|  |  |
| --- | --- |
| Roameo  Your personal travel planner | Procesrapport  Aarhus Tech  Jonas Wittstrøm  Svendeprøve – Datatekniker programmering |

Indhold

[Casebeskrivelse 2](#_Toc193360794)

[Problemformulering 2](#_Toc193360795)

[Metode 2](#_Toc193360796)

[Anvendte fagområder 2](#_Toc193360797)

[Anvendte teknoligier 2](#_Toc193360798)

[Estimerede tidsplaner 2](#_Toc193360799)

[Estimeret flowchart 2](#_Toc193360800)

[Mockups 2](#_Toc193360801)

[Udviklingsprocessen 2](#_Toc193360802)

[.NET MAUI 2](#_Toc193360803)

[AWS 2](#_Toc193360804)

[Google APIs 2](#_Toc193360805)

[Opsætning af sider 2](#_Toc193360806)

[Brugerhåndtering 2](#_Toc193360807)

[Generering af planer 2](#_Toc193360808)

[Realiseret tidsplan 2](#_Toc193360809)

[Logbog (Bilag?) 2](#_Toc193360810)

[Perspektivering 3](#_Toc193360811)

[Konklusion 3](#_Toc193360812)

# Casebeskrivelse

Formålet med denne app, er at hjælpe rejsende med at planlægge deres ture på en struktureret, overskuelig og visuel måde. Den kombinerer en dag-til-dag rejseplanlægger med integration af offentlig transport og brugerens personlige præferencer for mad og aktiviteter

# Problemformulering

Hvordan kan jeg udvikle en brugervenlig rejseplanlægningsapp, der kombinerer dag-til-dag planlægning, kortintegration, offentlig transport og personlige anbefalinger?

# Metode

## Anvendte fagområder

Jeg skriver her i punktform de fagområder jeg berører under mit projekt.

* Client-side programmering
* Server-side programmering
* Objektorienteret programmering (OOP)
* App-programmering
* Versionering
* Cloud-teknologi

Jeg føler at disse spænder bredt, og giver god mulighed for at vise hvad jeg kan inde for mange forskellige områder.

## Anvendte teknologier

Jeg skriver her i punktform de teknologier, som er brugt i mit projekt. Efterfølgende uddyber jeg hvorfor jeg har taget nogle af disse valg frem for andre, samt deres fordele og ulemper.

* Visual Studio Community 2022
* Github
  + Versionering
* ASP .NET
  + MAUI
  + Core Web API
* Amazon Web Services (AWS)
  + Lambda
  + API Gateway
  + DynamoDB (NoSQL)
  + AWS NoSQL Workbench
* Google Maps + APIs
  + Navigation
  + Geocoding
  + Places
* PlantUML
  + Som plugin i Obsidian (Notat-software)
  + Flowchart

### Begrundelse af valg

Valget af Visual Studio (VS22) var nemt, da det for det første er den IDE jeg har arbejdet med mest – både privat og på min læreplads. Udover dette, er det specielt bygget for .NET applikationer og tilbyder skabeloner for de fleste projekttyper, så man hurtigt kan komme i gang.   
Jeg kunne godt have lavet et tilsvarende projekt i Visual Studio Code (VSCode), som er mere letvægt (kræver færre ressourcer af din maskine) og mere fleksibelt på mange punkter.   
Hvis jeg lavede mit projekt i eksempelvis React Native frem for .NET, havde det givet meget bedre mening med VSCode.

Som nævnt ovenfor, valgte jeg .NET MAUI, da jeg i forvejen er mest komfortabel med .NET frameworket, som er det jeg har arbejdet mest med gennem hele uddannelsen – både i skole og på min læreplads. Denne løsning gør det også muligt at skrive apps til flere platforme på samme tid med en fælles kodebase, uden at jeg behøver at lære at skrive til Android eller Apple specifikt.   
Havde dette ikke været tilfældet, kunne en anden god løsning have været React Native, som er baseret på JavaScript, og har et stort fællesskab og et bredt udvalg af plugins og biblioteker. Dog kræver det også at man har noget dybere kendskab til de platforme man udvikler til, end hvad jeg har.

Grunden til at jeg valgte at gøre brug af AWS til mit projekt frem for Azure, som måske havde været det mere naturlige valg, var fordi jeg på hovedforløb 5 havde forsøgt netop dette, men fandt det meget svært at sætte op. Det endte derfor med at jeg ikke havde et produkt at vise efter flere ugers arbejde, fordi jeg sad meget fast.   
Azure ville dog give god mening, da det allerede er en integreret del af Microsoft, og på papiret skulle være meget nemt at sætte op til et projekt som mit.  
Desuden fandt jeg også opsætningsprocessen i AWS meget let, især deres UI var intuitivt at finde rundt i. Jeg følte derfor jeg kunne bruge mere tid og kræfter på udviklingen af mit projekt.

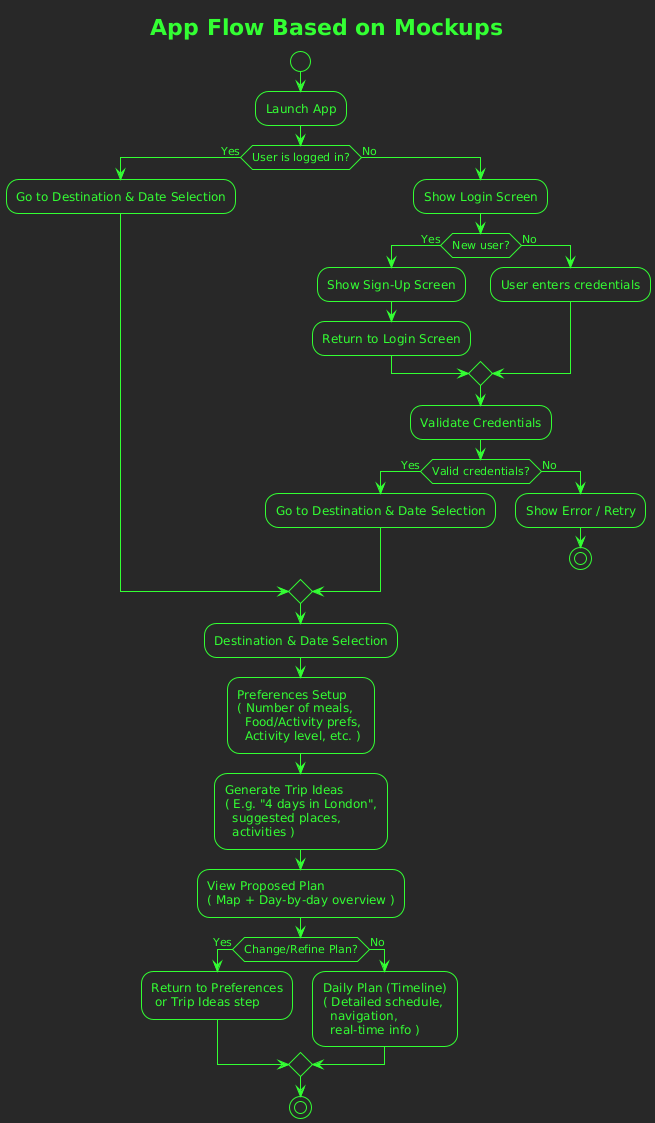
## Estimeret tidsplan



Ovenfor ses min estimerede tidsplan for projektet.

## Estimeret flowchart

Her vises et flowchart over, hvordan jeg havde tænkt mig at flowet skulle være, før jeg startede med at udvikle projektet.



## Mockups

I dette afsnit vil jeg gennemgå de mockups jeg fik lavet før jeg startede udviklingen. Disse ligger som bilag 1.

Mine mockups dækker ret godt over de funktioner, som også endte med at komme med i appen i sidste ende. Har bygget min app meget med udgangspunkt i disse.   
På tegningerne kan man se en login-side, hvor der også er gjort tanker om brugeroprettelse. Der ses et udkast til DestinationPickerPage, som er appens startside, hele vejen til siden for den detaljerede dagsplan med navigation, som er sidste side i min app.

# Udviklingsprocessen

I dette afsnit vil jeg skrive lidt om, hvad der gik godt og mindre godt under udviklingsprocessen. Jeg vil også sætte nogle ord på, hvordan jeg fik løst de problemer der opstod undervejs. Jeg vil komme forbi alle de mest centrale elementer i min app.

## .NET (MAUI og Web API)

Grunden til at jeg valgte MAUI, var en kombination af to ting: 1. Jeg ville udfordre mig selv med at lave en mobil app, efter mit fejlslagne projekt på hovedforløb 5. 2. Jeg er allerede godt bekendt med .NET, som nævnt tidligere.

Selve opsætningen gik nemt – Jeg valgte MAUI skabelonen i VS22 og på meget kort tid havde jeg nogle grundlæggende filer og mapper på plads.

Det første jeg gjorde, var at tilføje et Web API til mit projekt og begynde at bygge mine datamodeller og controllere.

Gennem hele processen, var dét at arbejde med .NET klart det der kørte mest flydende og med meget få problemer undervejs. De få der kom, indebar ofte at jeg manglede en NuGet pakke eller lign. Dette løses ret nemt gennem NuGet Packet Manageren i VS22.

Da jeg ikke har arbejdet meget med APIer var jeg lidt i tvivl om, hvordan jeg skulle bygge mine datamodeller og controllere. Dette fik jeg hjælp til ved google søgninger og AI.

Generelt kørte denne del af processen uden de helt store problemer.

## AWS

Til at hoste min app online, havde jeg tænkt at bruge AWS. De har et ret generøst Free Tier, så man hurtigt kan få sin app i luften uden omkostninger. I hvert fald ikke for et projekt så lille som mit.

Til selve hosting af appen brugte jeg Lambda, som en serverløs tjeneste, som kører kode uden behov for serveradministration. Denne aktiveres automatisk ved specifikke hændelser eller HTTP-kald.   
Da AWS ikke er en integreret del af Microsofts ”øko-system” som Azure, krævede det lidt mere opsætning. Til at få styr på denne del, fandt jeg en YouTube video, der på trods af at være indtalt af AI, gav en god guide til opsætningen.

Til at håndtere API kald brugte jeg API Gateway, som er en service, som gør det muligt at administrere og sikre mine APIer online og forbinde klienter med mine backend services.   
Opsætningen af denne foregik for det meste online, efter jeg havde bygget mine controllere i mit .NET projekt. Her skulle jeg opbygge min API til at spejle de endpoints og metoder jeg havde i mit projekt.

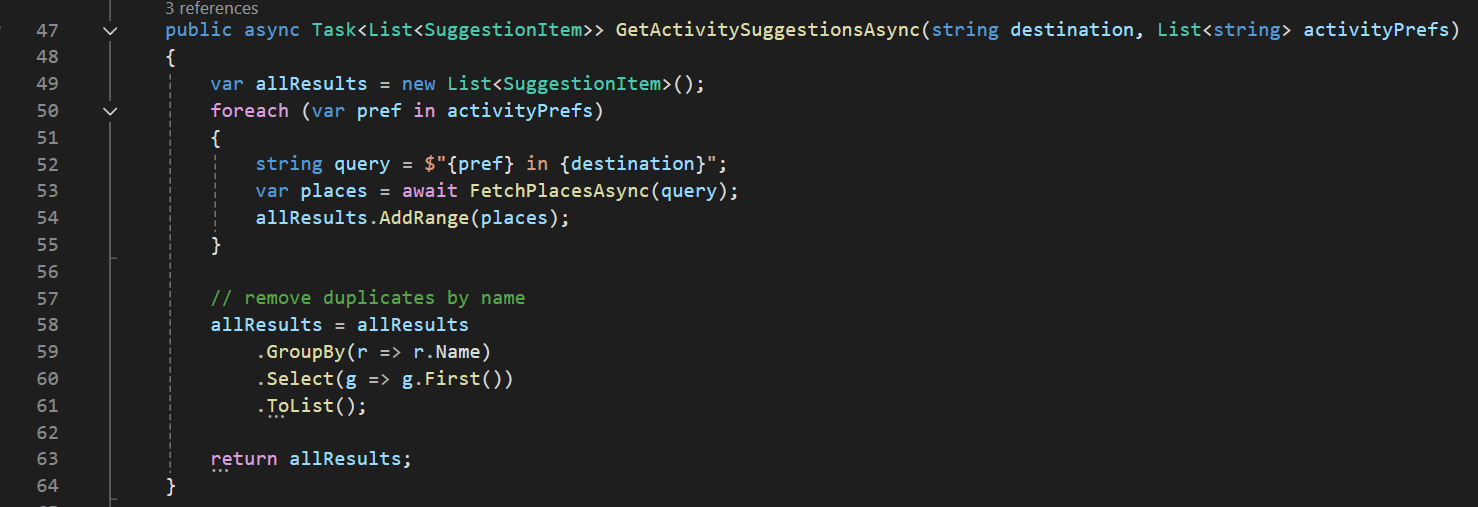
Som min database valgte jeg at gå med DynamoDB, som er en NoSQL database. NoSQL er en en type database, som gemmer data i fleksible strukturer, som f.eks. dokumenter eller nøgleværdier. Dette giver større fleksibilitet end traditionelle tabeller.  
Opsætningen af denne foregik gennem NoSQL Workbench, som er et stykke software AWS har lavet til netop dette formål.   
Da NoSQL er nyt for mig, skulle jeg også lige bruge lidt tid på at sætte mig ind i dette, men alt i alt gav det ikke de store problemer.

## Google Cloud APIs

Jeg vidste at jeg skulle bruge navigation og informationer om forskellige virksomheder til min app. Derfor føltes Google som det oplagte valg til netop dette.

Ligesom AWS har Google Cloud også et ret generøst Free Tier, der tillader brugen af de fleste af deres APIer i et ret stort omfang uden omkostninger.

For at komme i gang, skulle jeg bruge en API Key, som jeg fik kort efter jeg skrev mig op til deres services. Denne kunne jeg så bruge i mit projekt til at lave forespørgsler (queries) til den API jeg nu skulle hente informationer fra.   
I min app brugte jeg Geocoding API til at hente oplysninger om Byer/områder når jeg oprettede rejser fra min DestinationPickerPage.  
Senere brugte jeg den til at hente basale oplysninger om spisesteder og aktiviteter fra deres Places API.

Forespørgslerne til disse API’er er ret simple:  


I kodestykket her, viser jeg min GetActivitySuggestionsAsync-metode. Her henter jeg data omkring hvilke aktiviteter brugeren kan tage til inden for deres præferencer. På linje 52 vises et eksempel på hvordan en query til Googles API kan se ud. Der bliver bare sendt en forespørgsel omkring hvilke aktiviteter der kan findes i det valgte område.  
Derefter sorteres den, og dubletter fjernes, så brugeren ikke ser det samme resultat flere gange.

Generelt syntes jeg at brugen af Googles APIer var meget ligetil, og stødte ikke rigtig på nogle problemer.

## Generering af planer

## Brugerhåndtering

## Realiseret tidsplan

Hvordan afviger den fra den estimerede? Hvorfor?

## Logbog (Bilag?)

# Perspektivering

# Konklusion

# Kilder og referencer

## YouTube

[Amazon DynamoDB in .NET - The Getting Started Guide](https://www.youtube.com/watch?v=BJYDWMN8taI) – Mukesh Murugan

[Deploy .Net Core Web API (.Net 8) to AWS Lambda Function from VS2022 and Config with The API Gateway](https://www.youtube.com/watch?v=9a-wpPg0QyI) – Lucky Happy

## AI

[ChatGPT](https://chatgpt.com/)

GitHub Copilot

## Grafik

Michelle Wittstrøm Petersen – Mockups, app ikon og forsidebillede.

# Bilag

## Bilag 1 – Mockups

## 

