Aarhus Universitet
Semesterprojekt 4
Gruppe 5

Analyse

CarnGo

Gruppemedlemmer:
Edward Hestnes Brunton
(201705579)
Marcus Gasberg
(201709164)
Martin Gildberg Jespersen
(201706221)
Mathias Magnild Hansen
(201404884)
Tristan Moeller
(201706862)
Erik Mowinckel
(20107667)
Hamza Ben Abdallah
(201609060)

Vejleder:
Jesper Michael Kristensen, Lektor

29. maj 2019

Indhold

1 Analyse

I dette dokument beskrives det analyse arbejde, samt de forskellige overvejelser der har været i løbet af arbejdsprocessen. De største overvejelser for projektet har omhandlet den grafiske brugeroverfalde og databasen. I de næste to afsnit beskrives de overvejelser der har været, fordele og ulemper, samt det endelige valg.

1.1 Applikation og Grafisk Brugeroverflade

For at brugeren kan interagere med vores system, skal der være en grafisk brugeroverflade. En anden essentielt faktor er, hvilke operativsystemer skal applikationen kunne virke på. Hertil undersøgte vi flere systemer/platforme. (Vi undersøgte også browser baserede løsninger, men det blev forkastet efter vi valgte at systemet skulle være en applikation). Det var desuden også bestemt at det skulle være på dotNet platformen. Vi endte derved mellem to valg: WPF eller Xamarin applikation - her introduceres nogen af fordelene og ulemperne ved begge:

Windows Presentation Foundation (WPF, Windows Only):
Fordele:

- Stor erfaring fra undervisning og mulighed for at hjælp fra faglærere.
- MVVM Pattern - let at overholde lav kobling mellem GUI og business logik.

Ulemper:

- WPF er et subsystem til .NET platformen, som bruges til at lave applikation og den grafiske brugeroverflade dertil. Dette betyder at den er Windows only.
- Erfaring siger den har en længere læringskurve.

Xamarin (Mobile App, Cross Platform):
Fordele:

- Xamarin applikationer er 'native', dvs. de udarbejdes til specifikke platforme. Xamarin er platformsuafhængig og kan bruges til Windows, iOS og OS X.
- MVVM Pattern - let at overholde lav kobling mellem GUI og business logik.

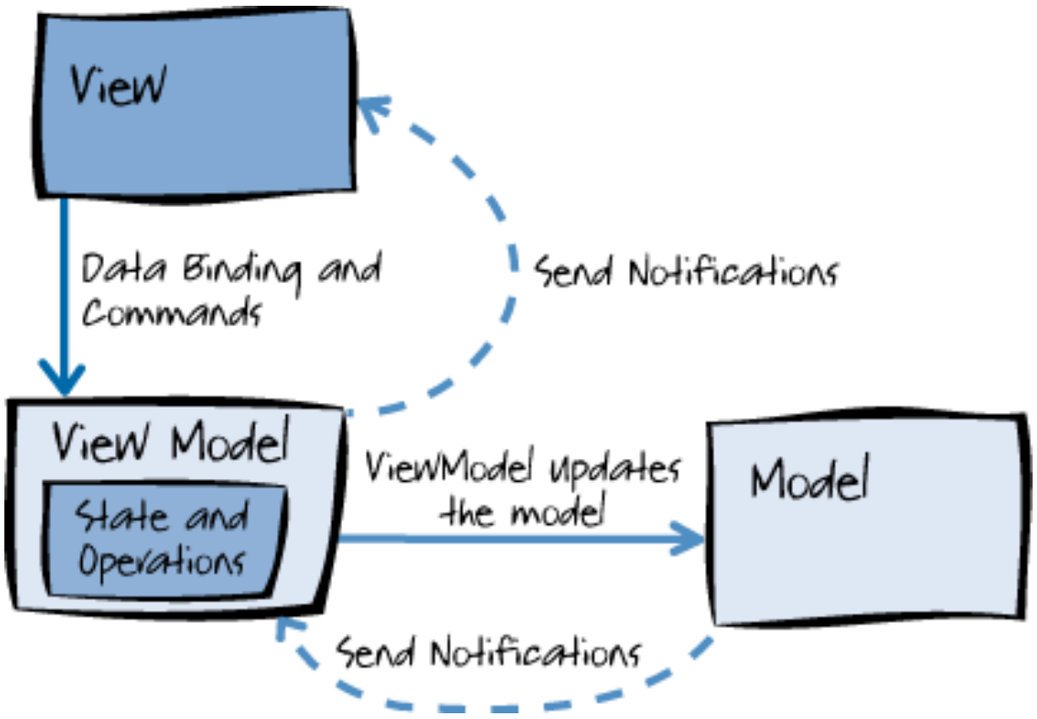
Ulemper:

- Ingen i gruppen har erfaring med Xamarin
- Ingen undervisning i Xamarin, alt skal undersøges selv

Ud fra ovenstående argumenter, valgte vi at lave en WPF applikation. Der er begrænset tid i løbet af et semesterprojekt, så det at have mulighed for at få hjælp af en faglærer, samt løbende at få mere erfaring for WPF, er en kæmpe fordel. Hvis derimod projektet skulle blive kommercialiseret, ville det have været nødvendigt at lave en applikation til alle platforme, for at konkurrere med GoMores platformuafhængige applikation.

1.1.1 MVVM

MVVM arkitekturen gør det muligt at holde applikationens forretnings- og præsentationslogik adskilt. Arkitekturen består af tre kernekomponenter: View, ViewModel og Model (se figur 1.1)



Figur 1.1: MVVM arkitektur

Views indkapsler brugeroverflade og dets logik. View'et er de visuelle elementer, som præsenteres for brugeren.
ViewModels indholder præsentationslogikken og dets stadie. Det har ingen direkte reference til View'et. ViewModellen indeholder properties og kommandoer, som databindes til View'et. View'et bestemmer, hvordan funktionalitetet skal gengives, hvor ViewModeller har til ansvar at koordinere View'ets interaktion med systemets data og modeller
Modeller indeholder applikationens forretningslogik og -data. Dette kan være hentning og administration af applikationsdata og for at sikre at forretningsregler, der sikrer datakonsistens og -gyldighed, bliver overholdt.

1.1.2 Fonts

Normalt er det først i designfasen, hvor man udvælger og definerer, hvilke fonts man bruger til de forskellige vinduer. Det er dog ikke alle fonts, som er gratis eller indbygget i WPF. Der bruges derfor kun fonts, som er gratis og indbygget i WPF.

1.2 Database:

I dette afsnit beskrives de overvejelser, der har været i forhold til kreationen af databasen. Heri undersøges både lagring af data i form af database udbyder, samt hvordan data traditionelt overføres mellem brugere.

1.2.1 Valg af kommunikationstype

Interval polling & event queries
Siden vi arbejder med en GUI som skal forbindes med databasen har vi valg at kombinere 2 måder at holde en applikation opdateret. Vi opdatere databasen når bestemte begivenhedder sker, og ved længere polles.
Vi har valgt en relationel database, da NOSQL er optimeret mere mod store mængder data af varierende type og mindre mod garantier for at data er fejlfri. Siden vi arbejder med relativt

små mængder data som involvere personlig info er det vigtigere at data er garanteret for at være korrekt.