Vareskanning

Procesrapport

Mini-svendeprøve

TECHCOLLEGE

Elev:
Omidreza Ahanginashroudkoli

Skolepraktik:
Teglværket 9400 Nørresundby

Projektnavn:
Vareskanning

Uddannelse:
Datatekniker med speciale i programmering

Projektperiode:
19/Sep/2022 – 25/Nov/2022

Afleveringsdato:
21/Nov/2022

Fremlæggelsesdato:
25/Nov/2022

Vejledere:
Lærke Brandhøj Kristensen

Elev underskrift

_____

Vejleder underskrift

_____

Indholdsfortegnelse

Læsevejledning

Denne rapport er en af to, der hører til projektet Vareskanning. Projekterne og rapporter blev lavet af Omidreza Ahanginashroudkoli. Der findes to slags rapporter til dette projekt. Dette er procesrapporten og den anden er produktrapporten.

Produktrapporten indeholder projektets produkt og procesrapporten indeholder forløbet af hvordan produktet blev lavet.

Jeg vil personligt anbefale at starte med at læse produktrapporten først, for at få en forståelse af projektet herunder hvad den går ud på og dernæst kan man læse procesrapporten, hvori jeg beskriver hvordan jeg kom frem til det færdige produkt.

Indledning

I denne rapport vil jeg illustrere, hvordan jeg kom frem til mit slutprodukt, som er beskrevet i den relaterede produktrapport. Jeg vil dernæst illustrere projektets case og problemformuleringen og forklare hvordan jeg planlægger at færdiggøre projektet. Derfor vil jeg gå i detaljer og illustrere mine valg af teknologier, samt hvad alternativet er for de teknologier, som man kan bruge i stedet.
Til slut vil jeg gennemgå hvordan projektet kunne forbedres, hvis forløbet havde strækket sig over en længere periode og dermed kunne lave en mere uddybende konklusion på projektforløbet. Projektet er udviklet af Omidreza Ahanginashroudkoli i perioden 19. september 2022 til 25. november 2022 med vejledning fra Lærke Brandhøj Kristensen.

Case beskrivelse

Teknologi ændrer sig, hvilket gør, at menneskets liv bliver lettere ved at bruge det. Dette projekt hjælper med at skabe et vareskanningssystem, hvor det er let at bruge for alle.

En interessant del ved projektet er, at man får en liste af de skannede varer på sin mobile app og en Graph af dem, som viser antal skannede varer og tidspunktet for scanningen, så man kan danne et overblik over sine varer.

Målgruppen er de virksomheder og folk, som har brug for et system, der kan hjælpe dem med at få et overblik over de solgte varer. Et system som ikke laver menneskelige fejl og ikke har behov for fysisk dokumentation. Vareskanningssystemet kan bl.a. organisere og hjælpe med at se hvor mange varer der er solgt og hvornår.

Problemformulering

Hvordan kan jeg skanne vare og visualisere dem på en app?

Projektplanlægning

Estimeret tidsplan

Se Biag A: Estimeret Tidsplan.

Arbejdsfordeling

Jeg har valgt at planlægge mit projekt ud fra min rutine, hvor der startes fra backend og dernæst videre til frontend. Min estimerede tidsplan kan ses i bilag 1.

Jeg vil derfor starte med min Library model, da den bruges gennem hele projektet. Ved hjælp af min Library Model og Entity Framework kan jeg sætte min database op og køre. Herefter kan jeg begynde med på min API som mine forskellige funktionaliteter skal laves igennem. Næste skridt bliver at sætte min mikrocontroller op og gøre den klar til at sende information til min database.

Til sidst vil jeg begynde med at programmere min Cross-platform mobil app gennem hvilken brugeren vil have adgang til deres oplysninger og nogle andre funktioner.

Rapporterne vil jeg arbejde på løbende gennem hele forløbet, for at forbedre dem.

På den vedlagte tidsplan har jeg introduceret alle dage projektet løber over. Ud fra det skema vi har fået udleveret af vores vejledere, har jeg noteret antal arbejdstimer for hver del af det tværfaglige projekt ud fra de 10 uger.
I de grå weekenddage har jeg planlagt kun at arbejde, hvis jeg er bagud i tidsplanen eller har lyst til at undersøge noget, der kan være gavnligt for projektprocessen. Jeg har valgt at tage alle dage med i min tidsplan og ikke kun de planlagte opgavedage, da jeg synes det giver det bedste overblik over hvor jeg er i projektet, uden at det bliver forvirrende at se på fra tid til anden.

Det er mit håb at være færdig med den tekniske del af det tværfaglige projekt nogle dage før at vi skal aflevere, så kan jeg bruge denne tid på at rette rapporterne igennem.

Metode- og teknologivalg

I dette afsnit vil jeg diskutere hvorfor jeg har valgt de teknologier og om der er alternativer for dem, som kan gøre det samme eller er bedre.

Database

Jeg har valgt at bruge MSSQL database i mit projekt. Det har jeg valgt, da det ifølge mig giver mest mening og er noget jeg har mest erfaring og kendskab til. MSSQL har også en applikation, Microsoft SQL Server Management Studio (IDE), som kan hjælpe med at skabe et overblik over sine data og ændre i den.

Relationel vs. Dokument

Dog er der også en anden slags database, som man kunne have brugt herunder dokumentdatabasen. I en dokumentdatabase ligger dataene som enkelte entries uden relation til hinanden i stedet for rækker i tabeller. Dokumentdatabaser er typisk hurtigere end relationelle databaser, men de har ikke mulighed for at have relationer mellem data, hvilket kan være en ulempe. Af denne grund har jeg valgt relationel database.

Entity Framework Core

Jeg har sat min database op via code-first princippet og Entity Framework. Code-first er smart, fordi man i stedet for at bruge en masse tid på at lave en database og skrive en kode, der kan kommunikere med den, så kan man lave sine modeller, skrive kode for dem og generere sin database ud fra det. Den smarte del er, at man minimerer menneskelige fejl og sparer en masse tid.

Web API

API 'er er nødvendige for at bringe applikationer sammen, for at udføre en designet funktion bygget op omkring deling af data og udførelse af foruddefinerede processer. API fungerer som mellemmand og giver udviklere mulighed for at bygge applikationer, som kan interagere med hinanden. Fordelen ved at bruge API 'er er, at det hjælper med at have bedre data kvalitet gennemgang og oprydning. API kan understøtte hurtigere og lettere datamigrering og det giver større fleksibilitet.

Jeg har valgt at bruge API i ASP.NET Core, fordi det er det, som jeg har kendskab til. Jeg har valgt .NET Core frem for .NET Framework, fordi det er nyere og understøtter flere platforme.

ASP.NET vs. Python vs. Node.js

WEB API er et bedre valg til enklere og lettere services. WEB API kan bruge ethvert tekstformat, inklusiv XML, og er hurtigere end WCF.

Der er forskellige måder at oprette en API i Python. Den mest brugte er FastAPI og Flask. Selvom begge disse Python-frameworks er enkle og nemme at bruge, så har FastAPI fordelen, da den kompenserer for Flasks begrænsninger. FastAPI har en høj ydeevne, som samtidighed nemt kan understøttes og den tilbyder et enkelt og letanvendeligt afhængighedsinjektionssystem.

Node.js er et open-source framework, som man kan bruge til at eksekvere JavaScript på en server. Siden Node.js bruger JavaScript, så er den nem at bruge og den foretrækkes derfor af mange. Node.js tilbyder nem skalerbarhed på grund af sin asynkrone og begivenhedsdrevne natur.
Jeg har brugt ASP.NET, fordi det er det, som jeg har mest erfaring med og kendskab til.

Mikrocontroller

Mikrocontroller er en komprimeret mikrocomputer fremstillet til kontrolfunktioner i Embedded system til for eksempel robotter, hjemmeapplikationer og mange andre.
I dette projekt vil min mikrocontroller kunne skanne varerne med stregkode(barcode) ved hjælp af en skanner (barcode reader) og vise den på et Oled Display. På samme tid vil den sende data til vores API med hjælp af NodeMCU (ESP8266), som dernæst bliver gemt på databasen.

NodeMCU (ESP8266) vs. ESP32

ESP8266 er billig Wi-Fi mikrochip med indbygget TCP/IP-netværk software og mikrocontroller kapacitet.
Et alternativ kunne være ESP32, som er bedre og hurtigere version af den. Da min embedded del af mit projekt er ikke så kompliceret, er det ikke nødvendigt bruge ESP32.

Raspberry Pi

Raspberry Pi er en lille computer, som integrerer CPU'en og GPU'en i et enkelt integreret kredsløb. Grunden til at jeg har valgt at bruge en Raspberry Pi i mit projekt, er fordi min barcode skanner har brug for USB-PORT og da NodeMCU ikke har USB-PORT, så var alternativet at købe en USB-HOST. Dog fandt jeg ud af, at det ikke længere er en mulighed, da det tager lang tid at få. Jeg har derfor valgt at benytte Raspberry Pi til at skanne varer og sende deres barcode til vores NodeMCU. Ved at bruge Raspberry Pi får jeg mulighed for at vise mit kendskab til de forskellige slags mikrocontrollere og deres sprog.

MAUI

.NET Multi-platform App UI (.NET MAUI) er et open-source cross-platform framework til at skabe native mobile- og desktop apps med C# og XAML. Med .NET MAUI kan man udvikle apps, der kan køre på Android, IOS, MacOS og Windows fra en enkelt delt kodebase. Jeg har valgt at bruge .NET MAUI på grund af nogle fordele herunder mit kendskab til .NET MAUI og de sprog, som den opererer på herunder C# og XAML.

Maui vs. Xamarin vs. Flutter

I stedet for .NET MAUI kunne man have brugt et udvalg af andre fronthend Frameworks. Grunden til at jeg har valgt .NET MAUI er, at den understøtter det seneste .NET 6 og det er det samme som jeg bruger til min Model Library og min API. .NET MAUI er beregnet til at forene og erstatte teknologier som WPF, UWP eller Xamarin.

Man kunne også have brugt Xamarin Froms, hvilket også er et open-source. Dog er Xamarin Forms ikke lige så god og opdateret som .NET MAUI, siden den kører på .NET Framework. Hvis jeg skulle have brugt Xamarin Froms, så var jeg nødt til at ændre det hele til .NET Framework.

Man kunne også have brugt Flutter til at lave sin Cross-platform mobile app. Flutter er også et open-source. Udvikling i Flutter er meget hurtig og effektiv, da den giver mulighed for at foretage øjeblikkelige ændringer på app'en, finde bugs og rette dem. Ulemper med Flutter, selvom den er meget populær, er at den ikke har eksisteret længe nok til at have en enorm ressourcebase og de apps som blev lavet, kan være vægtige. Flutter bruger Dart sprog, hvilket ikke er særlig populært og jeg har mindre erfaring indenfor det område.

Sprog

C#

Projektets primære sprog er C#. Min erfaring spiller en stor rolle i det valg. De teknologier jeg har valgt til mit projekt kører også på C#.

Arduino

Jeg har brugt Arduino sprog som er blanding af C/C++ til kode min mikrocontroller NodeMCU.

Python

Hovedsproget for kode i Raspberry Pi er Python. Python kommer forudindstillet på Raspberry Pi. Python er alsidig og brugervenlig, derfor valgte jeg at kode i Python.

Xaml

Min Cross-platform Application Maui bruger Extensible Application Markup Language (XAML) til sin frontend.

IDE

Integrated Development Environment (IDE) er en software applikation, der hjælper programmer med at udvikle softwarekode effektivt. Her er en liste over de IDE'er jeg har brugt for at udarbejde mit projekt.

Visual Studio vs. Visual Studio Code

Jeg har brugt Visual Studio til at sætte mit projekt op. Jeg har sat op min API, min mobile app, mikrocontroller og kodning af min database. Visual Studio tilbyder en masse funktioner og teknologier uanset hvilket sprog man bruger, der kan hjælpe programmer med at spare en masse tid med at skrive ren kode, teste og fejlfinde dem.
At have alle de funktioner kommer med en ulempe. Visual Studio er en tung software at installere og kompiler kode. Det vil derfor nogle gange gå ned og være frustrerende at arbejde med.

Alternativt til Visual Studio kunne være Visual Studio Code, men den er mere begrænsende i funktionerne, fordi den er en letvægt version af Visual Studio. Jeg har derfor valgt Visual Studio, idet jeg har mere kendskab til det.

Microsoft SQL Server Management Studio

Der er mange fordele ved brug af Microsoft SQL Server Management Studio. Nogle af dem er nemme at installere. Enhanced Performance med indbygget gennemsigtig datakomprimering og krypteringsfunktioner tilbyder SQL-serveren forbedret ydeevne og den er meget sikker.

Jeg har valgt at bruge Microsoft SQL Server Management Studio til at interagere med min database, fordi det er den jeg har brugt i mange år og har kendskab til.

Væsentlige elementer fra produktrapporten

Hvis jeg skal vælge en del af projektet, som jeg synes er mest spændende, så vil jeg nok vælge *Overordnet Arkitektur* i produktrapporten.

I dette afsnit beskriver jeg hvordan de forskellige dele af projektet kommunikerer sammen og det vil jeg mene er det mest interessante i hele projektet.

Afgrænsning

Hvis jeg havde mere tid og mulighed for at planlægge mere nøje, så ville jeg have inkluderet mere funktionalitet i både frontend og backend, så projektet ville have været mere komplet og ikke så meget som et proof of concept.

For at dette kunne lade sig gøre, skulle jeg have mulighed for at logge ind med forskellige brugere.

Man kunne også have flere endpoints i API, som giver brugeren mulighed for at ændre eller få fat i et enkelt produkt, salg eller lave en søgefunktion.

Hvis jeg havde mere tid, ville jeg have implementeret DTO(data transfer object), da det ville have været nyttigt i min API.

Realiseret tidsplan

Min realiserede tidsplan er vedlagt som Bilag 2. den ligner ikke meget den estimerede og det er der forskellige årsager til. Jeg havde få ideer til at gøre min estimerede tidsplan så nøjagtig og realistisk som muligt. Jeg lavede min tidsplan baseret på det skema, som lærerne havde udleveret og jeg indså derfor kort efter, at jeg var nødt til at foretage justeringer.

Jeg planlagde først at starte med mine rapporter og arbejde på dem igennem hele forløbet, så jeg kunne implementere noget nyt hver gang. Mens det var i gang, begyndte jeg at arbejde på mine NodeMCU og Raspberry Pi, som jeg havde planlagt at gøre. Næste skridt var at arbejde på min databaseside af projektet og få det til at køre. Jeg havde nogle problemer undervejs med min database relationer omkring de krav vi skulle opfylde.

De store ændringer var med min API og min frontend. Jeg besluttede mig for at starte med min API før jeg gik i gang med min frontend. Jeg indså, ved at gøre dette, at det ville gavne arbejdsgangen og give mig mulighed for at teste og sikre, at der var kommunikation igennem min projekter.

Konklusion

Gennem projektet har jeg udviklet en Proof Of Concept løsning til det problem, som jeg forklarede i min case beskrivelse og fastsatte i min problemformulering.

Jeg har lavet en løsning, der bruger to mikrocontroller, et API med tilhørende database og en Cross-platform mobil App(Maui), til at lave et system, hvor brugeren kan skanne produkter, hvor det bliver gemt i en database og kan hentes på en mobil app.

Projektet har været med til at give en ide om hvordan det er at udvikle et større IT-produkt og giver mulighed for at øve sig på at følge efter en tidsplan og tilpasse sig til nye krav. Dette tværfaglige projekt har også været en god øvelse til svendeprøven, som vi skal op til på H6.

Fordelen med projektet har været at undersøge og finde den bedste teknologi til hvor man kan optimere en masse tid og performance.

(Bilag)

A. Estimeret Tidsplan

Figur 1 Biag A: Estimeret Tidsplan

Vareskanning

SIMPLE GANTT CHART by Vertex42.com

https://www.vertex42.com/ExcelTemplates/simple-gantt-chart.html

Project Start: 19.09.22

Display Week: 1

OPGAVE	ANTAL TIMER	FREMSKRIDT	START	SLUT	RIGTIG START	RIGTIG SLUT
Rapporter	**48**		**19.09.22**	**24.11.22**	**19.09.22**	**21.11.22**
Opsætning	4	100%	19.09.22	19.09.22	19.09.22	19.09.22
Case beskrivelsen & Problemformulering	5	100%	23.09.22	23.09.22	23.09.22	23.09.22
Kravspecifikation & Tidsplan	8	100%	26.09.22	28.09.22	26.09.22	28.09.22
Processrapport content	14	100%	30.09.22	24.11.22	30.09.22	21.11.22
Produktrapport content	14	100%	31.10.22	24.11.22	31.10.22	21.11.22
Forbedring	3	100%	08.10.22	24.11.22	08.11.22	21.11.22
Web API	**27**		**07.11.22**	**22.11.22**	**18.10.22**	**27.10.22**
Opsætning	2	100%	07.11.22	07.11.22	18.10.22	18.10.22
Datahåndtering	11	100%	07.11.22	14.11.22	19.10.22	21.10.22
CRUD funktion	8	100%	17.11.22	17.11.22	26.10.22	27.10.22
Sikkerhed	6	100%	22.11.22	22.11.22	27.10.22	27.10.22
Database	**14**		**17.10.22**	**27.10.22**	**17.10.22**	**27.10.22**
Opsætning	3	100%	17.10.22	17.10.22	17.10.22	17.10.22
Datahåndtering	8	100%	17.10.22	17.10.22	17.10.22	17.10.22
Sikkerhed	3	100%	27.10.22	27.10.22	27.10.22	27.10.22
Embedded	**37**		**05.10.22**	**25.10.22**	**05.10.22**	**25.10.22**
Opsætning	3	100%	05.10.22	05.10.22	05.10.22	05.10.22
Skanner & Node MCU	17	100%	10.10.22	20.10.22	10.10.22	20.10.22
Raspberry pi (Linux)	6	100%	03.11.22	03.11.22	20.10.22	21.10.22
Datahåndtering	11	100%	21.10.22	25.10.22	21.10.22	25.10.22
Cross-platform App	**18**		**12.10.22**	**02.11.22**	**26.10.22**	**18.11.22**
Opsætning	3	100%	12.10.22	12.10.22	26.10.22	26.10.22
Datahåndtering	6	100%	18.10.22	18.10.22	02.11.22	02.11.22
Data præsentation	6	100%	26.10.22	26.10.22	17.11.22	17.11.22
Design	3	100%	02.11.22	02.11.22	18.11.22	18.11.22

Insert new rows ABOVE this one