Høgskulen på Vestlandet

Fakultet for teknologi, miljø- og samfunnsvitenskap

Institutt for datateknologi, elektroteknologi og realfag

Mobilapplikasjon for effektiv loggføring og kontroll av arbeidstimer

Mobile application for effective logging and monitoring of work hours

Systemdokumentasjon

Versjon <1.0>

REVISJONSHISTORIE

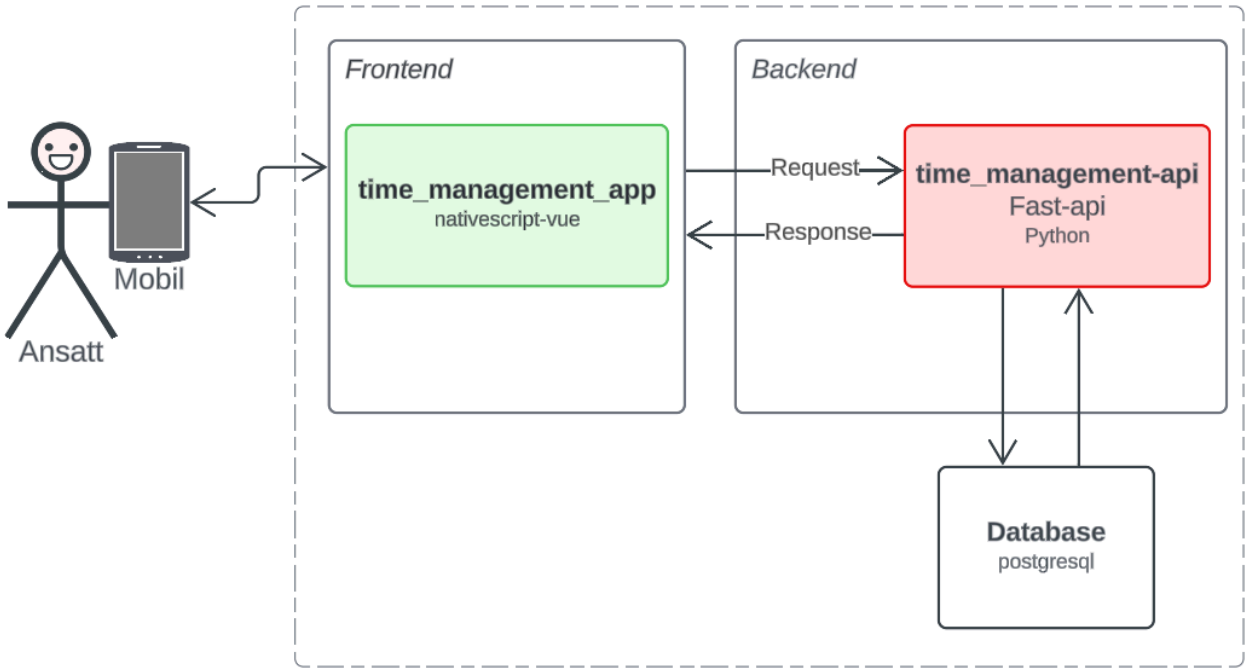
Dato	Versjon	Beskrivelse	Forfatter
03/05/2024	1.0	Lagt til arkitektur	Susanne Å. Løtvedt

INNHOLDSFORTEGNELSE

1 INNLEDNING

Systemdokumentet er skrevet for å gi en omfattende oversikt over utviklingen i prosjektet og hjelpe gruppen å forstå oppsettet i applikasjonen via arkitekturmodell og klassediagram. Gruppen har tatt med viktige deler fra koden for å vise de viktigste endepunktene, biblioteker og plugins som er brukt i prosjektet. Gruppen dokumenterer også hvordan mobilapplikasjonen blir testet, installert og kjørt under utviklingen.

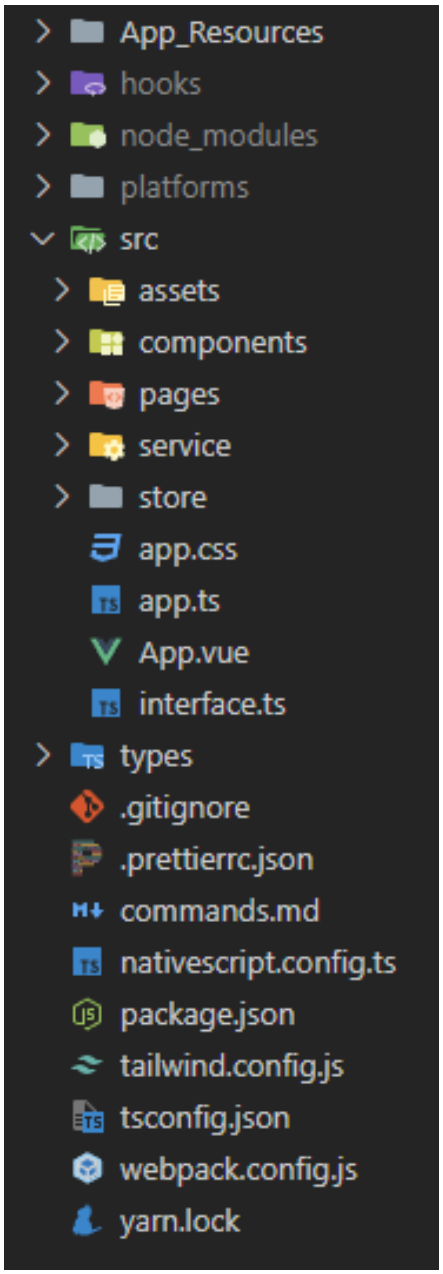
2 ARKITEKTUR



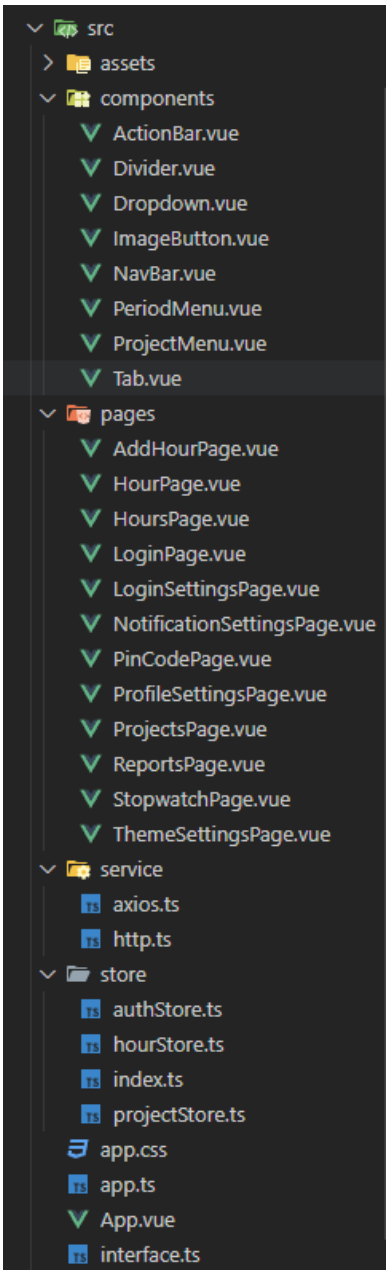
Figur 2.1 Overordnet arkitektur

Figuren ovenfor viser de ulike delene av systemet. Systemer som er uendret er markert rødt, og endringer er systemer som er lagt til er markert med grønn.

3 PROSJEKTSTRUKTUR



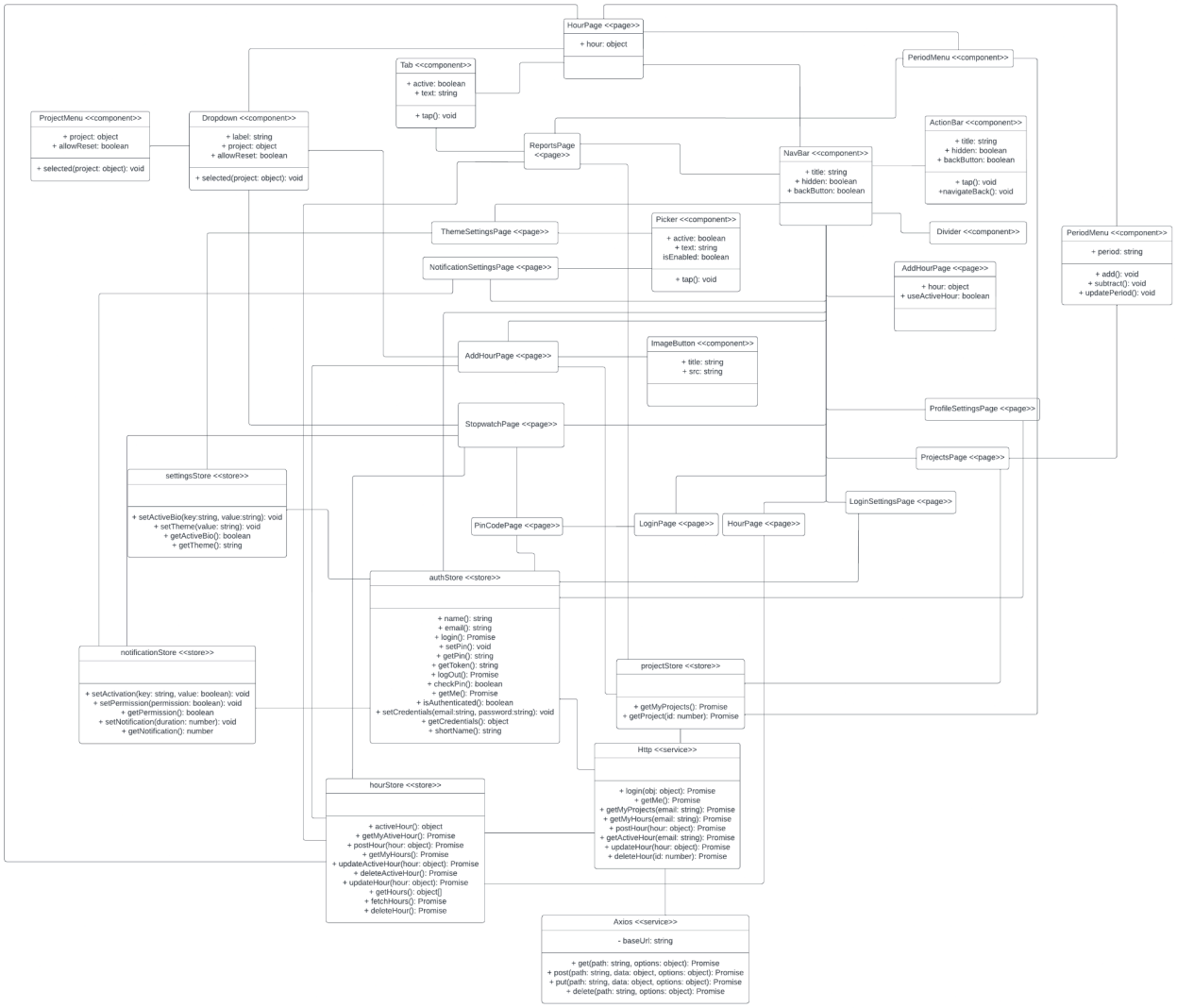
Figur 3.1 Overordnet prosjektstruktur



Figur 3.2 Prosjektstruktur med fokus på components, pages, service & store

I mobilappen blir *nativescript-vue* brukt, som er en variant av *nativescript*, men som inkluderer funksjonalitet fra *vue*. I tillegg brukes flere "*@nativescript-community/…*" og "*@nativescript/…*" biblioteker for å utvide nativescript med ekstra funksjonalitet. "nativescript-modal-datepicker" brukes for å vise en brukervennlig dato- eller tid-velger, og *dayjs* brukes for å håndtere dato-objekter i javascript.

4 KLASSEDIAGRAM



Figur 4.1

Diagrammet ovenfor viser den fullstendige arkitekturen av systemet. Fordi strukturen i vue, baserer seg på gjenbruk av komponenter, er det som regel mange koblinger til alle elementene. Et eksempel på dette er *NavBar*, denne komponenten brukes av alle sidene, og vil dermed ha mange koblinger til seg. Et annet eksempel er *autoStore*, denne blir brukt for å autentisere en bruker, samt å hente informasjon om den innloggede brukeren. Ettersom en bruker alltid må være innlogget for å kunne bruke tjenesten, er denne store'en brukt av de fleste av komponentene og sidene.

5 SERVER-TJENESTER

auth/login

Dette endepunktet brukes for å validere at brukernavn og passord stemmer med det som er registrert i databasen. I vår kode brukes denne når man først logger inn i systemet, for å bekrefte brukernavn og passord, og senere hvis man logger inn med biometri eller pin-kode. Etter at en bruker har blitt validert mottas en token som videre kan verifisere identiteten til en bruker

auth/me

Dette endepunktet brukes for å hente detaljer fra en innlogget bruker. Dette endepunktet krever at man har en gyldig token, og i vår kode brukes den til å hente fornavn og etternavn fra en bruker.

hours/

Dette endepunktet brukes for å opprette en tom time i databasen, en tom time er definert som en time uten slutt-tidspunkt. Dette endepunktet krever at man har en gyldig token.

hours/employee/{email}

Dette endepunktet brukes for å hente en liste av alle timer for en spesifikk epost, den kan også brukes med *?active_hour=true* for å bare returnere timer uten sluttidspunkt. Dette endepunktet krever at man har en gyldig token.

projects/employee/{email}

Dette endepunktet brukes for å hente en liste over alle prosjekter som en bruker er tilknyttet. Dette endepunktet krever at man har en gyldig token.

hours/{id}

Dette endepunktet brukes for å hente detaljer om en spesifikk time. I vår kode brukes dette endepunktet for å hente alle detaljene om en time, når kun denne timen skal vises. Dette endepunktet krever at man har en gyldig token.

6 SIKKERHET

Vår løsning gjenbruker backend apiet fra en tidligere bachelor, det er derfor ingen endring i hvordan passord blir hashet, overført og lagret. I mobilappen blir brukernavn og passord sendt som FormData på tilsvarende måte som i web-løsningen. I mobilappen er også en pin kode implementert for enklere innlogging, på grunn av vanskeligheter med kryptering av en kort tallkode, blir denne kun lagret i minne på mobilenheten. Ved innlogging med pin-kode blir denne verdien sammenlignet med pin-kode fra minne.

Ved innlogging i mobilappen blir det også generert en token fra den eksisterende backend-koden, denne token blir videre brukt for å identifisere en bruker i systemet i alle senere forespørsler til serveren. Etter et gitt antall timer (definert i backend), blir den tokenen ugyldiggjort, og krever at bruker logger inn på nytt med pin-kode eller biometri. Ved innlogging av biometri blir ingen data lagret i appen.

SQL-injection og cross-site scripting er to risikoer som må håndteres. Ettersom gruppens løsning er en app, og ikke en web-løsning, er cross-site scripting blitt ignorert ettersom dette krever en web-løsning for å bruke. SQL injection blir håndtert backend, og ettersom det blir brukt en eksisterende løsning, er det ikke en ting som blir prioritert her.

7 INSTALLASJON OG KJØRING

I dette kapittelet har vi fokusert på hvordan programmet kan kjøres på en Android-telefon, prosjektet er bygd i et rammeverk for kryssplattform-utvikling, så en lignende fremgangsmåte fungerer også for iOS.

For å kjøre applikasjonen kreves det at NodeJS er installert (https://nodejs.org/en). Yarn brukes i prosjektet for å håndtere installasjon av andre pakker (https://classic.yarnpkg.com/en/docs/install). Deretter kan prosjektet klones med følgende kommando *git clone https://BRUKERNAVN@bitbucket.org/timekontroll/time_management_app.git*. Deretter må pakkene som kreves for å kjøre prosjektet installeres, dette gjøres ved å kjøre *yarn install* i prosjektmappen.

7.1 Installasjon av nativescript

Applikasjonen bruker nativescript i bunnen, og krever derfor at et par verktøy er riktig konfigurert, før koden er klar for kompilering (https://docs.nativescript.org/setup/windows)

7.2 Kompilering

For kjøring av applikasjonen, kreves kompilering av en apk-fil, men for å gjøre dette kreves en keystore-fil. For generering av en keystore fil kan følgende kommando brukes, men ALIAS erstattes med ønsket verdi. *keytool -genkey -v -keystore key.keystore -alias ALIAS -keyalg RSA -keysize 2048 -validity 10000*. Deretter vil du bli spurt om et passord, dette passordet brukes i et senere steg. Deretter produserer kommandoen mange spørsmål for generering av en keystore fil, som for det meste kan dette steget hoppes over ved å klikke på enter, men i et av de siste spørsmålene står det [no] i stedet for [unknown], her må *yes* skrives i stedet. Deretter kan følgende kommando brukes, men erstatte ALIAS og PASSORD med verdier fra forrige kommando. *yarn ns build android –release –key-store-path key.keystore –key-store-password PASSWORD –key-store-alias ALIAS –key-store-alias-password PASSWORD*.

8 DOKUMENTASJON AV KILDEKODE

En kopi av kildekoden finnes her: https://github.com/mnervik/dat191-exam

brukernavn: exam-2024

passord: exam2024

Gruppen har brukt BiometricAuth() for å la brukeren logge inn via biometri. I koden under blir det lagd et nytt biometriAuth() objekt og deretter kalles available() metoden som sjekker om biometri er tilgjengelig på enheten til brukeren. Hvis biometri-innlogging er tilgjengelig på enheten, blir verdien til allowBiometrics lik "true" og programmet vil prøve å logge inn brukeren med ansiktsgjenkjenning eller fingeravtrykk. Kommentarene er de forskjellige måtene å logge inn på basert på hvilken enhet brukeren har.

```
18  const biometricAuth = new BiometricAuth()
19  biometricAuth.available().then(result => {
20    //ios     -> .face and .touch
21    //android -> .biometrics (face and touch)
22    //.any    -> if device has pin code
23    if (result.biometrics) {
24      allowBiometrics.value = true
25    }
26  })
27
```

Deretter vil koden under vente på at brukeren skal logge inn med biometri og setter resten av programmet på vent til brukeren har logget seg inn.

```
58          const result = await biometricAuth.verifyBiometric({
59            title: 'Logg inn med biometri',
60            message: 'Bruk biometri for å logge inn'
61          })
```

Notifikasjon

For at appen skal gi notifikasjoner, bruker gruppen en plugin fra nativescript som heter local-notifications. Denne logikken er ikke implementert i appen, men blir sett på som en videre løsning. Koden under viser et eksempel på hvordan programmet spør om tillatelse til å aktivere notifikajsoner ved hjelp av LocalNotifications.requestPermission(). Hvis brukeren gir tillatelse blir n verdien satt til true, hvis brukeren sier nei blir tillatelsen satt til false.

```
35  // Method askin user for permission to enable notifications
36  const notifPermission = () => {
37    LocalNotifications.requestPermission().then((n) => {
38      console.log("Gi tillatelse?" + n);
39      if (!n) {
40        notifStore.setPermission((check.value = false));
41        console.log("Hei");
42      }
43    });
44  };
```

Hamburgermeny

For å implementere en hamburgermeny i appen, bruker gruppen et rammeverk fra nativescript som heter drawer. Innholdet i hamburgermenyen nedenfor er samlet i en gridlayout med en atributt som heter leftDrawer som inneholder alle radene brukeren kan navigere seg til. Hvis hidden og backbutton er false vil gridlayout vises og hamburgermenyen kommer frem.

```
101     <Drawer ref="drawer">
102       <!-- leftDrawer contains the content of the drawer -->
103       <!-- Only render the drawer if 'hidden' and 'backButton' are false -->
104       <GridLayout
105         ~leftDrawer
106         v-if="!hidden && !backButton"
107         class="w-4/5 bg-sidebar py-1 text-2xl text-white"
108         rows="*, auto"
109         @tap=""
110       >
```

Bottom sheet

I app.ts er det konfigurert ulike funksjoner for at appen skal fungere som planlagt. For å bruke bottomsheet, en meny som kommer opp fra bunnen av skjermen når brukeren skal velge prosjekt, må programmet oprette en ny Vue app ved hjelp av createApp(App) funksjonen. Dette gjør at gruppen kan bruke app.use(BottomSheetPlugin) for å bruke funksjonaliteten til menyen.

```
21    app.use(BottomSheetPlugin)
22    app.use(DrawerPlugin)
```

dayjs

dayjs er et javascript bibliotek som gjør det lettere for gruppen å jobbe med datoer i prosjektet. Det er lettere å forkorte dager og måneder og gruppen kan subtrahere og legge til dager og timer for å få en lettere oversikt over timene brukt på ulike prosjekter.

Her er et eksempel på hvordan gruppen har brukt dayjs bibloteket. Ukedagene blir endret til norsk slik at dayjs vil bruke de norske ukedagene når 'nb' blir brukt i updateLocale. Dette vil også gjøre at gruppen kan arbeide med norske tider i applikasjonen i tillegg til de norske navnene.

```
14  // Setup custom locale
15  dayjs.updateLocale('nb', {
16    weekdays: ['Søndag', 'Mandag', 'Tirsdag', 'Onsdag', 'Torsdag', 'Fredag', 'Lørdag']
17  })
18  dayjs.locale('nb')
```

Secure storage er en plugin i nativescript-vue rammeverket som lar gruppen lagre sensitiv data fra applikasjonen. Secure storage lagrer og krypterer dataen basert på hvilken enhet brukeren er på, hawk på android og SAMkeychain på iOS.

9 KONTINUERLIG INTEGRASJON OG TESTING

Mobilappen testes ved å bruke android studio (https://developer.android.com/studio) og deretter kjøre kommandoen *yarn android* fra prosjektet. Dette vil starte en lokal versjon av prosjektet, hvor funksjoner kan testes manuelt.

10 REFERANSER