Systemdokumentasjon for FixrateApp

Versjon 1.2

Revisjonshistorie

Dato	Versjon	Beskrivelse	Forfatter
24.01.2020	1.0	Oppstart systemdokumentasjon	Kristian Kampenhøy og Simon Lilleeng
26.01.2020	1.1	Oppdatert kapittel Sikkerhet	Kristian Kampenhøy
28.04.2020	1.2	Oppdatert kapittel Kontinuerlig integrasjon og testing, og Sikkerhet	Kristian Kampenhøy

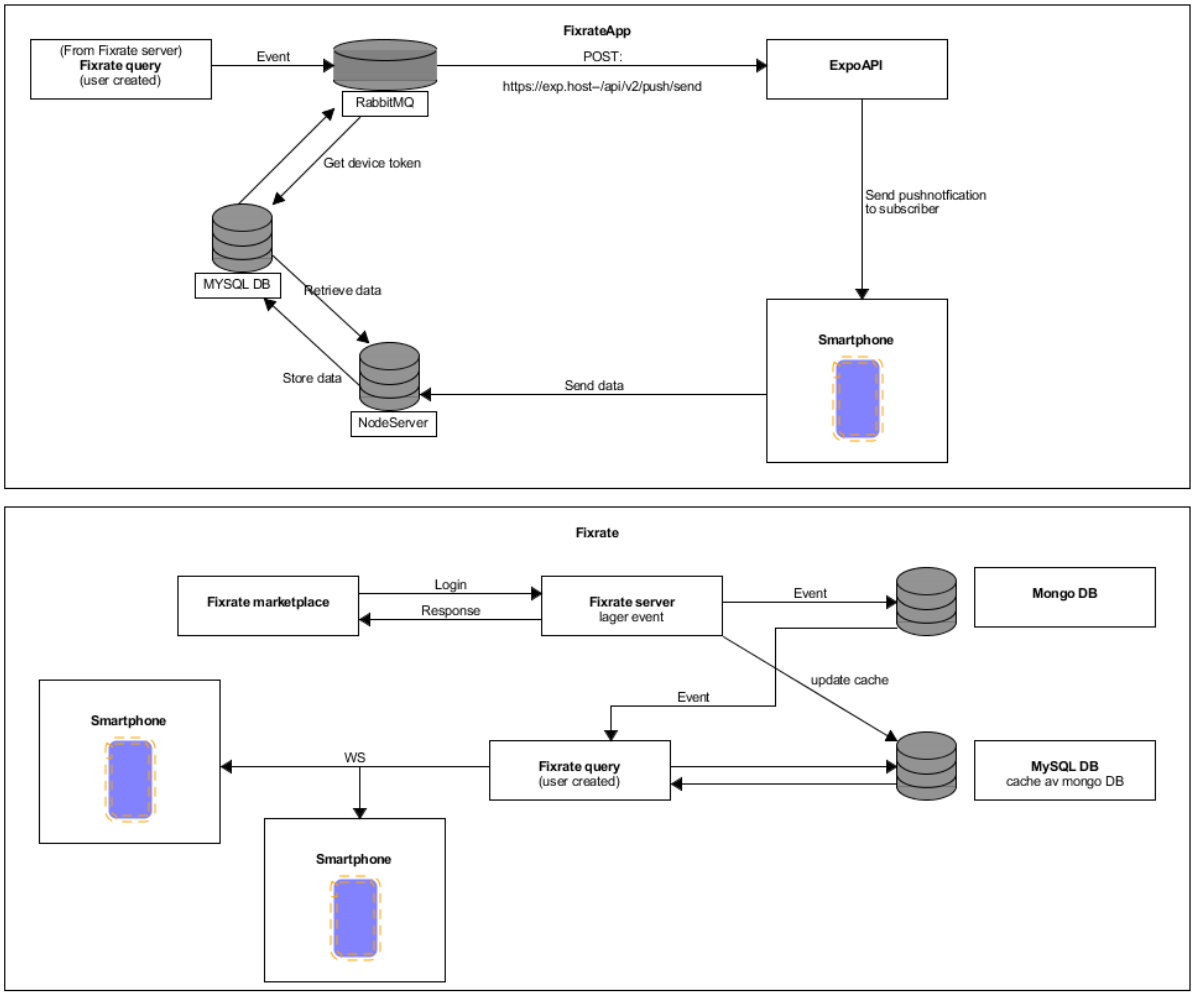
Innholdsfortegnelse

Figurer

1 Introduksjon

Dette dokumentet er skrevet i forbindelse med Bacheloroppgave 2020, der prosjektgruppen skal videreutvikle en mobilapplikasjon til Fixrate. Hensikten med dette dokumentet er å gi leseren en oversikt over hvordan systemet er laget og bygget opp ved tekniske beskrivelser. Dokumentet inneholder arkitektur tegninger, og forklaringer på hvordan testene til systemet kan gjennomføres.

2 Arkitektur



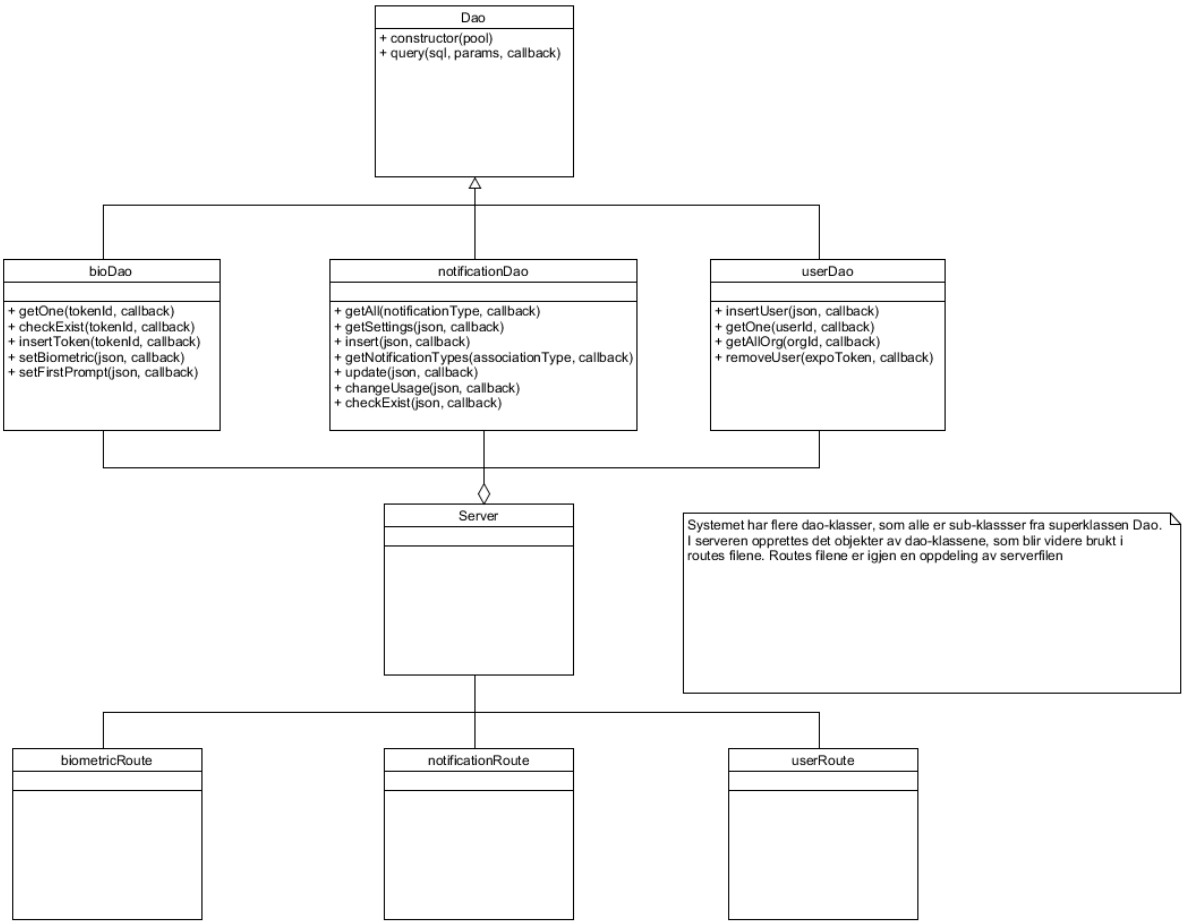
Figur 1: Arkitektur tegning

3 Prosjektstruktur

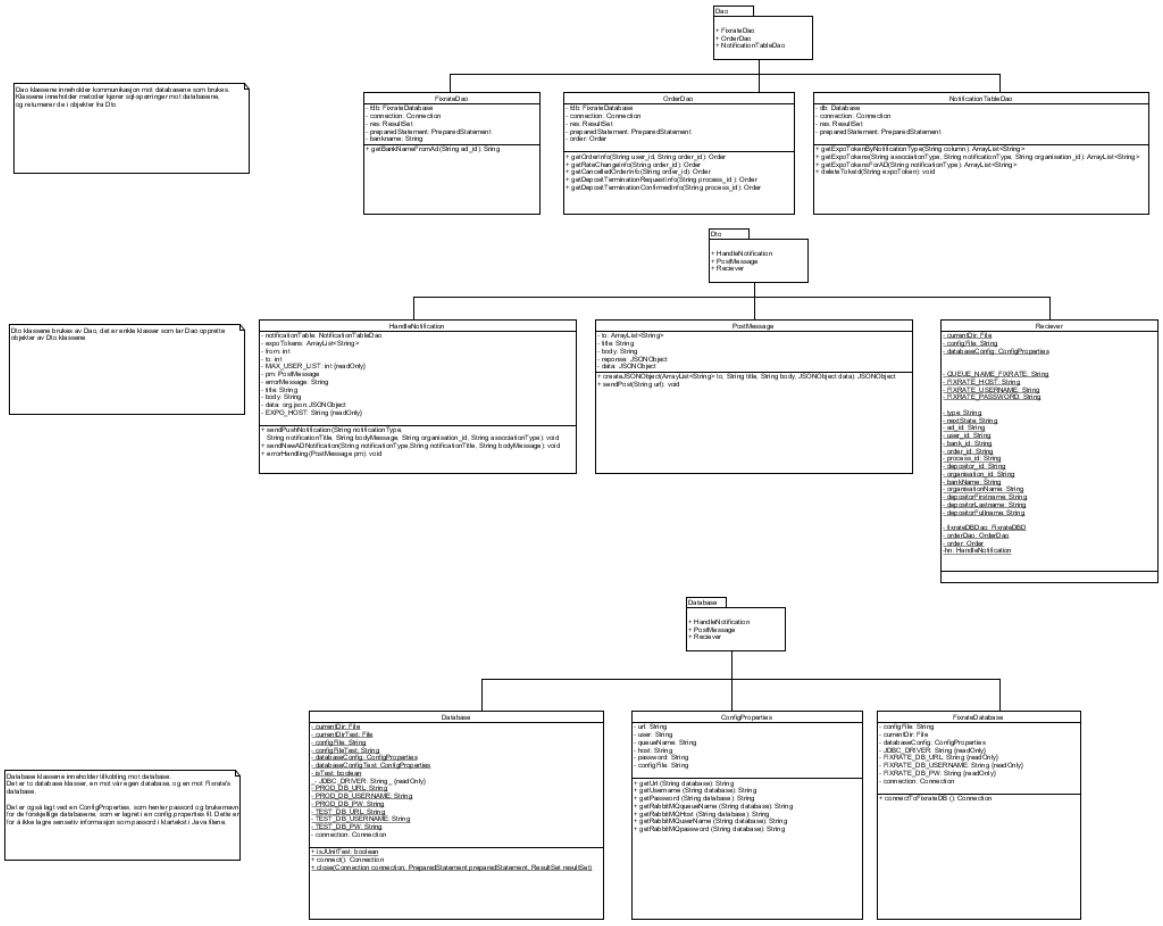
Frontend	Backend
React-Native	Node js
Expo	Nodemon
Axios	Jest
Enzyme	Express
Sinon	MySQL
Babel	
Jest	
React-Redux	
React-Navigation	
Redux-devtool-extension	

Se vedlagt fil i zip-mappe for prosjektstruktur

4 Klassediagram

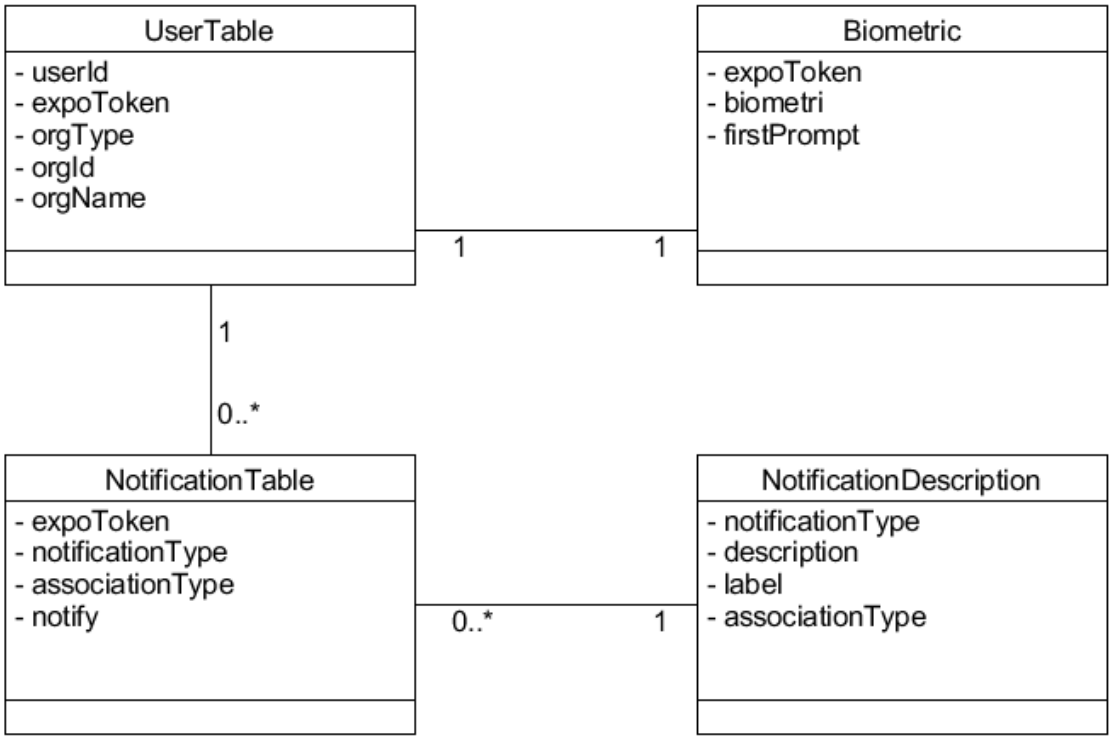


Figur 2: Klassediagram nodeServer



Figur 3: Klassediagram rabbitMQ

5 Databasemodell



Figur 4: Databasemodell

Se vedlegg for Databasemodell

6 Server-tjenester

Systemet inneholder en Node-server som tar for seg kommunikasjonen med Expo sitt api for pushvarsler, og kommunikasjon mot en egen database som vi har implementert selv, gjennom DigitalOcean.

Overordnet oversikt over de endepunktene som er implementert i systemet på serveren

Endepunkt	Verb	Hva	Type
/userId	GET	Henter en bruker på id fra databasen	Data
/	PUT	Oppdaterer bruker tabellen i databasen med ny bruker eller ny brukerinformasjon	Data

Endepunkt	Verb	Hva	Type
/notificationType	GET	Henter expoToken med gitt notifikasjons type	Data
/settings	PUT	Henter nye endringer på innstillingene til en bruker og oppdaterer databasen	Data
/settings/changeUsage	PUT	Oppdaterer innstillinger hos en bruker, der bruker kan bytte rolle i systemet	Data
/	PUT	Oppdaterer brukerinnstillinger	Data

Endepunkt	Verb	Hva	Type
/tokenId	GET	Henter innloggings innstillinger på expoToken	Data
/	PUT	Setter biometrisk innlogging innstillinger	Data
/prompt	PUT	Oppdaterer variabel i databasen som forteller om bruker er innlogget for første eller ikke	Data

7 Sikkerhet

Ved innlogging bruker applikasjonen et webview fra Signicat [7] som tar for seg autorisering av brukere gjennom innlogging med bankID. Etter at brukeren har autentisert sendes en Cookie med en SSID (unik ID for hver bruker). SSID en gjør det mulig at vi kan kjøre et rest kall mot endepunkt hos Fixrate. Fra dette endepunktet mottar applikasjonen deretter et JSON web token [5] som applikasjonen bruker videre for å hente riktig data fra endepunkter hos Fixrate for den aktuelle brukeren.

JSON web tokenet lagres så i SecureStore for å oppnå biometrisk innlogging, hvis brukeren velger det. Ved tilfeller der biometri er aktivert, og brukeren er autentisert, hentes JSON web tokenet fra SecureStore. Gyldigheten til web tokenet blir sjekket i webviewet [6]. Viser det seg at tokenet ikke er gyldig, blir brukeren logget ut igjen, og må igjen da logge normalt inn med bankID slik at applikasjonen kan motta en ny JSON web token.

SecureStore vil ha beskyttet tilgang ved å bruke LocalAuthentication. Applikasjonen får kun lov å hente ut data derfra om brukeren kan autentisere seg. Ved autentisering mottar applikasjonen JSON web tokenet fra SecureStore og blir deretter logget inn.

SecureStore tilbyr applikasjonen et eget system for lagring av sensitiv data. SecureStore krypterer og dekrypterer innholdet som lagres. iOS og Android bruker dette litt forskjellig da disse enhetene tilbyr ulike måter å lagre sensitiv informasjon på. Verdier som lagres på iOS lagres ved å bruke “Keychain services” [3]. Denne brukes på iOS hele tiden da det er denne som husker passord du lagrer på telefonen din.

På Android lagres verdier i en mappe som heter “SharedPreferences”, kryptert med “Android Keystore System” [4]. Denne fungerer på samme måte som på iOS. All data som lagres i SecureStore i applikasjonen er kun tilgjengelig på den enkelte enheten det ble lagret på [1].

LocalAuthentication er biblioteket som muliggjør biometrisk innlogging vha. FaceID/TouchID på iOS eller TouchID på Android. LocalAuthentication tar i bruk eksisterende biometrisk data som allerede ligger på telefonen til brukeren. Så når applikasjonen mottar biometrisk data sammenligner den det med dataen som allerede ligger lagret på telefonen. LocalAuthentication gir derfor bare en true/false på om autentiseringen er vellykket. Biblioteket tilbyr også mulighet til å sjekke om brukeren har nødvendig maskinvare for å bruke biometri, hvilken maskinvare brukeren har og om brukeren har konfigurert biometri slik at det kan brukes og ikke minst autentisering gjennom bruk av FaceID, TouchID eller passord [2].

Applikasjonen har også lagt inn beskyttelse mot SQL-injection, som er at kode og data blandes i spørringene før det sendes til serveren. Med PreparedStatements på server-siden vil da spørringer (kode) og data spørringer bli sendt separat til databaseserveren som brukes mot Expo sitt api for pushvarsler.

På serversiden har løsningen laget config.properties fil, der passord og brukernavn og annet sensitiv informasjon lagres. Denne filen er lagt til i gitignore slik at informasjonen ikke havner på gitlab og blir tilgjengelig for andre. Grunnen er at det ikke skal lagres sensitiv informasjon i klartekst i systemet.

8 Installasjon og kjøring

For forklaring på rammeverk og biblioteker, se hovedrapport under kapittel 3. Valg av teknologi og metode.

Installasjon:

Klon prosjektet selv og kjør gjennom egen mobilapplikasjon:

1. Installer Node.js og NPM

- Etter installeringen er gjennomført kan en bekrefte at det er installert med å bruke kommandoene i terminalen:

  $ node -v

  $ npm -v

2. Klon prosjektet gjennom gitlab med en git klient, for eksempel git bash:

   $ git@gitlab.com:kampen/fixrateapp.git

3. Installer node_modules i rotkatalogen med følgende kommando:

   $ npm install

4. For å starte applikasjonen på egen mobiltelefon, må Expo-client lastes ned fra Apple Store, eller Google Play alt ettersom hvilken type mobiltelefon en bruker.

- Gå i rotkatalogen og skriv inn kommandoen $ npm start for å starte applikasjonen. Det vil komme opp en QR-kode som en skanner med sin egen mobiltelefon gjennom Expo-client applikasjonen.

5. For at applikasjonen skal vise fram pushvarslene på telefonen må en sette opp et docker-image som kan kjøres gjennom DigitalOcean. I tillegg må en bruke kubernetes for å få tilgang til mysql databasen, og rabbitmq serveren til Fixrate.

- For at mobilapplikasjonen skal klare å få tilgang til node-serveren i systemet, kan ikke serveren kjøres lokalt, derfor er det lages det et docker-image som kan kjøres opp på DigitalOcean server.
- Åpne to terminaler, i begge terminalene skal du bruke port-forward, en for å hente MySQL-podden til Fixrate, siden det da kjøres spørringer mot deres database. Den andre terminalen skal brukes for å hente rabbitmq-podden, slik at eventene som sendes over rabbitmq server til Fixrate, kommer over til systemets rabbitmq server. Slik at systemet kan agere på de eventene som kommer inn.

  ```
  $ kubectl port-forward svc/fixrate-mysql 3306

  $ kubectl port-forward rabbitmq-0 5672
  ```

6. Etter at port-forwardingen er på plass må du starte main metoden i “Reciever.java” klassen. Det er vår rabbitmq server som da vil motta eventene som kommer fra Fixrate sitt nettsted.

9 Dokumentasjon av kildekode

Får å se dokument av kildekode, åpne prosjektmappen FixrateApp. Deretter åpne mappen docs og derfra åpne fil med filnavn index.html.

En ny nettside i nettleseren vil åpne dokumentasjonen av kildekoden til prosjektet.

10 Kontinuerlig integrasjon og testing

I dette systemet er det valgt å bruke Git som versjonskontroll med Gitlab og CI løsning da det er det vi er mest kjent med, og har valgt gitlab.com istedet for gitlab.stud.ntnu.no for økt stabilitet. Måten det jobbes på er ved at det lages egne brancher når det skal implementeres nye funksjoner ol. slik at master holdes ryddig og kjørbar hele tiden. Ved lagring av nye features og funksjoner merger vi master med den aktuelle branchen som det har blitt jobbet på, for å sikre at de nye funksjonene fungerer godt sammen med branch master. Deretter merger vi branchen inn på master. Når det merges med master kjøres alle tester automatisk gjennom en gitlab-ci.yml fil som oppretter et docker image med oppkobling til en test-database som er satt opp gjennom DigitalOcean.

Det er hovedsakelig laget tester for server-siden. Der testes det at de sql-spørringene som går mot databasen henter, oppdaterer, og legger til de verdiene som er ønskelig. Som nevnt går dette mot en test-database, slik at hoveddatabasen ikke blir påvirket når testene kjøres i systemet.

Det har ikke blitt lagt nok vekt på å få implementere tester på klient-siden. Noe i grunnen til dette er måten mange av komponentene som brukes er laget på. Systemet inneholder flere "smarte" komponenter, som gjør det vanskeligere å få testet komponenten isolert.

Systemet består av flere moduler. To server-deler, der vi har node-server mot egen database, og en RabbitMQ-server, som er en event-handler som ligger og lytter på events fra nettstedet til Fixrate. RabbitMQ serveren er laget i Java med Maven rammeverket. Der er det ikke laget tester for alle metodene som er implementert, og grunn i det er at det kjøres spørringer mot databasen til Fixrate. Det har blitt tatt en beslutning på at det ikke er nødvendig, siden Fixrate har sine laget sine egne tester i sitt system.

Referanser

[1] SecureStore https://docs.expo.io/versions/latest/sdk/securestore/ (Besøkt Februar. 2020)

[2] LocalAuthentication https://docs.expo.io/versions/latest/sdk/local-authentication/
(Besøkt Februar. 2020)

[3] Keychain Services iOS
https://developer.apple.com/documentation/security/keychain_services
(Besøkt Februar. 2020)

[4] Android Keystore System https://developer.android.com/training/articles/keystore.html
(Besøkt Februar. 2020)

[5] JSON Web Token Introduction https://jwt.io/introduction/ (Besøkt Februar. 2020)

[6] WebView: React Native https://reactnative.dev/docs/webview (Besøkt Februar. 2020)

[7] Signicat https://www.signicat.com/no/ (Besøkt Mai. 2020)

Vedlegg

- Arkitektur tegning
- Prosjektstruktur
- Klassediagram node_Server
- Klassediagram rabbitmq
- Databasemodell