad förståelse för Entity Framework-migrationer och modell-synkronisering.

# **Dag 4 - 5:**

* Implementerade **Spectre.Console** för att förbättra användargränssnittet:
  + Interaktiv huvudmeny med ASCII-art och färger.
  + Tabellformat för att visa sparade former.

**Reflektion**: Fokuserade på CRUD-funktioner i det nya gränssnittet.

Lade till och testade Update och Delete för Shapes.

Spectre.Console förbättrade användarupplevelsen avsevärt.

# **Dag 5 - 6:**

[**Implementering av CalculatorApp**](https://docs.google.com/document/d/1wYP4hq0QK28GqoPBQRdm29Ze_YvLZT3Kn-sLtkygZmk/edit?tab=t.0#heading=h.lckc7r1de69a) **:**

[CalculatorApp](https://docs.google.com/document/d/1wYP4hq0QK28GqoPBQRdm29Ze_YvLZT3Kn-sLtkygZmk/edit?tab=t.0#heading=h.hx9ewutg7kmy), [Refaktorisering och struktur](https://docs.google.com/document/d/1wYP4hq0QK28GqoPBQRdm29Ze_YvLZT3Kn-sLtkygZmk/edit?tab=t.0#heading=h.jitd3ir33j9v):

* Utvecklade **Calculator**:
  + Implementerade operationer: addition, subtraktion, multiplikation, division och modulus.
  + CRUD-operationer för beräkningar.

**Reflektion**: Problem med Repetitiv inputvalidering.

Det var givande att strukturera CRUD-operationerna effektivt.

# **Dag 6 - 7:**

[**Implementering av RspGame**](https://docs.google.com/document/d/1wYP4hq0QK28GqoPBQRdm29Ze_YvLZT3Kn-sLtkygZmk/edit?tab=t.0#heading=h.lckc7r1de69a)**App:**

* Implementerade **Rock, Scissors, paper Game**:
  + Användaren spelar mot datorn.
  + Historik och statistik visas.
  + Data sparas via en GameRepository.

**Reflektion**: Problem med databasmigreringar och using-direktiv.

Uppdaterade direktiv och beroenden.

Spelets interaktivitet och design känns mycket tillfredsställande.

# **Dag 7 - 8:**

* Refaktorerade CalculatorHandler:
  + Flyttade logik till specifika klasser.
  + Införde bättre felhantering för null-referenser.

**Reflektion**: Problem med långa metoder gjorde koden svår att underhålla.

Bröt ner klasser med Dependency Injection.

Refaktoriseringen har gjort koden mer hanterbar och skalbar.

# **Dag 8 - 9:**

* Skriv en detaljer till README-fil:
* Funktioner, teknologier, arkitektur och installationsinstruktioner.

**Reflektion**: Strukturerade information för tydlighet, delade in i sektioner.

README är nu en användbar resurs för nya användare.

# **Dag 9 -10:**

* Lade till stöd för soft-deleted Shapes
* Implementerade RestoreShape och hanterade återställning.

**Reflektion**: Problem vid hantering av soft-deleted poster.

Införde filter och förbättrad logik.

Soft delete-funktionen förbättrar användarvänligheten.

* Introducerade Dependency Injection (DI):
* Registrerade alla beroenden i DI-containern.
* Problemet med "Unable to resolve service"-fel.
* Registrerade rätt beroenden och använde AddScoped.

**Reflektion**: DI gör koden renare och mer testbar.

[**Sammanfattning och reflektion**](https://docs.google.com/document/d/1wYP4hq0QK28GqoPBQRdm29Ze_YvLZT3Kn-sLtkygZmk/edit?tab=t.0#heading=h.j77w1ojv780s)**:**

Det här projektet har lärt mig mycket om att arbeta med flera projekt samtidigt och hur man strukturerar deras ansvarsområden i olika mappar och klasser. Det här projektet har varit lärorikt, och jag har fördjupat mina kunskaper inom:

* Designmönster så som factory pattern och DI har varit väldigt lärorikt.
* Ren kod och refactoring
* Entity Framework och databashantering.
* Konsolappar och användargränssnitt.

Jag ser fram emot att applicera dessa erfarenheter på framtida projekt.