

## 4. Semesterprojekt

# CityCrawl

## Hovedrapport

Projektgruppe 1

Efterår 2021



Deltagere		
Studienummer	Navn	Studieretning
201705103	Andreas Stavning Erslev	Softwareteknologi
201807859	Asger Busk Breinholm	Softwareteknologi
201911338	Christina Boll Pedersen	Softwareteknologi
201904202	Johanne Berg	Softwareteknologi
201910327	Maagisha Mahenthirarajan	Softwareteknologi

Vejleder	
Poul Ejnar Rovsing	per@ece.au.dk



## 1 Resumé

Denne rapport beskriver et 4. semesterprojekt på Aarhus Universitet. Projektgruppen består af fem studerende fra Softwareteknologi-linjen. Udviklingsprocessen for dette projekt har bestået af Scrum, og produktudviklingen har bestået af en iterativ arbejdsgang. Formålet med CityCrawl-projektet har været at gøre det nemmere at møde nye mennesker og få nye bekendtskaber igennem en platform, især i denne tid, hvor Coronapandemien har haft stor påvirkning på vores sociale liv og gjort, at mange har været meget alene. Resultatet for projektet er en implementering af en WPF-applikation (CC-App), som er brugerens tilgang til CityCrawl, samt en implementering af en web-applikation (CC-Web), som er virksomhedens tilgang til CityCrawl. CC-Web indeholder et Web-API, der bruges til kommunikation mellem CC-App og en database (CC-Database). Web-API'et gør det muligt at CC-App kan skrive og læse data fra CC-Database. CC-Database indeholder nødvendige entiteter, som svarer til de informationer, der skal gemmes for brugeren, virksomheden og de bookede pubcrawls. Dermed kan virksomheden få en oversigt over tilmeldinger af pubcrawls, når en bruger booker et pubcrawl på CC-App.

## 2 Abstract

This project is a 4th semester project, by five students attending the faculty Software Technology at Aarhus University. The process for this project is based on the development technique Scrum and the product development is based on an iterative workflow. The goal of this projects is to make it easier for people to meet other people as part of a pubcrawl. Especially in the time of the Corona pandemic, which has a big influence on all our social lives and a lot of people are left alone. The result of this project is an implementation of a WPF-application (CC-App), which is for the users to interact with CityCrawl and a web-application (CC-Web) for the companies, who facilitates the pubcrawls, to interact with CityCrawl. The CC-Web consists of a Web-API, which is used for the communication between CC-App and a database (CC-Database). The Web-API gives CC-App the functionality to enable writing and reading of data from CC-Database. CC-Database has the necessary entities for CityCrawl, which contains the information about the user, the company and the booked pubcrawls. This enables the company to have access to a list of booked pubcrawls in CC-Web, which is updated whenever the user books a pubcrawl in CC-App.



## Indholdsfortegnelse

<b>1</b>	<b>Resumé</b>	<b>1</b>
<b>2</b>	<b>Abstract</b>	<b>1</b>
<b>3</b>	<b>Indledning</b>	<b>5</b>
<b>4</b>	<b>Problemformulering</b>	<b>6</b>
<b>5</b>	<b>Kravspecifikation</b>	<b>7</b>
5.1	Formelle krav . . . . .	7
5.2	Systemspecifikation . . . . .	7
5.3	User Stories . . . . .	8
5.3.1	User Stories for brugeren . . . . .	8
5.3.2	User Stories for virksomheder . . . . .	8
5.4	MoSCoW for User Stories . . . . .	9
5.5	Ikke-funktionelle krav . . . . .	10
5.6	MoSCoW for de ikke-funktionelle krav . . . . .	10
<b>6</b>	<b>Afgrænsning</b>	<b>11</b>
<b>7</b>	<b>Metoder</b>	<b>12</b>
7.1	SOLID . . . . .	12
7.2	Scrum . . . . .	12
7.3	Git . . . . .	14
7.4	Continuous Integration (CI) . . . . .	14
<b>8</b>	<b>Teknologianalyse</b>	<b>15</b>
8.1	Valg af platform . . . . .	15
8.1.1	WPF Applikation . . . . .	15
8.1.2	Web-applikation . . . . .	15
8.2	Valg af kodesprog . . . . .	15
8.2.1	C# . . . . .	15
8.2.2	XAML . . . . .	15
8.2.3	HTML . . . . .	15
8.2.4	CSS . . . . .	15
8.2.5	ASP.NET Core (MVC) . . . . .	16
8.2.6	Relationel Database . . . . .	16
8.2.7	Entity Framework Core . . . . .	16
<b>9</b>	<b>Systemarkitektur</b>	<b>17</b>
9.1	C4-model af CityCrawl . . . . .	17
9.2	4+1 view model for CityCrawl . . . . .	21
9.2.1	User Story View . . . . .	21
9.2.2	Logical View . . . . .	25
9.2.3	Deployment View . . . . .	26
<b>10</b>	<b>CC-App</b>	<b>27</b>
10.1	Design af CC-App . . . . .	27
10.2	Implementering af CC-App . . . . .	29
10.3	Test af CC-App . . . . .	31
10.3.1	Unit test af CC-App . . . . .	31
10.3.2	Systemtest af CC-App . . . . .	33
10.4	CCHttpClient . . . . .	34
<b>11</b>	<b>Web-API</b>	<b>35</b>
11.1	Implementering af Web-API . . . . .	35



11.2 Test af Web-API . . . . .	35
<b>12 CC-Web</b>	<b>37</b>
12.1 Design af CC-Web . . . . .	37
12.2 Implementering af CC-Web . . . . .	38
12.3 Test af CC-Web . . . . .	40
<b>13 CC-Database</b>	<b>42</b>
13.1 Design af CC-Database . . . . .	42
13.2 Implementering af CC-Database . . . . .	43
13.3 Test af CC-Database . . . . .	44
<b>14 Accepttest</b>	<b>46</b>
14.1 Accepttest for CC-App . . . . .	46
14.2 Accepttest for CC-Web . . . . .	46
14.3 Konklusion for accepttest . . . . .	46
<b>15 Opnåede erfaringer</b>	<b>47</b>
<b>16 Fremtidigt arbejde</b>	<b>47</b>
<b>17 Konklusion</b>	<b>48</b>
<b>18 Referencer</b>	<b>49</b>



Forkortelse	Definition
<b>CC</b>	CityCrawl systemet, som består af CC-App, CC-Web og CC-Database.
<b>Platform</b>	Fælles betegnelse for hele CityCrawl, som består af CC-App, CC-Web og CC-Database.
<b>CC-App</b>	Delsystem af CityCrawl som giver brugeren adgang til at tilmelde pubcrawl, som er baseret på WPF.
<b>CCHttpClient</b>	Klasse i CC-App som anvender HttpClient metoden og REST i form af GET og POST, til at kommunikere med Web-API controllren som ligger i CC-Web. Dette skal gøre at data fra CC-App kan sendes og læses fra CC-Database.
<b>Web-API</b>	API controller i CC-Web, som anvender REST til at oprette komunikation mellem CC-App og CC-Database, via CC-Web.
<b>CC-Web</b>	Delsystem af CityCrawl som giver virksomheden adgang til at tilbyde pubcrawl, som er baseret på ASP.NET Core.
<b>CC-Database</b>	Delsystem af CityCrawl som giver brugeren og virksomheden adgang til at skrive og læse data fra databasen i CityCrawl, som er en relationel database baseret på Entity Framework Core.

Tabel 1: Definitionslisten er en oversigt over de forkortelser som anvendes i hovedrapporten.

Arbejdsfordeling	
Ansvarsområde	Deltagere
CC-App	Asger, Christina og Maagisha
CC-Web	Andreas og Johanne
CC-Database	Andreas og Johanne
Web-API	Andreas, Christina og Johanne
CCHttpClient	Asger, Christina og Maagisha

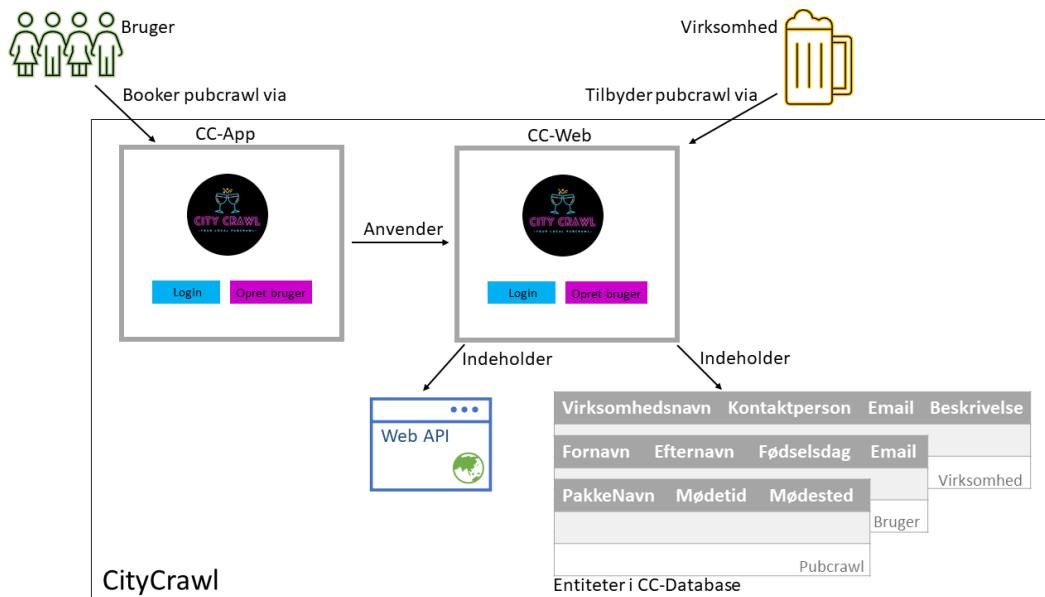
Tabel 2: Tabellen viser arbejdsfordelingen i projektgruppen. Arbejdet i projektgruppen er blevet inddelt i de ovenstående ansvarsområder.

### 3 Indledning

Pubcrawl platformen CityCrawl er blevet udarbejdet til at være en social platform, hvis formål er at skabe relationer, både ved nye og gamle bekendtskaber. At være social har altid været centralt for mennesket og behovet er bestemt ikke blevet mindre igennem de seneste år, hvor vores alles hverdag er blevet ændret markant. Derfor er CityCrawl blevet udviklet med en vision om at bringe mennesker sammen.

I sin simpleste form er CityCrawl en platform, hvor det er muligt for brugere at booke sig på en pubcrawl faciliteret af virksomheder i ens lokalområde. Platformens primære målgruppe er tilflyttede studerende, fordi det som tilflytter kan være svært at vide, hvor man kan finde nye relationer og bekendtskaber. CityCrawl er netop udviklet til at være byens nye samlingspunkt, som gør det nemt for mennesker at mødes i en uformel sammenhæng.

For at skabe et overordnet indblik for opbygningen af CityCrawl, er der blevet udarbejdet et rigt billede, som illustreres i figur 1. CityCrawl består overordnet af fire dele. Først en WPF-applikation, kaldet CC-App, hvor det er muligt for brugere at oprette en profil og tilmelde sig forskellige pubcrawl-pakker. Herefter en web-applikation, kaldet CC-Web, hvor det er muligt for virksomheder at registrere sig og dermed tilbyde et pubcrawl-event. Dernæst består CityCrawl af en database, kaldet CC-Database, som både kan skrives til og læses fra. Afslutningsvis består CityCrawl af et Web-API, som gør det muligt for CC-App at kommunikere med CC-Database, via CC-Web.



Figur 1: Rigt billede over CityCrawl



## 4 Problemformulering

I dette afsnit vil der beskrives, hvilket problem der gør at CityCrawl projektet er relevant. CityCrawl er blevet udarbejdet til at være en social platform, hvor der dannes relationer, både ved kendte og nye mennesker. Det sociale har altid været relevant for mennesker, men i denne Coronatid har det været ekstra vigtigt. Men forhåbentlig vil Corona ikke forblive et handicap for verden, og ikke mindst Danmark. Derfor har der ikke været fuldt fokus på Corona situationen, men nærmere på hvilke elementer der i normale situationer er påkrævet.

CityCrawl giver mulighed for private og offentlige begivenheder, som gør det muligt for nytilkommere til en given by, at skabe kendskab til både byen og indbyggerne. Da mange unge mennesker rundt i Danmark flytter til nye byer for at studere, kan det være svært at falde til. Byen er ny, måske stor og uoverskuelig, og relationer er en mangelvare. Ved opstart på universitetet, forsøges der at skabe et sammenhold mellem de studerende. Dog er perioden for opstart ikke uendelig lang, og der kan forekomme mange situationer, hvor ikke alle falder på plads i en gruppe. CityCrawl øger muligheden for at løbende skabe nye bekendtskaber. Alt dette fungerer også i andre situationer, såsom tilflytttere fra andre lande, som måske ikke taler dansk.

Som beskrevet er der også private muligheder, fx ved fødselsdage, eller blot efter en god uge i skolen, kan det være rart at feste og fejre det i byen. Da disse situationer ofte er tiltænkt at være tilknyttet en gruppe af venner og bekendte, skal muligheden for at holde det privat være tilgængelig. Hvis der da mødes nye venskaber løbende, kan gruppen selv skabe en mere åben pubcrawl. Dog vil selve fordelene ved den givne pubcrawl kun være mulig for dem, der har bestilt. Der er altså ikke en udvidelsesfunktion for CityCrawl.

En bytur defineres forskelligt fra person til person og gruppe til gruppe. Nogle gange skal en pubcrawl være starten på aftenen, andre gange er det fokus for turen. Derfor oprettes der forskellige pakker, som giver muligheden for at forme, hvordan byturen skal udfolde sig.

En pubcrawl kræver, at der er steder at besøge. Det vil sige, at der skal være mulighed for at have nogle virksomheder, som brugerne af CityCrawl kan besøge i løbet af deres pubcrawl. Disse virksomheder skal tilbyde og sikre, at der er plads til brugerne af CityCrawl. Yderligere vil virksomhederne uddele drinks, shots, øl eller hvad fantasien kan opfinde. Virksomheden får mulighed for at sælge alkohol og vise sig frem. Det vil altså sige, at der er mulighed for direkte salg, men også reklame. CityCrawl kan også sørge for, at virksomhederne er sikre på en vis mængde af gæster i løbet af en aften.

Det er gratis for både virksomheder og brugere at oprette sig. Dog koster det penge at bestille en pubcrawl, hvor prisen er bestemt ud fra, hvilken pakke der bliver bestilt af brugeren. For at øge tilgængeligheden for forskellige størrelser af pubcrawls, vil der også være forskel på, hvilken gruppstørrelse der er understøttet. Dette betyder, at visse virksomheder kun er tilgængelige ved en given mængde af besøgende.

For at teste virkningen af de forskellige brugergrænseflader og for at skabe den bedste oplevelse for brugere og virksomheder, er der blevet oprettet to grænseflader. Da det har været muligt at lave flere brugergrænseflader, er der blevet overvejet, hvilken som passer bedst. Derved har der været et ønske om at teste begge dele. Yderligere er det blevet vurderet, at den måde bruger og virksomhed interagerer med CityCrawl er forskellig. Derfor er der blevet oprettet en WPF-applikation, kaldet CC-App og en web-applikation, kaldet CC-Web. CC-App giver mulighed for, at brugere kan oprette sig samt tilknytte sig en given pubcrawl-pakke. CC-Web giver mulighed for, at en virksomhed kan oprette sig samt se deres profil, hvor der er oplysninger om virksomheden, og hvilke pubcrawl der er tilknyttet virksomheden.



## 5 Kravspecifikation

I dette afsnit forekommer kravspecifikationen for CityCrawl i form af de opstillede krav fra universitetet, en systemspecifikation, User Stories og en MoSCoW analyse for både funktionelle og ikke funktionelle krav.

### 5.1 Formelle krav

I dette afsnit vil der blive beskrevet, hvilke krav, der er blevet stillet af universitetet samt hvordan CityCrawl opfylder disse krav.

De formelle stillede krav, som universitetet har givet for 4. semesterprojektet er:

- Projektet skal inddrage faglige aspekter fra samtlige fag på 4. semester SW. Dette skal dokumenteres i projektrapporten og bør inddrages af den studerende til eksamen.
- Projektet skal have et passende omfang, så alle i projektgruppen kan arbejde med projektet.
- Projektet skal være en karakter der tillader, at læringsmål i faget opfyldes.
- Der skal afleveres et projektforslag med en problembeskrivelse og projektbeskrivelse

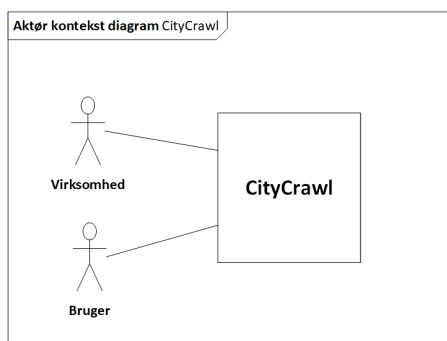
Tabel 3 viser hvilket fag fra semesteret og hvor viden fra disse fag er blevet anvendt i projektet.

Anvendte fag i semesterprojektet	
Fag	Anvendelse i projektet
GUI-programmering	Til udarbejdelsen af CC-App og CC-Web
Netværksprogrammering og grundlæggende kommunikationsnetværk	Til udarbejdelsen og test af Web-API
Databaser	Til udarbejdelsen af CC-database
Software Design	Til design og arkitektur for CityCrawl
Software Test	Til test af CityCrawls funktionalitet, i form af unit test og systemtest

Tabel 3: Tabellen viser hvor den tillærte viden fra semesterets forskellige fag er blevet anvendt til udviklingen af CityCrawl.

### 5.2 Systemspecifikation

CityCrawl er en platform hvor på virksomheder kan tilbyde at udbyde pubcrawl-events og brugere kan tilmelde sig pubcrawls. Aktør kontekst diagrammet i figur 2 viser de aktører som interagerer med CityCrawl. Systemet har to primære aktører som er virksomhed og bruger. Virksomhed repræsenterer de bører som ønsker at være tilgængelig på CityCrawl og dermed kunne tilbyde pubcrawls. Bruger repræsenterer de privatpersoner som ønsker at tilmelde sig en pubcrawl.



Figur 2: Aktør kontekst diagram



## 5.3 User Stories

I dette afsnit vil User Stories for de to primære aktører, bruger og virksomhed, blive beskrevet. User Stories er en metode som anvendes til at beskrive platformen CityCrawl's software specifikationer og de funktionelle krav.

### 5.3.1 User Stories for brugeren

1. Som bruger kan jeg tilgå CityCrawl via en Windows applikation.
2. Som bruger kan jeg oprette en profil, således CityCrawl kan kende mine personoplysninger.
3. Som bruger kan jeg logge ind på CityCrawl.
4. Som bruger kan jeg trykke mig ind på forskellige vinduer der for eksempel viser min profil og booking af pubcrawl-events.
5. Som bruger kan jeg verificere min alder i min brugerprofil, således at CityCrawl overholder myndighedernes aldersgrænse for servering af alkohol.
6. Som bruger kan jeg booke et pubcrawl-event til en bestemt dato.
7. Som bruger kan jeg se på et digitalt kortsystem, hvor de tilmeldte barer er beliggende.
8. Som bruger kan jeg oprette et privat pubcrawl-event.
9. Som bruger kan jeg slette et privat pubcrawl-event.
10. Som bruger kan jeg vælge pubcrawl-pakker med prædefinerede virksomheder.
11. Som bruger kan jeg se den korteste rute til et bestemt pubcrawl-event.
12. Som bruger kan jeg foretage en betaling af min pubcrawl-event via CityCrawl applikationen.

### 5.3.2 User Stories for virksomheder

1. Som virksomhed kan jeg tilgå CityCrawl via en Web baseret applikation.
2. Som virksomhed kan jeg oprette en profil, således CityCrawl kan kende min virksomheds oplysninger.
3. Som virksomhed kan jeg logge ind på CityCrawl.
4. Som virksomhed kan min lokation blive vist via et digitalt kortsystem.
5. Som virksomhed kan jeg tilbyde mine ydelser til et privat pubcrawl-event.
6. Som virksomhed kan jeg slette et pubcrawl-event.



## 5.4 MoSCoW for User Stories

Analysemødellen MoSCoW anvendes i det nedestående for at konkretisere systemets User Stories. Heraf er analysen af de funktionelle krav fra systemets User Stories inddelt i fire kategorier: Must have, Should have, Could have og won't have.

### Must have:

- *M1*: Der skal være en Windows applikation (CC-App), i .NET Core.
- *M2*: Der skal være en web baseret applikation (CC-Web), i ASP.NET.
- *M3*: Der skal være en database (CC-Database) der skal kunne lagre systemets data.

### Should have:

- *S1*: Der burde være en mulighed for at booke pubcrawl-event ud fra dato.
- *S2*: Der burde være en mulighed for at se hvor virksomhederne er beliggende på et kortsystem.
- *S3*: Der burde være mulighed for at tilgå flere underliggende GUI vinduer i begge applikationer.
- *S4*: Der burde være en aldersbegrænsning på minimum 18 år for at bruge CityCrawl platformen.

### Could have:

- *C1*: Der kunne være mulighed for at slette et pubcrawl-event.
- *C2*: Der kunne være mulighed for at vælge pakker med prædefineret virksomheder.
- *C3*: Der kunne være mulighed for at CityCrawl udregner den korteste rute for given pubcrawl.
- *C4*: Der kunne være mulighed for at lave et privat pubcrawl-event.

### Won't have:

- *W1*: Der vil ikke kunne foretages betaling på CityCrawl platformen
- *W2*: Der vil ikke registreres faktiske virksomheder



## 5.5 Ikke-funktionelle krav

I dette afsnit vil de ikke-funktionelle krav for CityCrawl blive beskrevet. Ud fra denne beskrivelse vil der blive udarbejdet en MoSCoW-analyse.

- Det skal være muligt, at en bruger og virksomhed kan tilgå CityCrawl platformen samtidigt.
- Det skal være muligt, at en bruger kan booke flere pubcrawls ad gangen.
- Der skal være minimum 2 fiktive virksomheder tilknyttet en pubcrawl.
- Det skal være muligt, at CC-App er operationel i 99,99% af tiden.
- Det skal være muligt, at CC-Web er operationel i 99,99% af tiden.
- Databasen skal kunne indeholde op til 100 virksomheder.
- Databasen skal kunne indeholde op til 100 brugere.
- Det vil ikke være muligt at kunne foretage betaling via platformen.
- Der skal være minimum en unit test per signifikante funktionalitetsområde.
- Man skal ikke kunne tilknytte sig flere pubcrawls.

Disse krav er stillet på baggrund af, at konceptet for CityCrawl består i, at deltagerne skal rundt og besøge forskellige barer. Derfor vil det netop forringe brugeroplevelsen, hvis der på platformen ofte opstår fejl. Hvis systemet ofte er utilgængeligt vil brugerne og virksomhederne bliver trætte af at bruge CityCrawl. Derudover er tallet for databasen et estimeret tal for at sikre, at der er plads nok på den anvendte database, og at det er en rimelig mængde data for en første iteration af CityCrawl.

## 5.6 MoSCoW for de ikke-funktionelle krav

Analysemødellen MoSCoW anvendes i det nedestående for at konkretisere systemets ikke-funktionelle krav. De ikke-funktionelle krav inddeltes i fire kategorier: *Must have*, *Should have*, *Could have* og *won't have*.

### Must have:

- $M1_I$ : En bruger og virksomhed skal kunne tilgå CityCrawl samtidig.
- $M2_I$ : Der skal være et minimum af to fiktive virksomheder tilknyttet et pubcrawl-event.
- $M3_I$ : Der skal være minimum en unit test pr. signifikante funktionalitetsområde.

### Should have:

- $S1_I$ : CC-App burde være operationel i 99,99% af tiden.
- $S2_I$ : CC-Web burde være operationel i 99,99% af tiden

### Could have:

- $C1_I$ : Der kunne være mulighed for at databasen kan indeholde op til 100 brugere.
- $C2_I$ : Der kunne være mulighed for at databasen kan indeholde op til 100 virksomheder.

### Won't have:

- $W1_I$ : Der vil ikke være en mulighed for at have flere måder at tilgå betaling (MobilePay, betalingskort).
- $W2_I$ : Der vil ikke være en mulighed for at tilknytte sig flere pubcrawls.



## 6 Afgrænsning

I dette afsnit vil projekt afgrænsningerne for CityCrawl blive beskrevet, samt en begrundelse for den besluttede afgrænsning.

Projektgruppen har arbejdet ud fra en iterativ proces hvor der først er blevet bestemt de mest basale features for platformen og herefter er der blevet udarbejdet nye features løbene. Det vil sige at første prioritet har været at få platformenes primære systemer, CC-App, CC-Web og CC-Database, til at kommunikere med hinanden og efterfølgende har prioriteten været at optimere på systemerne. Det har medført at nedenstående features er blevet nedprioriteret grundet tidsmangel.

Det vil i CC-App ikke være muligt for brugeren at tilgå et digitalt kort hvor beliggenheden på de barer som er inkluderet i den bookede pubcrawl, samt vil det heller ikke være muligt for brugeren at se den korteste rute til et pubcrawl-event.

Det vil ikke i CC-App være muligt for brugeren at hverken oprette eller slette et privat pubcrawl-event.

CC-App vil have ingen sikkerhed i forhold til at overholde myndighedernes aldersgrænse for servering af alkohol, dog skal brugere indtaste deres fødselsdag når de oprettes i systemet.

I CC-Web vil det ikke være muligt for en virksomhed at slette et pubcrawl-event, samt virksomhedens lokation vil ikke blive vist på et digitalt kortsystem.

Det heller ikke være muligt for en virksomhed at tilbyde andre ydelser end muligheden for at afholde pubcrawls.

Udover ovenstående afgrænsningerne er der til slut i projektet blevet taget en afgrænsning i forhold til nedprioritering af IT-sikkerhed. Projektgruppen har implementeret en sikkerhed i form af at brugerens oplysninger omdannes til JSON Web Tokens i Web-API'et. Det er blevet opdaget at JSON Web Token, som fungerer efter hensigten i CC-App og Web-API'et, ikke kan fungere samtidig med de anvendte cookies på CC-Web. Det medførte at når en virksomhed loggede ind på CC-Web forblev de ikke loggede ind, hvilket er en essiel funktion for applikationen. Derfor er det blevet besluttet i projektgruppen at fjerne implementeringen JSON Web Tokens fra Web-API'et således at cookies på CC-Web fungerer efter hensigten. Dette er dog et punkt som der ønskes at arbejde videre med i det fremtidigt arbejde for CityCrawl.

I implementeringen af CC-Web har gruppen oplevet en del forudsete udfordringer fordi de ønskede features til web applikationen er blevet undervist i semesters fag sideløbende med projektforløbet. Det har betydet at implementeringen af CC-Web ikke har fuldt den estimerende tidsplan, som konsekvens af dette er antallet af unit test for applikationen blevet nedprioriteret.



## 7 Metoder

I dette afsnit vil projektets anvendte metoder blive beskrevet, samt hvordan metoderne er anvendt i projektet.

### 7.1 SOLID

Projektgruppen har arbejdet SOLID principperne for at gøre software systemet så forståeligt og simpelt som muligt. Selvom SOLID principerne forholder sig til koderelateret designs, kan visse dele af principper også bruges til front-end, som der er gjort brug af ved design af CC-Web og CC-App. For at gøre siden så simpel som muligt, er der blevet lavet sider, der som hovedregel kun har én funktion for simplicitetens skyld. Her gøres brug af kendskab til Single-responsibility principle (S).

Ved opbygning af back-end systemerne for CC-Web og CC-App, har fokus været på at holde funktionaliteten så simpel som mulig. Derved er der blevet lavet klasser som kun indeholder ansvar for en del af systemet. Herved overholder back-end altså også Single-responsibility principle.

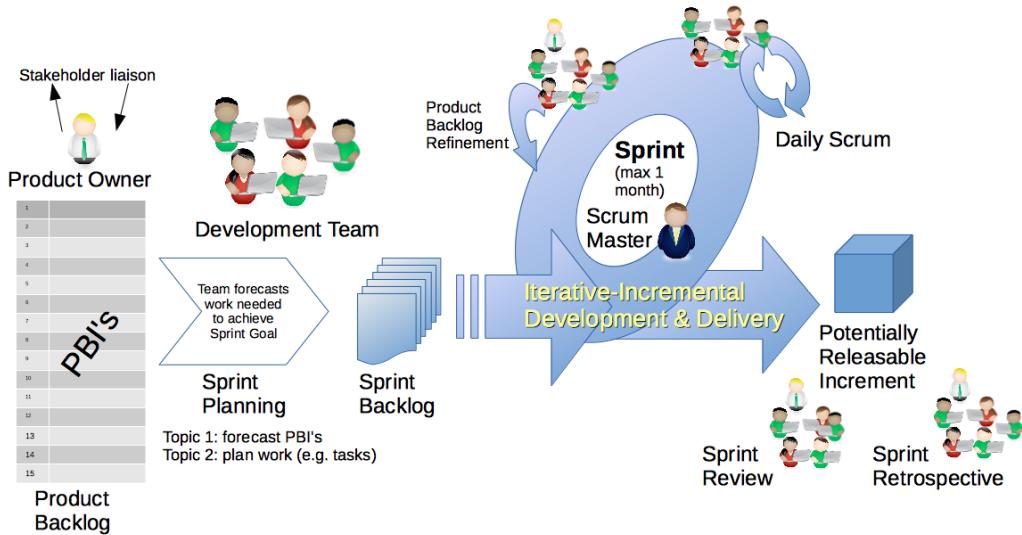
De forskellige front-end designs er oprettet således, at der ikke er behov for direkte ændring af siderne. Dog er der mulighed for at udvide dem med ekstra elementer, hvis det skulle vurderes nødvendigt. Open-closed principle (O) er altså overholdt.

Back-end delen af CC-Web og CC-App opsættes således at, en given funktionalitet skal fungere uden ændring. Det kan siges at en metode skal kunne virke vær gang, efter den er blevet oprettet. Dog skal der være mulighed for at udvide systemet, uden at den daværende funktionalitet ødelægges. Det vil sige at back-end delen opfylder Open-closed principle. Da CityCrawl er opsat med diverse unit tests, er der blevet oprettet abstrakte overbygninger for diverse klasser. Her er der valgt at lave et interface pr. klasse, i stedet for at lave et stort generelt interface. Herved undgås der i test, at implementere metoder som aldrig bliver brugt. Test delen har altså kun adgang til de metoder, der er relevante for den specifikke klasse der testes. Da er interface segregation principle (I) opfyldt. Endnu en del af at kunne teste CityCrawl, er der nødt til at blive brugt implementationer af metoder fra det konkrete hovedprojekt. Ved at skabe en abstrakt kobling, såsom et interface, mellem test- og hovedprojekt, vil der ikke være en afhængighed af kendskab mellem de to projekter og metoder. Derved opfyldes Dependency inversion principle (D). Liskov substitution principle (L) er ikke blevet brugt i CityCrawl. Dette skyldes hovedsageligt at der ikke bliver brugt noget arv eller polymorfi. Med andre ord indgår der ikke nogen subtype, og derved altså arv, som Liskov substitution principle afhænger af.

### 7.2 Scrum

Projektgruppen har arbejdet ud fra den agile arbejdsmetode Scrum, som har bidraget til et struktureret arbejdsforløb i projektperioden.

Scrum arbejdesprocessen er illustreret i figur 3, som er hentet fra *18, punkt 2*, der viser en typisk Scrum cyklus.

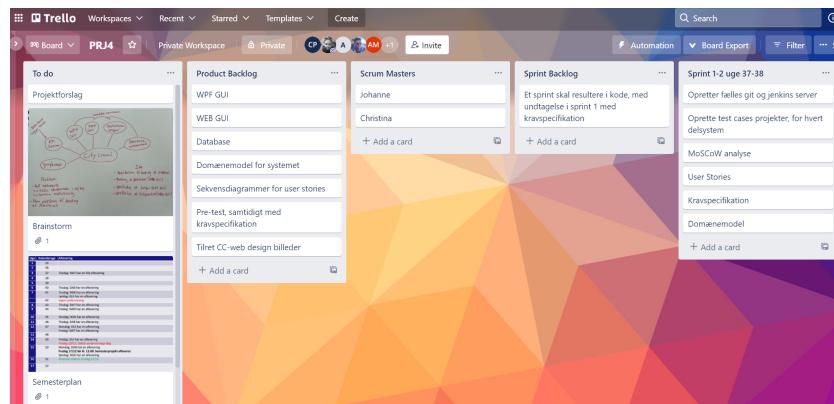


Figur 3: Oversigt over en typisk Scrum cyklus, der viser de aktuelle aktører i et Scrum forløb. Heraf ses det, at der tages emner fra *Product backloggen* og flyttes over i *Sprint backloggen*, som så udtages til de givne Scrum sprints. De forskellige sprints fungerer som en iterativ arbejdsproces. Efter et endt sprint vil der gradvist blive opbygget de essentielle dele, som gradvist skal danne det færdige produkt.

Projektgruppen har valgt at have to gruppemedlemmer som Scrum mastere, Johanne og Christina, hvor alle gruppemedlemmer har været aktive deltagere i selve Scrum teamet. Derudover har projektgruppen i samarbejde med projektgruppens vejleder fungeret som product owner af CityCrawl systemet. Projektgruppen har dannet Product backloggen, som indeholder systemets krav, ud fra user stories. På baggrund af dette er der løbende udvalgt emner til Sprint backloggen, som er udarbejdet i løbet af ca. 2-3 uger. Efter endt sprints er der løbende afholdt vejledermøder, som har givet løbende input til vurdering af projektarbejdet, der har været nødvendige for den gode fremdrift mod det endelige produkt.

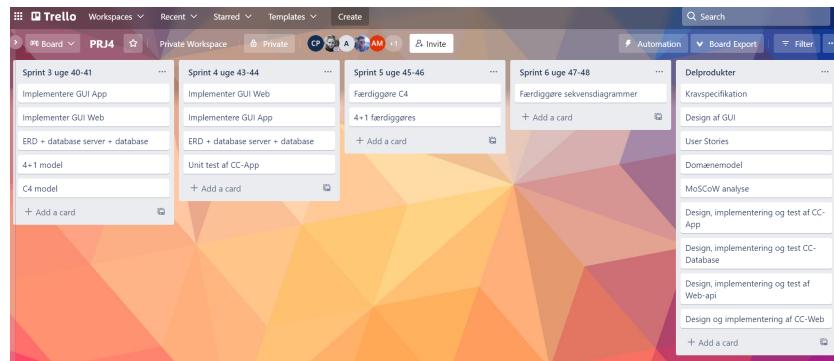
Afslutningsvis har projektgruppen anvendt et Trello board som *Scrum board*, for at holde en god struktur under de forskellige sprints i Scrum arbejdsprocessen.

I figur 5 vises et udklip af projektgruppens Scrum board fra starten af udviklingsfasen for CityCrawl.



Figur 4: Trello board fra starten af udviklingsfasen

I figur 5 vises et udklip af projektgruppens Scrum board mod slutningen af udviklingsfasen for CityCrawl.



Figur 5: Trello board mod slutningen af udviklingsfasen

### 7.3 Git

Projektgruppen har anvendt git for at opretholde et kontrolleret udviklingsforløb, hvor der løbende er blevet delt kode og dokumenteret på tværs af gruppemedlemmerne. Arbejdet med git har givet en mere struktureret og transparent arbejdssproces i projektet.

### 7.4 Continuous Integration (CI)

Projektarbejdet med unit test af CC-App er udarbejdet med fokus på continuous integration, for at prioritere en fremdrift i udviklingsfasen, der bygger på at kvalitetsikre koden, som sker automatisk efter hvert commit til projektgruppens fælles git repository. Brugen af continuous integration skal medføre en større sikkerhed for at undgå uden at det medfører en for stor arbejdsbyrde for udviklerne. Projektgruppen har lavet et webhook fra den fælles git til et job på en Jenkins server. Efter hvert commit til denne Jenkins server vil der ske en ny kørsel af clean, build, unit test og coverage, som også producerer en coverage rapport ud fra de opstillede unit test.

## 8 Teknologianalyse

I dette afsnit vil der være fokus på de teknologier der er brugt, samt overvejelserne der er blevet gjort om hvorfor denne teknologi er valgt.

### 8.1 Valg af platform

For at en aktuel bruger eller virksomhed, skal kunne gøre brug af CityCrawl, skal der være en form for brugergrænseflade. For at gøre platformen mest muligt tilgængeligt, er der valgt at ligge fokus på at have to forskellige brugergrænseflader. CityCrawl's brugergrænseflade består da af to dele, hvilket er en WPF applikation (CC-App), der skal kunne tilgås af en bruger og en web-applikation (CC-Web), som skal kunne tilgås af en virksomhed.

#### 8.1.1 WPF Applikation

I forbindelse med 4. semester, er der i faget GUI-programmering blevet anvendt WPF. Dette har givet en del inspiration i forbindelse med oprettelse af brugergrænsefladen. WPF giver mulighed for at skabe et let og overskueligt interface, som guider brugeren i forbindelse med mulighederne i applikationen. Via WPF er der mulighed for at lave flere vinduer, som skaber en god dynamik og nem tilgang til applikationen. Yderligere kan WPF styres af code behind, som kodes i C#, som gør det nemt at forme platformens funktionalitet.

#### 8.1.2 Web-applikation

Faget GUI-programmering har også givet mulighed for at bruge HTML, CSS og ASP.NET Core. Dette giver muligheder for at opsætte en hjemmeside. HTML og CSS gør det muligt at forme hjemmesiden, ved blot at beskrive indholdet i HTML og designet i CSS. På den måde er der en let tilgængelighed til at forme brugergrænsefladen. Hjemmesidens funktionalitet, ligesom med WPF, styres ved brug af C# via ASP.NET Core.

### 8.2 Valg af kodesprog

#### 8.2.1 C#

Ved 4. semester på Softwareteknologi-linjen, på Aarhus Universitet, bliver der aktivt brugt C# som hovedsprog. C# har mange fordele, da det er bygget til mange forskellige Frameworks, som gør det nemmere at bruge som programmør. C# bliver yderligere brugt i forbindelse med XAML og ASP.NET Core, som i denne sammenhæng er der, hvor der hovedsageligt bruges C#, i form af code behind GUI repræsentationen.

#### 8.2.2 XAML

XAML bliver brugt i forbindelse med CC-App, for at lave en Windows applikation. XAML har diverse kommandoer og tags, samt har mulighed for at kunne snakke sammen med C# kode, (som der kaldes code behind i denne sammenhæng) der gør det muligt at få det visuelle, til at agere på forskellige måder. XAML giver altså en visuel GUI, som er i stand til dynamisk, at ændre udseende løbende.

#### 8.2.3 HTML

Der er blevet brugt HTML som sprog, da dette er en nem og standard måde at opstille indhold på en hjemmeside, som netop skal bruges til CC-Web. HTML som kodesprog er simpelt og enkelt, hvor forskellige tags bruges til at bestemme indholdet på siden. Sproget er i sig selv forholdsvis simpelt, hvilket også er en fordel ved at vælge det.

#### 8.2.4 CSS

CSS (Cascading Style Sheets) bruges som stylingsprog, der kan lave design i forbindelse med HTML. Sproget i sig selv kan ikke stå alene, da det kræves at der skal være elementer der kan styles. Ligesom



HTML kan CSS godt beskrive styling af elementer som ikke finde i det tilknyttet HTML dokument. Derudover kan et CSS dokument tilknyttes flere HTML dokumenter, hvor forskellige og generelle styles bruges.

#### **8.2.5 ASP.NET Core (MVC)**

Der gøres brug af ASP.NET Core Framework, på MVC form. Med dette Framework kan der sammenstættes, blandt andet, HTML, CSS, C# og EF Core, som der ønskes at gøre brug af. Ved brug af MVC, gøres det muligt at lave Models, Views og Controllers, og dette skaber nemt et godt overblik. MVC giver også muligheden for at lave funktionaliteter på hjemmesiden, samt at skabe forbindelser mellem hjemmeside og database. Yderligere kan der gøres brug af Identity, som via cookies kan oprette en bruger, og ved login gøre brug af denne bruger og dens informationer. Yderligere kan der anvendes Web-API, som gør at WPF applikationen kan snakke sammen med databasen.

#### **8.2.6 Relationel Database**

Det er blevet besluttet at udarbejde CC-Database som en relationel database. Det er blevet besluttet både fordi en relationel database er den mest udbredte database model, blandt andet fordi det er nemt at tilgå data. Det er også nemt at påsætte sikkerhed på en relationel database fordi den er opbygget som tabeller og på den måde kan man kræve sikkerhed på udvalgte tabeller.

#### **8.2.7 Entity Framework Core**

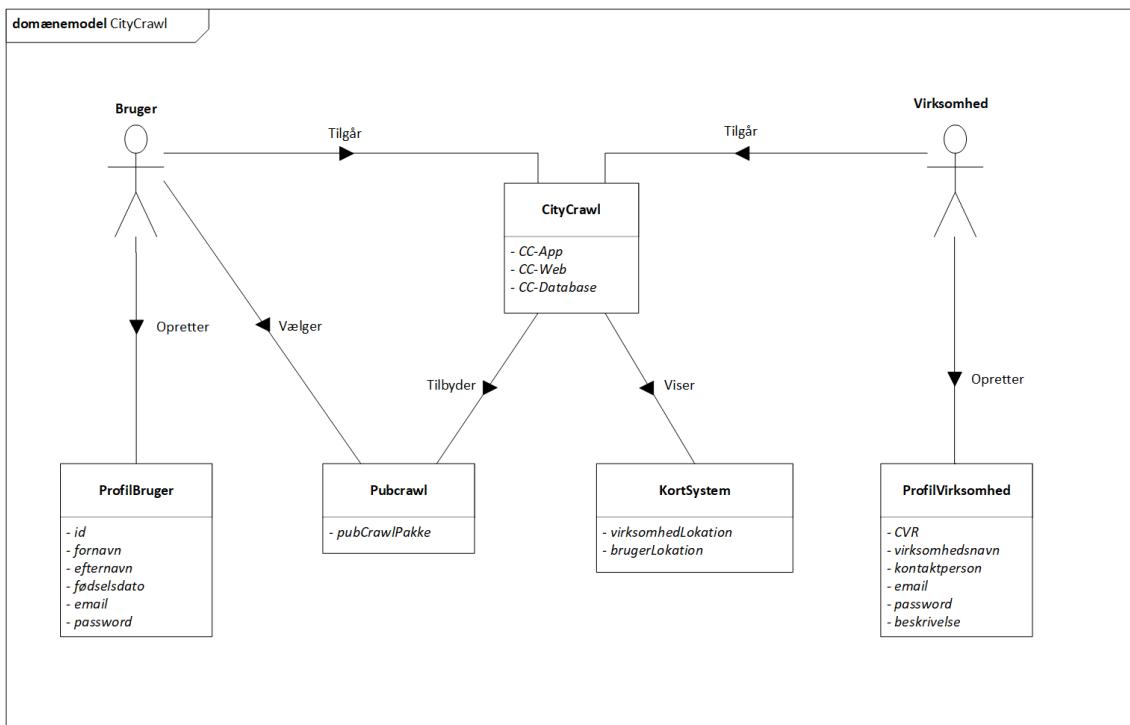
For at implementere af CC-Database er det blevet valgt at anvende Entity Framework Core, også kendt som EF Core. Valget skyldes at EF Core er et fleksibelt data access tool som kan bruges til at tilgå data i en database. Dette giver mulighed for at bruge en .NET tilgangsmetode, og dermed slipper man for selv at skrive SQL queries.

EF Core gør det altså nemt at tilpasse, udvide og tilrette, fordi frameworket laver det bagomlæggende arbejde og man blot selv skal udarbejde de ønskede Model klasser.

## 9 Systemarkitektur

I dette afsnit vil systemarkitekturen for CityCrawl beskrives i form af en domænemodel, en C4-model der beskriver de relevante indgangsvinkler til CityCrawl og afslutningsvis en 4+1 ViewModel.

Figur 6 viser domænemodellen for CityCrawl, som er lavet ud fra en system domæneanalyse, hvor de vigtigste begreber og relationer for CityCrawl er blevet beskrevet. Denne model bruges til at få dannet en bro mellem det opsatte krav, omverdenen og implementation.

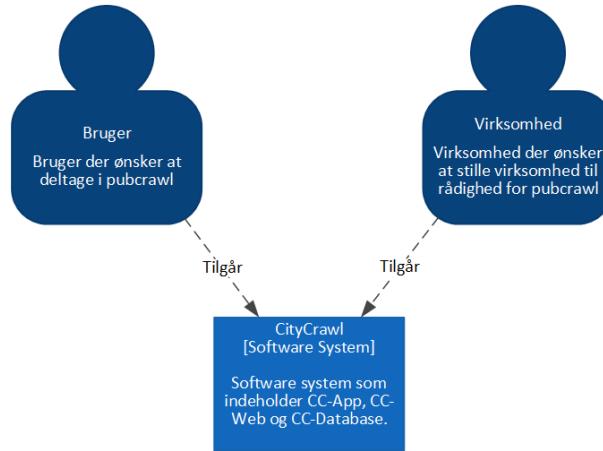


Figur 6: Domænemodel for CityCrawl

### 9.1 C4-model af CityCrawl

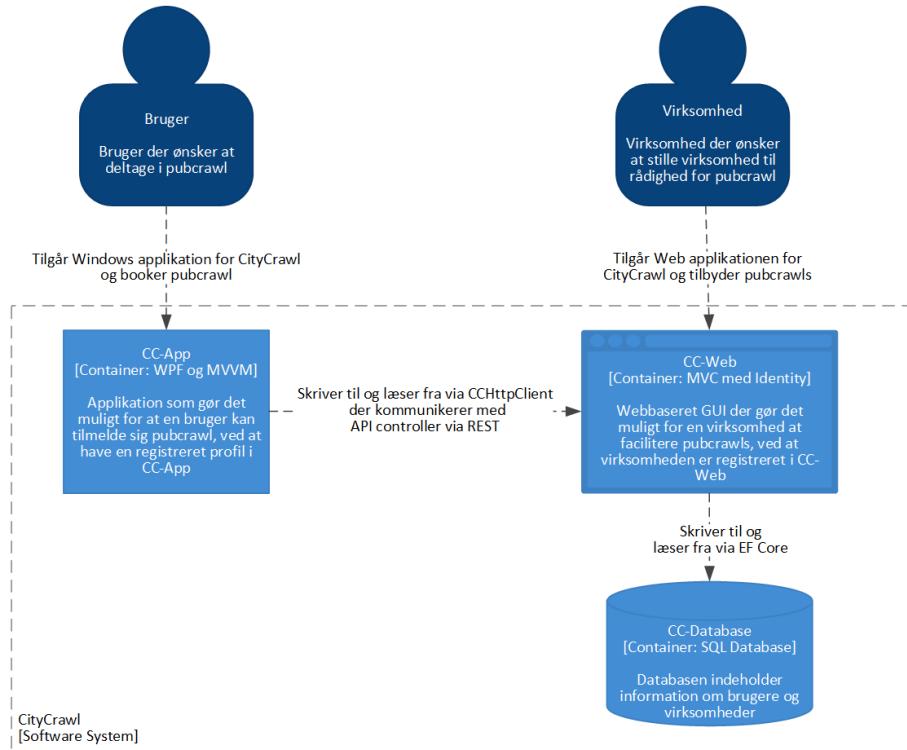
C4-modellen anvendes for at give et visuelt overblik over softwarearkitekturen af CityCrawl. Projektgruppen har valgt at illustrere tre ud af de fire niveauer af C4-modellen, idet der ikke kigges på det sidste lag, som omhandler kode. Heraf vil der først kigges på Context som er det yderste niveau af C4 modellen som skal vise hvilke aktører og eventuelle eksterne systemer, der tilgår CityCrawl. Dernæst kigges der ind i CityCrawl systemet, hvor der kigges på de Containers som indgår i CityCrawl. Dette niveau viser den interne kommunikation mellem de forskellige containere i CityCrawl. Derefter vil der kigges på relevante komponenter og deres interne interaktioner. Heraf kigges der på CC-App komponentet og CC-Web komponentet, som skal give et indblik i, hvordan de forskellige komponenter og aktører interagerer med hinanden.

Figur 7 viser Context niveauet for CityCrawl systemet via C4-modellen, som illustrerer hvordan aktørene interagerer med CityCrawl.



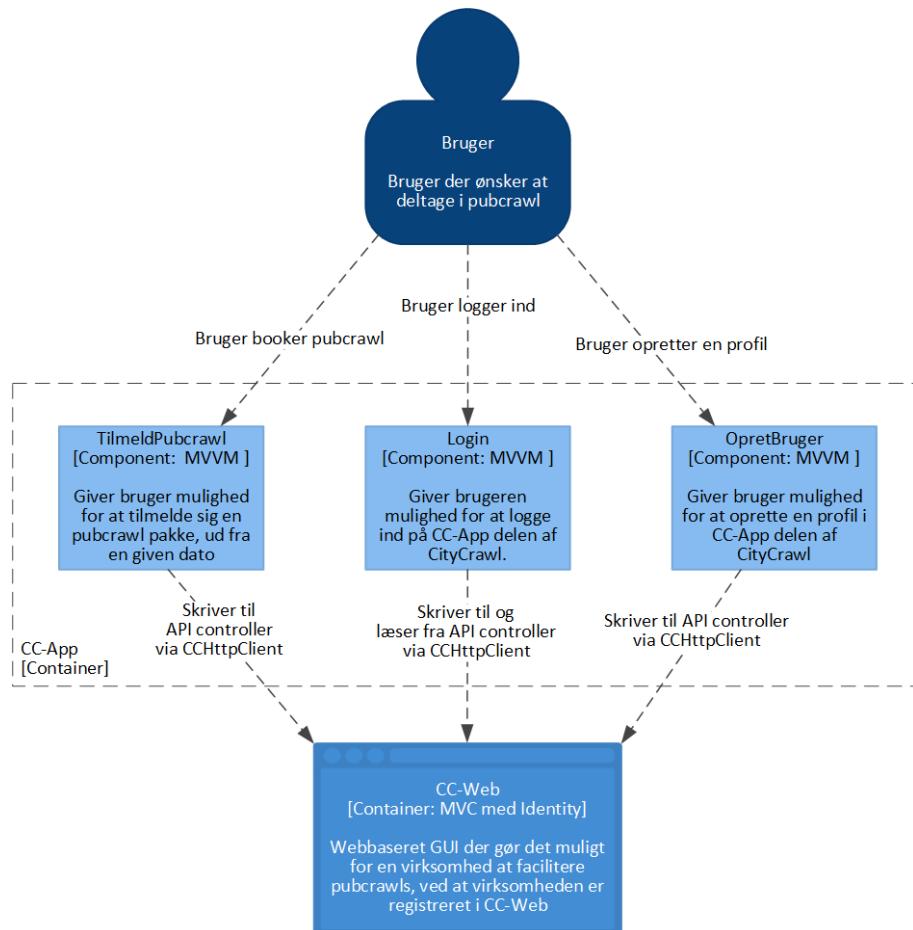
Figur 7: Niveau 1: System Context diagram for CityCrawl viser, at en bruger og en virksomhed skal kunne interagere med CityCrawl, samt at CityCrawl er et software system, som indeholder en WPF applikation, en Web applikation, og en Database.

Figur 8 viser Container niveauet for CityCrawl systemet i C4-modellen, her tydeliggøres det hvilke delsystemer CityCrawl består af.



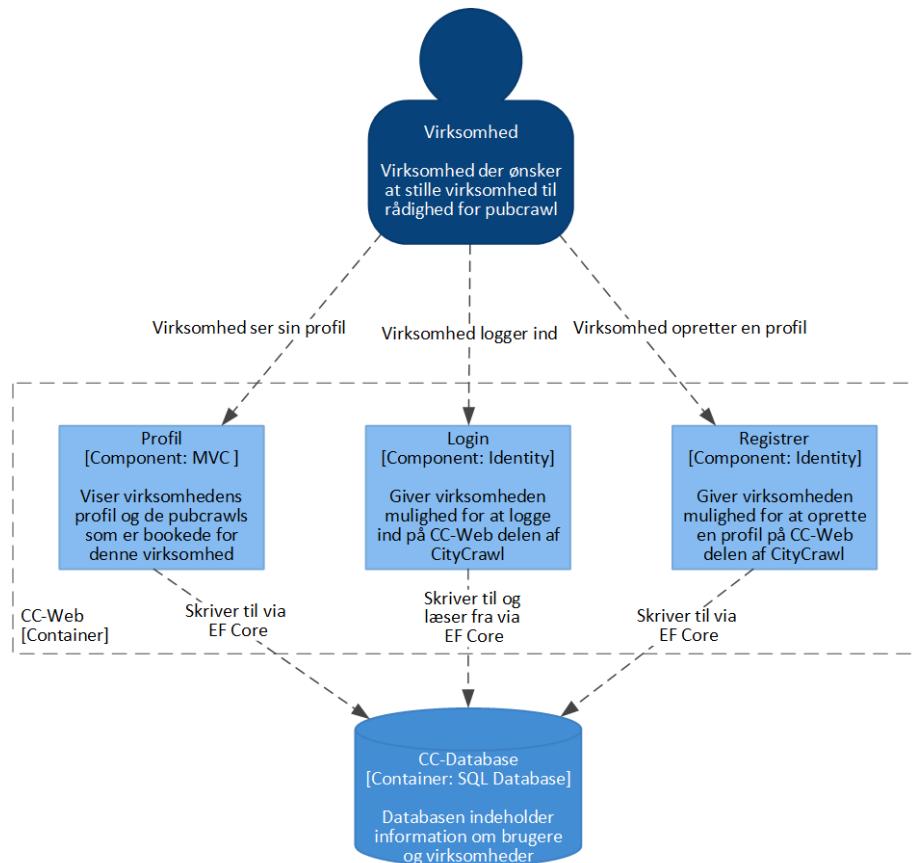
Figur 8: Niveau 2 af C4-modellen: Container diagram for CityCrawl viser, at brugeren tilgår CC-App, og virksomheden tilgår CC-Web, samt der vises, hvordan de respektive forbindelser til CC-Database fra både CC-App og CC-Web.

Figur 9 viser Component niveauet for CC-App containeren i C4-modellen for CityCrawl systemet. Dette niveau giver et visuelt indblik i hvordan brugeren kan til gå CC-App.



Figur 9: Niveau 3 for CC-App af C4-modellen: Component diagram for CC-App viser, hvordan de forskellige MVVM komponenter i CC-App interagerer med CC-Database, hvilken er en separat container i CityCrawl. Denne kommunikation til CC-Database containeren gør det muligt, at brugeren kan oprette en profil, logge ind og tilmelde en pubcrawl pakke ud fra en bestemt dato.

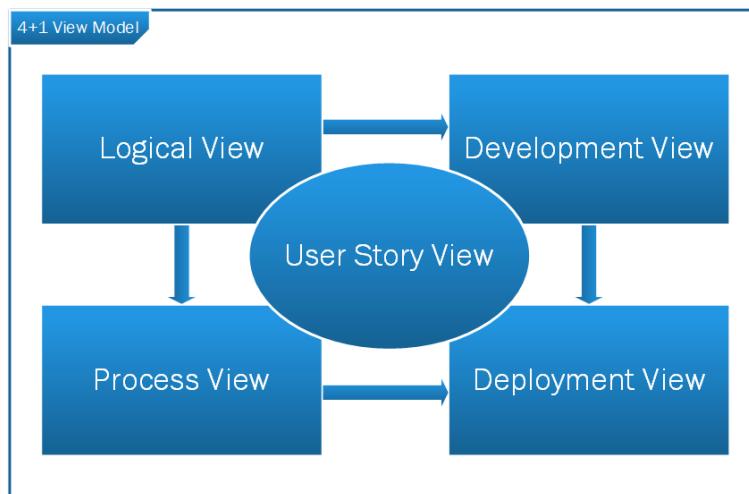
Figur 10 viser Component niveauet for CC-Web containeren i C4-modellen for CityCrawl systemet. Dette niveau giver et visuelt indblik i hvordan brugeren kan til gå CC-Web.



Figur 10: Niveau 3 for CC-Web af c4-modellen: Component diagram for CC-Web viser, hvordan MVC komponentet og hvordan de to Identity komponenter i CC-Web containeren kommunikerer med CC-Database containeren. Heraf illustrerer denne kommunikation, hvordan virksomheden har mulighed for at oprette en profil, logge ind og se sin profil. Under virksomhedens profil vil virksomheden kunne se, hvilke pubcrawls der er bookede.

## 9.2 4+1 view model for CityCrawl

I det følgende vil 4+1 view modellen for CityCrawl systemet beskrives, hvoraf der kigges på forskellige indgangsvinkler til systemet. Først vil User Story Viewet beskrives, som giver indblik i både brugerens og virksomhedens tilgang, bestemt ud fra de fastlagte User Stories under kravspecifikationen. Efterfølgende vil det logiske View af CityCrawl beskrives, som skal danne et indgangsblik i selve funktionaliteten af både CC-App og CC-Web, med fokus på MVVM og MVC metoderne. Heraf vil der også være en beskrivelse af et Deployment View, som skal beskrive, hvordan projektgruppen har planlagt at udvikle samarbejdet mellem delsystemerne i CityCrawl. Projektgruppen har valgt ikke at beskrive CityCrawls Process View og Development View, idet der ikke udarbejdes et flertrådet system.

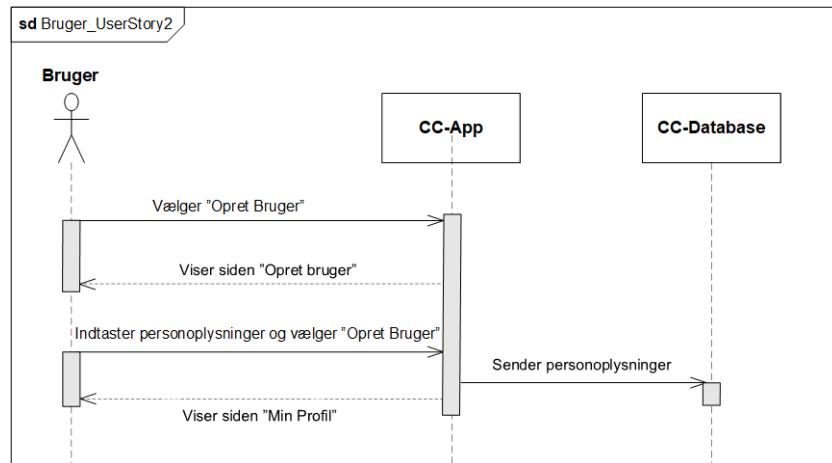


Figur 11: 4+1 ViewModel (fra 4+1view-architecture\_UML2.pdf fra lektion 5 i Software Design faget)

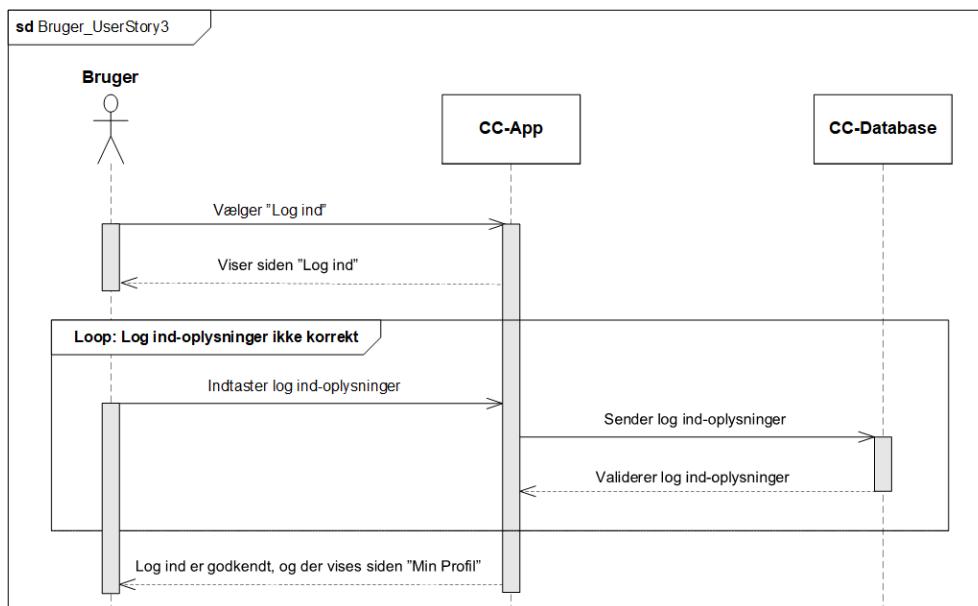
### 9.2.1 User Story View

I dette afsnit vil der være en beskrivelse af User Story Viewet for systemarkitekturen, i form af sekvensdiagrammer for de forskellige user stories for både brugeren og virksomheden. For brugeren er der udvalgt sekvensdiagrammerne ”Opret bruger på CC-App”, ”Log ind på CC-App”. Derudover er der også medtaget sekvensdiagrammerne ”Book pubcrawl til bestemt dato” og ”Vælge pubcrawlspakke med prædefinerede virksomheder” som er del af et sammenhængende diagram for brugerens handlinger. For virksomheden er der udvalgt sekvensdiagrammerne ”Logge ind på CC-Web” og ”Opret som virksomhed på CC-Web”. Disse sekvensdiagrammer er valgt for at vise funktionaliteten af den første udgave af CityCrawl.

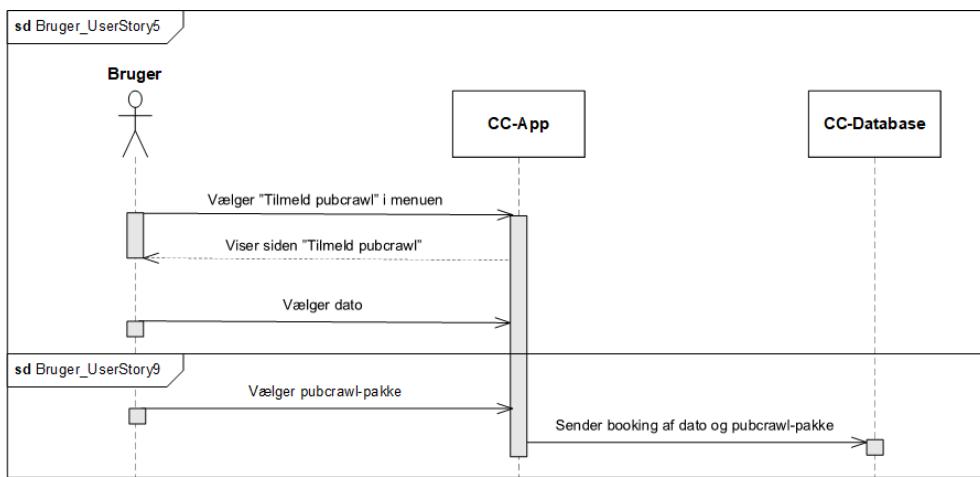
### User stories for bruger



Figur 12: Sekvensdiagram for User Story: ”Opret bruger på CC-App”. Brugeren trykker på *Opret bruger* knappen, og får vist modalvinduet for *Opret bruger*. Brugeren indtaster dernæst sine bruger-oplysninger og afslutter ved at trykke på *Opret bruger* knappen. Bruger-oplysninger sendes til database. Brugeren får vist modalvinduet for *Min profil*.

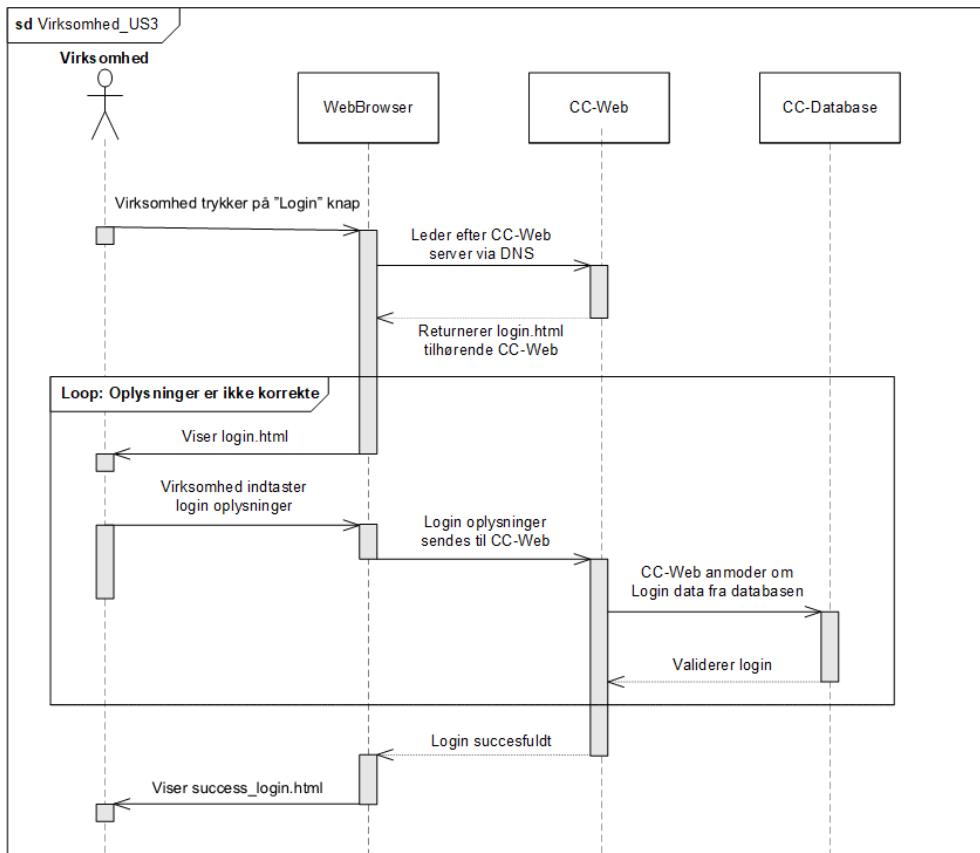


Figur 13: Sekvensdiagram for User Story: ”Log ind på CC-App”. Brugeren trykker på *Log ind* knappen, og får vist modalvinduet for *Log ind*. Brugeren indtaster dernæst sine log ind-oplysninger og afslutter ved at trykke på *Log ind* knappen. Log ind oplysninger sendes til database og valideres. Hvis log ind-oplysningerne ikke er korrekt, bedes brugeren om at indtaste log ind-oplysninger igen. Er log ind-oplysninger korrekte, så får brugeren vist modalvinduet for *Min profil*.

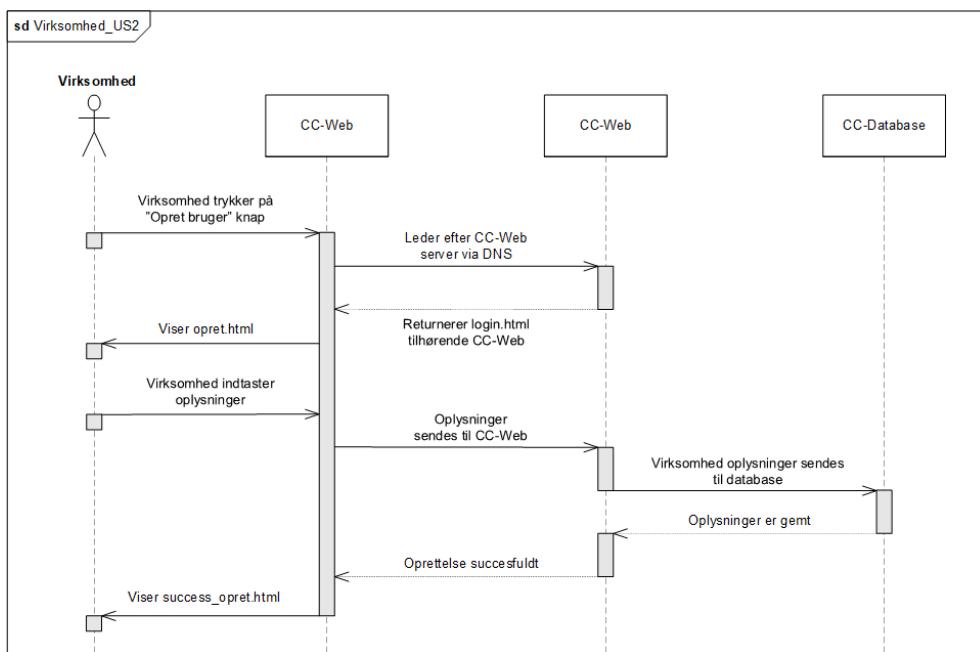


Figur 14: Sekvensdiagram for User Story: ”Book pubcrawl til bestemt dato” og ”vælge pubcrawlpakke” med prædefinerede virksomheder. Brugeren vælger *Tilmeld pubcrawl* fra menuen. Brugeren får vist modalvinduet *Tilmeld pubcrawl*. Brugeren vælger en dato for pubcrawl. Brugeren vælger en pubcrawl pakke, hvor valgte dato og pakke sendes til databasen.

### User stories for virksomhed



Figur 15: Sekvensdiagram for User Story: ”Logge ind på CC-Web”. Virksomheden trykker på *Login* knappen og bliver taget til *Login* siden. Virksomheden indtaster sine loginoplysninger. CC-Web validerer at loginoplysninger passer på databasen. Virksomheden får vist succesfuldt login.



Figur 16: Sekvensdiagram for User Story: ”Opret som virksomhed på CC-Web”. Virksomheden trykker på *Opret bruger* knappen og bliver taget til *Opret bruger* siden. Virksomheden indtaster sine bruger-oplysninger. CC-Web sender bruger-oplysninger til databasen. Virksomheden får vist succesfuldt oprettet bruger.

De resterede sekvensdiagrammer for user stories kan findes i bilagsrapporten under afsnittet for Systemarkitektur.

### 9.2.2 Logical View

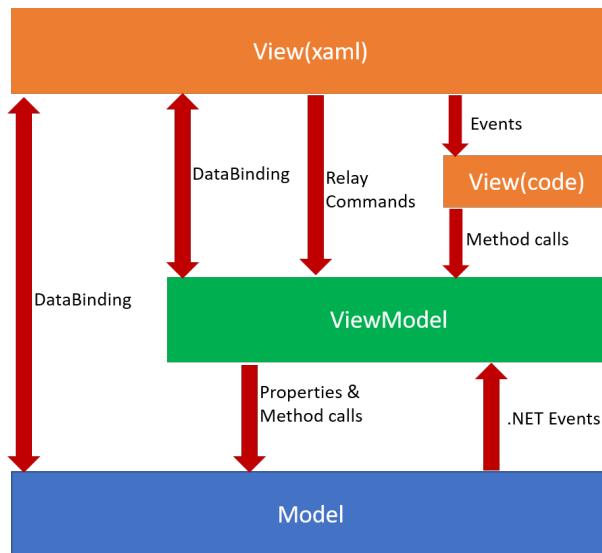
I dette afsnit vil det Logiske View af 4+1 modellen beskrives. Det logiske View omhandler de funktionaliteter, der har betydning for en bruger eller en virksomhed, som anvender CityCrawl systemet.

#### MVVM (Model-View-ViewModel)

MVVM bruges for at danne en god separation mellem View og ViewModel i bl.a. WPF. Heraf kan al code behind flyttes fra View filerne til ViewModel klasserne. Dette gør, at man får dannet en adskillelse mellem bruger grænsefladen og den bagvedliggende logik, som medbringer, at det bliver mere overskueligt at opretholde en god bruger grænseflade. Ved brug af denne adskillelse medfører det også at udviklerne har mulighed for at udføre test på ViewModel klasserne, uden at der tænkes på selve håndteringen af bruger grænsefladen.

MVVM bliver anvendt i CC-App, for at opdele CC-Apps funktionaliteter i en Model del, en View del og en ViewModel del, hvor ViewModel klasserne har ansvar for funktionaliteterne der skal ske på de tilhørende Views, samt at Model klasserne fungerer som domæne klasser som indeholder anvendt data.

I figur 17 (fra GUI MVVM.pdf, lektion 07 MVVM i GUI-programmering faget) illustreres strukturopbygningen af MVVM, hvor View delen indeholder XAML koden for de forskellige implementerede vinduer i CC-App, som er forbundet til ViewModel klasserne via databinding. Denne databinding oprettes ved, at DataContext propertien i View har en reference til det pågældende ViewModel, som det skal kommunikere med. Håndtering af events fra de forskellige vinduer gøres med brug af databinding til diverse Commands i ViewModel klasserne. ViewModel klasserne vil heraf kommunikere med Model klasserne for at tilgå nødvendig data i CityCrawl.



Figur 17: MVVM strukturen som viser at Views bestående af XAML kommunikerer med code behind delen af Views via events. Derefter illustreres det at Views kommunikerer med ViewModel klasserne og Model klasserne ved brug af databinding. Derudover kan det ses at ViewModel klasserne kommunikerer med Model klasserne ved at sende metodekald, som så svares tilbage via .NET events

#### MVC (Model-View-Controller)

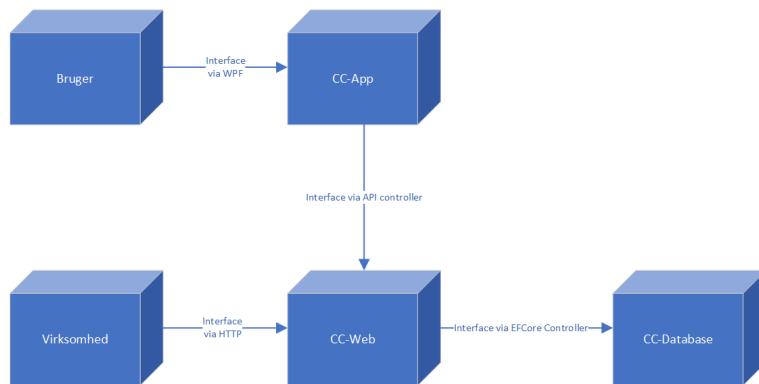
MVC anvendes i CC-Web for at forbinde Views, som er C# HTML sider via controlleren, der igen anvender data fra Models, iform af både entitet klasser fra CC-Database og alm. C# klasser. Heraf er Model delen af MVC ansvarlig for at håndtere det data der sendes ind fra brugergrænsefladen, som sendes ind til Model klasserne via en controller. Controller aspektet af MVC står for at være kommunikationslaget mellem Views og Models. Denne kommunikation opretholdes ved, at de forskellige Views kommunikerer til controlleren med brugerinput. Disse input behandles i controlleren, således at controlleren udfører den ønskede handling med brug af de forskellige data modeller. Derudover sørger controlleren også for at sende nødvendig information fra Model klasserne til View-siderne, således at det forventede data vises, i

korrekt HTML format.

### 9.2.3 Deployment View

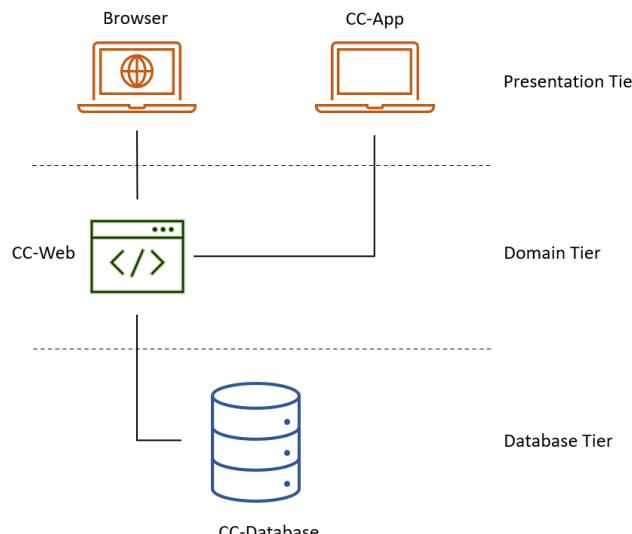
I dette afsnit vil Viewet for deployment blive beskrevet.

Figur 18 illustrerer Deployment Viewet for CityCrawl. Blokkene i Viewet er elementerne i systemet set i forhold til den tekniske infrastruktur. Figuren viser, at brugeren kun kan tilgå CC-App og virksomheden kun kan tilgå CC-Web. Det kan også ses at CC-App kan kommunikere med CC-Web's implementeret Web-API, som så kan kommunikere med CC-Database.



Figur 18: Deployment View

Figur 19 viser 3-tier arkitekturlaget for CityCrawl. Disse tre tier er opdelt i et Presentation Tier, Domain Tier samt en Database Tier. Præsentationslaget er den tier, der viser de applikationer som brugeren og virksomheden interagerer med. Denne tier er altså hvor der sendes og læses data fra brugeren. I domænelaget bliver data fra præsentationslaget processeret. Denne tier kan også læse og skrive til databaselaget. Databaselaget, er den tier hvor det processerede data bliver oplagret. Al kommunikationen mellem de 3 tier går altid igennem domænelaget, hvilket gør, at præsentationslaget og databaselaget ikke kan kommunikere direkte med hinanden.



Figur 19: 3-tier arkitekturlaget for CityCrawl illustrerer at CC-App kommunikerer med CC-Database via CC-Web, og at CC-Web kan kommunikere direkte med CC-Database. Derudover viser denne 3-tier struktur også at virksomheden tilgår CC-Web via browseren, hvorimod at brugeren har direkte tilgang til CC-App.

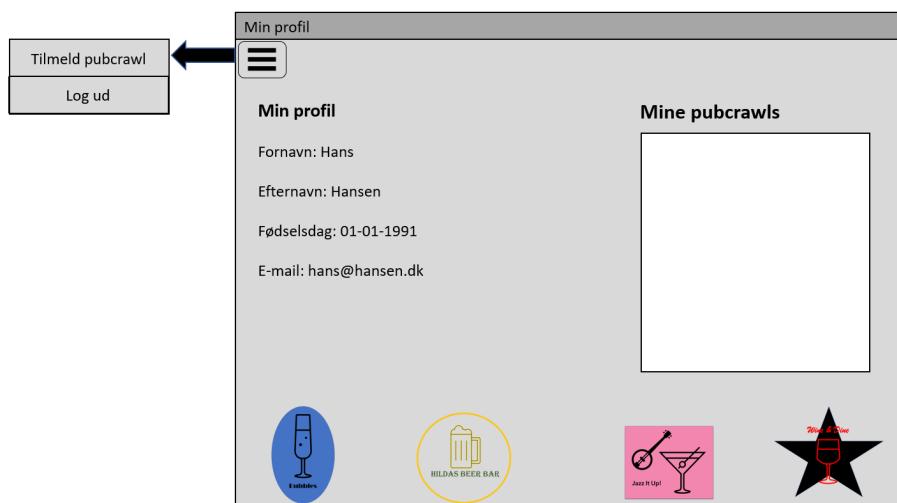
## 10 CC-App

### 10.1 Design af CC-App

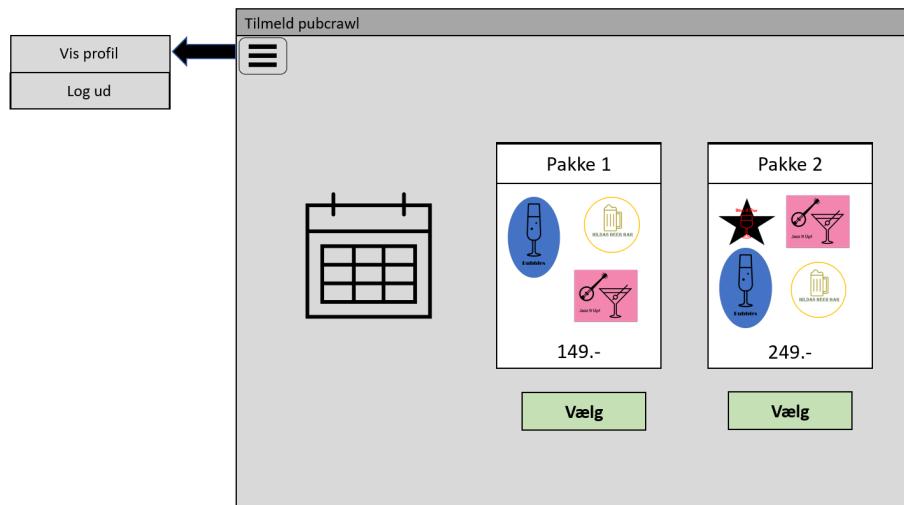
I det følgende afsnit vil designovervejelserne vedrørende CC-App blive beskrevet. Projektgruppen har valgt at danne CC-App som en WPF applikation, hvor en bruger kan anvende CityCrawl til at tilmelde pubcrawls ud fra en given dato. Heraf skal brugeren kunne oprette en profil, som gemmes i CC-Database, og efterfølgende vil brugeren kunne logge ind på CC-App, hvor brugeren kan se sine indtastede oplysninger. Derefter kan brugeren tilmelde sig fx to forskellige pakkeløsninger af pubcrawls, som også gemmes i CC-Database og vises under brugerens profil.



Figur 20: Design af CC-App forside. CityCrawl logo er repræsenteret stort og tydeligt i midten af forsiden med to knapper til hhv. *Login* og *Opret bruger*. Disse skal herefter kunne åbne de givne modalvinduer for henholdsvis *Login* og *Opret bruger*. Disse modalvinduer illustreres i bilagsrapporten afsnit ”Design af CC-App”, figur 24 og 26



Figur 21: Design af CC-App min profil viser et design, hvor brugeren skal kunne se sine indtastede oplysninger, samt tilmeldte pubcrawls. Heraf skal brugeren kunne skifte til tilmeld pubcrawl vinduet (figur 22) eller gå tilbage til forsiden af CC-App (figur 20).



Figur 22: Design af CC-App tilmeld pubcrawl viser design af vinduet, hvor brugeren skal kunne vælge en specifik dato og hernæst en pubcrawl-pakkeløsning, som så skal registreres under den pågældende bruger. Derudover skal brugeren have mulighed for at gå tilbage til min profil vinduet (figur 21) eller tilbage til forsiden (figur 20).

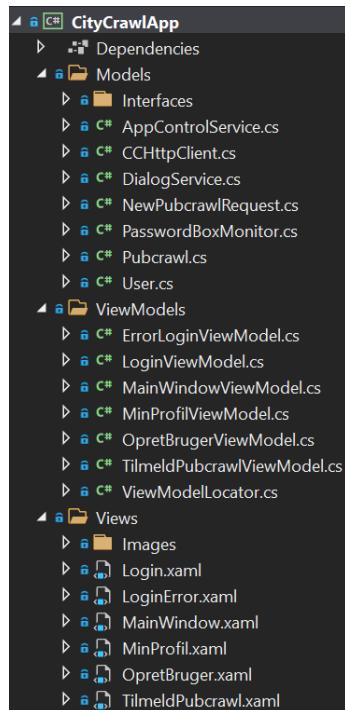
De resterede designbilleder for CC-App kan findes i bilagsrapporten under afsnittet for CC-App.



## 10.2 Implementering af CC-App

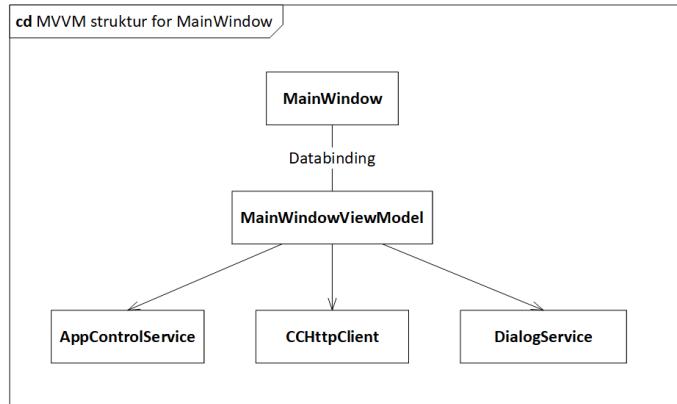
I dette afsnit vil det beskrives hvordan CC-App er implementeret, og hvilke overvejelser der har været under udviklingsprocessen. Projektgruppen har valgt at anvende softwarearkitektur mønstret MVVM (Model-View-ViewModel), som gør brug af databinding på tværs af Views og ViewModel kasserne, hvoraf at Views er XAML vinduerne i WPF, og ViewModel kasserne kommunikerer med de forskellige Views og de nødvendige Model klasser. Heraf har projektgruppen oprettet en bindingsklasse, kaldet ViewModelLocator.cs, som anvendes for at binde de forskellige Views til ViewModel kasserne.

MVVM mappestructuren af CC-App vises i figur 23



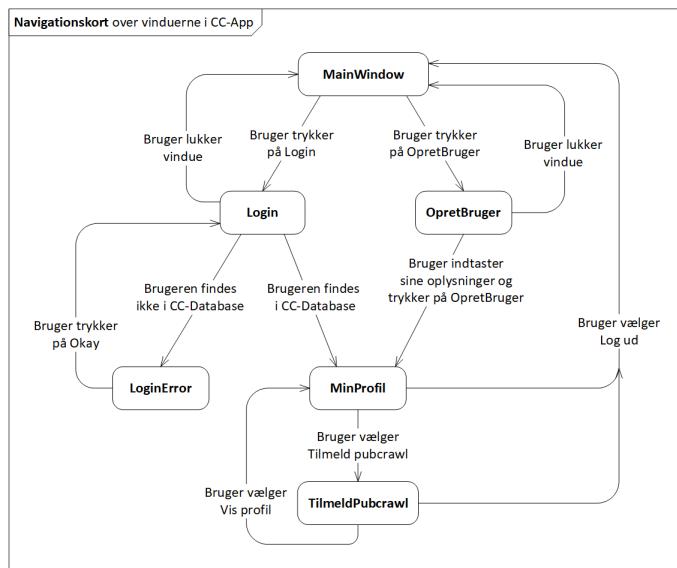
Figur 23: MVVM mappestruktur af CC-App, som viser at der er en ViewModel klasse som kommunikerer med et tilsvarende View.

I figur 24 illustreres et UML klassediagram af MVVM strukturen af MainWindow som skal vise et eksempel på MVVM strukturen i CC-App.



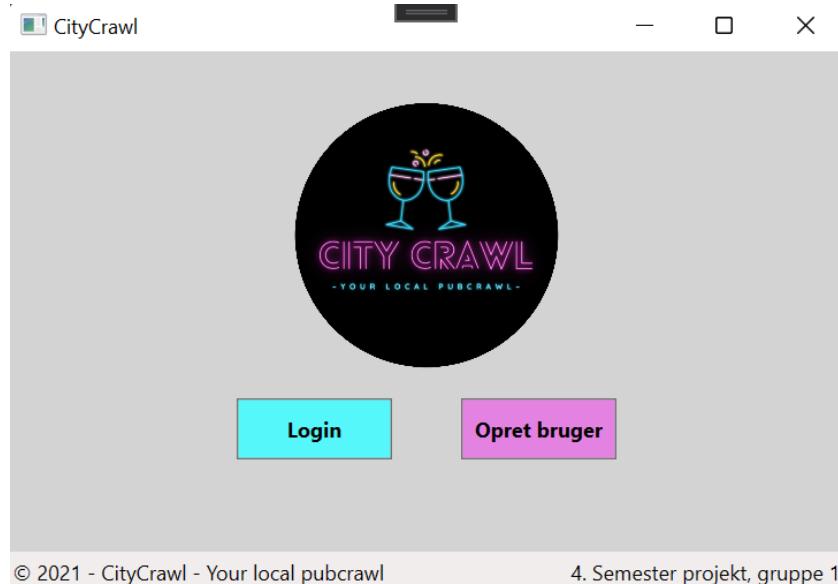
Figur 24: Klassediagram for MVVM strukturen af MainWindow, der viser at Viewet MainWindow kommunikerer med ViewModel klassen MainWindowViewModel via databinding. Derudover illustreres det at MainWindowViewModel klassen anvender data fra Model klasserne AppControlService, CCHttpClient og DialogService.

I det følgende vises et navigationskort over vinduerne i CC-App, som illustrerer hvordan de forskellige vinduer tilgas af brugeren.



Figur 25: Navigationskort af vinduerne i CC-App, som illustrerer hvordan en bruger kan tilgå de forskellige Views i CC-App.

Den implementerede forside af CC-App er illustreret i figur 26.



Figur 26: Den implementerede CC-App forside viser brugerens førstehåndsindtryk af CC-App. Her kan brugeren enten vælge at blive oprettet og efterfølgende at logge ind, eller direkte logge ind, hvis brugeren allerede findes i CC-Database.

CC-App indeholder seks Views og seks ViewModels, som er bundet sammen via ViewModelLocator.cs, samt tre Model klasser. De seks Views repræsenterer CC-App forsiden og de fem resterende modalvinduer. De seks ViewModel klasser indeholder al funktionalitet, der optræder som code behind funktionaliteten til alle vinduerne. Dertil fungerer de tre Model klasser som alm. C# klasser, der indeholder nødvendig data, som anvendes på tværs af ViewModel klasserne. Heraf indeholder Model klasserne også *data access* klasser som tilsvarer de entiteter der findes i CC-Database.

CC-App kommunikerer med CC-Database ved, at metoderne i CCHttpClient klassen kommunikerer med Web-API controlleren i CC-Web. Denne kommunikation beskrives nærmere i afsnit 10.4 (CCHttpClient) og afsnit 11 (Web-API).

## 10.3 Test af CC-App

I dette afsnit vil der kigges på forskellige unit test og systemtest af CC-App, med fokus på brugerens tilgang til CC-App, samt en efterfølgende coverage rapport, med brug af en Jenkins server.

### 10.3.1 Unit test af CC-App

Unit test af CC-App er blevet udført ved at designe koden på en måde som gør det muligt at unit teste, fx ved at anvende dependency injection. Dependency injection sørger for at al funktionalitet der omhandler CCHttpClient og visning af modal vinduer, flyttes ud i separate klasser med tilhørende interfaces. Heraf kan disse interfaces mockes, som medfører at det er muligt at danne unit test. Koden er struktureret så unit test er muligt, og derved opretholdes SOLID designprincippet Single Responsible Principle.

De enkelte unit test er dannet via NUnit med brug af NSubstitute, som anvendes for at danne mocks af CCHttpClient klassen og DisplayServices klassen. Dette medfører at der er opnået 100% test dækning af ViewModel klasserne, som indeholder den primære code behind funktionalitet for WPF vinduerne. Heraf testes hverken CCHttpClient eller DisplayServices klasserne ikke, idet de er mocket.

Projektgruppen har anvendt continuous integration for at køre de forskellige unit test af CC-App kontinuerligt. Derudover er projektet sat op til at køre test hver gang der pushes op til gruppens fælles GitHub.

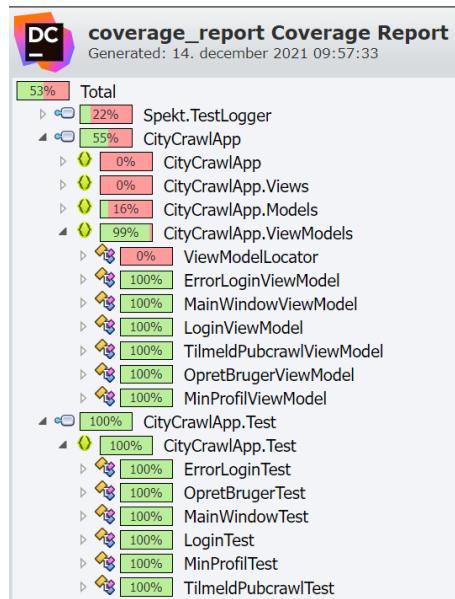
I figur 27 illustreres en kørsel for clean build, build, unit test kørsel, coverage kørsel, og en efterfølgende coverage rapport. Figuren viser at Jenkins server jobbet henter nødvendig information fra projektgruppens fælles GitHub, hvorefter der bygges og dannes coverage rapport af unit test.

### Stage View

	Clean Workspace	Fetch from Git	Clean Build	Build	Run coverage of unit tests	Publish Test Results	Publish Coverage Results
Average stage times: (Average full run time: ~37s)	92ms	8s	2s	5s	16s	188ms	7s
#84 Dec 14 09:57 1 commit	94ms	7s	1s	4s	16s	188ms	7s
#83 Dec 14 09:52 1 commit	94ms	8s	2s	7s failed			

Figur 27: Build og test af unit test for CC-App ved brug af Continuous Integration. Heraf ses det at der er sket en successfuld kørsel og test af unit test i CC-App, hvor der er dannet en tilhørende coverage rapport. Denne coverage rapport viser hvor meget af koden, som er dækket af test.

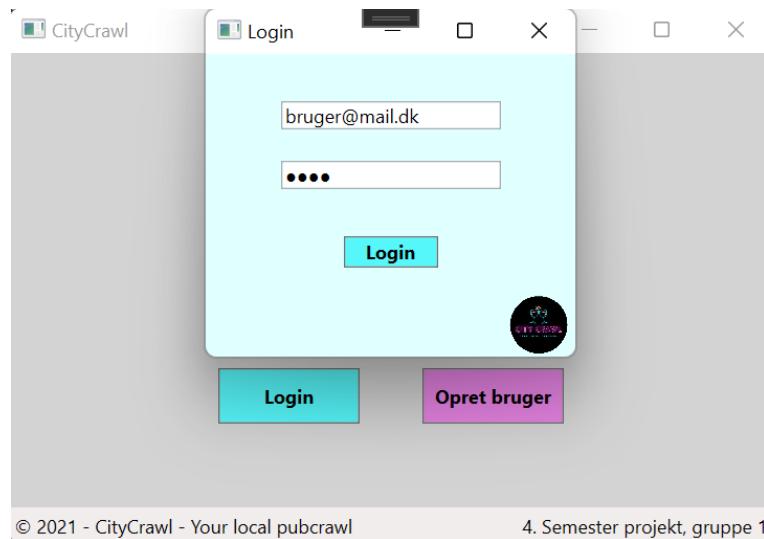
I figur 28 vises den udførte coverage rapport for unit test af CC-App, hvoraf det vises at der er udført unit test med 100% dækning af ViewModel klasserne.



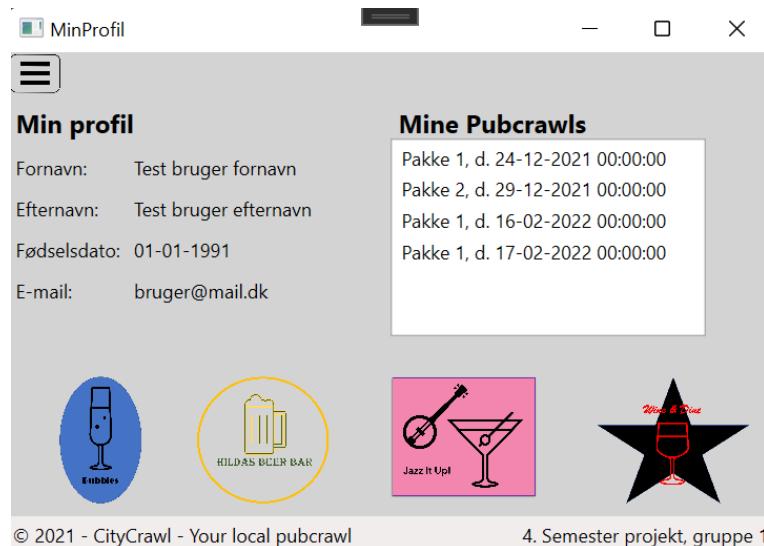
Figur 28: Coverage rapport for CC-App ved brug af Continuous Integration. Coverage rapporten viser at der er 100% dækning af ViewModel klasserne, som hænger godt sammen med at der er 55% dækning af hele CC-App.

### 10.3.2 Systemtest af CC-App

Systemtest af CC-App skal vise at en bruger kan oprette en profil, logge ind i CC-App med den registrerede e-mail og password, og efterfølgende tilmelde sig forskellige pubcrawls. Udførelse af denne systemtest er illustreret i figur 29 og 30.



Figur 29: Den implementerede CC-App forside med det tilhørende login modalvindue, som åbnes, når brugeren har trykket på login knappen, og herefter vil brugeren logge ind i CC-App, hvis brugeren kendes af CC-Database. Ellers vil brugeren modtage en besked om at genindtaste sine loginoplysninger via et nyt modalvindue.



Figur 30: Den implementerede CC-App min profil side viser brugerens profil i CC-App, samt brugerens nuværende tilmeldte pubcrawls. Brugeren kan fra Minprofil siden enten komme videre til *TilmeldPubcrawl* siden eller tilbage til forsiden via menu-ikonet.

Projektgruppen kan konkludere at CC-App er en velfungerende WPF applikation som tillader at en bruger kan logge ind, oprette en profil og tilmelde forskellige pubcrawl-pakker. Dette understøttes af de udførte unit test, som bygger på 100% dækning af de forskellige ViewModel klasser der indeholder selve handlingen i CC-App. Derudover viser den opstillede systemtest at en bruger kan se sine tilmeldte



pubcrawl-pakker inde på siden MinProfil, som er valgt efter en specifik dato.

#### 10.4 CCHttpClient

CCHttpClient er en klasse i CC-App som anvender REST til at kommunikere med Web-API controlleren i CC-Web. Web-API controlleren beskrives nærmere i afsnit 11.

Metoderne i CCHttpClient klassen anvendes for, at der kan sendes og hentes data mellem CC-App og CC-Database, via Web-API controlleren. I det nedestående vises metoderne i CCHttpClient.

- HttpClient GetUserFromServer
- HttpClient CreateUser
- HttpClient AddPubCrawls

Handlingsforløbet i metoden *HttpClientCreateUser* går ud på at brugerens indtastede oplysninger fra siden *OpretBruger* sendes til CC-Database. Dette sker via et POST REST kald til metoden *Registrer* i Web-API controlleren, som kommunikere direkte med CC-Database. Efterfølgende kan brugerens oplysninger hentes ud fra CC-Database ved et kald af metoden *HttpClient GetUserFromServer* som anvender et GET REST kald til metoden *Bruger* i Web-API controlleren.



## 11 Web-API

Design af Web-API I dette afsnit vil designet og de tilhørende overvejelser for CityCrawls Web-API præsenteres. Der ønskes at have en Web-API controller for at CC-App kan kommunikere med CC-Database via CC-Web og for at CC-Web kan tilgå brugerens bookede pubcrawls igennem CC-Database.

Web-API'et skal designes således at der kan gemmes bruger oplysninger i CC-Database når en bruger register sig på CC-App. API'et skal også kunne verificere at en bruger er eksisterende i CC-Database når brugeren forsøger at logge ind, samt kunne verificere at der anvendes de korrekte loginoplysninger når en eksisterende bruger forsøger at logge ind.

Det er også vigtigt at der er en form for sikkerhed for at beskytte brugerens oplysninger. Derfor ønskes API'et designet således at brugerens password bliver hashet. Når et password hashes vil det stadigvæk være muligt at kunne validere hvorvidt det korrekte password anvendes.

Tilsidst skal API'et bruges til at virksomhederne på CC-Web kan se når en bruger har bookeet en pubcrawl hos dem.

### 11.1 Implementering af Web-API

Web-API er blevet implementeret som et REST API, samt som en API controller with actions, using Entity Framework Core. Det betyder at der er blevet implementeret et API som kan kommunikere med CC-Database som er implementeret i Entity Framework Core.

I API'et er der implementeret en række HTTP-request som netop opfylder de ønskede designkrav. Der blandt andet har et HTTPGet-request, kaldet BrugerLogin, hvis opgave er at verificere at der anvendes de korrekte brugeroplysninger og et HTTPPost-request, kaldet Registrer, som gemmer en nyoprettet bruger i CC-Database.

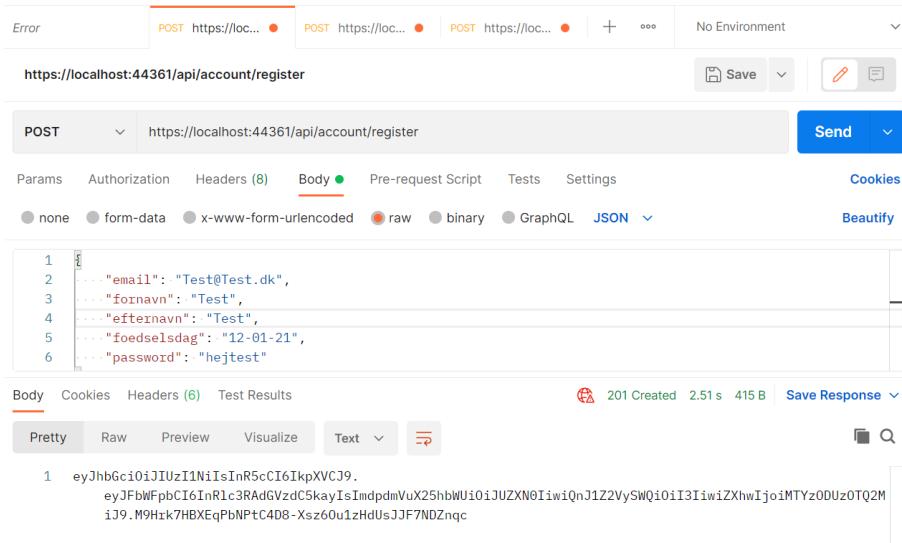
For at sikre brugerens password er bcrypt og JSON Web Token blevet anvendt. Bcrypt er en krypterings algoritme og JSON Web Token er en standard som anvendes til dele digitalt signerede information mellem en klient og en server.

### 11.2 Test af Web-API

Afslutningsvis er Web-API'et blevet testet for at undersøge om det rent faktisk er muligt at registrere en bruger og logge ind via Web-API'et, samt om brugerens oplysningerne bliver omdannes til JSON Web Tokens. Disse test er udarbejdet ved brug at Postman, som er en applikation som specifik anvendes til test af API.

Ved at anvende programmet Postman er det muligt at teste de implementeret HTTP-requests som netop anvendes til at kommunikere mellem CC-App og CC-Database.

I figur 31 illustreres det at det er muligt at oprette en bruger via API'et og at brugerens loginoplysningerne bruges til at danne en JSON Web Token.



```

1
2   ...
3   ...
4   ...
5   ...
6   ...

```

```

1 ... "email": "Test@Test.dk",
2 ... "fornavn": "Test",
3 ... "efternavn": "Test",
4 ... "fødselsdag": "12-01-21",
5 ... "password": "hejtest"
6

```

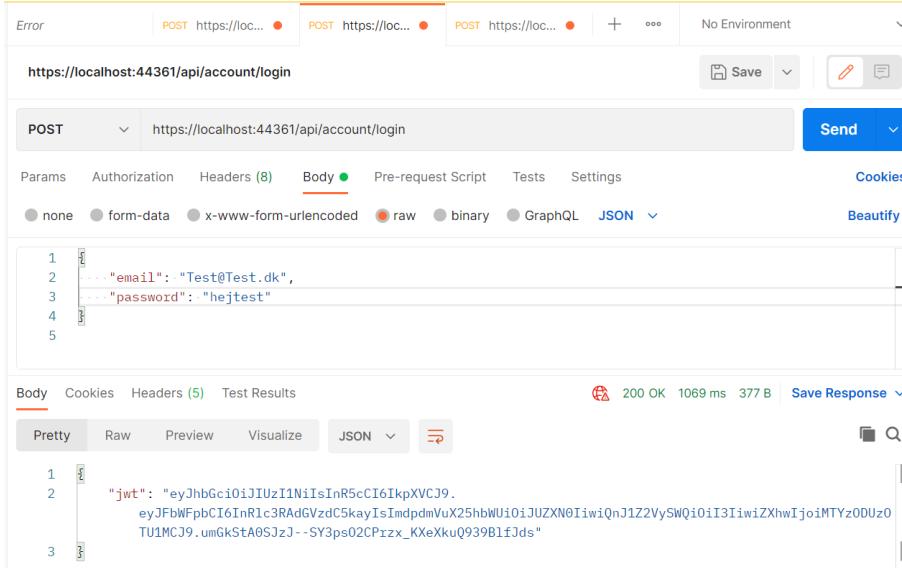
Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize Text

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  
eyJFbWFpbCI6InRlc3RAdGVzdC5kayIsImdpdmVuX25hbWUiOjJUZXN0IiwiQnJ1Z2VySWQiOjI3IiwiZXhwIjoiMTYzODUzOTQ2M  
i9. M9Hrk7HDXEqPbNptC4D8-Xsz60u1zHdUsJJF7NDZnqc

Figur 31: Test af Web-API hvor en bruger registrerer sig. Testen viser at loginoplysningerne omdannes til et sikkert JSON Web Token. Dette er implementeret fordi der logges automatisk ind når en bruger opretter en profil og derfor ønskes det at omdanne loginoplysningerne til et JSON Web Token.

Figur 32 viser at det er muligt at logge ind med den oprettet profil fra figur 31 og at loginoplysningerne bliver omdannet til et JSON Web Token.



```

1
2
3
4
5

```

```

1 ... "email": "Test@Test.dk",
2 ... "password": "hejtest"
3
4
5

```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

"jwt": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  
eyJFbWFpbCI6InRlc3RAdGVzdC5kayIsImdpdmVuX25hbWUiOjJUZXN0IiwiQnJ1Z2VySWQiOjI3IiwiZXhwIjoiMTYzODUzOTQ2M  
i9. M9Hrk7HDXEqPbNptC4D8-Xsz60u1zHdUsJJF7NDZnqc"

Figur 32: Test af Web-API hvor en bruger logges ind. Testen viser at brugeroplysningerne omdannes til et sikkert JSON Web Token.

Det kan heraf konkluderes at det er muligt at register en ny bruger via Web-API'et, samt logge ind med en eksisterende bruger. Det kan også konkluderes at det er muligt at omdanne brugerens loginoplysninger til et sikkert JSON Web Token.

Web-API'et bliver også testet når CC-Database testes i afsnit 13, idet API'et anvendes når CC-App læser og skriver data til og fra CC-Database. Som det konkluderes i afsnittet for test af CC-Database er det muligt at data kommunikeres korrekt mellem CC-App og CC-Database, og derfor kan det også konkluderes at Web-API'et også i praktisk fungerer efter hensigten.



## 12 CC-Web

### 12.1 Design af CC-Web

I dette afsnit vil designet for CityCrawls web-applikation, kaldet CC-Web, blive præsenteret. Designet for CC-Web er blevet udarbejdet i samhørighed med designet for CC-App for at have et gennemført og ensartet design igennem hele CityCrawl platformen.

Der er blevet udarbejdet en række designs som viser hvordan de forskellige sider på CC-Web skal se ud. Disse er blevet udarbejdet for at give udviklerne af web-applikationen et design at arbejde ud fra, samt at skabe en visuel enhed omkring web-applikationens udseende.

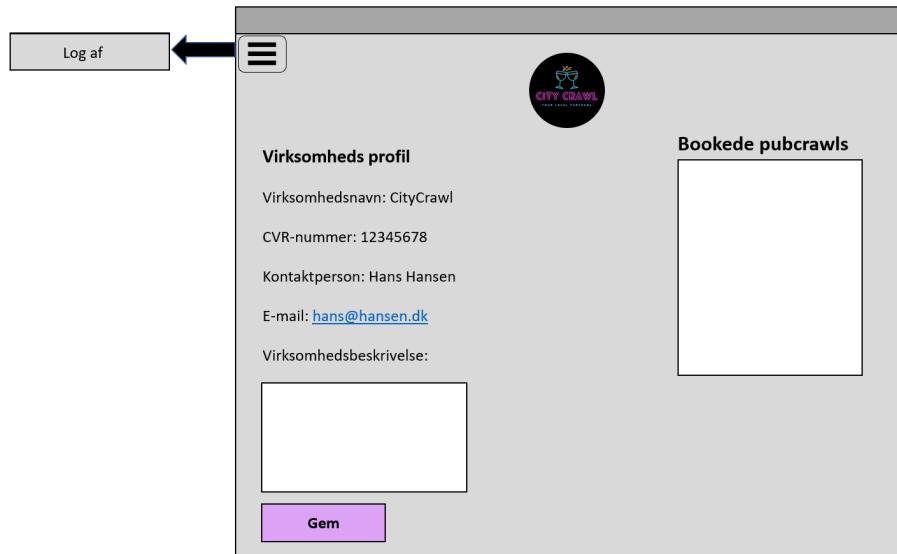
CC-Web skal give virksomheder mulighed for både at logge ind på en eksisterende bruger, samt oprette en ny bruger hvis de ønsker at være synlige på CityCrawl platformen. Det er med andre ord en applikation som diverse virksomheder tilgår via en internetbrowser. Det er altså ikke et program som virksomhederne skal have downloadet på deres egne computer, men blot en hjemmeside de har fri adgang til fra hvilket som helst enhed.

I figur 33 kan forsiden for CC-Web ses. Det er denne side som virksomhederne først bliver præsenteret for når de tilgår CC-Web. Her gives de to muligheder, nemlig enten at logge ind eller oprette en ny bruger.



Figur 33: CC-Web forside

Det sidste vindue, som er illustreret i figur 34, viser virksomhedens profil på CC-Web. Her bliver virksomhedens oplysninger vist og her har virksomheden mulighed for at lave en beskrivelse af deres virksomhed. Siden viser også de pubcrawls som er blevet booket og som skal forbi den pågældende virksomhed.



Figur 34: Virksomhedsprofil på CC-Web

De resterede designbilleder for CC-Web kan findes i bilagsrapporten under afsnittet for CC-Web.

## 12.2 Implementering af CC-Web

I dette afsnit det blive beskrevet hvordan CC-Web er blevet implementeret og hvilke overvejelser der været i den forbindelse derved.

CC-Web er blevet implementeret som et ASP.NET Core Web App projekt med ud fra Model-View-Controller principippet, fordi dette design giver mulighed for både at bygge en webapplikation, snakke sammen med en database og anvende et Web-API.

CC-Web projektet består af fem HTML-filer, også kaldet Views, som hver repræsentere de ønskede sider i applikationen, tilhørende controllers som back-end funktionalitet, samt en CSS-fil, som bruges til at style siden. Til implementeringen af CC-Web er der også blevet brugt Individual accounts som er en del af Identity.

Ved brug af HTML-filer, er selve elementerne blevet opsat for hjemmesiden. Her gøres brug af standard HTML tags. Der gøres også brug af en pre-defineret fil \_Layout, som bestemmer nogle generelle features på siden, for eksempel menu og footer. Siderne er blevet lavet, således at der kan gøres brug af og sendes data imellem de forskellige sider. Få steder er der blevet gjort brug af style-tagget, som gør det muligt at lave lokal styling af specifikke elementer.

Da der er elementer, så som knapper, der går igen mellem siderne, er der blevet brugt en CSS fil, som laver et generelt design, som kan bruge på alle HTML-siderne i projektet. Dette er for eksempel illustreret ved at de lyserrøde og blå knapper går igen med et ens design, der er altså lavet et generelt design for disse knapper.

Da HTML- og CSS-elementerne ikke er i stand til at udføre handlinger, så som at snakke med en given database, er der brug for noget back-end. Dette gøres i forbindelse med Controllers, der programmers via C#. Disse Controllers udfører operationer, såsom at tilføje og læse elementer til og fra en database. I dette tilfælde, kræves data overført til og fra HTML-siderne. Ved brug af form-tagget i HTML kan der sendes data mellem View og Controller. Denne data kan da processeres af Controlleren hvor den efterfølgende vil returnere. Et given View kan gøre brug af Models, som sendes fra Controlleren til View. Model gør det muligt at tilgå elementer, såsom en database, der da kan tilgå den data der findes deri.

I systemet gøres der også brug af Models, som i sin simpelhed, blot er måder at oprette forbindelse med elementer til en SQL relationel database. Det er da disse Models, som gør at databasen ved hvad den skal indeholde. Disse data er da det der kan bruges af Controllers og Views. For at gøre brug af Models



som database entiteter, bruges Entity Framework .NET Core. Dette kan for eksempel bruges også til at checke om data i CC-Database findes, som for eksempel Profil data for virksomheder i Virksomheds entiteten.

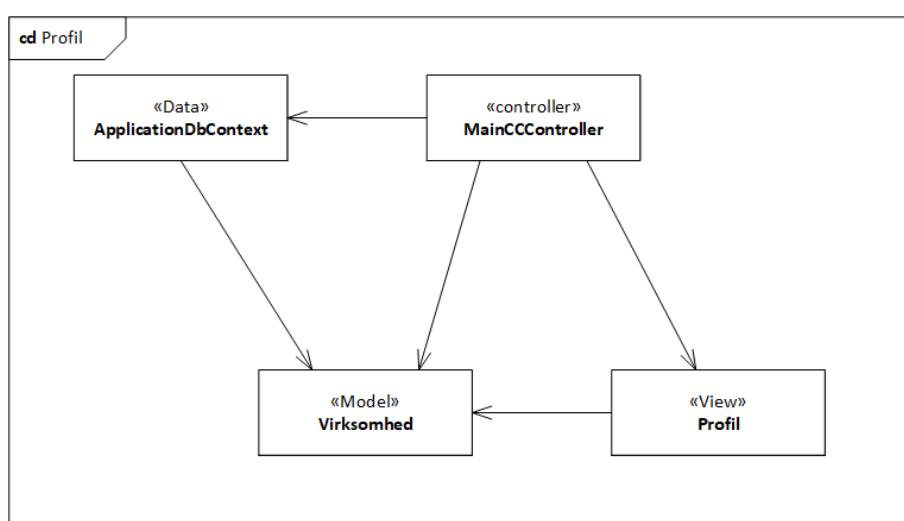
Da der er brug for at blive vist som information på CC-Web profil siden, er det nødvendigt at gemme data, som kan tilgås på alle givende sider. Dette gøres ved at bruge Identity, som gemmer profildata via cookies. Det vil sige, at når en bruger logger ind, vil data for det login blive gemt, indtil der logges ud. Herved kan en bruger identificeres. Identity giver også muligheden for at oprette en bruger, som bliver gemt i databasen. Det er denne data Identity gør brug af, til at bestemme hvilken bruger som er aktiv.

Sammenlignes implementeringen af forsiden for CC-Web i figur 35 og designet for forsiden i figur 33 er det illustreret at det ønsket design faktisk er blevet implementeret efter hensigten. Det illustreres også at der er lavet nogle meget få modifikationer af designet medført enten af nytænkning af designet og grundet anvendelsen af ASP.NET Core Web applikation projekt.



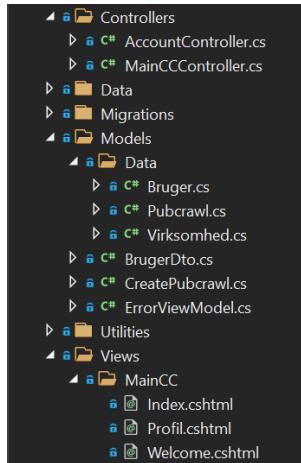
Figur 35: Den implementeret forside for CC-Web

I figur 36 illustrere blandt andet MVC strukturen for Profil Viewet. Ydermere viser diagrammet forbindelsen mellem systemets elementer. MainCCController etterspørger profildata via ApplicationDbContext, som så finder data fra Virksomheds modellen. MainCCController henter efterfølgende data var Virksomheds modellen og denne data vises på Profil Viewet.



Figur 36: Klassediagram for Profil Viewet

I figur 37 vises mapperne som opfylder MVC strukturen. Det ses ved mapperne Controller, Model og View.



Figur 37: Mappestruktur der viser MVC strukturen

### 12.3 Test af CC-Web

Som det sidste led i implementeringen af CC-Web der der dele af systemet blevet testet ved brug af unit testning frameworkt NUnit. Mængdeden af test for CC-Web er blevet nedprioriteret grundet tidsmangel, dette bliver yderligere uddybet i afgrænsningen i afsnit 6.

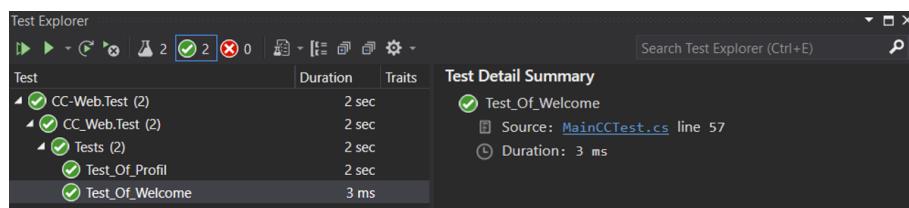
For at teste CC-Web er der blevet opstillet to NUnit test, for at se om den kommunikation der er mellem View og controller fungerer som efter hensigten. Herved har valget af test landet på Welcome og Profil metoderne i MainCCController. Welcome er den side virksomheden bliver præsenteret for når der er blevet logget ind på CC-Web og Profil er den side som viser virksomhedens oplysninger.

Welcome testes for at se om der omdiriges til den rigtige side. Testen kalder da Welcome, og tester herved om den givende side der returneres, er den som ønsket.

Profil siden testes for om der kan identificeres den rigtige virksomhed. Ved at sende et id med til Profil, testes der, om den virksomhed der passer til id'et er korrekt.

Ved brug af Profil siden, er der påkrævet adgang til CC-Database. Derfor opsættes adgang til CC-data ved start af testen. Denne forbindelse tester indirekte om databasen fungerer, fordi testen udføres ved at læse data fra CC-Database.

Figur 38 viser Test Explore resultaterne for MainCCController.



Figur 38: Test Explorer resultater af MainCCController. Udkippet viser at det to udarbejdet NUnit test for henholdsvis Welcome og Profil er vellykket.

Ved at lave test af systemet, kan det konkluderes at systemet fungerer korrekt. Da der er mulighed for at omdirigere mellem Views, samt identificere og læse data, ved hjælp af EF Core. Derfor opfylder systemet de krav der er sat, hvilket vil sige at backend controlleren fungere korrekt. Yderligere kan det ved visuel inspektion konkluderes at systemets front-end opfylder det ønskede design via korrekte opsætning af



HTML og CSS. Det kan også konkluderes at Identity kan bruges til at registrere nye virksomheder og logge ind og ud af web-applikation, samt at forbindelsen mellem controller, Views og database fungere korrekt og dermed er alle krav for CC-Web opfyldt.

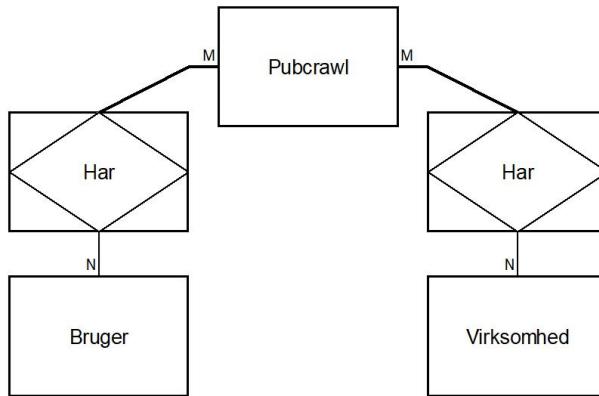
## 13 CC-Database

### 13.1 Design af CC-Database

I afsnittet vil designet og de forudgående overvejelser for CC-Database blive beskrevet. Det vil gøres ved brug af ERD- og DSD-diagrammer, idet disse diagrammer giver et detaljeret indblik i databasens forhold, samt hvordan databasen ønskes implementeret.

Det er blevet besluttet i projektgruppen at implementere en relationel database, hvor det skal være muligt både at kunne skrive til og læse fra databasen. Databasen skal indeholde følgende tre entiteter: Pubcrawl, Bruger og Virksomhed.

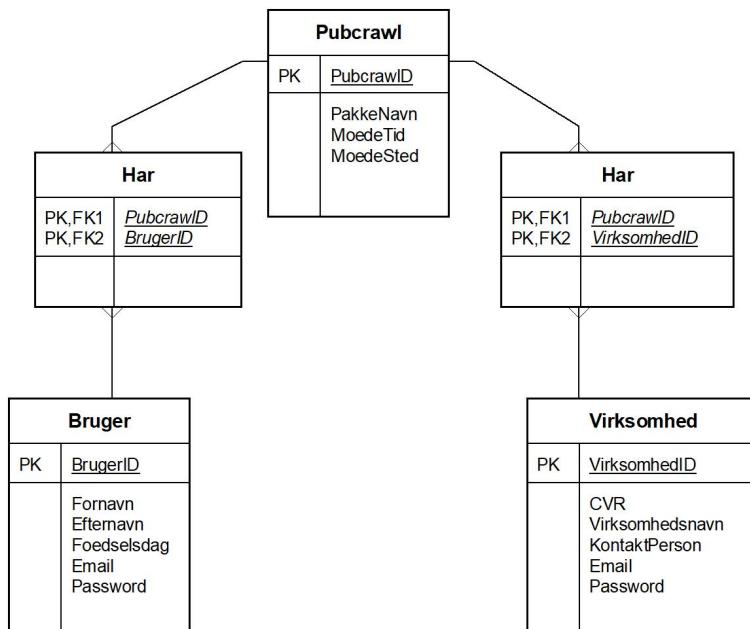
Pubcrawl entiteten har et mange til mange forhold med både bruger og virksomhed. Det betyder f.eks. at en bruger kan booke mange pubcrawls og at en pubcrawl kan have mange brugere. Ligeledes gælder det at forholdet mellem en virksomhed og et pubcrawl er mange til mange. Disse forhold tydeliggøres i ERD-diagrammet som illustreres i figur 39.



Figur 39: ERD diagram for CC-Database. Diagrammet viser forholdet imellem databasens forskellige entiteter

I overvejelserne for designet af både CC-Database, samt de to brugergrænseflader CC-Web og CC-App er der i projektgruppen blevet diskuteret, hvilke oplysninger som kan være relevante at have for både en bruger og en virksomhed. Dette er en vigtig beslutning, fordi der skal være samhørighed mellem den data som skal skrives til CC-Database igennem brugergrænsefladerne, og den data som læses fra CC-Database. I DSD-diagrammet i figur 40 vises de attributter som entiteterne skal have. Disse attributter repræsenterer også de relevante oplysninger, som en bruger og en virksomhed skal have.

DSD-diagrammet i figur 40 viser også hvilke attributter som er de primære nøgler (PK) for hver entitet og hvor i CC-Database der skal implementeres fremmed nøgler (FK). Disse oplysninger er vigtige, fordi de viser hvordan CC-Database skal implementere for at overholde de ønskede forhold.



Figur 40: DSD diagram for CC-Database. Diagrammet viser de attributter de forskellige entiteter indeholder

### 13.2 Implementering af CC-Database

I forbindelse med implementeringen af CC-Database har der været en række udfordringer i forhold til at kunne skrive og læse data til og fra CC-Web. Det har været udfordrende at finde den bedst mulige metode til at kunne læse data fra og skrive data til CC-Database igennem web-applikationen. Grunden til udfordringer med implementation af CC-Database har blandt andet været, at det relevante faglige indhold ift. integration af databasen er blevet undervist sent i semesteret.

Det er især kommunikationen mellem CC-Web og Virksomheds entiteten, som har krævet en ekstra indsats. Det har medført at den faktiske opbyggelse og anvendelse af databasen er blevet gentaenk et antal gange. Vi fandt dog ud af undervejs at det ville være smartere at anvende ASP.NET Core med Identity, da det allerede har et indbygget login system. Fordelen ved Identity er, at virksomhedens loginoplysninger gemmes ved hjælp af cookies. Dette har medført at projektgruppen skulle identificere hvordan loginoplysningerne gemmes i CC-Database.

Det har været et ønske at oprettelsen af en ny virksomhed gøres på en gang og det er her den store udfordring har været, fordi det ikke har været muligt at tilgå både en given Identity database og CC-Database på en gang.

Derfor er databasen blevet implementere således at email og password gemmes både i Identity's entitet og i Virksomheds entiteten, så det fra CC-Web's side ligner at oprettelsen kun sker en gang, selvom dataen bliver uddelegeret til to forskellige entiteter.

Det betyder at CC-Database bruges til at gemme oplysninger på alle virksomheder og alle brugere i deres respektive entiteter, samt at Identity indsættes i egen entitet, og bruges til at validere virksomhedernes loginoplysninger og adgange.

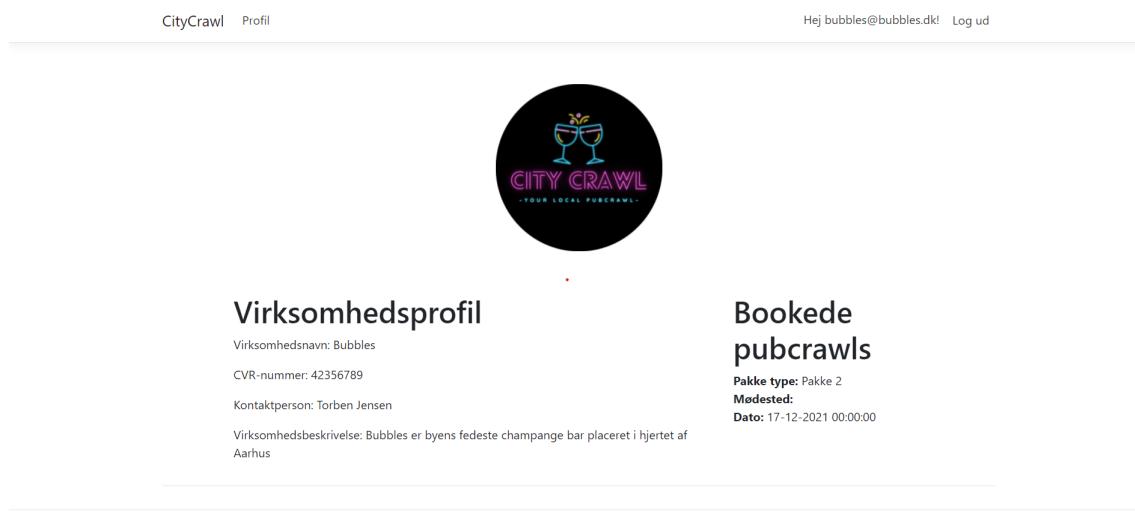
I slutningen af semesterprojektets forløb gik det op for projektgruppen at den oprindelige opbygningen af CC-Database skulle tilrettes, fordi det ikke var muligt at virksomheden kunne se de pubcrawls som en bruger havde bookede hos dem. Derfor blev designet af CC-Database lavet om, således at CC-Database også har en Pubcrawl entitet, som har et mange til mange forhold til både Bruger og Virksomhed entiteterne, som er illustreret i figur 39.

Udfordringerne har medført at der er sket en forsinkelse i forhold til den oprindelige tidsplan og at gruppen derfor har måtte nedprioritere andre opgaver i projektets forløb.

### 13.3 Test af CC-Database

Afslutningsvis udføres en systemtest som viser hvorvidt CC-App og CC-Web kan læse fra og skrive til CC-Database. For at gøre test af CC-Database mulig er der blevet implementeret de fire virksomheder, som indgår i de mulige pubcrawl-pakker i CC-App. Implementering af disse virksomheder bliver oprettet fra program start, således at der dannes en kobling af de bookede pubcrawl-pakker med de korrekte virksomheder der indgår i de valgte pubcrawl-pakker. Heraf er det muligt at få vist hvilke pubcrawl-pakker en virksomhed er en udbyder til på CC-Web.

Første test af CC-Database er blevet udført ved at køre både CC-Web og CC-App samtidigt, blandt andet for at CC-App har adgang til Web-API'et. Det testes hvorvidt data gemmes i CC-Database når en bruger booker en pubcrawl-pakke i CC-App, så samme data kan læses fra CC-Web og blive vist på den pågældende virksomheds profil side. Figur 41 viser resultatet for denne test, at det netop er muligt at læse den gemte pubcrawl data fra CC-App i CC-Web, via kommunikationen til CC-Database. Det betyder at er muligt at skrive til CC-Database fra CC-App og læse fra CC-Database til CC-Web.



The screenshot shows the CC-Web interface. At the top, there is a navigation bar with 'CityCrawl' and 'Profil' on the left, and 'Hej bubbles@bubbles.dk! Log ud' on the right. Below the navigation bar is a circular logo for 'CITY CRAWL - YOUR LOCAL PUBCRAWL' featuring two glasses. To the left of the logo is a section titled 'Virksomhedsprofil' containing the company name 'Bubbles', CVR number '42356789', contact person 'Torben Jensen', and a description: 'Bubbles er byens fedeste champange bar placeret i hjertet af Aarhus'. To the right of the logo is a section titled 'Bookede pubcrawls' showing a booking for 'Pakke type: Pakke 2', 'Mødested:', and 'Dato: 17-12-2021 00:00:00'.

Figur 41: Et billede af CC-Web hvor data fra Virksomheds entiteten og Bruger entiteten læse fra CC-Database og vises på CC-Web.

Figur 41 viser også at det er muligt at læse data fra Virksomheds entiteten, således at de korrekte virksomhedsoplysninger bliver vist for den pågældende virksomhed som er logget ind på CC-Web.

Afslutningsvis viser figur 41 at det er muligt at logge ind med en eksisterende virksomhedsprofil. Når der logges ind på CC-Web bliver det tjekket hvorvidt om den indtastede e-mail og det indtastede password passer med de registrerede oplysninger for en virksomhed som allerede eksisterer i CC-Database. Overensstemmelse af de indtastede loginoplysninger for en virksomhed vil gøre at virksomheden kan logge ind i CC-Web, samt at der kan gemmes loginoplysninger, der giver mulighed for at vise data for virksomheden der er logget ind. Disse data kan ses på profilsiden for virksomheden. Uoverensstemmelse af disse loginoplysninger gør at virksomheden bliver præsenteret for en fejlmeddeelse, som er illustreret i figur 42.

CityCrawl

Opret bruger Login



• The Password field is required.

bubblesbubbles.dk
<input type="password"/>

The Password field is required.

Remember me?

**Log in**

© 2021 - CityCrawl - projektgruppe 1 - 4. semester

Figur 42: Udklip fra CC-Web som viser at virksomheden bliver præsenteret for en fejlmeddelelse når forkerte loginoplysninger bliver indtastet.

Efter den første test er det blevet undersøgt hvorvidt data gemmes som forventet i CC-Database for at sikre at der kommunikeres korrekt gennem de forskellige aspekter i CityCrawl. I henholdsvis figur 43, figur 44 og figur 45 viser at der er blevet gemt data i CC-Database.

	BrugerID	Fornavn	Efternavn	Fødselsdag	Email	PwHash
▷	1	Jan	Jespersen	02/08/1987	jan@jep.dk	\$2a\$10\$kMUFa...
▷	2	Hans	Andersen	04/12/1996	hans@andersen...	\$2a\$10\$PdpvM...
∅	NULL	NULL	NULL	NULL	NULL	NULL

Figur 43: Udklip fra Bruger entiteten som viser at der er to bruger gemt i CC-Database, som er sendt fra CC-App.

	PubcrawlId	PakkeNavn	MoedeSted	MoedeTid
▷	1	Pakke 1		18-12-2021 00...
∅	NULL	NULL	NULL	NULL

Figur 44: Udklip fra Pubcrawl entiteten som viser at der et pubcrawl gemt i CC-Database, som er sendt ind fra CC-App.

	VirksomhedID	CVR	Virksomhedsna...	KontaktPerson	Email	Password	Beskrivelse
▷	1	42356789	Bubbles	Torben Jensen	bubbles@bubbl...	Bubbles1!	Bubbles er byen...
▷	2	34127865	Hildas Beer Bar	Jytte Sørensen	jytte@hildsbeer...	HildasBeer2!	Boobies and Be...
▷	3	56349865	Jazz it up	Simone Nielsen	info@jazzitup.c...	Jazzitup3!	Jazz op dit liv
▷	4	67895432	Wine & Dine	Lars Clausen	wine@dine.com	Wine&dine4	Drik dig fuld, så...
∅	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figur 45: Udklip fra Virksomhed entiteten som viser at fire virksomheder gemt i databasen. Disse virksomheder er oprettet ved at af program kørsel

Projektgruppen kan konkludere at der kan både skrive til og læse fra CC-Database, at data gemmes i de korrekte entiteter i CC-Database og at login verificeres korrekt ved hjælp af CC-Database.



## 14 Accepttest

I dette afsnit bliver accepttestene for både CC-App og CC-Web beskrevet. Disse accepttest er udarbejdet for at undersøge, hvorvidt de opstillede krav for CityCrawl opfyldes. Begge accepttest er udarbejdet som observationstests for systemet.

De udførte accepttest for henholdsvis CC-App og CC-Web kan findes i billagsrapporten i afsnittet Accepttest. Der er også udarbejdet en præsentationsvideo for CityCrawl hvor der vises programkørsel af CC-App og CC-Web. Denne video, kaldet præsentation\_af\_citycrawl, som kan findes i bilagsmappen.

### 14.1 Accepttest for CC-App

I accepttesten for CC-App testes det, hvorvidt det er muligt for en bruger at oprette sig via applikationen samt både kunne logge ind og ud af applikationen. Det testes også, hvorvidt brugeren bliver præsenteret for de korrekte fejlbeskeder, hvis forkerte oplysninger indtastes og til slut testes det, hvorvidt det er muligt for en bruger at tilmelde sig en pubcrawl for en given dato.

### 14.2 Accepttest for CC-Web

I accepttesten for CC-Web bliver det testet hvorvidt det er muligt for en virksomhed at registrere sig på web-applikationen, samt både at kunne logge ind og ud af web-applikationen. Det bliver også test om hvorvidt virksomheden bliver præsenteret for fejlbeskeder når forkerte oplysninger indtastes. Herudover testes det også hvorvidt virksomheden får vist sine virksomhedsoplysninger på virksomhedsprofil-siden.

### 14.3 Konklusion for accepttest

Det kan for accepttesten konkluderes, at den overordnet gik som forventet, og derfor tilregnes den for vellykket. Det var kun test nr. 5, i CC-App, for indtastning af forkerte loginoplysninger som fejlede. Her var det forventet, at brugeren skulle modtage en fejlbesked, men idet der kom en exception i Web-API'et som ikke blev fanget af CCHttpClient modtog brugeren ikke den pæne fejlbesked. Fejlen blev hurtigt identificeret i CCHttpClient, som blev udvidet med den manglende fejlhåndtering for når brugeren ikke findes i CC-Database. Efterfølgende har denne korrigering medført at den korrekte fejlmeddeelse præsenteres for brugeren.



## 15 Opnåede erfaringer

I dette afsnit vil der være en beskrivelse af projektgruppens opnåede erfaringer under udarbejdelsen af CityCrawl. CityCrawl er et 4. semesterprojekt som har medført øget kendskab til de forskellige fag, ved at der er blevet oprettet grafiske brugergrænseflader og anvendt en relational database sammen med Entity Framework Core. Derudover er der også foretaget forskellige designovervejelser, samt udarbejdelse af HTTP-kommunikation og diverse test. Dette har skabt en dybere forståelse af undervisningen, da det er nødvendigt at sammensætte de forskellige fag, for at få et ønskeligt produkt. Yderligere er der dannet erfaring med systemer som git og Jenkins.

Tidligere er der skrevet om udfordringen med løbende undervisning og projektarbejde. Dette har givet projektgruppen erfaring med selvstudie, både forud for undervisningen og uden for undervisningsområdet. Denne erfaring har skabt indsigt i muligheden for selv at finde svaret på de problemer, der opstår undervejs i projektet. Erfaring som selvstændighed skal vise sig nyttig og vigtig i forbindelse med, for eksempel, uddannelse, praktik og arbejdsmarkedet.

Der er også blevet gjort nye erfaringer med allerede kendte systemer og metoder, såsom UML-diagrammer, Scrum og samarbejde. Selvom systemerne og metoderne er de samme, er de dog blevet brugt på nye måder, hvilket har skabt større kendskab til brugen af netop disse.

## 16 Fremtidigt arbejde

I dette afsnit bliver det fremtidige arbejde for CityCrawl præsenteret. Det vil beskrives, hvordan projektngruppen tænker, at det næste skridt i videreudviklingen af platformen vil kunne se ud, samt hvordan det endelige produkt vil kunne se ud. Det fremtidige arbejde vil bære præg af de afgrænsninger, der er blevet foretaget i forbindelse med udviklingen af CityCrawl til dette projekt.

Første del af en videreudvikling af CityCrawl skal indeholde en forbedring af delsystemerne CC-App, CC-Web og CC-Database. Her skal der fokuseres på, at brugeren skal have mulighed for at se afstanden til det valgte pubcrawl sted, ud fra et digitalt kortsystem, som skal være tilgængelig på CC-App. Dertil skal udviklingen af sikkerhed på CC-App færdigimplementeres, således at når en bruger opretter en profil, vil relevante data blive krypteret. Yderligere ønskes det også at få en gennemført IT sikkerhed på CC-Web, hvor det implementerede JSON Web Token kan fungere i samspil med cookies, således at virksomheden forbliver loget ind i CC-Web, uden at man går på kompromis med sikkerheden.

Anden del af det fremtidige arbejde vil bestå af, at der tilføjes endnu mere sikkerhed til brugen af CC-App ved, at der indføres tjk af brugerens alder ift. den gældende aldersgrænser angående køb af pubcrawl-pakker. Det ønskes også, at brugeren har mulighed for at oprette og slette et privat pubcrawl-event, som så kan være tilgængelig for andre brugere. Yderligere skal det være muligt for en virksomhed, at kunne tilbyde andre ydelser end kun at kunne tilbyde pubcrawls. Virksomheden skal også have mulighed for at slette et bestemt pubcrawl-event.

Afslutningsvis skal en endelige udviklingsfase af CityCrawl indeholde en udbygning af CC-App som består af, at CC-App bliver omstruktureret til en mobil applikation. Det er også ønskeligt, at CC-Database og CC-Web bliver sat op til at køre på en dedikeret server, som betyder at CC-Web er tilgængelig online, og ikke kun lokalt.



## 17 Konklusion

CityCrawl er en pubcrawl platform udviklet med en vision om at være et samlingspunkt for både nye og eksisterende bekendskaber. En platform hvor en bruger kan tilmelde sig pubcrawls i sit lokalområde, og en platform som hjælper lokale virksomheder med eksponering.

Det kan for projektet konkluderes, at det er lykkes at udvikle en platform, som opfylder det basale for CityCrawls opstillede krav, samt de formelle krav stillet af Aarhus Universitet. Det er med andre ord lykkes at udvikle en platform, som overordnet består af tre delsystemer: En WPF applikation kaldet CC-App, en web-applikation kaldet CC-Web og en relationel database kaldet CC-Database.

CC-App er brugernes indgangsvinkel til CityCrawl. Her er det muligt at oprette en bruger, logge ind med validering af brugeroplysninger, samt booke en pubcrawl til en ønsket dato.

CC-Web er virksomhedernes indgangsvinkel til CityCrawl. Her er det også muligt for en virksomhed at oprette sig, logge ind med validering af virksomhedsoplysninger, samt se hvilke pubcrawl der er booket hos den givne virksomhed og virksomhedens profiloplysninger.

CC-Database indeholder alle brugeroplysninger, virksomhedsoplysninger og de bookede pubcrawls. Det er muligt for både CC-App og CC-Web at skrive og læse fra CC-Database. CC-Web kommunikerer direkte med databasen, hvor CC-App kommunikerer med databasen igennem et Web-API.

Alle delsystemer er også blevet testet tilstrækkeligt ved både brug af unit test, systemtest samt via visuel inspektion af design.

Det kan derfor konkluderes, at projektgruppen har formået at designe, implementere og teste et velfungerende system, hvor brugeroplevelse og funktionalitet har været hovedfokus.



## 18 Referencer

- Model-view-controller, <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>, d. 09.12.2021, kl. 15:14.
- Scrum billede, figur 3, hentet fra [https://en.wikipedia.org/wiki/Scrum\\_\(software\\_development\)](https://en.wikipedia.org/wiki/Scrum_(software_development)), d. 11.12.2021, kl. 10:30.
- 4+1 view model billede 11, hentet fra 4+1view-architecture\_UML2.pdf fra lektion 5 i Software Design faget.
- MVVM billede 17, hentet fra fra GUI MVVM.pdf, lektion 07 MVVM i GUI-programmering faget.
- Til implementering af JSON Web Tokens der hentet inspiration udvalgt materiale undervisningslektion 12 om Security fra faget Netværksprogrammering og grundlæggende kommunikation. Nedenstående filer kan findes i mappen Reference-bilag i det afleverede materiale.
  - [1] 12b JSON Web Token.pdf
  - [2] Hvordan du tilføjer bruger login til et web API.pdf
- SOLID <https://en.wikipedia.org/wiki/SOLID>  
d. 12.12.2021 kl. 12:37.
- N-Tier <https://www.ibm.com/topics/three-tier-architecture>  
d. 09.12.2021 kl. 10:35.