Projektledning & Agila metoder - Projekt Dagbok

Dag 1 - Starten av Projektet 7/5-25

Har planerat min Trello Board och försökt dela upp projektet till lagom stora bitar, features, i min product backlog. Detta för att senare kunna planera sprintar där jag kan hålla tidsplaneringen. Har planerat att den första sprinten ska vara lite mindre då jag vet att jag inte har så mycket tid just den här veckan.

Städade upp Trello boarden efter input från Richard, keep it DRY!

NOTE TO SELF: En del features verkar som att de kan vara lite stora och behöver därmed delas på, hur kan jag göra det på ett bra sätt så att jag inte delar upp tasks som behöver varandra till olika features?

Dag 2 - 14/5-25

Implementerade Spectre.Console och insåg att jag inte alls utnyttjade det till sin fulla potential i hotellappen. Kommer hädanefter försöka att alltid använda det till mina konsolappar, inte bara för att det ser snyggare ut utan främst för att det minskar kod i menyfunktionaliteten. Även när man inte har särskilt mycket tid att lägga på just utseendet i en konsolapp, så är den lilla extratiden man lägger ner definitivt värt det.

Dag 3 - 18/5-25

Försökte göra min app mer DRY så att det ska bli lättare att underhålla den och om jag hade velat i framtiden lägga till ytterligare projekt likt shapes, calculator och RPS. Insåg i slutändan att hur jag hade lagt upp min arkitektur från början inte fungerade så bra. Oavsett hur jag flyttade runt olika delar för att det skulle bli rätt så var det alltid någon circular project reference som hade behövts. Vid det här laget är det nog lättare att starta om från början för att strukturen ska bli rätt redan från början.

Dag 4 - Den stora omorganiseringen. 20/5-25

Nu använder jag AutoFac modules till DI istället för att det ska finnas bara en fil som registrerar allt. Nu registrerar varje module som finns i varje library bara sin egen typ, så bara services i ServiceModule osv. Mer DRY eftersom det som delas mellan apparna inte måste upprepas i appens ContainerConfig klass, istället registreras hela modulen. Det blir också mer scalable eftersom det blir mycket lättare att lägga till fler layers och registreringar allt eftersom att det behövs.

Dag 5 - Shapes 21/5-25

Implementerade Strategy Pattern för Rectangle. Jag kan verkligen se fördelen med att använda sig av det, även om setup tog lite tid att få till så kommer alla efterföljande shapes som använder sig av IShapeStrategy att gå mycket snabbare att få till. Framförallt om man vill lägga till ännu fler shapes som t.ex cirkel i ett senare skede eller i ett annat projekt. Strategy Pattern = mer DRY kod.

Dag 6 - 22/5-25

Lade till full CRUD. Fick göra en refactoring då jag från början hade skapat olika modeller beroende på vilken form det handlade om, men när jag kom till Update så insåg jag att det inte alls gick att göra DRY på något smidigt sätt. Bytte till att använda bara en model (ShapesModel) vilket också bantade ner all min tidigare skrivna kod då jag inte behövde ha lika många olika metoder för t.ex Read.

Pga lite ont om tid blev det bara Hard Delete, men om jag hinner lägger jag till soft delete innan inlämningen också (Fler sätt att skydda oss från dumma användare? Ja tack)

Det man inte har i huvudet får man ha i benen (fingrarna), men fördelen är att man får skriva mer kod och lär sig mer.

Note to self till framtida projekt - börja med att tänka på hur metoderna kommer att se ut för att veta när det är bäst att ha en större model eller flera mindre. Ska modellerna göra olika sak? Är mycket information samma? Finns det någon meningsfull fördel med att ha flera eller räcker det med en?

Dag 7 - Rock Paper Scissors 27/5-25

Den uppgift som hittills känts mest straightforward, bara Read på CRUD delen, själva spellogiken är inte alltför krånglig och eftersom jag använt mig av spectre prompts för användarens val så behöver jag inte heller oroa mig lika mycket för “dumma användare”. Jag valde att lägga in GamesWonAverage direkt i modellen så att när ett spel är avslutat och vinst ratio uträknat så sparas det i databasen. Vilket också betyder att det aldrig behöver räknas ut igen för just det spelet (vilket man annars skulle behövt göra i sin ReadAll metod).

Det kändes också som den lättare (och mer självklara) lösningen för mig då jag då inte behöver göra någon komplicerad metod för att räkna ut varje värde får varje spel i efterhand.

Jag tror att man väldigt lätt skulle kunna utöka spelet också om man t.ex skulle vilja lägga till lokala spelregler som rep (rep vinner alltid men bara första spelet i en bäst av x match annars förlorar det) eller Spock och ödla som i big bang theory versionen.

Dag 8 - Calculator 5/6-25

Ännu ett projekt som använder sig av strategy pattern, och eftersom det dels är det andra projektet som använder sig av det så har det gått smärtfritt att implmentera det. Detta för att

många delar blir väldigt likt Shapes, och om man inte gjort som jag att varje projekt är helt utomstående från main projektet så kan man säkert göra det väldigt DRY med att olika projekts strategies skulle kunna använda samma metoder. Men eftersom att jag har byggt som jag byggt så blir det en del snarlika metoder som upprepas i de olika projektmapparna i mitt ServiceLibrary. De olika mapparna i Services är uppbyggda så för att man ska kunna ta bort allt som inte behövs i de fall att man plockar ut bara ett av projekten. T.ex vill man använda Calculator så tar man helt enkelt bort Shapes och RPS mapparna från Services.

Utöver det så känner jag att det går fortare och fortare att skriva koden då sättet att skriva t.ex Spectre syntax blir liknande mellan projekten. Men också mycket för att ju mer kod som jag skriver in som liknar tidigare kod i min solution så verkar Visual Studio bli bättre och bättre på att ge bra förslag vilket också det speedar upp processen. Något jag uppskattar just nu då det inte är så mycket tid kvar till inlämning (grupp-projektet har fått gå före hela tiden), men jag tror också att jag skulle lära mig ännu mer om jag var utan alla förslag.

Dag 9 - 6/6-25

Lade till resterande CRUD operationer som var snarlika med Shapes CRUD vilket återigen underlättade eftersom jag inte behövde komma på hur jag skulle göra ytterligare en gång. Det är väl ändå den största fördelen med att vi gör många projekt, utan att vi egentligen tänker på det så bygger vi långsamt upp ett stort “boilerplate” bibliotek. Med funktioner och kod som bara behöver modifieras lite för att kunna användas i andra projekt. Varför uppfinna hjulet igen när det räcker att uppgradera det (allt eftersom att vi samlar på oss ny kunskap) och modifiera det (så att det går att implementera i det nya sammanhanget).

Slutlig Reflektion

Har det varit lite mycket att arbeta med 2 projekt samtidigt? Ja, en aning ibland. Har det varit orealistiska krav och förväntningar? Absolut inte. Jag tror att den här kursen i stort på många sätt speglat det arbetsliv som vi förhoppningsvis en dag kommer att få en plats i.
Det jag tyckt varit mest utmanande har varit:

- Prioritering: Skulle kunnat vara bättre, det har varit en utmaning att låta det här projektet ta tid från grupp-inlämningen och det har istället varit min fritid som tagit en smäll. Inte heller det har varit helt lätt att prioritera bort då min fritid inte bara tillhör mig utan främst min 2- åriga son som redan tappat en del av min tid i och med att han fick en lillebror. Att mamma då inte har tid för att hon måste plugga är inte alltid helt lätt att förstå och acceptera. Kan ärligt säga att min Shapes inlämning nog hade varit sen om sprintmötet hade skett på onsdagen och inte blivit flyttad till fredag (vilket för min del slutligen blev måndag). När man har många bollar i luften så kommer man till slut att tappa några och då handlar det bara om att veta vilka av bollarna som är av plast och vilka som är av glas.
- Tidsplanering: Har också varit en utmaning, det är svårt att ta med en stor refactoring eller en bebis som bara skriker och vill bli buren 2 timmar en kväll med dedikerad pluggtid i tidsplaneringen, men i slutändan så skickar jag ändå in inlämningen i tid så på något sätt har det ändå gått ihop.

Och mest givande:

- Att repetera: Det känns inte som att det var så länge sedan vi satt och jobbade i konsolen senast. Men när det hela tiden kommer ny information så glömmer man fort. Fördelen är också att det går snabbare att komma ihåg för varje gång man plockar upp det igen också.
- Att kunskap utvecklas och solidifieras: Jag kan se många sätt hur jag hade kunnat förbättra min Hotell-app, både i planering av arbetet samt strukturering av koden. Jag hade t.ex kunnat använda mycket mer spectre.console. Det känns bra att kunna känna att jag är en bättre programmerare idag än jag var när jag startade kursen.
- Kanban: Ett strukturerat sätt att planera och kartlägga sina branches och commits på ett överskådligt och tydligt sätt. Det har hjälpt mig att till stor del hålla mina commits små och branches relevanta.

Allt som allt ser jag fram emot att kunna utnyttja den kunskap och de tekniker som jag lärt mig under den här kursen i våra kommande kurser också!

Rut Frisk 2025