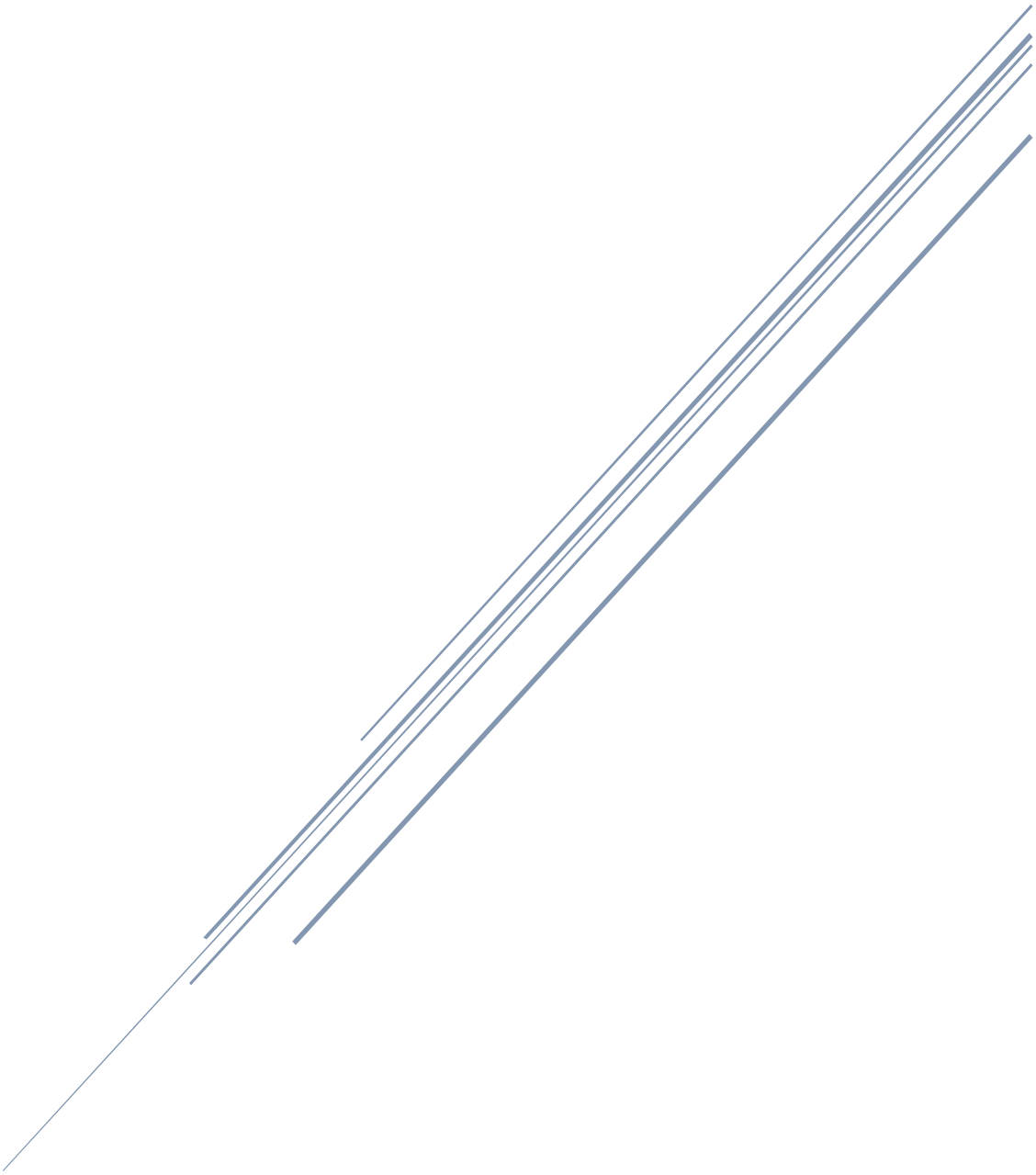
PROJECT PROPOSAL

Clocker



Howest 2021-2022

Trending Topics In Software Development

Contents

Inleiding

Clocker is een planning applicatie dat bedoeld is om het telewerk en on-site werk zo gemakkelijk mogelijk bij te houden. Wij kwamen met dit idee omdat Hybrid working de nieuwe norm word maar qua planning en weten wie waar werkt er nog niet veel dedicated programs voor zijn.

In Clocker zijn er twee soorten gebruikers: de Manager en de Werknemers, De werknemers zijn vrij om hun weekplanning in te vullen zodat zowel de manager als de andere werknemers weten wie wanneer werkt. Als je een werkmoment inplant dan ga je moeten aanduiden of je gaat telewerken of on-site gaat werken. Hierdoor kunnen anderen direct weten wanneer je in je bureau aanwezig zult zijn. En als extra ga je ook de kans kunnen zien dat iemand komt werken, deze kans wordt berekend door behulp van een AI.

Het doel is dus om een overzichtelijk planning en in- en uitklok systeem te maken zodat er minder verwarring zou kunnen ontstaan tussen telewerken en on-site werken.

Functionalities

Deliverables

Front-end PWA

- employer panel
- badge system
- Ticket system (wie werkt aan wat)
- Planning system (wie werkt wanneer)
- settings page
- login page
- no internet connection page

Front-end windows

- employer panel
- badge system
- Ticket system (wie werkt aan wat)
- Planning system (wie werkt wanneer)
- settings page
- login page

Back-end

- badge system back
- ticket system backend
- planning system backend
- employer panel backend
- SRP

Database

- badge system
- ticket system
- planning system
- employee/employer system

AI

- Data API
- AI API
- Web panel

Top Tasks

- In- en uitklokken
- Planning aanpassen en opvolgen
- Kans berekenen en zien dat iemand werkt
- Gewerkte en te werken uren zien
- Werknemers toevoegen en verwijderen

User Stories

- Willem wilt kunnen in- en uitloggen in het systeem zodat hij er toegang tot heeft.
- Willem wilt kunnen in- en uitklokken zodat zijn baas weet wanneer hij aan het werken.
- Willem wilt zijn planning van de komende week zien zodat hij weet wanneer hij moet inklokken.
- Willem wilt voor de volgende week zijn planning kunnen invullen zodat zijn collega's zien wanneer hij zal werken
- Willem wil kunnen zien wanneer zijn collega's werken zodat hij weet wanneer er nog mensen nodig zijn om te werken
- Willem wilt de hoeveelheid ingeplande uren en de minimale hoeveelheid te werken uren zien voor elke week zodat hij weet hoeveel uur hij nog moet inplannen of hoeveel uren teveel hij werkt
- Willem wil via zijn mobiele toestellen ook kunnen inloggen zodat hij niet altijd een computer nodig heeft om de planning te raadplegen
- Zowel Willem als zijn manager willen kunnen zien wat de kans is dat anderen zullen komen werken op de ingeplande uren
- Willem zijn manager wilt alle personeelsleden hun planning kunnen zien zodat ze weten wanneer ze werken
- Willem zijn manager wilt nieuwe gebruikers aan zijn departement kunnen toevoegen zodat hij nieuw werknemers kan aannemen
- Willem zijn manager wilt oude gebruikers kunnen verwijderen uit zijn departement zodat hij werknemers kan ontslaan of transfereren naar een ander departement
- Willem wil ook op zijn computer een programma hebben voor toegang tot het systeem zodat hij gemakkelijke toegang heeft tot de planningen zonder weblinks en webadressen

Use cases

- moet veilig kunnen inloggen/uitloggen
- moet kunnen inklokken/uitklokken (met locatie)
- moet planning kunnen zien
- moet planning kunnen wijzigen
- moet kunnen zien wie ingeklokt is
- moet kunnen zien wie zal inklokken (ai)
- moet werk quotas kunnen zien
- moet verlofuren kunnen zien
- moet werklocaties kunnen zien

Trending technologies

back-end

- ### rust
  Rust is known for its speed and memory efficiency and reliability, as well as extensive documentation. Besides this it has recently been gaining popularity for it's debuggability in memory critical application over competitors such a Go, reaching the top 20 most loved languages in Tiobe popularity index, as well as most popular language on stackoverflow for the fifth consecutive year.

- ### Actix web
  When searching for a backend framework for rust, 3 main frameworks popped up: Rocket, Actix and yew. Most frameworks are relatively new to the scene and lack reliability and stability, so even though Rocket is more user friendly, we opted to use actix as it is a combination of most worked out and stable.

front-end

- ### pwa
  We chose to develop a pwa for it's easy cross platform form factor. This way we don't have to diverge into different applications for different platforms even though functionality would be the same.

- ### Electronjs app
  To deploy our web experience in native environments, we will be using the Electronjs framework. This allows us to use a lot of the pwa codebase in a chromium shell, allowing the native experience to intertwine with our existing web infrastructure.

- ### bootstrap
  We will use bootstrap as css framework for it's quick and effective use. With it we can easily set up our pages without having to spend a huge amount of time on this. Thuss focusing more on other things.

- ### angular
  There are quite a few web frameworks in the run right now, in contrast to our backend most of these are tried and tested. Angular.js has been around for quite a while and started to become outdated until it was rewritten from the ground up using typescript. We opted for a typescript based framework as it was quite new for us yet very relevant, this in combination with the maturity of angular made for a simple choice.

communicatie front/back

- API
- SRP

  The SRP (secure remote password) protocol is a modern way of using passwords on web applications; it prevents the risk of a password being intercepted on it's way to a server by never sending it in the first place. A simple way of explaining it is that you don't show your password to the server, you simply prove that you know it. This trough a public private key process. We will be using existing libraries to implement this.

  securing data

- JWT

database

- Datahike

  At first we were planning on using Datomic, it's uniqueness to conventional databases we have used before made it quite appealing, however it turned out it is made purely for AWS which we will not use. As such we looked for an azure friendly alternative, landing us on Datahike. An easy to use data querying framework based on Clojure.

hosting

- azure

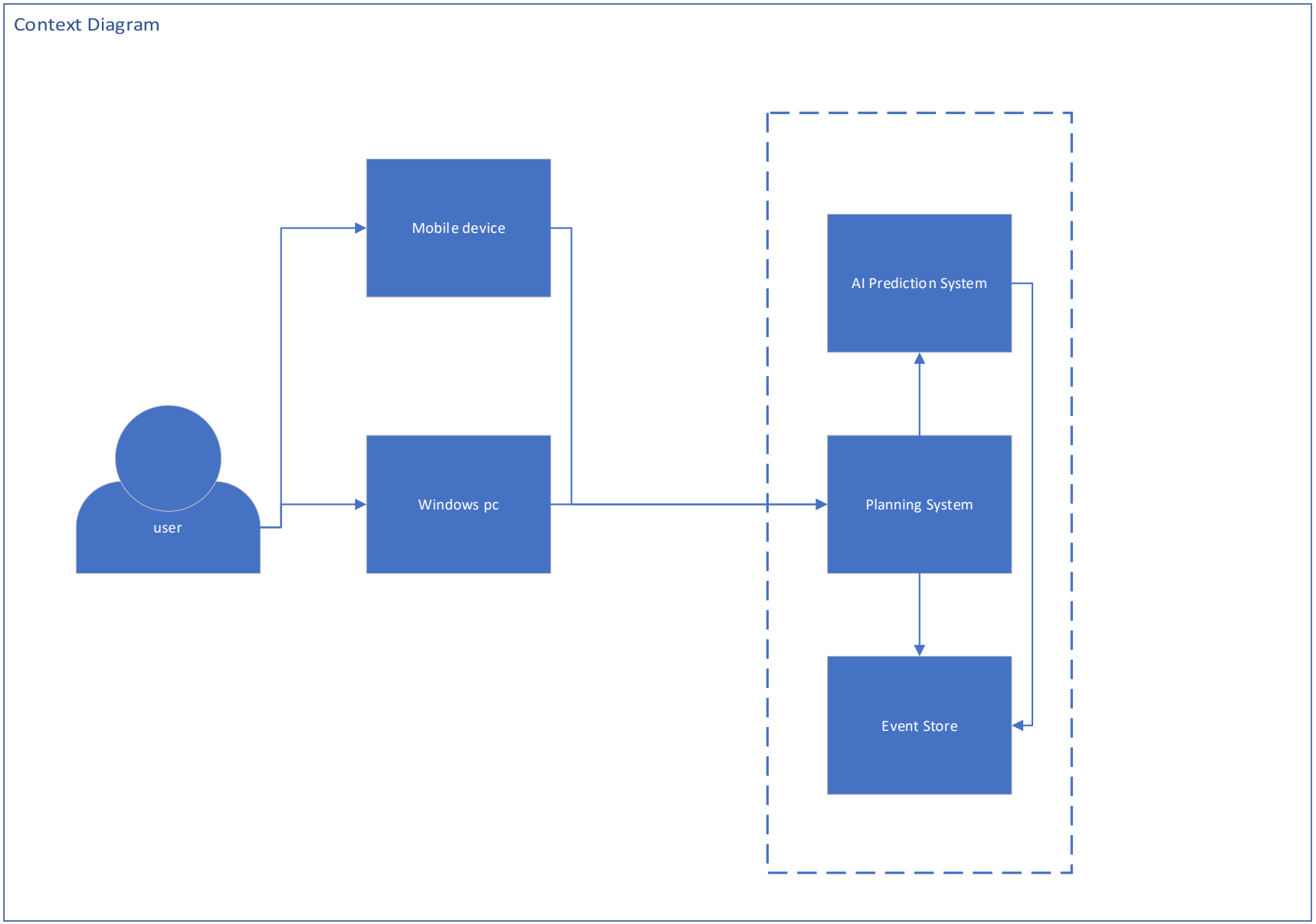
  For hosting our application and it's related deployables we chose Azure, simply for it's wide range of pre-made environments suited for any type of application. Besides this it has a lot of useful monitoring tools that we can use to measure, debug our application.

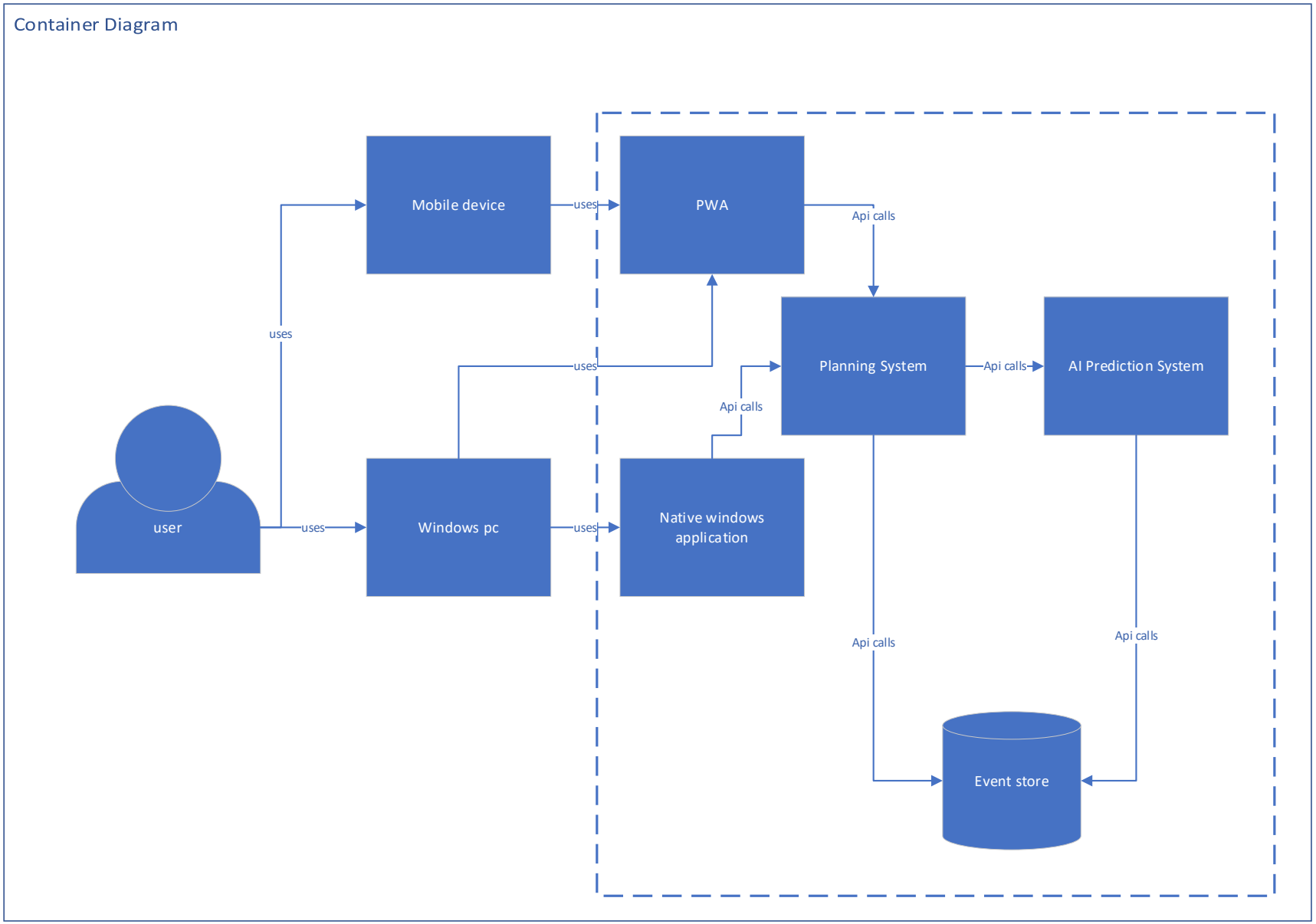
- Github

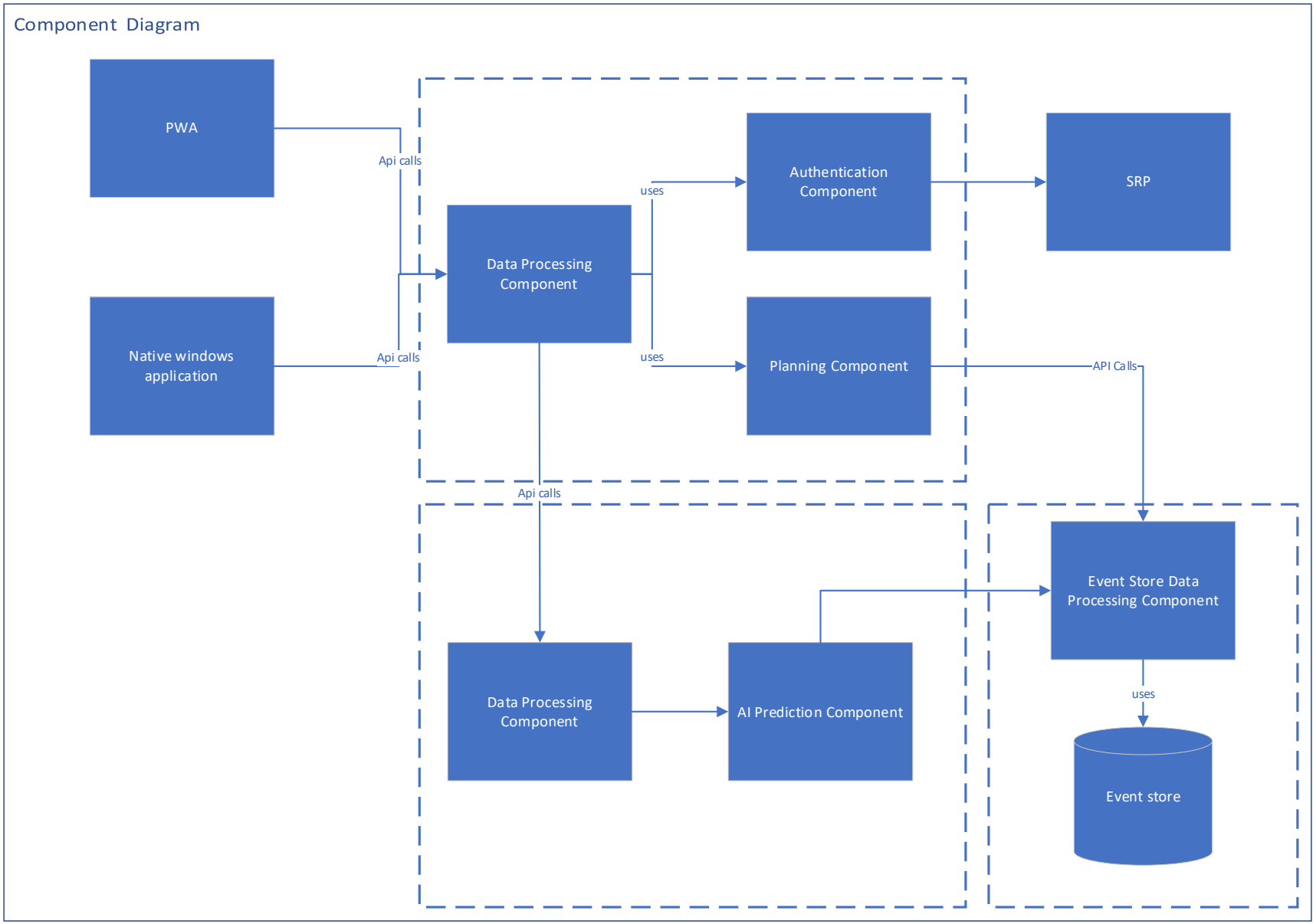
  We chose github as our preferred remote repository for it's easy integration with azure through github actions, allowing us to easily build and deploy our project on the go.

Diagrams

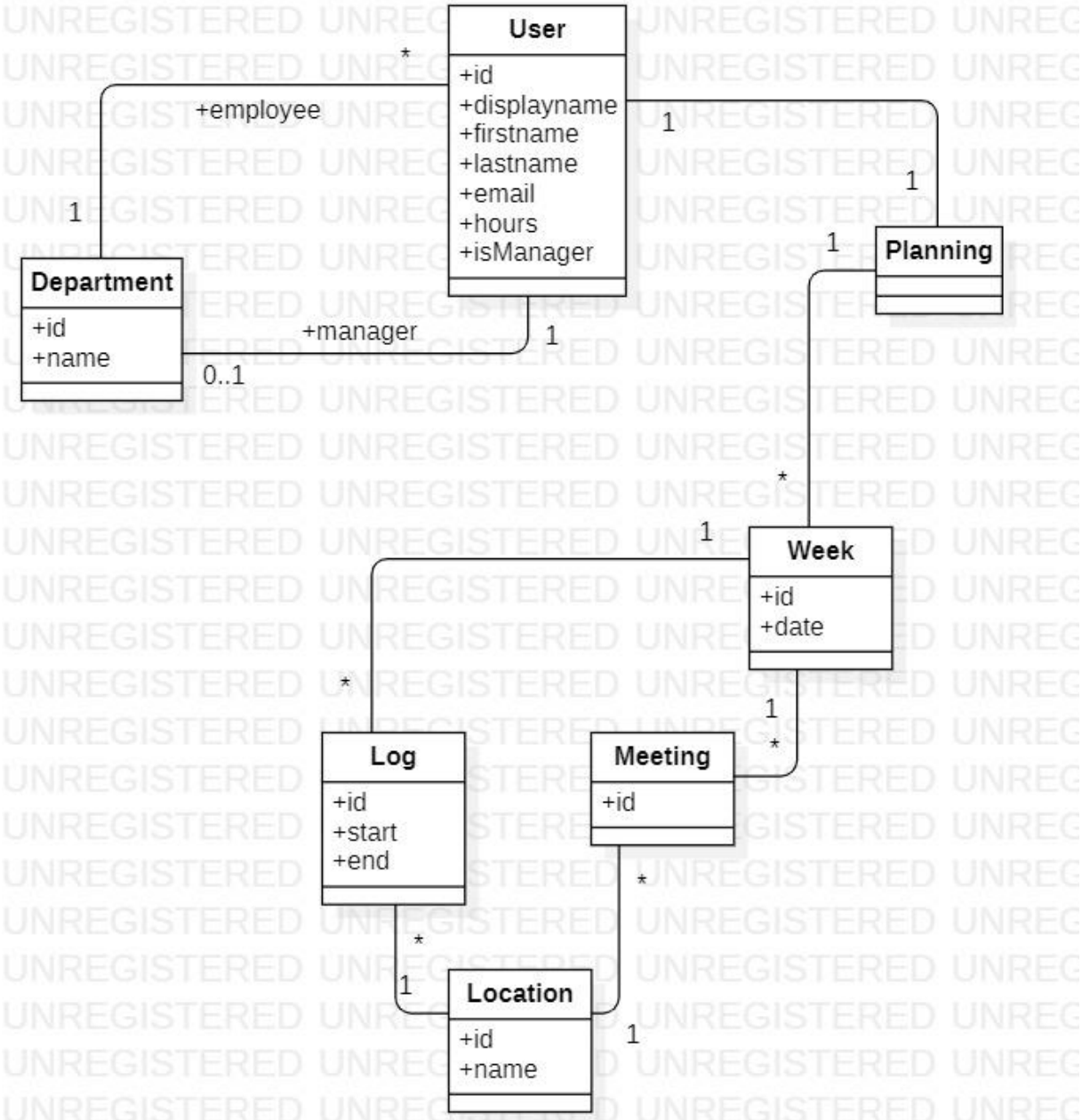
C4

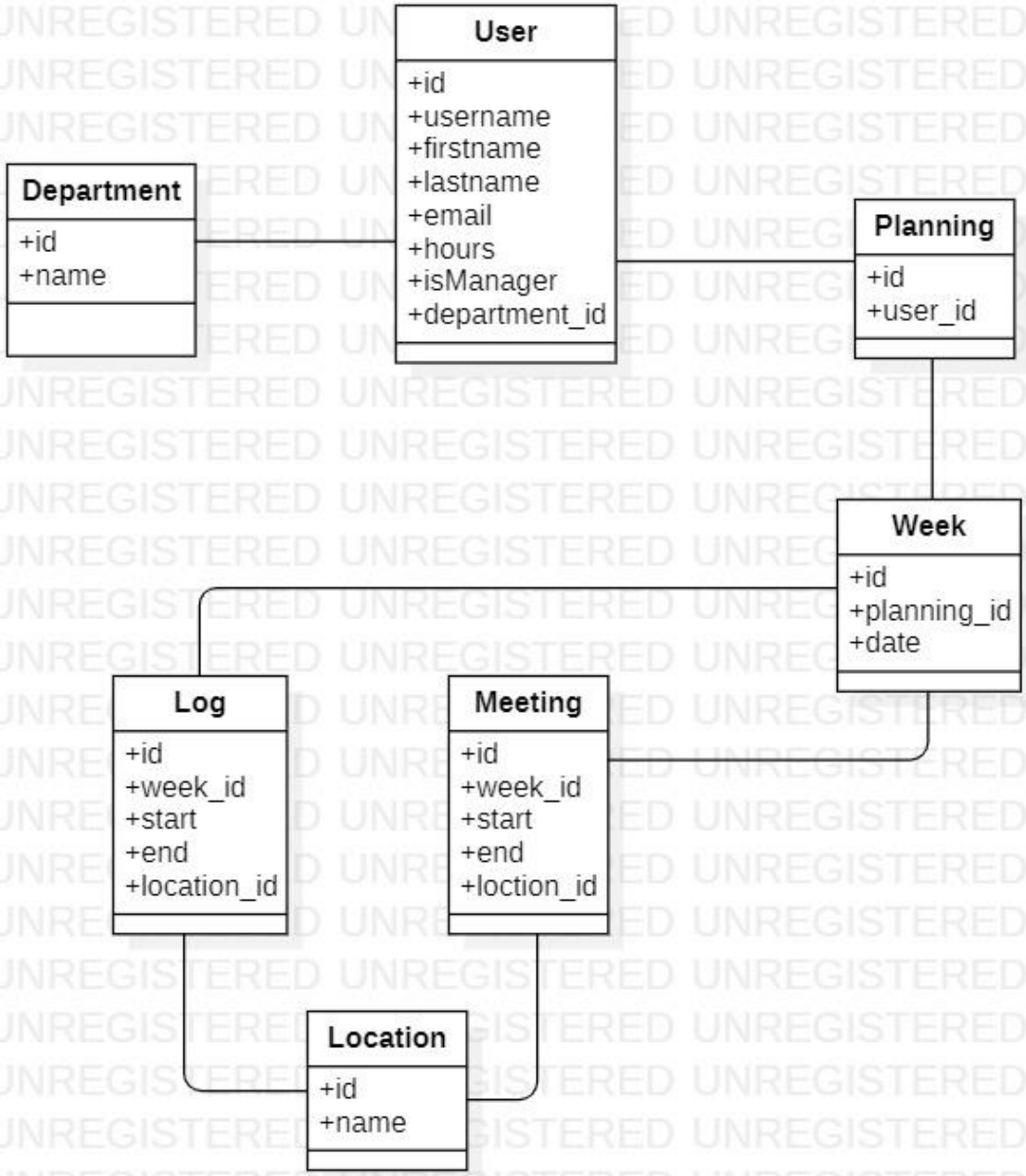

Context Diagram
Mobile device
AI Prediction System
user
Windows pc
Planning System
Event Store


Container Diagram
Mobile device
uses
PWA
Api calls
uses
Planning System
Api calls
AI Prediction System
uses
Api calls
user
uses
Windows pc
uses
Native windows application
Api calls
Api calls
Event store


Component Diagram
PWA
Native windows application
Api calls
Api calls
Data Processing Component
uses
uses
Authentication Component
Planning Component
SRP
API Calls
Api calls
Data Processing Component
AI Prediction Component
Event Store Data Processing Component
uses
Event store

DB


User
+id
+displayname
+firstname
+lastname
+email
+hours
+isManager
+employee
*
1
1
1
Planning
Department
+id
+name
+manager
1
0..1
*
1
Week
+id
+date
*
1
*
Log
+id
+start
+end
Meeting
+id
*
*
1
Location
+id
+name
1


User
+id
+username
+firstname
+lastname
+email
+hours
+isManager
+department_id
Department
+id
+name
Planning
+id
+user_id
Week
+id
+planning_id
+date
Log
+id
+week_id
+start
+end
+location_id
Meeting
+id
+week_id
+start
+end
+loction_id
Location
+id
+name