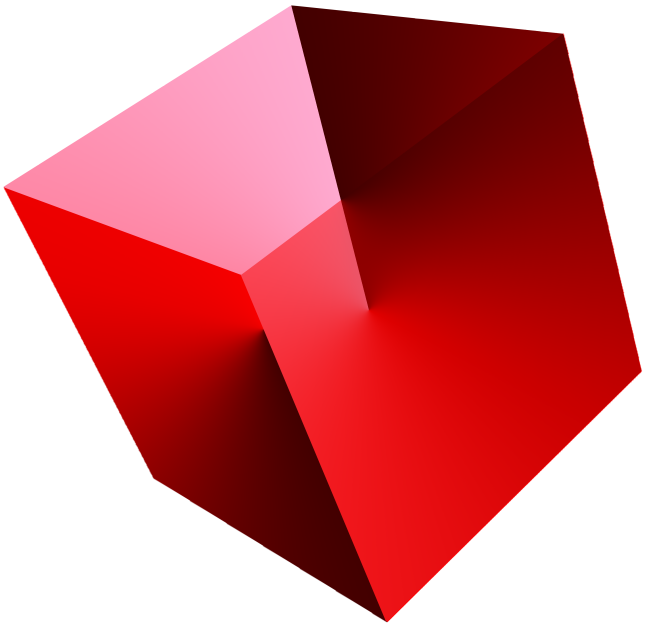
NTNU

Kunnskap for en bedre verden

Joint Company IT Workspace

IE303612 Bacheloroppgave



Bacheloroppgave 2020

Jonathan Øie (10001)
Liv Anne Nyland (10002)

Totalt antall sider inkludert forsiden: 143
Ålesund, 20/12/2020

Tittel:

Joint Company IT Workspace

Kandidatnummer (navn):

Jonathan Øie (10001)

Liv Anne Nyland (10002)

Dato:	Emnekode:	Emne:	Dokument tilgang:
20/12/2020	IE303612	Bacheloroppgave	

Studium:	Ant. sider/vedlegg:	Bibl. nr:
Dataingeniør	143 / 10	

Veileder:

Arne Styve

Sammendrag:

Gruppen har laget en egendefinert oppgave. Hensikten bak oppgaven er å gjøre utvikling og distribuering av multiplatformapplikasjoner enklere.

Oppgaven sitt omfang inkluderer et plug-in system med utviklingsrammeverk og med en server-klient arkitektur. For å løse oppgaven bruker gruppen Visual Studio og Xamarin med ekstra biblioteker for networking og grafikkhåndtering.

Resultatet er en løsning med en JCIW Android og Desktop klient, en JCIW Admin klient, en server og et utviklingsrammeverk med støttemaler til Visual Studio. Konklusjonen av oppgaven er at både planlegging og fremgangsmåten har fungert bra.

Innholdsfortegnelse

SAMMENDRAG

Gruppen har laget en egendefinert oppgave. Hensikten bak oppgaven er å gjøre utvikling og distribuering av multiplatformapplikasjoner enklere.

Oppgaven sitt omfang inkluderer et plug-in system med utviklingsrammeverk og med en server-klient arkitektur. For å løse oppgaven bruker gruppen Visual Studio og Xamarin med ekstra biblioteker for networking og grafikkhåndtering.

Resultatet er en løsning med en JCIW Android og Desktop klient, en JCIW Admin klient, en server og et utviklingsrammeverk med støttemaler til Visual Studio.
Konklusjonen av oppgaven er at både planlegging og fremgangsmåten har fungert bra.

Figurliste

TERMINOLOGI

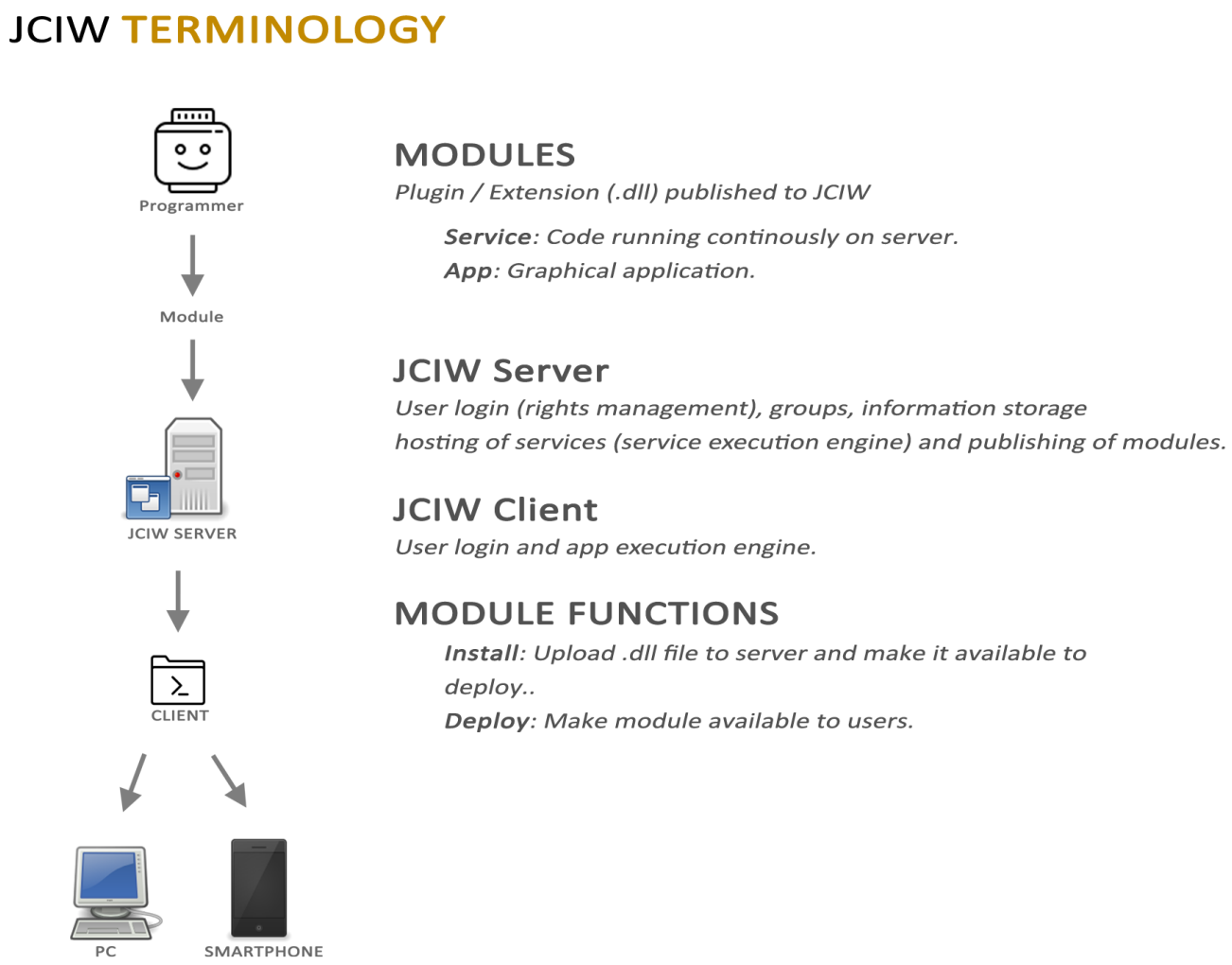


Figure 1: JCIW Terminologi

Begreper

Notasjoner

Symboler

Forkortelser

- **JCIW** = Joint Company IT Workspace

1 INNLEDNING

1.1 Bakgrunn

Joint Company IT Workspace (JCIW) er en selvdefinert oppgave basert på egenerfarte utfordringer som gruppen har hatt med programvareutvikling.

1.2 Problemstilling

Hva gruppen skal lage: En cross platform løsning for utvikling, distribuering og kjøring av plug-ins med innebygd grafikkmotor, nettverksfunksjonalitet, databasefunksjonalitet, platformfunksjoner og bruker / gruppe-funksjonalitet.

Hvorfor: Problemstillingen har oppstått av mangel på eksisterende løsninger og fra gruppen sine egne erfaringer med utvikling og distribuering av applikasjoner.

Gruppen har laget ulike nettverksapplikasjoner tidligere. Et problem som alltid har oppstått er hvordan håndtere administreringen i ettertid når serveren er oppe og brukerne har begynt å bruke systemet.

Det har da stått mellom det å lage nye funksjoner i selve applikasjonen kun for administrative oppgaver, eller å legge inn kommandolinjer i konsollet på applikasjonen for kjøring lokalt på serveren.

Det første alternativet krever mye tidskrevende arbeid med grafiske elementer som må lages og nettverkskode som må implementeres. I gruppen sin erfaring er dette noe som ofte blir utelatt særlig i hobbyprosjekter siden det kan skyve utgivelsen av applikasjonen tilbake flere uker. I noen tilfeller er dette arbeidet like mye som det å lage selve applikasjonen.

Det andre alternativet er mye enklere å implementere, men som raskt blir tungvindt å bruke. Spesielt om mann har flere nettverksapplikasjoner kjørende på samme server. Det innebærer å logge inn på serveren, identifisere den kjørende tjenesten mann ønsker å sende kommandoer til og åpne. Om man bruker programvare som Screen eller lignende software involverer det kommandoer og hurtigtaster for å navigere. En må i tillegg lese loggfiler fra filsystemet for å identifisere uforutsette problemer med applikasjonen. Dette er noe som krever gode ferdigheter for å kunne gjøre raskt (spessielt på Linux som er hva gruppen bruker), men som er tidskrevende for den vanlige brukeren.

Et annet problem som gruppen kom over er distribueringen og oppdateringen av applikasjonene. Dersom man skal oppdatere en Android app må man kompilere med en ny versjonskode, laste opp og ofte vente 4-7 dager før oppdateringen dukker opp på

Play Store. Ellers så kan mann sende installasjonsfilen direkte til brukeren via mail eller lignende sånn han raskt kan installere, men dette er ikke et bra alternativ for massedistribuering og det gir heller ingen god varsel til brukeren om at der er en ny versjon tilgjengelig sånn som Play Store gjør. Brukerne er heller ikke vant til det.

Utvikler laster opp ny versjon, venter 7 dager på at oppdateringen skal bli tilgjenglig på Play Store, bruker rapporterer en ny bug, utvikler fikser buggen, laster opp ny versjon... 7 dager senere blir fiksen tilgjengelig. Det er ikke en effektiv måte å utvikle på.

På desktop applikasjoner er det vanlig å legge inn en versjonsjekk også vise en dialogboks til brukeren dersom der er en ny versjon tilgjengelig. Det eller å automatisk oppdatere programvaren med egen kode eller Microsoft Store, Software Center eller lignende. Det kommer an på applikasjonen, men i mange tilfeller så vil denne funksjonaliteten være mye arbeid, og igjen, ikke noe som taes tid til i for eksempel hobbyprosjekter.

Noen tillegg til problemstillingen:

- Om en ikke lager webapplikasjoner har en problemer med at brukergrensesnitt ikke fungerer cross platform.
- Innloggingssystem og brukerhåndtering tar tid. Det ville spart mye tid å ha dette innebygd i løsningen.
- Sette opp og konfiguere database. Tar igjen litt tid og burde være innebygd.
- Skrive nettverkskode for sending og mottakelser av pakker. Utvikler vil bare definere pakker og sende dem.

1.3 Hensikt og målsetting

Målet med oppgaven er og lage et rammeverk med alle de nødvendige systemene rundt for å kunne utvikle og distribuere applikasjoner cross platform med støtte for TCP og en robust løsning for brukergrensesnitt. De nøyaktig samme applikasjonene skal kunne kjøre på Windows, Linux, Mac og Android.

Følgende er noen illustrasjoner av planlagt klient og admin workspace.

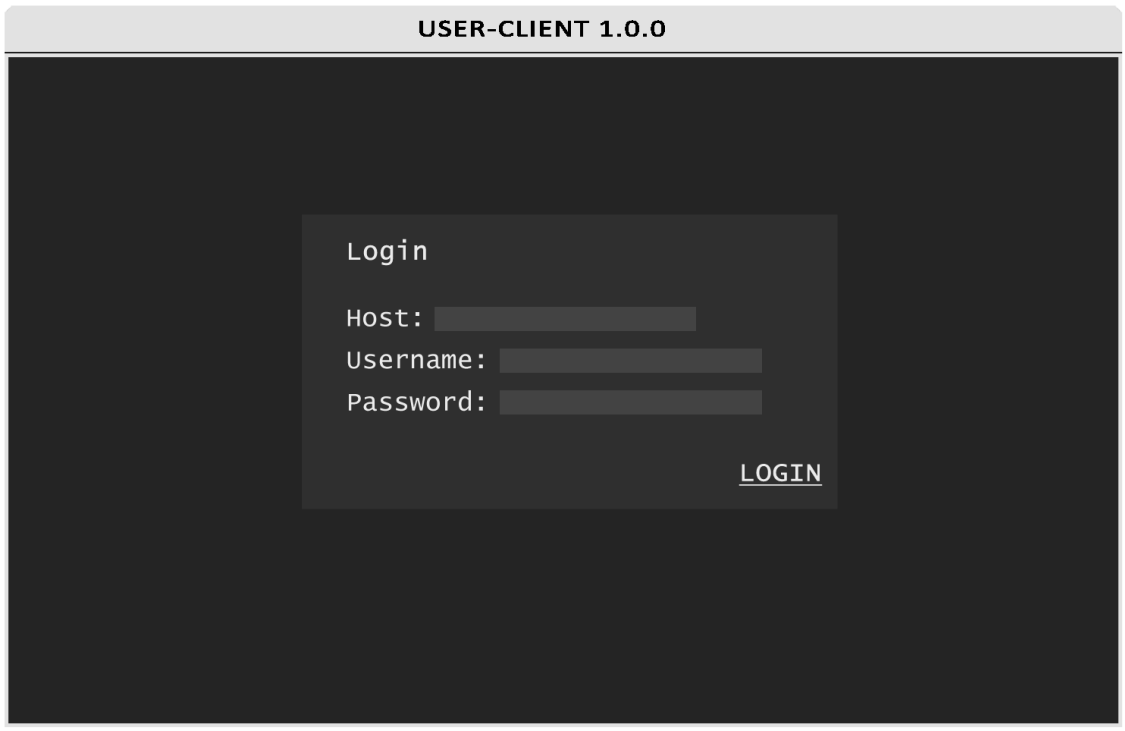


Figure 2: Illutrasjon av klient login

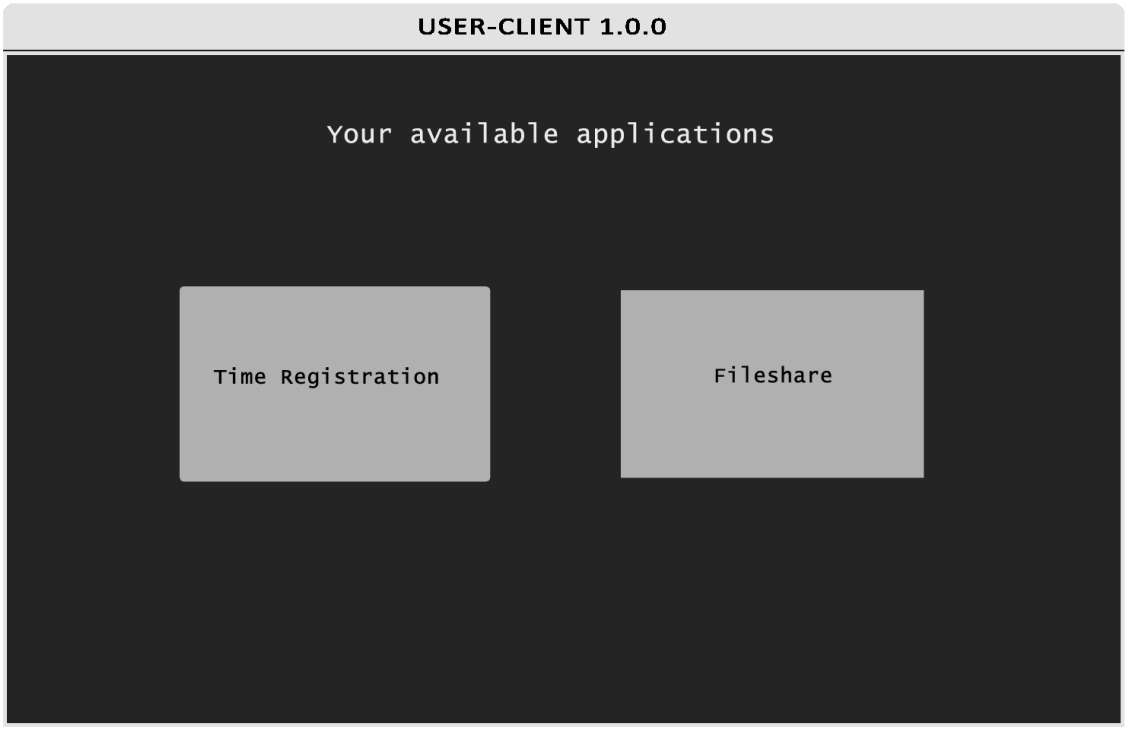


Figure 3: Illutrasjon av app liste

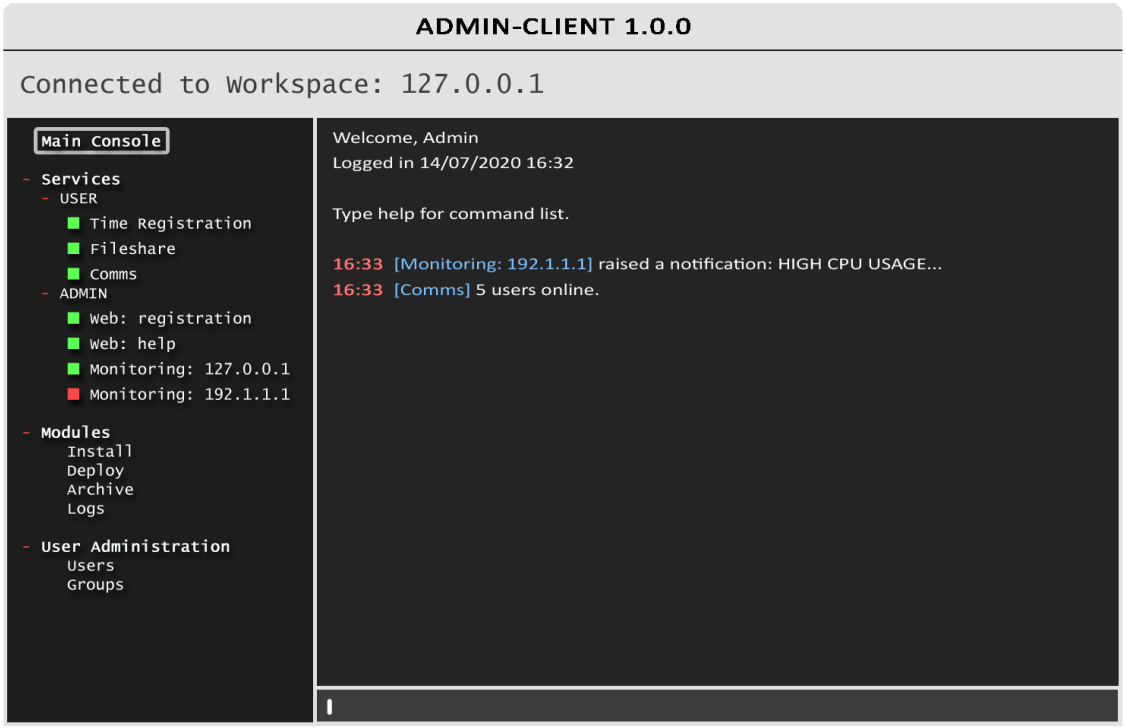


Figure 4: Illutrasjon av Adminkonsoll 1

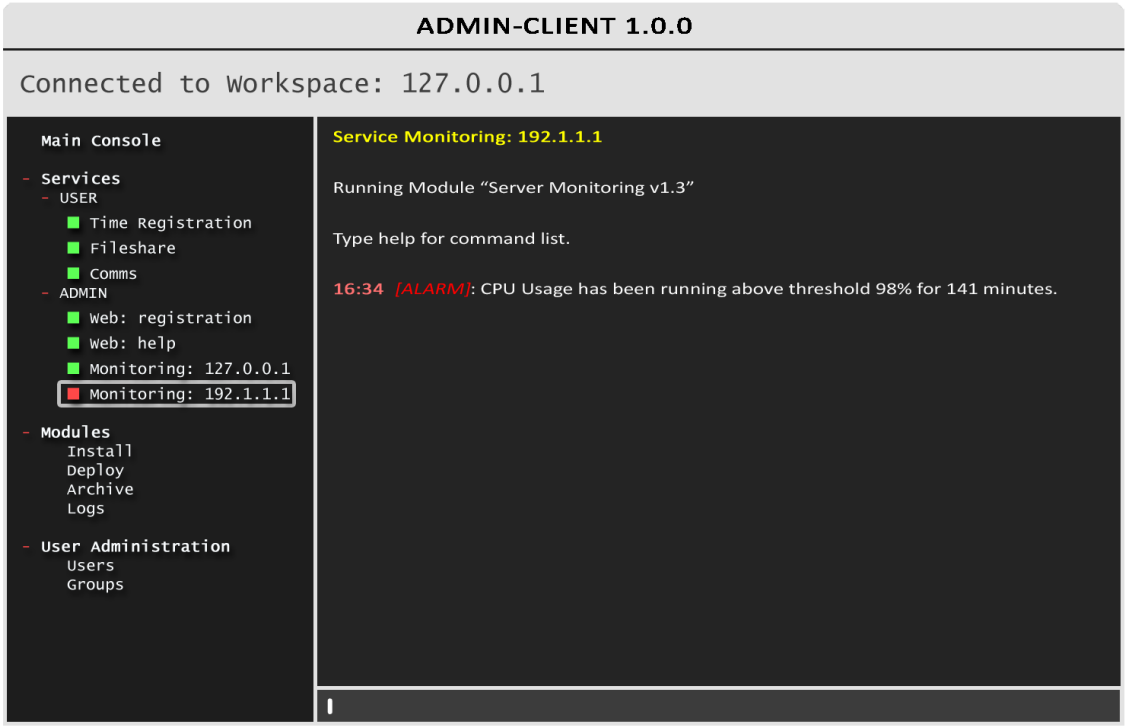


Figure 5: Illutrasjon av Adminkonsoll 2

Følgende er illustrasjoner av en demo applikasjon for å demonstrerer JCIW sine funksjoner.

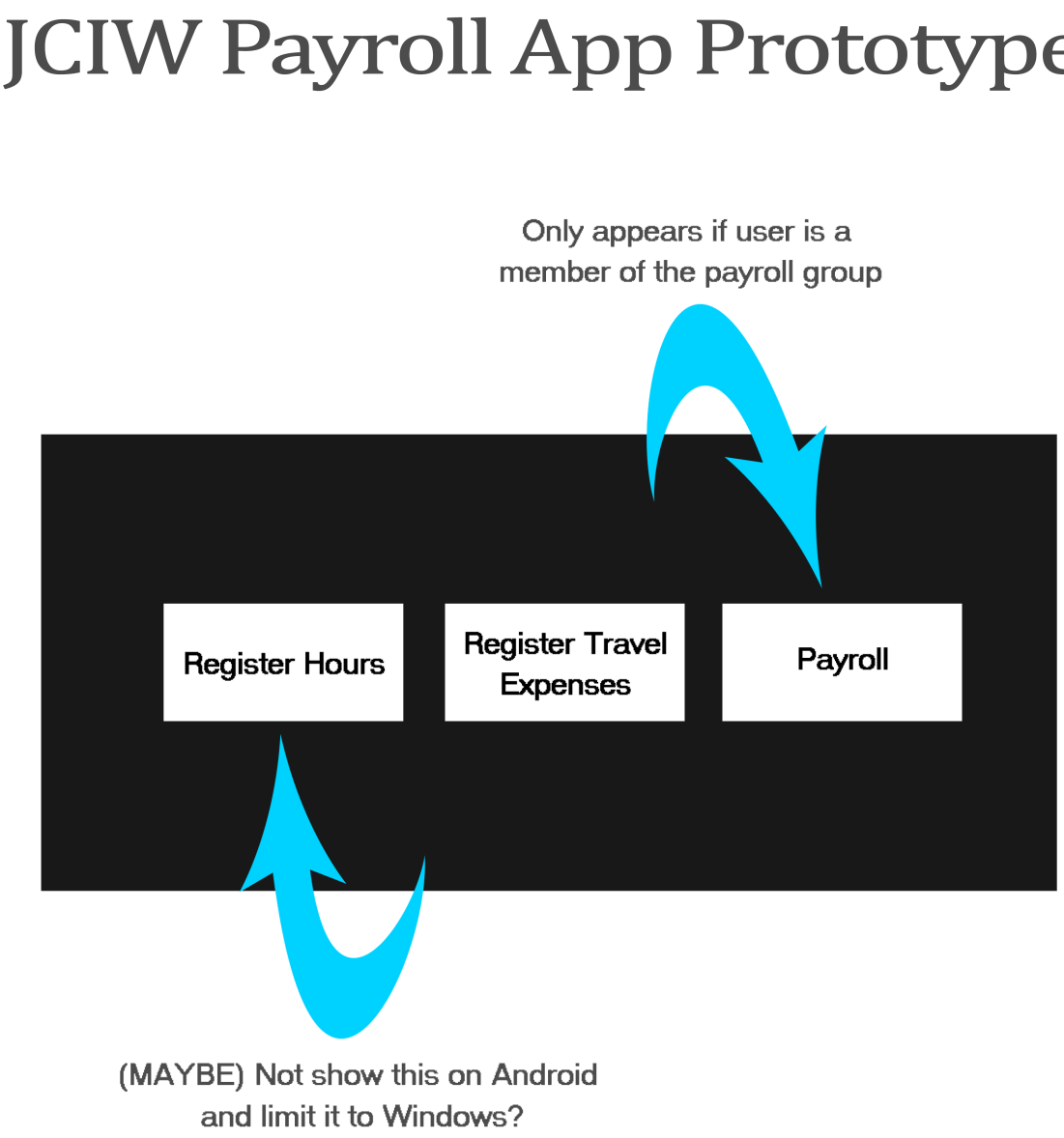


Figure 6: Illutrasjon av Payroll startskjerm

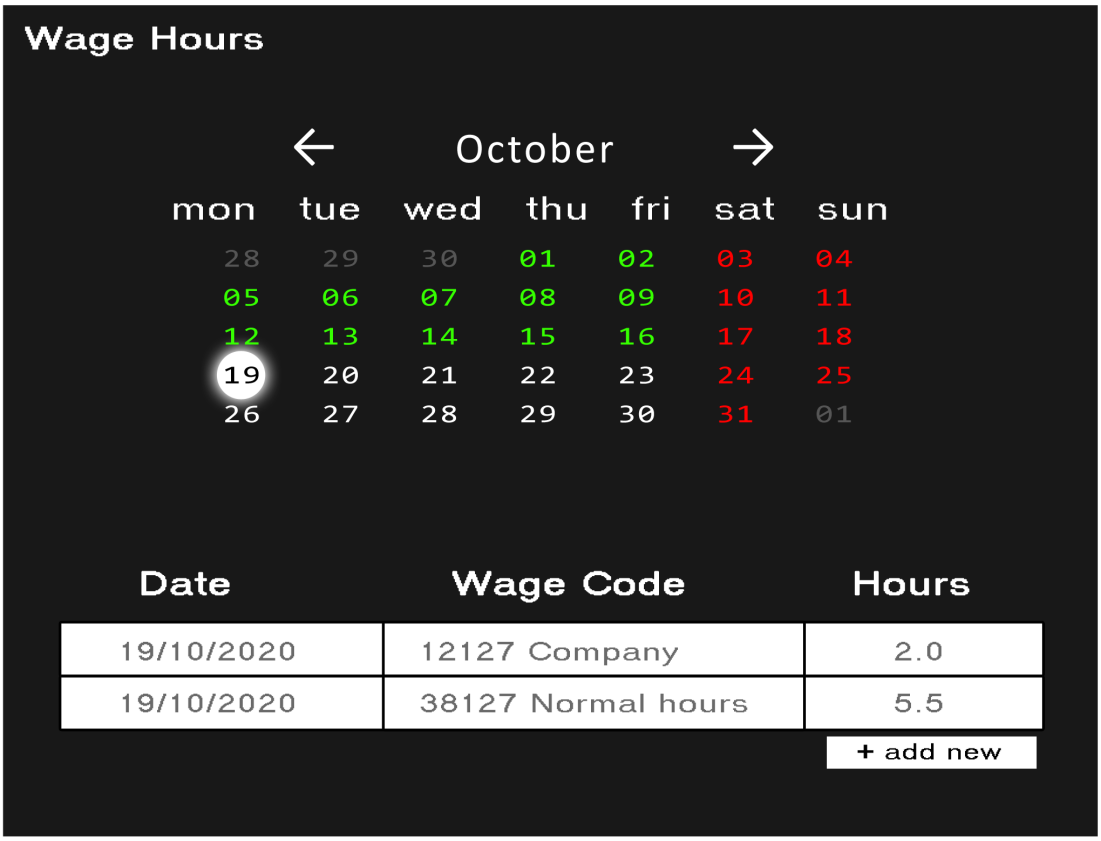


Figure 7: Illustrasjon av Payroll app - Wage Hours

Figure 8: Illustrasjon av Payroll app - Travel Expense

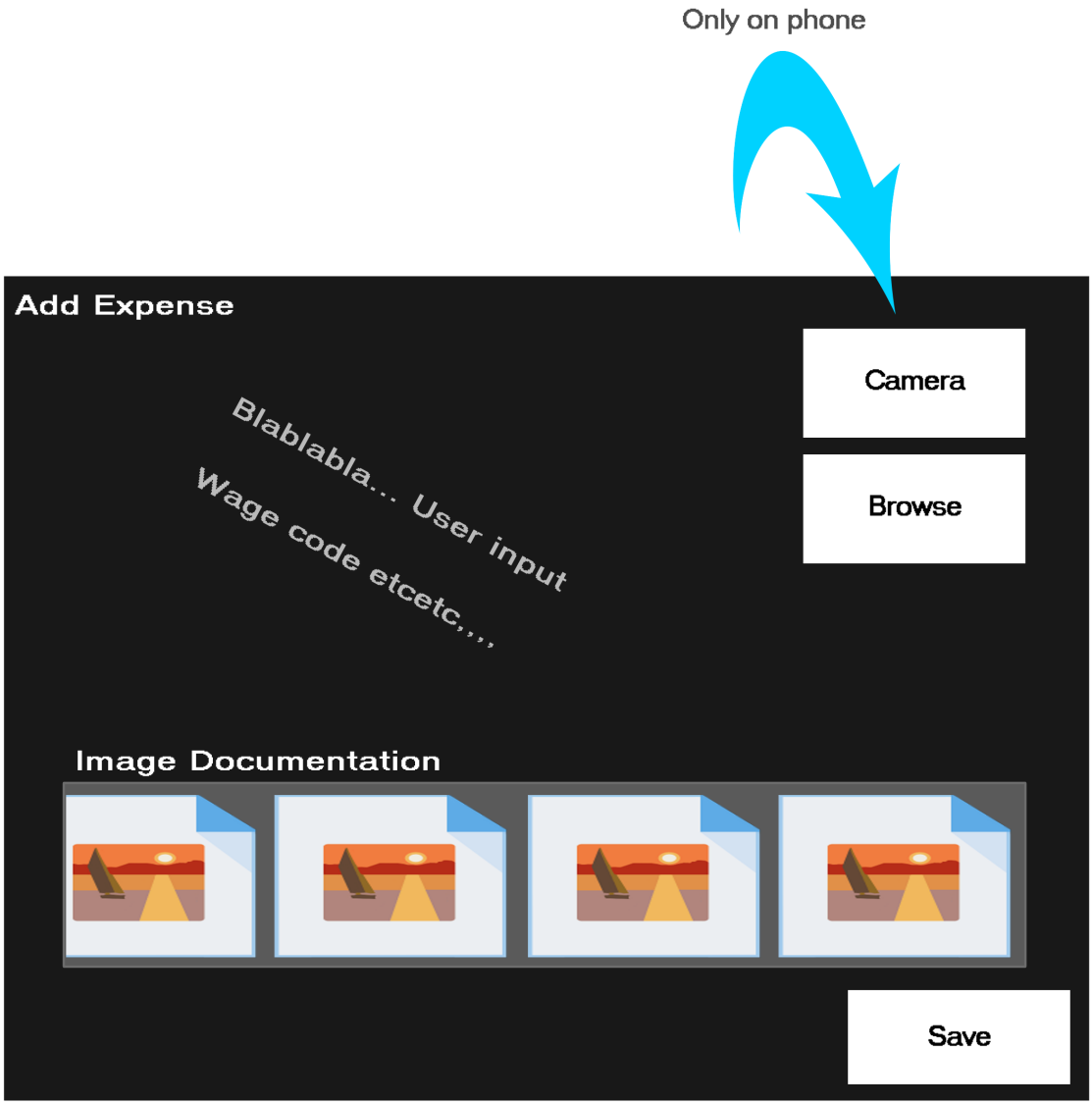


Figure 9: Illustrasjon av Payroll app - Add Expense

1.4 Oppgaven sitt omfang

Systemet vil bestå av følgende tre hoveddeler (med forenklet forklaring).

Backend
Server - Distribuere JCIW applikasjoner og kjøre JCIW tjenester.
Database - Håndtere datalagring.

Frontend
Admin klient - Installere og distribuere applikasjoner og administrere brukere og gruppetilganger.

JCIW Desktop klient - Logge inn og kjøre JCIW applikasjoner.
JCIW Android klient - Logge inn og kjøre JCIW applikasjoner på Android.

Rammeverk
JCIW - Baseklasser for å utvikle JCIW moduler.
JCIW Module - Kjøring av JCIW moduler.

1.5 Rapportens videre innhold

Teoretisk grunnlag: Dette kapittelet inneholder det teoretiske grunnlaget for oppgaven. Kodemønster og arkitektur.
Metoder og Materialer: Dette kapittelet inneholder teknologi og metodikken for løsningen av oppgaven.
Resultater: Dette kapittelet dokumenterer prosessen den ferdige løsningen.
Drøfting: Dette kapittelet inneholder en subjektiv drøfting av resultatene.
Konklusjon: Dette kapittelet inneholder en konklusjon av prosjektet.

2 TEORETISK GRUNNLAG

2.1 Arbeidsflyt

2.1.1 Programvareutviklingsmetode

Scrum er basert rundt en sprintsyklus[2].

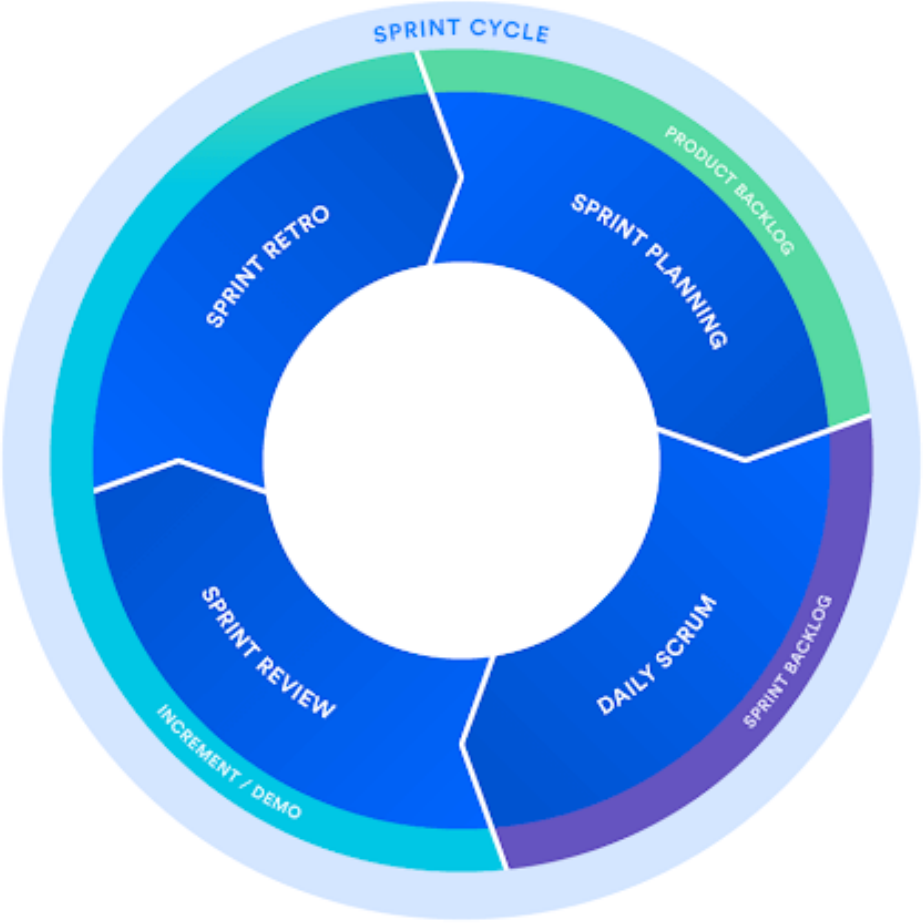


Figure 10: Scrum-syklus

1. Sprint Cycle: En kort periode hvor arbeid blir utført. Vanligvis 2-4 uker.
2. Increment / Demo

   (a) Sprint Review: Ved sprintslutt viser teamet arbeidet som har blitt gjort iløpe av sprint.
   (b) Sprint Retro: Etter sprint review kan teamet gå gjennom hvordan sist sprint har gått og lage en plan for forbedringer.

3. Product & Catalog
   (a) Sprint planning: Ved sprintstart lager teamet en plan over hvilke oppgaver skal løses iløpet av sprinten.

4. Sprint Backlog
   (a) Daily Scrum: Et daglig kort møte hvor gruppen diskuterer arbeidet og diskuterer kommende arbeid til neste møte.

2.2 Objektorientert Programmering

Fra Wikipedia om Objektorientert programmering[3]:

> Object-oriented programming (OOP) is a programming paradigm based on the concept of ”objects”, which can contain data and code: data in the form of fields (often known as attributes or properties), and code, in the form of procedures (often known as methods).
>
> A feature of objects is that an object’s own procedures can access and often modify the data fields of itself (objects have a notion of this or self). In OOP, computer programs are designed by making them out of objects that interact with one another.[1][2] OOP languages are diverse, but the most popular ones are class-based, meaning that objects are instances of classes, which also determine their types.

Objektorientert programmering er designet rundt bruken av objekter.

2.3 MVVM Pattern

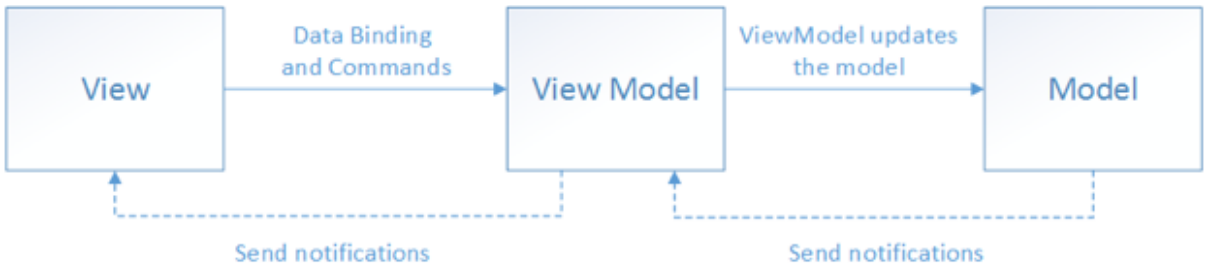


Figure 11: MVVM Pattern

Model-View-Viewmodel er et softwarearkitekturisk pattern som legger til rette for skilling av utviklingen av brukergrensesnittet og back-end-logikken.

Fra Microsoft sin dokumentasjon[4]:

> The Model-View-ViewModel (MVVM) pattern helps to cleanly separate the business and presentation logic of an application from its user interface (UI). Maintaining a clean separation between application logic and the UI helps to address numerous development issues and can make an application easier to test, maintain, and evolve. It can also greatly improve code re-use opportunities and allows developers and UI designers to more easily collaborate when developing their respective parts of an app.

MVVM hjelper med å separere business logikk fra GUI kode.

2.4 Repository Pattern

Fra Microsoft sine sider [5]:

> Use a repository to separate the logic that retrieves the data and maps it to the entity model from the business logic that acts on the model. The business logic should be agnostic to the type of data that comprises the data source layer. For example, the data source layer can be a database, a SharePoint list, or a Web service.
>
> The repository mediates between the data source layer and the business layers of the application. It queries the data source for the data, maps the data from the data source to a business entity, and persists changes in the business entity to the data source.

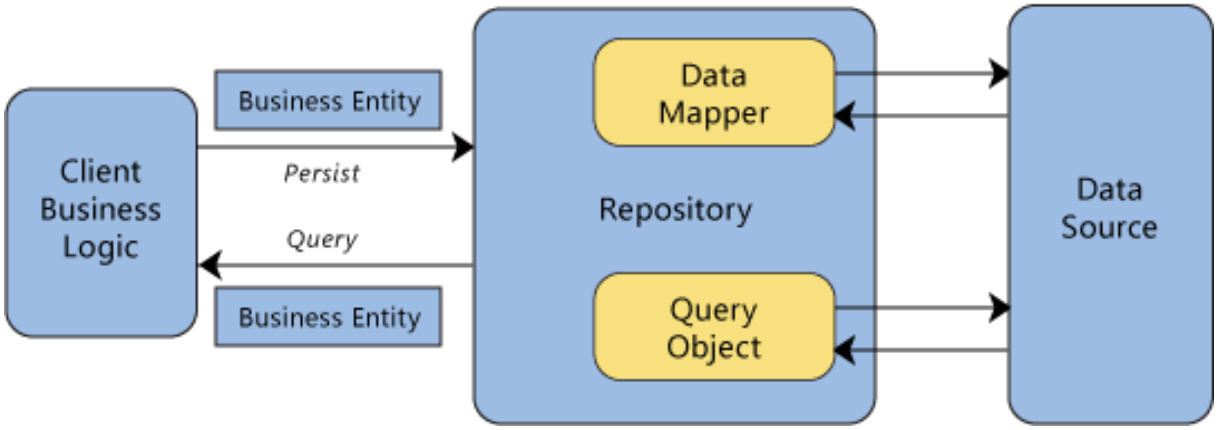


Figure 12: Repository pattern

Ved å bruke repository pattern som lag mellom datakilde (ofte med interface) kan en gjøre det mye enklere i ettetrid å bytte datakilde. For eksempel det å gå fra database til å bruke Microsoft sin Active Directory eller det å bytte databaseteknologi.

2.5 Client-Server Software Architecture

Client-server arkitekturen separerer arbeid mellom nettverksenheter. Klient sender forespørsler og server sender ressurser tilbake.

Illustrasjon fra Wikipedia[6]:

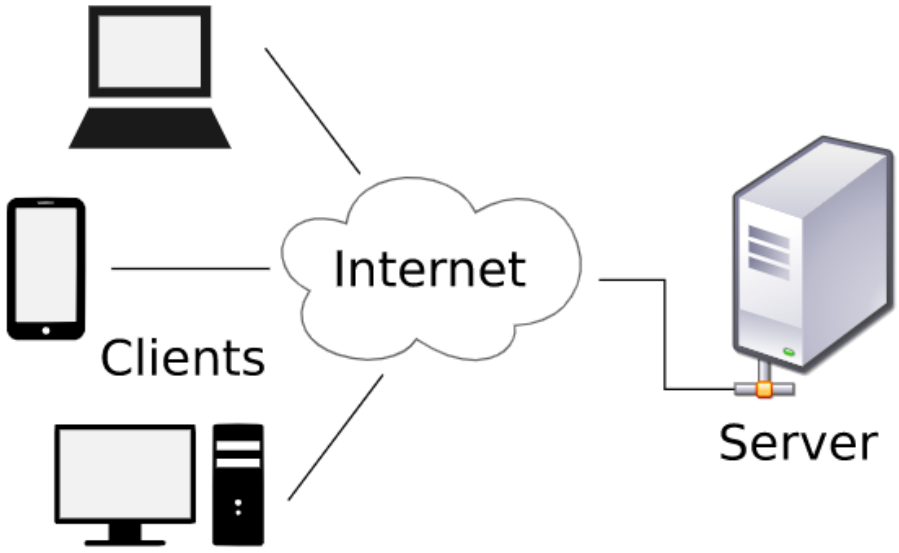


Figure 13: A computer network diagram of clients communicating with a server via the Internet

2.6 Plug-in Software Architecture

Plug-in software arkitektur er en software komponent som laster tillegg/utvidelser til eksisterende programvare.

Illustrasjon fra Wikipedia[7]:

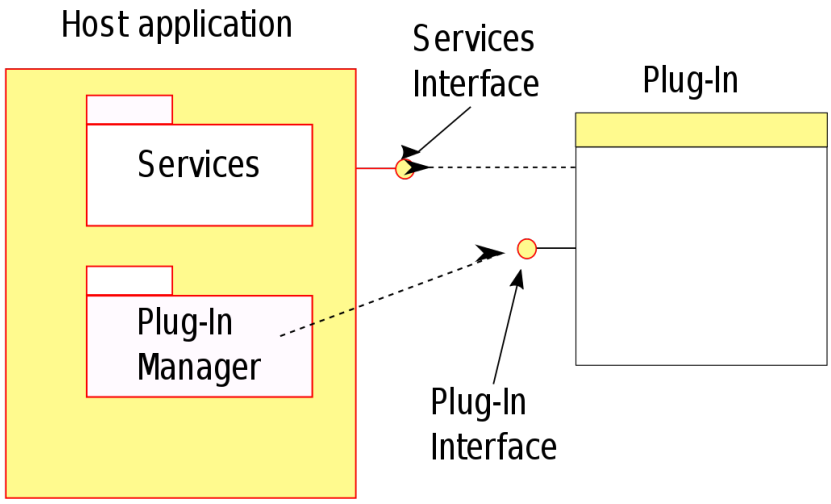


Figure 14: Example Plug-In Framework

3 MATERIALER OG METODER

3.1 Prosjektorganisasjon

3.1.1 Prosjektgruppen

Gruppen består av to datastudenter ved NTNU i Ålesund.

3.1.2 Veileder

Gruppen sin veileder var Arne Styve.

> Arne er utdannet Sivilingeniør ved University of Newcastle upon Tyne innen Mikroelektronikk og Programvareutvikling. Arne har mer enn 20 års erfaring fra norsk ITindustri fra brannvarslingssystemer og digital TV distribusjon til utvikling av kommando kontroll informasjons system (KKIS) for det norske forsvar. Fra 2006 var han med å utvikle avanserte simulatorer for samhandlingstrening innen offshore (ankerhåndtering m.m.) og var med å starte Offshor Simualtor Centre (OSC AS). -*Innsida, Arne Styve*

Veilederen sin rolle i prosjektet var som rådgiver i prosjektgjennomføring og med utviklingsarbeidet. Gruppen møtte med veileder annen hver uke for å diskutere sprintslutt og gruppesamarbeid. I møtene fikk de også råd om verktøy og metoder relatert til systematisering og programmering.

3.1.3 Styringsgruppe

Styringsgruppen for prosjektet bestod bare av veileder.

3.1.4 Rådgivning

Arne Styve og i mindre grad Girts Strazdins fungerte som rådgivere til gruppen.

3.1.5 Prosjektorganisering

Gruppen hadde bare to medlemmer og derfor ikke bruk for å dele ut så mange offisielle roller. I praksis hadde gruppen et flatt hieraki, men på starten av prosjektet ble det delt ut to roller som gruppen trodde ville bli nødvendig.

Den første rollen var prosjektleder. Denne rollen ble tildelt for å styre prosjektet og utviklingsplanen dersom noe uforutsett skulle oppstå. Dette ble ikke nødvendig da gruppen følgte forprosjektet uten avvik og var dermed en ubrukt rolle i praksis.

Den andre rollen var Scrum Master. Det er en Scrum rolle hvor rolletaker har ansvar for at scrum agile metodene blir fulgt gjennom prosjektet.

3.2 Utviklingsmetodikk

Gruppen har følgt den agile utviklingsmetodikken kalt scrum. Se teori for beskrivelse av Scrum. Se resultat for beskrivelse av gruppen sin implementasjon.

3.2.1 Prosjektstyring med Atlassian Jira og Confluence

Gruppen brukte Atlassian Jira og Confluence til prosjektstyring.

Jira ble hovedsakelig brukt til sakshåndtering. Verktøyet ble valgt for dets innebygde agile-funksjoner[8], noe som gjorde det enkelt for gruppen å følge scrum metodikken.

Confluence ble brukt til å skrive sprintrefleksjon og møtereferat. Confluence er et søsterprodukt til Jira og ble valgt for enkelheten av å ha hele prosjektorganiseringen på ett sted.

3.3 Metoder

3.3.1 Valg av rammeverk

JCIW er utviklet i .Net Framework.

Prosjektet krevde mange ulike komponenter som måtte passe sammen. Det ble resonnert av gruppen at det ville hjelpe på effektiviteten dersom prosjektet holdt seg til så få ulike utviklingsmijøer og programmeringsspråk som mulig.

Rammeverket som ble valgt gjorde at gruppen kunne kode opp alle systemene i samme

solution og kompilere alt samtidig. En endring i en dataklasse ville med andre ord oppdateres umiddelbart på både server og klienter (spessielt Android) uten ekstra arbeid eller konfiguering av miljøet. Grafikk og nettverkskode kunne også dermed deles mellom platformer.

.Net Framework med Xamarin ga gruppen mulighet til å kompilere både til Android og iOS og var av den grunn bedre enn for eksempel Android Studio for gruppen sitt bruk.

Mobilkompabilitet var en viktig faktor i gruppen sitt valg av rammeverk. Det hjelpte også at gruppen hadde noe kompetanse i .Net fra før.

3.4 Programvare

3.4.1 Xamarin/Android

Xamarin er en utvidelse av .Net Framework med det formålet å kunne bygge .Net applikasjoner for andre platformer.

Med Xamarin kan mann utvikle hele systemer i C# uavhengig av platform og dele kode mellom dem (Android, iOS og Windows).

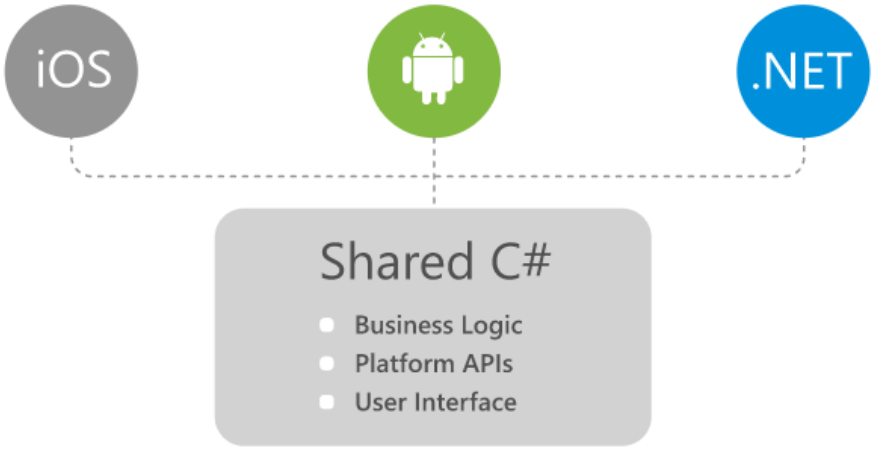


Figure 15: Develop everything in C-sharp[1]

Xamarin kan bygge applikasjoner til iOS, Android, macOS, tvOS, watchOS og mer.

Produktet er gratis og open source.

Måten gruppen planlagte å gå frem med å bygge JCIW Mobil-klienten var å skrive så lite Xamarain / Android spesifikk kode som mulig. Desktop og mobilkoden skulle altså være identisk når det kom til å tegne til skjermen, user input og nettverkskall. Der var bare èn mobilspesifikk funksjon som måtte utvikles spesfikt for Android og det var kamerafunksjonen på grunn av at gruppen ønsket å kunne ta bilder fra JCIW apper.

Android applikasjonen består av følgende avhengigheter:
Nettverksimplementasjon for Android:
Client-Networking.Android

Brukergrensesnitt:
ImGui.NET

Grafikk backend:
MonoGame
GraphicsEngine

For å kjøre JCIW koden:
JCIW
JCIW.App
JCIW.Data
JCIW.Module
Networking.Data

Dette er de samme avhengighetene som desktopklienten har, men de har ulik implementasjon. Begge implementerer en interface kalt IPlatformFunctions for at JCIW skal ha tilgang til ting som å åpne kamera, registrere tastetrykk og hente ut pikseltettheten på skjermen. Les mer på dette i resultater.

Kamerafunksjonen bruker Android sin Camera2 som den viser i en ny aktivitet. Les mer på dette i resultater.

Til slutt er der en del eksterne biblioteker som brukes. Der er to sett med eksterne biblioteker som blir brukt. Det første settet er AndroidNativeLibrary for cimgui. Dette er C++ kode som må kompileres via Android Studio. Uten disse fungerer ikke brukegrensesnittet. Les mer på dette i resultater.

Det andre settet er JCIW avhengighetene. Disse må kunne kjøres av appene som lastes ned og må dermed være tilgjengelig på disken. Altså ikke pakket inn i en APK. Når man bygger i release mode, og slår av Use Shared Runtime så ligger ikke disse bibliotekene tilgjengelig på disken lenger. Les mer på dette i resultater.

3.5 Materialer og utstyr

3.5.1 Linux VPS

En privat Debian GNU/Linux 8.11 (jessie) VPS ble brukt til servertjenester under prosjektet.

3.6 Distribuering av applikasjon

3.7 Git

Git, skapt av Linus Torvalds, er en distribuert versjonskontroll for å kunne spore endringer i enkeltfiler. Git var oprinnelig skapt for å koordinere arbeid mellom ulike programmerere under softwareutvikling, men brukes nå også for å ha kontroll på ulike versjoner av et softwareprosjekt.

3.7.1 Atlassian Bitbucket

Bitbucket frå Atlassian er ein web-basert versjonskontroll som bruker Git-servere.

3.8 .Net Framework

Fra Microsoft sin dokumentasjon[9]:

> `.NET is a developer platform made up of tools, programming languages, and libraries for building many different types of applications.`
>
> `There are various implementations of .NET. Each implementation allows .NET code to execute in different places|Linux, macOS, Windows, iOS, Android, and many more.`

.Net Framework er platformen JCIW er bygget på.

3.9 Visual Studio

Visual Studio er et integrert utviklingsmiljø (IDE) laga av Microsoft, som i hovedsak støtter programutvikling med bruk av C# med .Net rammeverk, men den støtter også programmeringsspråkene C++ og Visual Basic, samt Python.

3.10 SQLite

Fra SQLite sin hjemmeside[10]:

> SQLite is a C-language library that implements a small, fast, self-contained. high-reliability, full-featured, SQL database engine. QLite is the most used database engine in the world. SQLite is built into all mobile phone and most computers and comes bundled inside countless oter applications that people use every day.
>
> The SQLite file format is stable, cross-platform, and backwards compatible and the developers pledge to keep it that way through the year 2050. SQLite database files are commonly used as containers to transfer rich content between systems and as a long-term archival format for data. There are over 1 trillion SQLite databases in active use.
>
> SQLite source code is in the public-domain and is free to everyone to use for any purpose.

3.11 MonoGame

Fra MonoGame sin Github[11]:

> MonoGame is a simple and powerful .NET library for creating games for desktop PCs, video game consoles, and mobile devices.
>
> Based on Microsoft's XNA Framework, it provides the following features:
>
> Game framework 2D and 3D rendering Sound effect and music playback Keyboard, mouse, touch, and controller inputs Content building and optimization Math library optimized for games

MonoGame er en Open Source og brukes av prosjektet til å tegne ImGui til å tegne brukergrensesnittet til skjermen.

3.12 Dear ImGui

Fra GitHub prosjektbeskrivelsen[12]:

> Dear ImGui is a bloat-free graphical user interface library for C++.
> It outputs optimized vertex buffers that you can render anytime in
> your 3D-pipeline enabled application. It is fast, portable, renderer
> agnostic and self-contained (no external dependencies).
>
> Dear ImGui is designed to enable fast iterations and to empower
> programmers to create content creation tools and
> visualization / debug tools (as opposed to UI for the average end-user).
> It favors simplicity and productivity toward this goal, and lacks
> certain features normally found in more high-level libraries.
>
> Dear ImGui is particularly suited to integration in games engine
> (for tooling), real-time 3D applications, fullscreen applications,
> embedded applications, or any applications on consoles platforms
> where operating system features are non-standard.

Formålet med å bruke ImGui er å gi modulutviklere en enkel måte og lage robuste grensesnitt til applikasjonene sine. Det var også viktig i valget av brukergrensesnittbibliotek at grensesnitt og kontroller kunne skaleres siden det skal fungere på mobil som har høy pikseltetthet. Det ble først testet å lage kontroller direkte selv i MonoGame, men av tidshensyn så vel som begrensningene på antall kontroller som gruppen kunne fått laget så ble det bestemt å finne et ferdiglaget bibliotek å bruke.

3.12.1 cimgui

Fra GitHub prosjektbeskrivelsen:

> This is a thin c-api wrapper programmatically generated for the excellent C++
> immediate mode gui Dear ImGui. All imgui.h functions are programmatically
> wrapped. Generated files are: cimgui.cpp, cimgui.h for C compilation. Also
> for helping in bindings creation, definitions.lua with function definition
> information and structs_and_enums.lua. This library is intended as a
> intermediate layer to be able to use Dear ImGui from other languages
> that can interface with C (like D - see D-binding)

cimgui er en tynn c-api wrapper som har blitt generert for GUI'et Dear ImGui[13].

Bibliotekfiler til Windows og Linux var allerede tilgjengelig, men manglet for Android. Dette måtte altså kompileres selv.

3.12.2 ImGui.NET

Fra GitHub prosjektbeskrivelsen:

> This is a .NET wrapper for the immediate mode GUI library, Dear ImGui (https://github.com/ocornut/imgui). ImGui.NET lets you build graphical interfaces using a simple immediate-mode style. ImGui.NET is a .NET Standard library, and can be used on all major .NET runtimes and operating systems.
>
> Included is a basic sample program that shows how to use the library, and renders the UI using Veldrid, a portable graphics library for .NET. By itself, Dear ImGui does not care what technology you use for rendering; it simply outputs textured triangles. Example renderers also exist for MonoGame and OpenTK (OpenGL).
>
> This wrapper is built on top of cimgui, which exposes a plain C API for Dear ImGui. If you are using Windows, OSX, or a mainline Linux distribution, then the ImGui.NET NuGet package comes bundled with a pre-built native library. If you are using another operating system, then you may need to build the native library yourself; see the cimgui repo for build instructions.

ImGui.Net er en .Net wrapper for cimgui[14].

ImGui.NET er en wrapper for cimgui og lar .Net kommunisere med biblioteket. Det var også kritisk at den hadde en MonoGame implementasjon som kunne følges sånn det ikke trengtes å gjøres fra grunnen av. Det fant gruppen her: https://github.com/mellinoe/ImGui.NET/tree/master/src/ImGui.NET.SampleProgram.XNA

3.13 Networkcomms.Net

Networkcomms.Net er et multiplatform .Net nettverksbibliotek. Det er laget med hensikten å gjøre nettverksprogrammeringen så enkel som mulig for utvikleren. Gruppen har brukt NetworkComms.net med google sin Protobuf protokoll til å sende pakker over TCP.

Fra NetworkComms.net sin nettside[15]:

> The motivation behind NetworkComms.Net was to create a fully featured .Net C# network library with which network functionality could be effortlessly added to any .net network application. i.e. C#, VB .Net etc. We wanted to

require little to no knowledge of networking to make things work first time alongside all of the desirable power features for more experienced network developers. If you check out our feature list below and tutorials we hope you will agree that these aims have been met.

NetworkComms.Net contains the following features:

- TCP (including SSL), UDP & Bluetooth (Bluetooth only on native .net 3.5+ platforms).
- IPv4 & IPv6.
- Unmanaged connections for interfacing with embedded controllers and external libraries.
- Integrated serialisation, compression & encryption.
- Data prioritisation.
- Peer discovery.
- RPC (Remote Procedure Call) capabilities (.net 3.5+).
- Connection and peer security features (i.e. DOS protection).
- Completely thread-safe.
- Extensive number of tools, useful for networking features in your applications.
- Extremely flexible usage-cases.
- Support for sending v.large (1TB) files.
- Multiple network adapter support.
- Full logging and debugging capabilities.

3.14 Android

Android er et mobilt operativsystem basert på linux kjernen. Gruppen ville utvikle til iOS også, men hadde ikke Mac PCer eller mobiler å teste med. Android 10 mobiler er derfor eneste mobilene som har blitt utviklet mot.

3.15 WPF

Windows Presentation Foundation er en feature i Visual Studio for utvikling av grafiske Desktop applikasjoner på Windows.

Fra Microsoft sine sider[16]:

> Windows Presentation Foundation (WPF) lets you create desktop client applications for Windows with visually stunning user experiences.
>
> The core of WPF is a resolution-independent and vector-based rendering engine that is built to take advantage of modern graphics hardware. WPF extends the core with a comprehensive set of application-development features that include Extensible Application Markup Language (XAML), controls, data binding, layout, 2D and 3D graphics, animation, styles, templates, documents, media, text, and typography. WPF is part of .NET, so you can build applications that incorporate other elements of the .NET API.

WPF er designet for å gjøre det enkelt å kode med MVVM kodemønsteret. WPF er også open source.

3.16 System.Reflection

Fra Microsoft sin API dokumentasjon:

> `Contains types that retrieve information about assemblies, modules, members, parameters, and other entities in managed code by examining their metadata. These types also can be used to manipulate instances of loaded types, for example to hook up events or to invoke methods. To dynamically create types, use the System.Reflection.Emit namespace.`

System.Reflection er et bibliotek som (blant annet) kan laste og kjøre eksterne .Net biblioteker[17].

3.17 System.AppDomain

Fra Microsoft sin API dokumentasjon:

> Represents an application domain, which is an isolated environment where applications execute. This class cannot be inherited.
>
> Application domains, which are represented by AppDomain objects, help provide isolation, unloading, and security boundaries for executing managed code.
>
> Use application domains to isolate tasks that might bring down a process. If the state of the AppDomain that's executing a task becomes unstable, the AppDomain can be unloaded without affecting the process. This is important when a process must run for long periods without restarting. You can also use application domains to isolate tasks that should not share data.
>
> If an assembly is loaded into the default application domain, it cannot be unloaded from memory while the process is running. However, if you open a second application domain to load and execute the assembly, the assembly is unloaded when that application domain is unloaded. Use this technique to minimize the working set of long-running processes that occasionally use large DLLs.

System.AppDomain er et bibliotek som lar utvikleren laste og kjøre eksterne biblioteker i et domene adskilt fra programmet.[18].

3.18 Windows Forms

Winbdows Forms er en Visual Studio feature for utvikling av Windows desktopapplikasjoner.

Fra Wikipedia om Windows Forms[19]:

> Windows Forms (WinForms) is a free and open-source graphical (GUI) class library included as a part of Microsoft .NET Framework or Mono Framework,[1] providing a platform to write rich client applications for desktop, laptop, and tablet PCs.[2] While it is seen as a replacement for the earlier and more complex C++ based Microsoft Foundation Class Library, it does not offer a comparable paradigm[3] and only acts as a platform for the user interface tier in a multi-tier solution.[4]

> At the Microsoft Connect event on December 4, 2018, Microsoft announced releasing Windows Forms as an open source project on GitHub. It is released under the MIT License. With this release, Windows Forms has become available for projects targeting the .NET Core framework. However, the framework is still available only on the Windows platform, and Mono's incomplete implementation of Windows Forms remains the only cross-platform implementation.[5][6]

Windows Forms er en enklere måte å lage brukergrensesnitt (drag & drop) enn WPF. Blir brukt til brukeradministrering og lignende oppgaver i Adminklienten.

3.19 DocFX

DocFX er et verktøy for å generer statiske dokumentasjon fra Visual Studio prosjekter (blant annet).

Det kan installeres for Visual Studio og bygges på samme måte som vanlige prosjekter bygges.

3.20 SonarLint

SonarLint er et tillegg til Visual Studio som gruppen brukte til å rydde og skrive bedre kode.

3.21 Visual Paradigm

Visual Paradigm er et verktøy som gruppen brukte til å tegne klassediagram.

4 RESULTATER

4.1 Systemarkitektur

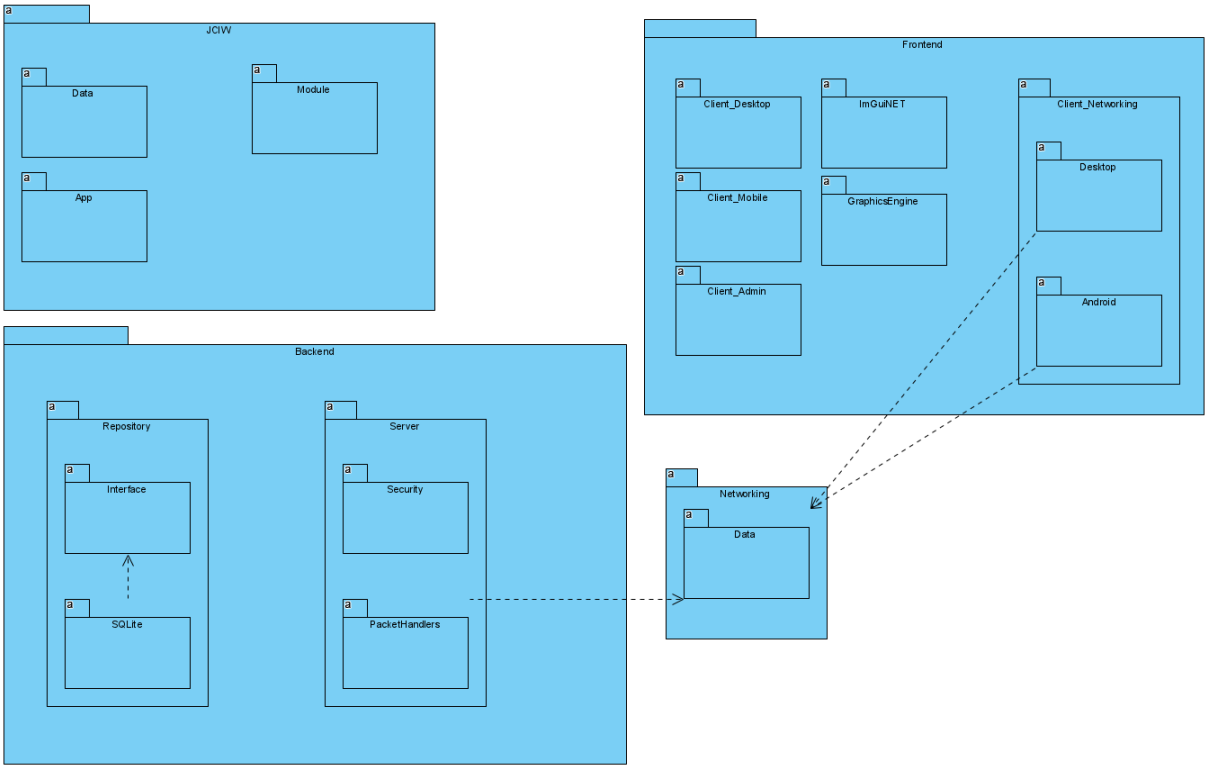


Figure 16: All projects in JCIW Solution

JCIW: Prosjektene som utgjør JCIW rammeverket. Modullasting og JCIW klient.

Frontend: Prosjektene som utgjør frontend. Brukerklient og adminklient.

Backend: Prosjektene som utgjør backend. Server og database.

Networking: Data prosjektet. Pakker og pakkenavn.

4.2 Database

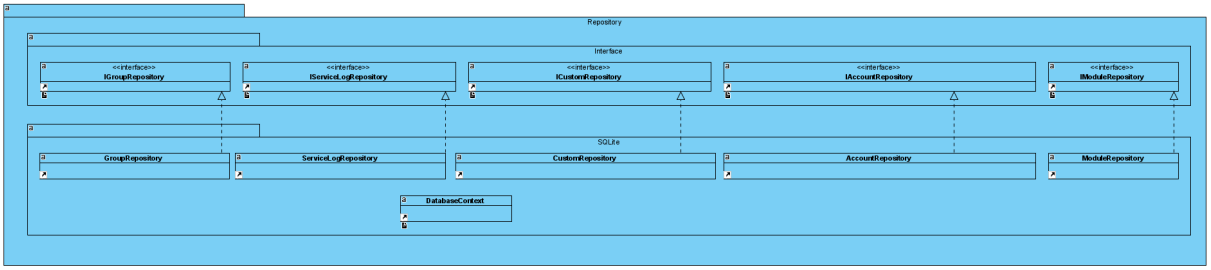


Figure 17: Repository klassene

IAccountRepository: Klasse for å lagre / lese brukerinformasjon.

ICustomRepository: Klasse for å gi service lese / skrivemuligheter.

IGroupRepository: Klasse for å lagre / lese gruppeinformasjon.

IModuleRepository: Klasse for å lagre / lese modulinformasjon.

IServiceLogRepository: Klasse for å lagre / lese service logger.

Repository.SQLite er implementeringen av Repository klassene.

DatabaseContext: For å åpne skrivetilgang til SQLite databasefil.

Databasen er kodet med databasepattern. Det gjør at databaseteknologien enkelt kan byttes ut ved å implementere Repository interfacene i et nytt prosjekt.

4.2.1 SQLite database

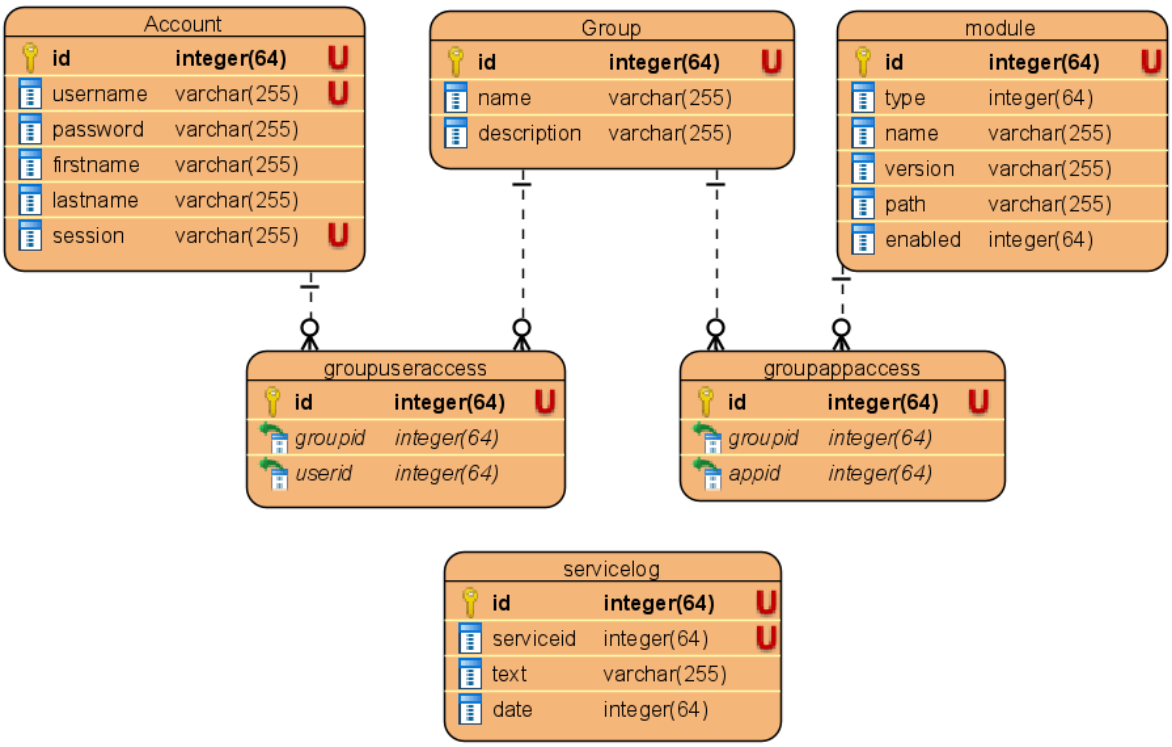


Figure 18: ER-diagram av databasen

Enitetene:

- **ACCOUNT** = bruker eller brukerkonto
- **GROUP** = gruppe
- **MODULE** = modul eller app

Relasjon mellom de ulike entitetene:

- En **bruker** kan være medlem av flere **grupper**
- En **gruppe** kan ha null eller flere **brukere** som medlemmer
- En **gruppe** kan ha tilgang til null eller flere **apper**
- En **app** kan gi tilgang til null eller flere **grupper**

4.3 JCIW Klient

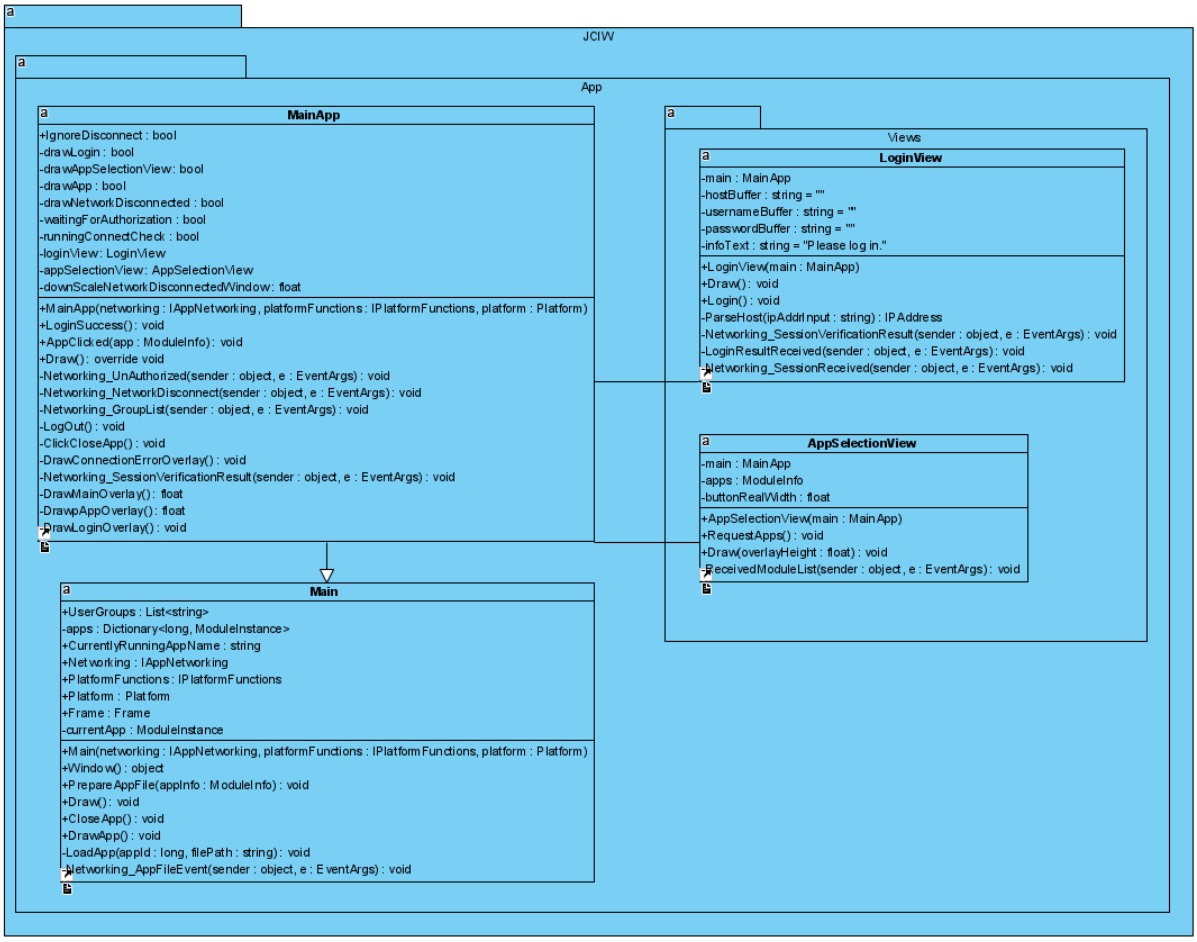


Figure 19: JCIW.App klassene

JCIW klientkoden er i JCIW.App prosjektet (se diagram over).

Main: MainApp:

LoginView: AppSelectionView:

Denne klassen står for innlogging og lastingen / visningen av apps. JCIW klienten, på samme måte som modulene, er laget med ImGui. Den er kodet på samme måte som en app modul ville blitt laget.

4.3.1 Bilder av klienten

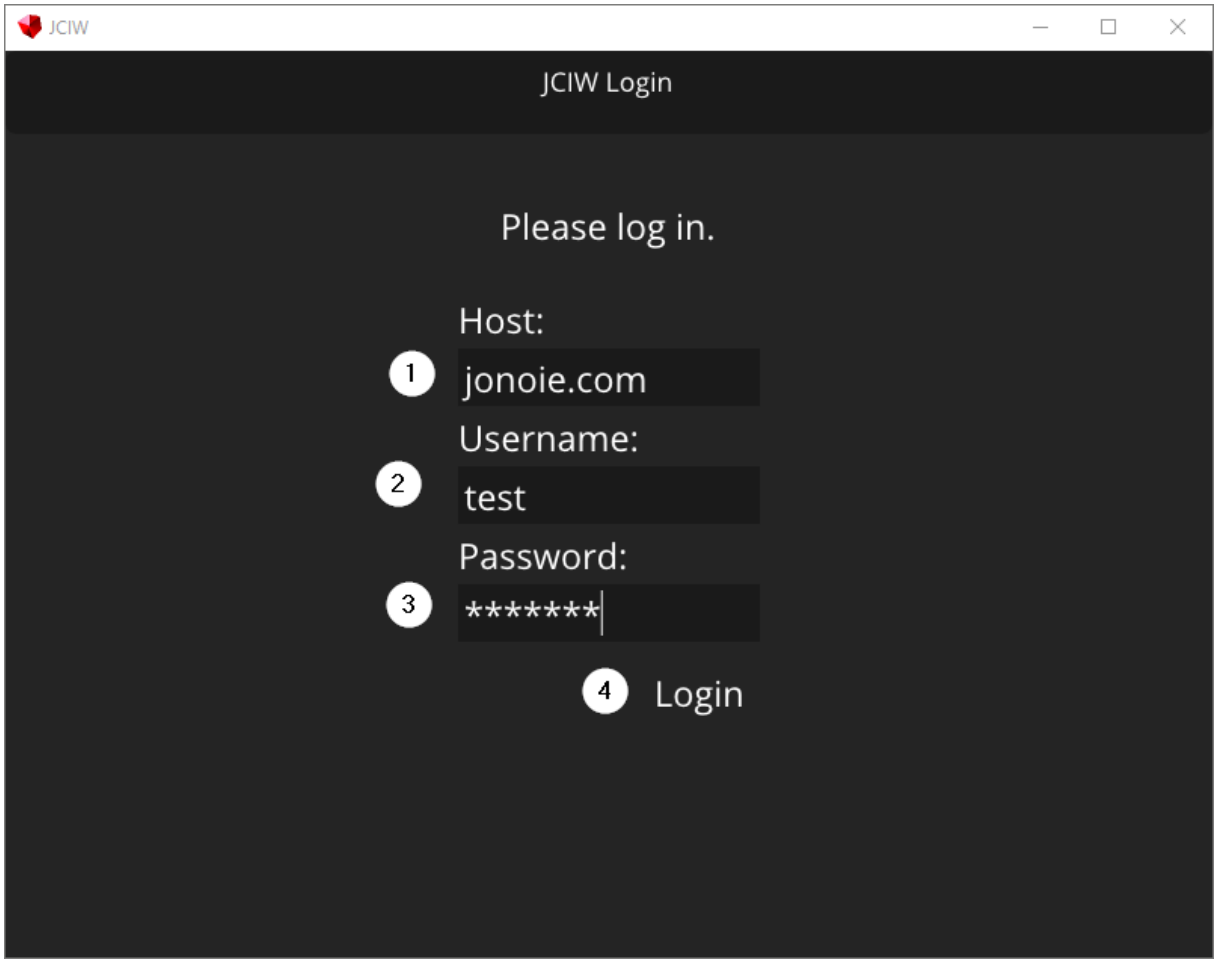


Figure 20: JCIW Login

1. IP Adresse eller domenenavn til JCIW server.
2. Brukernavn
3. Passord
4. Logg inn

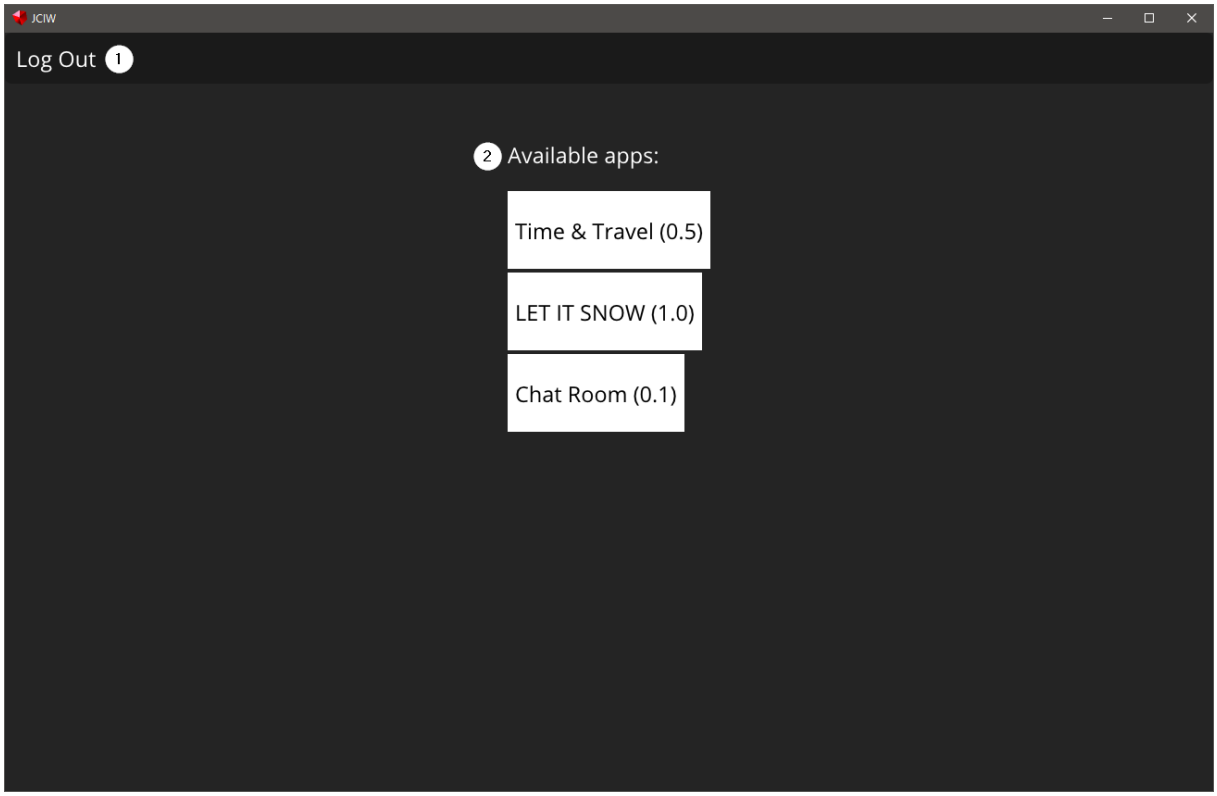


Figure 21: App Selection View

1. App selection overlay. Inneholder bare knapp for å logge ut.
2. App selection liste. Skalerer og sentrerer seg etter vindustørrelse. Scrollbar dukker opp dersom listen er høyere enn vinduet.

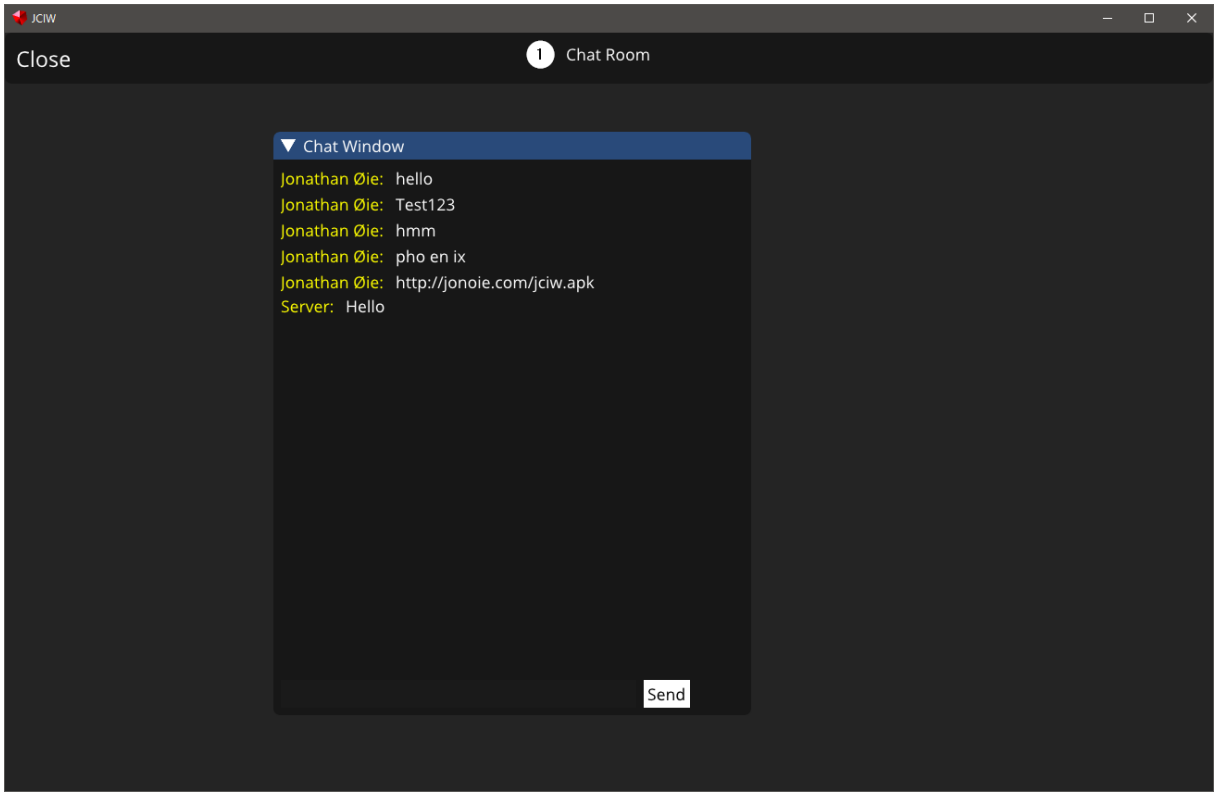


Figure 22: App view

1. App view overlay. Dette overlayet tegnes over alle applikasjoner som lastes. Den viser navnet på applikasjonen som er lastet og en close knapp for å gå tilbake til app selection listen. Alle kontroller som legges til vil automatisk bli tegnet under baren. Med mindre brukeren spesifikt legger inn posisjon på vinduet med SetCursorPosY.

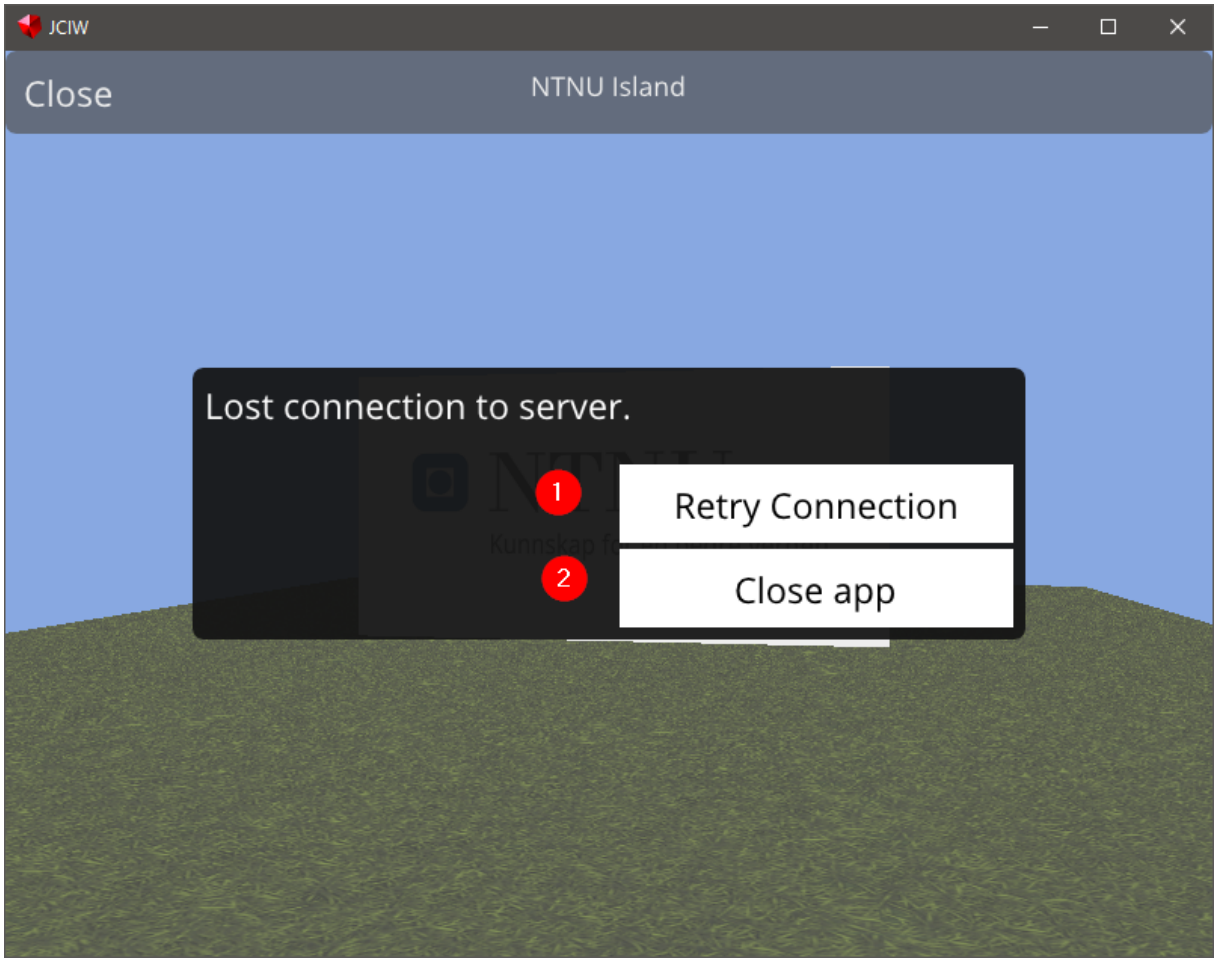


Figure 23: Lost connection to server

1. Forsøker å koble til server. Dersom en får kontakt med server og brukeren var innlogget så sender den session ID. Dersom session ID er ugyldig blir brukeren sendt til innloggingsbildet. Dersom ID er gyldig forsvinner overlayet.
2. Lukk programmet.

4.3.2 Time & Travel demo app

Gruppen lagde en demo app kalt Time & Travel for å vise frem noen av JCIW sin funksjoner.

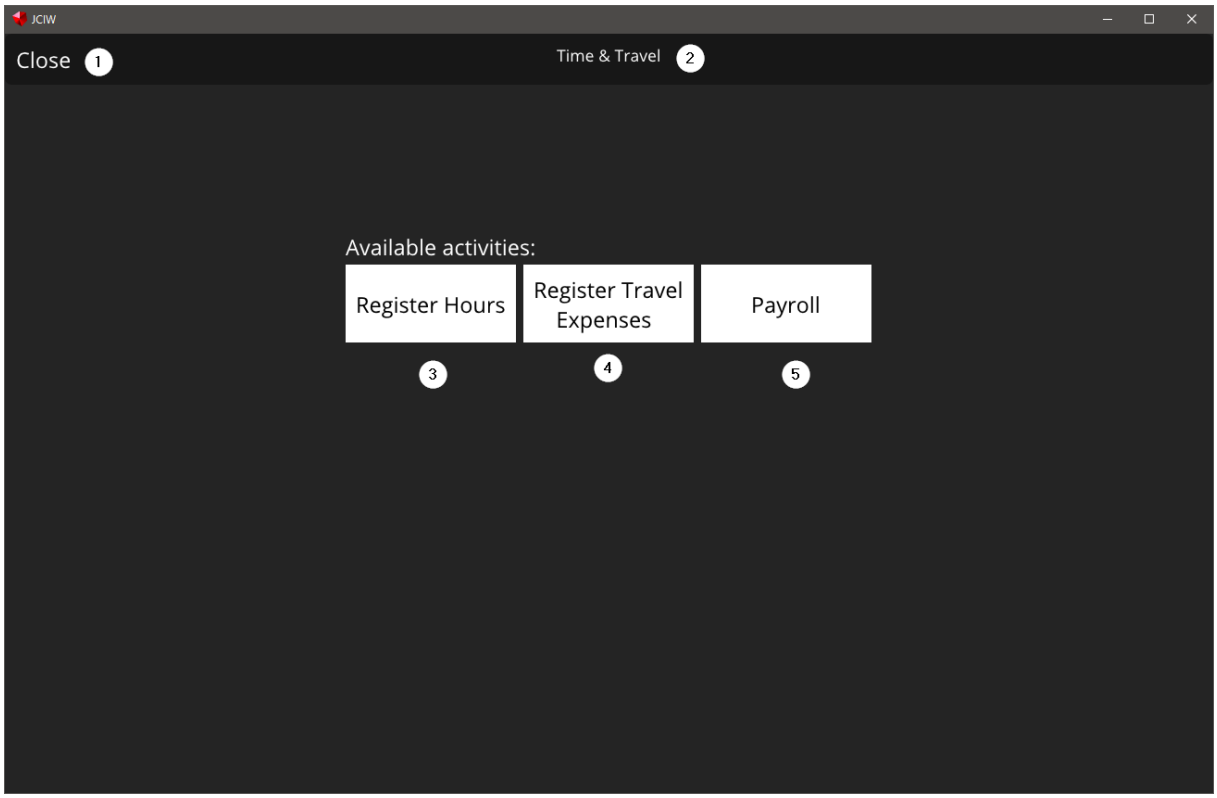


Figure 24: Time & Travel App selection view

1. Lukk Time & Travel app
2. App tittel
3. Åpne register hours viewet til Time & Travel
4. Åpne register travel expenses viewet til Time & Travel
5. Åpne payroll viewet til Time & Travel. Denne dukker bare opp dersom brukeren er medlem i Payroll gruppen og bruker desktopklienten.

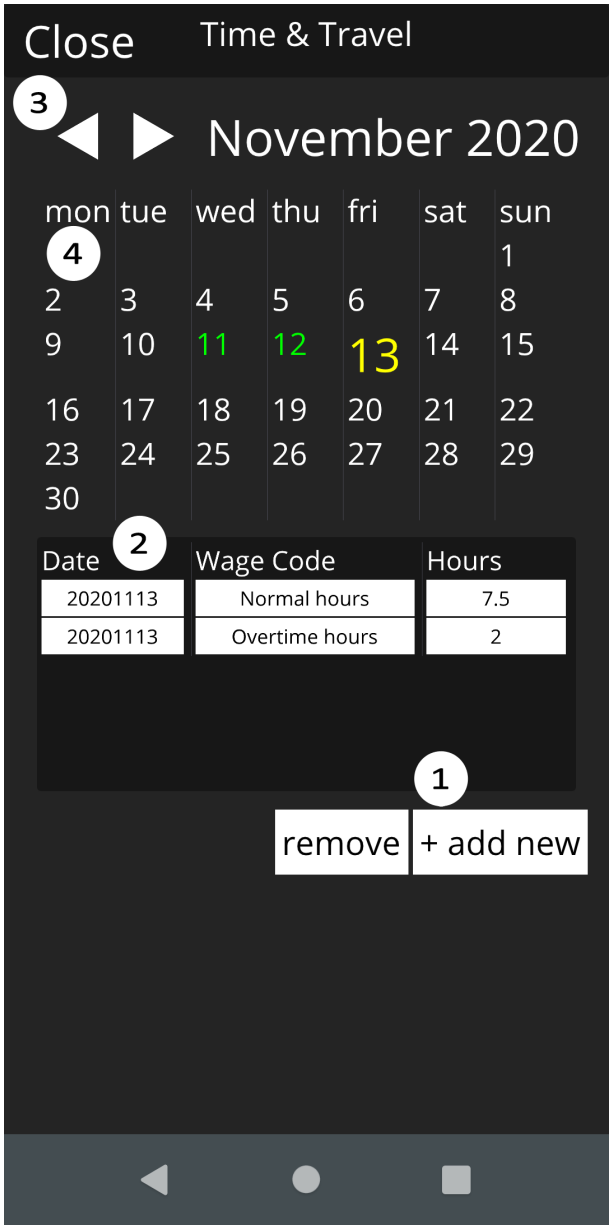


Figure 25: Wage Hour view on Time & Travel app

1. Åpne overlay for å legge til timer på dato.
2. Den valgte datoen sine timeføringer.
3. Gå en måndte tilbake.
4. Kalendermåndte. Gul er den valgte datoen. Grønn er dagene som har timeføringer registrert på seg.

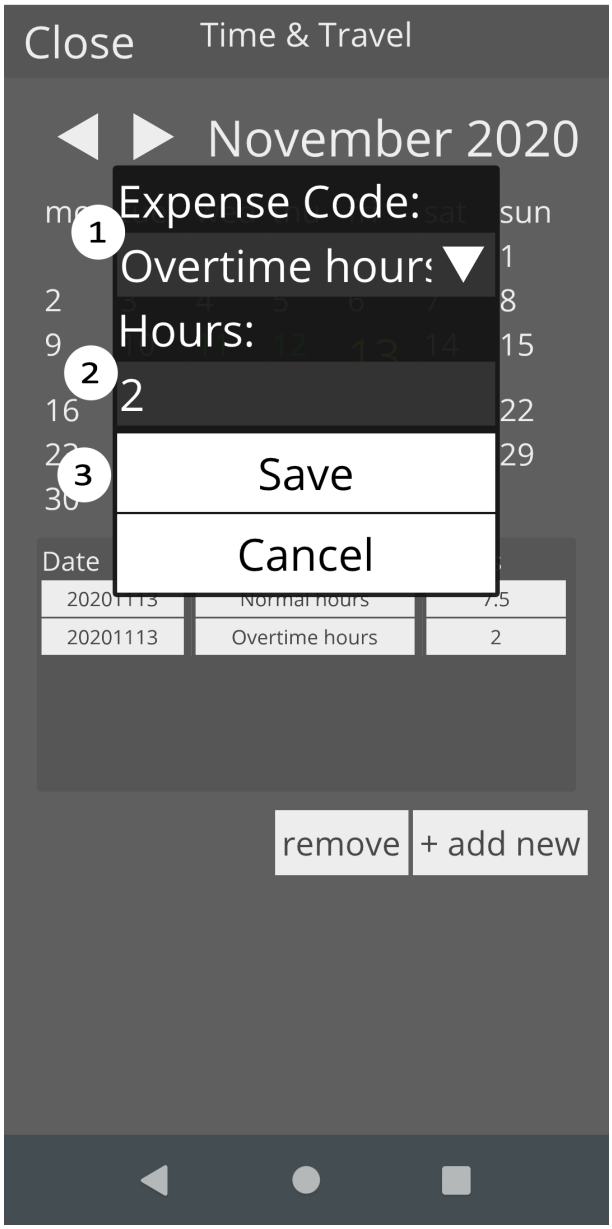


Figure 26: Add wage hours on Wage Hour view on Time & Travel app

1. Liste over utgiftskoder.
2. Antall timer å timeføre.
3. Lagre / send til server.

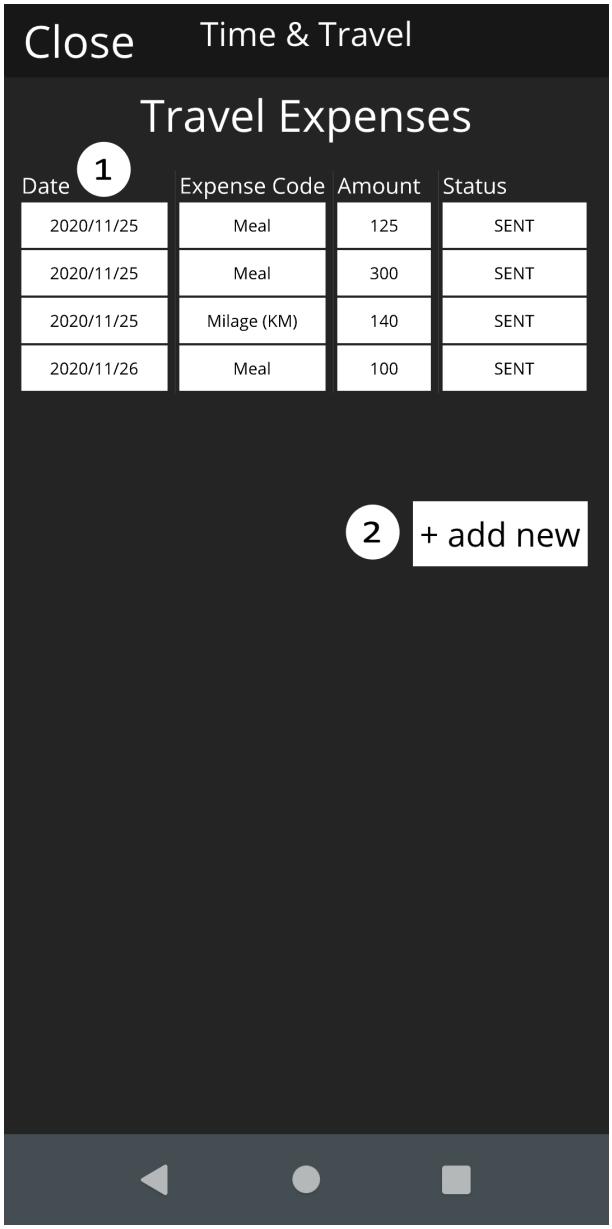


Figure 27: Travel Expenses view on Time & Travel app

1. Travel expense liste.
2. Åpne popup for å legge til ny travel expense.

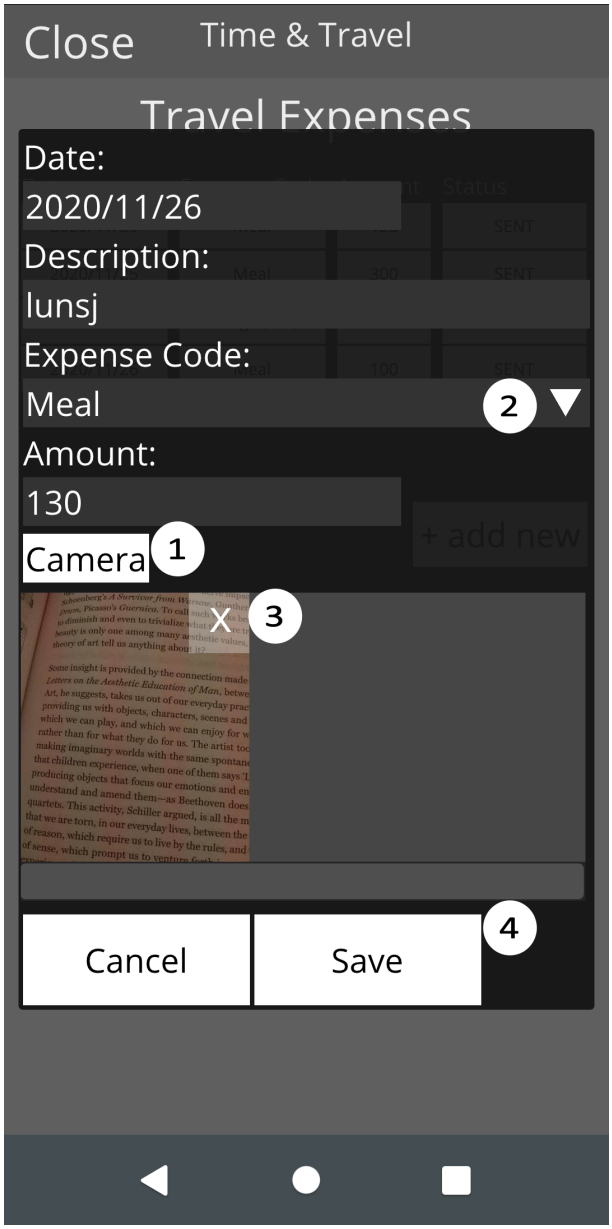


Figure 28: Add expense view on Travel Expenses view on Time & Travel app Mobile

1. Kamerknapp. På Android dukker Camera opp og på desktop dukker Browse opp. Åpner kamera aktivitet for at brukeren skal kunne ta bilde av kvittering.
2. Liste over utgiftskoder.
3. Liste over bilder å laste opp.
4. Lagre / send til server.

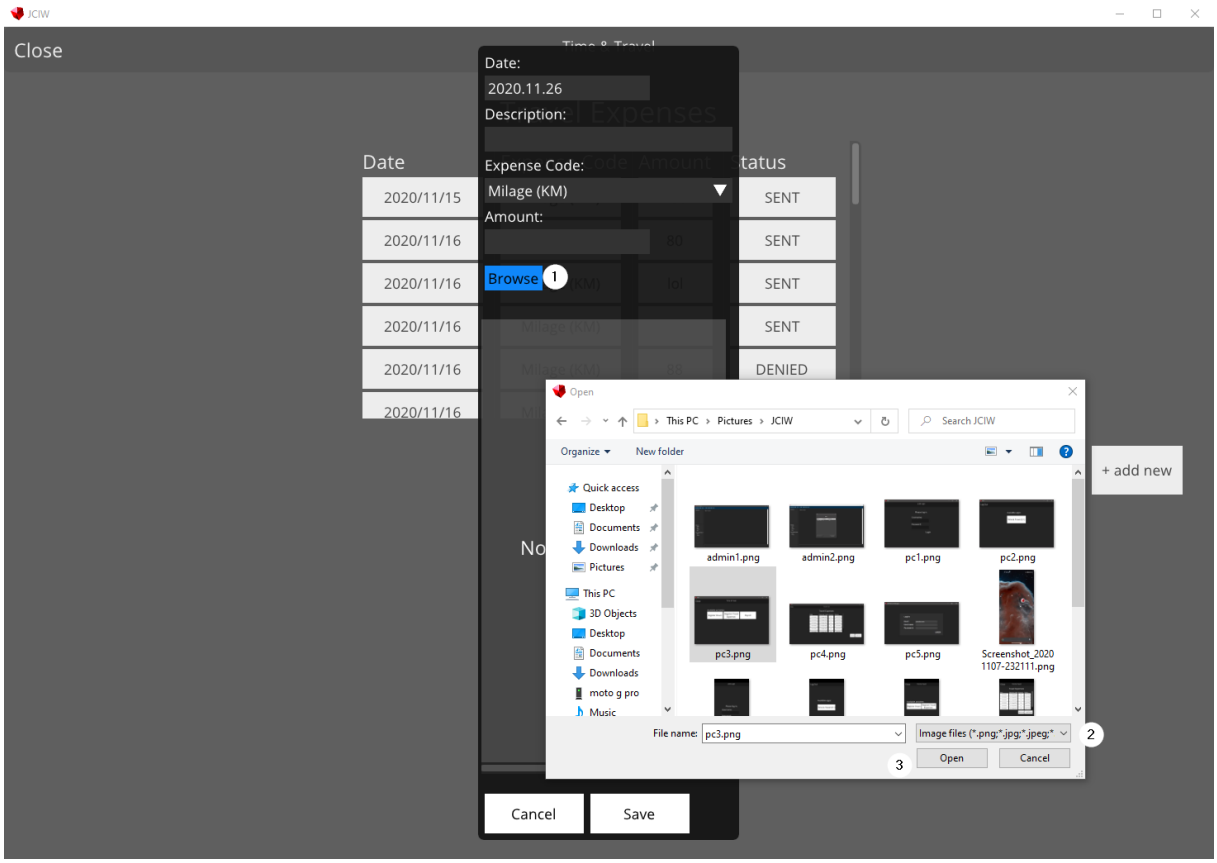


Figure 29: Add expense view on Travel Expenses view on Time & Travel app Desktop

1. Browse knapp. Åpner browse vindu på dekstop for å laste bilder til opplasting.
2. Browse vindu støtter filter for å bare vise visse type filer.
3. Save / legg til i bildeliste.

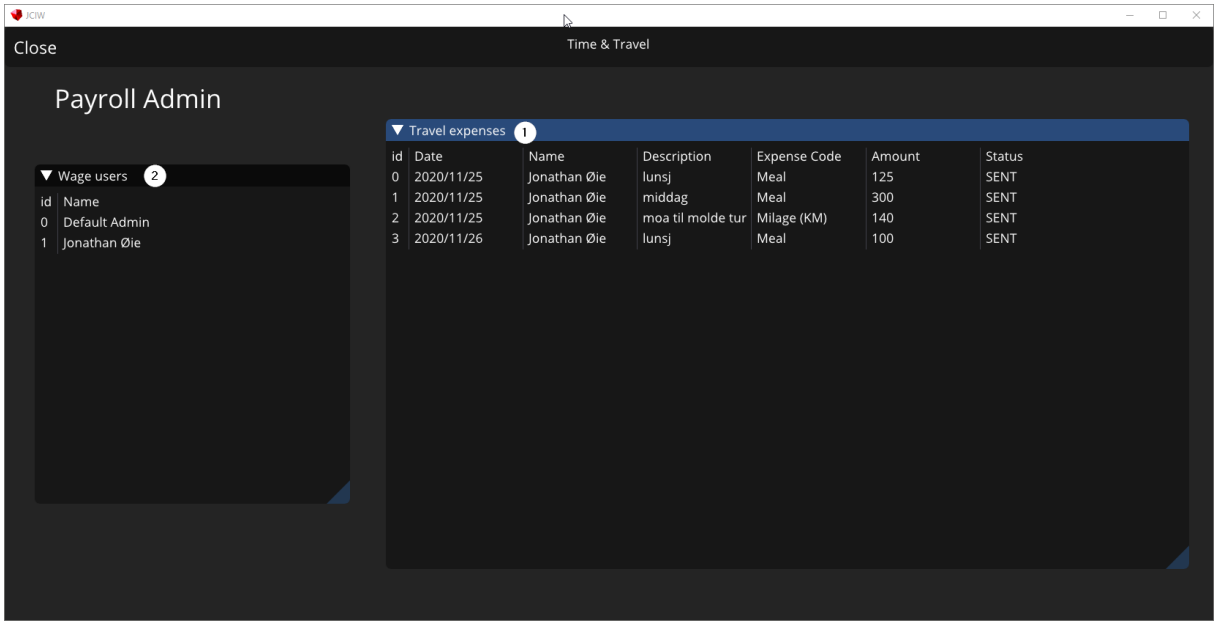


Figure 30: Payroll view on Time & Travel app

1. Wage users - brukerne sine timeføringer.
2. Travel expenses - brukerne sine reiseregninger.

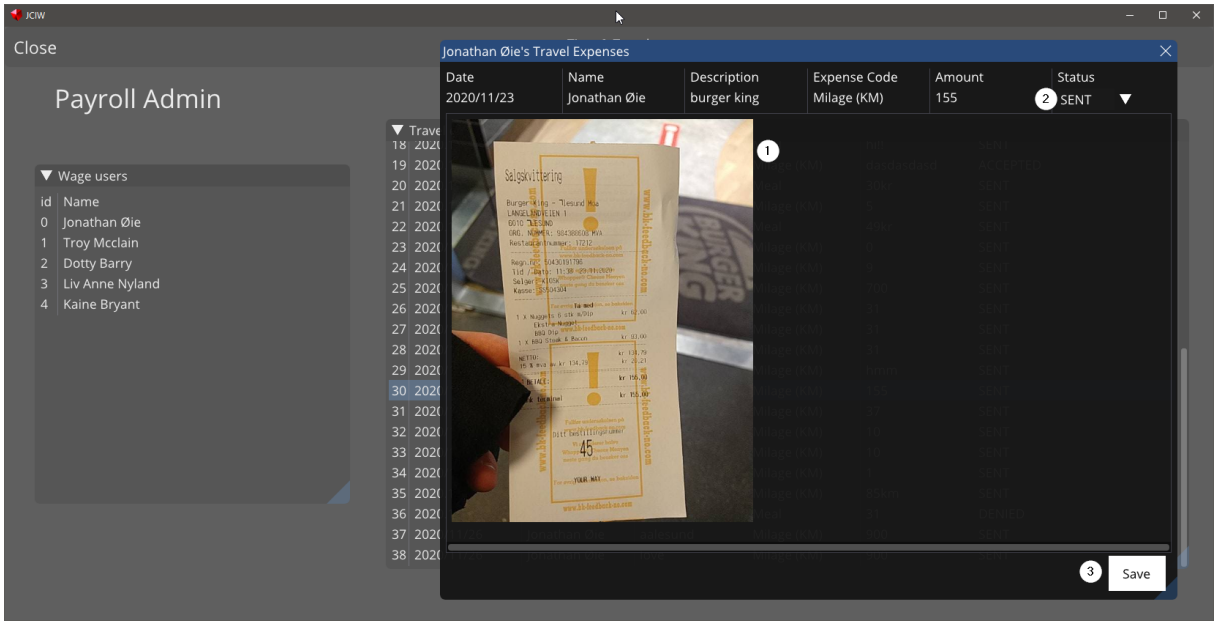


Figure 31: Exepense view on Payroll view on Time & Travel app

1. Liste over bilder som brukeren har lagt til i reiseregningen sin.
2. Status å vise til bilder. SENT / DENIED / ACCEPTED.
3. Save / send til server.

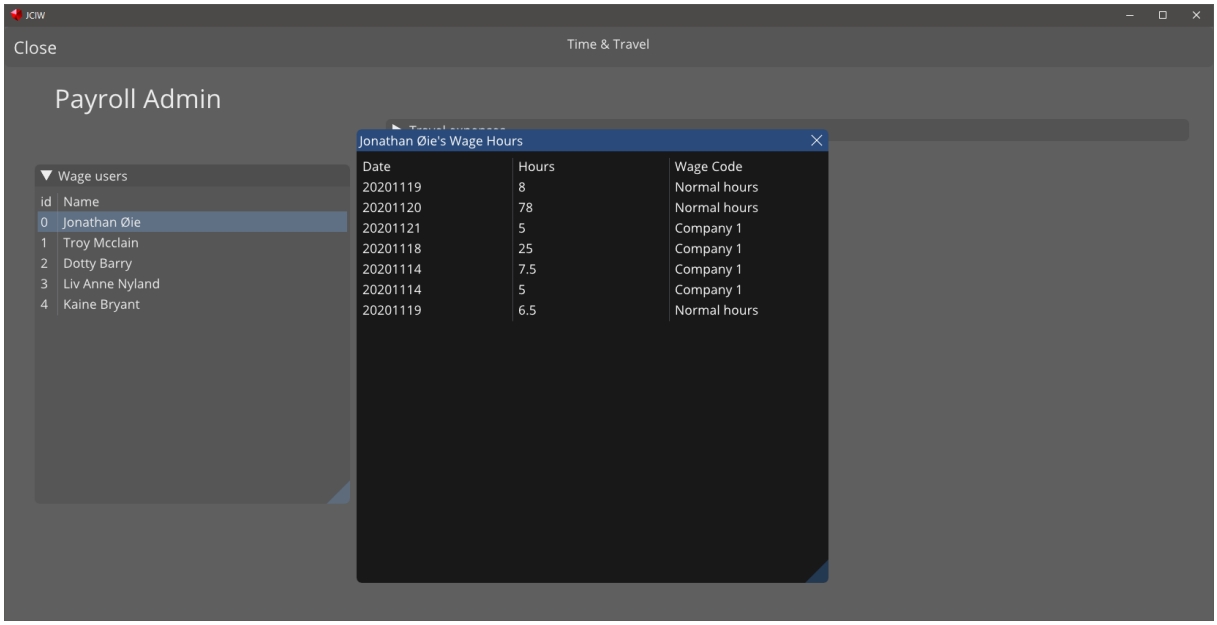


Figure 32: Wage hour view on Payroll view on Time & Travel app

4.4 Admin Klient

Følgende er klassene i admin klienten.

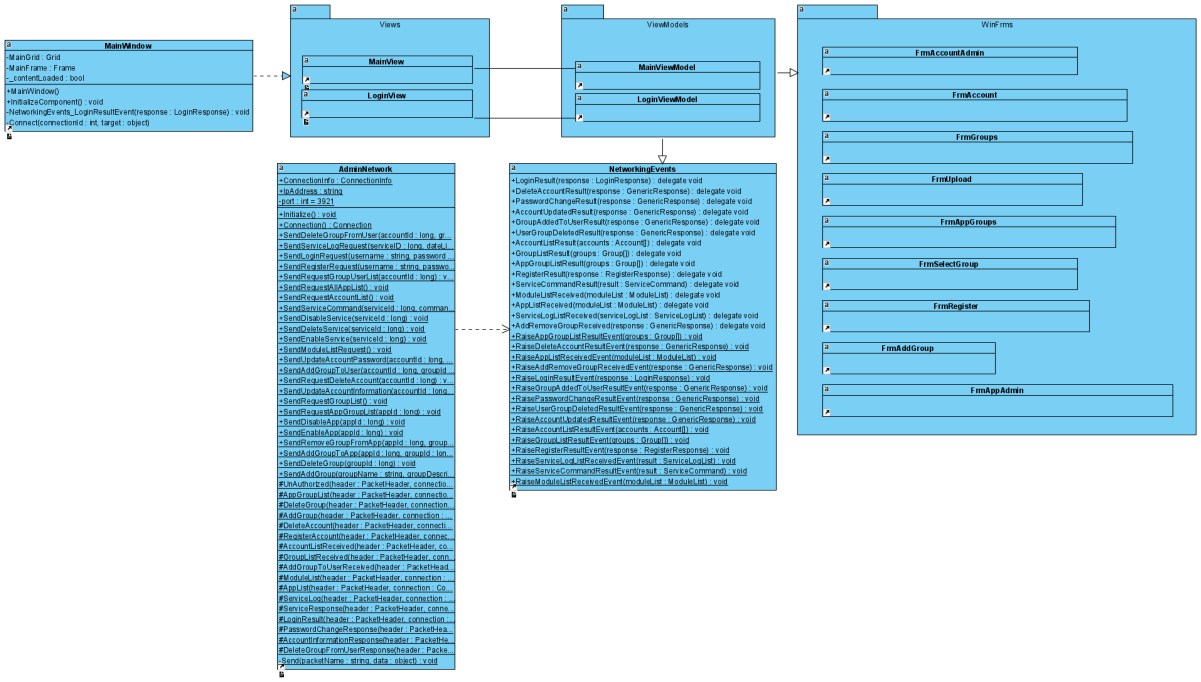


Figure 33: Admin Client classes

Admin klienten er bygget med Windows Presentation Foundation (WPF) med Model-View-ViewModel (MVVM) kodemønster. Views blir lastet i MainWindow og business logikken står i ViewModels.

AdminNetwork definerer NetworkComms pakkebehandlerne som igjen sender pakkene til NetworkingEvents. Disse blir abonnert på i View-Modellene.

De fleste av sidefunksjonene som å opprette nye brukere eller laste opp filer ble laget med Windows Forms.

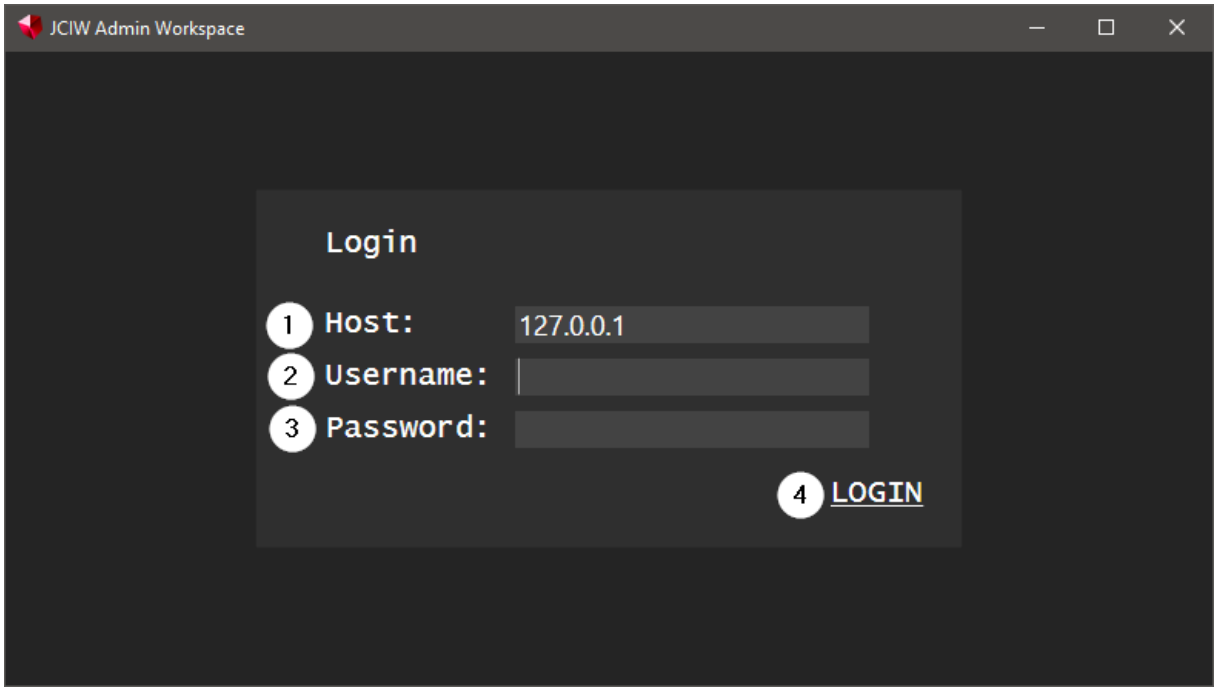


Figure 34: JCIW Admin Workspace Login

1. IP Addresse eller domenenavn til JCIW Server
2. Brukernavn-felt
3. Passord-felt
4. Loginknapp

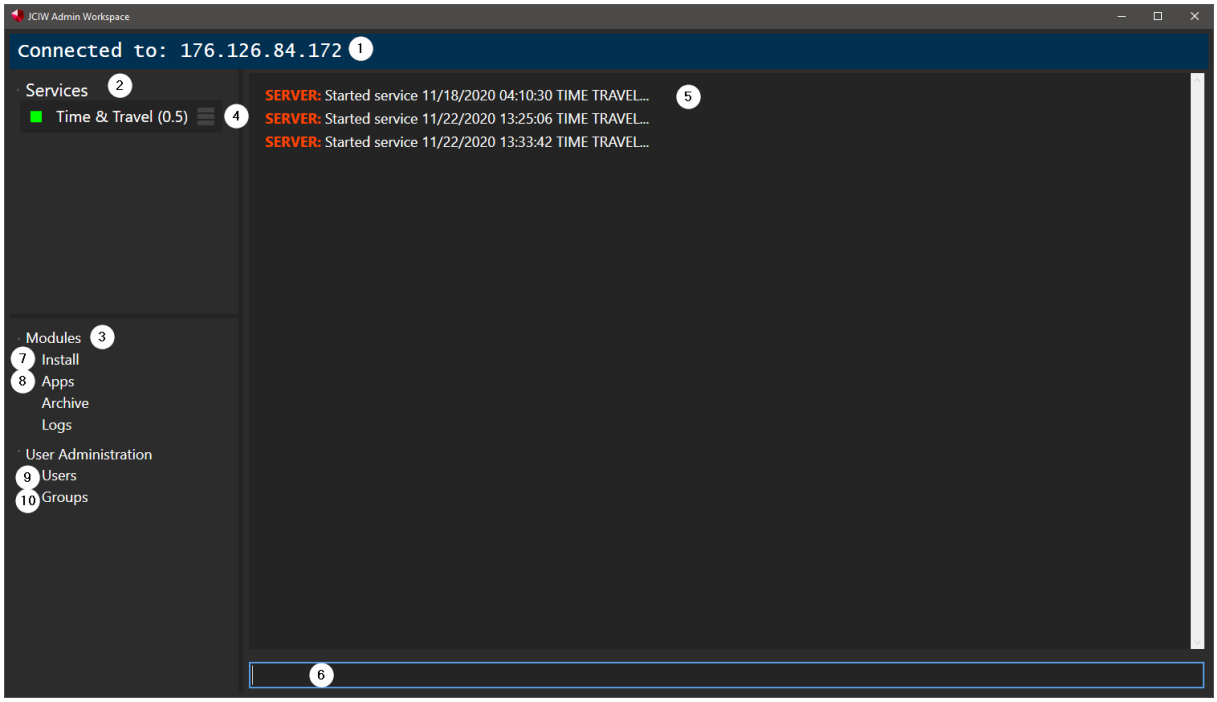


Figure 35: JCIW Admin Workspace

1. Overlay viser tilkoblet JCIW Server sin IP Adresse
2. Liste over serveren sine tilgjengelige tjenester
3. Liste over Modulfunksjoner
4. Drop-down meny for tjenester for å enable / disable og slette
5. Tjenestelogg
6. Kommando input tekstboks
7. Installere ny modul på server
8. Installerte applikasjoner
9. Brukeradministrasjon
10. Grupperadministrasjon

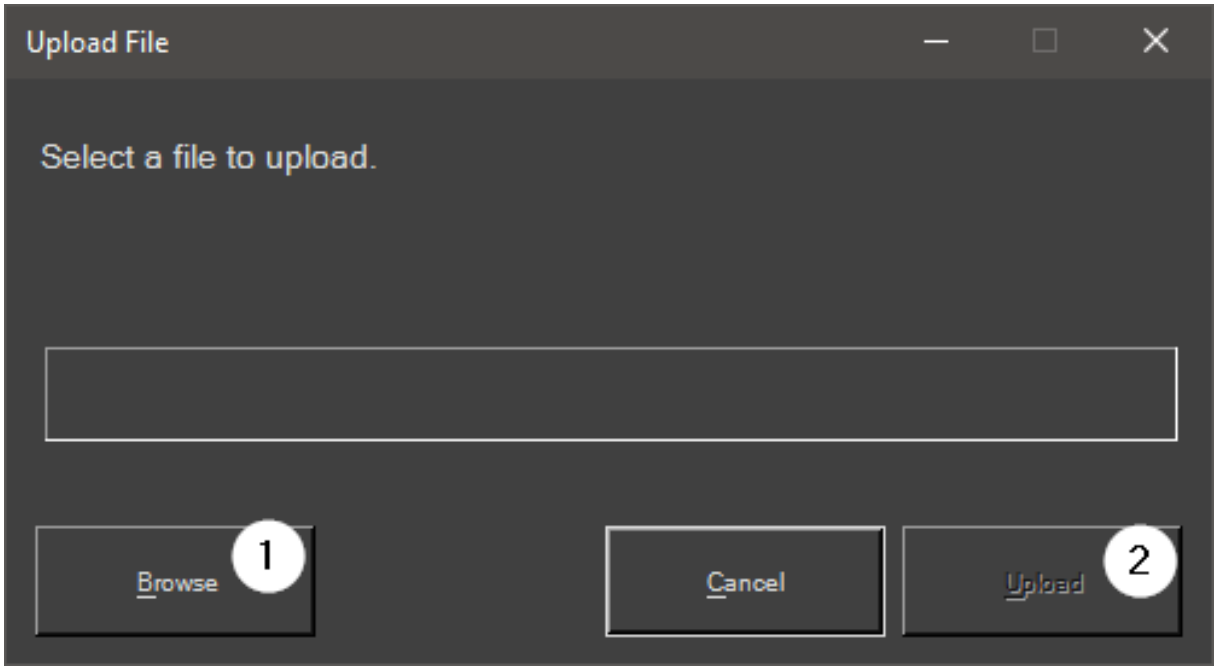


Figure 36: Upload File (Install)

1. Velg modul .dll file å laste opp.
2. Upload knapp. Blir enabled når valgt modulfil har blitt verifisert.

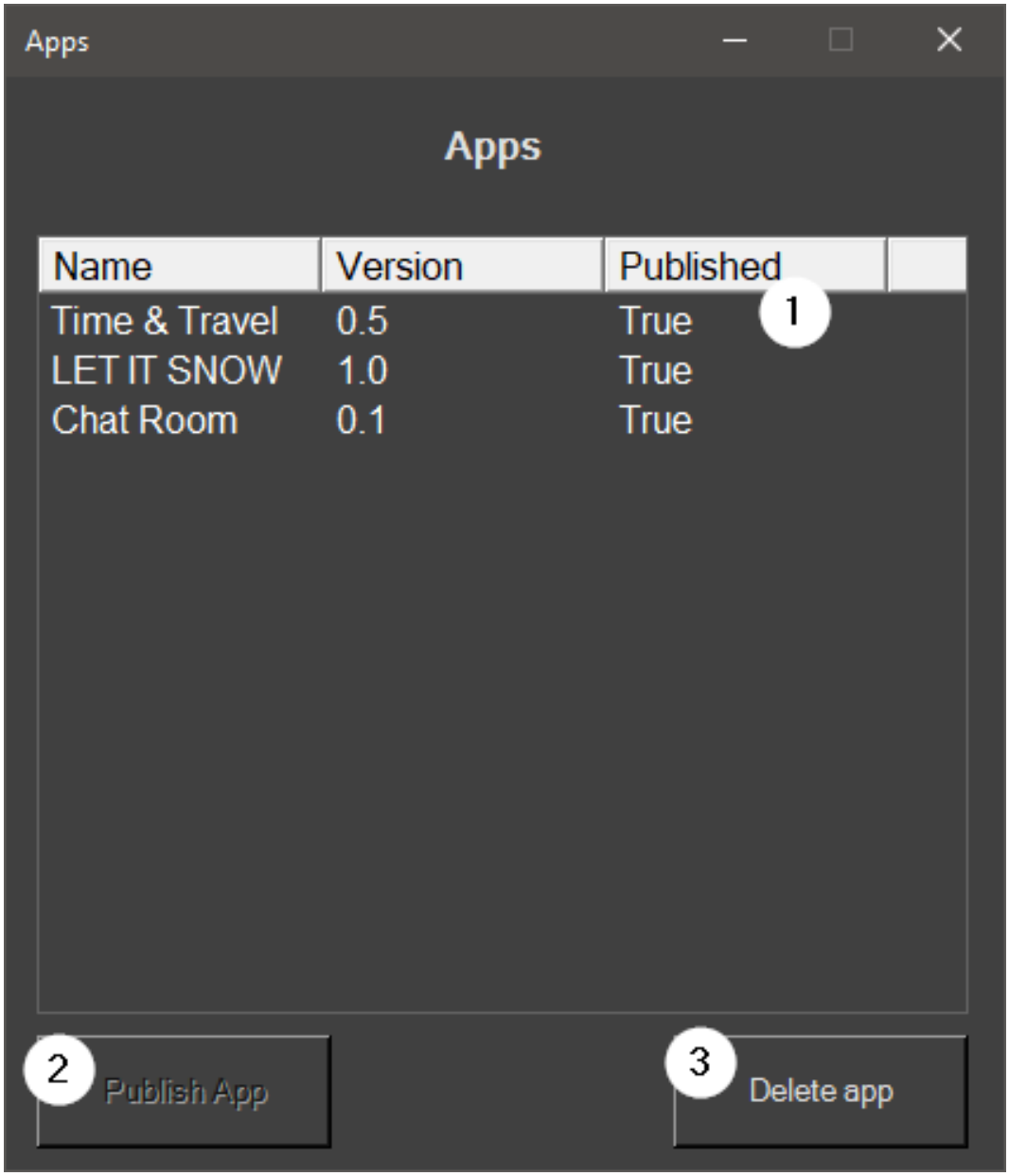


Figure 37: Apps

1. Applikasjonsliste. Dobbeltklikk på applikasjon åpner appen sine gruppetilganger.
2. Publiser applikasjon. Gjør apper synlige for brukere.
3. Slett app.

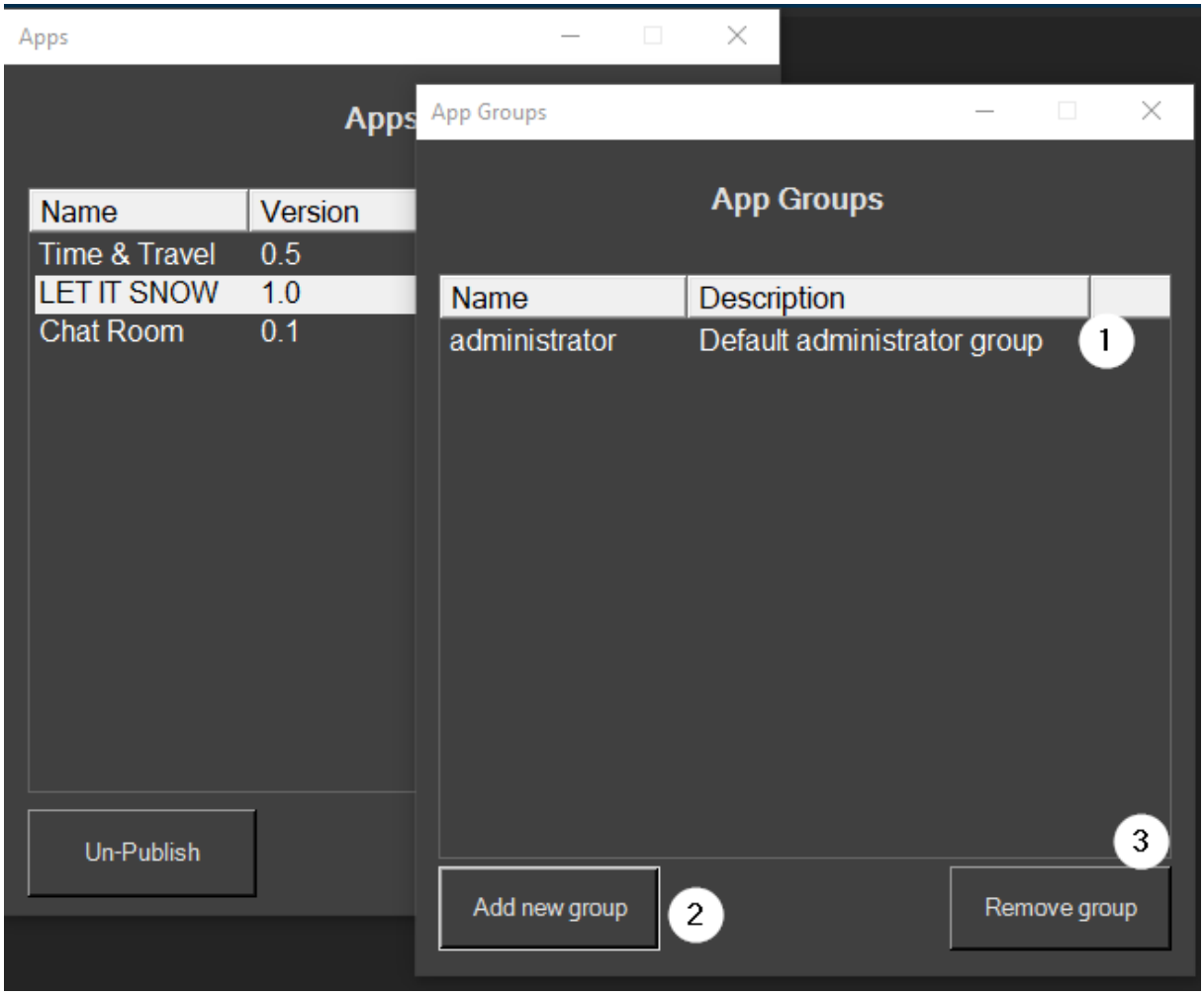


Figure 38: App Groups

1. Gruppeliste
2. Legg til grupperettighet til applikasjon.
3. Fjern grupperettighet til applikasjon.

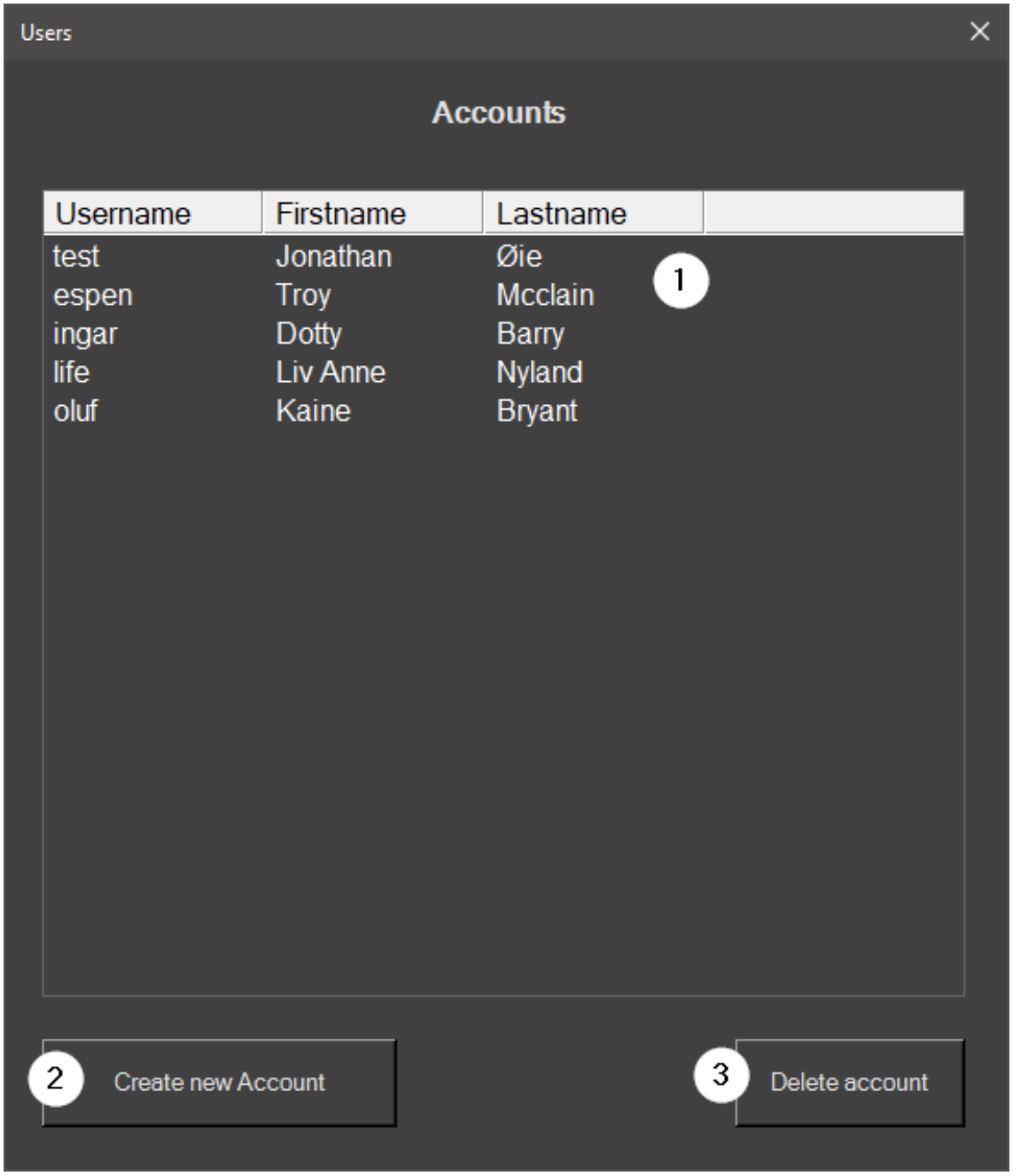


Figure 39: Users

1. Brukerliste. Åpner brukeradministrering ved dobbeltklikk.
2. Opprett ny bruker.
3. Slett bruker.

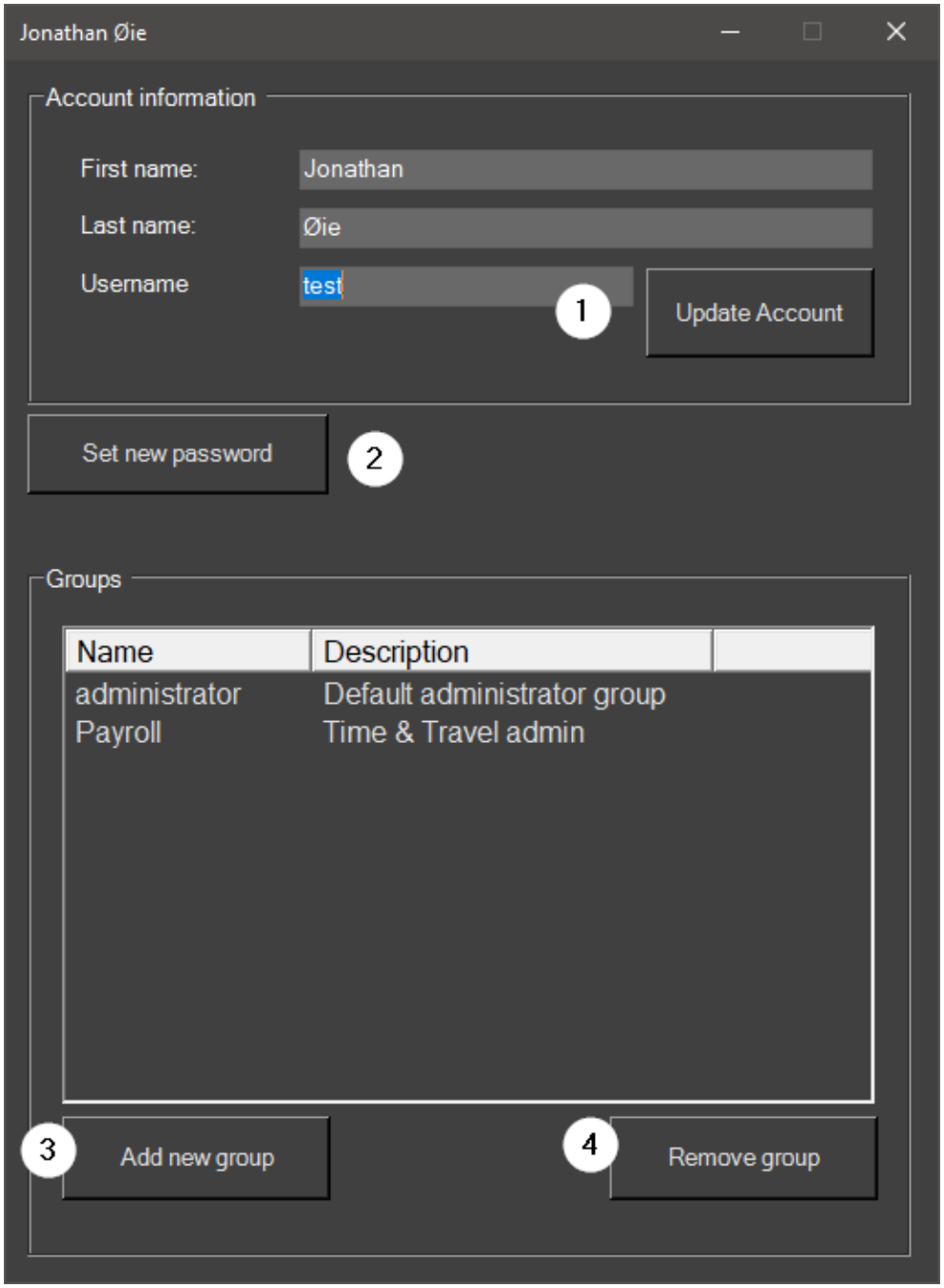


Figure 40: Specific user info

1. Oppdater brukerinformasjon.
2. Sett nytt passord på brukeren.
3. Legg til gruppemedlemskap på brukeren.
4. Fjern gruppemedlemskap fra brukeren.

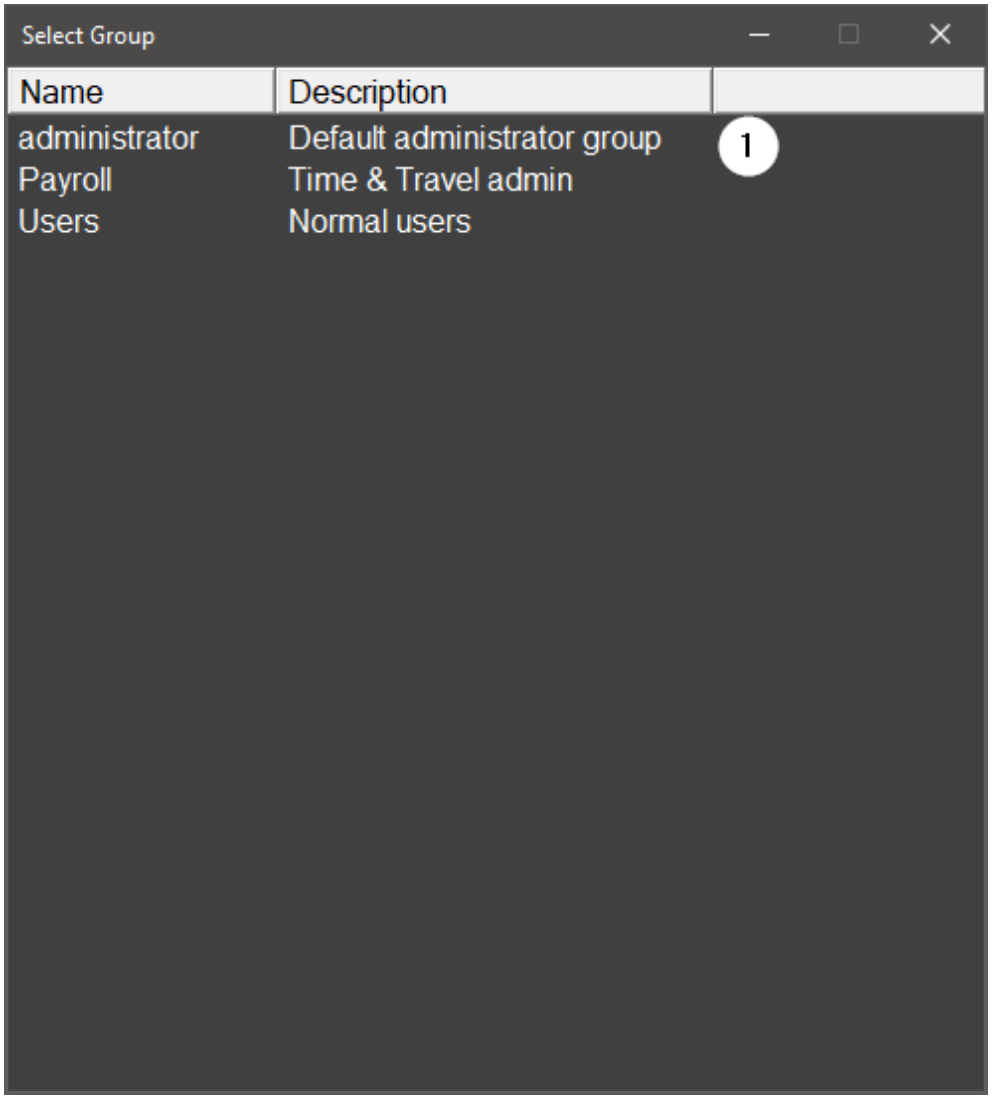


Figure 41: Add group membership to user

1. Gruppeliste. Dobbeltklikk for å velge gruppen.

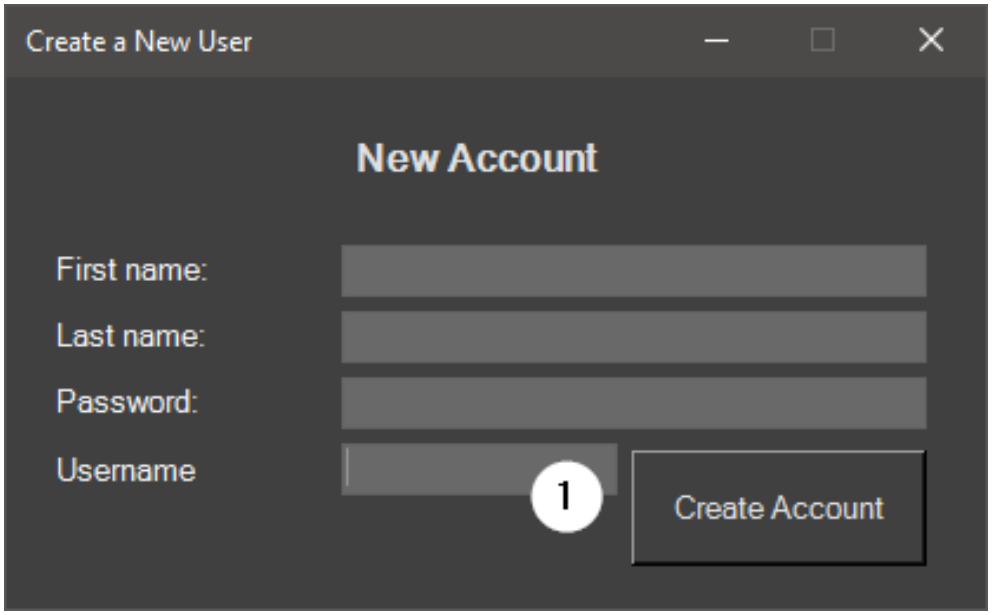


Figure 42: Create new User

1. Opprett ny bruker.

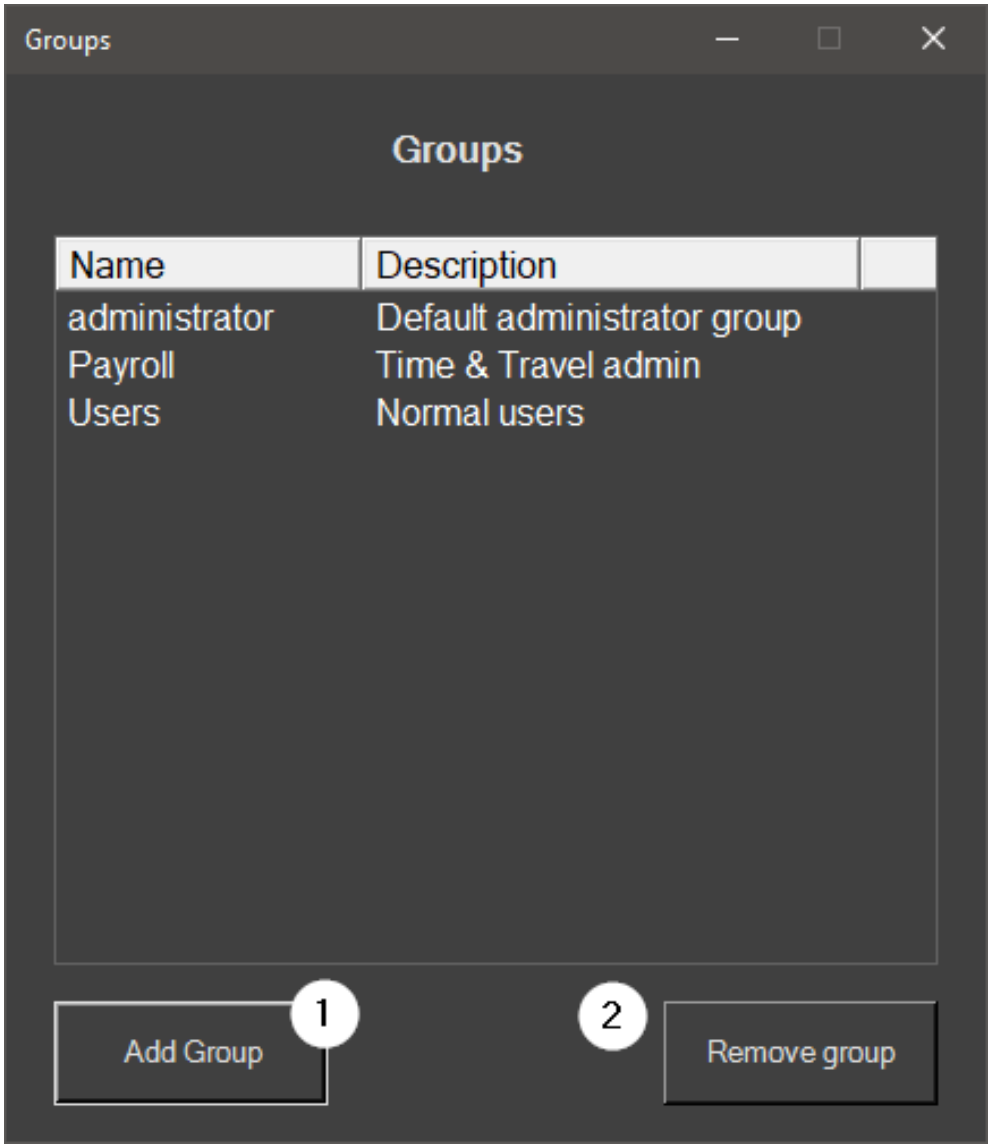


Figure 43: Groups

1. Legg til ny gruppe på server.
2. Fjern gruppe fra server.

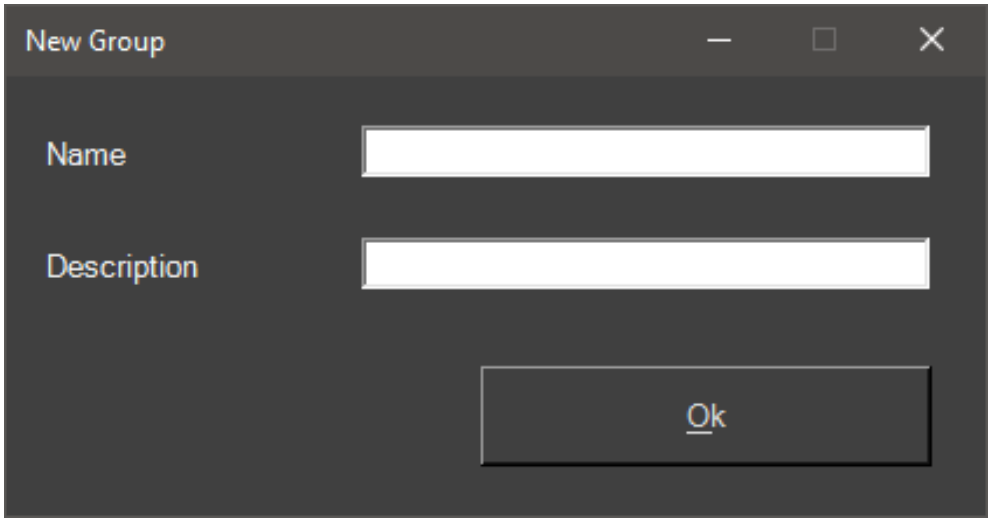


Figure 44: Create new group

4.5 Android Implementasjon

Den Android spesifikke koden ligger i prosjektene Client-Mobile.Android og Client-Networking.Android.

4.5.1 Client-Mobile.Android

Figure 45: Android JCIW Klient Ikon

Følgende Android rettigheter blitt brukt på klienten:

```
android.permission.CAMERA
android.permission.WRITE_EXTERNAL_STORAGE
android.permission.READ_EXTERNAL_STORAGE
android.permission.INTERNET
```

Eksterne Biblioteker
Som nevnt i Materialer og Metoder så har klienten noen eksterne avhengigheter som må kopieres til disken for at modullastingen skal fungere.

Dette ble løst ved å legge filene til i prosjektet som Embedded Resource som igjen kan leses til en filstrøm og lagres på disken.

Disse filene lagres manuelt til

```
/storage/emulated/0/Android/data/NAVNET PÅ DATAPAKKEN/files/.__override__
```

Dette skjer èn gang på oppstart og krever deretter en restart av applikasjonen.

Platformfunksjoner

Android klienten implementerer visse android spesifikke funksjoner. To av følgende er eksempel på dette og er implementeringer av interfacet IPlatformFunctions.

Tastatur
Løsningen bruker Android sin InputMethodManager Soft Input for å plukke opp tastetrykk.

Dette er implementert i klassen AndroidFunctions.cs. Brukergrensesnittet abonnerer på KeyPressEvent i denne klassen som sender char.

Åpne kamera

Kameraaktiviteten er kopiert fra Microsoft sitt camera2 eksempel:
https://docs.microsoft.com/en-us/samples/xamarin/monodroid-samples/android50-camera2basic/

Kamera er også kalt fra AndroidFunctions og resultat er sendt som et JCIW.Data.Drawing.ImageSource objekt til KeyPressEvent.

Kameraaktiviteten heter CameraActivity og åpnes via et StartActivityForResult kall. Den returnerer byte array som plassers i ImageSource objektet.

Brukeren må da laste dette som et bilde eller sende til server selv.

4.5.2 Client-Networking.Android

Client-Networking.Android prosjektet implementerer kun IAppNetworking. Den er identisk med desktopimplementasjonen med et unntak at NetworkComms sin Establish-Connection metode kræsjet og fryste applikasjonen i situasjoner hvor server ikke var tilgjengelig. Det måtte derfor lages en egen tilkoblingssjekk i implementasjonen med en vanlig TcpClient. Bortsett fra dette er det ikke noe unikt med Android sin implementasjon av IAppNetworking.

4.6 Desktop Implementasjon

Den desktop spesifikke koden ligger i prosjektene Client-Desktop og Client-Networking.Desktop.

4.6.1 Client-Desktop

Eksterne Biblioteker
De eksterne bibliotekene som brukes er bare cimgui på desktop klienten. Cimgui biblioteket for Linux, Windows og Mac ligger med build action Content i prosjektet. Dette gjør at bibliotekene havner i bin mappen til klientprogrammet.

Platformfunksjoner
Desktop klienten implementerer visse desktop spesifikke funksjoner. To av følgende er eksempel på dette og er implementeringer av interfacet IPlatformFunctions.

Tastatur
Desktopklienten henter tastaturtrykk fra spillvinduet. Den abonnerer på vinduet sin TextInput event så bruker den DesktopFunctions sin KeyPressEvent til å sende det videre til brukergrensesnittet.

Browse
Desktopklienten støtter dialogboks for å laste opp bilder og filer. Denne bruker Windows Forms sin SaveFileDialog og OpenFileDialog til å åpne dialogbokser for åpning og lagring av filer.

4.6.2 Client-Networking.Desktop

Client-Networking.Desktop implementerer IAppNetworking. Funksjonen til klassen er å håndtere innkommende pakker og sende utgående pakker til server.

Følgende pakker er implementert:

Pakkenavn	Beskrivelse
Custom	Egendefinerte pakker fra tjenester
ReSessionVerificationResult	Verifikasjonsresultat fra session request
ReSessionId	Session ID sendt fra server etter innlogging
ReLoginResult	Resultat på innlogging
ReModuleList	Brukeren sin modulliste
ReAppFile	Applikasjonsfil fra server
ReUserGroupList	Brukeren sine grupper
ReUnauthorized	Melding fra server om at brukeren ikke er autorisert

JCIW mottar disse pakkene ved å abonnere på følgende events i klassen:

```
public event EventHandler LoginEvent;
public event EventHandler ModuleListEvent;
public event EventHandler AppFileEvent;
public event EventHandler GroupList;
public event EventHandler SessionReceived;
public event EventHandler SessionVerificationResult;
public event EventHandler UnAuthorized;
public event EventHandler NetworkDisconnect;
```

”Custom” pakker håndteres av JCIW sin PacketManager klasse.

4.7 App Modul

Alle JCIW Apps bruker base klassen AppBase.

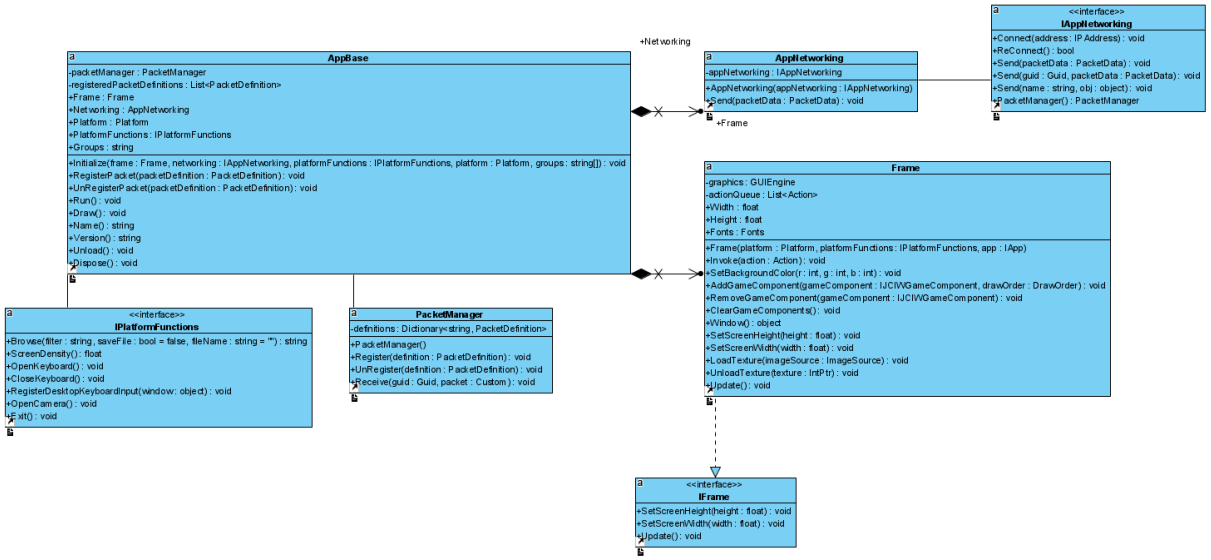


Figure 46: Klasser knyttet til AppBase

AppBase: Base class for JCIW Apps. Responsible for module data and networking and drawing.
AppNetworking: This class is responsible for exposing networking methods to app developers.
IAppNetworking: This interface is responsible for networking methods for AppBase.
IPlatformFunctions: This interface is responsbile for implementing platform specific functions for clients. Desktop / Android.
PacketManager: This class handles custom user defined packets.
Frame: This class handles graphics.
IFrame: This interface is used to implement some Frame methods.

Denne er kritisk for at modulsystemet skal laste og kjøre riktig. For å utvikle en ny app overrider en Name(), Version(), Dispose(), Run() og Draw()

```
// DO NOT CHANGE THE NAMESPACE
namespace Module
{
    // Only one app per module is supported.
    // The name must be "App" to load correctly.
    // DO NOT CHANGE THE CLASS NAME.
    class App : AppBase
    {
        // Run is called once when app is enabled.
        // Use it to initialize packets and set up database tables etc.
        public override void Run()
        {
            base.Run();

            // Write your service initilization code here.
        }

        // This method is used to draw the app user interface using ImGui.
        public override void Draw()
        {
            // Place your GUI code here.

            ImGui.ShowDemoWindow();
        }

        // This is the name that will be displayed in the client module list.
        public override string Name()
        {
            return "CHANGE-ME";
        }

        // This is the version that will be displayed in the admin client.
        public override string Version()
        {
            return "0";
        }

        // Dispose is called once when service is exited.
        // Dispose and stop any threads running in the background etc.
        public override void Dispose()
        {
            base.Dispose();
        }
    }
}
```

Figure 47: Template til apputvikling

4.8 Service Modul

Alle JCIW Services bruker base klassen ServiceBase.

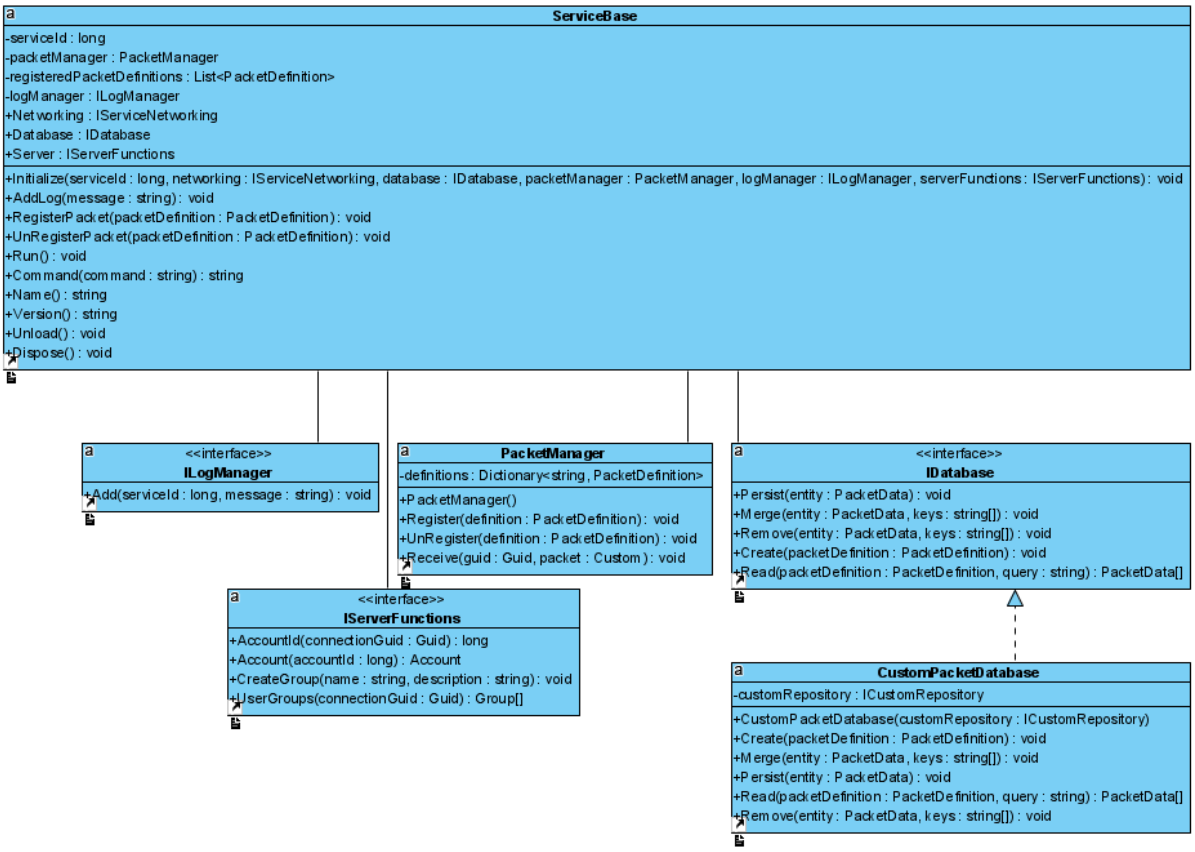


Figure 48: Klasser knyttet til ServiceBase

Denne er kritisk for at modulsystemet skal laste og kjøre riktig. For å utvikle en ny app overrider en Name(), Version(), Dispose(), Run() og Command().

ServiceBase: Base class for JCIW Service. Database og pakkehåndtering.
ILogManager: Interface for lesing og lagring av logger.
PacketManager: Klasse ansvarlig for brukerdefinerte pakker.
IDatabase: Interface ansvarlig for databaseoperasjoner.
IServerFunctions: Interface for å gi bruker tilgang til noen server funksjoner. Påloggede brukere og sånne ting.
CustompacketDatabase: Klasse som implementerer IDatabase

```
// DO NOT CHANGE THE NAMESPACE
namespace Module
{
    // Only one service per module is supported.
    // The name must be "Service" to load correctly.
    // DO NOT CHANGE THE CLASS NAME.
    public class Service : ServiceBase
    {
        // Run is called once when service is enabled.
        // Use it to initialize packets and set up database tables etc.
        public override void Run()
        {
            base.Run();

            // Write your service initilization code here.
            AddLog("Started service " + Name() + " " + DateTime.Now.ToString());


        }

        // This is the name that will be displayed in the admin client.
        public override string Name()
        {
            return "CHANGE-ME";
        }

        // This is the version that will be displayed in the admin client.
        public override string Version()
        {
            return "0";
        }

        // Dispose is called once when service is disabled.
        // Dispose and stop any threads running in the background
        public override void Dispose()
        {
            base.Dispose();
        }

        // This method receives the input from the admin client.
        // Use it to create commands for administrators to run.
        public override string Command(string command)
        {
            switch (command.ToLower())
            {
                case "help":
                    return "help not implemented.";
                default:
                    return "no commands implemented.";
            }
        }
    }
}

```

Figure 49: Template til apputvikling

4.9 Modulsystem

Modulsystemet er bygget med System.Reflection og AppDomain.

AppDomain gjør at systemet kan åpne og lese en biblioteksfil til et annet domene sitt minne. Modulen har Version og Name string-metoder i hovedklassen sin som leses av AppDomenet. Dersom det fungerer så blir det returnert via et proxy objekt ellers så får mann null. Hovedpoenget er at minnet og kjøringen kjøres i et eget domene sånn at det ikke påvirker klient eller server.

Følgende er klassene for lesing og lasting av modulklasser (JCIW.Module):

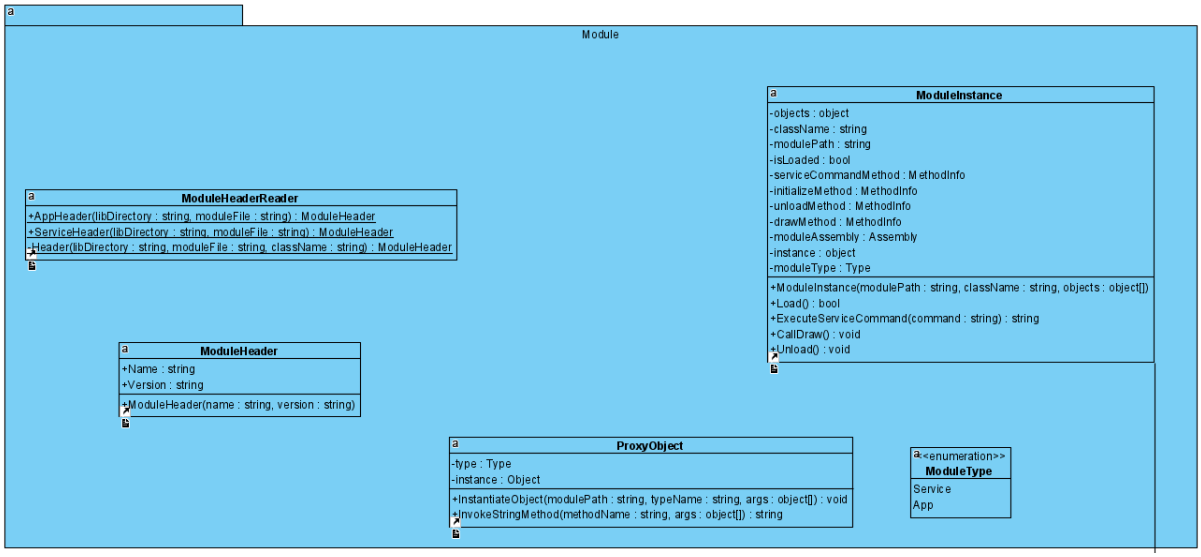


Figure 50: JCIW Module klassene

ModuleHeaderReader bruker AppDomain til å verifisere modulfilen.
ModuleInstance er klassen en oppretter for å laste og bruke en modulfil. Den har allerede metoder for kalling av initialize, draw og unload.

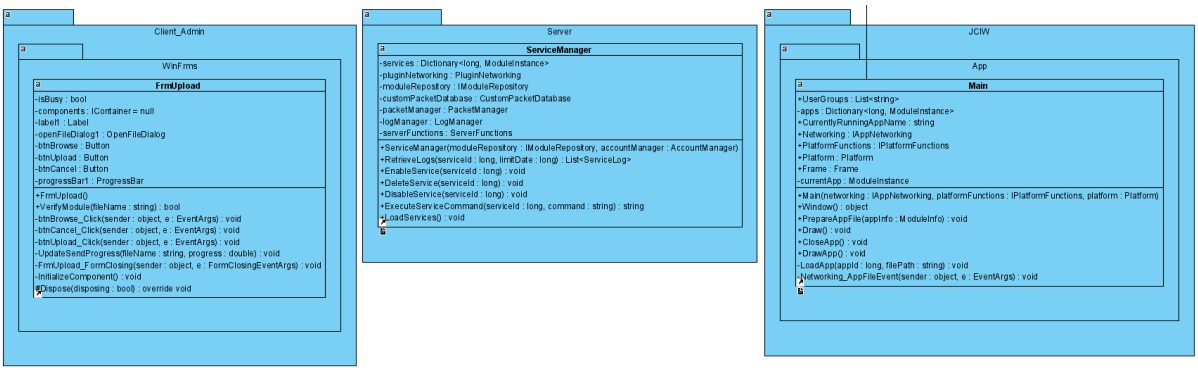


Figure 51: Klasser som bruker JCIW.Module klassene

Diagrammet viser klassene som bruker JCIW.Module til å enten laste eller verifisere moduler.

Client.Admin.WinFrms.FrmUpload.cs klassen bruker ModuleHeaderReader til å verifisere at modulen brukeren forsøker å laste opp er en gyldig JCIW Modul.
Server.ServiceManager bruker ModuleInstance til å laste og kjøre tjenestemoduler.
JCIW.App.Main bruker ModuleInstance til å laste og kjøre appmoduler.

4.9.1 Pakkesystem

Modulsystemet har et eget pakkesystem som lar brukeren definere og sende og motta pakker mellom app og service.

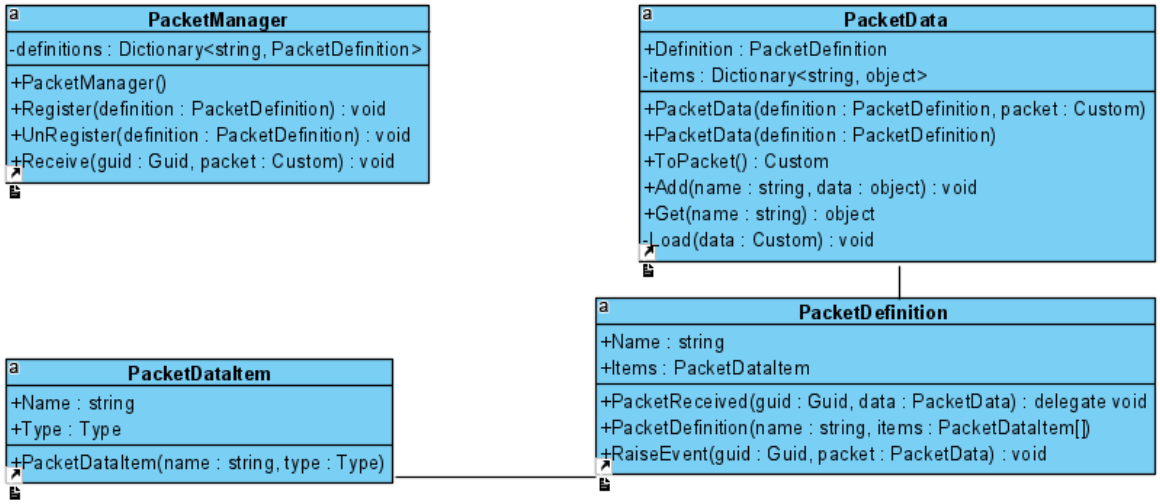


Figure 52: Klassene i pakkesystemet

En utvikler definerer navn og feltene i pakken (PacketDefinition), registrerer den i PacketManager og lytter på innkommende pakker ved å abonnere på PacketReceived. En får da PacketData tilbake.

PacketData kan også brukes i modulsystemet sin datalagring til å opprette og lagre tabeller i databasen.

```
// Packet defintion
public static PacketDefinition TestPacket = new PacketDefinition("PacketName", new PacketDataItem[]
{
    new PacketDataItem("id", typeof(double)),
    new PacketDataItem("name", typeof(string))
});

public override void Run()
{
    base.Run();

    // Register packet so packets will be received to object.
    RegisterPacket(TestPacket);

    TestPacket.PacketReceivedEvent += TestPacket_PacketReceivedEvent;
}

private void TestPacket_PacketReceivedEvent(Guid guid, PacketData data)
{
    // TestPacket received.
    double id = (double)data.Get("id");
    string name = data.Get("name").ToString();
}
```

Figure 53: Eksempel hvordan registrere og motta pakker

```
// Packet defintion
public static PacketDefinition TestPacket = new PacketDefinition("PacketName", new PacketDataItem[]
{
    new PacketDataItem("id", typeof(double)),
    new PacketDataItem("name", typeof(string))
});

public override void Run()
{
    base.Run();

    // Create packet data.
    PacketData sendTestPacket = new PacketData(TestPacket);
    sendTestPacket.Add("id", 0);
    sendTestPacket.Add("name", "test");

    // Send packet to server.
    Networking.Send(sendTestPacket);

}
```

Figure 54: Eksempel hvordan sende pakkedata

4.9.2 Datalagring

Modultjenester har mulighet til å skrive og lese fra databasen med pakkedata.

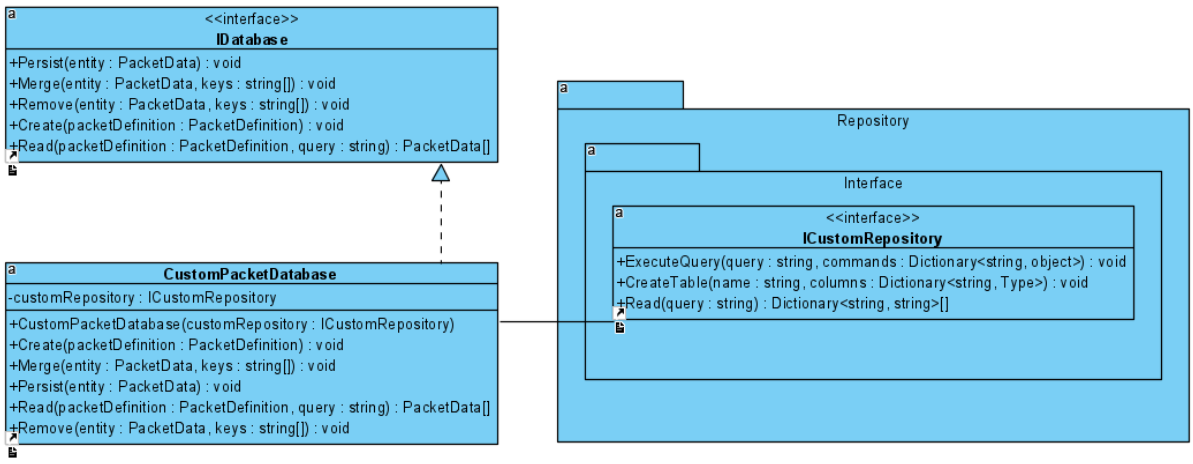


Figure 55: Implementeringen av IDatabase

IDatabase har metoder for Persist, Merge, Remove, Create og Read.

```
/// <summary>
/// This interface is used to expose database functionality to the service developer.
/// </summary>
public interface IDatabase
{
    /// <summary>
    /// Add row in table.
    /// </summary>
    /// <param name="entity">The row to add.</param>
    void Persist(PacketData entity);

    /// <summary>
    /// Update existing rows in table.
    /// </summary>
    /// <param name="entity">The row to update.</param>
    /// <param name="keys">The row columns to update as conditional statements.</param>
    void Merge(PacketData entity, string[] keys);

    /// <summary>
    /// Delete rows from table.
    /// </summary>
    /// <param name="entity">The row to remove.</param>
    /// <param name="keys">The row columns to add as conditional statements.</param>
    void Remove(PacketData entity, string[] keys);

    /// <summary>
    /// Create a new table.
    /// </summary>
    /// <param name="packetDefinition">The table definition.</param>
    void Create(PacketDefinition packetDefinition);

    /// <summary>
    /// Read from table.
    /// </summary>
    /// <param name="packetDefinition">Table columns.</param>
    /// <param name="query">SQL Query to run.</param>
    /// <returns>PacketData array.</returns>
    PacketData[] Read(PacketDefinition packetDefinition, string query);
}
```

Figure 56: Beskrivelse av IDatabase

4.10 Installatør

JCIW er distribuert via Github med full kildekode og kompileringsinstruker i tillegg til fire installasjonsfiler.

Android klienten har sin egen installatør med APK formatet.

De tre andre er laget med Inno Setup.

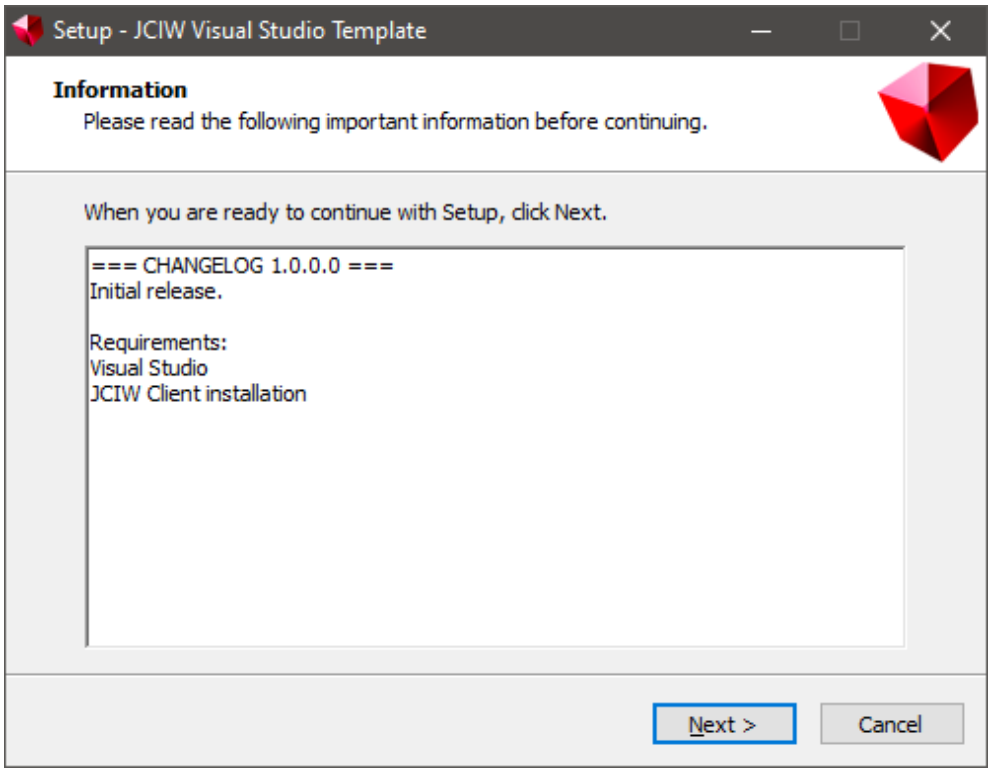


Figure 57: JCIW Visual Studio Template Installatør

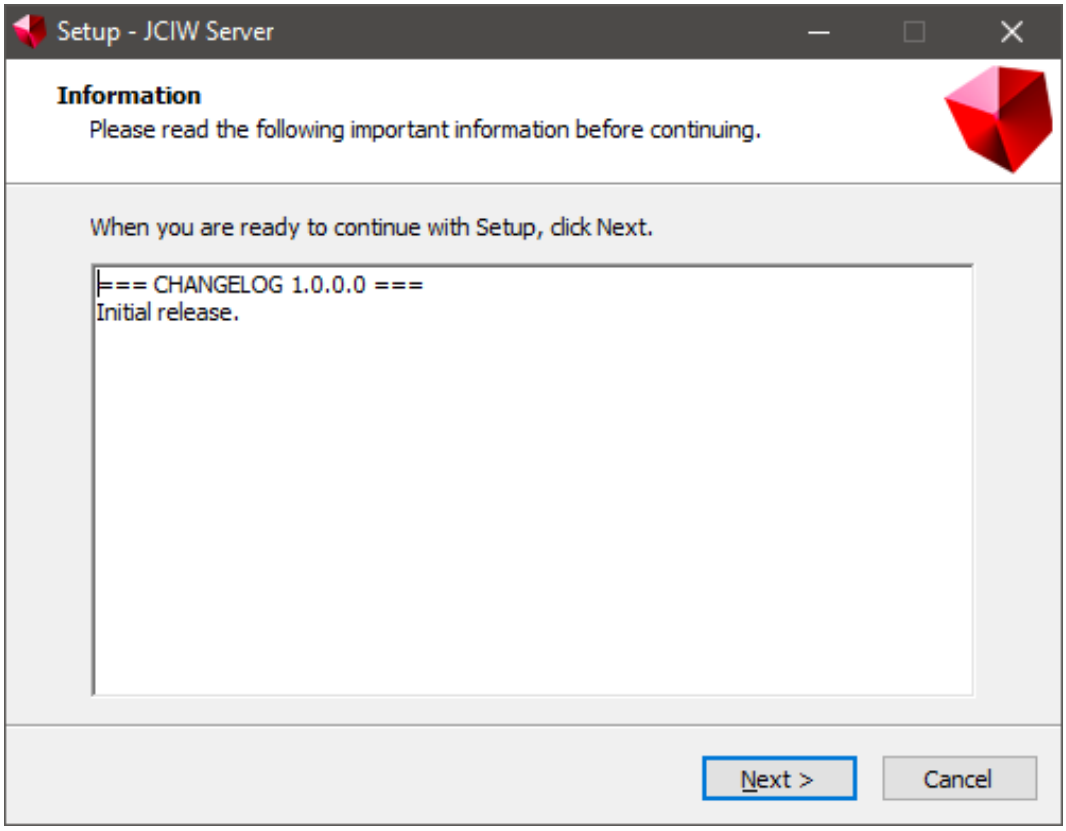


Figure 58: JCIW Server installatør

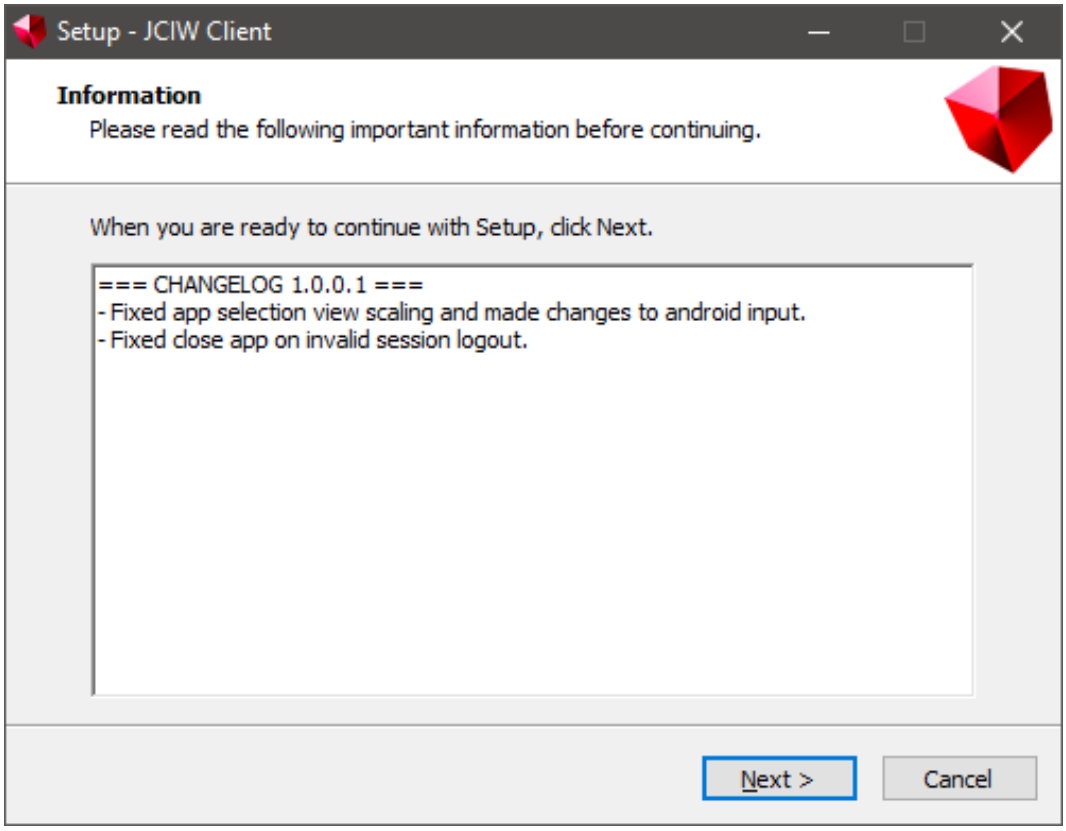


Figure 59: JCIW Client installatør

Server og klient-installatørene installerer JCIW programvaren til katalogen brukeren velger selv.

Template-installatøren må pekes på Visual Studio dokumentmappen.

JCIW VS malen krever at klienten er installert for at bibliotekreferansene skal fungere.

Dette kommer av at installsjonsscriptet til klienten inkluderer en registernøkkel. Katalogen til klientinstallasjonen legges til User Variables i Windows sine environment variables.

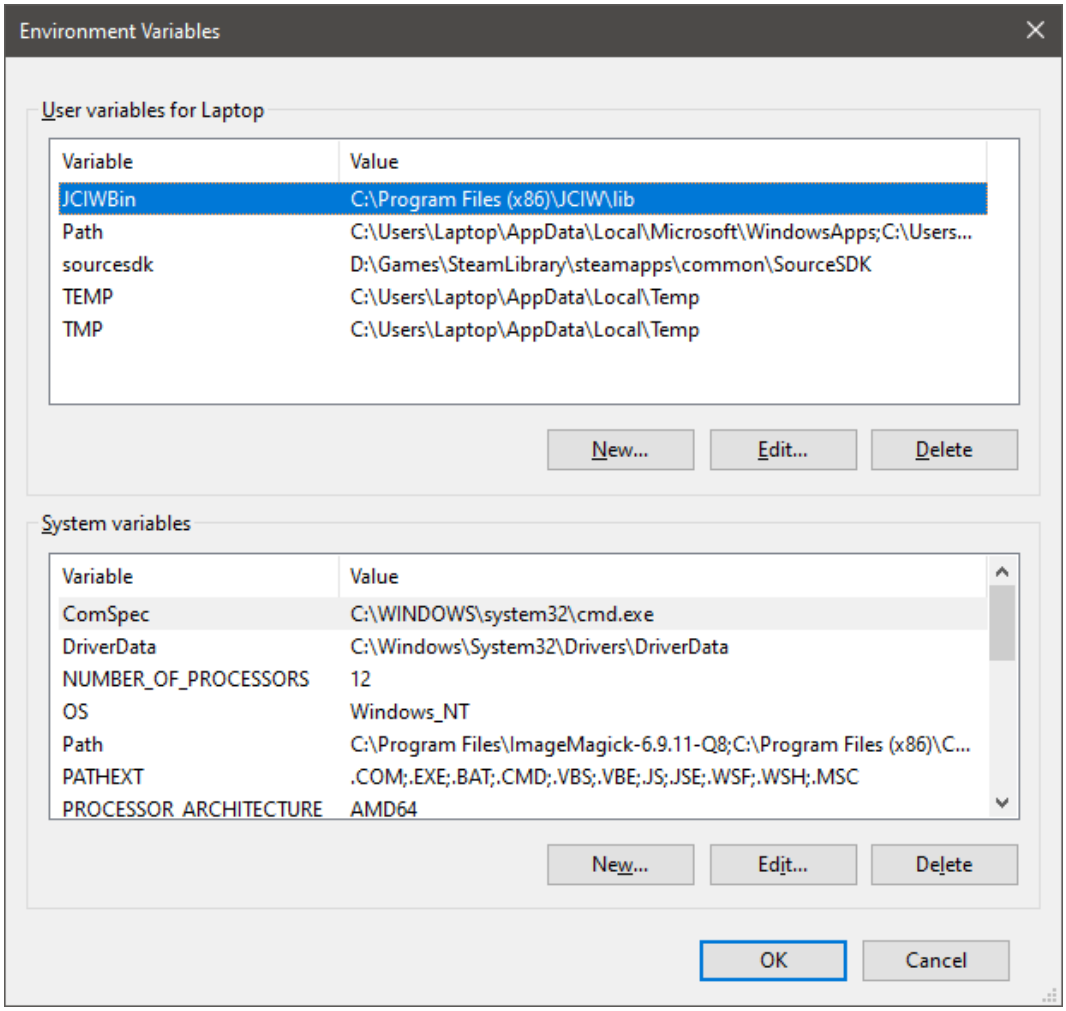
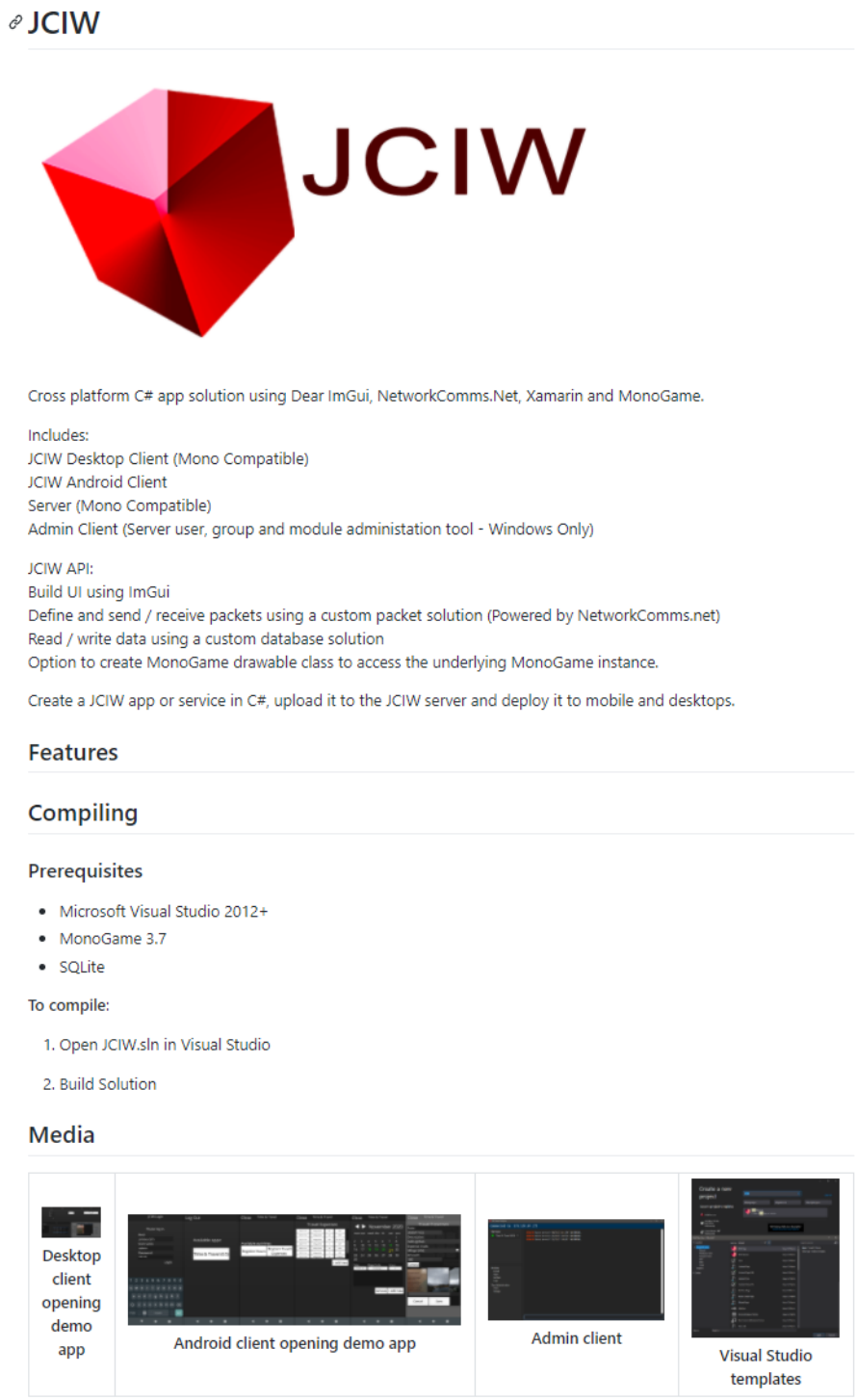


Figure 60: Environment variables

JCIWBin brukes i Visual Studio malen.

4.11 Github

Prosjektet er lagt ut med kildekode og installatør på https://github.com/j0nat/JCIW.



JCIW

Cross platform C# app solution using Dear ImGui, NetworkComms.Net, Xamarin and MonoGame.

Includes:
JCIW Desktop Client (Mono Compatible)
JCIW Android Client
Server (Mono Compatible)
Admin Client (Server user, group and module administation tool - Windows Only)

JCIW API:
Build UI using ImGui
Define and send / receive packets using a custom packet solution (Powered by NetworkComms.net)
Read / write data using a custom database solution
Option to create MonoGame drawable class to access the underlying MonoGame instance.

Create a JCIW app or service in C#, upload it to the JCIW server and deploy it to mobile and desktops.

Features

Compiling

Prerequisites

- Microsoft Visual Studio 2012+
- MonoGame 3.7
- SQLite

To compile:

1. Open JCIW.sln in Visual Studio
2. Build Solution

Media

Figure 61: Fremsiden av JCIW sin Github prosjektside

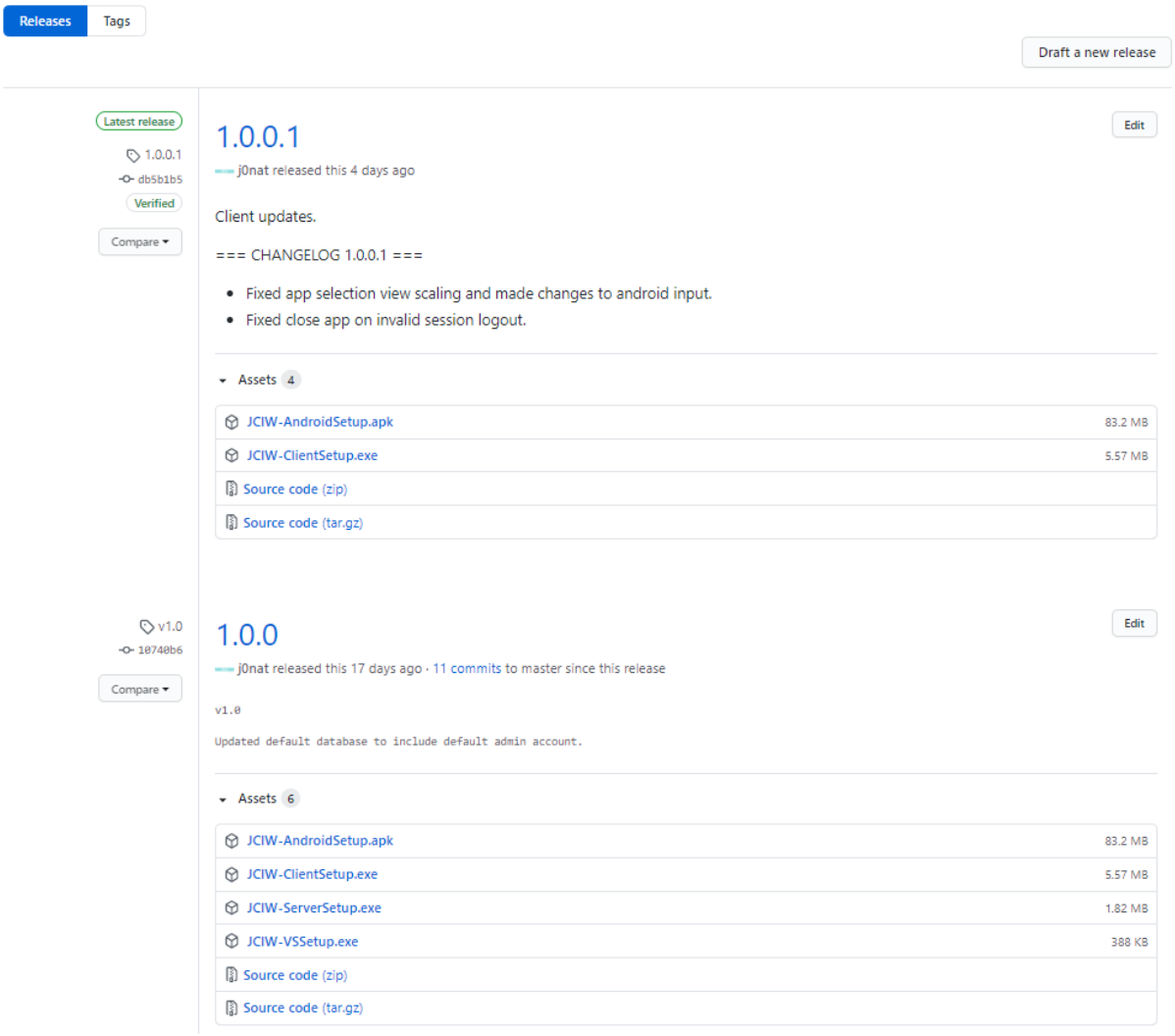


Figure 62: Release siden til JCIW sin Github prosjektside

4.12 Visual Studio Template mal

Visual Studio malene består av to maltyper. En project template og to item template. Disse blir generert med Visual Studio sitt ”Export Template Wizard” verktøy.

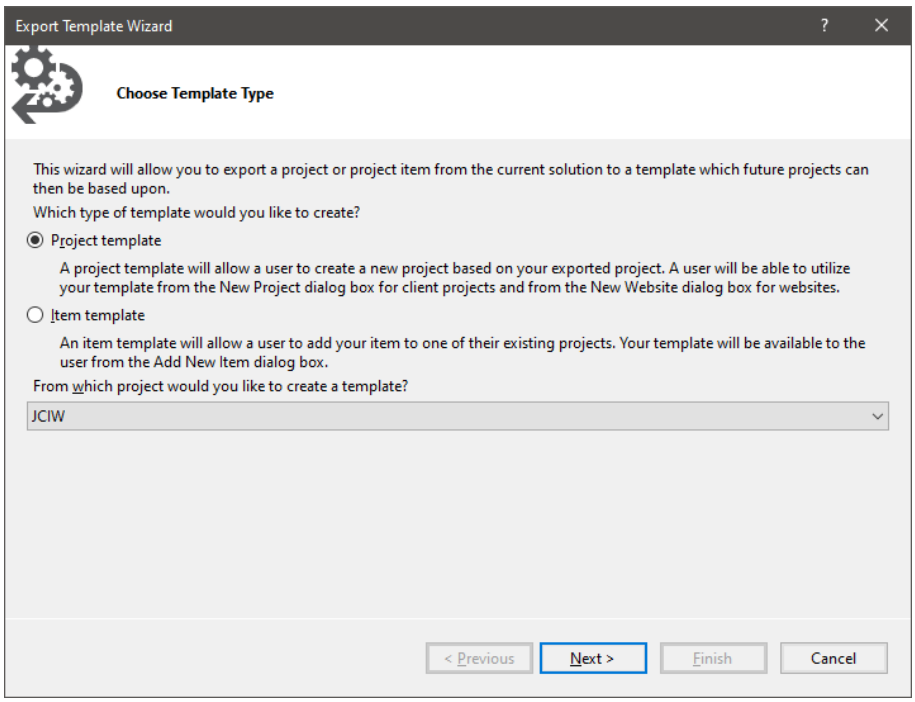


Figure 63: Visual Studio Export Template Wizard

Resultatet ser sånn ut i Visual Studio:

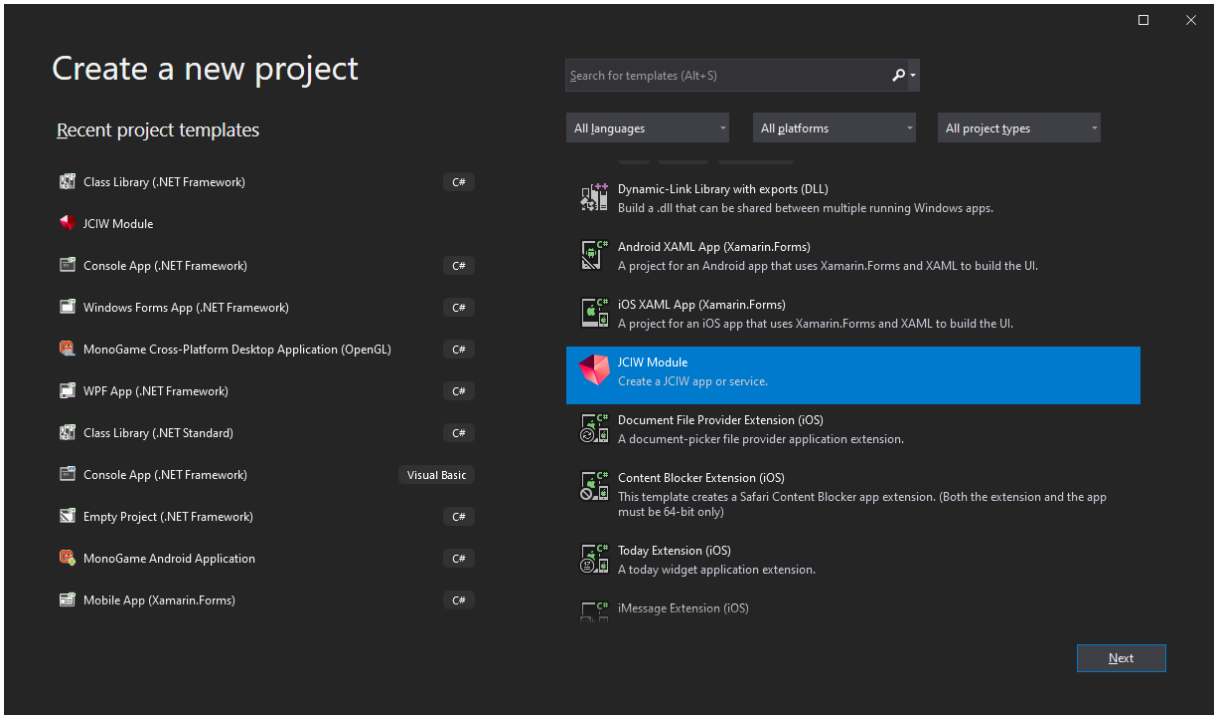


Figure 64: Startbildet til Visual Studio. Viser JCIW Template.

Prosjektmalen sin rolle er å sette opp alle JCIW referansene.

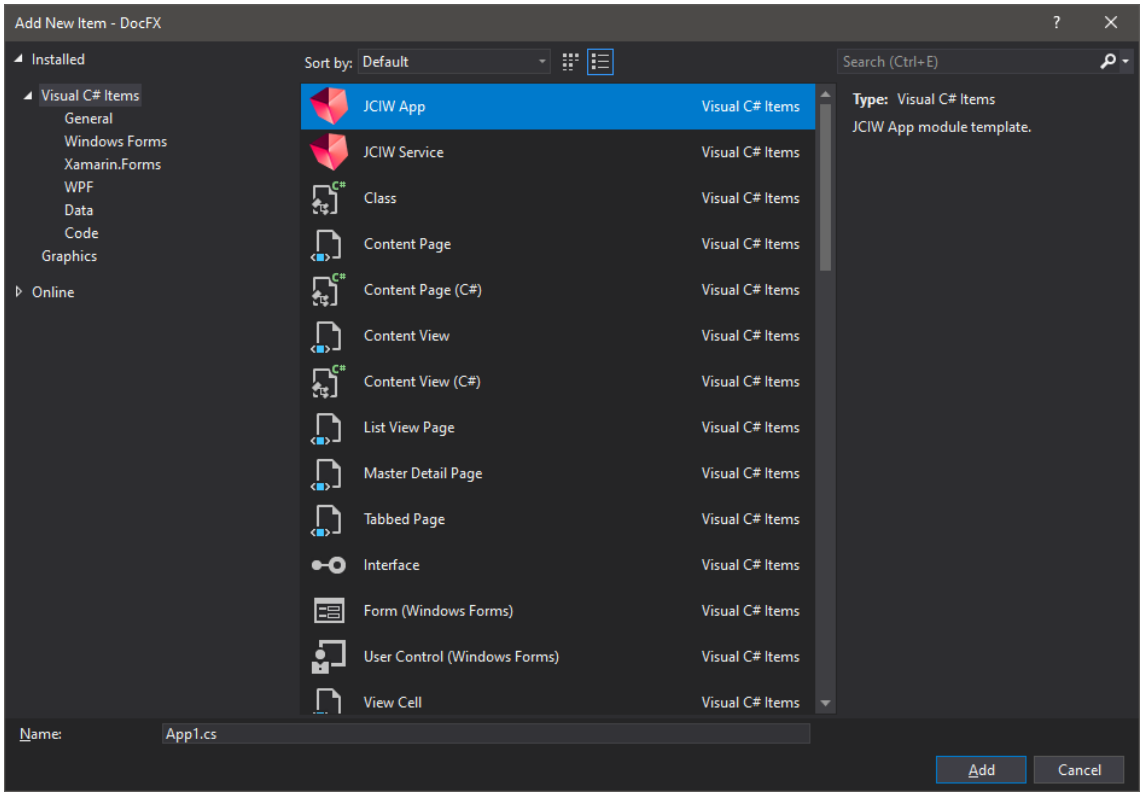


Figure 65: Add New Item... vinduet til Visual Studio

Item-malene sin rolle er å sette opp App og Service klassene slik brukeren bare trenger å fylle ut metodene. See app-modul kapittel for mer informasjon.

```
<Reference Include="ImGui.NET">
  <HintPath>$(JCIWBin)\ImGui.NET.dll</HintPath>
</Reference>
<Reference Include="JCIW">
  <HintPath>$(JCIWBin)\JCIW.dll</HintPath>
</Reference>
<Reference Include="JCIW.Data">
  <HintPath>$(JCIWBin)\JCIW.Data.dll</HintPath>
</Reference>
<Reference Include="Networking.Data">
  <HintPath>$(JCIWBin)\Networking.Data.dll</HintPath>
</Reference>
```

Figure 66: Prosjektmal referansesti

Prosjektmalen ble redigert til å inkludere JCIWBin user environment variabelen i referansestien.

4.13 IJCIWGameComponent

JCIW App har et interface tilgjengelig kalt IJCIWGameComponent. Denne klassen gir brukeren tilgang til å tegne med MonoGame sin GraphicsDevice. Med GraphicsDevice kan en tegne vertice og bilder til skjermen. Under er et eksempel på 3d app med kamera og ImGui kontroller (på Android).

Figure 67: JCIW App 3D grafikk Android

NTNU Island på Android.

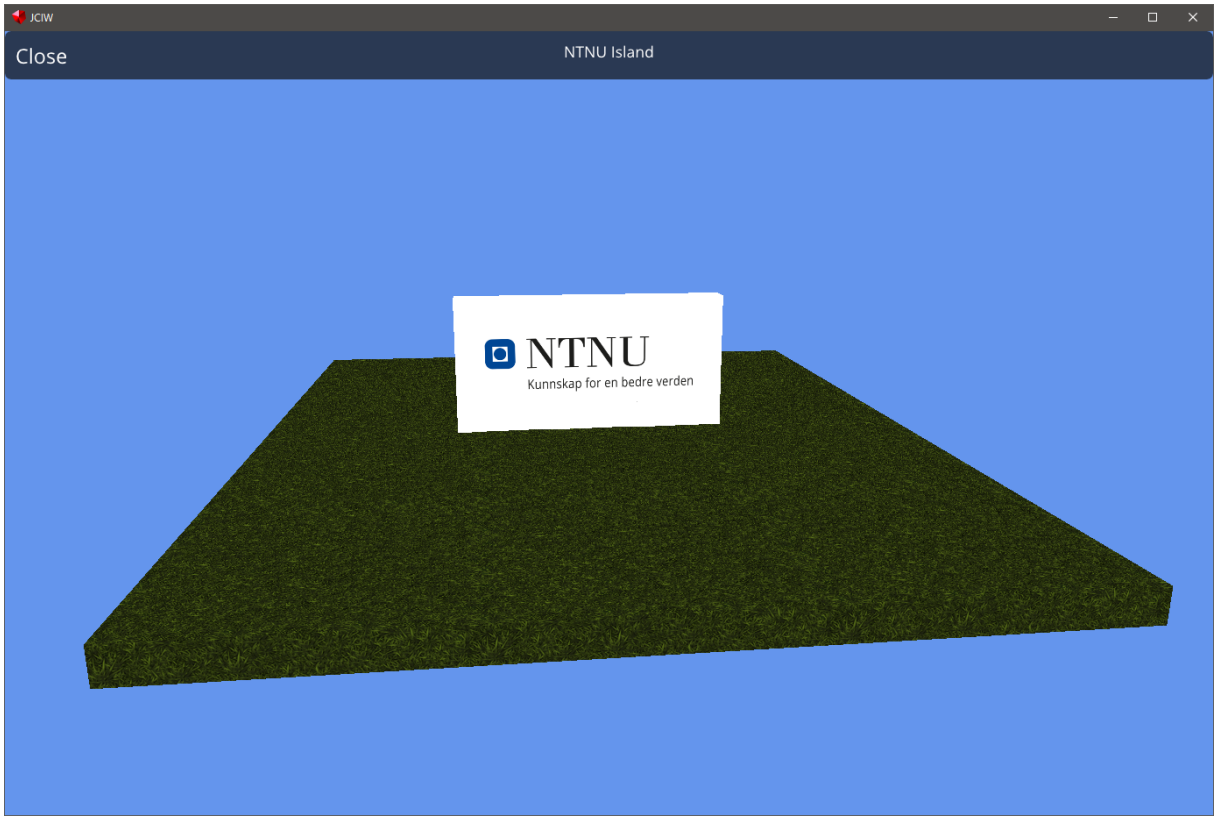


Figure 68: JCIW App 3D grafikk Desktop

NTNU Island på Desktop. Bruker tastatur til å bevege kameraet og ikke knapper.

4.14 Server

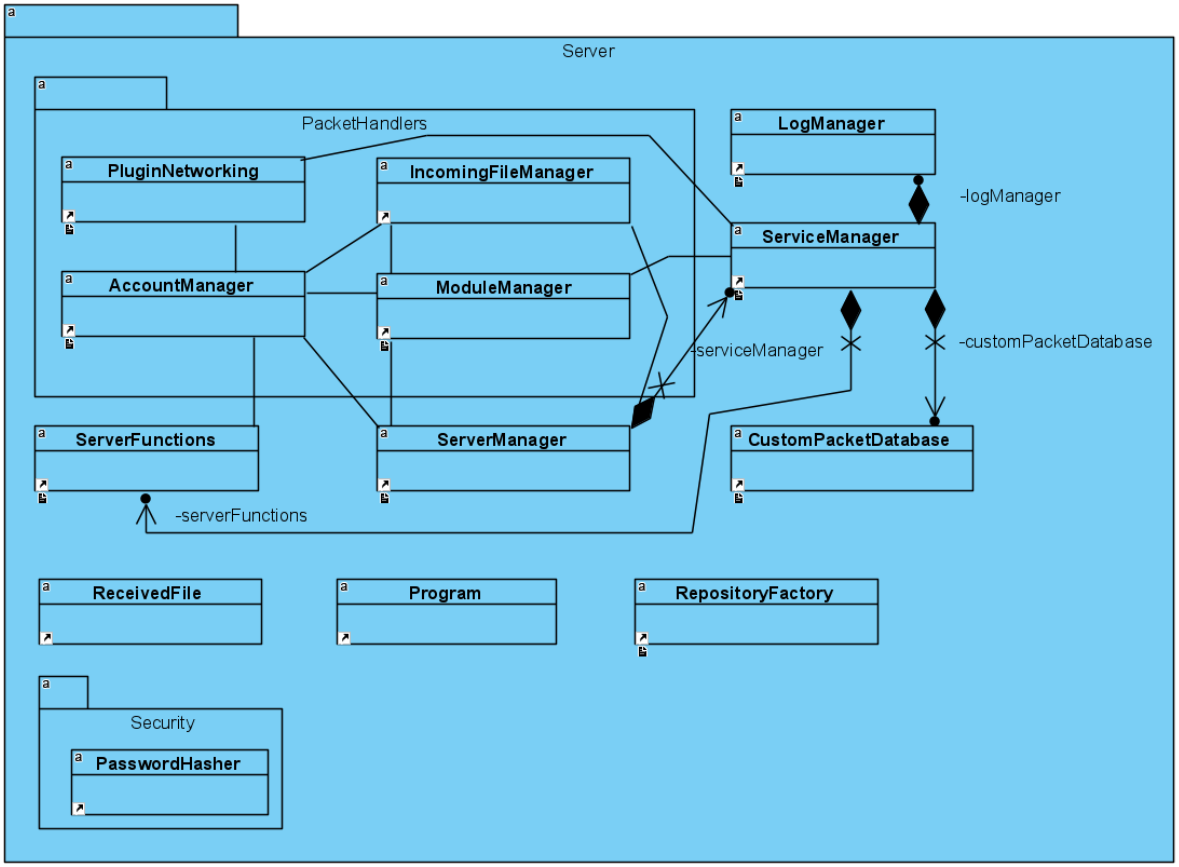


Figure 69: Klassediagram av serverprosjektet

PluginNetworking: Denne klassen har ansvar for å motta og sende Custom pakker.
IncomingFileManager: Denne klassen har ansvar for å motta modulfiler fra admin-klienten.
AccountManager: Denne klassen har ansvar for alle brukerkontorelaterte funksjoner.
ModulManager: Denne klassen har ansvar for alle modulrelaterte funksjoner.
LogManager: Denne klassen er gir service tilgang til logfunksjoner.
ServiceManager: Denne klassen har ansvar for alle servicerelaterte funksjoner.
CustomPacketDatabase: Denne klassen gir service tilgang til databasen.
ServerFunctions: Denne klassen gir service tilgang til serverfunksjoner.
ServerManager: Denne klassen har ansvar for nettverksbiten av serveren.
ReceivedFile: Denne klassen blir brukt av IncomingFileManager for å lagre data.
Program: Dette er Main-klassen til serveren.
RepositoryFactory: Denne klassen er ansvarlig for å opprette repository implementasjoner.
PasswordHasher: Denne klassen hasher passord slik at de ikke står i plain tekst i databasen. Den verifiseren også passordet som brukeren skriver inn når hen logger inn.

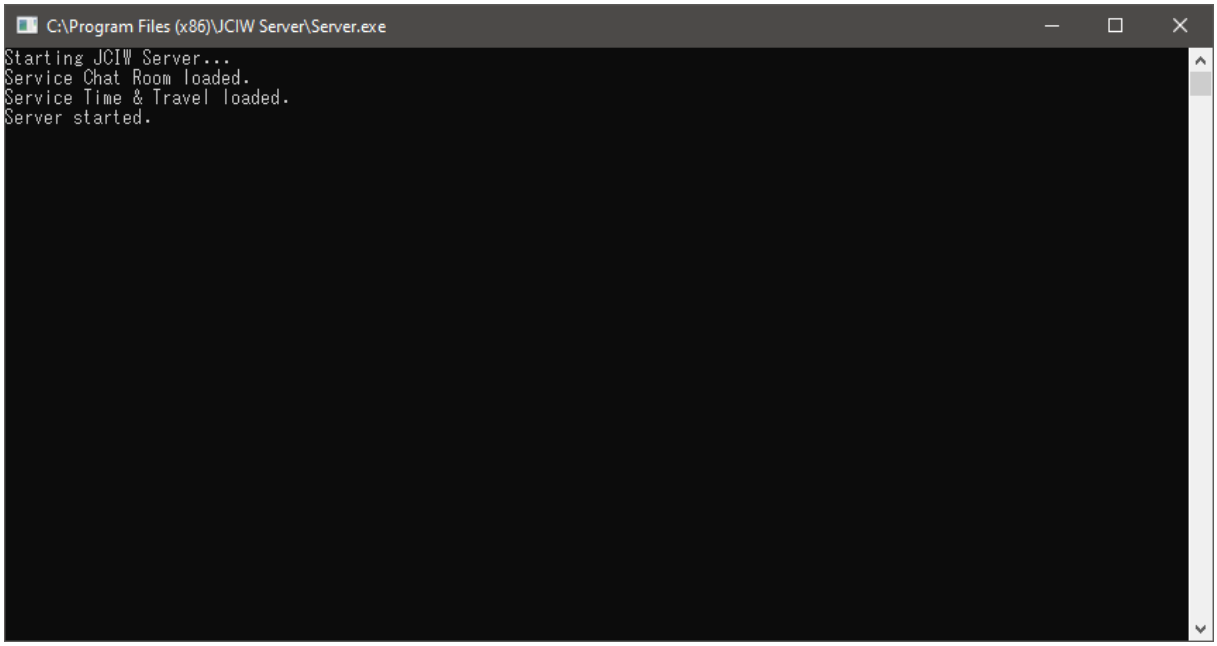


Figure 70: Skjermbilde av server

`JCIW serveren er en C# Console app.`

Den er ansvarlig for å ta imot brukere og autorisere dem ved innlogging. Den holder en hash-liste over autoriserte brukere i AccountManager med connection guid. Dette er NetworkComms.net sin egen identifikasjonsmetode for å identifisere tilkoblinger. Brukerne får i tillegg en session ID som den sender til serveren i tilfeller hvor klienten har blitt disconnected eller åpner klienten for første gang og har en session id lagret til disk fra sist innlogging.

Den håndterer også Servicer, distribuering av apps og database.

Følgende er alle de implementerte pakkene på serveren.

Pakkene sine navnestandard er:

ReXXX = Svar fra server
PostXXX = Innkommende pakke, forventer ikke svar
RequestXXX = Innkommende pakke, forventer svar

```
ReModuleList
ReLoginResult,
ReDeleteGroupFromUser,
```

```
ReDeleteGroup,
ReDeleteAccount,
ReAppGroupList,
ReAppFile,
ReAllAppList,
ReAddGroupToUser,
ReAddGroup,
ReAccountList,
ReUserGroupList,
ReUpdatePassword,
ReUpdateAccountInformation,
ReUnauthorized
ReSessionVerificationResult,
ReSessionId,
ReServiceResponse,
ReServiceLog,
ReRegisterAccount,
RequestUserGroupList,
RequestUserAppList,
RequestUpdatePassword,
RequestUpdateAccountInformation,
RequestSessionVerification,
RequestServiceLog,
RequestServiceCommand,
RequestRegisterAccount,
RequestModuleList,
RequestLogin,
RequestGroupList,
RequestDeleteGroupFromUser,
RequestDeleteGroup,
RequestDeleteAccount,
RequestAppGroupList,
RequestAppFile,
RequestAllAppList,
RequestAddGroupToUser,
RequestAddGroup,
RequestAccountList,
PostRemoveGroupFromApp,
PostEnableService,
PostEnableApp,
PostDisableService,
PostDisableApp,
PostDeleteService,
PostAddGroupToApp
```

4.15 cimgui

For å kompilere android biblioteker må en bruke Android Studio og deretter importere i Visual Studio.

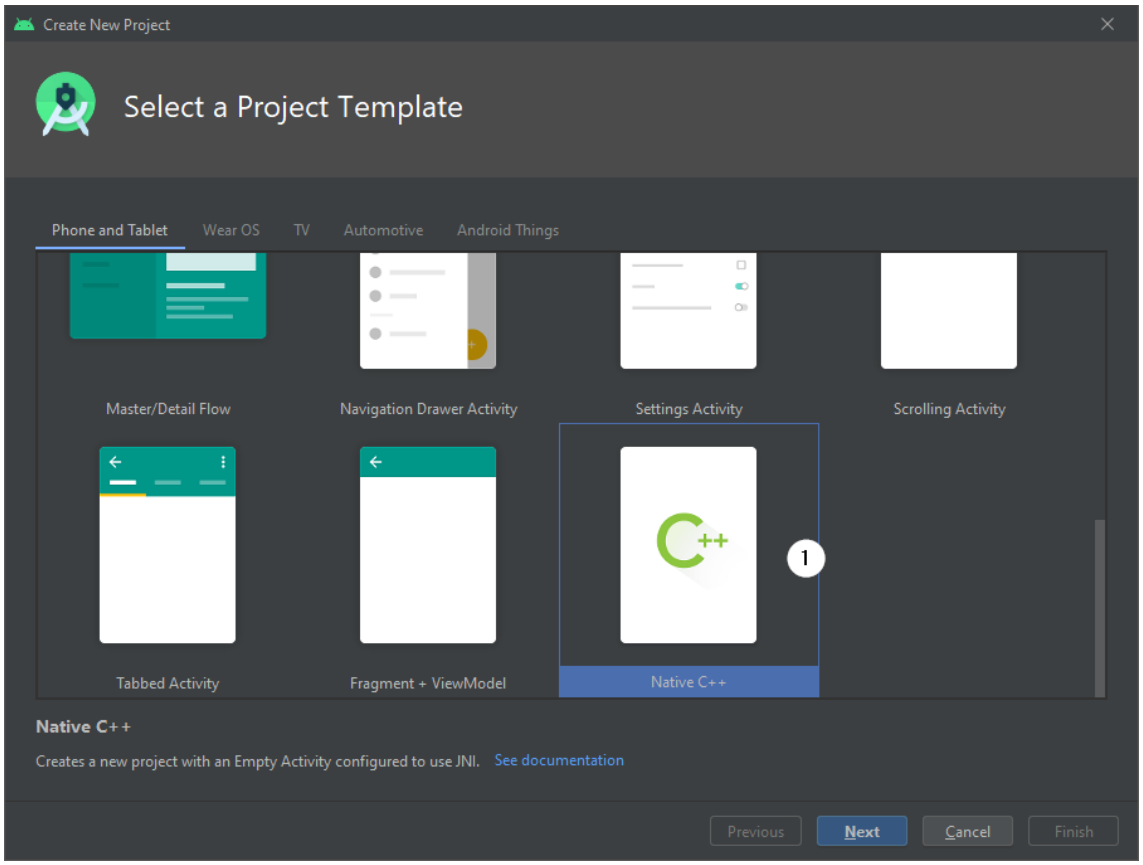


Figure 71: Project Template

1. Native C++. ImGui er kodet i C++.

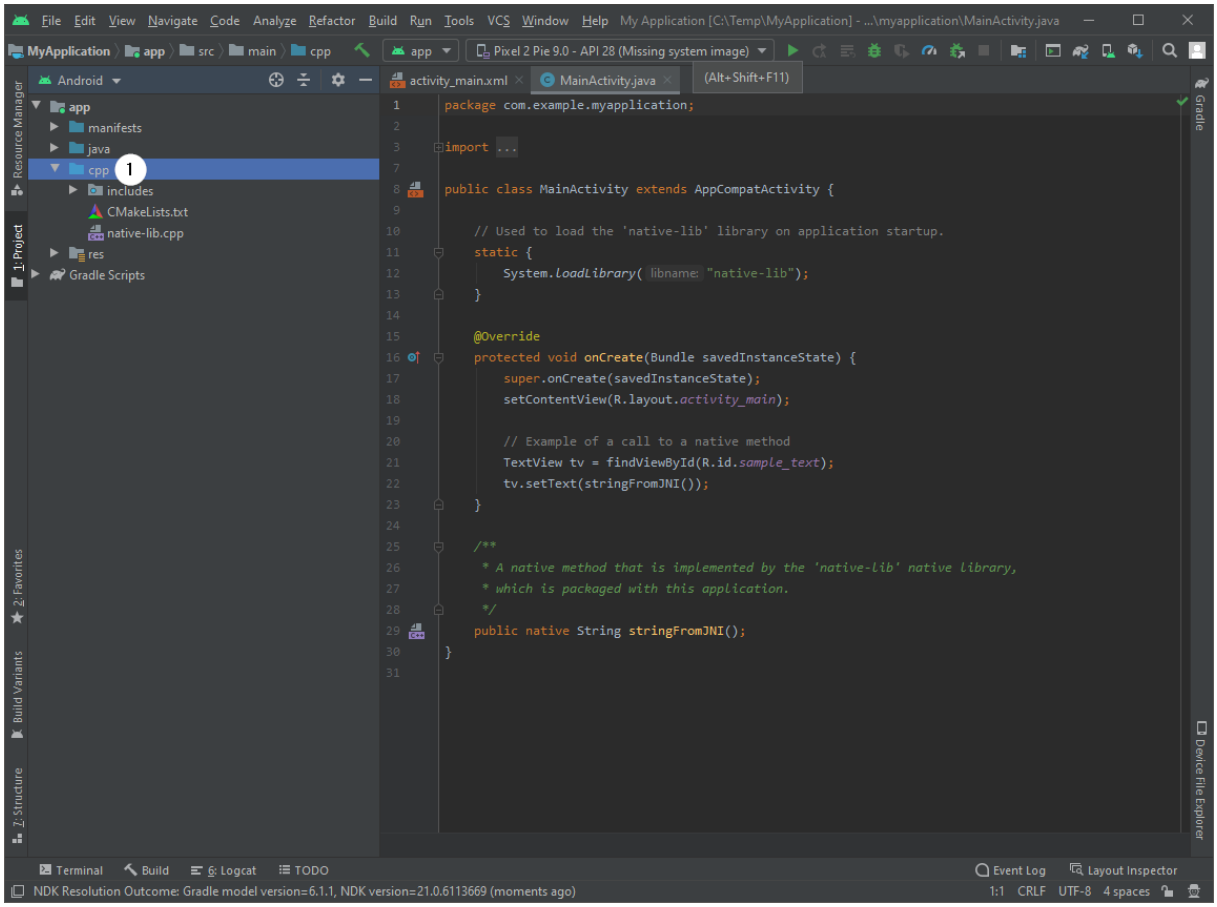


Figure 72: Android Studio

1. Legg imgui koden i cpp

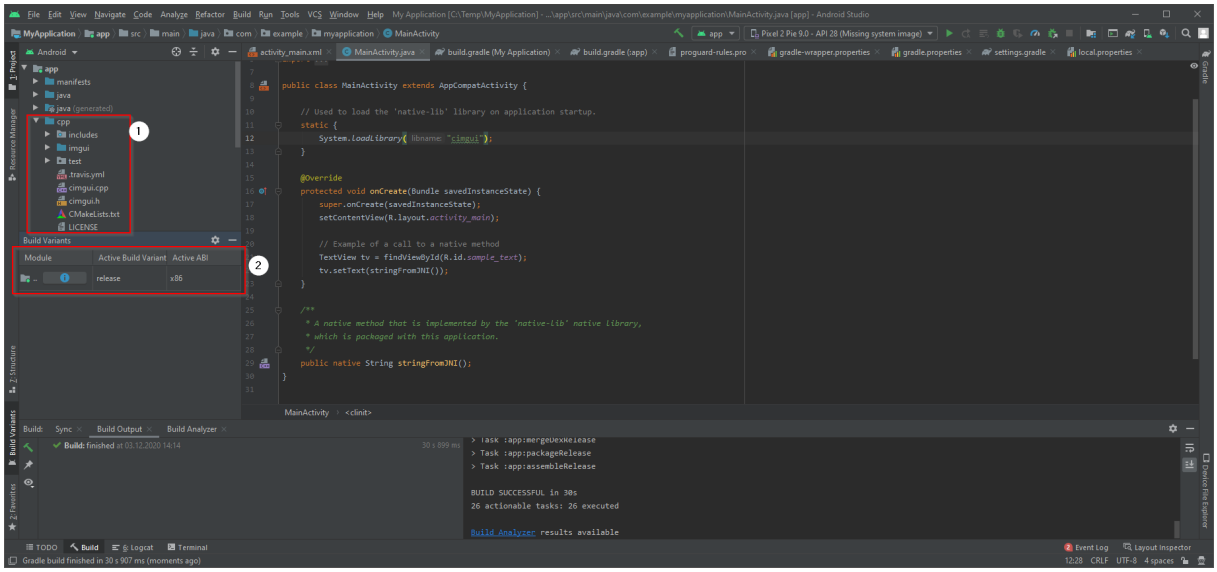


Figure 73: Android Studio build

1. cimgui koden lagt i cpp katalogen
2. Sett release build variant

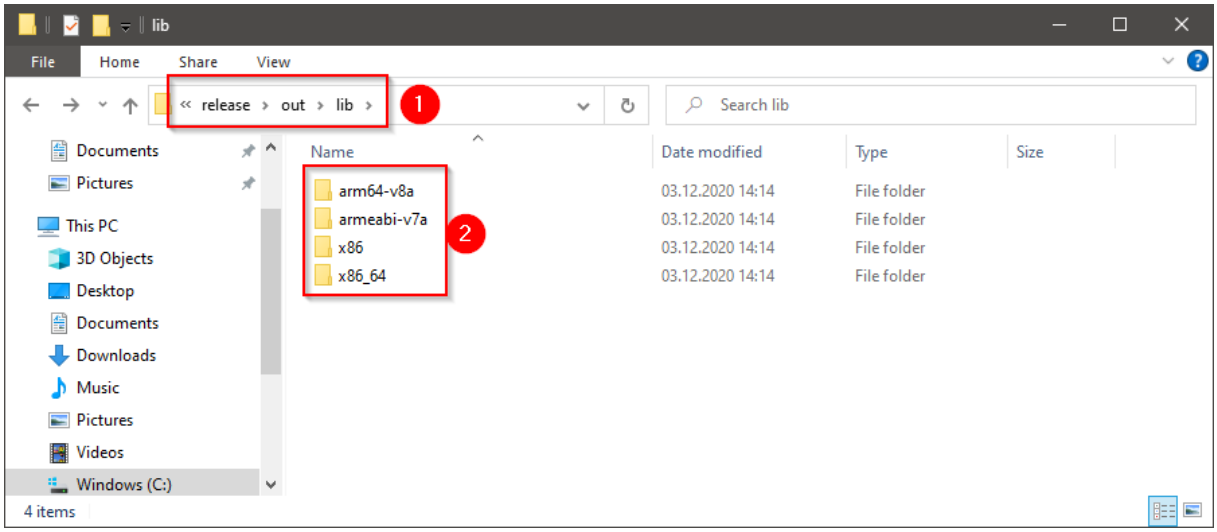


Figure 74: Kompilerte android bibliotek

1. Filene blir kompilert og lagret til release/out/lib
2. Kopier alle filene i mappen

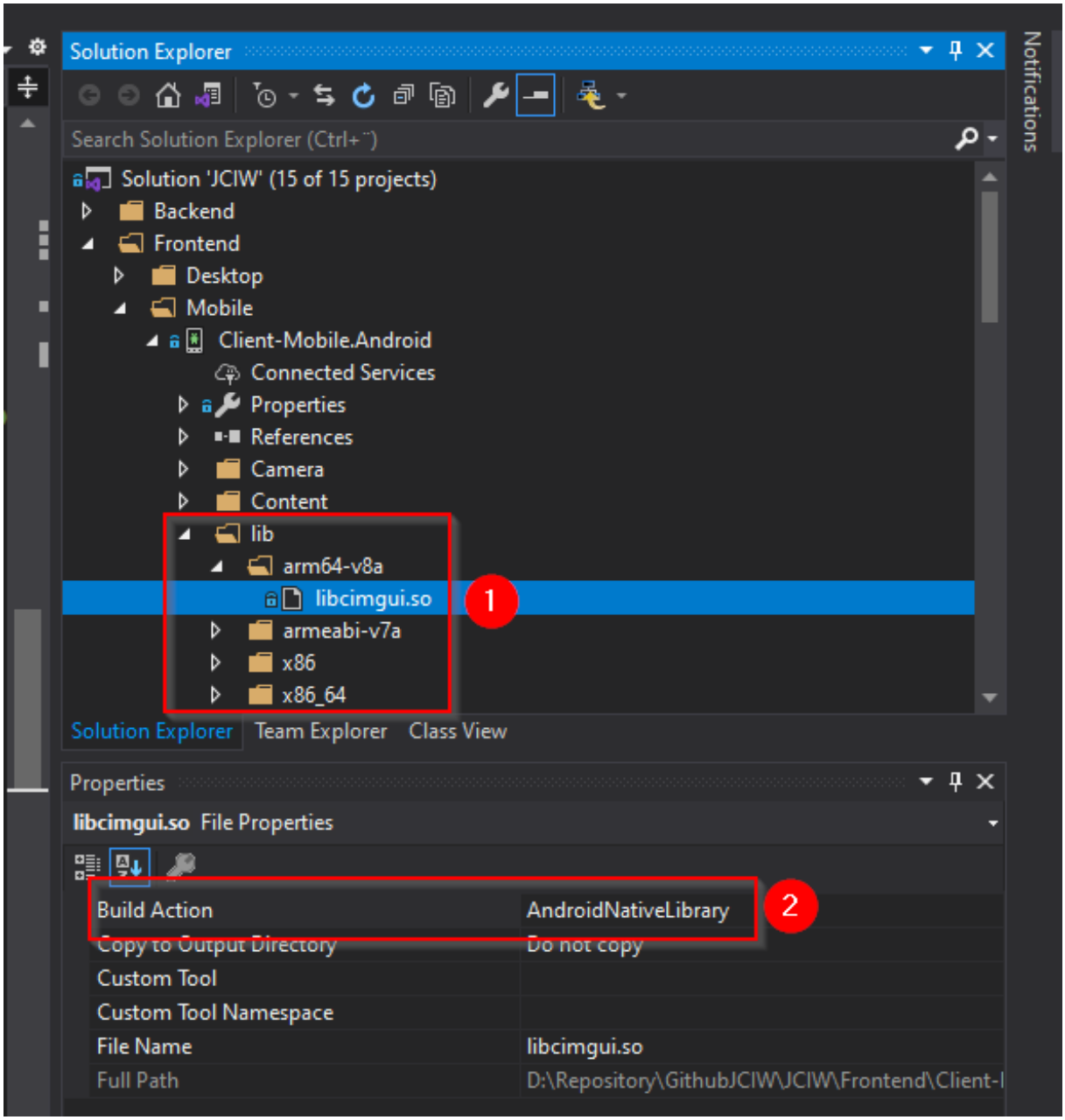


Figure 75: Android native library i Visual Studio

1. Alle androidbiblioteker må ligge i mappestruktur lib/arkitektur. Filene må begynne med lib.
2. Biblioteket må merkes med AndroidNativeLibrary build action.

Gruppen hadde problemer med user input som museklikk og tastetrykk på Android telefoner med ARM 64 arkitektur (alle moderne android telefoner...) noe gruppen ikke klarte å finne ut av årsaken til. Det fungerte på emulator x86 mobilene. Løsningen på problemet ble til slutt å skrive noen forenklete metoder selv i cimgui og imgui biblioteket bare for funksjonene som Android trengte. Følgende er metodene som gruppen la til ImGui (for at de skulle bli synlige måtte de også legges inn i CimGui og deretter ImGui.Net).

```
void ImGui::SetMousePos(int x, int y)
{
        ImGuiContext& g = *GImGui;

        g.IO.MousePos.x = x;
        g.IO.MousePos.y = y;
}

void ImGui::SetKeyDown(int key, int down)
{
    ImGuiContext& g = *GImGui;

    if (down == 1)
    {
        g.IO.KeysDown[key] = true;
    }
    else
    {
        g.IO.KeysDown[key] = false;
    }
}
int ImGui::IsKeyboardWanted()
{
    ImGuiContext& g = *GImGui;
    ImGuiInputTextState* edit_state = &g.InputTextState;

    if (edit_state->ID != 0 && g.ActiveId == edit_state->ID)
    {
        return 1;
    }

    return 0;
}

void ImGui::SetMouseDown(int down)
{
    ImGuiContext& g = *GImGui;

    if (down == 0)
    {
        g.IO.MouseDown[0] = false;
    }
    else
    {
        g.IO.MouseDown[0] = true;
    }
}
```

Figure 76: Fire metoder lagt til i ImGui biblioteket

Disse metodene blir bare brukt av Androidklienten.

4.16 ImGui.NET

ImGui.Net prosjektet og MonoGame implementasjonen er brukt mer eller mindre uten endringer.

Den eneste store endringen som ble gjort var å endre rammeverk fra .Net Standard 2.0 til .Net Framework 4.5 som er hva resten av JCIW prosjektet bruker.

Dette involverte å endre på .csproj og deretter endre på .Net Standard metodene som ikke eksisterer i .Net Framework 4.5.

Figure 77: Eksempel på kodeendringer fra .Net Standard 2 til .Net Framework 4.5

4.17 Statisk kodedokumentasjon

Statisk dokumentasjon ble generert med DocFX.

En dokumentasjonside med JCIW klassene for å utvikle moduler og én full dokumentasjon med alle klassene i løsningen.

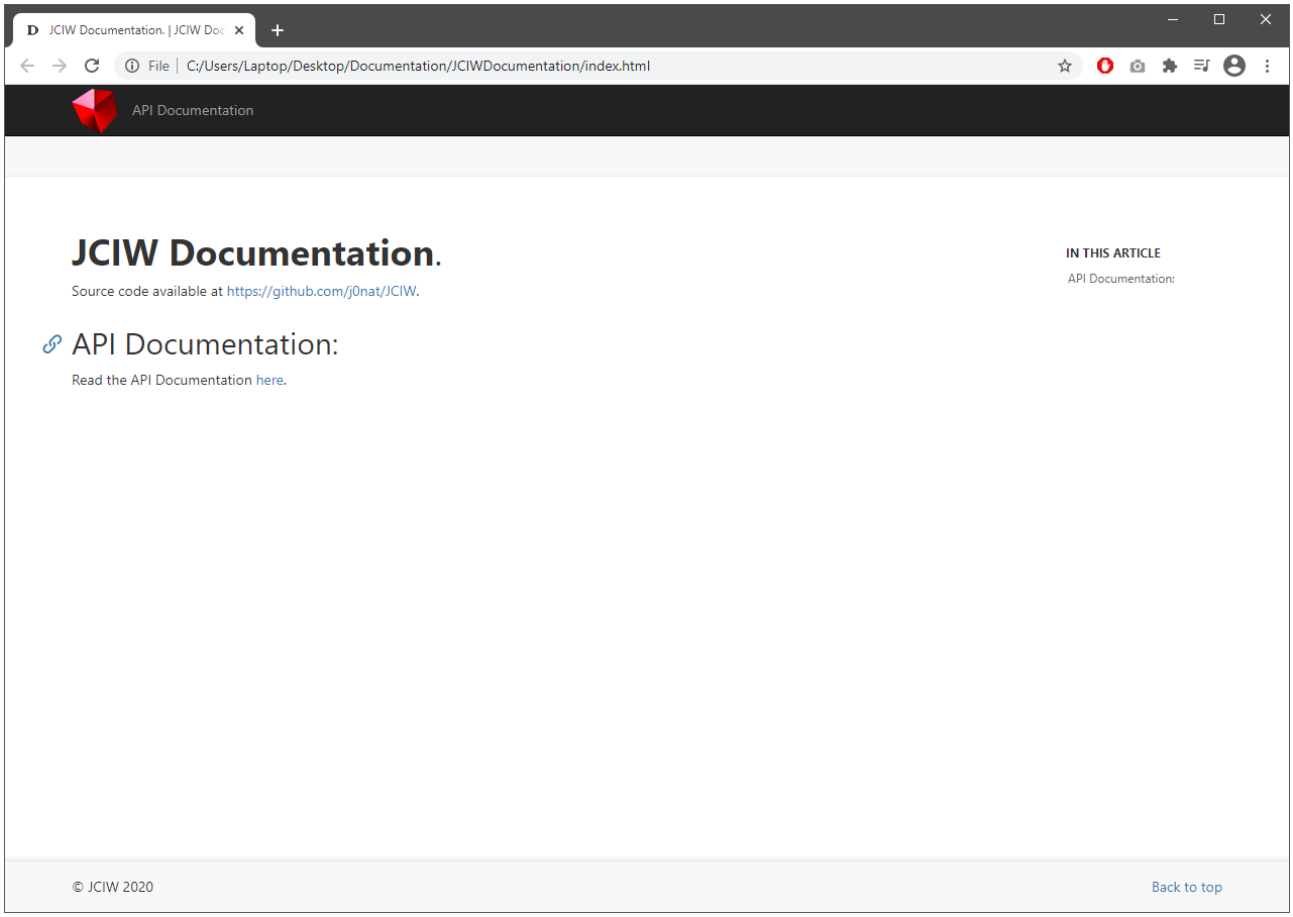


Figure 78: Statisk kodedokumentasjon fremside

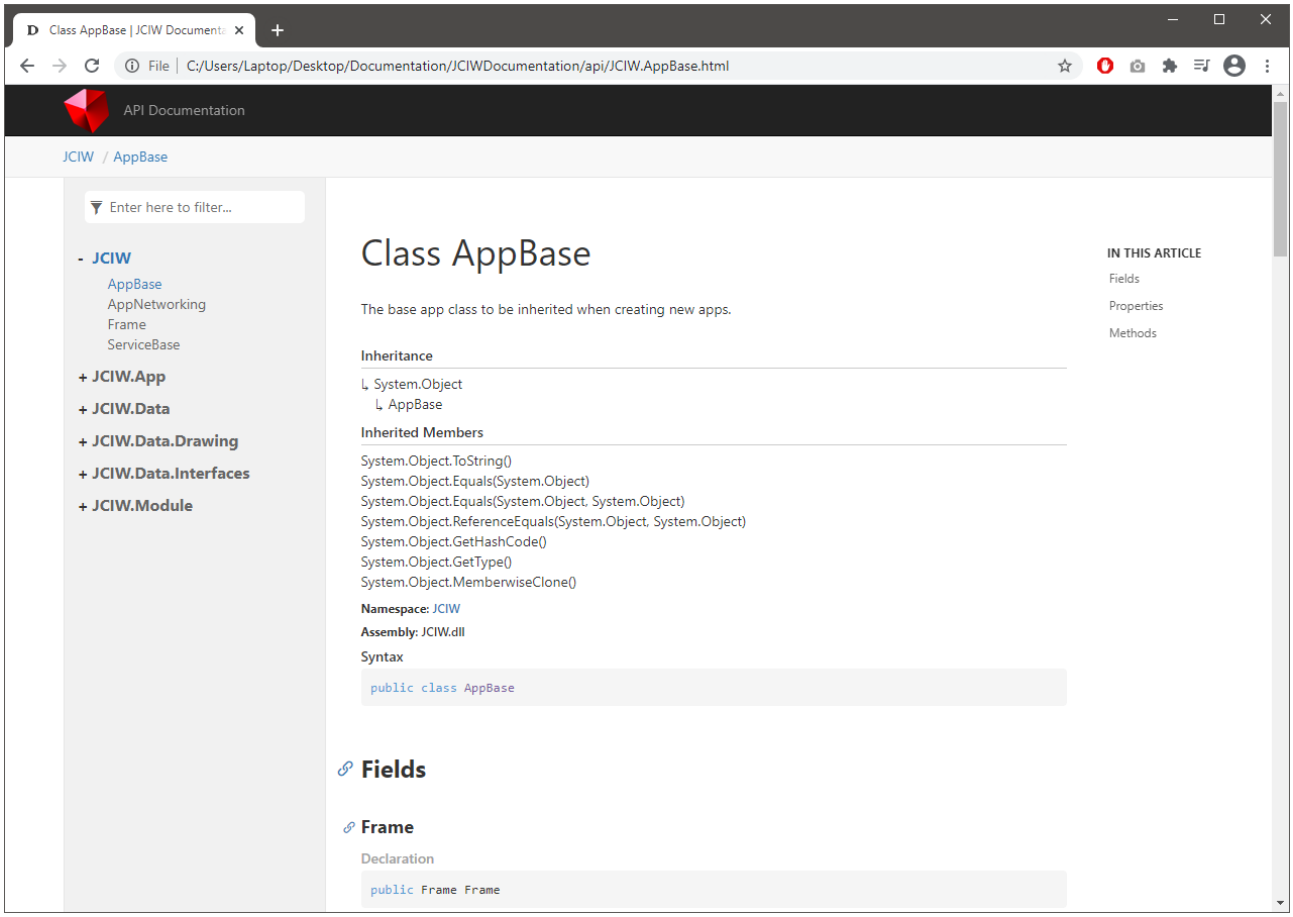


Figure 79: Statisk kodedokumentasjon API Documentation

Sidene ble generert med DocFX sin Visual Studio pakke.

4.18 Fremdriftsplan

Dette er fremdriftsplanen fra forprosjektet:

Hovedaktiviteter i videre arbeid

SPRINT	Beskrivelse
August	Lagde prototype av plugin/grafikk-motoren
Sep 2. - 18.	Implementerte nettverksfunksjonalitet
Sep 18. – Okt 2.	Service og database/datalagring
Okt 2. – 16.	Adminklient og platformspesifikke funksjoner (kamera osv)
Okt 30. - Nov 13.	Ferdigstille grafikkmotor
Nov 13. - Nov 27.	Lage timeførings app for prototype
Nov 27. - Des 11.	Ferdigstille løsning (skriver her rapport)

Figure 80: Fremdriftsplan forprosjekt

Fremdriftsplanen ble følgt nøyaktig, men der manglet en sprint i planen. Sprintene var på to uker.

2. til 16. oktober var den fjerde sprinten og i følge planen startet ikke neste sprint før 30. oktober. Gruppen hadde altså en hel sprint som ikke var planlagt. Denne ble brukt til å jobbe på klientene og timeførings-appen.

4.19 Arbeidsflyt

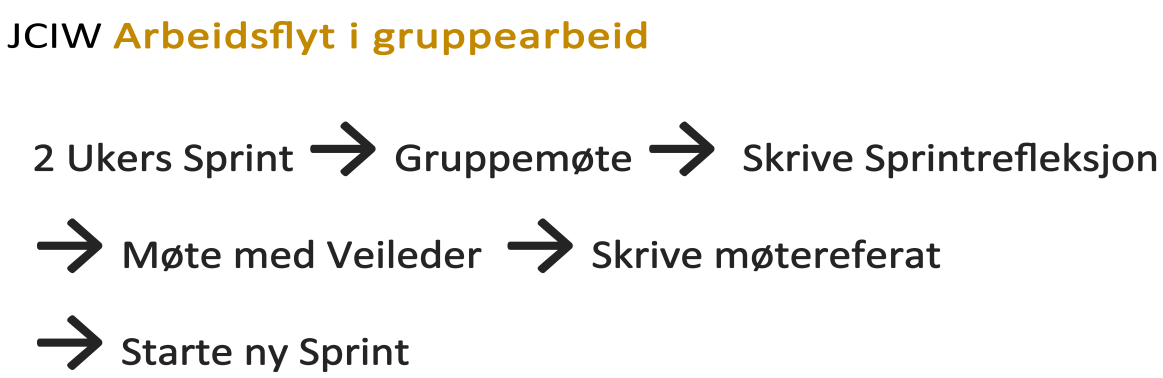


Figure 81: Arbeidsflyt i gruppearbeid

Agile Scrum er et rammeverk for prosjektarbeid. Den definerer roller og rutiner å følge for å løse en oppgave.

JCIW sin arbeidsflyt er basert på utviklingsmetoden Scrum. Daily scrum ble utført på Discord.

4.20 Kjente feil og mangler

- JCIW Klient starter ikke dersom ingen playback device er koblet til PCen.

5 DRØFTING

5.1 Evaluering av resultat

Løsningen ser fra gruppen sitt perspektiv ut til å fungere bra.

Det er i bunn og grunn bare en samling av eksisterende teknologier for å lage et rammeverk. Hovedutfordringen var å få de til å fungere sammen og cross platform.

Oppgaven var veldig nøye planlagt før gruppen startet arbeidet så der var nesten ingen uforutsette hendelser iløpet av utviklingen.

Bruken av ImGui var ikke planlagt i starten av oppgaven og endte opp med å ta ca en uke å implementere. Dette på grunn av arkitekturproblemer som nevnt i resultater. Det var derimot verdt tiden brukt og sparte trolig gruppen mye arbeid. Gruppen sin erfaring med biblioteket har vært positiv.

Androidimplementasjonen har fungert fint på alle mobilene som gruppen har testet. Løsningen på å laste JCIW klienten sine avhengigheter som beskrevet i resultater komme bare fra testing og feiling. Gruppen la merke til at andre applikasjoner som Vipps og LinkedIn brukte den datamappen, og at Xamarin brukte datamappen til å lete etter biblioteker. Gruppen prøvde derfor bare å kopiere inn disse filene manuelt og det viste seg å fungere. Det er derfor mulig at der er en mye enklere og riktigere måte å løse dette problemet.

Et annet problem gruppen hadde med Android var at utvikler ikke har kontroll over brukeren sitt tastatur. Dette er en begrensning ved å bruke Android sitt Soft Input tastatur. En mye bedre løsning på dette hadde vært å brukt ImGui til å tegne tastaturet selv. Da hadde android klienten hatt mye bedre kontroll over tastaturet. Blant annet hvor tastaturet er og hvor stor plass det tar opp på skjermen for å flytte kontrollene. Sånn det er nå så får en ikke ut dette på noen enkel måte.

Gruppen hadde allerede erfaring med NetworkComms.Net, WPF, Repository pattern og MVVM. Det var derfor ikke noen problemer som dukket opp her. Eneste nedsiden ved å bruke WPF til adminklienten var at klienten er begrenset til å kjøre på Windows. I ettertrid ser gruppen at adminklienten også kunne ha blitt kodet med ImGui istedenfor WPF, men det hadde igjen tatt mye lenger tid.

Modullastingen generelt løste seg som planlagt. Det var planen å bruke reflection til å laste biblioteksfiler fra starten av, men gruppen var ikke klar over problemstillingen med un-loading av filene i ettertid. Dette lot seg ikke gjøre og filen forblir ”lastet” helt til hovedprogrammet lukkes. Det ble derfor viktig at modulutvikler implementerer dispose for alle objekter og aktiviteter ordentlig. AppDomain ble en løsning for enkel verifisering av moduler, men det lot seg ikke bruke for modullastingen i seg selv. Dette på grunn av at klasseobjekter som PlatformFunctions eller Frame må sendes til modulfilen for at

modulen skal kunne bruke den. Å sende meldinger frem og tilbake med AppDomain hadde blitt veldig tungvindt. Her er trolig også forbedringspotensiale.

5.1.1 Forbedringer til resultat:

- Et design view som utvikler kan bruke til apputvikling. Sånn det er nå så må en laste opp modulen og teste på JCIW klienten for å se hvordan UIet vil se ut.
- Et eget tastatur til Android klienten. For å ha bedre kontroll på størrelse og plassering.
- Et pakkesystem til moduler for å støtte eksterne biblioteker. Sånn det er nå så er det begrenset til bare en modulfil. Om det hadde vært en .zip kunne en lagt til flere filer om modulen trenger.

5.2 Evaluering av prosjektet

5.2.1 Scrum

Som tidligere nevnt så har vi brukt Scrum som utviklingsmetodikk. I hovedtrekk så har Scrum funkert Ok. Saker med Scrum som ikke har fungert like bra har ikke hatt noe med Scrum i seg selv å gjøre, men heller med oss selv og at vi ikke har utnyttet det fult ut.

Fordeler:

- Planlegging av den kommende sprinten og fordeling av oppgaver.
- Problemidentifisering på grunn av møte minst annenhver uke.
- Veileder har oversikt over arbeidsprosessen.

Ulemper:

- Personlige ulikheter innenfor gruppen som gjorde at Scrum ikke ble brukt optimalt.

5.2.2 Samarbeid

Samarbeidet og kommunikasjonen kunne fungert bedre, spesielt med tanke på at et medlem liker å få ting gjort med en gang, mens det andre medlemmet jobber best

under tidspress. Dette har ført til at det første medlemmet har gjort det meste av arbeidet før det andre medlemmet fikk "sjansen" til å gjøre noe. Dette ble noe bedre etter et sprintrefleksjonsmøte tidlig i prosjektet, men kunne blitt enda bedre dersom det andre medlemmet hadde satt spesifike tidsfrister for de ulike issuene sine.

Vi var heller ikke så flinke til å dele opp issuer i små oppgaver. Hadde de vært mindre, så kunne de nok vært mer fordøyelige, og det ville da vært lettere å komme seg gjennom hvert enkelt issue.

5.2.3 Avvik

Det eneste avviket vi som gruppe hadde er at vi hadde to uker mer på å fullføre oppgaven enn det vi hadde planlagt.

5.2.4 Håndtering av utfordringer

Vår største utfordring var nok samarbeidet, med tanke på at vi hadde ulike arbeidsmetoder. Tidlig prøvde vi å løse dette ved å sette en fast rutine for når vi skulle sette oss ned og jobbe sammen, men dette skeia fort ut og endte med at vi likevel jobba hver for oss til ulike tider som vi først starta med.

En løsning vi gjorde noen ganger var samkoding, der en delte skjermen sin og den andre var med på å foreslå hva som burde skrives i koden.

6 KONKLUSJON

Følgende var målsettingen i starten av prosjektet:
Lage et rammeverk med alle de nødvendige systemene rundt for å kunne utvikle og distribuere applikasjoner cross platform med støtte for TCP og en robust løsning for brukergrensesnitt. De nøyaktig samme applikasjonene skal kunne kjøre på Windows, Linux, Mac og Android.

Løsningen oppfyller målsettingen og inkluderer følgende generelle funksjoner.

- ImGui som er et utbredt og robust bibliotek for å lage brukergrensesnitt.
- Kompatibel med MONO (Derav Linux, Mac og Android støtte)
- Nettverkskode som støtter opp distribueringen av moduler og TCP kravet til løsningen
- Infrastruktur (BACKEND)
  - Opprette og administrere brukere
  - Opprette og administrere grupper
  - Installere og administrere moduler (Service & Apps)
  - Publisere applikasjoner
  - Kjøre modultjenester
  - Begrense publisering med grupper
  - Lagre data i database
- Frontend
  - Desktop og Android klient
    - Innlogging
    - Lasting av modulapplikasjoner
    - Kamerafunksjon på Android klient
    - Brukergrensesnittmotor
    - Browse funksjon på desktop
    - Input på Android
    - Håndtering av disconnect event
    - Håndtering av unauthorized event
  - Adminklient
    - Administrere backend (distribuering av applikasjoner, bruker og gruppeadministrering osv...)
    - Kommunisere og lese logg fra installerte tjenester

- Rammeverk
  - Visual Studio templates
  - Egendefinerte datapakker
  - Platformfunksjoner (kamera, screen density, browse osv....)
  - Tegne direkte til skjerm med MonoGame

Lot problemet seg løse slik gruppen hadde antatt med de valgte metodene?
Gruppen såg tidlig i prosjektet at løsningen ville fungere. Der var utfordringer med implementasjon og noen kurskorrigeringer særlig når det kom til brukergrensesnittet. Bortsett fra valg som gruppen gjorde av tidsmessige grunner så har prosjektet gått nøyaktig som planlagt.

Hva har gruppen lært av dette prosjektet, både teknisk, og ikke minst i forhold til det å jobbe i prosjekt?
Teknisk har gruppen lært å jobbe med pluginsystem.

Gruppen har lært å jobbe med Scrum.

Referanser

[1] Microsoft. Xamarin shared c#. https://dotnet.microsoft.com/learn/xamarin/what-is-xamarin, december 2020.

[2] Atlassian. Scrum - what it is, how it works, and why it's awesome. https://www.atlassian.com/agile/scrum, december 2020.

[3] Wikipedia. Object-oriented programming. https://en.wikipedia.org/wiki/Object-oriented$_{p}rogramming, december 2020.$

[4] Microsoft. The model-view-viewmodel pattern. https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm, december 2020.

[5] Microsoft. The repository pattern. https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff649690(v=pandp.10), december 2020.

[6] Wikipedia. Client-server model. https://en.wikipedia.org/wiki/Client%E2%80%93server$_{m}odel, de$

[7] Wikipedia. Plug-in (computing). https://en.wikipedia.org/wiki/Plug-in$_{(}computing), december 2020.$

[8] JIRA. What is jira used for? https://www.atlassian.com/software/jira/guides/use-cases/what-is-jira-used-forjira-for-agile-teams, november 2020.

[9] Microsoft. What is .net framework? https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet-framework, december 2020.

[10] Hwaci. Sqlite. https://www.sqlite.org/index.html, december 2020.

[11] MonoGame. Monogame. https://docs.monogame.net/, december 2020.

[12] Omar Cornut. Dear imgui. https://github.com/ocornut/imgui, december 2020.

[13] Stephan Dilly. cimgui. https://github.com/cimgui/cimgui, december 2020.

[14] Eric Mellino. Imgui.net. https://github.com/mellinoe/ImGui.NET, december 2020.

[15] NetworkComms.Net. Library features. https://www.networkcomms.net/features/, december 2020.

[16] Microsoft. Wpf overview. https://docs.microsoft.com/en-us/dotnet/desktop/wpf/introduction-to-wpf?view=netframeworkdesktop-4.8, december 2020.

[17] Microsoft. System.reflection. https://docs.microsoft.com/en-us/dotnet/api/system.reflection, december 2020.

[18] Microsoft. System.appdomain. https://docs.microsoft.com/en-us/dotnet/api/system.appdomain, december 2020.

[19] Wikipedia. Windows forms. https://en.wikipedia.org/wiki/Windows$_{F}orms, december 2020.$

7 VEDLEGG

1. Forprosjektrapport
2. Møtenotat
3. Retrospektiv
4. GIT Committer
5. Liste over Jira issuer
6. Arbeidslogger, i .zip-fil
7. Kodedokumentasjon, i .zip-fil
8. Installasjonsfiler, i .zip-fil
9. Setup script, i .zip-fil
10. Kildekode, i .zip-fil
    (a) JCIW løsning
    (b) Chat module
    (c) Time & Travel module

FORPROSJEKT - RAPPORT

FOR BACHELOROPPGAVE

NTNU
Kunnskap for en bedre verden

TITTEL:
JCIW (Joint Company IT Workspace)

KANDIDATNUMMER(E):
-

DATO:	EMNEKODE:	EMNE:	DOKUMENT TILGANG:
17.09.2020	**IE303612**	**Bacheloroppgave**	- Åpen

STUDIUM:	ANT SIDER/VEDLEGG:	BIBL. NR:
DATAINGENIØR	10/0	- Ikke i bruk -

OPPDRAGSGIVER(E)/VEILEDER(E):
Arne Styve

OPPGAVE/SAMMENDRAG:

Lage et rammeverk for utvikling og distribuering av cross-platform nettverksapplikasjoner for en bedrift.

Denne oppgaven er en eksamensbesvarelse utført av student(er) ved NTNU i Ålesund.

Postadresse
Høgskolen i Ålesund
N-6025 Ålesund
Norway

Besøksadresse
Larsgårdsvegen 2
Internett
www.hials.no

Telefon
70 16 12 00
Epostadresse
postmottak@hials.no

Telefax
70 16 13 00

Bankkonto
7694 05 00636
Foretaksregisteret
NO 971 572 140

INNHOLD

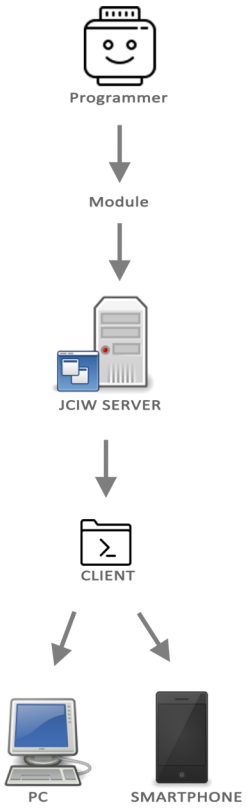
Table of Contents

INNLEDNING

Oppgaven er å lage en løsning for å utvikle cross-platform nettverksapplikasjoner og enkelt distribuere applikasjonene i en bedrift. Tenk Citrix workspace, bare at appene er selvutviklet istednfor å være strømmet fra server. Den skal oppfylle noe av behovet for et Citrix miljø samtidig som å være en altomfattende IT-løsning for en bedrift sine applikasjonsbehov både på server og klient. Dette krevet et robust utviklings API hvor utviklere enkelt kan definere nettverkskommunikasjon, lage grafiske brukergrensesnitt og mulighet til å bruke platformspesifikke funksjoner som å ta bilde med Android telefon. Målet er å gjøre dette så enkelt som mulig for utvikleren og så enkelt som mulig for IT å deretter distribuere.

BEGREPER

JCIW TERMINOLOGY



MODULES

Plugin / Extension (.dll) published to JCIW

- ***Service**: Code running continously on server.*
- ***Job**: Code running once on server and posting results.*
- ***App**: Graphical application.*

JCIW Server

User login (rights management), groups, information storage
hosting of services (service execution engine) and publishing of modules.

JCIW Client

Admin workspace and app execution engine.

MODULE FUNCTIONS

- ***Install**: Upload .dll file to server and make it available to deploy..*
- ***Deploy**: Launch an instance / deployment of a module.*
- ***Archive**: Deactivated deployments (includes the log and database and module file (.dll).*

PROSJEKTORGANISASJON

Prosjektgruppe

Studentnummer(e)
460142 (Jonathan Øie) 476526 (Liv Anne Nyland)

Oppgaver for prosjektgruppen - organisering

Jonathan Øie (prosjektleder):

- Definere oppgaven i sitt omfavn, tydeliggjøre arbeidsoppgaver og metoder.
- Prosjektleder for oppgaven
- CTO for oppgaven

Liv Anne Nyland:

- Scrum master
- Utvikler

Styringsgruppe (veileder og kontaktperson oppdragsgiver)

Gruppen har ingen oppdragsgiver. Oppgaven er selvdefinert. Veilederen er Arne Styve. Han underviser i programmering hos NTNU i Ålesund.

AVTALER

Arbeidssted og ressurser

Gruppen jobber eksternt hver for seg og benytter egne PC'er. Den eneste ressursen som benyttes offisielt er veileder.

Gruppenormer – samarbeidsregler – holdninger

Møte med veileder annenhver fredag, med møte innad i gruppen i forkant. Agile med to ukers sprinter brukes slik gruppen enkelt kan gjøre endringer og justeringer dersom problemer oppstår som forsinker tidsplanen.

Gruppen vil derfor jobbe systematisk og diskutere problemer som dukker opp for å finne løsninger. Kodestruktur og dokumentasjon vil ha prioritet siden systemet rettes mot utviklere og IT.

PROSJEKTBESKRIVELSE

Problemstilling - målsetting - hensikt

Problemstilling: Løsningen gjør livet enklere for to grupper. Utviklere og IT i en bedrift.

For **IT** er problemet det å kunne distribuere applikasjonen på en pålitelig og sentralisert måte med tilgangs- og brukerkontroll. Dette gjøres i et enkelt adminkonsoll hvor alle tjenester og moduler kan administreres og som gjør utrullingen av oppdateringer og nye applikasjoner raskt og enkelt. Dette begrenser også den nødvendige brukeradministrasjonen fra IT sin side til én bruker og gjør feilsøking av applikasjonsproblemer mye enklere da alle kjører samme versjon. Brukeren sin klient laster alltid ned nyeste versjon av applikasjon fra server.

For **utviklerne** av applikasjonene så har løsningen store fordeler for effektivitet. Spesielt når det kommer til nettverksapplikasjoner. Løsningen sitt rammeverk effektivisere skrivingen av nettverkskode ved å abstrahere bort pakkebehandling og datalagring. Utvikleren trenger heller ikke tenke på brukerhåndtering, bortsett fra ved gruppekontroll om det er ønskelig. En annen fordel for utvikleren er å kunne skrive i én app og distribuere på server, mobil, og datamaskin uten ekstra arbeid. Løsningen sin adminkonsoll gjør det også enkelt å debugge tjenester siden utvikleren enkelt kan definere egne kommandoer i service-koden og kjøre disse via adminkonsollet for å hente ut status.

Målet er derfor å:

- Kunne skrive og kjøre applikasjoner til datamaskin og mobil
- Kunne skrive og kjøre tjenester til server (med og uten applikasjon)
- Distribuere applikasjonene via en sentralisert hub
- Lage et robust rammeverk for utvikling av apllikasjoner
- Lage en robust administrasjonskonsoll for å hjelpe IT og utviklere distribuere og administrere applikasjonene.

Krav til løsning eller prosjektresultat – spesifikasjon

Under er noen bilder som illustrerer ønsket prosjektresultat.

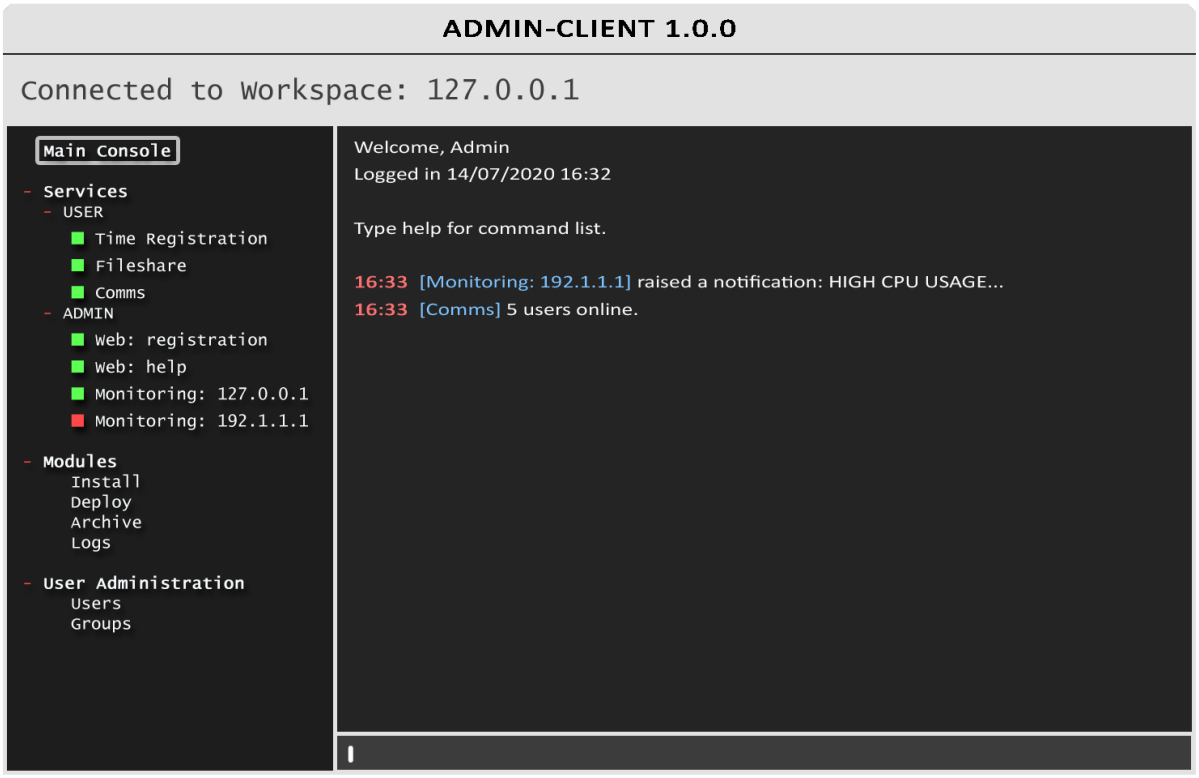
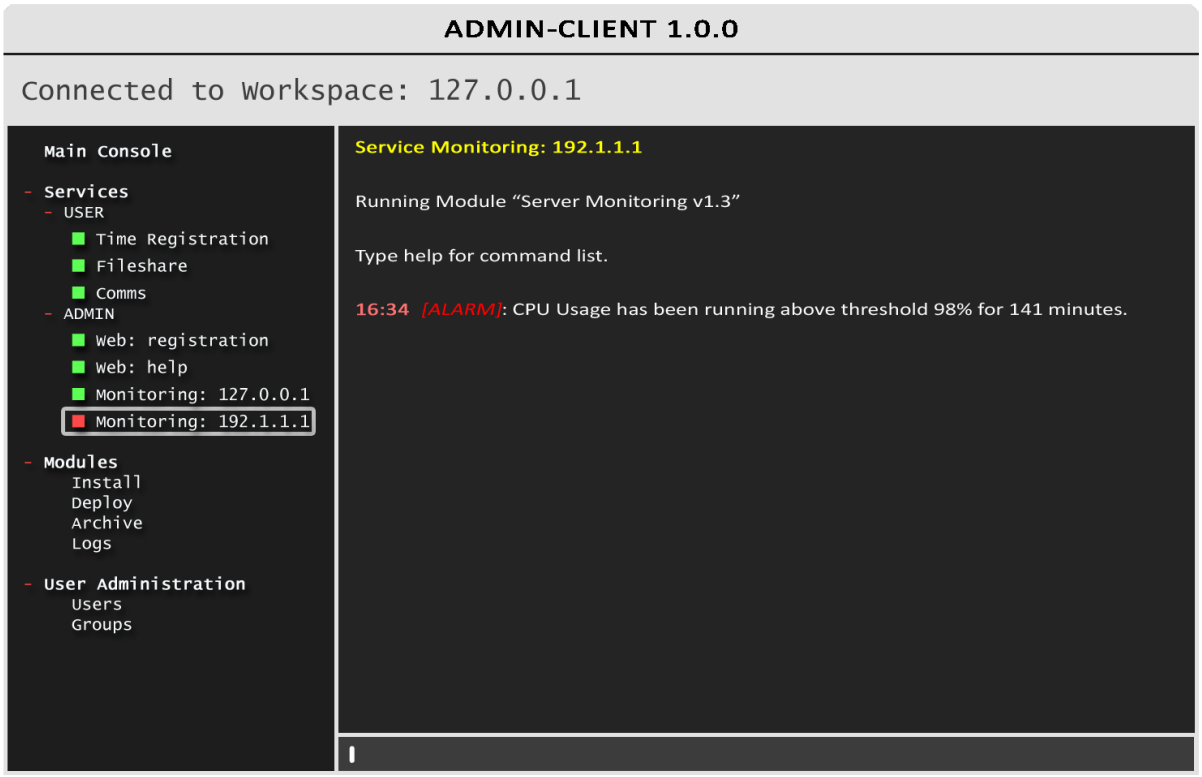


Figure 1: Adminkonsoll



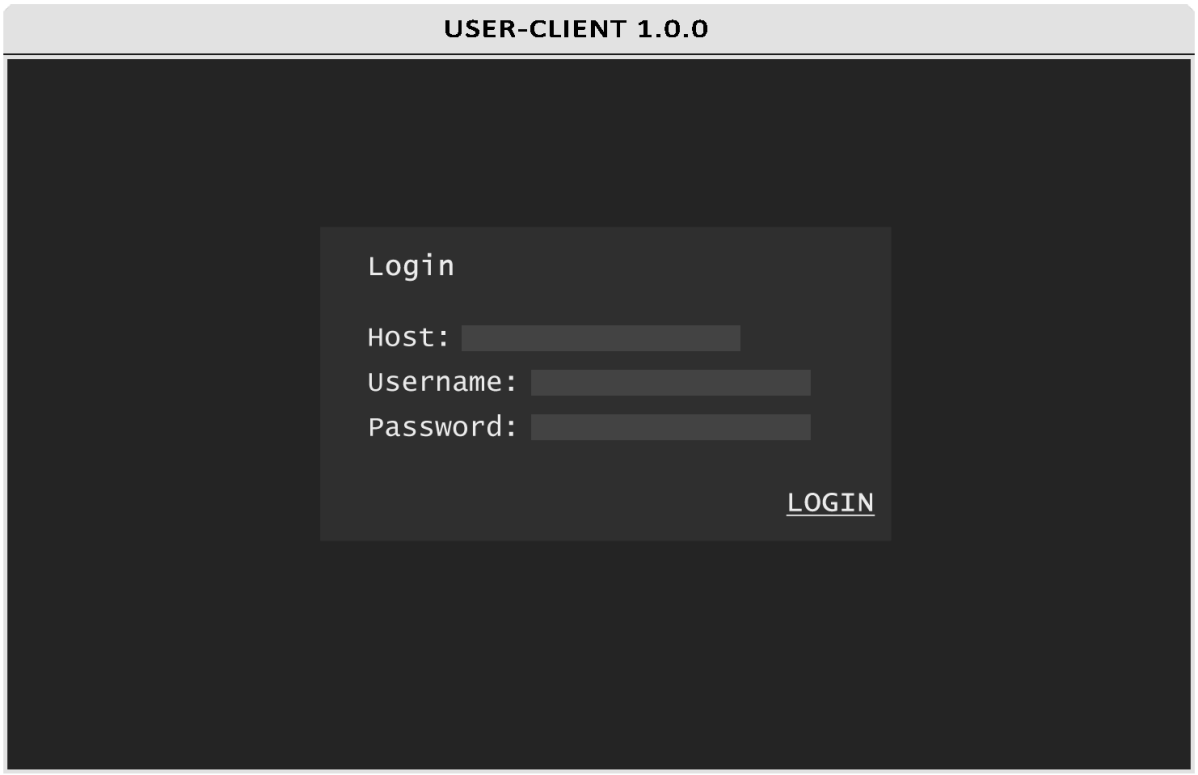
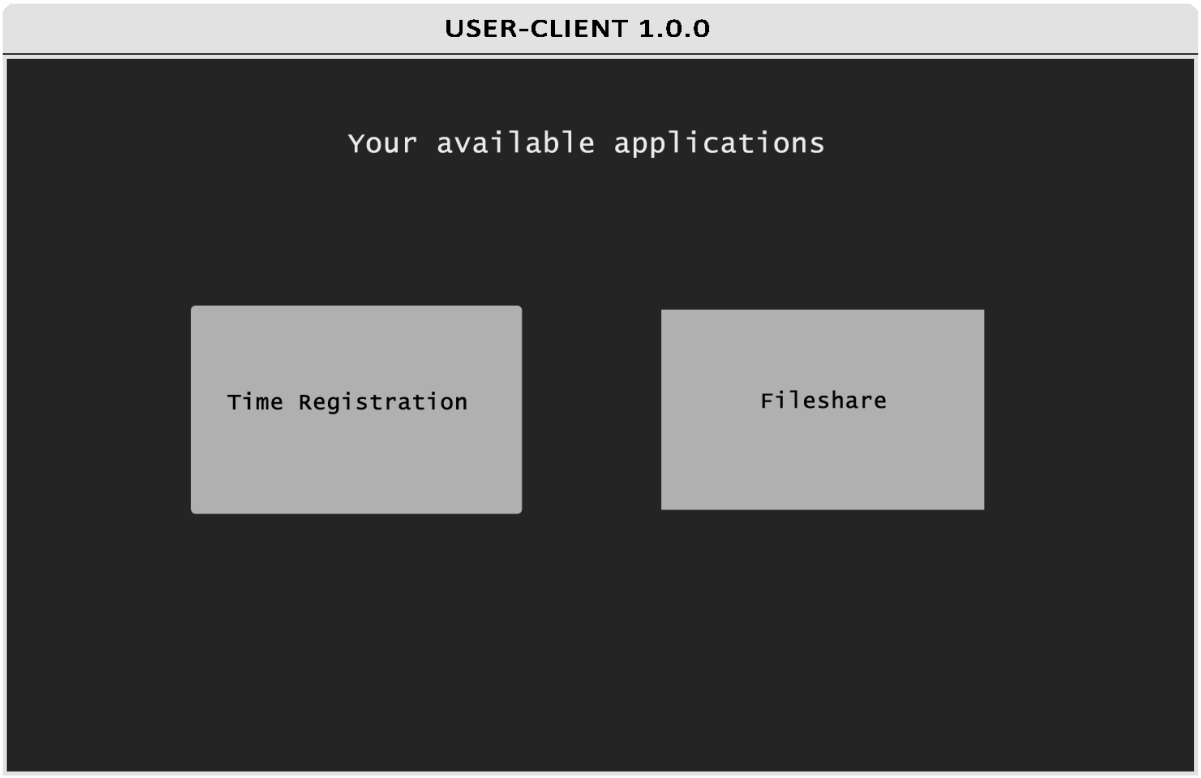


Figure 2: Brukerklient



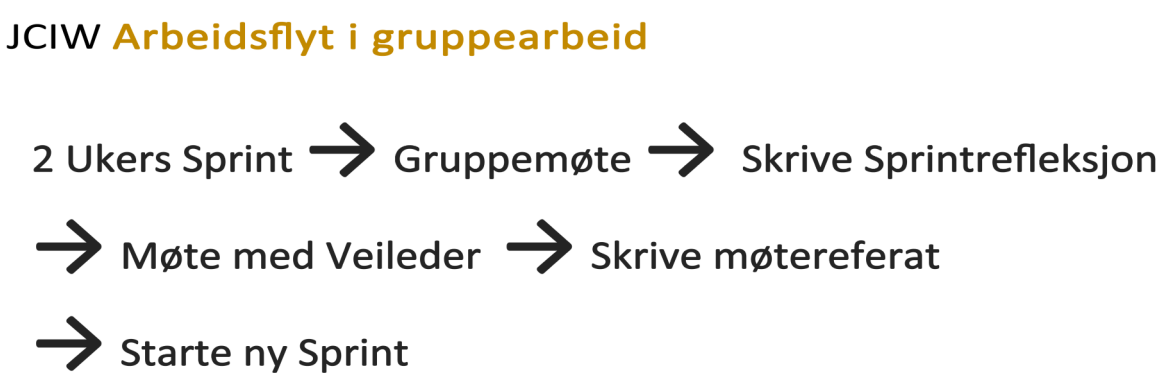
Ikke illustrert på bildene er pluginutviklingen. Dette er det tredje prosjektresultatet.

En utvikler skal kunne utvikle en plugin hvor han lager en service eller en app eller en "job" (eller alle tre) og distribuere disse til brukerne av systemet. Det skal fungere på Desktop og mobil.

USE CASES:

Utvikler	**Sluttbruker**	**IT**
Lage app	Logge inn	Installere og publisere apps og plugins
Lage service	Åpne apps	Opprette grupper
Lage nettverkskommunikasjon		Opprette brukere
Definere datalagring		Administrere moduler
Lage brukergrensesnitt		Restriktere publisering med grupper
Bruker grupperettigheter fra JCIW brukeren		Logge inn på adminkonsoll

Planlagt framgangsmåte(r) for utviklingsarbeidet – metode(r)



Informasjonsinnsamling – utført og planlagt

På tidspunktet forprosjektet er skrevet så har gruppen allerede funnet løsninger og laget prototyper som forhåpentligvis vil la seg skalere. Gruppen vil prøve å lage en grafikkmotor som ligner Visual Studio sin Windows Forms og Java sin Swing.

Nettsøk burde være tilstrekkelig til eventuelle videre kodeproblemer som oppstår, men gruppen har ingen sentral ressurs hvor byggingen av systemet er beskrevet.

Gruppen har noe kunnskap om prosjektområdet fra semesteroppgave i Systemutvikling (http://jonoie.com/STWFA/) studering på fritid (http://jonoie.com/2019/09/30/mvvm-gui-large-database-operations-application.php) og en del prosjekter ellers: https://github.com/j0nat?tab=repositories.

Vurdering – analyse av risiko

Tidsplanen for å realisere prosjektet er realistisk da prosjektets omfang kan minimeres og antall tilbydde funksjoner for utvikleren senket. Det å kode brukergrensesnitt vil kanskje komme til å ta lenger tid enn det er satt av til. Dette på grunn av at det skal fungere både på mobil og desktop samtidig, noe som skaper utfordringer med skalering og sideforhold. Det er mulig brukergrensesnittet må avgrenses i løpet av prosjektet.

Det som vil være viktig for å lykkes med prosjektet er en god og ryddig kodestruktur og kode. Der er mange komponenter som skal jobbe sammen og dersom strukturen er dårlig skaper det mye ekstraarbeid og misforståelser, noe som taper gruppen tid.

Hovedaktiviteter i videre arbeid

SPRINT	**Beskrivelse**
August	Lagde prototype av plugin/grafikk-motoren
Sep 2. - 18.	Implementerte nettverksfunksjonalitet
Sep 18. – Okt 2.	Service og database/datalagring
Okt 2. – 16.	Adminklient og platformspesifikke funksjoner (kamera osv)
Okt 30. - Nov 13.	Ferdigstille grafikkmotor
Nov 13. - Nov 27.	Lage timeførings app for prototype
Nov 27. - Des 11.	Ferdigstille løsning (skriver her rapport)

Framdriftsplan – styring av prosjektet

Hovedplan

Planen er å følge sprintene og bygge løsningen stegvis.

Styringshjelpemidler

Gruppen bruker Jira og Bitbucket.

Intern kontroll – evaluering

Evaluering og kontroll blir gjort i sprintmøtene.

Beslutninger – beslutningsprosess

En bestemmelse må gjøres i slutten av sprintene.

DOKUMENTASJON

Rapporter og tekniske dokumenter

Kodedokumentasjon med for.e DocFX

Codestyle

Møtereferat

Sprintrapport

PLANLAGTE MØTER OG RAPPORTER

Møter

Møter med styringsgruppen

Møte med veileder annenhver fredag, der vi gjennomgår arbeid som har blitt utført, diskusjon om hvordan vi skal fortsette, og andre relevante spørsmål.

September:
18. Møte med veileder og sprintslutt

Oktober:
2. Møte med veileder og sprintslutt
16. Møte med veileder og sprintslutt
30. Møte med veileder og sprintslutt

November:
13. Møte med veileder og sprintslutt
27. Møte med veileder og sprintslutt

Desember:
11. Mulig møte

Prosjektmøter

Prosjektgruppen har blitt enig om å ha møter hver torsdag, der vi går gjennom arbeid som har blitt utført, diskutere ting som har vært vanskelig, planlegge nye sprinter annenhver torsdag, og ofte jobbe sammen på enten selve prosjektet, eller på prosjektrapporten.

Periodiske rapporter

Framdriftsrapporter (inkl. milepæl)

To rapporter lages annenhver uke. Sprintrapport og møtereferat.

PLANLAGT AVVIKSBEHANDLING

Etter hver sprint må det vurderes om funksjoner må kuttes ut eller om gruppen skal bruke mer tid på noe dersom det ikke er ferdigstilt innen den planlagte tidsrammen.

Møtenotater

2020-09-18 møtenotater

Opprettet av Arne Styve, sist endret av Liv Anne Nyland nå nettopp

Dato

18.sep.2020

Deltakere

- @Arne Styve
- @Jonathan Øie
- @Liv Anne Nyland

Mål

- Få tilbakemelding på Forprosjektrapporten
- Gå gjennom Sprint og Use Cases

Debattpunkter

Tidspunkt	Element	Hvem	Notater
0 min	Gå over til Discord i staden for Skype for Business		• Discord fungerte mye bedre
4 min	Jira og Confluence		• Startet med privat Atlassian-domene • Arne fiksa Jira/Confluence prosjekt til oss under skolen sitt domene • Gikk gjennom hvordan å bruke Confluence
30 min	Forprosjektrapport		• Rapporten forklarer litt for dårlig • Skriv hovedoppgava på en sånn måte at en med ikke-teknisk bakgrunn kan forstå den
40 min	Visual Paradigm		• Gikk gjennom hvordan å bruke Visual Paradigm for bruk til UML og sekvenssiagram • Arne fikk i oppgave om å fikse lisens for Visual Paradigm til oss
57 min	Kodekvalitet		• SonarQube • SonarLint
1t 1min	Vår kode		• Rask gjennomgang av koden vår og hva den kan gjøre til nå • Gjennomgang av hvordan vi ser for oss at applikasjonen skal gjøre og hvordan den skal se ut • Koden mangler dokumentasjon
1t 16min	Møte ferdig		

Handlingselementer

- [x] @Arne sett opp JIRA og Confluence
- [x] @Arne fikser lisens til Visual Paradigm
- [x] @Jonathan legg til @Arne i Bitbucket

2020-10-02 Meeting notes

Opprettet av Jonathan Øie, sist endret av Liv Anne Nyland nå nettopp

Date

02.okt.2020

Attendees

- @Arne Styve
- @Liv Anne Nyland
- @Jonathan Øie

Discussion items

Time	Item	Who	Notes
20min	Overleaf	Arne	• Anbefaler ikke at vi begynner på dette nå om vi ikke har brukt det før • Fungerer ellers veldig bra men er en liten terskel å bruke • Arne kan lage mal til oss dersom vi ikke finner på blackboard
10min	Teknisk nivå	Arne	• Bachelorrapporten fokuserer på selve prosjektet. Hva gikk bra, hvilke beslutninger ble tatt og hvorfor etc. • Systemdokumentasjon skal inneholde eksempel og veiledning for bruk av systemet. Trenger ikke dette i rapporten. • Helst ikke så mye kode i rapporten.

Action items

- [x] @Jonathan Øie Sjekk på Blackboard om vi finner Latex mal til bacheloroppgaven. Visst ikke kan Arne fikse.
- [x] @Arne Styve Skal se om han finner noen software bachelorrapporter og sende til oss.
- [x] @Jonathan Øie @Liv Anne Nyland https://datasift.github.io/gitflow/IntroducingGitFlow.html Se på å bruke flere branches i bitbucket

2020-10-16 møtenotater

Opprettet av Liv Anne Nyland, sist endret den okt 29, 2020

Dato

16.okt.2020

Deltakere

- @Liv Anne Nyland
- @Jonathan Øie
- @Arne Styve

Mål

- Gå gjennom sprint
- Få tak i LaTex mal
- Purre arne på bachelorfrist
- Eksempel på bachelorrapport

Debattpunkter

Tidspunkt	Element	Hvem	Notater
	Overleaf og LaTex		• Malen vi fant var Ok
	Bachelorrapport		• Detaljnivå i rapport • Bestemme sjøl • Arne skal sjekke bachelorfrist
	Demonstrere prosjekt		• Vise koden • Koden kompilerer • Demonstrere funksjonalitet • Fjerning av moduler kan vere en "pitfall"

Handlingselementer

- [x] @Arne finn bachelorfrist
- [x] @Jonathan og @Liv må begynne å dokumenter kode
- [x] @Jonathan og @Liv må samareide tettere

2020-10-30 møtenotater

Opprettet av Liv Anne Nyland, sist endret av Jonathan Øie på nov 13, 2020

Dato

30.okt.2020

Deltakere

- @Liv Anne Nyland
- @Jonathan Øie
- @Arne Styve

Mål

- Gå gjennom fullført sprint
- Vise fremgangen i prosjektet

Debattpunkter

Tidspunkt	Element	Hvem	Notater
10	Gjennomgikk gammel sprint		
10	Demonstrerte nye funksjonalitet	@Jonathan Øie	
10	Snakket om sprint og kommende frister.		

Handlingselementer

- [x] Føre inn utviklingsplan i backlog @Jonathan Øie
- [ ] Legge inn releases i Jira

2020-11-13 Meeting notes

Opprettet av Jonathan Øie, sist endret den nov 27, 2020

Date

13.nov.2020

Attendees

- @ Jonathan Øie
- @ Liv Anne Nyland
- @ Arne Styve

Goals

- Gå gjennom seneste sprinten
- Oppdatere veileder

Discussion items

Time	Item	Who	Notes
15	Presenterer sist sprint	@ Jonathan Øie	•
10	Diskuterte bachelerrapportskriving	@ Jonathan Øie @ Arne Styve @ Liv Anne Nyland	
25	Installerte og testet Sonarlint	@ Jonathan Øie	

Action items

- [x] Ha som mål å reinskrive i uke 51
- [x] Vurdere hvordan presentasjonen burde være (PowerPoint?)
- [x] Skal rapport og kildekode levers i Inspera? @ Arne Styve

2020-11-27 Meeting notes

Opprettet av Jonathan Øie, sist endret av Liv Anne Nyland på nov 27, 2020

Date

27.nov.2020

Attendees

- @Jonathan Øie
- @Liv Anne Nyland
- @Arne Styve

Goals

- Gå gjennom user stories (hvordan lage)
- Gi Arne tilgang til rapport

Discussion items

Time	Item	Who	Notes
	Rapport og presentasjon		• Overleaf • Objektorientert programmering • Bruk boka
	User stories		• User story tekst • User story diagram • Sekvensdiagram • Klassediagram
	Presentere fungerende program		
	Planen videre		• To uker rapport • En uke presentasjon • Ta kontakt før 11. desember om rapporten er ferdig

Action items

- [ ]

2020-12-11 Meeting notes

Date

11.des.2020

Attendees

- Jonathan Øie
- Liv Anne Nyland
- Arne Styve

Goals

> Hva finnes i markedet i dag som løser dette/prøver å løse dette? Hva skal JCIW tilføre som ikke løsninger i markedet allerede løser?
> Dec 8, 2020 1:40 PM

- Gå gjennom introduksjon
- Resten av rapporten?

Discussion items

Time	Item	Who	Notes
10min	Snakket on presentasjon	Arne	
20min	Snakket om rapport (architectual pattern)		

Retrospektiv

2020-09-18 Tilbakeblikk

Dato	18.sep.2020
Deltakere	Jonathan Øie Liv Anne Nyland

Vi jobbet mest med forprosjektrapporten og hadde derfor ikke så mye tid til koding denne gangen. Neste sprint blir vår første ordentlige sprint i prosjektet.

Tilbakeblikk

Hva fikk vi bra til?

- Fullførte forprosjektrapport
- Begynte på nettverkskode

Hva burde vi ha gjort bedre?

- Til neste sprint vil vi dele inn utviklingsarbeidet jevnere

2020-10-01 Tilbakeblikk

Dato	01.okt.2020
Deltakere	Liv Anne Nyland Jonathan Øie

Tilbakeblikk

Hva fikk vi bra til?

- Jobbe jevnt
- Koden kompilerer

Hva burde vi ha gjort bedre?

- Fullføre sprint

Handlinger

- [ ] Legge poeng på kvart issue
- [x] Jonathan Øie Lage prototype av timeføringsapp

2020-10-15 Retrospective

Date	15.okt.2020
Participants	Jonathan Øie Liv Anne Nyland

Retrospective

What did we do well?

- Vi fullførte sprinten

What should we have done better?

- Kunne ha fordelt oppgavene litt bedre i forkant av denne sprinten sånn mer funksjonalitet var forberedt til når adminklient skulle opprettes.

  Ble en mye større sprint enn nødvendig.
- Fordele oppgavene og samarbeide bedre med issuene.
- Planlegge når vi starter også jobbe sammen med issues.

2020-10-29 Tilbakeblikk

Dato	29.okt.2020
Deltakere	Liv Anne Nyland Jonathan Øie

Tilbakeblikk

Hva fikk vi bra til?

- Gjort alt viktigste som sto i opprinnelig sprint
- Starta dokumentering (ferdig med tre prosjekt)
- Bedre sammarbeid i gruppen

Hva burde vi ha gjort bedre?

2020-11-13 Retrospective

Date	13.nov.2020
Participants	Jonathan Øie Liv Anne Nyland

Retrospective

What did we do well?	What should we have done better?
• Fikset små og store design og stabilitetsproblemer med løsningen da vi plutselig hadde mere tid enn vi hadde planlagt.	

Ifølge den originale utviklingsplanen som vi la inn i forprosjektet så skulle vi være ferdig med utviklingen denne sprinten. Det viste seg at vi hadde 2 ekstra uker grunnet feiltelling av uker når planen ble laget.

2020-11-27 Retrospective

Date	27.nov.2020
Participants	Jonathan Øie Liv Anne Nyland

Retrospective

What did we do well?	What should we have done better?
• Feature complete • Dokumentasjon ferdigstilt (har ikke generert statiske html filer enda) • Fått en god start på bachelorrapport • Begynt å mestre Overleaf • Fått ut prosjektet på Github	

2020-12-11 Retrospective

Date	11.des.2020
Participants	Jonathan Øie Liv Anne Nyland

Retrospective

What did we do well?

- Kommet godt i gang med å fullføre rapporten

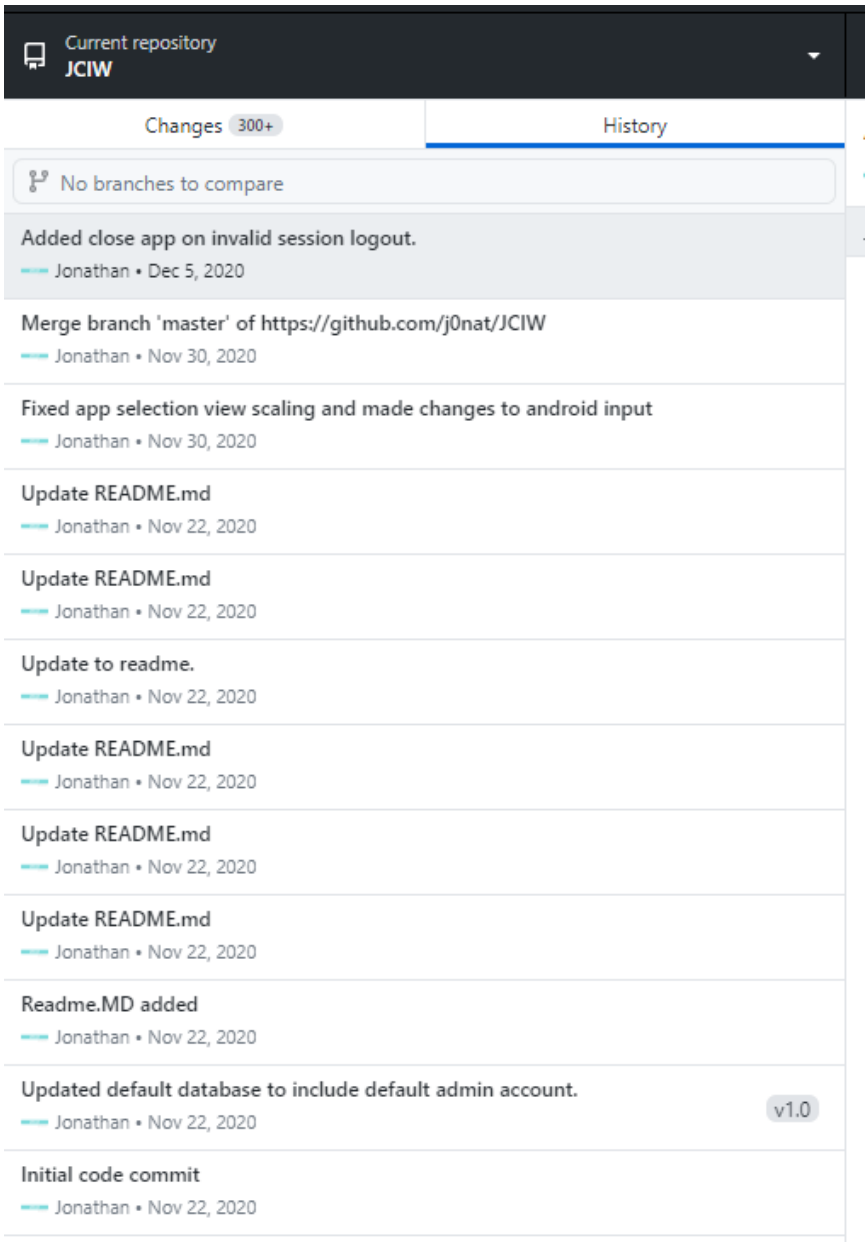
What should we have done better?

- Fikk ikke fylt ute hele rapporten. Skulle egentlig være ferdig til idag.

Actions

- [x] Renskrive introduksjonen
- [x] Fullføre presentasjon
- [x] Fullføre DocFX statisk dokumentasjon

GIT Committer



Author	Commit	Message	Date
Jonathan	44936e6	Documentation	2020-11-22
Jonathan	6592682	Server documentation	2020-11-22
Jonathan	4137cd1	Changes to repository (changed user to account) and added param name comm...	2020-11-22
Jonathan	bfd30a2	Changed userId to accountId solution wide. Also including database change.	2020-11-22
Jonathan	1b15dea	Client-admin partically documented.	2020-11-22
Liv Anne	6b0e943	Repository and Repository.SQLite documentation	2020-11-22
Jonathan	4e69374	Networking.Data documentation.	2020-11-21
Jonathan	dc03d5c	GraphicsEngine documentation.	2020-11-21
Jonathan	cfe3eeb	Documentation	2020-11-21
Jonathan	cad0a86	Added SetBackgroundColor method to frame.	2020-11-19
Jonathan	9190733	Made JCIWGameComponent interface. Allows modules to access MonoGame dr...	2020-11-19
Jonathan	e4b4107	Linux / macos fix	2020-11-18
Jonathan	f973b68	Change to browse	2020-11-18
Jonathan	005cd04	Added browse (exclusively to desktop...)	2020-11-18
Jonathan	3f3a465	Changed how texture loading is done	2020-11-17
Jonathan	df83e69	Set some writelines on the server.	2020-11-17
Jonathan	1dc9901	Added domain to admin client login	2020-11-17
Jonathan	0ec9439	Fixed packets only appended once.	2020-11-17
Jonathan	e283ff7	Created installers for server, template and clients	2020-11-17
Jonathan	19b8dae	Raised FPS back up to 30. (tried 15)	2020-11-17
Jonathan	d61bf50	Changed register limit to 1 char down from 3. Added host input to clients.	2020-11-17
Jonathan	fb1cf00	Fixed android dependencies and path issue on server (under mono)	2020-11-16
Jonathan	2aa89a4	Fixed android camera function and added ability for packages to send byte array.	2020-11-14
Jonathan	d28f590	Fixed mouse behavior on android	2020-11-13
Jonathan	37c33ec	Added JSON support to modules	2020-11-13

Author	Commit	Message	Date
Jonathan	acb97e6	Add packet field now supports updating field.	2020-11-13
Jonathan	bb8e9ba	Changes to app selection list design	2020-11-11
Jonathan	32b552f	Set window padding to once (SetNextWindowPos)	2020-11-11
Jonathan	c38396e	Resized diconnected (retry close) buttons	2020-11-10
Jonathan	6e023c0	Added padding to apps (to draw the content under the overlay)	2020-11-09
Jonathan	9687657	Added threaded connect check (to avoid freezing ui for long)	2020-11-08
Jonathan	c61dbec	Changed how mobile scaling is done. Fonts are now scaled in stead of windows.	2020-11-08
Jonathan	38cde8c	Added overlays to all JCIW.app views.	2020-11-08
Jonathan	710f418	Design changes	2020-11-08
Jonathan	fba62f4	Added title bar to apps	2020-11-08
Jonathan	3614a56	Changed application icon for desktop and mobile.	2020-11-07
Jonathan	da8538b	Fixed service enable / disable crash.	2020-11-07
Jonathan	bdfc7ee	Fixed keyboard focus code (now checks if a text input is focused)	2020-11-07
Jonathan	4f0dd8b	Implemented scaling for network disconnect window.	2020-11-06
Jonathan	4c2cc6b	Updated android icon	2020-11-06
Jonathan	a108186	Added read database function.	2020-11-05
Jonathan	068acfd	Fixed android networking crash	2020-11-05
Jonathan	9ed85c3	Implemented authorization check and network disconnect on client	2020-11-04
Jonathan	26b113c	Added authorization check and implemented session on clients.	2020-11-04
Jonathan	9da81e0	More design changes to admin client	2020-11-03
Jonathan	a0256ee	Design changes to Admin Client	2020-11-02
Jonathan	901c92c	set admin client back to 127.0.0.1	2020-11-02
Jonathan	4a45a02	Fixed camera function and reduced image size	2020-11-02
Jonathan	2905ea7	arm64 mouse fix	2020-10-29
Liv Anne	42a6a06	Session	2020-10-29

Author	Commit	Message	Date
Jonathan	78f38fd	Implemented open keyboard on android	2020-10-27
Jonathan	326e971	JCIW.Module documentation.	2020-10-26
Jonathan	8ef4e27	JCIW.Data project documentation.	2020-10-26
Jonathan	8e26562	JCIW project documentation.	2020-10-26
Jonathan	98a2edd	Password hashing implemented	2020-10-26
Jonathan Øie	f0ded7f	MERGED Merged feature/Develop into master	2020-10-26
Jonathan	5d56ea5	Refactoring passwordhashing	2020-10-26
Jonathan	35b26eb	Added close button to app overlay.	2020-10-24
Liv Anne	9baed73	Added Security package	2020-10-22
Liv Anne	2481667	Update AccountRepository.cs Implemented passwordhashing	2020-10-22
Jonathan	4c7cd1c	Fixed some compiling issues (Repository.SQLIte is missing interop - added to bin...	2020-10-22
Jonathan	80ff975	Re-adding gitignore	2020-10-22
Jonathan	1597d86	Implemented ImGui.Net	2020-10-22
Jonathan	61d35fd	Desktop and mobile now both loads game internally. Added ScreenDensity to IP...	2020-10-20
Jonathan	725ee22	Enable / disable apps + specify app groups.	2020-10-15
Jonathan	4777917	Made ServerFunctions for service. To get user info and create groups.	2020-10-14
Jonathan	e95ead6	Group list added to AppBase	2020-10-14
Jonathan	0fa15e4	Client can now login and get app list and open / load app.	2020-10-14
Jonathan	1a0507d	New project JCIW.app - started making the login and app loading system	2020-10-13
Jonathan	b8cf054	Added add / remove groups.	2020-10-13
Jonathan	660144c	Add / remove user, add / remove user from group, update details and set new p...	2020-10-12
Jonathan	056d3ac	Started on group access	2020-10-12
Jonathan	9f199c4	Started on user admin (create user finished)	2020-10-10
Jonathan	38d6d8a	Added service logging.	2020-10-09
Jonathan	6bdc22d	Implemented service command function	2020-10-09

Author	Commit	Message	Date
Jonathan	72c98af	Added service delete.	2020-10-09
Jonathan	e1c3220	Fixed input for mobile and desktop. Upper / Lower case and space and backspace. ...	2020-10-08
Jonathan	31d99c4	Started mainview implementation of Admin client	2020-10-07
Jonathan	da60b04	Plugin database supoprt	2020-10-01
Jonathan	aff9ad0	Re-organized project a little.	2020-09-30
Jonathan	73429e5	Basic authorization implemented with database	2020-09-30
Jonathan	1ec3070	Fixed camera capture to file	2020-09-30
Liv Anne	94756b9	Added basic database functionality	2020-09-29
Jonathan	dc08e93	Started on platform specific functions. Added camera function. Returns string (url ...	2020-09-27
Jonathan	3f43c59	Started admin client login system and server account management.	2020-09-23
Jonathan	0d68574	Started on service code	2020-09-23
Jonathan	35e3d14	Started on login view admin client	2020-09-18
Jonathan	9213a49	Restored gitignore	2020-09-14
Jonathan	ca89084	Began networking code	2020-09-14
Jonathan	269eac1	Moving around / renaming things	2020-09-02
Jonathan	4e0c4f3	GUI Test with MonoGame	2020-08-31
Jonathan	3f0a894	Initial Commit Initial Commit	2020-08-25
Jonathan Øie	66a3cf5	README.md created online with Bitbucket	2020-08-25

Time and Travel wage app commits:

Author	Commit	Message	Date
Jonathan	dd35039	Updated wage app	2020-11-22
Jonathan	9088b8b	Fixed cursor placement	2020-11-18
Jonathan	ca4d3c9	implemented browse for desktop	2020-11-18
Jonathan	a294d9c	Updated image loading	2020-11-18
Jonathan	fecc897	Gui fixes	2020-11-15
Jonathan	af0356c	Added payroll.	2020-11-15
Jonathan	dcaac85	Fixed navigationbutton positions	2020-11-15
Jonathan	8fbca02	Fixed requesting and displaying of travel expenses.	2020-11-15
Jonathan	8138ac1	Fixed menu scaling on beginning	2020-11-15
Jonathan	61db3ca	Added Travelexpense "add new" function	2020-11-14
Jonathan	17dffb7	Fixed bug where row item in wage hour would not select because the style was instantly po...	2020-11-13
Jonathan	2a913d6	Finished register hours	2020-11-13
Jonathan	2b77a36	Started travel expense view.	2020-11-11
Jonathan	4d52e07	Further changes to start menu	2020-11-11
Jonathan	4bb02bf	Changes to menu style	2020-11-10
Jonathan	353f199	Wage hours view design	2020-11-09
Jonathan	6d7d4a1	Added travel and wage view and menu select on start.	2020-11-08
Jonathan	80b5061	Project start	2020-11-06
Jonathan Øie	cb6a3f4	Initial commit	2020-11-02

Liste over Jira issuer

T	Key	Summary	Assignee	Reporter	P	Status	Resolution	Created ↓	Updated
	JCIW-103	Fullføre conclusion	Jonathan Øie	Jonathan Øie		DONE	Done	11/Dec/20	13/Dec/20
	JCIW-102	Theory: Software architecture (tenker da på server-client og plugin)	Jonathan Øie	Jonathan Øie		DONE	Done	11/Dec/20	13/Dec/20
	JCIW-101	Fikse beskrivelsene på resultat (klassediagramma)	Jonathan Øie	Jonathan Øie		DONE	Done	11/Dec/20	13/Dec/20
	JCIW-100	Fikse sammendrag	Liv Anne Nyland	Jonathan Øie		IN PROGRESS	*Unresolved*	11/Dec/20	16/Dec/20
	JCIW-99	Fikse innledning	Jonathan Øie	Jonathan Øie		DONE	Done	11/Dec/20	16/Dec/20
	JCIW-98	Opprette statisk DocFX dokumentasjon for å legge ved innlevering.	Jonathan Øie	Jonathan Øie		DONE	Done	11/Dec/20	17/Dec/20
	JCIW-97	Opprette PowerPint i O365 og inviter Jonathan	Liv Anne Nyland	Jonathan Øie		DONE	Done	11/Dec/20	13/Dec/20
	JCIW-96	Rapport: Drøfting Evaluering av prosjektet	Liv Anne Nyland	Liv Anne Nyland		DONE	Done	11/Dec/20	11/Dec/20
	JCIW-95	JCIW API	Jonathan Øie	Jonathan Øie		DONE	Done	05/Dec/20	07/Dec/20
	JCIW-94	JCIW.App	Jonathan Øie	Jonathan Øie		DONE	Done	05/Dec/20	10/Dec/20
	JCIW-93	Rapport: Legge til retrospektive og møtenotat i vedlegg (som bilder) + GIT commit history	Liv Anne Nyland	Jonathan Øie		DONE	Done	01/Dec/20	07/Dec/20
	JCIW-92	Rapport: ImGui Implementasjon	Jonathan Øie	Jonathan Øie		DONE	Done	01/Dec/20	03/Dec/20
	JCIW-91	Rapport: JCIW Modulsystem	Jonathan Øie	Jonathan Øie		DONE	Done	01/Dec/20	06/Dec/20
	JCIW-90	Rapport: Teoretisk Grunnlag	Liv Anne Nyland	Jonathan Øie		DONE	Done	01/Dec/20	16/Dec/20
	JCIW-89	Rapport: Database	Liv Anne Nyland	Jonathan Øie		DONE	Done	01/Dec/20	10/Dec/20
	JCIW-88	Rapport: Server	Liv Anne Nyland	Jonathan Øie		DONE	Done	01/Dec/20	10/Dec/20
	JCIW-87	Rapport: Android klient	Jonathan Øie	Jonathan Øie		DONE	Done	01/Dec/20	02/Dec/20
	JCIW-86	Rapport: Desktop klient	Jonathan Øie	Jonathan Øie		DONE	Done	01/Dec/20	04/Dec/20
	JCIW-85	Rapport: Admin klient	Jonathan Øie	Jonathan Øie		DONE	Done	01/Dec/20	05/Dec/20
	JCIW-84	Endre på app view (legge til v forann versjon) legge ikon eller noe? GJøre knappene like store	*Unassigned*	Jonathan Øie		DONE	Done	27/Nov/20	30/Nov/20
	JCIW-83	Se på presentasjon	Jonathan Øie	Jonathan Øie		DONE	Done	27/Nov/20	16/Dec/20
	JCIW-82	Fikse app view listen	*Unassigned*	Jonathan Øie		DONE	Done	27/Nov/20	30/Nov/20
	JCIW-81	Opprette statisk dokumentasjon	Jonathan Øie	Jonathan Øie		DONE	Done	27/Nov/20	07/Dec/20
	JCIW-80	Legge inn ÆØÅ (og noen spesialteng til)	*Unassigned*	Jonathan Øie		DONE	Done	27/Nov/20	30/Nov/20
	JCIW-79	Finne ut hvor ÆØÅ problemet ligger (er det i JSON serializingen?)	*Unassigned*	Jonathan Øie		DONE	Done	27/Nov/20	30/Nov/20
	JCIW-78	Fiks Samsung reconnect problem	Liv Anne Nyland	Liv Anne Nyland		DONE	Done	21/Nov/20	23/Nov/20
	JCIW-77	Fylle ut kort på rapportoverskrift: Terminologi	Liv Anne Nyland	Jonathan Øie		DONE	Done	20/Nov/20	01/Dec/20
	JCIW-76	Fylle ut kort på rapportoverskrift: Resultater	Liv Anne Nyland	Jonathan Øie		DONE	Done	20/Nov/20	01/Dec/20
	JCIW-75	Fylle ut kort på rapportoverskrift: Teoretisk grunnlag	Liv Anne Nyland	Jonathan Øie		DONE	Done	20/Nov/20	01/Dec/20

JCIW-74	Fylle ut kort på rapportoverskrift: Konklusjon	Jonathan Øie	Jonathan Øie	DONE	Done	20/Nov/20	24/Nov/20
JCIW-73	Fylle ut kort på rapportoverskrift: Drøfting	Jonathan Øie	Jonathan Øie	DONE	Done	20/Nov/20	01/Dec/20
JCIW-72	Fylle ut kort på rapportoverskrift: Materialer og metoder	Jonathan Øie	Jonathan Øie	DONE	Done	20/Nov/20	25/Nov/20
JCIW-71	Fylle ut kort på rapportoverskrift: Innledning	Jonathan Øie	Jonathan Øie	DONE	Done	20/Nov/20	24/Nov/20
JCIW-70	Få prosjektet ut på Github	Jonathan Øie	Jonathan Øie	DONE	Done	13/Nov/20	22/Nov/20
JCIW-69	Skrive ei ramme og begynne og skrive litt på avsnitta på prosjektrapporten.	*Unassigned*	Jonathan Øie	DONE	Done	13/Nov/20	21/Nov/20
JCIW-68	Klargjøre Overleaf prosjektrapport prosjekt til skriving i desember. (Dropbox backup osv)	*Unassigned*	Jonathan Øie	DONE	Done	13/Nov/20	17/Nov/20
JCIW-67	Fixed session id / re-authorization on disconnect / auto-login	Jonathan Øie	Jonathan Øie	DONE	Done	13/Nov/20	13/Nov/20
JCIW-66	Fixed network disconnect handling	Jonathan Øie	Jonathan Øie	DONE	Done	13/Nov/20	13/Nov/20
JCIW-65	Fixed keyboard function on mobile (show only when textbox is focused)	Jonathan Øie	Jonathan Øie	DONE	Done	13/Nov/20	13/Nov/20
JCIW-64	Finished Android icon and desktop icon	Jonathan Øie	Jonathan Øie	DONE	Done	13/Nov/20	13/Nov/20
JCIW-63	Finished client login and overlay design.	Jonathan Øie	Jonathan Øie	DONE	Done	13/Nov/20	13/Nov/20
JCIW-62	Finished admin client design	Jonathan Øie	Jonathan Øie	DONE	Done	13/Nov/20	13/Nov/20
JCIW-61	Opprette view: Payroll view (hente ut timeføring og travel expense for måneden. - godkjenne og se på bildene av kvitteringene og sånt - desktop only)	*Unassigned*	Jonathan Øie	DONE	Done	02/Nov/20	15/Nov/20
JCIW-60	Opprette view: (fortsettelse på travel expenses)	Jonathan Øie	Jonathan Øie	DONE	Done	02/Nov/20	15/Nov/20
JCIW-59	Opprette view: Add travel expenses	Liv Anne Nyland	Jonathan Øie	DONE	Done	02/Nov/20	13/Nov/20
JCIW-58	Opprette view: Startbilde på app (Register hours, register travel expenses og Payroll (for lønnsansatte))	Liv Anne Nyland	Jonathan Øie	DONE	Done	02/Nov/20	13/Nov/20
JCIW-57	Opprette view: Register hours (Wage hours)	Jonathan Øie	Jonathan Øie	DONE	Done	02/Nov/20	13/Nov/20
JCIW-56	Admin klient enter login	Jonathan Øie	Jonathan Øie	DONE	Done	29/Oct/20	03/Nov/20
JCIW-55	Fikse admin klient custom IP.	Jonathan Øie	Jonathan Øie	DONE	Done	29/Oct/20	03/Nov/20
JCIW-54	Fikse mobil input (keyboard må dukke opp når brukeren trykker på et input)	Jonathan Øie	Jonathan Øie	DONE	Done	26/Oct/20	28/Oct/20

JCIW-53	Implementere session funksjonalitet (autentisere med session fra klient)	Liv Anne Nyland	Jonathan Øie		DONE	Done	26/Oct/20	29/Oct/20
JCIW-52	Passordhashing	Liv Anne Nyland	Liv Anne Nyland		DONE	Done	20/Oct/20	22/Oct/20
JCIW-51	Close button on apps (to go back to app list)	*Unassigned*	Jonathan Øie		DONE	Done	19/Oct/20	24/Oct/20
JCIW-50	Control: Popup-panel?	*Unassigned*	Jonathan Øie		DONE	Done	19/Oct/20	26/Oct/20
JCIW-49	Control: Grid spreadsheet style (To view data)	*Unassigned*	Jonathan Øie		DONE	Done	19/Oct/20	22/Oct/20
JCIW-48	Control: Listbox (select item from list - listitems should be taggable)	*Unassigned*	Jonathan Øie		DONE	Done	19/Oct/20	22/Oct/20
JCIW-47	Control: Slide view panel	*Unassigned*	Jonathan Øie		DONE	Done	19/Oct/20	26/Oct/20
JCIW-46	Control: ImageView / Image	*Unassigned*	Jonathan Øie		DONE	Done	19/Oct/20	22/Oct/20
JCIW-45	Control: Calendar	Jonathan Øie	Jonathan Øie		DONE	Done	19/Oct/20	13/Nov/20
JCIW-44	Control: Textbox	Jonathan Øie	Jonathan Øie		DONE	Done	19/Oct/20	22/Oct/20
JCIW-43	Control: Button	*Unassigned*	Jonathan Øie		DONE	Done	19/Oct/20	22/Oct/20
JCIW-42	Begynne på VS template	Jonathan Øie	Jonathan Øie		DONE	Done	16/Oct/20	17/Nov/20
JCIW-41	Lage logo	Liv Anne Nyland	Jonathan Øie		DONE	Done	16/Oct/20	18/Oct/20
JCIW-40	Designe prototype av timeføringsapp	*Unassigned*	Jonathan Øie		DONE	Done	16/Oct/20	18/Oct/20
JCIW-39	Dokumentere Client-Admin prosjektet	Jonathan Øie	Jonathan Øie		DONE	Done	16/Oct/20	22/Nov/20
JCIW-38	Dokumentere Client-Desktop prosjektet	Jonathan Øie	Jonathan Øie		DONE	Done	16/Oct/20	21/Nov/20
JCIW-37	Dokumentere Client-Mobile.Android prosjektet	Jonathan Øie	Jonathan Øie		DONE	Done	16/Oct/20	21/Nov/20
JCIW-36	Dokumentere Client-Networking.Android prosjektet	Jonathan Øie	Jonathan Øie		DONE	Done	16/Oct/20	21/Nov/20
JCIW-35	Dokumentere Client-Networking.Desktop prosjektet	Jonathan Øie	Jonathan Øie		DONE	Done	16/Oct/20	21/Nov/20
JCIW-34	Dokumentere GraphicsEngine prosjektet	Jonathan Øie	Jonathan Øie		DONE	Done	16/Oct/20	21/Nov/20
JCIW-32	Dokumentere Repository prosjektet	Liv Anne Nyland	Jonathan Øie		DONE	Done	16/Oct/20	22/Nov/20
JCIW-31	Dokumentere Repository.SQLite prosjektet	Liv Anne Nyland	Jonathan Øie		DONE	Done	16/Oct/20	22/Nov/20
JCIW-30	Dokumentere Server prosjektet	Jonathan Øie	Jonathan Øie		DONE	Done	16/Oct/20	22/Nov/20
JCIW-29	Dokumentere JCIW.App prosjektet	Jonathan Øie	Jonathan Øie		DONE	Done	16/Oct/20	21/Nov/20
JCIW-28	Dokumentere JCIW.Data prosjektet	*Unassigned*	Jonathan Øie		DONE	Done	16/Oct/20	26/Oct/20
JCIW-27	Dokumentere JCIW.Module prosjektet	Jonathan Øie	Jonathan Øie		DONE	Done	16/Oct/20	26/Oct/20
JCIW-26	Dokumentere Networking.Data prosjektet	Jonathan Øie	Jonathan Øie		DONE	Done	16/Oct/20	21/Nov/20
JCIW-25	Dokumentere JCIW prosjektet (5 klasser)	*Unassigned*	Jonathan Øie		DONE	Done	16/Oct/20	26/Oct/20
JCIW-24	Undersøke alternativer til GUI. Enten alternativer til MonoGame løsningen eller utvidelser til MonoGame (https://github.com/aloisdeniel/awesome-monogame#user-interfaces)	Jonathan Øie	Jonathan Øie		DONE	Done	16/Oct/20	22/Oct/20
JCIW-23	Grafikkmotor: Designe og utvikle løsning for skalering og plassering av kontroller (For eksempel et gridsystem som i WPF eller noe sånn som docking i Windows forms)	Jonathan Øie	Jonathan Øie		DONE	Done	16/Oct/20	22/Oct/20

JCIW-22	Fikse en slags AdminManager type overklasse hvor disse adminspesifikke tingene blir gjort.	Liv Anne Nyland	Jonathan Øie	DONE	Done	02/Oct/20	12/Oct/20
JCIW-21	Plugin utvikler må kunne definere kommunikasjonen (hva skal de ulike kommandoene gjøre på den spesifikke pluginen?)	Jonathan Øie	Jonathan Øie	DONE	Done	02/Oct/20	09/Oct/20
JCIW-20	Må kunne kommunisere med service fra adminklient	Jonathan Øie	Jonathan Øie	DONE	Done	02/Oct/20	09/Oct/20
JCIW-19	Designe group access / user admin vinduer (Windows Forms?)	Jonathan Øie	Jonathan Øie	DONE	Done	02/Oct/20	12/Oct/20
JCIW-18	Designe MainView	Jonathan Øie	Jonathan Øie	DONE	Done	02/Oct/20	07/Oct/20
JCIW-17	Platformspesifikke funksjoner -> Implementere "browse" for å finne filer på filsystem for desktop og android	Jonathan Øie	Jonathan Øie	DONE	Done	02/Oct/20	18/Nov/20
JCIW-16	Implementere autorisering på serversside (GUID må kobles opp mot brukeren. Denne sendes på hver request og vi kan da verifisere hvem som kommer med requests).	Liv Anne Nyland	Jonathan Øie	DONE	Done	02/Oct/20	12/Oct/20
JCIW-15	Designer nettverk til å pulle fra server hvert x sekund istedenfor å pushe hele tiden fra server.	Jonathan Øie	Jonathan Øie	DONE	Done	02/Oct/20	10/Oct/20
JCIW-14	Service må kunne loggføre sånn at admin kan lese meldingen i konsoll. Hvordan håndtere dette?	Jonathan Øie	Jonathan Øie	DONE	Done	02/Oct/20	09/Oct/20
JCIW-13	Modulhåndtering -> Klient må kunne laste opp modul til server	Jonathan Øie	Jonathan Øie	DONE	Done	02/Oct/20	05/Oct/20
JCIW-12	Modulhåndtering på server side (database - hvilke plugins er skrudd på og hva heter de, versjon osv).	Jonathan Øie	Jonathan Øie	DONE	Done	02/Oct/20	07/Oct/20
JCIW-11	En vanlig bruker må ikke kunne logge seg inn på Adminkonsoll. Hvordan løse dette? Grupperettigheter?	Liv Anne Nyland	Jonathan Øie	DONE	Done	02/Oct/20	12/Oct/20
JCIW-10	Applikasjoner (plugins) må kunne begrenses etter gruppetilgang på admin siden.	Liv Anne Nyland	Jonathan Øie	DONE	Done	02/Oct/20	15/Oct/20
JCIW-9	Utvikler av plugins og tjenester må kunne opprette gruppe	Liv Anne Nyland	Jonathan Øie	DONE	Done	02/Oct/20	14/Oct/20
JCIW-8	Utvikler av plugins og tjenester må kunne hente ut brukerinformasjon. Identifikasjon og hvilke grupper?	Jonathan Øie	Jonathan Øie	DONE	Done	02/Oct/20	14/Oct/20
JCIW-7	Lage til group access funksjonalitet. En admin skal kunne logge inn på adminkonsoll og opprette grupper og legge til brukere	Liv Anne Nyland	Jonathan Øie	DONE	Done	02/Oct/20	13/Oct/20
JCIW-6	Fikse input til å støtte backspace, space og store/små bokstaver på mobil og desktop	Jonathan Øie	Jonathan Øie	DONE	Done	27/Sep/20	08/Oct/20
JCIW-5	Starte dokumentere allerede skrevet kode	Jonathan Øie	Jonathan Øie	DONE	Done	18/Sep/20	18/Oct/20
JCIW-4	Undersøke hvilke alternativer vi har for sammerbeid på bachelor-rapport (Overleaf?)	*Unassigned*	Jonathan Øie	DONE	Done	18/Sep/20	17/Oct/20
JCIW-3	Kode datalagring til modulsystem	Jonathan Øie	Jonathan Øie	DONE	Done	18/Sep/20	01/Oct/20
JCIW-2	Kode grunnleggende funksjonalitet for tjenester på server	Jonathan Øie	Jonathan Øie	DONE	Done	18/Sep/20	23/Sep/20
JCIW-1	Kode grunnleggende databasefunksjonalitet med repository pattern til server	Liv Anne Nyland	Jonathan Øie	DONE	Done	18/Sep/20	01/Oct/20