Høgskulen på Vestlandet

BACHELOROPPGAVE

Tofaktorautentisering (2FA) ved bruk av Tidsbasert engangspassord (TOTP) i MinID

Two-factor authentication (2FA) using time-based onetime password (TOTP) in MinID.

Jørgen Vikan Eidsvåg, Ulrik Fagerberg og Erik Fiksdal

Fakultet for ingeniør- og naturvitenskap
Institutt for datateknologi, elektroteknologi og realfag

Informasjonsteknologi
Veileder: Per Christian Engdal
Innleveringsdato: 22/05/2023

Høgskulen på Vestlandet

Fakultet for ingeniør- og naturvitskap

Institutt for datateknologi, elektroteknologi og realfag

TITTELSIDE FOR HOVEDPROSJEKT

Rapportens tittel:	*Dato:*
Tofaktorautentisering (2FA) ved bruk av Tidsbasert engangspassord (TOTP) i MinID	*22. mai 2023*
Forfatter(e):	*Antall sider u/vedlegg: 57*
Jørgen Vikan Eidsvåg, Ulrik Fagerberg og Erik Fiksdal	*Antall sider vedlegg: 78*
Studieretning:	*Antall disketter/CD-er:*
Informasjonsteknologi	0
Kontaktperson ved studieretning:	*Gradering:*
Per Christian Engdal	Ingen
Merknader:	
Ingen	

Oppdragsgiver:	*Oppdragsgivers referanse:*
Digitaliseringsdirektoratet	Ingen
Oppdragsgivers kontaktperson:	*Telefon:*
Hallvard Bjørdalsbakke	41749924

Sammendrag:

Prosjektet handler om å utforske og implementere en løsning for tredjeparts 2FA-TOTP applikasjoner for autentisering, i MinID. Løsningen er en implementasjon basert på spesifikasjonene i RFC6238 og tar hensikt i sikkerhet, brukermasse og kompatibilitet hos de mest populære autentiseringsapper våren 2023.

This project covers the research and implementation of a solution for a third-party 2FA-TOTP application for authentication, in MinID. The solution is an implementation based on the specifications in RFC6238 and considers security, userbase and compatibility with the most popular authentication apps during spring 2023.

Stikkord:

MinID	Two Factor Authentication (2FA)	Time-based onetime password (TOTP)

Høgskulen på Vestlandet, Fakultet for ingeniør- og natuvitskap
Postadresse: Postboks 7030, 5020 BERGEN Besøksadresse: Inndalsveien 28, Bergen
Tlf. 55 58 75 00 Fax 55 58 77 90 E-post: post@hvl.no Hjemmeside: http://www.hvl.no

FORORD

Følgende rapport er vært laget i sammenheng med bachelor oppgave i å innføre Tofaktorautentisering (2FA) ved bruk av Tidsbasert engangspassord (TOTP) i MinID. Prosjektet er utført av Jørgen Vikan Eidsvåg, Ulrik Fagerberg og Erik Fiksdal ved Høgskulen på Vestlandet (HVL), våren 2023.

Takk til Bengt Edvardsen og Hans Jakob Reite for at de har stilt med arbeidslokaler og utstyr for bachelorprosjektet.

Takk til Hallvard Bjørdalsbakke og Jørn Stenehjem ved DigDir for godt samarbeid og god kommunikasjon.

Takk til Per Christian Engdal som har vært prosjektgruppens veileder gjennom hele prosjektet med høyt engasjement, positiv innstilling og villighet til å svare på spørsmål ved alle døgnets tider.

INNHOLDSFORTEGNELSE

Ordliste

Tabell 1 Ordliste

2FA	Tofaktorautentisering (Two-Factor-Authentication)
DigDir	Digitaliseringsdirektoratet
Hash	En streng med symboler, endret på en deterministisk måte.
HOTP	Engangskode-generering ved å telle opp for hver generert kode.
KRR	Kontakt- og reservasjonsregisteret
MFA	Multi-Factor-Authentication(Samlebegrep for autentisering med bruk av flere enn en faktor)
MinID-app	DigDirs sin egenutviklede autentiseringsapplikasjon
OTC	Engangskode (One Time Code)
Tofaktorautentisering	En bekreftelse på hvem bruker er som følge av innlogging vha. f.eks. kode på epost eller mobil. (Totrinnsbekreftelse)
TOTP	Tidsbasert engangskode (Time-based onetime password)

1 INNLEDNING

1.1 Kontekst

Digitaliseringsdirektoratet (DigDir) leverer en tjeneste med navn ID-porten, som tilrettelegger for innlogging på nettbaserte offentlige tjenester. I dag er nesten alle innlogginger gjennom ID-porten utført med BankID. DigDir leverer sin egen løsning for innlogging kalt MinID, men denne er ikke nevneverdig brukt sammenlignet med BankID. For å logge inn med MinID må man i dag enten skrive inn en kode mottatt på SMS eller bruke DigDir sin egen app for autentisering.

MinID-appen er lite i bruk som autentiseringsmetode, og SMS ansees ikke å være like sikker som en app-basert autentisering (Edwards, 2022). DigDir har som ønske å utforske hvordan brukere benytter seg av en *Two-Factor-Authentication* (2FA) *Time-Based onetime password* (TOTP) app som brukere har fra før, til autentisering med MinID. Dette kan igjen føre til at flere tar i bruk MinID som innlogging med ID-porten.

1.2 Motivasjon

Tall fra DigDir (personlig kommunikasjon 21 februar 2023) for januar 2023 viser at ID-porten hadde over 16 millioner unike påloggинger i perioden, hvor BankID sto for over 95% av disse. Det vil si at et fåtall av innlogginger med ID-porten brukte MinID. Dette er tall som DigDir ønsker å endre, siden MinID er en tjeneste de selv leverer.

Det finnes instanser hvor en bruker er nødt til å komme opp i høyeste sikkerhetsnivå (nivå 4). I disse tilfellene er ikke MinID god nok siden det er på sikkerhetsnivå: “betydelig” (nivå 3). Det kan for eksempel handle om å lese helseopplysninger, og en bruker vil i disse tilfellene måtte autentisere med BankID, Commfides eller Buypass da alle disse er av høyeste sikkerhetsnivå. Men i mange tilfeller vil MinID være av tilstrekkelig sikkerhetsnivå for en bruker sitt behov.

DigDir har i kontakt med prosjektgruppen uttrykt et ønske om at MinID skal opp på høyeste sikkerhetsnivå i fremtiden, men det ligger en del endringer til grunn for at det skal skje. Blant annet har MinID fjernet muligheten til å autentisere pålogging med PIN-koder fra 16 januar 2023.

Hva som ligger til grunn for at fåtallet bruker MinID til pålogging er det lite kjennskap til. DigDir har et ønske om å finne ut hvordan de skal implementere 2FA-TOTP ved bruk tredjeparts autentiseringsapplikasjoner. Dette vil gi brukerne større valgfrihet ved at de kan nytte den autentiserings-appen de selv ønsker.

1.3 Prosjekteier

Prosjekteier er DigDir, som er en norsk statlig organisasjon med rundt 330 ansatte fordelt på kontorer i Brønnøysund, Leikanger og Oslo. Direktoratet ble formelt opprettet 1. januar 2020 og har som mål å bidra til modernisering og digitalisering av offentlig sektor i Norge. DigDir er underlagt Kommunal- og distriktsdepartementet og har ansvar for å koordinere og lede digitaliseringsarbeidet på tvers av offentlige virksomheter.

DigDir har blant annet ansvaret for å utvikle og implementere felles nasjonale IKT-løsninger, herunder MinID for autentisering, standarder og retningslinjer for offentlige virksomheter, samt å bidra til å sikre personvern og informasjonssikkerhet i digitaliseringen av offentlige tjenester.

Direktoratet har også en rolle som rådgiver og samarbeidspartner for andre offentlige virksomheter, og jobber tett med både kommuner, fylkeskommuner og statlige etater for å bidra til en mer effektiv og brukervennlig offentlig sektor i Norge. (Digitaliseringsdirektoratet, n.d)

1.4 Problembeskrivelse og mål

DigDir ønsker at et større antall brukere benytter seg av MinID for autentisering. Dagens løsning tilbyr 2FA ved bruk av SMS-kode og proprietær app, men ønsker å tilby flere, sikre og kjente autentiseringsmetoder. DigDir vurderer å implementere 2FA-TOTP ved bruk av utbredte og anerkjente tredjeparts autentiseringsapplikasjoner, som kan senke terskel for bruk av MinID og treffe flere brukere. Det er viktig for DigDir å kunne tilby en rask autentiseringstjeneste, som ikke er for ressurskrevende å implementere og vedlikeholde.

Under prosjektets gang er det ønsket å finne svar på dette spørsmålet:

Hvordan kan man implementere en løsning for autentisering ved bruk av 2FA-TOTP, og tredjeparts autentiseringsapplikasjoner i MinID.

1.5 Oppbygging av rapporten

Kapittel 2 tar for seg detaljer rundt prosjektet som løsningen vil måtte utvikles rundt, faktorer som oppdragsgivers krav, foreslåtte løsning og resursene prosjektgruppen vil ha til rådighet.

Kapittel 3 går inn i detaljer på den tidlige løsningen, samt planleggingen og metodikken som inngår i arbeidet opp imot denne løsningen.

Kapittel 4 vil presentere løsningen i mer detaljer og hva som ligger i fremgangen til- og bakgrunner for den konkluderte løsningen.

Kapittel 5 vil gå inn i resultatene, hvordan de evalueres underveis og hvilke resultater evalueringen gir.

Kapittel 6 skal ta for seg valgene under prosjektet og hvilken innvirkning de har på det endelige resultatet, spesielt hvor det angår forbedringspotensialer.

Kapittel 7 skal ta for seg de viktigste refleksjonene og anbefalinger for alle parter som prosjektet har relevans for.

2 PROSJEKBESKRIVELSE

2.1 Praktisk bakgrunn

2.1.1 Tidligere arbeid

MinID ble lansert mot slutten av 2008 (Fornyings- og administrasjonsdepartementet, 2008) for å gjøre det enklere for norske skattebetalere å få tilgang til offentlige nettsteder. Den tid var det to alternativer for autentisering, PIN-kodebrev og SMS. DigDir lanserte i 2021 en egen app som hadde som mål å erstatte disse alternativene. 16 januar 2023 ble PIN-kodebrev fjernet som alternativ for autentisering og der er per nå MinID-app og SMS som er gyldige autentiseringsmetoder for MinID. Det er tidligere ikke gjort noen nøye analyse og vurdering i DigDir relatert til innføring av 2FA-TOTP.

2.1.2 Initielle krav

I utgangspunktet hadde DigDir som krav at løsningen skulle være en endring på MinID-innstillinger.

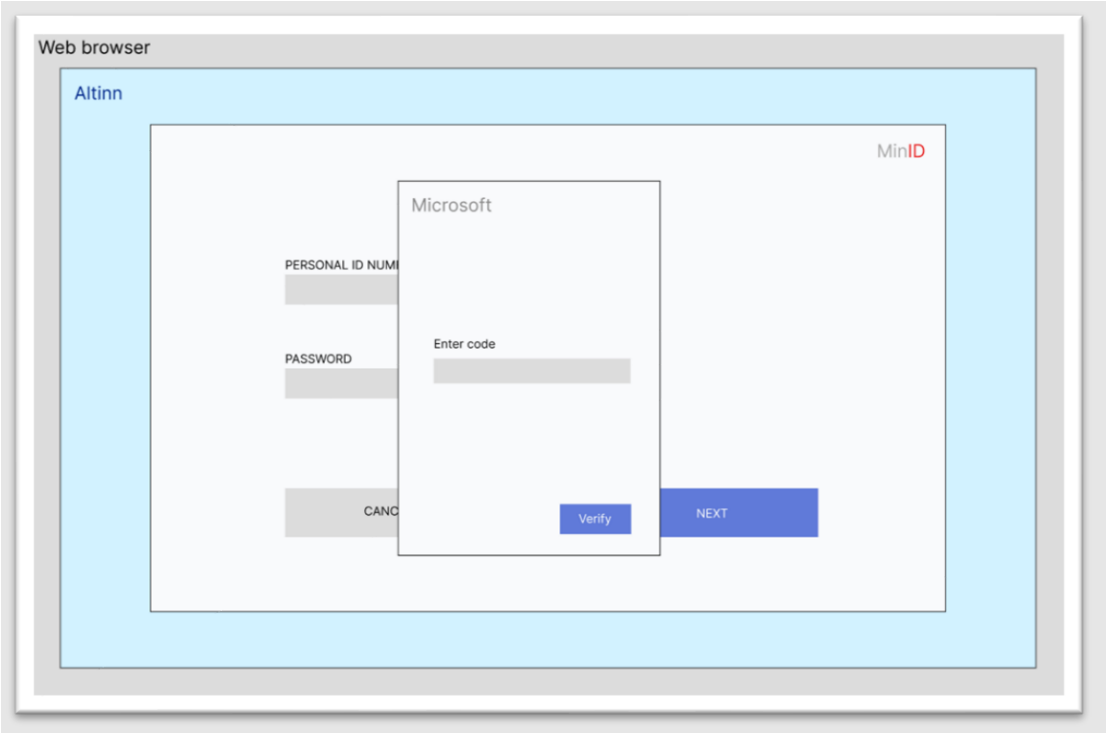
Oppgaven har som en del av tidlig fase etter avklaringer blitt utvidet til å omhandle det å tilby 2FA-TOTP autentisering ved bruk av tredjeparts autentisering til sine brukere. Dette i tillegg til eksisterende autentiseringsløsninger som MinID-app og SMS. Det skal også tas med endringer av MinID-innstillingene, hvor det skal ikke være mulig å endre mobilnummer, da dette blir hentet fra Kontakt- og reservasjonsregisteret (KRR). Innlogging ved hjelp av PIN skal ikke kunne velges som autentiseringsalternativ under MinID-innstillinger, da dette ble utfaset fra januar 2023. I tillegg skal det innføres muligheten for at brukere kan avregistrere MinID-app som autentiseringsalternativ.

Løsningen skal bygges og demonstreres som en prototyp, som er mest mulig uavhengig av MinID sin infrastruktur og systemer.

Løsningen skal være enkel å sette opp og ta i bruk, og bruker skal kunne velge denne løsning på samme måte som andre autentiseringsløsninger via MinID-innstillinger.

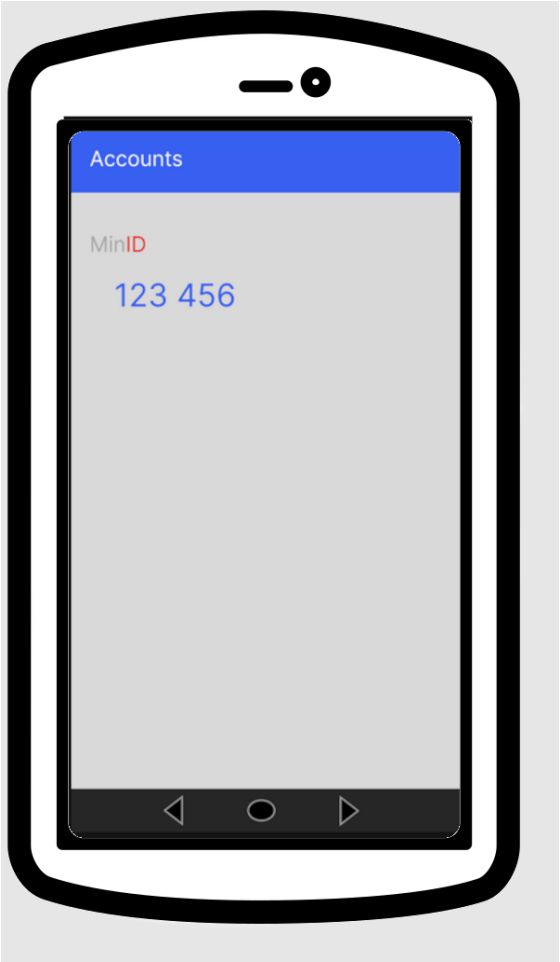
2.1.3 Initiell løsnings-idé

Initiell løsnings-idé er å tilby en bruker av MinID, å velge 2FA-TOTP som autentisering ved bruk av tredjeparts autentiseringsapplikasjon, som for eksempel Google Authenticator, som en del av MinID-innstillinger. Her skal bruker ved førstegangsbruk, kunne registrere og ta i bruk ny autentiseringsapplikasjon. Når brukeren velger å logge på ved bruk av MinID neste gang, vil 2FA-TOTP med valgt autentiseringsapplikasjon bli presentert.



Figur 1 Illustrasjon av brukermiljø

Figur 1 viser en tidlig illustrasjon av tenkt brukermiljø for innlogging med 2FA-TOTP. Det vil fungere nesten identisk med dagens løsning for SMS, hvor en bruker må skrive inn personnummer og passord, for å så sendes til en side hvor bruker må skrive inn generert kode for verifisering. Dersom kode stemmer overens med systemets kode vil en bruker bli innlogget.



Figur 2 TOTP-kode fra mobil

2.2 Avgrensninger

Det er gjort følgende avgrensninger i prosjektet:

- Gjenbruk av eksisterende design og stiler for brukergrensesnitt.
- Basere all utvikling på Java-17, Spring Boot og Thymeleaf som er gjeldene språk og rammeverk hos DigDir.
- Basere seg på gjeldene databasestrukturer og API, så langt det er hensiktsmessig for prosjektet.

Skulle det bli behov for å prioritere funksjonalitet, skal løsning for 2FA-TOTP autentisering ved bruk av tredjeparts autentiseringsapplikasjon ha høyeste prioritet foran oppdatering av MinID-Innstillinger.

2.3 Ressurser

Prosjektgruppen har fått tilgang til de tekniske ressurser som de mener er nødvendige ut ifra den planlegging som er utført. Oppdragsgiver har sendt Cascading Style Sheet (CSS) filer som MinID bruker, slik at prosjektgruppen har muligheten til å bruke denne i deres løsning. I tillegg til dette har oppdragsgiver sendt noe Structured Query Language (SQL)-kode som gir prosjektgruppen ett innblikk i hvordan databasestruktur oppdragsgiver bruker. Det vil føre til at prosjektgruppen har muligheten til å lese ut ifra denne koden å

bygge deres database med en struktur som bruker de relevante tabeller og verdier. Oppdragsgiver har også levert noe statistikk om antall innlogginger hos ID-porten og hvilke tjenester som er benyttet.

Prosjektgruppen har tilgang til kontorplasser hos Høgskulen på Vestlandet, Campus Verftet.

Videre har prosjektgruppen Per Christian Engdal som intern veileder ved HVL, som har stilt seg tilgjengelig for tilbakemeldinger og veiledning for prosjektet. Hallvard Bjørdalsbakke har bidratt med tilbakemeldinger, kompetanse og vært prosjektgruppens kontaktperson hos prosjekteier.

2.4 Litteratur om problemstillingen

TOTP er en vanlig form for tofaktorautentisering. Unike 6-siffrede passord genereres for et gitt tidsintervall (for eksempel 30 sekunder) og brukes som et ekstra lag med sikkerhet i tillegg til brukernavn og passord ved innlogging (Twilio, n.d). TOTP er en utvidelse av HMAC-based One-Time Password (HOTP) og bruker en tidsverdi som del av algoritmen, istedenfor en hendelses-teller (M'Raihi *et al.,* 2011).

Det sekssifrede passordet genereres ved hjelp av en kryptografisk autentiseringsteknikk som heter Hash-Based Message Authentication Codes (HMAC). Ved å gi HMAC en tidsverdi og en gitt hashfunksjon genereres en hash som algoritmen kan bearbeide videre til det endelige engangspassordet. I TOTP sitt tilfelle kan hashfunksjoner som SHA-1, SHA-256 og SHA-512 brukes i denne sammenhengen.

For informasjon til fordypning rundt TOTP, HOTP og HMAC vil det anbefales å lese de tilhørende dokumentene hos rfc-editor.org

TOTP - RFC6238

"*TOTP: Time-Based One-Time Password Algorithm*" av M'Raihi, D., Machani, S., Pei, M., og J. Rydell går inn i detaljen på TOTP, med krav, begrensinger og anbefalinger til implementeringen. (M'Raihi *et al.*, 2011)

HOTP - RFC4226

TOTP bygger på grunnlag av HOTP som dekkes i dokumentet for RFC4226: "*HOTP: An HMAC-Based One-Time Password Algorithm*" skrevet at M'Raihi, D., Bellare, M., Hoornaert, F., Naccache, D., og O. Ranen (M'Raihi *et al.*, 2005)

HMAC – RFC2104

HMAC-algoritmen ligger til grunn for både HOTP og TOTP, videre informasjon om dette finnes under "*HMAC: Keyed-Hashing for Message Authentication*" av Krawczyk, H., Bellare, M., og R. Canetti. (Krawczyk, Bellare og Canetti, 1997)

3 DESIGN AV PROSJEKTET

3.1.1 Forslag til løsning

Ved produktets utvikling må noen avgjørelser tas angående valg av tredjeparts autentiseringsløsning og hashing algoritmer for implementasjon. Det er et stort antall tredjeparts 2FA-TOTP apper på markedet. Appene med størst utbredelse vil bli trukket frem og vurdert separat.

3.1.2 2FA-TOTP apper

3.1.2.1 Google Authenticator

Tall fra Google viser at Google Authenticator er 2FA-TOTP appen med flest nedlastninger på global basis med over 100 millioner nedlastninger. Appen har 437 tusen brukeranmeldelser, hvor nylige anmeldelser har gitt appen 3,6 av 5 stjerner (Google authenticator, n.d).

3.1.2.2 Microsoft Authenticator

Etter Google Authenticator kommer Microsoft Authenticator med over 50 millioner nedlastninger. Appen har 1,19 millioner brukeranmeldelser, med 4,7 av 5 stjerner ut ifra nylige anmeldelser (Microsoft authenticator, n.d).

3.1.2.3 Twilio Authy Authenticator (Authy)

Authy kommer sist blant de tre mest populære appene med over 10 millioner nedlastninger og 58900 brukeranmeldelser mellom 2016 og 2023. Ut ifra nylige anmeldelser har Authy 4,4 av 5 stjerner (Twilio Authy authenticator, n.d).

3.1.3 Sikre hashing algoritmer

TOTP er en utvidelse av HOTP (M'Raihi *et al.,* 2005) som er basert på HMAC-SHA-1 algoritmen (Krawczyk, Bellare, og Canetti, 1997). Ved implementasjon av TOTP kan SHA-256 eller SHA-512 hashing funksjonene velges som alternativer til SHA-1 (M'Raihi, *et al.,* 2011). SHA-1 er ikke lengre sett på som kryptografisk sikkert for bruk ved digital signering (Stevens, *et al.,* 2017), men forblir relevant innenfor One-Time Password bruk. Den genererte hash verdien blir videre forkortet for å produsere en 6-sifret kode, noe som gjør jobben til en eventuell angriper mye vanskeligere (Willoughby, 2005). SHA-1 er standard for HMAC implementasjon per dags dato.

3.1.4 Diskusjon

3.1.4.1 2FA-TOTP apper

Ved utvikling av en ny MinID autentiseringsløsning er det viktig at produktet kan bli tatt i bruk av 2FA-TOTP apper med størst utbredelse. Ved første øyekast kan det virke som om Google Authenticator med over 100 millioner nedlastninger er langt mer utbredt enn Microsoft Authenticator med over 50 millioner nedlastninger. Det er viktig å poengtere at disse tallene er hentet fra Google sine sider, og at det ikke er utenkelig at Google ønsker å stille sin egen app i et bedre lys enn sine konkurrenter. Microsoft Authenticator kan for eksempel ha 95 millioner nedlastninger og det ville fremdeles stemt med representasjonen

av nedlastninger. Det samme gjelder Authy som i beste fall ligger bak Google Authenticator med minst 50 millioner nedlastninger.

Om man ser på nylige brukeranmeldelser kan man se en trend hvor Microsoft Authenticator med 4,7 av 5 og Authy-brukere med 4,4 av 5 stjerner er godt fornøyde, mens brukeranmeldelser for Google Authenticator med 3,6 av 5 stjerner hvor brukere enten er svært fornøyde eller svært misfornøyde kan tyde på at brukere ofte opplever problemer med appen.

Analyser og tester utført i prosjektet viser at 2FA-TOTP løsningen som skal utvikles vil inneholde en Quick Response code (QR-kode) som kan skannes av både Google Authenticator, Microsoft Authenticator og Authy. QR-koden vil inneholde lik informasjon uansett hvilken 2FA-TOTP app som er i bruk, noe som tyder på at produktet vil kunne fungere for hvilken som helst 2FA-TOTP app. Hvilken applikasjon som benyttes er irrelevant så lenge applikasjonen støtter SHA-algoritmen løsningen benytter, da det ikke er 2-veis kommunikasjon mellom systemet og applikasjonen. Det er da ikke mulig å verifisere om en bruker benytter Google eller Microsoft Authenticator.

3.1.4.2 Sikre hashing algoritmer

Det er noe uklart hvilke SHA-versjoner som er implementert i tredjeparts 2FA-TOTP appene på markedet, da kildekode eller dokumentasjon ikke ligger offentlig. Dette er noe som vil utforskes videre under utvikling og testing, men sett at SHA-1 er standarden for HMAC vil man trolig nå flest ved å legge til rette for SHA-1. Selv med SHA-1 sine sikkerhetsmangler er algoritmen brukt for HOTP og TOTP. Sikkerhetsforskjellen mellom SHA-1, SHA-256 og SHA-512 er ikke nok til å avgjøre hvilken som burde bli implementert i produktet. SHA-1 er standardvalget for 2FA-TOTP apper, med mulighet for SHA-256 og SHA-512 om indikert via parameter i QR-koden. Om bruk av SHA-1 viser seg å skape sårbarheter for TOTP i framtiden, ville det nok lønnet seg å kun lagt til rette for bruk av SHA-256 og/eller SHA-512.

3.2 Valgt løsning

For å tilby MinID-brukere best mulig fleksibilitet, er det valgt en løsning som legger til rette for bruk av flere forskjellige 2FA-TOTP apper. Brukere vil kunne velge å registrere tredjeparts autentiserings-app under MinID-innstillingene sine. Prosjektgruppen finner ingen god grunn for at en bruker skal ha flere forskjellige registrerte tredjeparts apper, derfor velger prosjektgruppen at bruker kun kan ha en slik app registrert. På MinID-innstillinger vil brukere kunne velge å ta i bruk SMS eller MinID-app alternativene som automatisk vil gjøre den registrerte tredjeparts-appen inaktiv som autentiseringsvalg. Under utvikling vil løsningen bli brukertestet ved hjelp av Google Authenticator, Microsoft Authenticator og Authy. Dersom løsningen fungerer for de tre største er det sikret at en størst mulig brukerdel er dekket, og det vil trolig bety at andre tredjeparts 2FA-TOTP apper også fungerer. SHA-1, SHA-256 og SHA-512 er testet og vurdert for løsningen. Mest vekt er lagt på SHA-1 implementasjon da de testede appene faller på denne algoritmen som standard. SHA-1 er uansett vurdert som sikker tatt i betraktning at det er en kontinuerlig endrende tidsfaktor i bildet, og autentiseringen skjer på server siden. Dette diskuteres videre i kapittel 6.2.

3.3 Valg av verktøy

Java

Java (Oracle, n.d) er et høynivå programmeringsspråk som ble utviklet på 1990-tallet av Sun Microsystems og senere kjøpt av Oracle Corporation. Java er et objektorientert språk, noe som betyr at all funksjonalitet er organisert som klasser og objekter, og at koden kjører på en virtuell maskin. Java-17 benyttes under utvikling på grunn av krav stilt av oppdragsgiver.

Spring Boot

Spring Boot (Spring Boot, n.d) er et åpent-kildekode rammeverk som bygger på Spring Framework (Spring.io, n.d) for å gjøre det enklere og raskere å utvikle Java-baserte webapplikasjoner. Spring Boot gir en omfattende plattform for å utvikle mikrotjenester og webapplikasjoner med minimal konfigurasjon og enkel implementering. Spring Boot er valgt etter ønske fra oppdragsgiver.

Thymeleaf

Thymeleaf (Thymeleaf, n.d) er en utbredt template-motor som bygger på Java og gjør enkelt å integrere HTML-templates med Java-kode som kjører på en server. Thymeleaf anvendes etter ønske fra oppdragsgiver.

IntelliJ IDEA

IntelliJ IDEA (IntelliJ IDEA, n.d) er et integrert utviklingsmiljø (IDE) for utvikling av programvare i flere programmeringsspråk for eksempel Java, Kotlin, Scala og andre. IntelliJ IDEA er utviklet av JetBrains og er en av de mest populære IDEene for Java-utvikling. Det er hovedsakelig denne IDEen som de fleste på prosjektgruppen har erfaring med fra før og benyttes derfor til prosjektets utvikling.

MariaDB

MariaDB benyttes som databasesystem (MariaDB, n.d) da det er samme type som oppdragsgiver benytter. MariaDB er en åpen kildekode-relasjonsdatabase basert på MySQL som er kjent hos prosjektgruppen.

HeidiSQL

HeidiSQL (HeidiSQL, n.d) er en populær og gratis open-source databaseadministrasjonsverktøy som er utviklet for å administrere forskjellige databaseplattformer, inkludert MySQL, MariaDB, Microsoft SQL Server og PostgreSQL. Verktøyet er spesielt rettet mot utviklere og databaseadministratorer som trenger et brukervennlig grensesnitt for å administrere og manipulere databaser. HeidiSQL benyttes siden det standardvalget under installasjon av MariaDB.

GitHub

GitHub (GitHub, n.d) er en nettbasert tjeneste som gir en plattform for å administrere og dele kode til programvareprosjekter. Dette gir prosjektgruppen en felles plattform for å lagre, samarbeide, dele og administrere kildekoden og dens endringer gjennom hele livssyklusen til prosjektet. GitHub tilbyr funksjonalitet som versjonskontroll, filbehandling, kodegjennomgang og mer som prosjektgrupper vil benytte seg av.

Slack

Slack (Slack, n.d) er en kommunikasjonsplattform for team og bedrifter som gir et sentralt sted for å kommunisere og organisere samtaler. Prosjektgruppen bruker hovedsakelig Slack til å kommunisere med oppdragsgiver i tekstformat.

Figma

Figma (Figma, n.d) er en nettbasert designplattform som lar brukere opprette, samarbeide og dele designprosjekter. Figma gir brukere et bredt spekter av verktøy for å opprette og redigere design, inkludert muligheten til å lage interaktive prototyper og animasjoner. Det er disse funksjonene prosjektgruppen benytter seg av for å opprette alt fra UX-skisser og Wireframes til modeller og diagrammer for systemdesign.

Trello

Trello (Trello, n.d) er også en webbasert samarbeidsplattform og har en del funksjonalitet som gjør det mulig å opprette Kanban-board som prosjektgruppen benytter for å visualisere oppgaver, og tidsfrister.

Microsoft 365

Microsoft 365 (Microsoft 365, n.d) er en tjeneste fro Microsoft som tilbyr mange skybaserte tjenester og programmer. Det er hovedsakelig Word, Excel og SharePoint prosjektgruppen benytter. Word for dokumentasjon, Excel for Gannt-diagram, timesføring og risikoanalyse, og SharePoint for deling av disse dokumenter og diagrammer.

Teams

Teams (Teams, n.d) er en sky-basert kommunikasjonsplattform som prosjektgruppen benytter for å foreta videomøter med oppdragsgiver og veileder.

3.4 Prosjektmetodikk

3.4.1 Utviklingsmetodikk

Utviklingen er lagt opp til å følge en smidig utviklingsmetodikk (Agile Alliance, n.d). Dette gjøres ved å dele arbeid opp i mindre iterasjoner, det vil si to-ukers intervaller med definerte mål. For å dokumentere de forskjellige fokusområdene i løpet av hver iterasjon, blir iterasjonene dokumentert i et Kanban-board. Kanban er en teknikk og et verktøy som ofte brukes i smidig utvikling. Med å bruke en slik utviklingsmetodikk vil det resultere i en redusert usikkerhet og fokus på risiko. Siden prosjektgruppen har god tilgang på brukerrepresentanter og oppdragsgiver i løpet av prosjektet passer en slik tilnærming også godt. Etter endt iterasjon vil prosjektgruppen i samarbeid med oppdragsgiver gjennomgå resultatet i form av demonstrasjoner og validere resultatet ved hjelp av tilbakemeldinger og vurderinger. De tilbakemeldinger og vurderinger som oppdragsgiver gir vil bidra til å dokumentere eventuelle mangler eller viktige fokusområder til kommende iterasjon.

På slutten av hver uke, diskuterer gruppen uken som har vært; hvordan samarbeid og fremdriften har vært, og om det er nødvendig med tiltak for å holde fremdrift og tidsplanen videre.

Testdrevet Utvikling

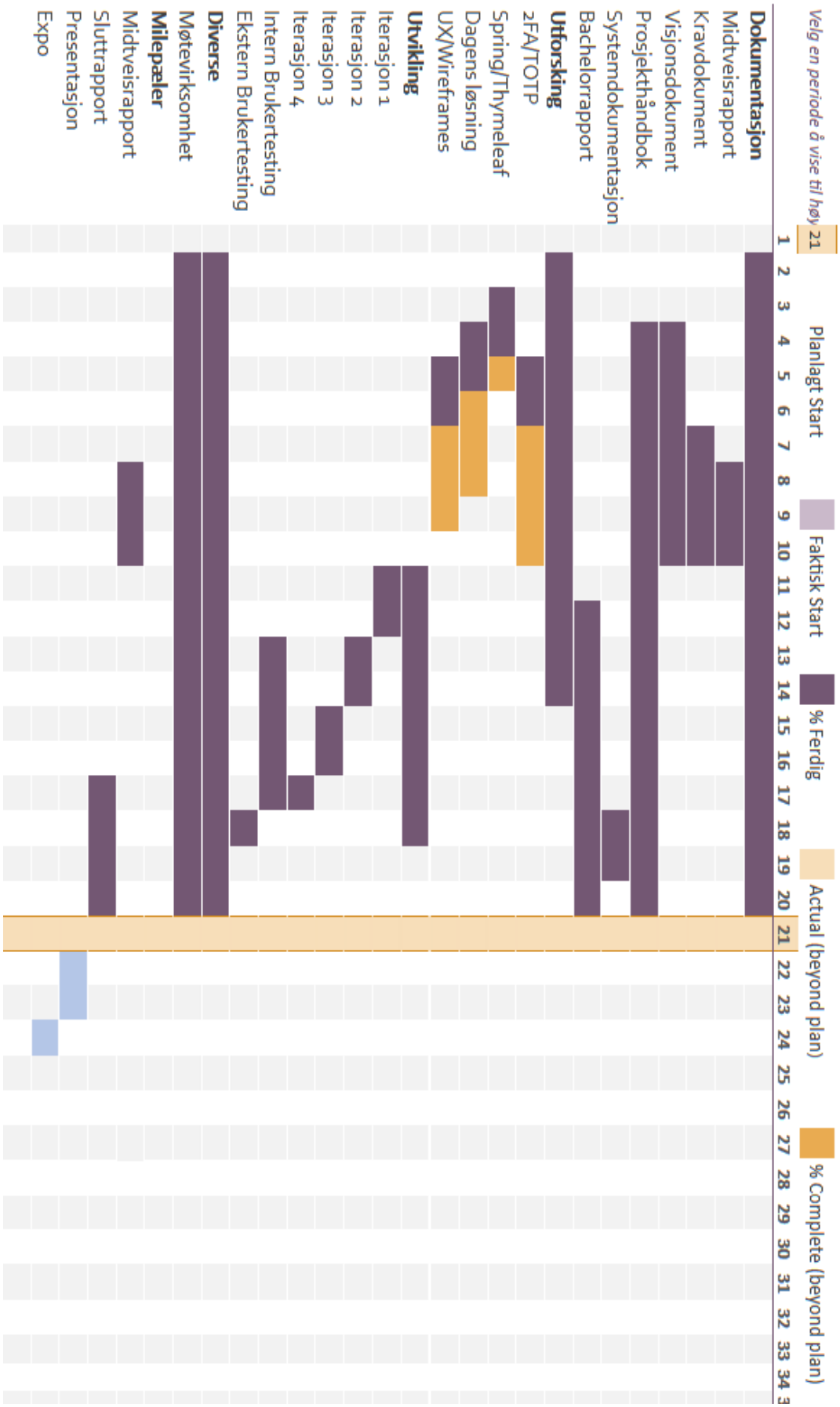
Når det kommer til utviklingsfasen, er det planlagt å ta i bruk testdrevet utvikling (TDD). Dette er en utviklingsmetode der tester skapes i forveien av den koden som testes. Testene skal i første omløp feile, deretter skrives kode som skal kunne bestå testen. I etterkant av dette vil ytterligere tester kunne etableres for å sikre en enda mer robust og feilfri kode, enten det dreier seg om en ny metode eller endring av eksisterende metoder.

Fordeler med TDD er at det tvinger frem kode fra utviklere som er mer testbar, utvidbar, fleksibel og modulær. Det vil også resultere i at en stor andel av koden blir dekket for testing, ettersom den er skrevet for å passere allerede implementerte tester.

3.4.2 Prosjektplan

Prosjektplanen i Figur 3 er delt opp i 5 forskjellige aktivitetsgrupper. Én for dokumentasjon som omfavner de ulike dokumentene som blir skrevet i løpet av prosjektperioden. Utforskning som handler om de ulike utforskingsbitene som legger til rette for store deler av dokumentasjonen og planleggingen. Utvikling som er delt opp i 4 iterasjoner. En del for diverse som handler om de ulike møter som gjennomføres i løpet av prosjektets livssyklus. Til slutt kommer milepæler, som er de større hovedpunktene som fastsetter ulike målepunkter for prosjektet. En mer detaljert versjon av Gannt-diagrammet finnes i vedlegget 1: Prosjekthåndbok.

Prosjektet er delt opp i forskjellige faser som går på tvers av de ulike delene i prosjektplanen. Oppstartsfasen går ut på å definere en problemstilling og komme i gang med prosjektarbeidet. Deretter følger planleggings og utforsknings-fasen, som går ut på å planlegge og dokumentere diverse figurer og modeller som legger til rette for resten av prosjektet. I denne fasen blir det for eksempel skrevet en midtveisrapport som bygger på de ulike støttedokumenter, som igjen inneholder blant annet brukstilfelle-diagram, wireframes, domenemodeller og annet som resten av prosjektet skal bygges videre på. Resterende tid av prosjektet blir dekket av to faser. En fase for utvikling og en fase for dokumentasjon. Disse vil for det meste foregå i parallell hvor utviklingsfasen er igjen delt opp i 3 iterasjoner med 2-ukers intervaller, og en siste iterasjon på 1 uke for de 7 ukene som er planlagt å bruke på utviklingen av løsningen. Parallelt med utviklingsfasen vil det også foregå en dokumentasjonsfase, dette er da basert på sluttrapporten og støttedokumenter.



Figur 3 Gannt pr. Uke 20

Utviklingsfasen er som nevnt tidligere delt opp i 4 iterasjoner.

Iterasjon 1: Sette opp programmet med databasetilkobling og innloggingsfunksjonalitet med et grunnleggende grensesnitt.

Iterasjon 2: Innebærer tilrettelegging for en 2FA-TOTP løsning ved innlogging. Dette krever kode for tilfeldig generering av sikkerhetsnøkler, hashing-algoritmer, QR-kode generering. I etterkant av dette blir funksjonaliteten for 2FA-TOTP implementert i brukergrensesnittet, som skal gjøre det mulig å demonstrere en fungerende autentiseringsprosess.

Iterasjon 3: Består av å fullføre funksjonaliteter som ikke fikk prioritet tidligere, samt fullføre enhetstester som gir bedre dekke hos funksjoner i systemet relatert til TOTP-kode validering/-generering og QR-kode generering. Her forekommer også sikkerhetstesting ved hjelp av OWASP ZAP (OWASP ZAP, n.d).

Iterasjon 4: Refaktorering og ferdigstilling av løsning, samt ekstern brukertesting. Utvikling og endringer som skjer ved dette punktet vil være finjusteringer før overrekkelse til oppdragsgiver.

3.4.3 Risikovurdering

Tabell 2 Risikoanalyse Iterasjon 1.2

	Hendelser	Årsak(er)	Virkning(er)	Risikoprodukt	Risikoreduserende tiltak
1	Mister arbeidstid og fokus fra bachelor prosjektet	Semesterprosjekt innenfor valgfag krever tid og oppmerksomhet fra prosjektgruppen	Tidspress og forstyrrelser i arbeidsfordeling mellom bachelor og valgfag	**8**	Fortsette å jobbe innenfor faste tider for bachelorprosjekt og la dette ha prioritet over andre fag.
2	Manglende kompetanse	Usikkerhet i forhold til krav	Lite fremgang	**8**	Grundige undersøkelser, tidlig i prosjektfasene
3	For stort omfang	For høyt ambisjonsnivå		**4**	Iterativ utviklingsprosess som setter søkelys på de viktigste kravene
4	Misforståelser innen problemdomenet	Manglende kommunikasjon mellom oppdragsgiver og bachelorgruppe		**8**	Kontinuerlig kommunikasjon og regelmessige demonstrasjoner
5	Ikke-funksjonell kode flettet inn i master. Feil introduseres. Git historie blir rotete og uleselig	Manglende erfaring med version-control	Tidsforbruk, arbeid går tapt, frustrasjon, vanskeligere å vedlikeholde og korrigere kode.	**6**	Gjennomfør arbeid på branches og kun merge branches så lenge kode er velfungerende og passerer tester. Beskytt Master/Main fra direkte pushing av kode
6	Taper database eller tilgang til database	Sletting av database ved feil eller glemt passord for tilgang	Vil måtte sette opp en ny database med ny data, kjøring av programmet vil måtte vente.	**4**	Lokale kopier av database for individuelt arbeid
7	For lite enhetstesting	TDD blir ikke gjennomført, lite disiplinert og høyt engasjement for å jobbe videre med kode.	En kodebase som ikke er verifisert med nok tester, og ikke lett lar seg refraktorers.	**16**	Ha som krav at alle funksjoner og metoder som hører til TOTP og autentisering skal ligge innenfor testsettet.

Tabell 3 Risikoprodukt

Sannsynlighet	**Svært Høy (5)**	**5**	**10**	**15**	**20**	**25**
	Høy (4)	**4**	**8**	**12**	**16**	**20**
	Middels (3)	**3**	**6**	**9**	**12**	**15**
	Lav (2)	**2**	**4**	**6**	**8**	**10**
	Svært Lav (1)	**1**	**2**	**3**	**4**	**5**
		Svært Lav (1)	**Lav (2)**	**Middels (3)**	**Høy (4)**	**Svært Høy (5)**
	Konsekvens					

3.5 Evalueringsplan

Under utviklingen vil prosjektgruppen kontinuerlig skrive enhetstester for all funksjonell kode som et følge av at det praktiseres testdrevet utvikling. Disse testene blir skrevet på forhånd av kode, og det vil kreves at testene godkjennes for at implementasjonen av koden er godtatt. Dette vil resultere i et stort og dekkende sett med tester som vil gjøre det enklere for prosjektgruppen å godkjenne fortløpende endringer.

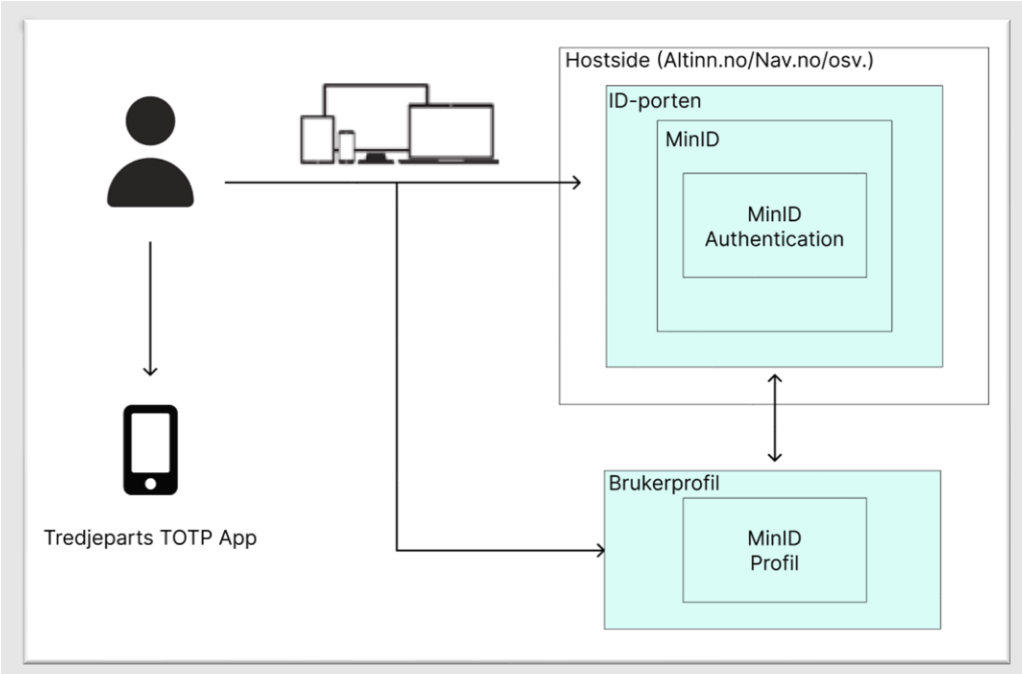
Prosjektgruppen har planlagt å kjøre brukertesting regelmessig igjennom prosjektets livssyklus. Det vil si at prosjektgruppen vil teste løsningen manuelt ut ifra funksjonalitet som blir innebygget underveis. Det er også planlagt å kjøre slik brukertesting med eksterne brukerrepresentanter for å få innblikk i og mulige nye perspektiver på løsningen.

Det er et ønske fra oppdragsgiver at det blir gjennomført en regelmessig demonstrasjon av løsningen slik den er til gitt tid. I samtale med oppdragsgiver er det kommet til enighet om å gjøre dette hver 14. dag. I en tidlig planleggingsfase vil det være wireframes av den planlagte løsningen som demonstreres, dette skjer via interaktive wireframes designet i Figma som vil gjøre det mulig å illustrere den løsningen/flyten som er tenkt ut hos prosjektgruppen.

4 DETALJERT LØSNING

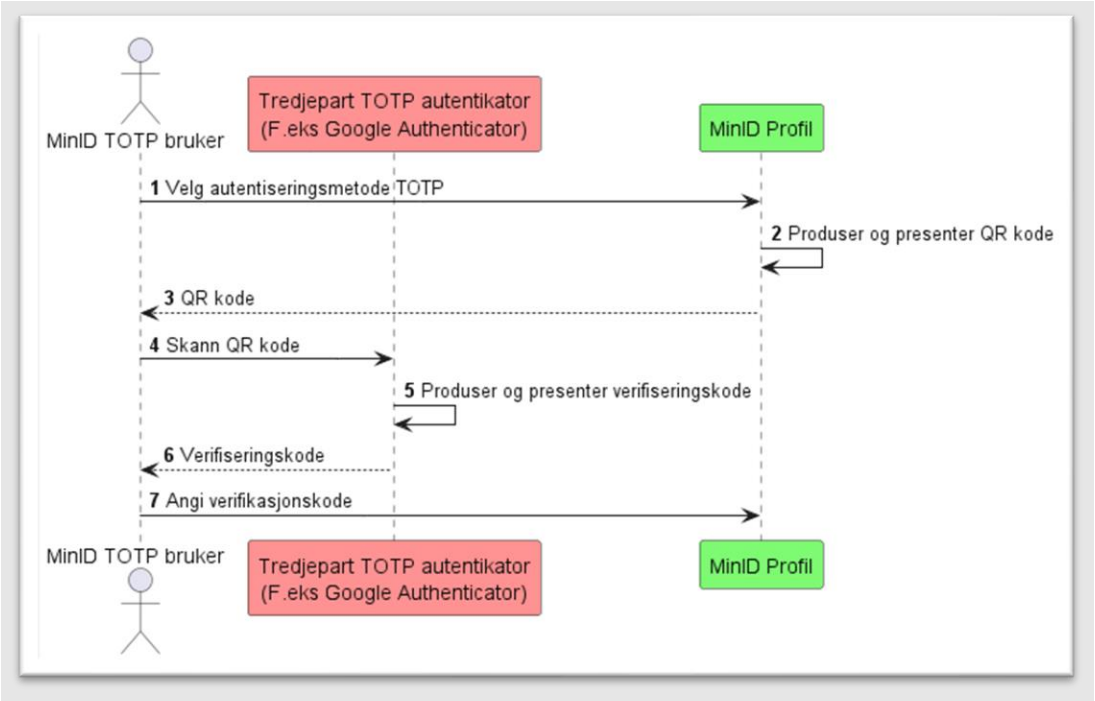
4.1 Konseptuell oversikt

Tjenestene for autentisering ved hjelp av MinID / ID-porten, nås via offentlige nettjenester som Altinn.no og Nav.no.



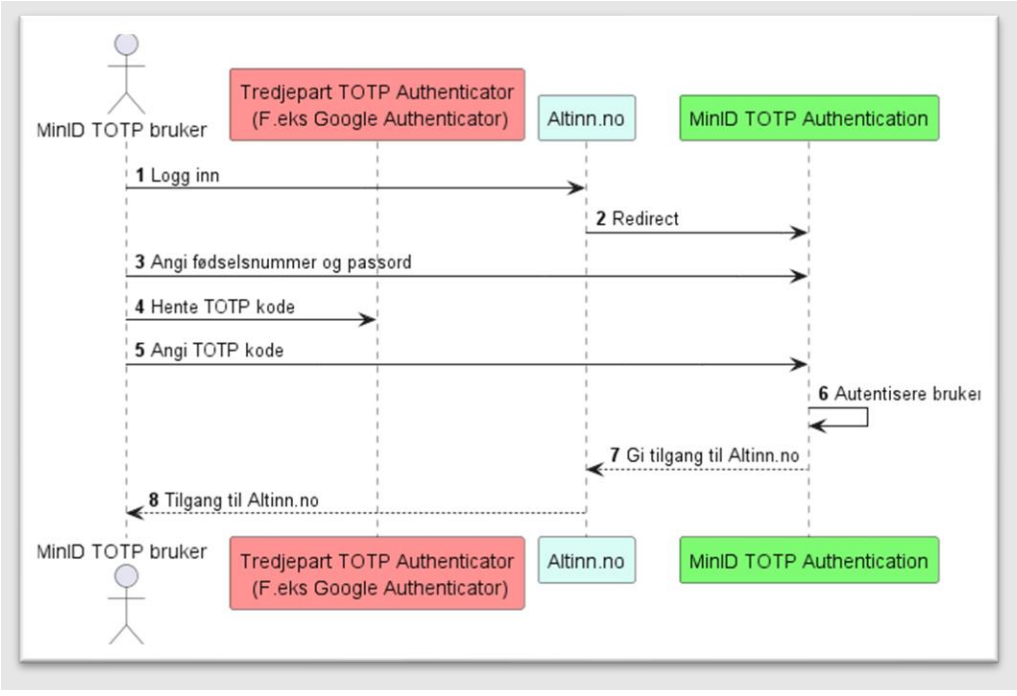
Figur 4 Bruker i systemet

For en bruker som ønsker å ta i bruk autentisering ved hjelp av TOTP, vil vedkommende først registrere dette på sin brukerprofil/MinID profil. I løpet av registreringen lagres nødvendig informasjon i tredjepartsautentiseringsapplikasjonen, som for eksempel Google Authenticator, ved å skanne QR-koden.



Figur 5 Sekvensdiagram for registrering av 3.parts TOTP autentiseringsapplikasjon

Neste gang bruker velger å autentisere seg ved hjelp av MinID, presenteres vedkommende automatisk for TOTP autentisering, og blir bedt om å angi fødselsnummer og MinID passord, før de henter ut TOTP-kode fra valgt tredjeparts autentiseringsapplikasjon. TOTP-koden angis så i MinID for validering og autentisering.



Figur 6 Sekvensdiagram for autentisering med TOTP

4.2 Funksjonalitet/Brukseksempler

Prosjektet endte opp med fem brukstilfeller for å oppfylle kravene. For en mer detaljert gjennomgang av disse brukstilfellene, se Vedlegg 3: Kravdokumentasjon.

Use case: *Se brukerinnstillinger*

Bruker velger MinID fra ID-portens meny av alternativer og logger inn med personnummer og passord. Bruker blir videreført til autentisering basert på hvilken autentiseringsmetode som er registrert.

Use case: *Endre Brukerinnstillinger*

Bruker endrer passord eller innloggingsmetode ved å trykke ikon ved høyre for egenskap som ønskes forandret.

Use case: *Initialisere 2FA-TOTP autentisering.*

Bruker trykker på ikon for endring av autentiseringsmetode og velger relevant autentiseringsapp. En QR-kode kommer frem, denne skannes av brukers mobilkamera via aktuell 2FA-app. Bruker må legge inn en bekreftelseskode produsert av app.

Use case: *Avregistrere MinID-app*

Bruker trykker på ikon for endring av autentiseringsmetode, velger knapp for fjerning av app. Etter å ha bekreftet skal SMS-kode automatisk velges som autentiseringsmetode.

Use case: *Autentisering ved hjelp av 2FA-TOTP.*

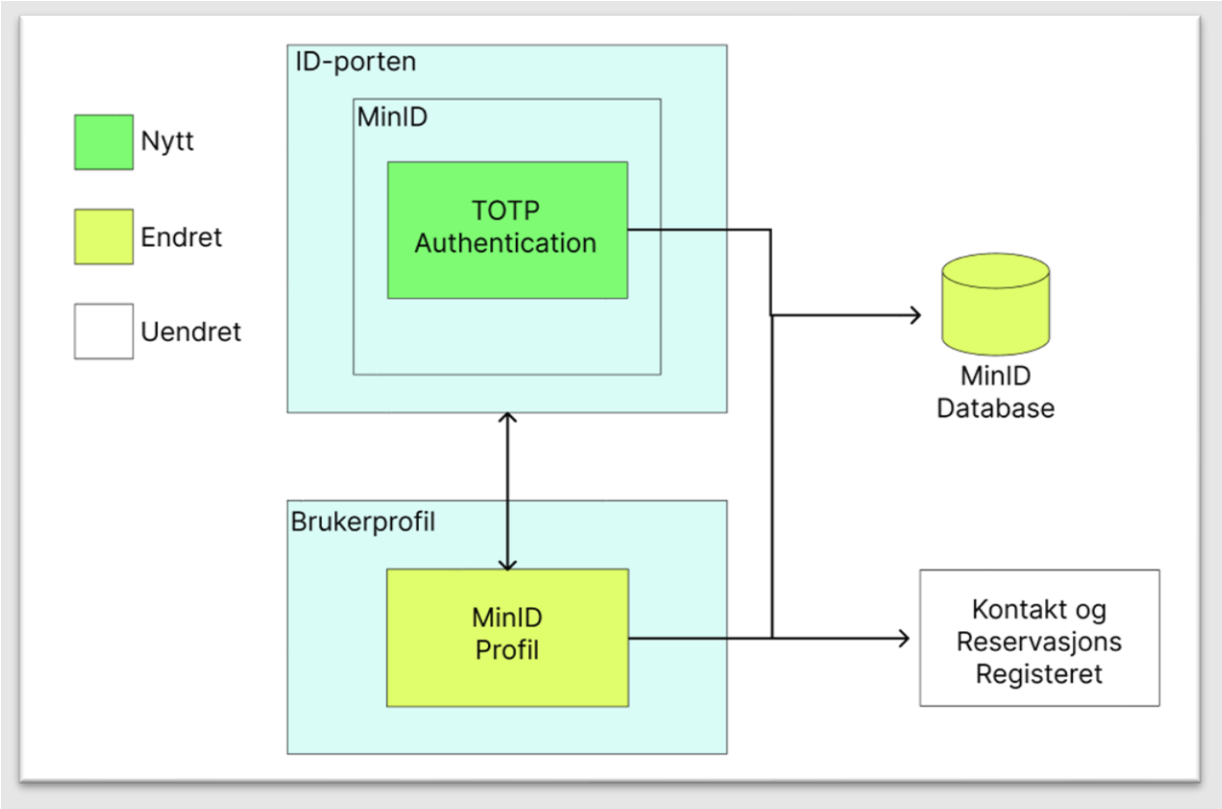
Bruker åpner opp valgt autentiseringsapp, og fyller inn 6-sifrret kode ved 2FA steget.

4.3 Arkitektur

Arkitekturmessig består løsningen av to hovedmoduler:

- MinID TOTP Authentication: I denne modulen håndteres all brukerinteraksjon og logikk knyttet til selve autentiseringsprosessen ved bruk av TOTP autentisering.
- MinID Profil: I denne modulen håndteres all brukerinteraksjon og logikk knyttet til innleggelse og uthenting av data for en MinID-bruker sin brukerprofil. Herunder også valg av og oppsett for TOTP som autentiseringsmetode.

For mer detaljert innsikt i systemarkitekturen, se vedlegg 4: Systemdokumentasjon



Figur 7 Arktitekturdiagram

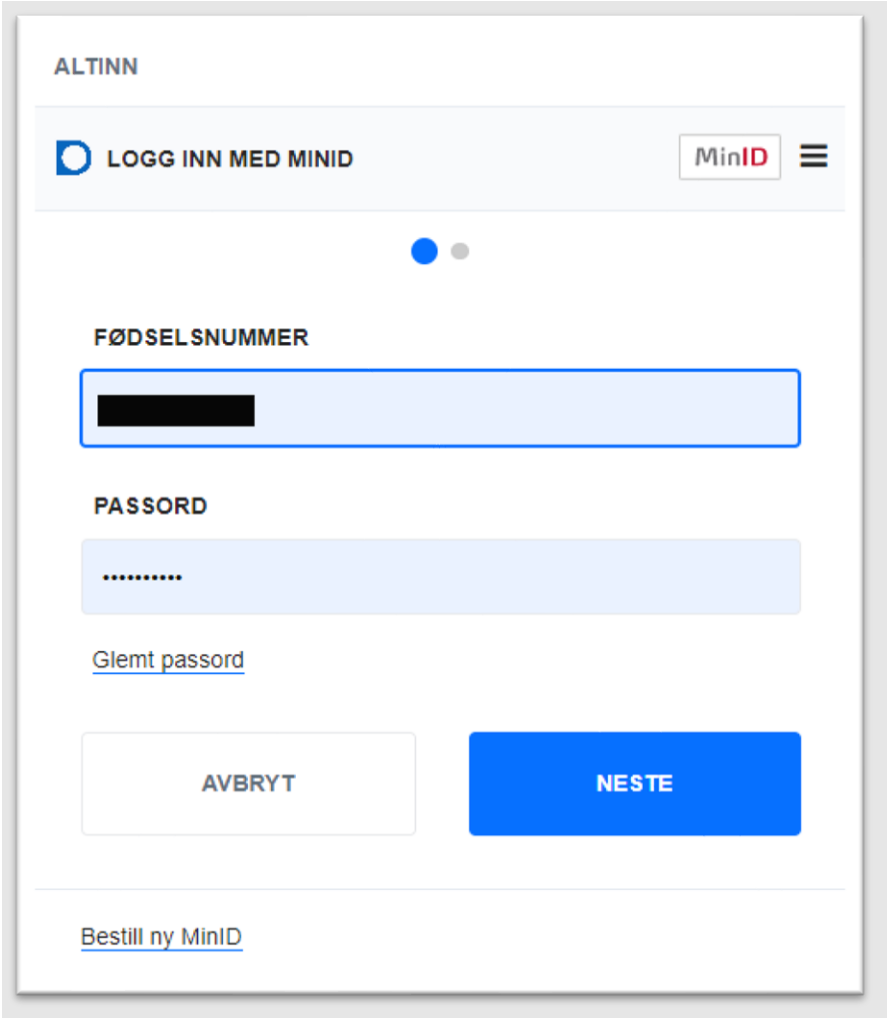
4.3.1 Modul: MinID TOTP Authentication

4.3.1.1 Brukerdialog

Brukerdialogen for autentisering ved hjelp av MinID TOTP, som del av MinID TOTP Authentication modulen består av 2 steg.

Steg 1:

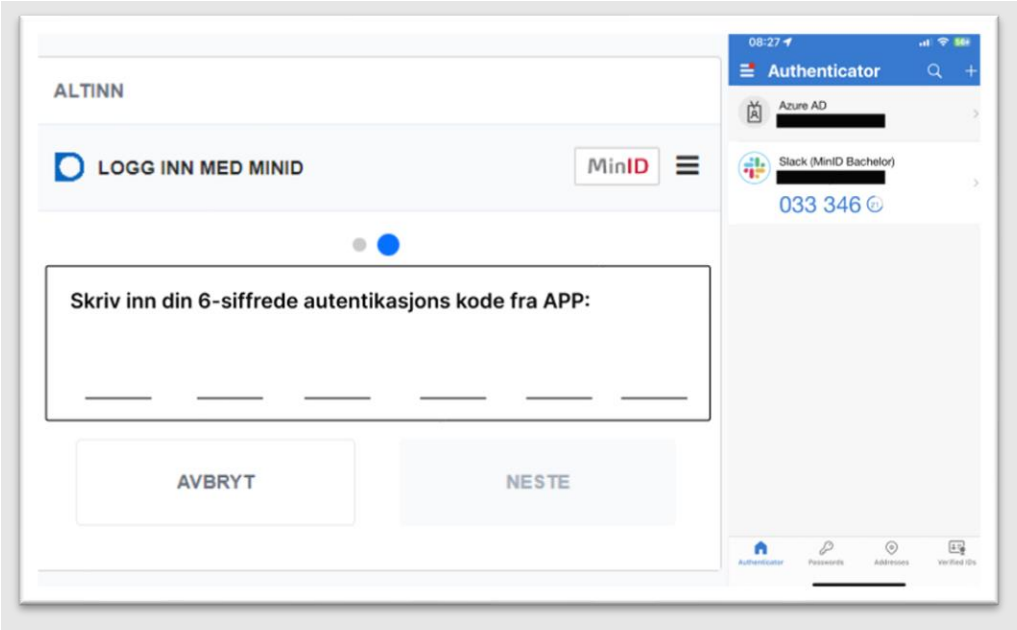
Bruker fyller inn fødselsnummer og passord som i eksisterende påloggingsflyt.



Figur 8 Innlogging med fødselsnummer og passord

Steg 2:

Bruker vil åpne sin valgte tredjeparts 2FA-TOTP applikasjon og skrive inn koden som appen presenterer i feltet på skjermbildet.



Figur 9 Tofaktorautentisering i brukers perspektiv

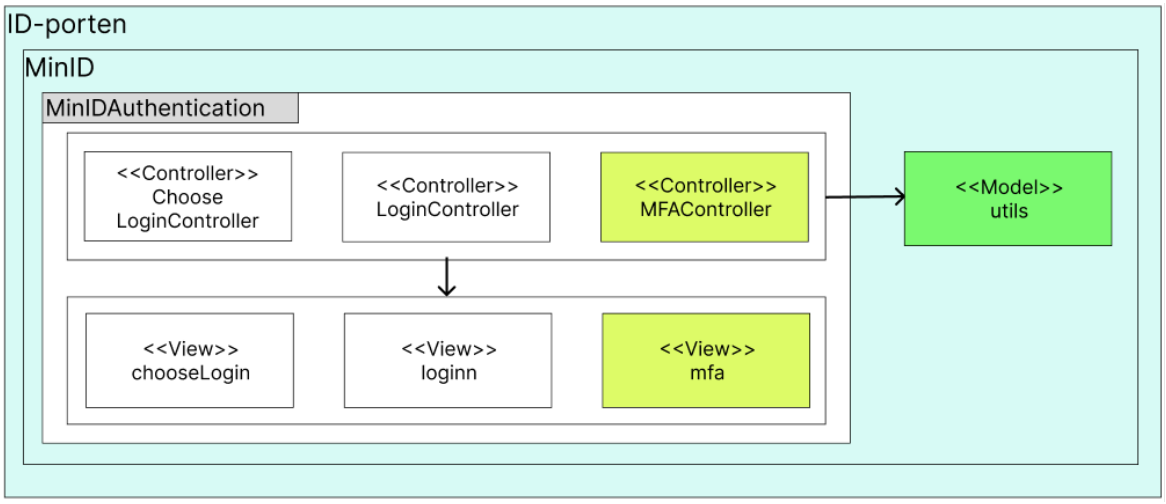
4.3.1.2 Teknisk løsningsdesign

Løsningen baserer seg på designmønsteret Model-View-Controller (Mozilla, 2023), og implementeres ved hjelp av Thymeleaf og Spring Boot.

I denne modulen deler vi inn i 3 view med tilhørende kontrollere, hvor hvert view og kontroller håndterer følgende:

- Choose Login: Har ansvar for å håndtere de ulike innloggingsalternativene i ID-porten: MinID, BankID, Comfides og Buypass ID.
- Login: Står for selve innlogging og autentiseringsprosessen for brukernavn og passord.
- Multi Factor Authentication(MFA): Håndterer innlogging og autentisering av bruker med den autentiseringsmetode som bruker er satt opp med via MinID-innstillinger. Når bruker er satt opp med TOTP er det denne delen som håndterer brukerinput og validering av autentisering ved hjelp av TOTP.

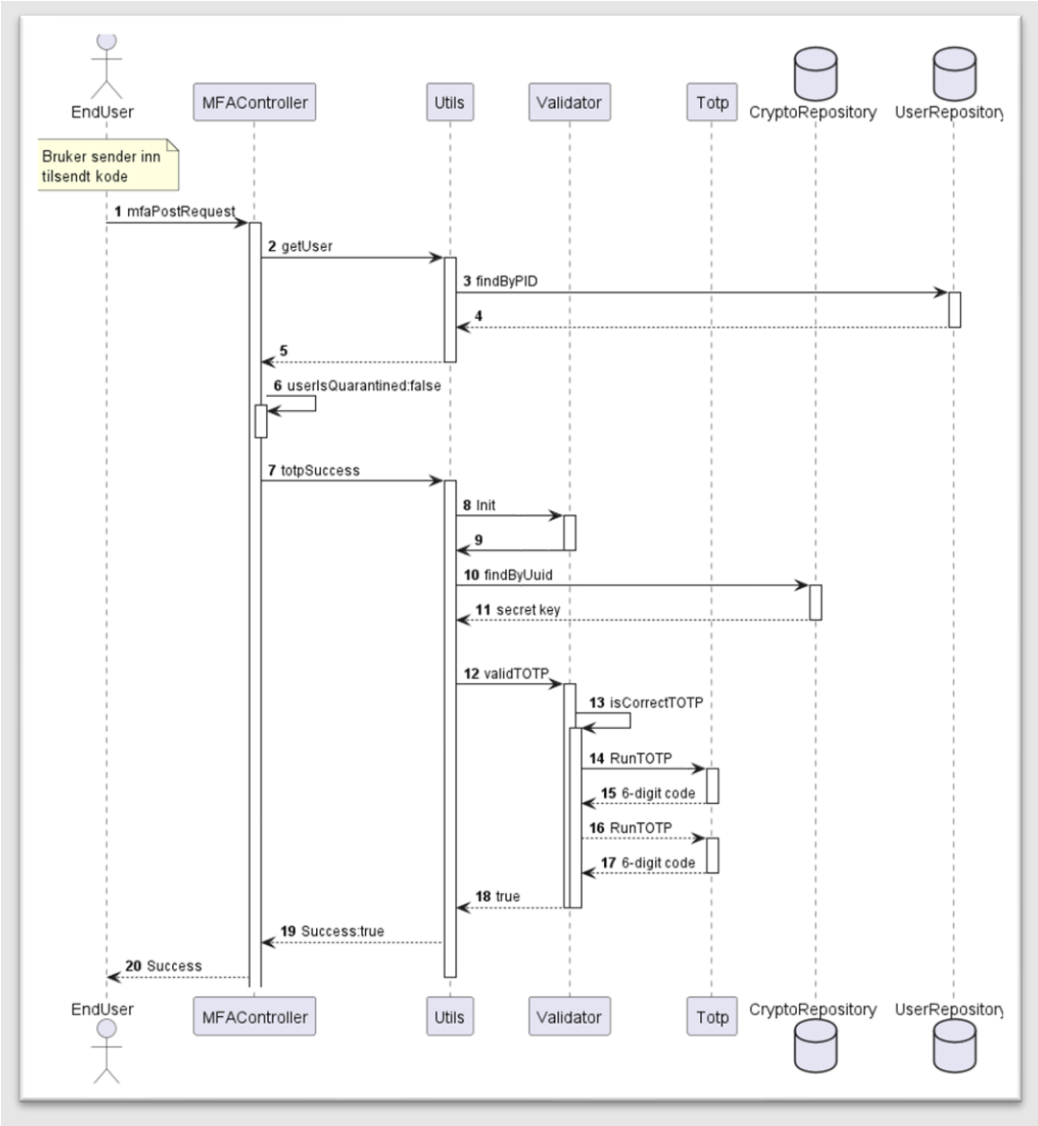
Følgende klassestruktur benyttes i denne modulen:



Figur 10 Klassestruktur for MinID-Authentication

- Utils: Håndterer validering av brukerinput og status om bruker eksisterer/er låst ute/ er allerede logget inn/oppdaterer brukerinformasjon.

Løsningen for brukseksemplet *Autentisering vha. TOTP* kan illustreres ved hjelp av følgende sekvensdiagram:



Figur 11 Sekvensdiagram for TOTP autentisering

Her vil prosjektgruppen også trekke frem løsning og fremgangsmåte for valideringen av angitt TOTP-kode:

Første steg i TOTP-autentisering skjer i validTOTP-metoden som passer på at den samme TOTP-koden ikke kan brukes 2 ganger på rad. Dette er i henhold til en av kravene spesifisert i RFC6238.

Videre vil isCorrectTOTP sammenligne koden bruker har skrevet inn med det som genereres av systemet i det nåværende tretti sekunders tidsintervallet, samt intervallet som kom før. Dette gjøres for å unngå feil om bruker skriver inn TOTP-koden rett etter det neste tidsintervallet har begynt, eller potensielle forsinkelser under kommunikasjon. runTOTP-metoden kjøres derfor to ganger på rad. runTOTP-metoden tar den tilfeldig genererte sikkerhetsnøkkelen og tidsinformasjon i millisekunder som parameter, og konverterer tidsinformasjonen til en representasjon av et trettisekunders tidsintervall. Tidsintervallet kan ikke ha en lengde på mindre enn seksten tegn, derfor legges det til ledende nuller hvor nødvendig.

```java
public String runTOTP(String key, long time){
    long T0 = 0;
    long timePeriod = 30;
    String steps = "0";

    try {
        steps = Long.toHexString((time - T0)/ timePeriod).toUpperCase();
        while(steps.length() < 16){
            steps = "0" + steps;
        }
    } catch (final Exception e) {
        System.out.println("Error :" + e);
    }

    String returnDigits = "6";
    return generateTOTP(key, steps, returnDigits);
}
```

Figur 12 Metoden runTOTP

Videre kalles generateTOTP fra runTOTP, som dekoder tidsintervallet fra "time" og sikkerhetsnøkkelen fra "key" i parameter. HmacSha-metoden kalles for å generere en hash basert på sikkerhetsnøkkelen og tidsverdien.

```java
byte[] msg = hexStr2Bytes(time);
Base32 decoder = new Base32();
byte[] k = decoder.decode(key);
byte[] hash = hmacSha(crypto, k, msg);
```

Figur 13 Uthenting av byte-verdi i generateTOTP-metoden

Videre avkortes hashen ved bitvis sammenslåing og forflyttet mot venstre. Gjenværende Integer brukes til å hente ut et seks-sifferet-tall som resulterer i TOTP-koden

```java
int offset = hash[hash.length - 1] & 0xf;

int binary = ((hash[offset] & 0x7f) << 24) |
        ((hash[offset + 1] & 0xff) << 16) |
        ((hash[offset + 2 ] & 0xff) << 8) |
        (hash[offset + 3] & 0xff);

int otp = binary % DIGITS_POWER[codeDigits];
```

Figur 14 Uthenting av TOTP-kode i generateTOTP-metoden

Metoden hmacSha bruker Java Cryptography Extension for å hente den kryptografiske algoritmen. Deretter genereres en hashet autentiseringskode basert på gitt hashing-algoritme, sikkerhetsnøkkel og tidsintervall.

```java
private byte[] hmacSha(String crypto, byte[] keyBytes, byte[] text){
    try {
        Mac hmac;
        hmac = Mac.getInstance(crypto);
        SecretKeySpec macKey = new SecretKeySpec(keyBytes, algorithm: "RAW");
        hmac.init(macKey);
        return hmac.doFinal(text);
    } catch (GeneralSecurityException gse) {
        throw new UndeclaredThrowableException(gse);
    }
}
```

Figur 15 Generering av hash-produkt

Om koden fra generateTOTP stemmer med hva bruker har skrevet inn fullføres innloggingen og databasen blir oppdatert med ny innloggingsdata. Om koden fra bruker ikke stemmer forblir brukeren på MFA siden og bruker sin teller for feilede 2FA-forsøk økes.

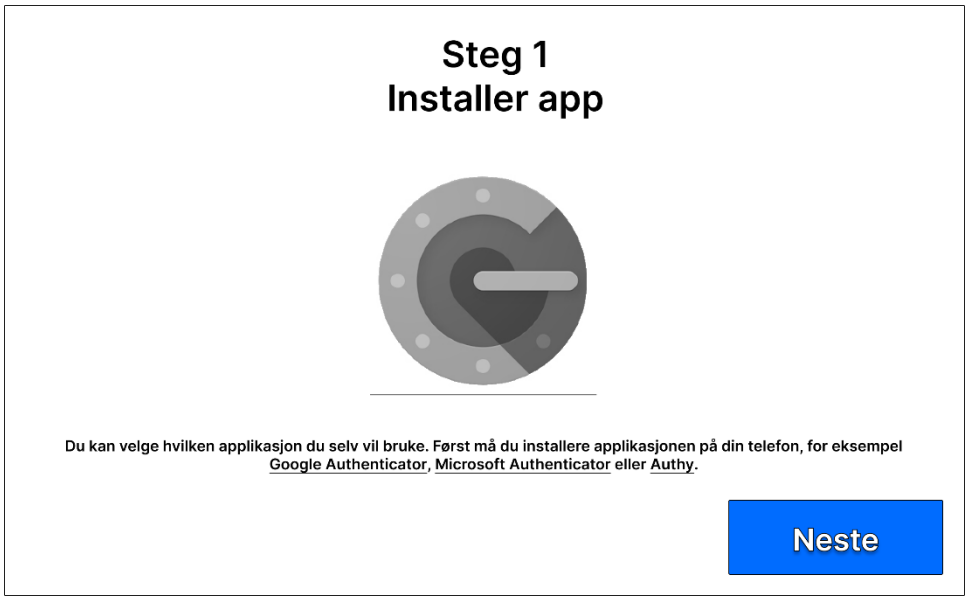
4.3.2 Modul: MinID Profil

4.3.2.1 Brukerdialog

Brukerdialogen for registrering av TOTP som autentiseringsmetode i MinID Profil, består av 2 steg:

Steg 1:

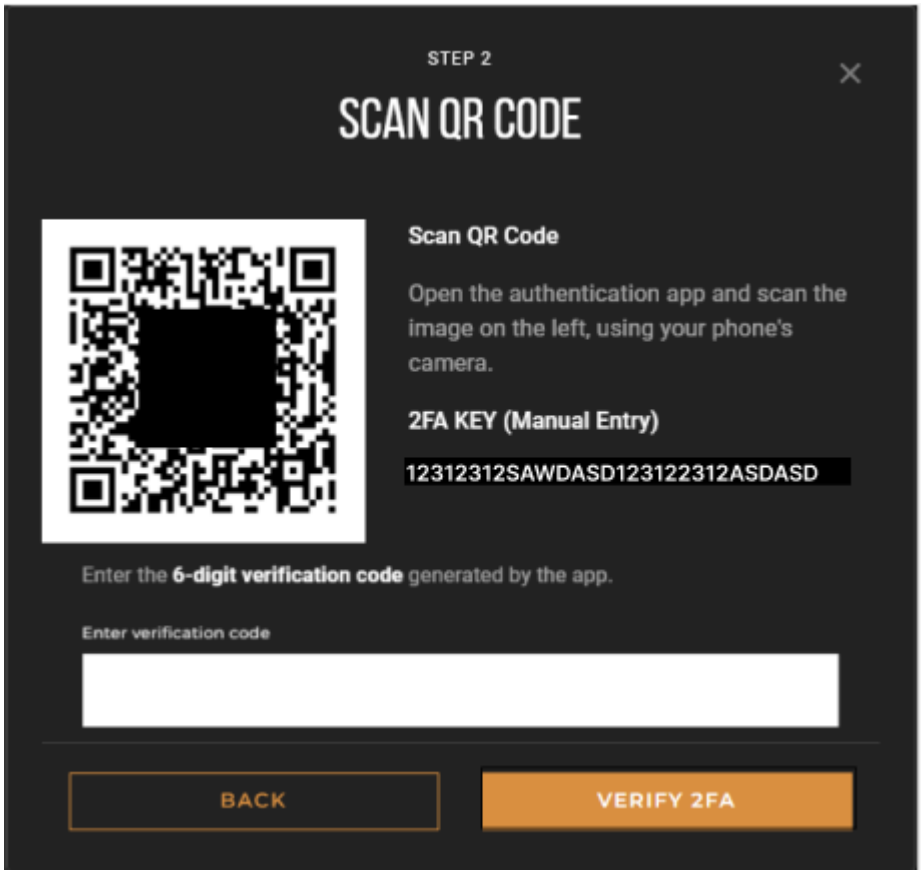
Ved registrering av TOTP-autentisering vil bruker først bli tatt til et skjermbilde som informerer bruker om hva som behøves for å kunne benytte seg av denne autentiseringsmetoden, samt eksempler på de tre mest utbredte 2FA-TOTP appene.



Figur 16 Henvisning til anbefalte applikasjoner

Steg 2:

Bruker blir sendt til registeringssiden hvor en tredjeparts TOTP-app tas i bruk for å skanne den genererte QR-koden eller manuelt innføre sikkerhetsnøkkelen. Brukeren sin valgte TOTP-app vil deretter produsere en seks-sifret-tallkode som må skrives inn i feltet nedenfor for validering.



Figur 17 Inspiratsjon for egen implementasjon

For oppdatering og endring av innstillinger i brukerprofil i MinID, gjøres det på følgende måte:

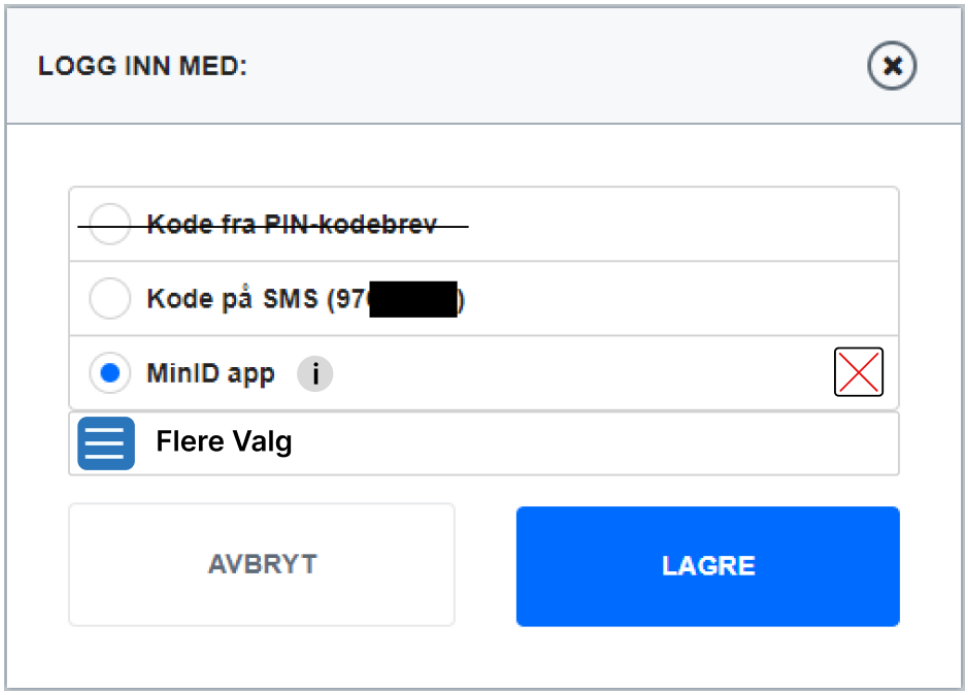
Steg 1:

Når bruker er logget inn og autentisert vil bruker bli møtt med et skjermbilde som viser informasjon relatert til autentiseringsmetode, registrert telefonnummer og passord. Funksjonaliteten bak endring av passord forblir uendret og muligheten for å endre registrert telefonnummer erstattes med en lenke til KRR sine sider, hvor bruker kan gjøre ønskede endringer.

Figur 18 Innstillinger-meny presentert i settings viewet

Steg 2:

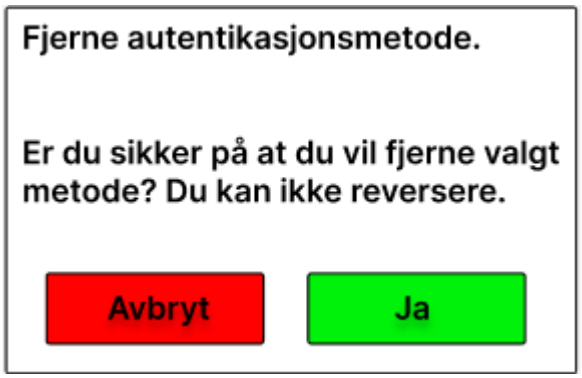
For å endre autentiseringsmetode trykker bruker på blyant tegnet øverst til høyre i Figur 18. Bruker vil da bli møtt av neste skjermbilde, og informasjonen som viser brukerens valgte og eksisterende autentiseringsmetoder. Det vil si enten SMS, SMS og MinID-app, eller SMS og tredjeparts 2FA-TOTP applikasjon.



Figur 19 Valgmenyen for autentiseringsmetode i en tidlig fase

Steg 3:

Bruker kan velge å avregistrere MinID-appen eller en tredjeparts TOTP-app ved å trykke på det røde krysset markert ovenfor. For å unngå uhell vil bruker bekrefte avregistreringen. Etter å ha avregistrert appen vil SMS-autentisering automatisk bli valgt for innlogging.



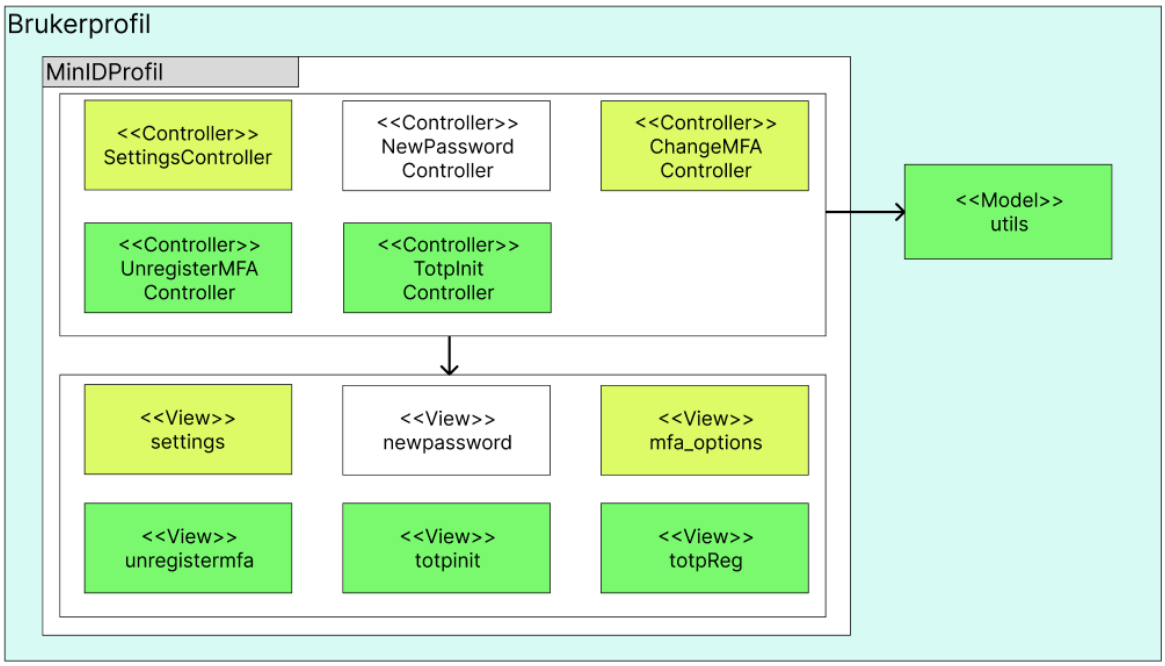
Figur 20 Bekreftelse-vindu

4.3.2.2 Teknisk løsningsdesign

I denne modulen deler vi inn i 6 view med 5 tilhørende kontrollere, hvor hvert view og kontroller håndterer følgende:

- Settings: Har som ansvar å vise bruker hvilken informasjon som er registrert, det vil si hvilken 2FA-metode og hvilket telefonnummer en bruker er registrert med.
- New Password: Håndterer oppdatering av passord.
- Change MFA: Viser en bruker hvilke 2FA-metoder bruker har muligheten til å velge og gir muligheten til å legge til tredjeparts 2FA-TOTP applikasjon eller fjerne dersom bruker er registrert med slik fra før.
- Unregister MFA: Har som ansvar og informere og validere om bruker faktisk ønsker å fjerne en valgt 2FA-metode.
- TOTP init: Setter i gang registreringsprosessen for tredjeparts 2FA-TOTP applikasjon for brukeren. Først informeres bruker om anbefalte tredjepartsapplikasjoner og deretter genereres QR-kode. Når bruker har skannet QR-koden kan verifisering av koden bruker må skrive inn håndteres.

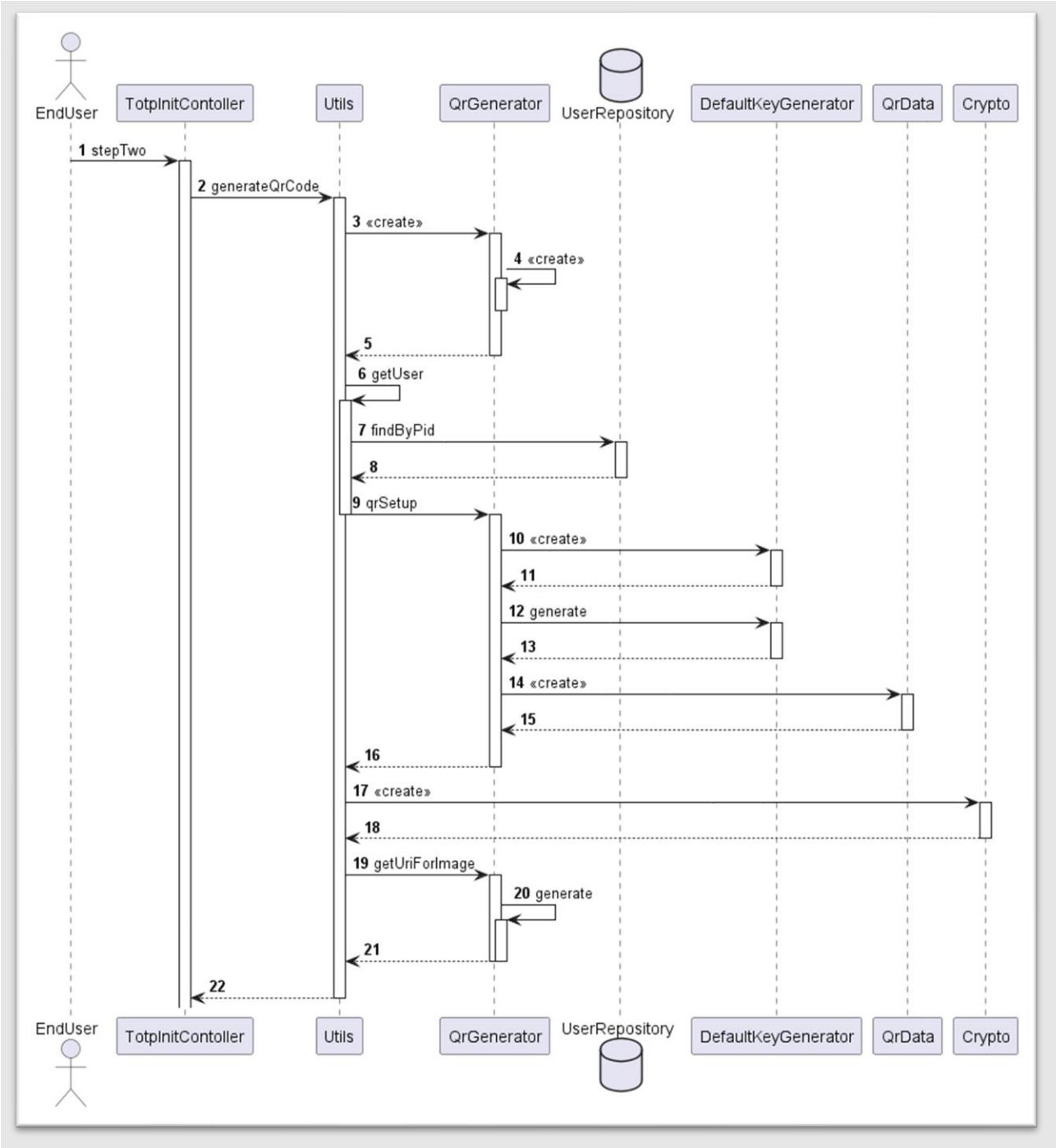
Følgende klassestrukur benyttes i denne modulen:



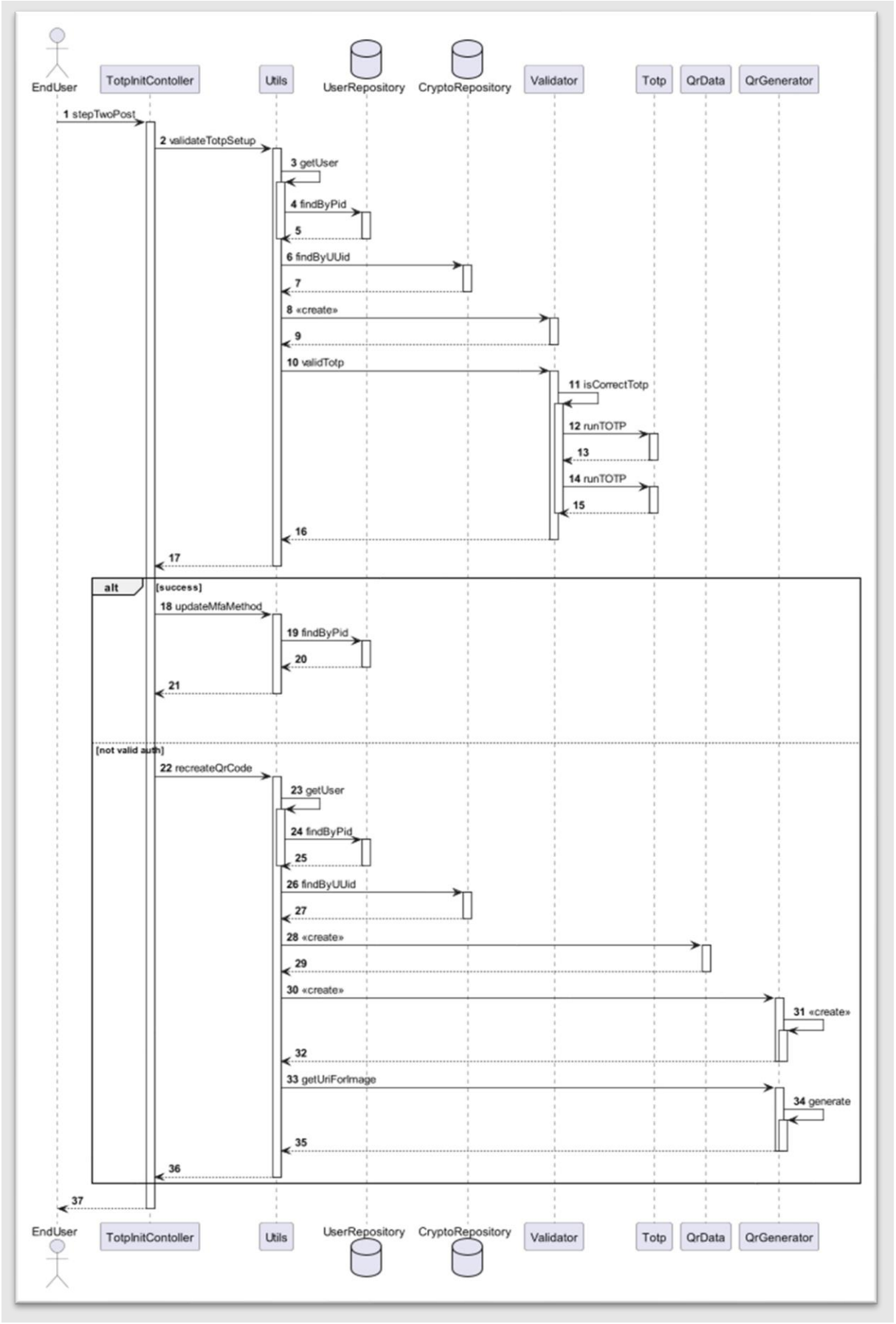
Figur 21 Klassestrukturen for MinID-Profil

- Utils: Håndterer endringer, oppdatering og opprettelse av bruker/crypto informasjon. Håndterer også validering av TOTP-kode for registrering av TOTP autentisering, og validering av passord for endring av passord.

Løsningen for brukseksemplet *Initialisere 2FA-TOTP autentisering* kan illustreres ved hjelp av følgende sekvensdiagrammer:



Figur 22 Sekvensdiagram for registrering av TOTP og generering av QR-koder



Figur 23 Sekvensdiagram for validering og eventuell gjenskaping av QR-kode for TOTP registrering

Mye av det som skjer i Figur 22 og Figur 23 blir enklere å forstå om man ser nærmere på hvordan QR-koden blir generert:

Tredjeparts TOTP applikasjoner krever at QR-koden er på bestemt form for å kunne tolke i informasjonen som ligger i koden riktig. Når det blir sendt et GET kall til /stepTwo, tar kontrolleren og initierer en generateQrCode metode. Denne metoden skaper et QrGenerator objekt som til slutt returnerer en Uniform Resource Identifier (URI) som representeres i bildeformat i frontend. Men for å få gjort det må objektet ha en del informasjon. Programmet henter derfor inn brukeren som ønsker å ta i bruk TOTP. Deretter opprettes et QrData objekt ved å kalle på QrGenerator klassens metode qrSetup.

```
1 usage   Jørgen Vikan Eidsvåg
public String generateQrCode(HttpSession session) {
    QrGenerator qrGenerator = new QrGenerator();
    User user = getUser(session);
    assert user != null;
    QrData qrData = qrGenerator.qrSetup(user);
    cryptoRepository.save(new Crypto(user.getUuid(), qrData.getSecret(),""));
    return qrGenerator.getUriForImage(qrData, "image/png");
}
```

Figur 24 Metoden for generering av QR-kode

Denne metoden oppretter et DefaultKeyGenerator objekt som brukes til å opprette en sikkerhetsnøkkel. Sikkerhetsnøkkelen genereres tilfeldig med en gitt lengde på 32 karakterer. Deretter returneres QrData objektet med informasjonen om hvilken type det er, hvem bruker er, hva sikkerhetsnøkkelen er, hvem som utsteder, hvilken algoritme, hvor mange siffer koden skal bestå av og hvor ofte koden skal oppdateres. I denne settingen brukes informasjonen i type for å indikere om HOTP eller TOTP skal brukes.

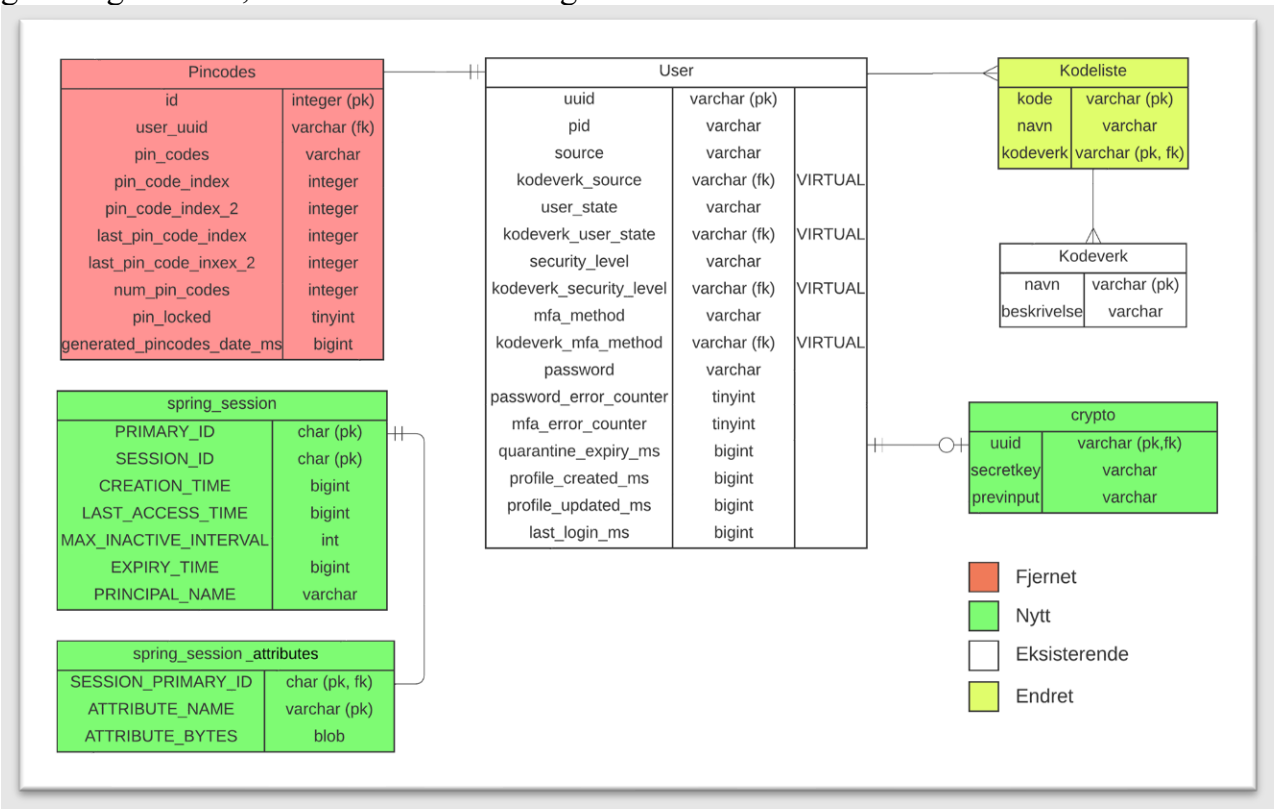
```
1 usage   Ulrik Fagerberg
public QrData qrSetup(User user){
    DefaultKeyGenerator keyGenerator = new DefaultKeyGenerator();
    String secret = keyGenerator.generate();
    return new QrData("totp", user.getPid(), secret, "MinID", "SHA1", 6, 30);
}
```

Figur 25 Data for QR-kode samles i objekt

Sikkerhetsnøkkelen til brukeren lagres deretter i databasen, og URI sendes til frontend representert som et bilde slik at brukeren kan skanne QR-koden med sin smarttelefon. Brukeren må så skrive inn koden som blir generert i appen på nettsiden og denne må valideres. Når koden er validert har brukeren lagt til tredjeparts TOTP applikasjon som autentiseringsmetode.

4.3.3 Database

Designet av databasen er basert på den eksisterende løsningen hos DigDir. Et tildelt script har gitt innsikt i tabellene: User, Kodeliste, Kodeverk og Pincodes. Løsningen er bygget på grunnlag av dette, men med noen endringer.



Figur 26 Database ER-diagram

Pinkoder

Tabell for pinkoder er blitt ekskludert da autentisering med pinkode ikke vil har relevans for prosjektet og vil kunne utelukkes uten konsekvenser. I tillegg er pinkodeordningen faset ut av DigDirs løsninger innen prosjektet ble fullført.

User

User tabellen er urørt i forhold til eksisterende løsning, og det er kun brukt en delmengde av kolonnene for egen bruk i prosjektet.

Crypto

For hver bruker som setter opp TOTP vil det være behov for å lagre de hemmelige nøklene, dette gjøres i en egen tabell med fremmednøkkel koblet opp imot hver bruker.

Hvorvidt en sikkerhetsnøkkel skal lagres i det heletatt er basert på om bruker tar nytte av TOTP eller ikke, det er derfor hensiktsmessig å lagre denne dataen i en egen tabell for å unngå tomme kolonner for hver bruker som ikke har satt opp TOTP autentisering, dette gjøres med tanke på god databasenormalisering. (Kristoffersen, 2020)

Med UUID som primær- og fremmednøkkel har brukere ingen eller én tilhørende sikkerhetsnøkkel i Crypto tabellen. Dette vil si at det per nå ikke mulighet til å legge til flere sikkerhetsnøkler uten å erstatte en eksisterende.

Da det ikke er noen to-veis kommunikasjon mellom en gitt TOTP app og systemet, blir det aldri informert om en bruker skulle finne på å slette sikkerhetsnøkkel på sin mobil. Den eneste måten å vite om en rad i Crypto kan erstattes er hvis et forsøk på å registrere en ny TOTP instans kommer inn. Hvis en bruker skal kunne opprette samtlige TOTP tilkoblinger, har systemet ingen måte å vite om gamle nøkler vil kunne slettes. Og Crypto tabellen vil fylles opp med rader som ikke vil kunne slettes med god visshet.

Kodeliste

I kodeliste tabell ligger de forskjellige MFA metoder som er tilgjengelige, TOTP legges til som en data entry blant de diverse koder som tas i bruk i user tabellen.

5 RESULTATER

5.1 Prosjektresultat

5.1.1 Funksjonelle og ikke-funksjonelle krav

Følgende krav er stilt til løsningen. (Se Vedlegg 2: Visjonsdokument)

Tabell 4 Funksjonelle krav-tabell

Funksjonelle Krav	Planlagt	Levert
Produktet skal gjør det mulig å bruke tredjeparts 2FA-TOTP applikasjon for autentisering.	Ja	Ja
Som del av innstillinger for MinID, skal bruker kunne legge til og velge 2FA-TOTP autentisering.	Ja	Ja
Produktet skal tilby mulighet for å endre passord.	Ja	Ja
Produktet skal tilby mulighet for å registrere ny TOTP basert 2FA app som autentiseringsvalg.	Ja	Ja

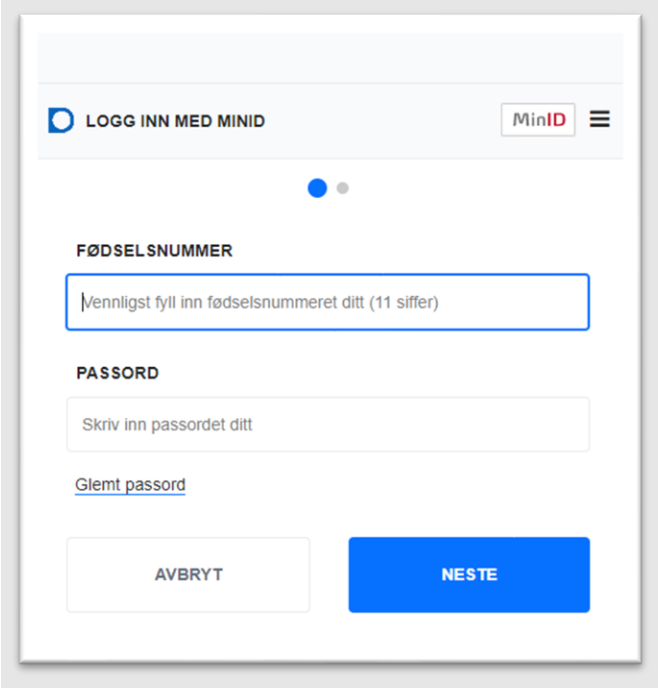
Tabell 5 Ikke-funksjonelle krav-tabell

Ikke-funksjonelle krav	Planlagt	Levert
Brukervennlig i form av antall trykk som trengs fra sluttbruker skal lik som ved bruk av SMS 2FA og sammenlignbar tid brukt ved innlogging som ved bruk av MinID-appen.	Ja	Ja
Integrerte tredjepart autentiserings-app(er) må være blant de mest brukte alternativene.	Ja	Ja
Produktet skal utvikles med Java 17 og Spring Boot.	Ja	Ja
Produktets design og utseende skal være tilnærmet den eksisterende løsningen så langt det lar seg gjøre for å redusere hvor mye tilvenning en bruker skal måtte gjøre.	Ja	Ja
Produktet skal utvikles i henhold til krav stilt av WCAG 2.1 (W3C, 2005)	Ja	Nei

Produktet skal utvikles i henhold til OWASP ASVS (The OWASP Foundation, n.d)	Ja	Ja

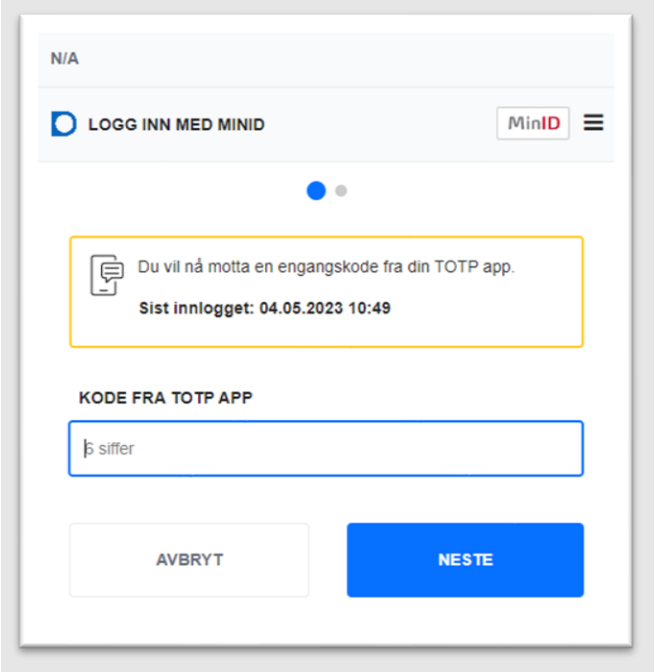
5.1.2 Modul: MinID TOTP Authentication

For brukseksemplet *Autentisering vha. TOTP* vil bruker, etter valg av MinID, først bli presentert for brukerdialog for angivelse av fødselsnummer og passord.



Figur 27 Brukergrensesnitt: Innlogging

Deretter vises dialogen for angivelse av kode generert fra 2FA-TOTP applikasjon (for eksempel Google Authenticator).



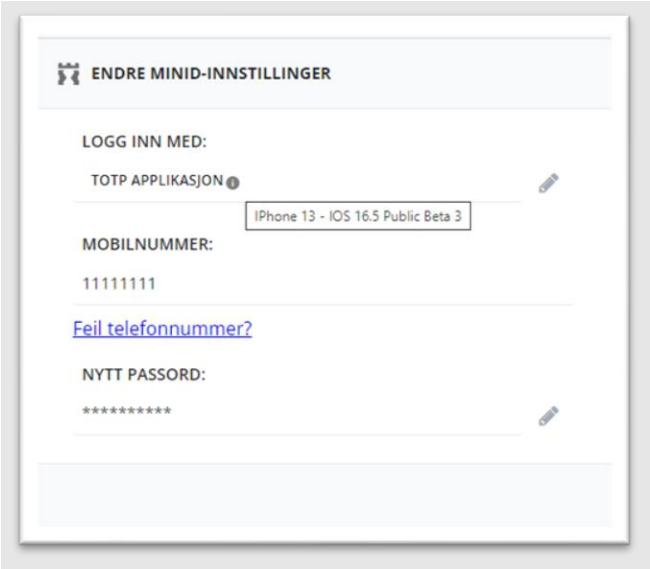
Figur 28 TOTP Brukergrensesnitt

Figur 29 TOTP-koder generert av Google Authenticator

Om angitt TOTP-kode er korrekt og gyldig, blir autentiseringsprosessen fullført.

5.1.3 Modul: MinID Profil

For brukseksemplet *Initialisere 2FA-TOTP autentisering* vil bruker, etter gjennomført pålogging og autentisering, først bli presentert for brukerdialog innstillinger.



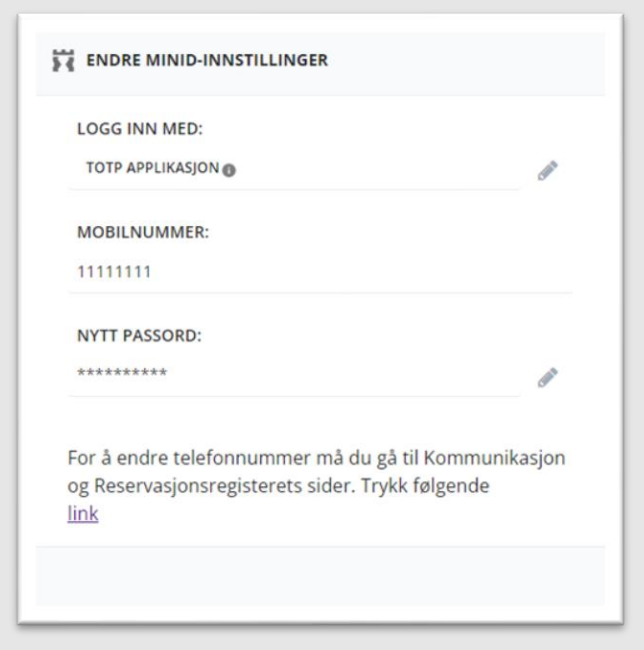
Figur 30 Brukergrensesnitt: Brukerinnstillinger

I **Figur** 30 vises det nye settings viewet. Det er nå støtte for å bruke tredjeparts TOTP applikasjoner som vist under feltet: LOGG INN MED.

Der er det også en info boks som kun skal ligge der hvis brukeren har MinID-app som valgt autentiserings metode, men grunnet at prosjektgruppens løsning ikke har støtte for å bruke MinID-applikasjonen som autentiserings metode, ble det derfor lagt til på TOTP

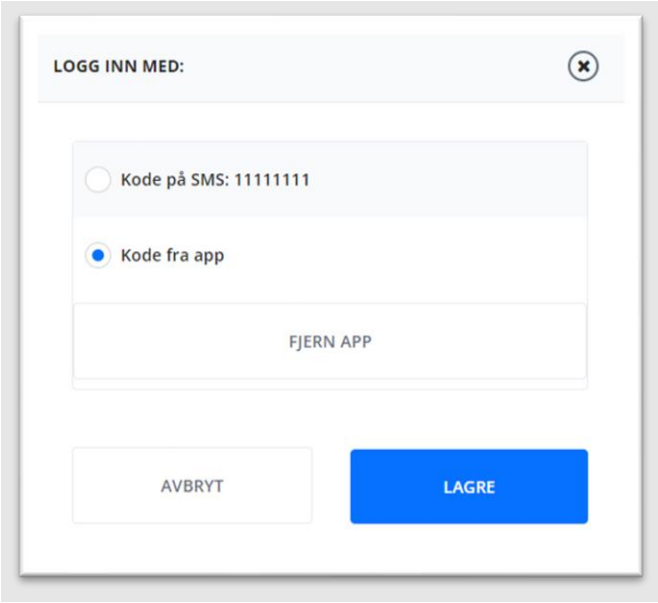
applikasjonen som et representativt visuelt alternativ. Dersom bruker holder musen over informasjonsboksen, vil det vises hvilken enhet og dets operativsystem.

Det er ikke lenger mulig å gjøre endringer på mobilnummer, men derimot kan bruker, dersom mobilnummer ikke stemmer, benytte lenken: “Feil telefonnummer?”. Bruker vil da oppdage at skjermbildet oppdateres som vist i Figur 31. Endring av passord vil foregå på samme måte som tidligere. Neste skjermbilde viser endringen etter at bruker trykker på lenken. Det vises informasjon om at KRR nå håndterer det registrerte mobilnummeret til brukeren, med en lenke til hvor bruker kan endre dette.



Figur 31 Henvisning til KRR

Dersom en bruker trykker på blyanten for å endre innloggingsmetode, er det dette skjermbildet brukeren vil møte. Dersom brukeren allerede har enten MinID-app, eller tredjeparts TOTP applikasjon som valgt innloggingsmetode, vil en knapp med teksten FJERN APP vises.

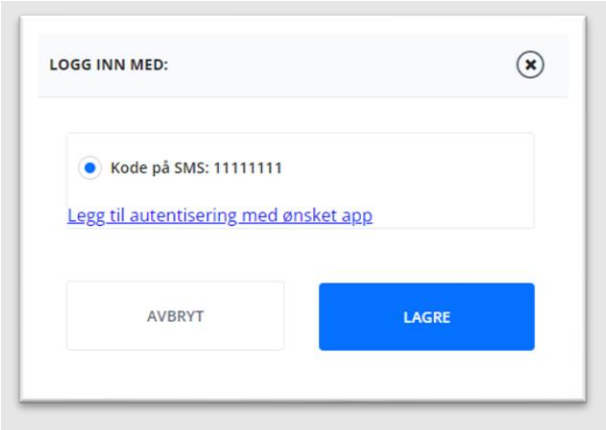


Figur 32 Brukergrensesnitt: Endre 2FA-metode

Når bruker trykker på FJERN APP, blir man bedt om å bekrefte avgjørelsen.

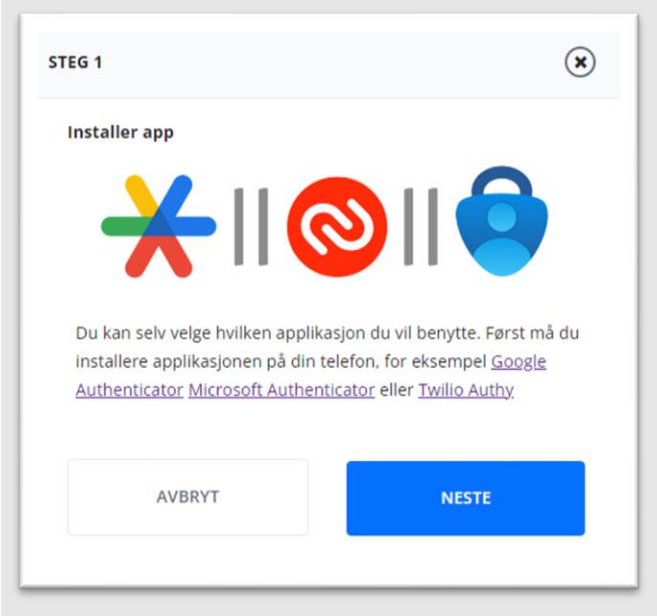
Figur 33 Brukergrensesnitt: Bekrefte avregistrering av app-basert autentisering

Dersom bruker har SMS som valgt innloggingsmetode, vil skjermbilde under vises. Bruker vil da ha muligheten til å trykke på lenken “Legg til autentisering med ønsket app”, for å starte registreringsprosessen.



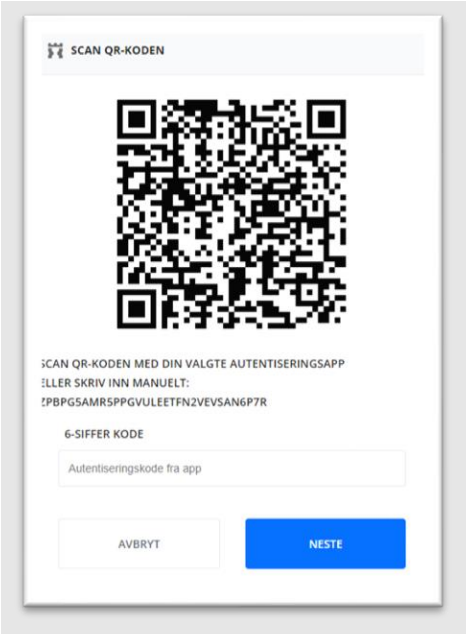
Figur 34 2FA side med SMS som valgt metode

Når bruker trykker lenken, vil skjermbildet under vises. Dette skjermbildet inneholder informasjon om de tre applikasjoner som prosjektgruppen har valgt å basere løsningen rundt med både tekst og bilde. Brukeren har også valget om å avbryte eller fortsette prosessen.



Figur 35 Informasjon om autentiseringsapplikasjoner

Dersom bruker fortsetter prosessen, vil skjermbilde nedenfor vises. Skjermbildet inneholder en QR-kode som bruker kan skanne med tredjeparts 2FA-TOTP applikasjon på en smarttelefon, eller så kan sikkerhetsnøkkelen skrives inn manuelt i appen. Bruker må deretter skrive inn kode som vises i appen i feltet: “Autentiseringskode fra app” for å fullføre prosessen. Koden som skrives inn, blir validert opp mot systemet for å sjekke at oppsettet er fullført. Ved fullført registrering blir bruker sendt tilbake til siden for brukerinnstillinger som vist i Figur 30.



Figur 36 QR-kode visning med manuelt alternativ

Figur 37 TOTP i bruk på app

5.2 Evalueringsmetode

Kommunikasjon mellom oppdragsgiver og utviklere er viktig både før og under utvikling. Verdien av å konkretisere for begge parter hva som skal produseres kan ikke undervurderes, da endringer som blir gjort senere er dyrere enn endringer som gjøres tidlig i prosjektets gang.

5.2.1 Valideringsmetode

Tidlig i prosjektet ble det utformet UX-skisser for en mulig løsning. Disse skissene illustrerer systemet med tanke på både form og funksjon. Skissene ble senere interaktive for å kunne gi et innblikk i flyten som en bruker går igjennom i de forskjellige brukstilfeller. Etter hvert som prosjektet gikk over i en utviklingsfase, ble oppdragsgiver presentert for demoer av programmet under kjøring. Demoer ble holdt annenhver uke, hvor oppdragsgiver fikk mulighet til å komme med innspill tidlig i utviklingsprosessen. Dette ga begge parter en bedre oversikt over prosjektets status.

Under utvikling har det blitt kontinuerlig utført brukertester, hvor prosjektgruppen har inntatt rollen som sluttbruker. Potensielle feil som oppstår ved endring eller ny kode blir dermed tidlig oppdaget og fikset. Som del av evaluering ble det og utført brukertesting av eksterne representanter av sluttbruker. De eksterne testerne besto av medstudenter med begrenset kjennskap til problemdomenet, som fikk utgitt et script for testprosessen med gitt testbruker.

5.2.2 Verifikasjonsmetode

For å skape best mulig produkt er det viktig å verifisere at produktet fungerer på riktig måte. Det ble drevet TDD i begynnelsen av prosjektet, men ble mindre fokusert på underveis i prosjektet. Likevel er det laget enhetstester for størsteparten av koden. Noe av koden byr på utfordringer når det kommer til å teste den. Et eksempel på dette kan være testing av QR-kode generering og de diverse appers evne til å skanne disse. Dette lar seg ikke skrive enhetstester for på en lett måte, men det som kan testes er hvorvidt QR-koder lages på et forventet format.

For elementer av prosjektet som TOTP og databaseoppsettet ble det gjennomgått en teknisk prototyping som var isolert fra resten av prosjektet. Når dette i seg selv viste seg å fungere, og hadde forventet oppførsel, ble det sett på som modent for innføring til resten av prosjektet.

Det ble gjennomgått utforskning på de ulike tredjeparts applikasjoner for å finne ut av hvilke algoritmer de støtter. Dette ble gjort for at prosjektgruppen skulle klare å lage en løsning som støttet størst mulig brukermasse i forhold til disse applikasjonene. Det ble utført utprøving av de 3 største applikasjoner, Microsoft Authenticator, Google Authenticator og Twilio Authy.

Under utvikling ble kode lagret på GitHub for versjonskontroll, hvor validert kode lå på en master branch og ved endringer, eller ny funksjonalitet ble det opprettet en «pull request». Slik fikk prosjektgruppen mulighet til å inspisere hverandres kode før den kunne flettes inn

på master. I tilfeller hvor det kunne være konflikter i koden, eller store mengder kode skulle flettes inn, ble det utført slike inspeksjoner for å verifisere at hverandres arbeid var akseptabelt.

5.3 Evalueringsresultat

5.3.1 Resultat fra validering

Validering av tidlige wireframes påvirket blant annen hvordan registrering av 2FA-TOTP som autentiseringsmetode skulle foregå. Tidlig løsningsdesign indikerte at registreringsprosessen skulle kunne startes fra innloggingsiden, noe som potensielt kunne gjøre det enklere for nye MinID-brukere å komme i gang. Etter tilbakemelding ble denne funksjonaliteten flyttet til brukerinnstillinger-siden. Denne løsningen gjør det mindre imøtekommende for potensielle nye brukere, men er mer intuitivt siden det ligger på samme sted som andre MFA-relaterte funksjoner.

I Figur 38 ligger en oversikt over scriptet som ble brukt under ekstern brukertesting, samt forventede resultater og faktiske resultater. Disse brukertestene ble utført sent i iterasjon 4, i etterkant av intern brukertesting.

	Beskrivelse	Forventet resultat	1. brukertester	2. brukertester
1	Logg inn med gitt testbruker ved hjelp av SMS autentisering.	Tester når fram til siden for brukerinnstillinger ved å autentisere seg med gitt testbrukerinformasjon.	Som forventet.	Som forventet.
2	Gjennomfør registreringsprosessen for TOTP autentisering på siden for brukerinnstillinger.	Tester skanner QR-koden med tredjeparts 2FA-TOTP app, skriver koden de ser på appen som forventet input og blir tatt tilbake til siden for brukerinnstillinger.	Som forventet.	Som forventet.
3	Sjekk hva som står valgt under "logg inn med".	Totp applikasjon blir automatisk valgt etter registrering.	Som forventet.	Som forventet.
4	Logg ut.	Tester blir tatt tilbake til innloggingsiden ved gitt link.	Som forventet.	Som forventet.
5	Logg inn igjen, denne gangen med TOTP-kode fra tredjeparts 2FA-app.	Tester gjennomfører vellykket autentisering ved hjelp av TOTP-koden de finner på 2FA-TOTP appen sin.	Som forventet.	Som forventet.
6	Fjern den nylig registrerte autentiseringsmetoden under "logg inn med".	Etter den nye autentismetoden blir fjernet av tester, ender man opp på siden for brukerinnstillinger igjen med SMS som valgt innloggingsmetode.	Som forventet.	Som forventet.
7	Fritt fram.		Tester endret passord og fulgte telefonnummer link til KRR sine sider.	Tester navigerte seg og fram til KRR sine sider. Ga tilbakemelding som førte til endring i avregistreringsprosessen.

Figur 38 Resultater fra ekstern brukertesting

5.3.2 Resultat fra verifikasjon

Kjernefunksjonaliteten i prosjektet er autentisering av TOTP-koder. Det har derfor vært viktig at kode som omhandler TOTP-kodevalidering, generering av TOTP-koder, nøkler og QR-koder dekkes av testingen.
Enhets-testene dekker 100% av klassene innenfor dette området, 70% av metodene og 80% av kodelinjer. Dette er ikke en perfekt måleenhet for hvor omfattende testingen er da noe av kode som er blitt utelatt, tilhører getters, setter og annen genererbar kode. I tillegg finnes det utallige måter å teste samme metode på for å finne uforutsette feil. Enhetstestene bidrar likevel til en god robusthet for et senere tidspunkt om refaktorering skulle forekomme. De vil også ha en mindre rolle som dokumentasjon av koden som testes hvor navngivning på testene er beskrivende for oppførselen blant klasser og metoder, som vist i Figur 39.

Figur 39 Tester med navn som beskriver oppførsel

Underveis i utviklingen har kontroller-klassene blitt implementert, og over lengre tid har disse klassene inneholdt en økende grad av logikk for å håndtere requests. Dette har gitt utfordringer å skape tester for kontrollerne med tanke på enhetstester, og har resultert i en høy grad av brukertesting fra utviklernes side. Sent ut i prosjektet ble denne logikken trukket ut og håndtert av en enkel klasse (Utils), noe som gir mulighet for å skrive ytterligere tester.

5.4 Prosjektgjennomføring

Prosjektgjennomføringen har for det meste gått smidig, med noen utfordringer tidlig i utviklingsfasen. Arbeidslokalet var til tider utfordrende å jobbe i, da det ble ofte dårlig luft utover dagene, dette gjorde at energinivået til prosjektgruppen ble tidvis senket. Det var i begynnelsen utfordrende å bruke Spring Boot, da prosjektgruppen ikke hadde erfaring med Spring Boot, men effektiviteten økte ettersom prosjektgruppen tilegnet seg denne erfaringen. Prosjektgruppen brukte mye tid på å forsøke å benytte DigDirs APIer, men ble nødt til å avgrense oppgaven da resultatene av dette ikke var optimale. Underveis i utviklingen ble prosjektgruppen anmodet til å redusere den planlagte lengden på utviklingsfasen. I den originale prosjektplanen strakk utviklingsfasen seg utover 10 uker, og besto av 5 iterasjoner. Grunnet stor fremgang og for å sørge for at nok tid var til rådighet til rapportskriving, ble denne fasen kortet ned til 4 iterasjoner over 7 uker. Selv med færre iterasjoner og redusert tid ble målene satt av funksjonelle og ikke-funksjonelle krav i mindre grad påvirket.

Tidsbudsjettet som ble satt tidlig i planleggingen har vist at prosjektet har hatt en raskere fremdrift enn det som ble planlagt. Det har også tidvis vært utfordringer grunnet andre forpliktelser som prosjektmedlemmene har hatt i løpet av prosjektets gang.

Det var planlagt å benytte TDD under utvikling, og det var tilfellet i begynnelsen av utviklingsfasen, men grunnet mangel på kunnskap og erfaring med metodikken frafalt dette i løpet av utviklingen. Det ble derimot brukt enhetstesting for store deler av metoder i programmet.

6 DISKUSJON

6.1 Støtte for 3.parts TOTP App

Valget av hvilke tredjeparts applikasjoner var for det meste basert på å lage støtte for de tre største. Grunnet manglende informasjon om hvordan de fungerte, tilsa dette at prosjektgruppen måtte utforske hvordan de fungerte og hvilke algoritmer de støttet. Denne utforskningen avdekket at alle de tre applikasjonene støttet SHA-1, mens Google Authenticator støttet SHA-256 og SHA-512 i tillegg, men bare på iOS. Dermed falt valget på å støtte SHA-1, slik at løsningen kunne treffe størst brukermasse. Dersom det hadde vært et krav om å bruke andre hashing-algoritmer måtte det ha blitt gjort en større undersøkelse opp imot hvilke applikasjoner som støttet de algoritmer som kreves.

En utfordring som dukket opp under utforskningen er at alle tre applikasjonene som ble prøvd, "støtter" alle algoritmene. Med det menes at en QR-kode generert med andre SHA-algoritmer enn SHA-1 vil resultere i at applikasjonen vil gi en tallkode tilbake og visuelt indikere at det fungerer. Dette skjer fordi applikasjonene klarer å tolke QR-koden, men hvis algoritmen i QR-koden er noe annet en SHA-1, vil applikasjoner som ikke støtter SHA-256 og SHA-512 falle tilbake på SHA-1. Dette resulterer i at bruker oppfatter at alt fungerer som det skal, da appen genererer engangspassord som vanlig. Denne koden vil ikke stemme med hva systemet forventer.

Resultatet av å støtte SHA-1 er i teorien at alle typer tredjeparts 2FA-TOTP applikasjoner kan benyttes, ikke bare de som er nevnt i rapporten.

Dersom en ikke skal støtte SHA-1, men heller kreve SHA-256 eller SHA-512 ligger kode allerede i systemet til å benytte dette. Det som må gjøres er å utforske hvilke applikasjoner som støtter disse algoritmene, og informere om disse til bruker på informasjonssiden. Dersom en vil støtte flere enn 1 algoritme er det nødvendig å lagre hvilken algoritme som en bruker benytter i databasen, mest fornuftig vil da og legge det til under User tabellen i kolonnen mfa_method der prosjektgruppen har lagt til TOTP som valg for bruker. Dette kan da endres til for eksempel TOTP (256) eller TOTP (512). Dette vil igjen kreve enten veldig god informasjon på informasjonssiden eller at bruker allerede har innsikt i hvilke algoritmer en applikasjon støtter.

6.2 Sikkerhet og valg av hashing-algoritmer

Prosjektgruppen hadde en tidlig hypotese om at valget av SHA algoritme ville ha en påvirkning på sikkerheten av TOTP genereringen. Dette var basert på at SHA1 var sett på som utrygg i sammenheng med hashing av passord (Stevens, *et al*., 2017).

For TOTP-generering derimot; er SHA1 en solid komponent. Dette kommer av at koden som skal knekkes for TOTP kun eksisterer innenfor et lite tidsvindu som er definert av programmet (30 sekunder i prosjektet). I etterkant av tidsintervallet vil tidligere forsøk ikke ha noe verdi, da de ikke utelukker koder for fremtidige intervall.

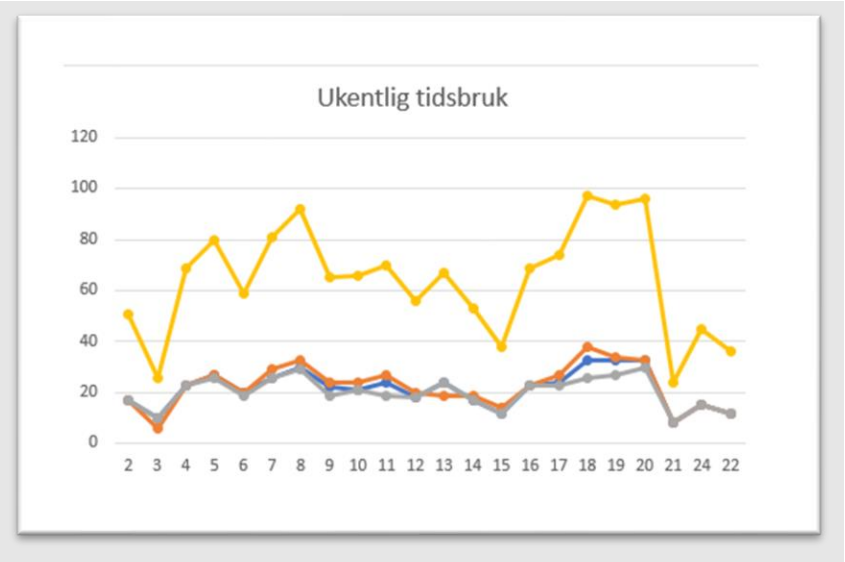
Autentiseringen av en engangskode vil skje på serverens side, dette innebærer at alle forsøk sendes over nett. Det vil gjøre et Brute Force forsøk vanskelig med et begrensede antall forsøk innenfor tidsrammen. I tillegg med nettverkshastighet som flaskehals, vil det være mulig å låse en bruker ute etter 3 feilede forsøk for en gitt tidsperiode.

6.3 Utviklingsmetode og prosjektgjennomføring

Utviklingsmetoden og prosjektgjennomførings-planen som ble laget tidlig har vært en stor bidragsyter i forhold til resultatet. Det ble tidlig gjennomgått initielle skisser og prototyper med både oppdragsgiver og veileder. Dette var med på å skape felles forståelse for hvilke funksjonelle og ikke-funksjonelle egenskaper løsningen ville ha. Det å tidlig dele opp de ulike egenskapene som skulle bli implementert i iterasjoner var med på å skape en felles forståelse av hva som burde bli prioritet, noe som hjalp prosjektgruppen til å planlegge utviklingen. Selv med anbefalt endring underveis i utviklingsfasen gikk det uansett ikke på bekostning av funksjonalitet. Hva som inngikk i hver iterasjon ble også grundig gjennomgått i samvær med oppdragsgiver og veileder, noe som ga prosjektgruppen tilbakemeldinger både før, under og etter utviklingen, som igjen bidro til å skape en felles forståelse for funksjonalitet, fremgang og resultat.

Til prosjektet ble prosjektgruppen lovet et kontorlokale som var under produksjon, men som kom til å ta litt tid før det sto ferdig. I mellomtiden kunne prosjektgruppen disponere ett kontor. Disse løsningene ble levert av Campus Verftet, og gjorde at prosjektgruppen hadde et fast sted å arbeide. Prosjektgruppen jobbet hovedsakelig på felles lokale, noe som gjorde at kommunikasjonen innad i gruppen var effektiv. Kommunikasjon mellom prosjektgruppen og veileder besto delvis av skriftlig kommunikasjon på Discord (Discord, n.d), og møtevirksomhet både fysisk og over Teams. Mellom oppdragsgiver å prosjektgruppen var kommunikasjonen på Slack for skriftlig, og Teams for møter, med unntak av oppstartsmøte som foregikk fysisk på DigDirs lokaler i Leikanger.

Det var noen utfordringer med prosjektgjennomføringen, en del i forhold til mye endringer og avgrensninger i begynnelsen som førte til noen grad av usikkerhet rundt oppgaven. I tillegg var det jevnt over mindre tidsbruk enn det som var planlagt. Selv om timebruken er lavere enn planlagt (Figur 40), viser resultatet at prosjektet har utført alle planlagte aktiviteter, og levert en løsning i henhold til definert krav.



Figur 40 Felles ukentlige arbeidstimer

Utviklingsmetodikken og utviklingen generelt ga også en del utfordringer. Testdrevet utvikling var en metodikk som ble planlagt å bruke i større grad enn det faktisk ble. Det var noe kode som krevde kryptografisk tilfeldige verdier som var tilnærmet umulig å teste. Kunnskapen til prosjektgruppen rundt denne metodikken var også mangelfull, slik at gjennomføringen av TDD frafalt. Bruken av DigDirs APIer ble brukt mye tid og ressurser på å forsøke å implementere, dog uten suksess. Derfor ble det benyttet egenlagde løsninger som fungerer på en indikativ måte.

6.4 Tofaktorautentisering med tidsbasert engangspassord for økt bruk av MinID

En stor del av bachelorprosjektet var å utforske om 2FA med TOTP for MinID kunne senke terskelen for å ta MinID i bruk og treffe flere brukere. Sammenlignet med de andre alternativene som finnes i dag (SMS, MINID-APP), er det en del punkter som prosjektets løsning er bedre på.

Sammenlignet med SMS er prosjektgruppens løsning bedre eller like god på alle områder. Som nevnt tidligere i rapporten er ikke SMS regnet som en like sikker løsning som app-basert tofaktorautentisering. *Tofaktorautentisering (2FA) ved bruk av Tidsbasert engangspassord (TOTP) i MinID* har derfor en fordel ved at det er en sikrere løsning enn SMS, fordi det skjer kun toveis kommunikasjon ved den opprinnelige oppsetting av TOTP. Løsningen er også enkel å ta i bruk dersom brukeren allerede har en tredjeparts 2FA-TOTP applikasjon installert, ved at brukeren kun trenger å skanne en QR-kode for å kunne bruke løsningen. Dersom brukeren ikke har en tredjeparts TOTP applikasjon installert, vil muligens løsningen være mer utfordrende for en bruker, med at bruker først må installere en applikasjon før den kan tas i bruk. Hovedsakelig henvender prosjektgruppens løsning seg mot den brukermassen som allerede har en eller flere slike applikasjoner tilgjengelig. Tredjeparts 2FA-TOTP applikasjoner er blitt mer og mer utbredt i dagens samfunn, og flere tjenester har som krav eller oppfordring om at brukere benytter en slik form for tofaktorautentisering.

Sammenlignet med MinID-applikasjonen har ikke prosjektgruppens løsning noen nevneverdig forskjell i forhold til sikkerhet. Det kan diskuteres om løsningen er vanskeligere å bruke grunnet at sluttbrukeren må i begge tilfeller åpne en app og i MinID-appen trykke godkjenne, mot i løsningen må brukeren åpne appen, lese koden og deretter skrive inn koden i brukergrensesnittet. Hvis man da teller tastetrykk, vil løsningen være noe mindre effektiv å bruke.

Det er mindre sannsynlig at eksisterende brukere av MinID benytter prosjektgruppens løsning sammenlignet med nye brukere. Dette fordi at eksisterende brukere kan være mer motvillig til å endre 2FA metode, sammenlignet med nye brukere.

Prosjektgruppen mener at løsningen i stor grad henvender seg mot sluttbrukere som ikke benytter seg av MinID fra før og har minst én tredjeparts 2FA-TOTP applikasjon som de benytter. Innføringen av *Tofaktorautentisering (2FA) ved bruk av Tidsbasert engangspassord (TOTP) i MinID*, vil trolig kunne bidra til at flere velger å benytte MinID for autentisering, men det er kanskje ikke grunn til å tro at dette alene skal bidra til store endringer i antall brukere.

7 KONKLUSJON OG VIDERE ARBEID

7.1 Konklusjon

Målet med dette prosjektet var å utvikle et produkt som gjorde det mulig å benytte 2FA-TOTP autentisering ved bruk av tredjeparts autentiseringsapplikasjoner i MinID. Produktet skulle ha noen endringer av MinID-innstillinger, både på grunn av ny autentiseringsmetode, og at det ikke skulle være mulig å endre autentiseringsmetode til PIN. I tillegg skulle det ikke være mulig for en bruker å endre telefonnummer, da dette blir hentet fra KRR.

Prosjektgruppen kom frem til følgende forskningsspørsmål:

- *Hvordan kan man implementere en løsning for autentisering ved bruk av 2FA-TOTP, og tredjeparts autentiseringsapplikasjoner MinID.*

Resultatet bachelorgruppen endte opp med, er en applikasjon hvor en sluttbruker har mulighet til å registrere og benytte tredjeparts 2FA-TOTP applikasjon i MinID. Sluttbruker kan også fjerne applikasjonen som autentiseringsvalg, samt endre passord. Bruker får presentert informasjon om enhet applikasjonen er installert på ved hjelp av en midlertidig løsning, da nødvendig kobling til brukers enhet ikke er implementert. Prosjektgruppen har gjennomført regelmessige demonstrasjoner av funksjonalitet i produktet for å få validert løsningen. Basert på tilbakemeldinger fra oppdragsgiver på disse demonstrasjoner, vurderes løsningen som er utviklet for autentisering ved bruk av tredjeparts 2FA-TOTP autentiseringsapp, å være god og i henhold til kravene. De andre krav som den opprinnelige oppgaven besto av, det vil si kravene om MinID-Innstillinger, er ikke løst på en like god måte. Avgrensingene og avklaringene gjorde at relevansen til disse kravene frafalt noe ved at produktet ikke skulle kommunisere med dagens system i like stor grad. Prosjektgruppen vil likevel konkludere at produktet uansett er en representativ løsning for de originalt dokumenterte kravene. Det ble derimot ikke lagt stor vekt på universell utforming og WCAG prinsipper som var en del av ikke-funksjonelle krav. Grunnen til dette er at funksjonalitet ble prioritert som mer essensielt.

7.1.1 Videre arbeid

Sikkerhet:

For å ta prosjektgruppens løsning og sette den i et produksjonsmiljø, vil prosjektgruppen anbefale å ha en eks*tern lagring* av sikkerhetsnøkkel for å unngå at direkte angrep kan gi en angriper tilgang til både brukerinformasjon og sikkerhetsnøkkelen.

"The key store MUST be in a secure area, to avoid, as much as possible, direct attack on the validation system and secrets database. Particularly, access to the key material should be limited to programs and processes required by the validation system only."
(M'Raihi, et al., 2011)

Videreutvikling:

Andre ting som kan/bør gjøres er å oppdatere nye views slik at de følger WCAG standarder. Det kan også være naturlig og endre to av prosjektgruppens views til å heller være JavaScript popup. Spesielt da henvisning til KRR (Figur 31) og informasjon om applikasjoner (Figur 35), siden de ikke har noe funksjonalitet utover å informere sluttbruker.

Utils-klassen er for omfattende og har forbedringspotensial. Prosjektgruppen vil anbefale å i hvert fall dele opp klassen slik at én klasse ikke holder alt av verktøy. En mulig løsning kan være å dele opp klassen slik at det finnes en utils klasse for MinID-Profil og en for MinID-Authentication, om ikke gjøre en større oppdeling basert på spesifikke ansvarsområder.

Det er mulig å benytte andre SHA-algoritmer om ønskelig, da det ligger løsninger for dette i koden (Se vedlegg: Kildekode). En endring i hvilken generateTOTP metode som kalles i runTOTP metoden vil endre slik at programmet benytter SHA-256, eller SHA-512-algoritmer om ønskelig, da det ligger løsninger for dette i koden. I tillegg må verdien for hvilken SHA-metode som brukes i QrSetup endres til ønsket SHA-metode.

Gjenbruk:
Deler av løsningen kan i teorien benyttes av andre som har et ønske om å implementere en 2FA-TOTP løsning da vår løsning følger RFC-6238 standarden for TOTP. RFC-6238 går ut på å skape en åpen og delt algoritme som legger til rette for en interoperabilitet mellom kommersielle og åpen-kildekode løsninger for tofaktorautentisering med TOTP.

8 REFERANSER

Agile Alliance (n.d) *What is Agile Software Development? | Agile Alliance.* Tilgjengelig fra: https://www.agilealliance.org/agile101/ (Hentet: 21. mai 2023).

Digitaliseringsdirektoratet (n.d) *Kva er Digitaliseringsdirektoratet? | DigDir.* Tilgjengelig fra: https://www.digdir.no/digdir/kva-er-digitaliseringsdirektoratet/703/ (Hentet 23. februar 2023)

Discord (n.d) *Discord | Your Place to Talk and Hang Out.* Tilgjengelig fra: https://discord.com/ (Hentet 21. mai 2023)

Edwards, B. (2022) *Why SMS Needs to Die.* Tilgjengelig fra: https://www.howtogeek.com/787957/why-sms-needs-to-die/ (Hentet: 22. februar 2023)

Figma (n.d) *Figma: the collaborative interface design tool.* Tilgjengelig fra: https://www.figma.com/ (Hentet 24. februar 2023)

Fornyings- og administrasjonsdepartementet (2008) *Slutt på PIN-kodekaos | regjeringen.no.* Tilgjengelig fra: https://www.regjeringen.no/no/dokumentarkiv/stoltenberg-ii/fad_2006-2009/nyheter-og-pressemeldinger/nyheter/2008/slutt-pa-pin-kode-kaos/id538909/ (Hentet 23. februar 2023)

GitHub (n.d) *About* Tilgjengelig fra: https://github.com/about (Hentet 24. februar 2023)

Google authenticator (n.d) *Google.com.* Tilgjengelig fra: https://play.google.com/store/apps/details?id=com.google.android.apps.authenticator2 (Hentet: 07. mars 2023).

HeidiSQL (n.d) *HeidiSQL* Tilgjengelig fra: https://www.heidisql.com/ (Hentet 16. mai 2023)

IntelliJ IDEA (n.d) *IntelliJ IDEA – the Leading Java and Kotlin IDE* Tilgjengelig fra: https://www.jetbrains.com/idea/ (Hentet 24. februar 2023)

Krawczyk, H., Bellare, M., og Canetti R. (1997) *HMAC: Keyed-Hashing for Message Authentication* Tilgjengelig fra: https://www.rfc-editor.org/rfc/rfc2104 (Hentet 08. mars 2023)

Kristoffersen, B. (2020) *"8.1 Normalisering," i Databasesystemer*. S.l.: UNIVERSITETSFORLAGET.

MariaDB (n.d) *About MariaDB Servers* Tilgjengelig fra: https://mariadb.org/about/ (Hentet 24. februar 2023)

Microsoft 365 (n.d) *Microsoft 365* Tilgjengelig fra: https://www.microsoft365.com/ (Hentet 24. februar 2023)

Microsoft authenticator (n.d) *Google.com.* Tilgjengelig fra: https://play.google.com/store/apps/details?id=com.azure.authenticator (Hentet: 07. mars 2023).

Mozilla (2023) *MVC –MDN Web Docs Glossary* Tilgjengelig fra: https://developer.mozilla.org/en-US/docs/Glossary/MVC (Hentet 16. mai 2023)

M'Raihi, D. et al. (2005) *HOTP: An HMAC-Based One-Time Password Algorithm* Tilgjengelig fra: https://www.rfc-editor.org/rfc/rfc4226 (Hentet 08. mars 2023)

M'Raihi, D. et al. (2011) *TOTP: Time-Based One-Time Password Algorithm* Tilgjengelig fra: https://www.rfc-editor.org/info/rfc6238 (Hentet 08. mars 2023)

Oracle (n.d) *Java Software* Tilgjengelig fra: https://www.oracle.com/java/ (Hentet 24. februar 2023)

OWASP Foundation (n.d.). OWASP Application Security Verification Standard. Tilgjengelig fra: https://owasp.org/www-project-application-security-verification-standard/ (Hentet 23. januar 2023).

OWASP ZAP (n.d) *OWASP ZAP.* Tilgjengelig fra: https://www.zaproxy.org/ (Hentet 21. mai 2023)

Slack (n.d) *Features | Slack* Tilgjengelig fra: https://slack.com/features (Hentet 24. februar 2023)

Spring Boot (n.d) *Spring Boot* Tilgjengelig fra: https://spring.io/projects/spring-boot#overview (Hentet 24. februar 2023)

Spring.io (n.d) *Why Spring?* Tilgjengelig fra: https://spring.io/why-spring (Hentet 24. februar 2023)

Stevens, M. et al. (2017) *Announcing the first SHA1 collision* Tilgjengelig fra: https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html (Hentet 08. mars 2023)

Teams (n.d) *Microsoft Teams* Tilgjengelig fra: https://www.microsoft.com/en-us/microsoft-teams/group-chat-software/ (Hentet 24. februar 2023)

Thymeleaf (n.d) *Thymeleaf* Tilgjengelig fra: https://www.thymeleaf.org/ (Hentet 24. februar 2023)

Trello (n.d) *Manage Your Team's Prosjects From Anywhere.* Tilgjengelig fra: https://trello.com/ (Hentet 24. februar 2023)

Twilio (n.d) *What is a Time-based One-time Password (TOTP)?* Tilgjengelig fra: https://www.twilio.com/docs/glossary/totp (Hentet 12. mai 2023)

Twilio Authy authenticator (n.d) *Google.com.* Tilgjengelig fra: https://play.google.com/store/apps/details?id=com.authy.authy (Hentet: 07. mars 2023).

W3C (2005). *WCAG 2 Overview | Web Accessibility Initiative (WAI) | W3C.* Tilgjengelig fra: https://www.w3.org/WAI/standards-guidelines/wcag/ (Hentet: 24. januar 2023).

Willoughby, M. (2005) *SHA-1 flaw seen as no risk to one-time password proposal* Tilgjengelig fra: https://www.computerworld.com/article/2556478/sha-1-flaw-seen-as-no-risk-to-one-time-password-proposal.html (Hentet 08. mars 2023)

9 VEDLEGG

1. Prosjekthåndbok
2. Visjonsdokument
3. Kravdokument
4. Systemdokumentasjon
5. Kildekode