Høgskulen på Vestlandet

Fakultet for teknologi, miljø- og samfunnsvitenskap

Institutt for datateknologi, elektroteknologi og realfag

Miside bedrift – Selvbetjeningsportal for bedriftskunder hos Enivest

Systemdokumentasjon

Versjon 3.0

REVISJONSHISTORIE

Dato	Versjon	Beskrivelse	Forfatter
14.04.2024	0.1	Arkitektur og databasemodell	Anders Fimreite
16.04.2024	1.0	Endepunkter lagt inn	Jørgen Fjølstad
01.05.2024	2.0	Litt forskjellig	Anders Fimreite
12.05.2024	3.0	Ferdigstilt dokument	Jørgen Fjølstad

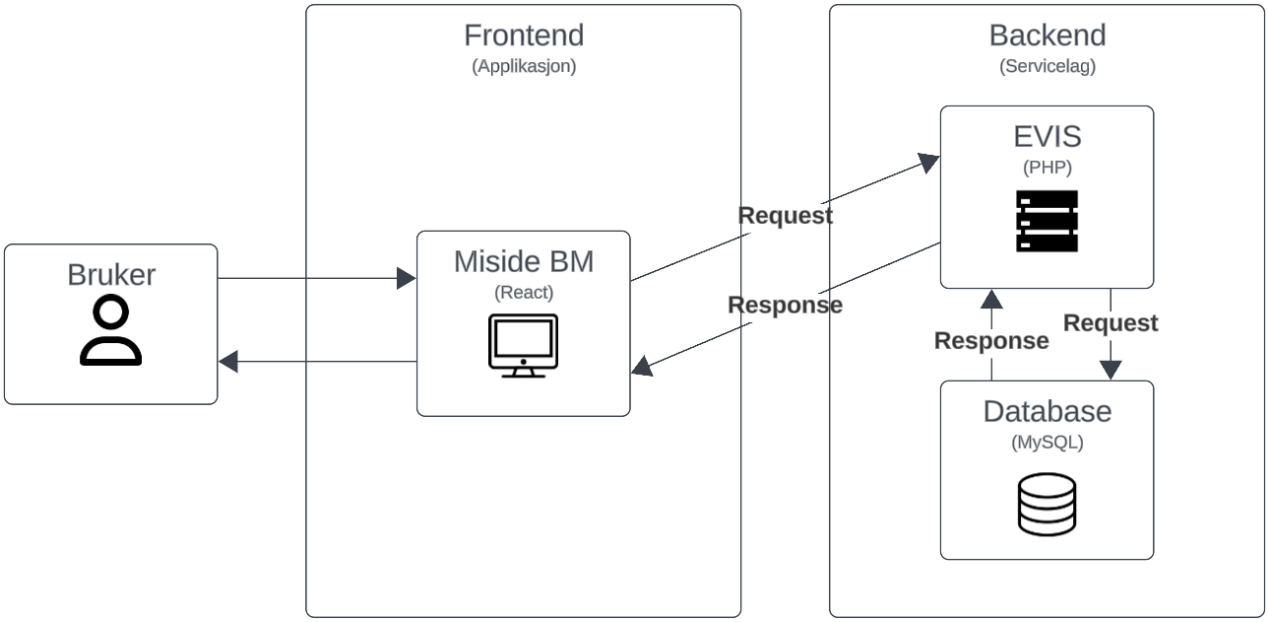
INNHOLDSFORTEGNELSE

1 INNLEDNING

I forbindelse med utviklingen av løsningen i dette prosjektet har systemdokumentet som hensikt å beskrive systemet slik som det er designet og laget. Dokumentet starter med å beskrive arkitekturen for løsningen, deretter løsningens struktur i form av filstrukturen. Videre kommer klassediagram, databasemodell og liste over server-tjenester med beskrivelse. Så forklares noen aspekter rundt løsningens sikkerhet. Videre beskrives hvordan applikasjonen blir kjørt. Til slutt blir det beskrevet hvordan testing er blitt implementer.

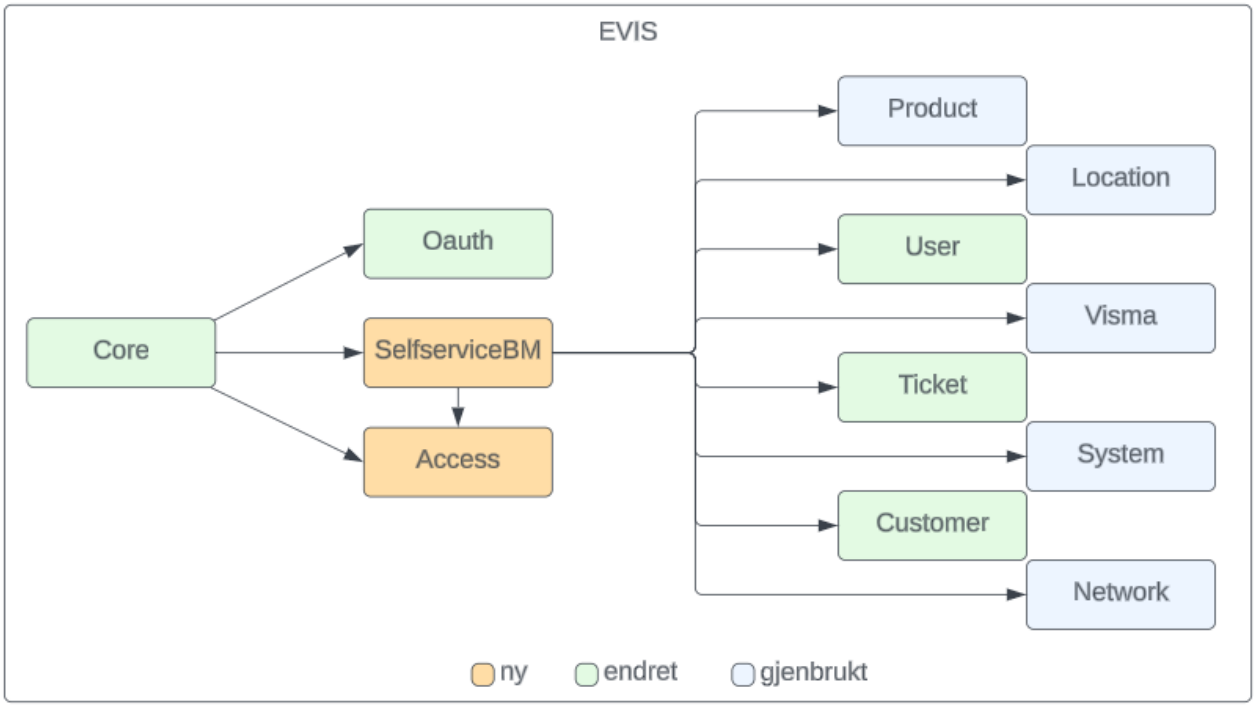
2 ARKITEKTUR

På høyeste nivå kan løsningen deles inn i komponenter som vist i Figur 2-1. Miside BM er applikasjonens brukergrensesnitt og kjører i brukerens nettleser. Den sender spørringer mot EVIS og får data tilbake. I EVIS handteres alt av forretningslogikk. Når den mottar en spørring, utfører den validering og annen logikk før den henter eller oppdatere data i databasen og til slutt sender en respons til frontend-komponenten.



Figur 2-1: Overordnet arkitekturskisse

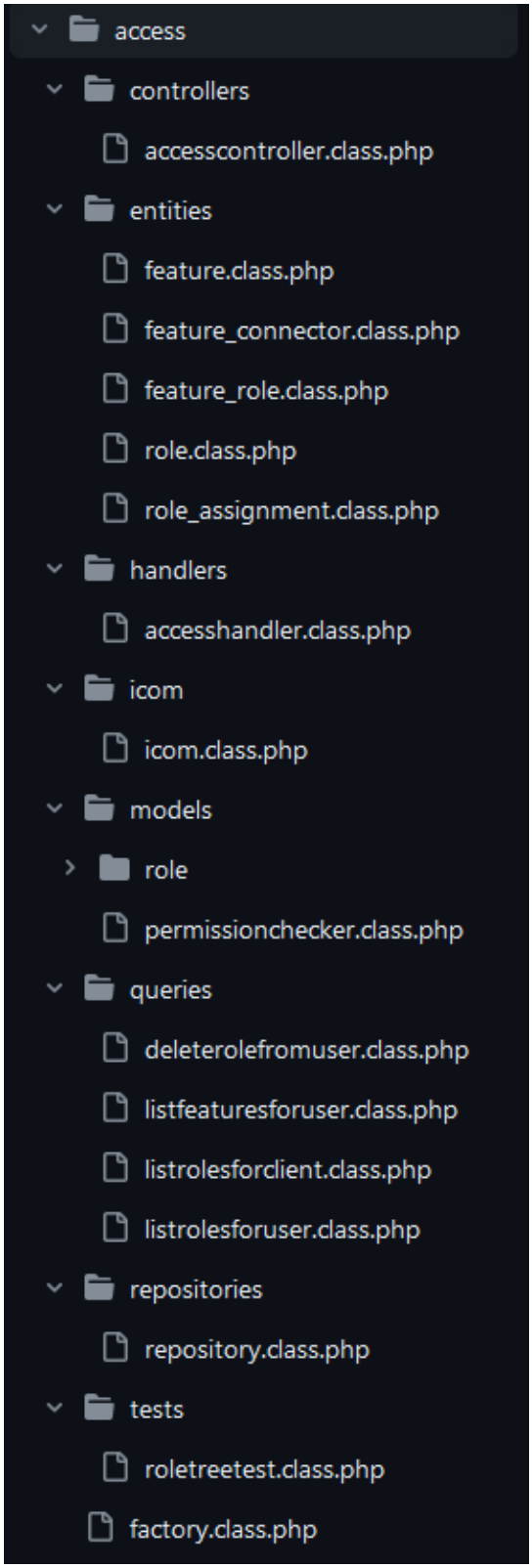
EVIS består av flere ulike komponenter. Figur 2-2 viser de komponentene som er involvert i løsningen. Alle endepunkt som blir tilgjengeliggjort for klienten er definert i SelfserviceBM-komponenten. Denne kommuniserer med andre komponenter i systemet. Før en spørring kommer til SelfserviceBM er den innom Core, som blant anna handterer ruting, autentisering og autorisering.



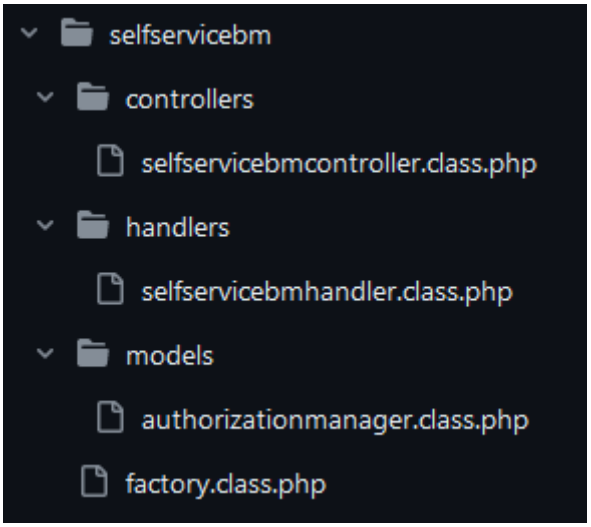
Figur 2-2: Serverarkitekturskisse EVIS: komponenter involvert i løsningen

3 PROSJEKTSTRUKTUR

3.1 Backend

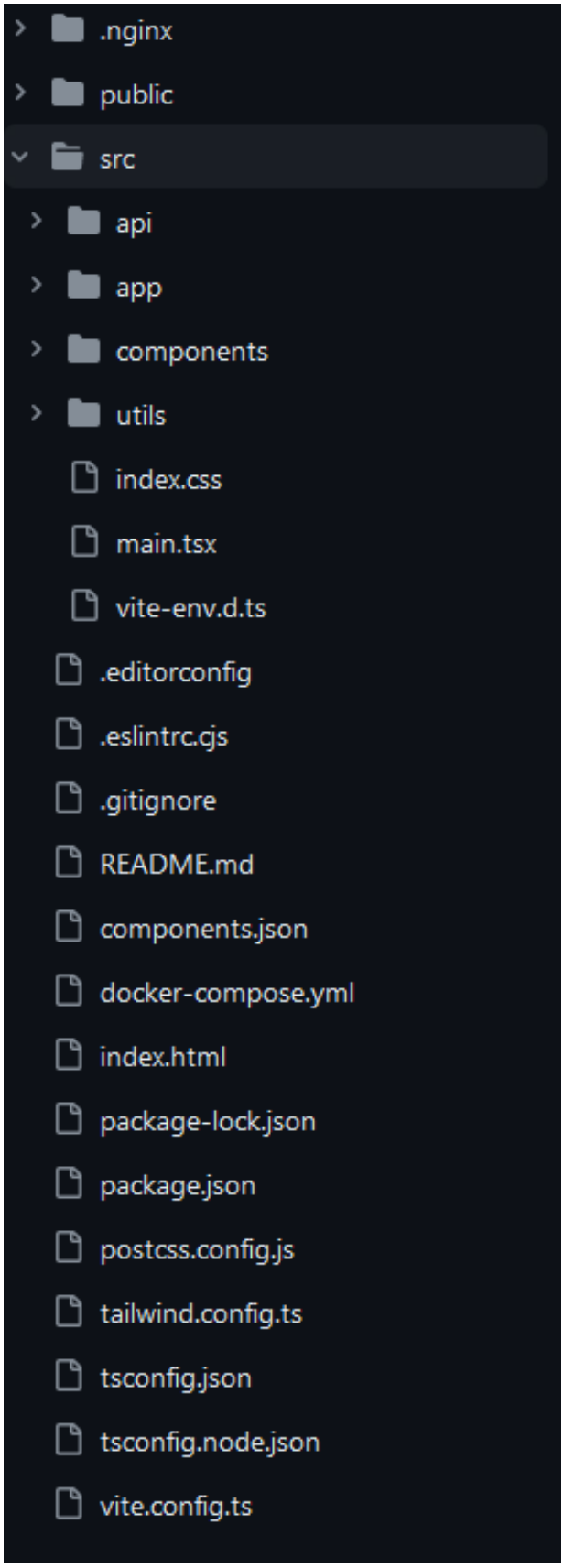


Figur 3-1: Access-komponent filstruktur

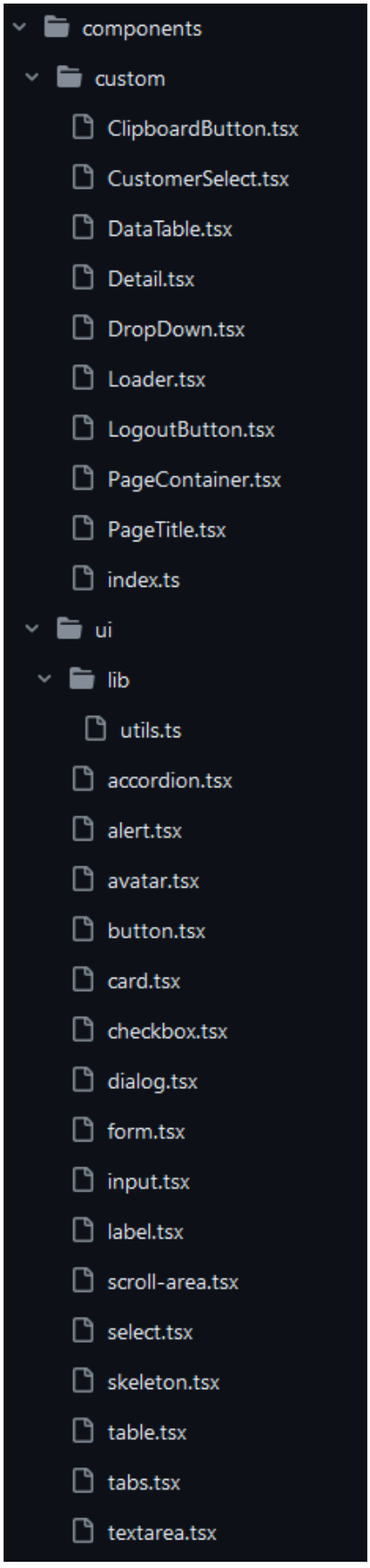


Figur 3-2: Selfservicebm-komponent filstruktur

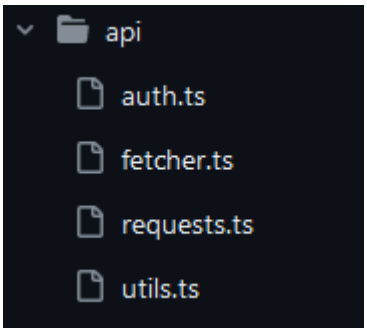
3.2 Frontend



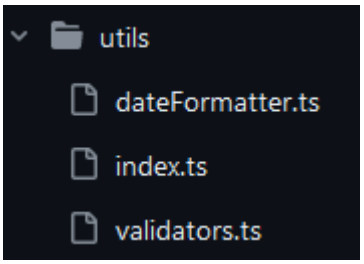
Figur 3-3: Frontend overordnet filstruktur



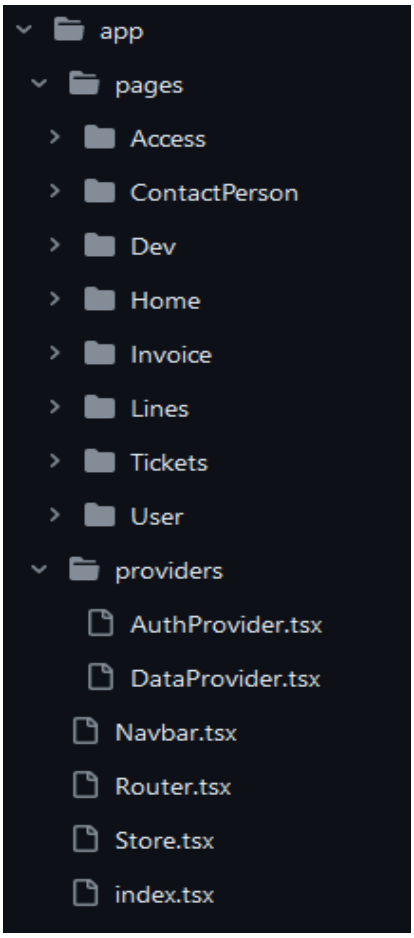
Figur 3-4: Frontend src/components



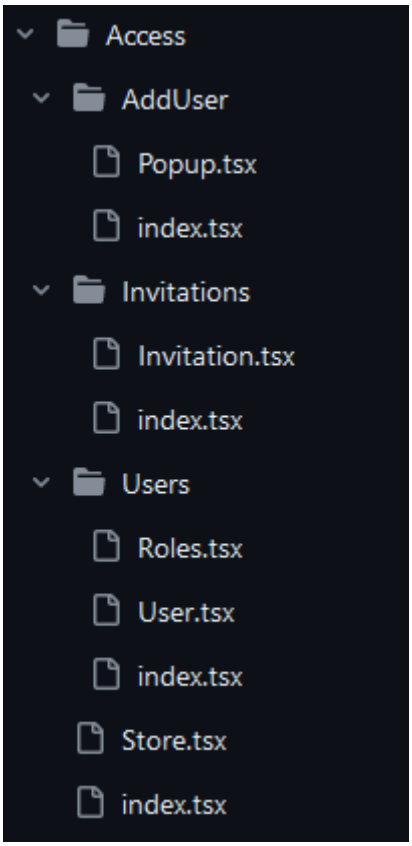
Figur 3-5: Frontend src/api



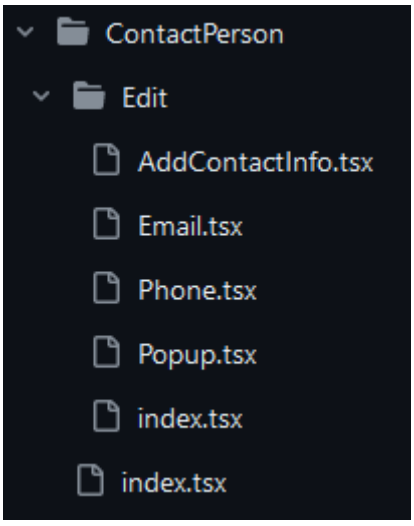
Figur 3-6: Frontend src/utils



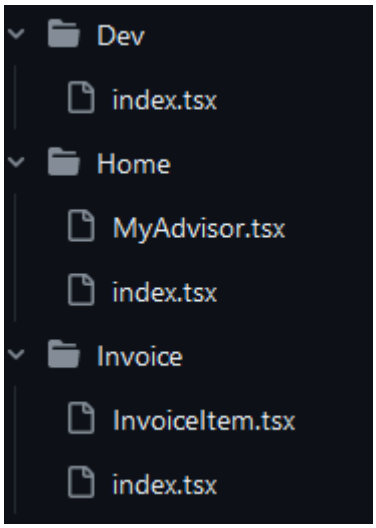
Figur 3-7: Frontend src/app



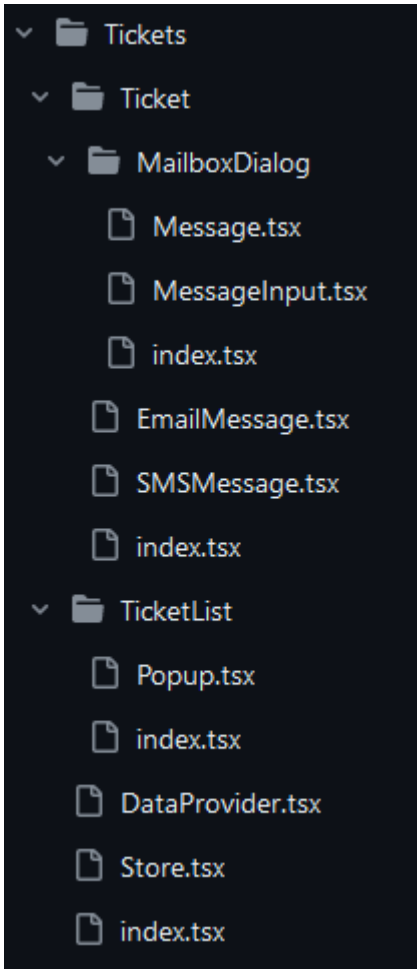
Figur 3-8: Frontend src/app/pages/Access



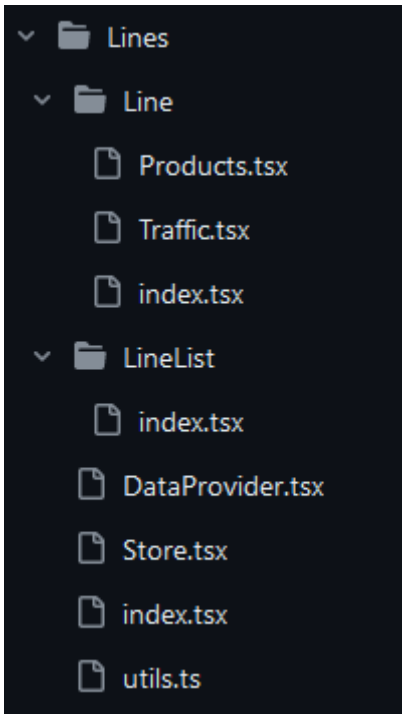
Figur 3-9: Frontend src/app/pages/ContactPerson



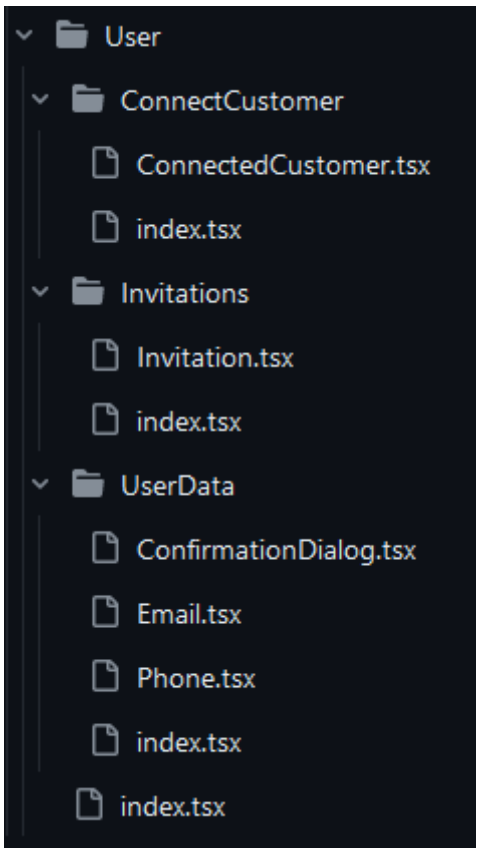
Figur 3-10: Frontend src/app/pages/...



Figur 3-11: Frontend src/app/pages/Tickets

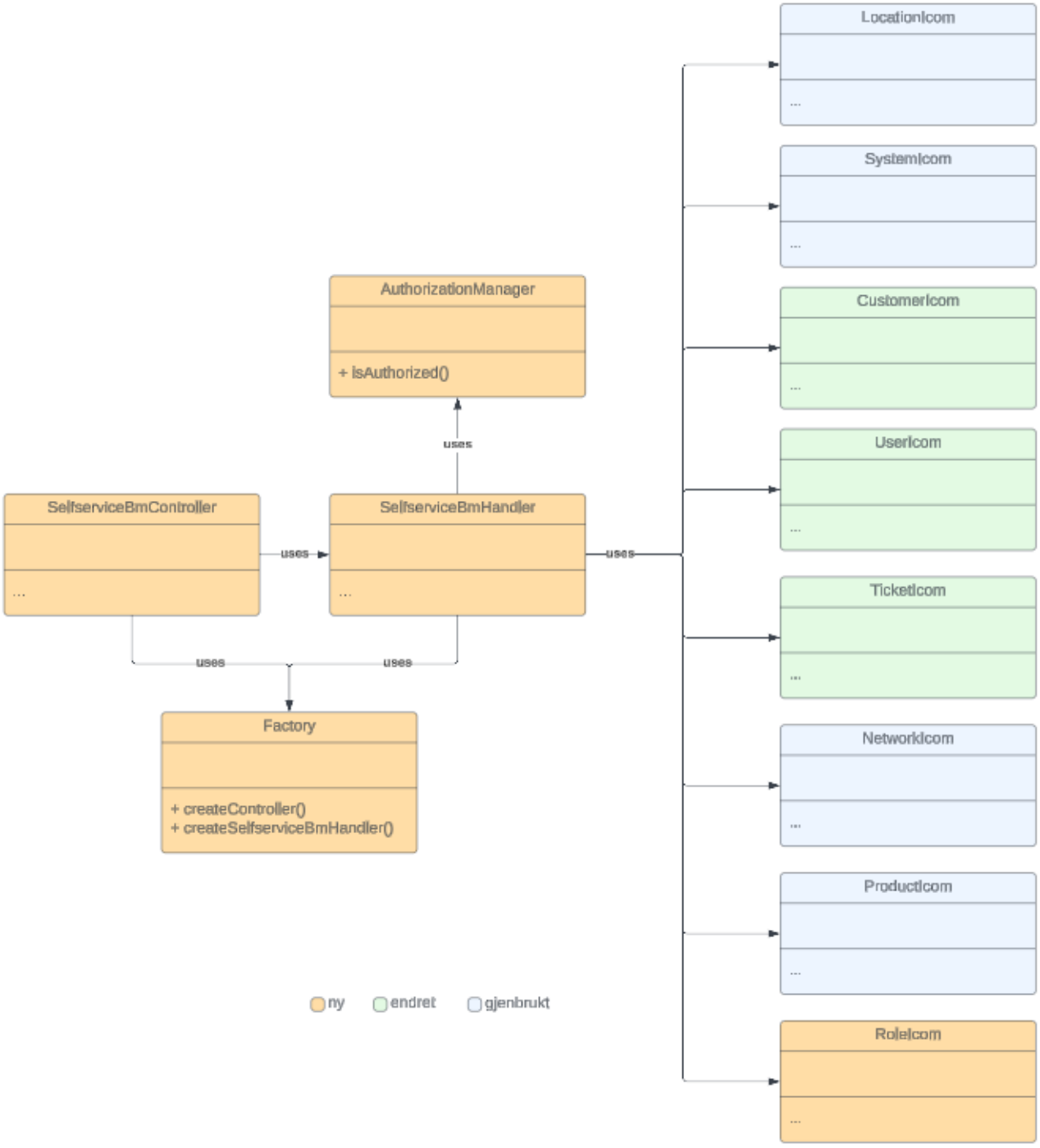


Figur 3-12: Frontend src/app/pages/Lines

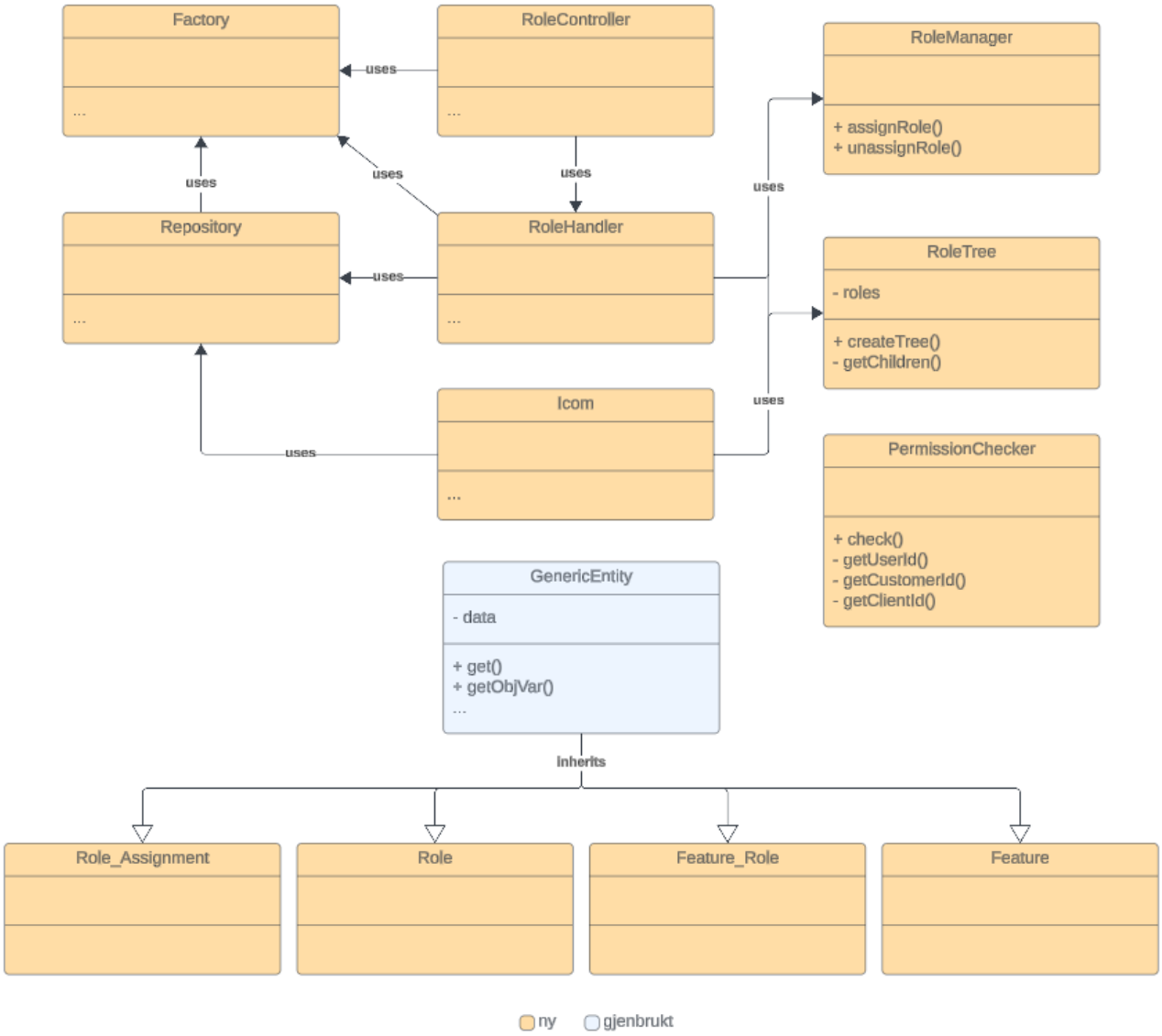


Figur 3-13: Frontend src/app/pages/User

4 KLASSEDIAGRAM

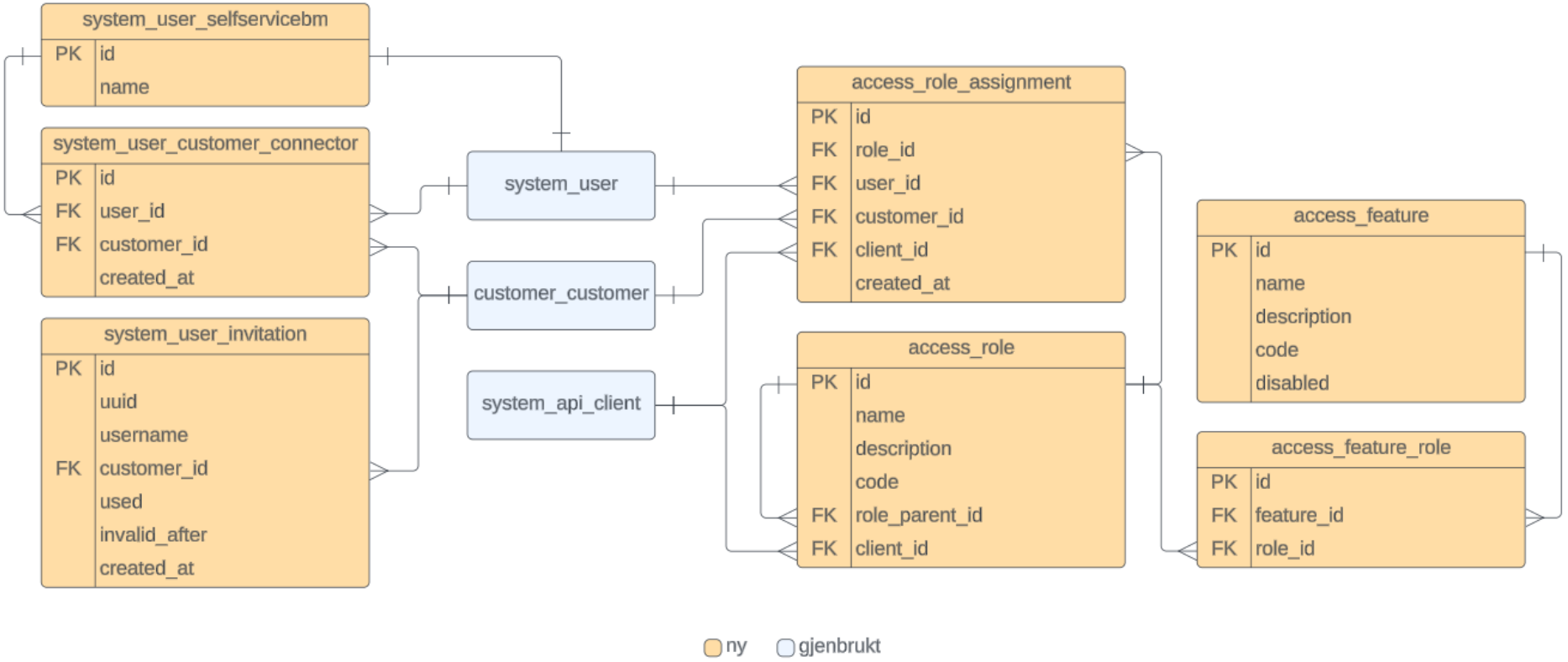


Figur 4-1: Klassediagram Selfservicebm



Figur 4-2: Klassediagram Access

5 DATABASEMODELL



Figur 5-1: Databasemodell viser nye tabeller og hvordan de er koblet til eksisterende tabeller

6 SERVER-TJENESTER

Liste over alle endepunkter laget for løsningen:

User

- getUser: henter ut brukeren
- getConnectedCustomers: henter kunder som er knyttet til brukeren
- deleteCustomerConnection: bruker fjerner kobling mellom seg og kunde
- getUserFeatures: lister ut funksjonalitet som er tilgjengelig for bruker
- postUserEmail: endre brukers epostadresse
- postUserPhone: endre brukers telefonnummer
- postContactInfoOtp: bekreft engangskode ved endring av epost eller telefonnummer

Customer

- getCustomer: henter kunden
- getConnectedUsers: henter brukere koblet til kunden
- deleteUserConnection: fjern kobling mellom kunden og en bruker
- getPortfolioOwner: henter ut kundens rådgiver

Invitation

- getInvitations: henter invitasjoner knyttet til en kunde
- postInvitation: sender en invitasjon til en bruker
- deleteInvitation: sletter en sendt invitasjon
- getUserInvitations: henter invitasjoner sendt til en bruker
- postUserAcceptInvitation: aksepter invitasjon

Role

- getRoleTree: henter rolle-treet fra databasen
- getUserRoles: henter de ulike rollene en gitt bruker har for en kunde
- postUserRole: sender endring til ny rolle for en bruker til en kunde

- deleteUserRole: fjerner en rolle fra en gitt bruker til en kunde

Invoice

- getInvoices: henter ut faktura
- getInvoiceByKey: hent en faktura pdf

Ticket

- getTickets: henter saker koblet til en kunde
- getTicketMessages: henter ut tekstmeldinger og eposter sendt for en gitt sak
- getMailboxDialog: henter ut en meldingsdialog på en sak
- getSubjectCategories: henter meldingskategorier for opprettelse av ny sak
- postAppMessageInNewTicket: opprettelse av en ny sak med ny melding
- postAppMessage: legg til ny melding på sak
- postMarkMessagesAsRead: markerer meldinger som lest

Contact Person

- getCustomerChildren: list ut kontaktpersoner
- postContactPersonEmail: endre kontaktperson epost
- postContactPersonPhone: endre kontaktperson telefonnummer

Lines

- getContracts: hent ut kontrakter/linjer
- getContract: hent en kontrakt/linje
- getTrafficGraphs: hent trafikkgraf for linje

7 SIKKERHET

7.1 Access token

For å få tilgang til api-et må en bruker ha en gyldig jwt-token (Auth0, (u.å.)). Tokenet inneholder informasjon om hvem brukeren er, og hvilke tilganger den har. Denne typen token er kryptografisk sikret og kan ikke endres på av en angriper. Flyten for tildeling av tokens følger OAuth2 protokollen (OAuth 2.0, (u.å.)), som er industristandard for autorisasjon. Denne løsningen er ikke utviklet i dette prosjektet.

7.2 SQL-injection

Alle spørringer mot databasen som inneholder brukerinput er parameterisert. Dette blir gjort med php-biblioteket PDO (The PHP Group, (u.å.)). Ved å bruke denne teknikken vil databasen kunne skille mellom hva som er spørring og hva som er data. Dette forhindrer angrep i form av SQL-injections.

7.3 Cross-site scripting

All input-data blir sendt igjennom en validator som både sikrer at dataen er korrekt med tanke på de reglene som satt for endepunktet og at dataen ikke inneholder elementer som kan brukes til cross-site scripting. Validatoren nytter php-biblioteket GUMP (Wixel, (u.å.)).

8 INSTALLASJON OG KJØRING

Når løsningen skal settes i produksjon er det flere steg som må gjøres. EVIS, som er bygd på LAMP-stacken (IBM, (u.å.)), må kjøre for at løsningen skal fungere. Oppsett og konfigurasjon av dette er utenfor omfanget til dette dokumentet, men nødvendig. Applikasjonen Miside bedrift er konfigurert til å kjøre i en docker-container gjennom docker compose (Docker, (u.å.)). Først må alle avhengigheter installeres. Dette gjøres gjennom Node Package Manager (NPM (u.å.)) og kommandoen «npm install». Når alle pakkene er installert må man bygge applikasjonen. Dette gjøres med kommandoen «npm run build». Til slutt kjører man applikasjonen med kommandoen «docker compose up -d». Dette starter opp en docker-container med en NGINX-server (ref) som hoster appen. Figur 8-1 viser alle kommandoene som må kjøres, i rekkefølge.

```
$ npm install
$ npm run build
$ docker compose up -d
```

Figur 8-1: Kommandoer for kjøring av Miside bedrift

9 TESTING

I koden er det brukt enhetstester for noen deler av systemet der det lar seg gjøre. Denne typen tester har som mål å verifisere at koden som testes, fungerer som forventet. Dette er nyttig for å sikre at koden gjør som den skal nå og at den fortsetter å gjøre som den skal når kravene knyttet til koden, og dermed koden, endrer seg.

Klassen med ansvar for å konstruere et tre fra en liste med roller, er et eksempel i fra koden som er dekket av enhetstester. Figur 9-1 viser en av enhetstestene som dekker klassen. Testen lager til en liste med roller, som den fører inn i klassen med ansvar for å bygge treet. Der blir det utført en algoritme og et tre blir returnert. Til slutt validerer den at treet som blir konstruert tilfredsstiller forventningen til hvordan det skal være strukturert basert på dataen ført inn i algoritmen.

```
1  public function testBasicTree() {
2      $roles = [
3          (new Role())->setWithArray(["id" => 1, "role_parent_id" => null]),
4          (new Role())->setWithArray(["id" => 2, "role_parent_id" => 1]),
5          (new Role())->setWithArray(["id" => 3, "role_parent_id" => 2]),
6      ];
7
8      $roletree = (new RoleTree($roles))->createTree();
9
10     $this->assertIsArray($roletree);
11     $this->assertCount(1, $roletree);
12     $this->assertCount(1, $roletree[0]->get("children"));
13     $this->assertCount(1, $roletree[0]->get("children")[0]->get("children"));
14 }
```

Figur 9-1: Utdrag fra enhetstester – testBasicRoleTree

10 KODE

Kode utviklet gjennom prosjektet er tilgjengelig under sensur gjennom Github-lenken under, dersom man logger inn med brukernavn og passord spesifisert.

Github-lenke: https://github.com/EnivestAS/misidebm-bachelor-kode

Github-brukernavn: sensorbachelor2024

Github-passord: PassordForSensor2024

11 REFERANSER

Auth0 (u.å.) *JSON Web Tokens are an open, industry standard RFC 7519 method for representing claims securely between two parties.* Tilgjengelig fra: https://jwt.io/ (Hentet: 29. april 2024)

OAuth 2.0 (u.å.) *OAuth 2.0* Tilgjengelig fra: https://oauth.net/2/ (Hentet: 29. april 2024)

The PHP Group (u.å.) *MySQL Functions (PDO_MYSQL)* Tilgjengelig fra: https://www.php.net/manual/en/ref.pdo-mysql.php (Hentet: 29. april 2024)

Wixel (u.å.) *GUMP* Tilgjengelig fra: https://github.com/Wixel/GUMP (Hentet: 29. april 2024)

IBM (u.å.) *What is the LAMP stack* Tilgjengelig fra: https://www.ibm.com/topics/lamp-stack (Hentet: 01. mai 2024)

Docker (u.å.) *Docker Compose overview* Tilgjengelig fra: https://docs.docker.com/compose/ (Hentet: 01. mai 2024)

NPM (u.å.) *Build amazing things* Tilgjengelig fra: https://www.npmjs.com/ (Hentet: 01. mai 2024)

NGINX (u.å.) *What is NGINX?.* Tilgjengelig fra: https://www.nginx.com/resources/glossary/nginx/ (Hentet: 01. mai 2024).