Systemdokumentasjon

Utvikling av mobilapplikasjoner for helsepersonell

Team 138

Revisjonshistorie

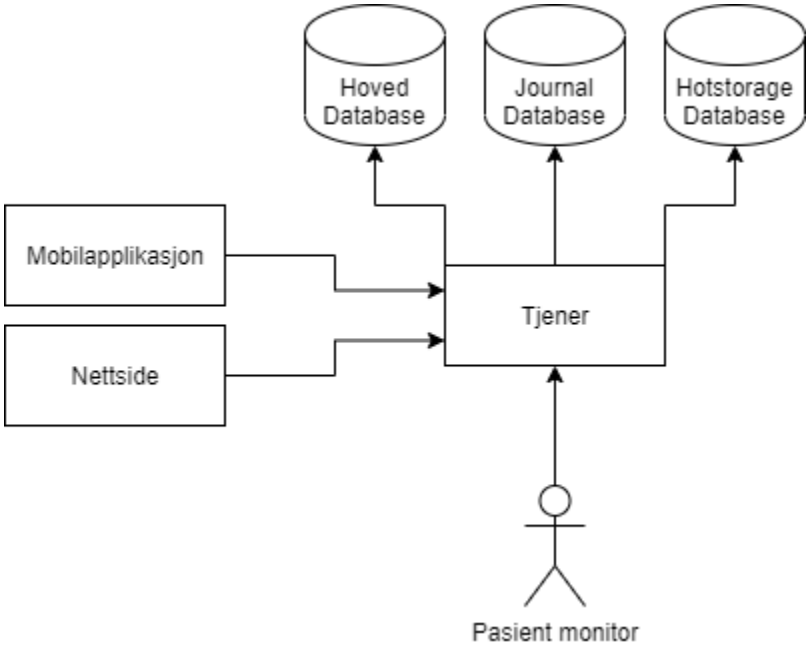
Dato	Versjon	Beskrivelse	Forfatter
18/05/21	1.0	Første ferdige utgave	Emir Derouiche, Nikolai Dokken, Ian Evangelista, Kasper Gundersen

Innholdsfortegnelse

1. Introduksjon

Dette dokumentet er skrevet i forbindelse med bacheloroppgaven *Utvikling av mobilapplikasjoner for helsepersonell* av Emir Derouiche, Nikolai Roede Dokken, Ian-Angelo Roman Evangelista og Kasper Vedal Gundersen våren 2021. Dokumentet skal forklare systemet i en praktisk forstand for å bistå fremtidig videreutvikling av prosjektet. Det følger med en rekke vedlegg med oversikt og forklaring mot slutten av dette dokumentet.

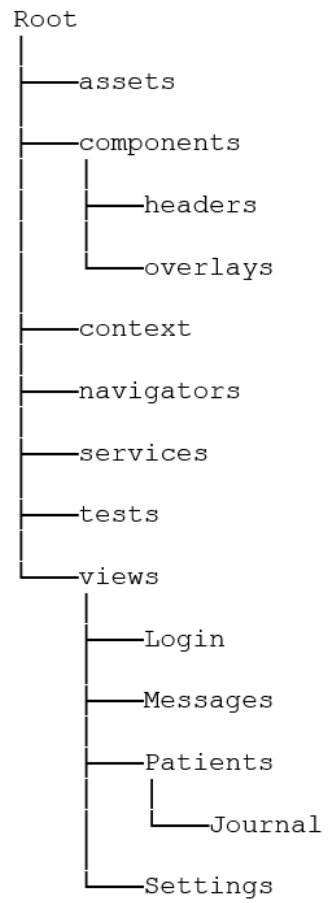
2. Arkitektur



Mobilapplikasjonen har vært vårt hovedfokus ettersom det er den vi hadde ansvar for å utvikle, men vi har også videreutviklet tjeneren og databasene for å støtte ny funksjonalitet. Nettsiden er den eksisterende klienten som appen skulle basere seg på. Tjeneren og nettsiden er laget i Django og de tre databasene er PostgreSQL databaser, mens mobilapplikasjonen er laget i React Native. Det er tre databaser for å adskille journalsystemet og midlertidig lagring av livedata. Pasientmonitoren strømmer data direkte til tjeneren gjennom blant annet mobildata som betyr at pasienten ikke behøver å befinne seg på sykehuset for å bli monitorert.

3. Prosjektstruktur

3.1 Mappestruktur



På bildet over ser du mappestrukturen til mobilapplikasjonen. Assets er bilder vi bruker i appen. Components er gjenbrukbare react native komponenter vi har laget, den har også du mapper hvor vi har skilt ut headers og overlays. Headers er egendesignede top menyer for steder vi trengte flere valgmuligheter enn kun en tilbakeknapp på top. Overlays er componenter som dukker opp over viewen. Context inneholder globale variabler som for eksempel fargene for fargetemaet som brukeren har valgt. Navigators inneholder alle navigasjonskomponenter for appen. Services inneholder alle klassene som tar kontakt med tjeneren gjennom http. Views inneholder alle sidene en bruker kan navigere til og har undermapper for de sidene med flere sider innad.

@react-native-community/masked-view
@react-native-community/slider
@react-navigation/bottom-tabs
@react-navigation/material-top-tabs
@react-navigation/native
@react-navigation/stack
axios
base-64
buffer
crypto-js
expo
expo-av
expo-camera
expo-document-picker
expo-file-system
expo-haptics
expo-image-picker
expo-notifications
expo-screen-orientation
expo-status-bar
moment
react
react-dom
react-native
react-native-appearance
react-native-elements
react-native-gesture-handler
react-native-gifted-chat
react-native-lightbox

react-native-paper
react-native-reanimated
react-native-safe-area-context
react-native-screens
react-native-svg
react-native-svg-charts
react-native-tab-view
react-native-video
react-native-webview
reanimated-bottom-sheet
rn-pdf-reader-js

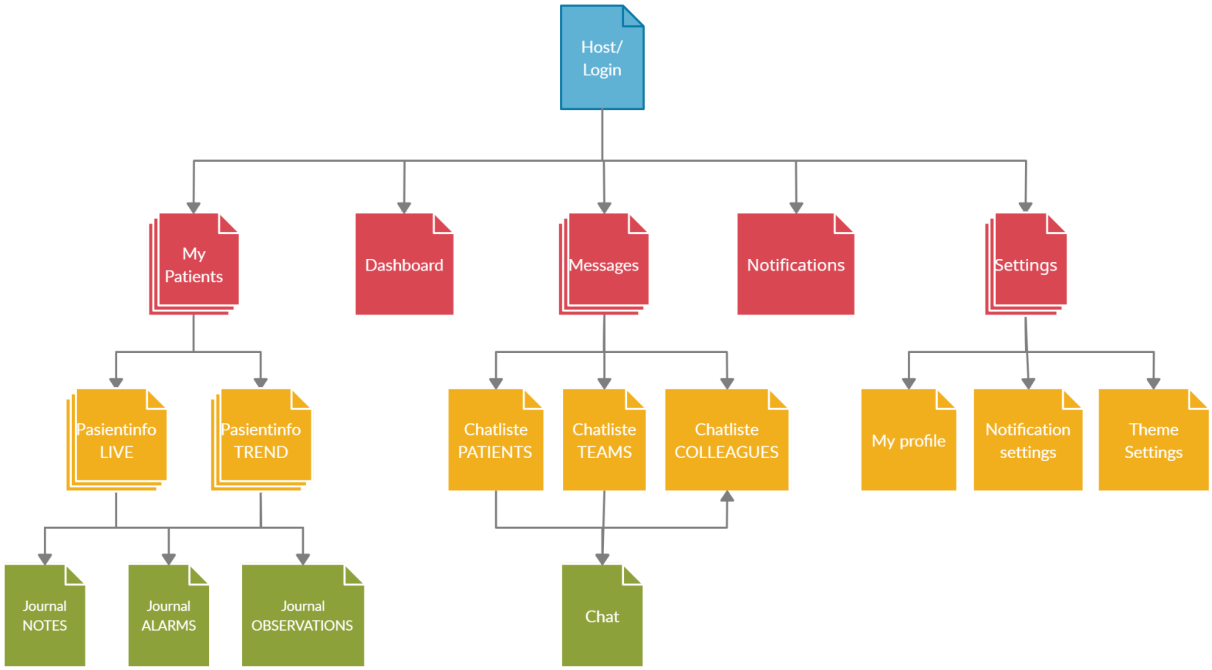
3.2 NPM-biblioteker

På bildene over ser du alle bibliotekene vi bruker. De viktigste er expo, react og react native som er rammeverkene vi har brukt for prosjektet. Axios og base-64 brukes for å gjøre https kall. Crypto-js, base-64, expo-camera og buffer brukes for å dekryptere qr-koder. Moment brukes for å formatere tidsstempel. rn-pdf-reader-js brukes for å lese pdf filer i chat. reanimated-bottom-sheet brukes for informasjonskort som forklarer funksjoner. React-native-svg brukes for å tegne grafer for trenddata. React-native-paper gir oss gjenbrukbare komponenter som følger googles material design prinsipper. react-native-gifted-chat brukes for å vise og opprette meldinger i chatten.

4. Klassediagram

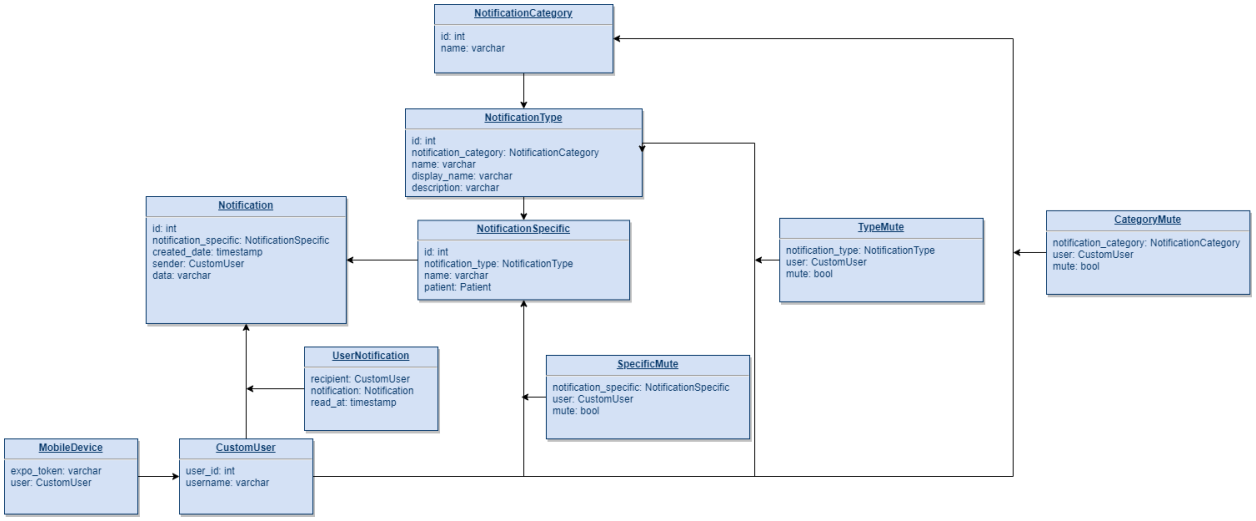
Koden vår tar nytte av react funksjonelle komponenter og er dermed ikke objektorientert. På lik linje er koden for tjeneren skrevet i funksjonell python. Vi mener derfor at det ikke er hensiktsmessig å lage klassediagram.

5. Navigasjons i appen



Figuren viser navigasjons hierarkiet til applikasjonen. Fargene viser dybden og ark bunkene viser til at for eksempel “My Patients” er en stack navigator som tillater den å ha flere enn ett barn. Hvis man for eksempel vil navigere seg fra “pasientinfo” til “theme settings” må man først navigere seg til “settings” og deretter til “theme settings”. Den eneste måten å komme seg tilbake til “Host/login” er å trykke på logg ut knappen i settings.

6. Databasemodell



Databasemodellen viser kun den delen av databasen vi utviklet helt selv. De fullstendige diagrammene for de tre databasene til systemet er svært store og oppdragsgiver ville ikke at vi skulle offentliggjøre de, så vi har derfor unnlatt å vise det fullstendige bildet. Databasen kjører Postgres. Vi satte opp systemet slik at hver notifikasjon gjelder en spesifikk pasient som vi har kalt en “notification specific”. Hver specific har en type og hver type har en kategori. Vi har utviklet det slik for å tilrettelegge for at en bruker kan stanse push varsling på forskjellige nivåer. Hvis brukeren får mange push varslinger fra en spesifikk pasient kan brukeren velge å sette den pasienten på stum, og hvis brukeren ikke vil ha push varsler fra chatten i det hele tatt kan brukeren også gjøre det. Dette har vi oppnådd med mange til mange relasjoner mellom brukere og specifics, typer og kategorier. Vi innså at et stort sykehus kunne potensielt få enormt mange rader i disse koblingstabeller som aldri ville bli brukt, derfor implementerte vi de systemet slik at radene kun opprettes når de skal brukes. Hvis brukeren ikke har en rad i koblingstabell for notifikasjonens type for eksempel, vet vi at brukeren ikke har vært inne og endret på innstillingene og dermed vet vi at brukeren ikke har mutet. Hver bruker kan også ha flere enheter og alle vil få push varslinger hvis brukeren ikke har logget ut på enheten. Notifikasjonssystemet er utdypet i vedlegg 8, 9 og 10.

7. Server-tjenester

Vi har dokumentert alle REST-ressurser med swagger i løpet av prosjektet. Vedlegg 11 er en oversikt over alle endepunktene. Vi har utviklet endepunktene for IM, Journals, notifications, utils og noen av de på devices.

Vi brukte også to endepunkter for websockets. Den ene var utviklet av infiniwell og var for å strømme live data til den digitale pasientmonitoren. Den andre utviklet vi i samarbeid med infiniwell og den var for å sende og motta chat meldinger i sanntid. Vi har utdypet websockets i appen i vedlegg 12.

8. Sikkerhet

Vi sørget for at appen vår benyttet seg av infiniwell sitt robuste autentiseringssystem som er bakt inn i Django og baserer seg på Oauth 2.0 standarden. Alle passord blir automatisk hashet og saltet. Vi bruker Bearer tokens sammen med Client Id og Client Secret for å verifisere at brukeren har en gyldig sesjon. Hvis brukeren ikke har en gyldig sesjon vil ingen av endepunktene bortsett fra innlogging være tilgjengelige. Django har god innebygget støtte for input validering og prepared statements, noe som vi sørget for å bruke på alle endepunktene våre. Cross-site scripting er ikke relevant ettersom vi utviklet en app og ikke en nettside, og koden vår kan dermed ikke tolke tekst som HTML kode.
For all kommunikasjon til og fra tjeneren bruker vi https.

Vi implementerte login via QR-kode som benyttet AES-256 kryptering for å skjule dataen. QR-koden inneholder kun adressen til apiet og brukernavnet til brukeren. Brukeren må skrive inn passordet sitt i tillegg til å scanne QR-koden for å logge inn for å bevare maksimal sikkerheten.

9. Installasjon og kjøring

9.1 Kjøre kildekoden lokalt

Forutsetninger for PC

- Node og NPM må være installert
- Expo må være installert: skriv “npm install --global expo-cli” i et terminalvindu for å installere
- Kildekoden må være lastet ned

Forutsetninger for mobil

- Applikasjonen “Expo Go” må være installert
    - App store - https://apps.apple.com/no/app/expo-go/id982107779?l=nb
    - Google play - https://play.google.com/store/apps/details?id=host.exp.exponent&hl=no&gl=US

Kjøring av koden

1. Åpne en terminal i roten av prosjektets kildekode og kjør “npm i --force”.
2. Kjør “npm start” og scan QR-koden som dukker opp i terminalen med vanlig kamera hvis du har iPhone eller det innebygde kameraet i Expo Go-appen hvis du har android mobil.
3. Appen vil nå bygges og du vil komme til en skjerm som lar deg skrive inn “Sentiohost”. Gå til avsnittet om innlogging lengre ned.

NB! Dersom problemer med installering eller kjøring av expo oppstår, kan du også laste ned appen fra TestFlight eller Google play. Se kapittel 9.2 under.

Det er i tillegg en guide tilgjengelig i prosjektets readme fil.

9.2 Last ned appen fra Testflight eller Google play store

Vi har også lastet opp applikasjonen vår til Apples og Googles respektive testprogrammer for enklere tilgang. Følg lenkene under for å installere den ferdigbygde appen til din enhet(Det enkleste er å scanne QR koden med enhetens kamera)

For iOS	For Android
Scan QR-koden med mobilens kamera og følg instruksjonene: Du kan også bruke denne linken på din mobil: https://testflight.apple.com/join/J22qHbZT	Scan QR-koden med mobilens kamera eller en QR-scanner app: Du kan også bruke denne linken på din mobil: https://play.google.com/store/apps/details?id=com.nikord.SentioApp

9.3 Innlogging i appen

Appen er avhengig av en tjener for å fungere og brukeren må få en bruker registrert i systemet ettersom det ikke er ment at brukere skal registrere seg selv. For dette formålet har vi opprettet en test-tjener, som kan brukes for eksterne.
Vi skal be oppdragsgiver holde tjeneren vår kjørende så lenge som mulig.
Når du er inne på appen kan du tillate appen å bruke kameraet og scanne QR koden under med kameraet som er på startsiden i appen:

Denne vil fylle ut host og brukernavn for deg. Skriv så inn passord: “ntnutest”.

Alternativt kan du skrive inn “test.sentioweb.com” i host feltet og deretter logge inn med:
Brukernavn “testuser1”.
Passord: “ntnutest”.

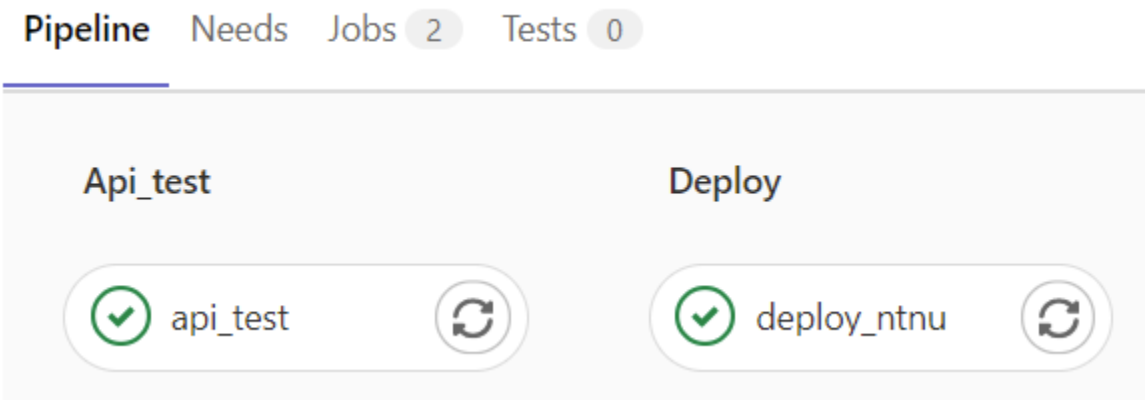
10. Dokumentasjon av kildekode

Swagger dokumentasjon er generert automatisk når man starter tjeneren og er tilgjengelig på nettsiden når man har logget inn. De legges til ved å skrive en swagger kommentar over definisjonen av funksjonen til endepunktet og generes ved hjelp av et pip3 bibliotek som heter drf-yasg. Dokumentasjon for hvordan man legger til swagger på endepunkter med drf-yasg er er offentlig tilgjengelig[1].

Kommentarer er lagt til der vi mener det ikke er åpenbart hva koden gjør for en utvikler som er ny til kildekoden. Utover det har vi prøvd å skrive koden intuitivt og benytte beskrivende variabelnavn.

11. Kontinuerlig integrasjon og testing

Testene for tjenerkoden bruker Django sin integrerte testløsning og kjøres lokalt med “make test”. Disse testene kjøres automatisk med Gitlab CI i en Ubuntu 18.04 docker container og om testene består oppdaters koden på tjeneren med den nyeste kildekoden med kontinuerlig utrulling. Vi benyttet regresjonstesting for å forsikre oss om at ny kode fra nye sprinter ikke ødela andre steder i koden. Vi simulerer for eksempel en samtale mellom to kollegaer ved å kjøre kall til apiet og sjekke om endringer skjer som forventet ved kall til andre api endepunkter. Vi har mange slike virkelighetsnære tester for å forsikre oss om at resultatet av api-kallene er forutsigbare og uendret ved endring andre steder i koden. All testing benytter seg av egne test databaser som kun opprettes når testene kjøres og slettes når testingen er ferdig. For testing av den tjener-koden vi har utviklet har vi 100% dekning med tester.
Da tjener-koden er Inifiniwells eie, og vi bare har utviklet deler av den, vil vi ikke legge ved hele kodebasen til tjener, og det vil dermed ikke være mulig å kjøre disse testene.



I bildet over ser du hvordan pipelinen ser ut for CI/CD i backend.

Testing av appkoden bruker Jest og kjøres med “npm test”, for appen har vi snapshot-tester og enhetstester. Vi tester altså at om rendering av komponenter fungerer og at funksjoner vi har laget fungerer slik de er ment til å fungere. Testing av appen er mindre omfattende ettersom vi møtte på problemer med headless rendering ettersom appen vår er avhengig av et omfattende navigasjonssystem og en global context, og siden hovedfokuset vårt når det kom til testin var på tjeneren. Vi merket også at noen av bibliotekene appen er avhengig av ikke var kompatible med testmiljøet og ga oss store problemer. Vi har derfor valgt å ta med testing av klientside(appen) i kapittelet om videre arbeid i hovedrapporten vår, da dette vil være en viktig del av videre utvikling.

12. Vedlegg

For kode overrekkingen til infiniwell produserte vi del del dokumenter og figurer for å beskrive systemene vi har utviklet. Disse skal også være tilgjengelige sammen med dette dokumentet. Her er en oversikt over vedleggene:

1) Add note.png
   Sekvensdiagram for hva som skjer når en bruker legger til et journal note
2) App initialization (BottomTabNavigator).png
   Sekvensdiagram for hva som skjer når appen initialiseres
3) Chat.png
   Sekvensdiagram for hva som skjer når en bruker bruker chatten
4) Code Handover.pdf
   Lysark fra presentasjonen vår for infiniwell
5) Dashboard.png
   Sekvensdiagram for hva som skjer når brukeren åpner dashbordet
6) Future changes_ New parameters.pdf
   Et dokument som forklarer hvilke deler av koden man må ta hensyn til om man vil endre parametere for pasientmonitoren
7) Messages and attachments in the app and backend.pdf
   Hvordan meldinger og vedlegg i meldinger fungerer både i klienten og tjeneren
8) Notification.png
   Databasemodell for notifikasjon systemet
9) Notification hierarchy.png
   En oversikt over notifikasjons kategoriene og deres typer
10) Notifications.pdf
    En forklaring av notifikasjon systemet vi utviklet.
11) Swagger.html
    Swagger dokumentasjon for endepunktene til REST apiet
12) Websockets in the app.pdf
    En forklaring av hvordan websockets er brukt i appen
13) Infiniwell Baseline Design Guideline.pdf
    Design-retningslinjer fra Infiniwell

13. Referanser

[1] *Quickstart* [lest 11.05.21] drf-yasg
https://drf-yasg.readthedocs.io/en/stable/readme.html#quickstart