|  |
| --- |
| [Forfatterens navn]  01-12-2023 |

|  |
| --- |
|  |
| Procesrapport |
| CitizenTaxi |



|  |
| --- |
| **Elev:**  Daniel Simonsen |
| **Firma:**  Skoleoplæringscenter |
| **Projekt:**  CitizenTaxi |
| **Uddannelse:**  Datatekniker m. speciale i programmering |
| **Projektperiode:**  06/11/2023 – 07/12/2023 |
| **Afleveringsdato:**  01/12/2023 |
| **Fremlæggelsesdato:**  07/12/2023 |
| **Vejledere:**  Frank Rosbak  Lars Thise Pedersen  (Per Madsen)  (Thomas Vrangbæk Thomsen) |
|  |

# Titelblad



Techcollege Aalborg,

Struervej 70,

9220 Aalborg

# Læsevejledning

Jeg anbefaler at læse denne rapport først.

// TODO: Ord-/fortkortelsesliste

Indholds fortegnelse

[Titelblad i](#_Toc150432571)

[Læsevejledning 1](#_Toc150432572)

[Indledning 3](#_Toc150432573)

[Case beskrivelse 3](#_Toc150432574)

[Problemformulering 3](#_Toc150432575)

[(Afgrænsning) 3](#_Toc150432576)

[Projektplanlægning 4](#_Toc150432577)

[Estimeret tidsplan 4](#_Toc150432578)

[Arbejdsfordeling 4](#_Toc150432579)

[Metode- og teknologivalg 4](#_Toc150432580)

[Realiseret tidsplan 5](#_Toc150432581)

[Konklusion 5](#_Toc150432582)

[Diskussion 5](#_Toc150432583)

[Bilag 6](#_Toc150432584)

[Logbog 7](#_Toc150432585)

# Indledning

Borgerne i Region Nordjylland klager over den lange ventetid, når de ringer til Flexsygehustaxa. De vælger derfor at ringe til lægesekretærerne, og bede dem om at bestille en taxa for borgeren.

Lægesekretærerne kan derefter ringe via internt nummer og få taxaen booket til borgeren hurtigere, end hvis borgeren selv havde ringet, dog tilføjer det endnu en opgave hos lægesekreterne, som kunne have været undgået, hvis borgeren selv havde bestilt taxaen.

# Case beskrivelse

## Problemformulering

Hvordan kan man få borgerne til selv at bestille deres Flexsygehustaxa uden hjælp fra lægesekretærerne og uden den unødvendige lange ventetid, og kan man ellers formindske lægesekretærernes ekstra arbejde?

## Afgrænsning

For at gennemføre projektet som en prototype løsning til Flexsygehustaxas lange ventetid, afgrænser jeg bl.a. sygehus adgang (f.eks. bruge deres eksisterende login oplysninger) og rigtig taxa bestilling (hvis de har et offentligt api til rådighed).

Produktet har egne brugere og simulation af taxabestilling af denne årsag.

# Projektplanlægning

## Estimeret tidsplan

Billedet nedenunder viser min estimerede tidsplan i form af Gantt.

A screenshot of a computer

Description automatically generated

## Arbejdsfordeling // TODO

Hvordan har du fordelt arbejdet på programmet og rapporterne.  
Har du fyldt din tidsplan?  
Er der noget du har brugt for lang tid på, og hvordan har det påvirket din planlægning?

# Metode- og teknologivalg // TODO: Rephrase

**Backend**

Til backend bruger jeg ASP.NET. Igennem uddannelsen har jeg fået meget erfaring med projekttypen, da den følger god mappestruktur og tilbyder ”Swagger”, som er et redskab til at teste backendens controllere, hvilket sparer en masse tid, når endpointsene bruges af frontenden.

Alternativt kan jeg vælge at bruge Node.js. Jeg er stærk i JavaScript og har tidligere også lavet nogle backends i node.js. Dog er jeg stor fan af ”controller” mønsteret, der bruges i ASP.NET, som node.js ikke benytter som standard.

**Database**

Til database bruger jeg Microsoft SQL (MSSQL). Microsoft SQL er en relationel database, hvilket er vigtigt for projektet, da der er flere relationer i spil – bl.a. mellem borger og borgerens notat samt bestillinger.

Alternativt kan jeg bruge MySql eller MariaDB, som andre bud på relationelle databaser. Det ville i sidste ende ikke gøre nogen forskel for projektet, men jeg bruger MSSQL, da jeg i min backend bruger EntityFramework NuGet pakken (til at håndtere min database for mig), som er opsat til MSSQL som standard.

Hvis jeg vælger en non-relationel database som f.eks. MongoDB, PouchDB eller CouchDB, vil jeg miste mine relationer mellem borger og notat samt bestilling. Jeg kunne også have blandet objekterne sammen, så de alle gemmes under den samme borger, men ren datastrukturmæssigt ville det blive noget rod.

**Frontend**

Til frontend bruger jeg React i TypeScript. Jeg har efterhånden god erfaring med frameworket, men overvejede også at bruge ASP.NET fra C# som frontend. Jeg konkluderede, at fordi mit design brugte modals, en step-by-step formular og ”conditional rendering”, ville React være en bedre løsning, da man er tættere på klienten end ASP.NET, selvom jeg skal oversætte mine C# modeller til TypeScript.

Alternativt kunne jeg bruge Blazor i C# i stedet. Blazor er tættere på klienten end ASP.NET. Dog har jeg aldrig brugt Blazor før, og jeg tænker ikke, at jeg har tid nok til at lære et nyt framework at kende.

**IDE**

Som IDE bruger jeg både Visual Studio og Visual Studio Code. Jeg bruger Visual Studio til at starte min backend, og Visual Studio Code til at starte mit frontend.

Visual Studio er bygget til at køre C# programmer, og Visual Studio Code giver en bedre oplevelse, når det gælder frontend udvikling. Jeg kunne nok sagtens nøjes med at bruge én IDE, men ikke uden at gøre det unødvendigt svært for mig selv. I Visual Studio behøver jeg bare at trykke på ”Start” for at starte mit API og Swagger, hvorimod Visual Studio Code kræver noget eksternt konfiguration. Samtidig bruger jeg Visual Studio Code til frontend, da syntaksen i Visual Studio Code er mere læseligt, end hvis jeg havde brugt Visual Studio.

Alternativt kunne jeg bl.a. bruge JetBrain’s ”Rider” IDE, som jeg har hørt meget godt om, når det gælder udvikling i C#. Visual Studio er et meget tungt program, og håndterer ikke computerens resurser lige så godt som Rider. Dog har jeg aldrig brugt IDE’en før, og jeg har ikke tid til at finde ud af, hvordan man sætter et API projekt op og stadig bliver tilbudt Swagger.

**Design**

Til design af frontenden bruger jeg Figma. Figma er et program, som designere bruger til app/hjemmeside design. Jeg ville gerne designe min hjemmeside, før jeg gik i gang med at kode, selvom jeg ikke har meget erfaring med Figma.

Alternativt kunne jeg selvfølgelig have tegnet noget på et stykke papir, eller brugt et paint program som Microsoft Paint eller Paint.NET, som jeg allerede har på min computer. Jeg ville dog hellere bruge et program, som er bygget til UserInterface design. Jeg kunne også have valgt at bruge Adobe XD, som er endnu et populært værktøj indenfor design verden, men jeg valgte at bruge Figma, da Figma’s gratis version har de minimale features jeg har brug for, for at lave et godt design.

Havde jeg brugt et paint program, skulle jeg tegne og farvelægge hele tiden og forestille mig brugerinteraktion, hvorimod Figma også kan lave funktionelle prototyper og komponenter udover at bare tegne et design. Det samme kan Adobe XD, men for at få adgang til XD, skal jeg have en licens, som vi ikke har som datateknikker elever.

**Version- og opgavestyring**

Til version- og opgavestyring bruger jeg Github. De fleste af mine programmeringsprojekter ligger allerede på Github, og jeg har gennem uddannelsen brugt servicen meget. Fornyeligt er jeg begynd at bruge deres ”projekt” feature, som tilbyder opgaveplanlægning i form af Github’s ”issues”.

Alternativt kunne jeg bruge Azure DevOps. Som Skoleoplæringscenter-elev benytter vi ofte Azure DevOps til både versionsstyring og opgavestyring. Jeg vælger dog i stedet at fortsætte med at bruge Github, da mine projekter i Azure DevOps er tilkoblet min skolekonto, hvor jeg helst vil beholde projektet efter min skolekonto er blevet slettet.

## Realiseret tidsplan // TODO

# Konklusion // TODO

Skal hænge sammen med case beskrivelsen og problemformuleringen.

## Diskussion // TODO

Diskutere forskellige side af løsningen, fordele/ulemper.

Hvis der var mere tid, hvor kunne produktet udvides eller forbedres.

Reflekter over projektet

Hvad har du lært?

Valgte du de rigtige teknologier?

# Bilag // TODO

[Figur 1 Placeholder billede 4](#_Toc150432394)

[Figur 2 Biag A: Estimeret Tidsplan 7](file:///C:\Users\Ejer\Downloads\Processraport%20template.docx#_Toc150432395)

# Logbog

**6. november**

Ringede til mor ang. Patienttaxa ide og fik nogle svar besvaret. Brugte dog dagen på at brainstorme min pakkeleg ide.

**7. november**

Fik brainstormet patienttaxa idéen, og efter vejledning med Frank, konkluderede at det var det mest fyldegørende projekt af de to. Jeg lavede derefter mine kravspecs.

**8. november**

Jeg brugte hele dagen på frontend design i Figma, så jeg har et bedre overblik over projektet, og hvad jeg skal have lavet og tænkt over. Her kom mange nye idéer til properties men også bl.a. integration med Google Maps. Efter vejledning med Lars, besluttede jeg mig at omdøbe ”patienttaxa” til ”borgertaxa”, da borgerne ikke nødvendigvis er patienter ”endnu”.

**9. november**

Efter jeg blev færdig med mit design i går, har jeg fået et godt overblik over, hvad jeg skal have udviklet og implementeret. Jeg har derfor brugt dagen på at oprette opgaver og sætte dem i tidsplan. Derudover fik jeg startet på procesrapporten.

**10. november**

Fik oprettet mine backend projekter (Common, DataAccess, Business & API) samt relevante modeller og funktionalitet – dog uden SignalR og authenticate funktionalitet.

Havde nogle problemer med mine migrations og mine GET operationer, som jeg kæmpede med i noget tid.

Sluttede dagen af med at dokumentere det nye kode.

**11. november**

Fik lavet DataAccessTest projekt i NUnit frameworket. Jeg havde lidt svært ved at instanserne en in-memory database, da jeg ikke kunne kalde metoden fra EntityFrameworkCore. Jeg fandt så ud af, at UseInMemoryDatabase metoden findes i EntityFrameworkCore.InMemory NuGet pakken, hvilket var lidt irriterende.

Derudover opdagede jeg, at DbSet.ToList() kalder databasen hvorimod DbSet.Find() bruger cache, hvilket var ret forvirrende, da jeg skulle teste min Exists og Add metoder i mine repositories.

**12. november**

Har arbejdet hele min dag væk på API test. Jeg var ret forvirret over, hvordan jeg overhovedet testede mit api projekt og havde også noget EntityFramework tracking problemer, når jeg testede min Update metode.

Men jeg nåede endelig i mål, efter at have kæmpet med det i næsten 12 timer…