# Server

Vi har valgt å benytte ASP.net Core API til å håndtere Backend av vår applikasjon. Det er mye som er ferdig oppsatt med kodeeksempler som gjør at å sette opp et slikt API går ganske raskt når alt er på plass fra før.

Siden ene på gruppen allerede har en SQL-server kjørende valgte vi å sette lagringen direkte til denne i første omgang, denne er nå endret på til å bruke Apache Cassandra for å håndtere data.

.net Core WebAPI har innebygd mange deler av sikkerhet fra før slik at vi ikke trenger å bygge opp alt fra grunnen.

Metodene er beskrevet i et annet dokument.

CORS på serveren er satt opp til å godta alle forespørsler, Dette er gjort for å enklere sette det opp. I en reell sitasjon burde nok disse være satt opp slik at serveren kun responderte på ønskede http-metodene som er satt opp ellers i løsningen.

Kontrollerne som er satt opp styrer hvert sitt område slik at det er enkelt å videreutvikle. Hvert område (User, Event og Reminder) er knyttet opp mot et DAO og et DTO. DAO (Data Access Object) er en instans som knytter løsningen opp mot lagringsenheten. Siden vi bruker Apache Cassandra for datalagring vil API koden måtte ta unna det meste av filtrering. Dette for en raskere datatrafikk løsning. DAO klassene inneholder metoder for å kunne kommunisere på enklere måter mot databasen. For å sende data videre til klienter har vi valgt å benytte DTO (Data Transfer Object). Disse klassene forminsker mengden data som sendes mellom klient og API. Det er ikke alle feltene som er relevant å sende videre til en forbruker. Enkelte felt i et DTO kan bli nullet ut basert på rettigheter registrert i databasen.

Det mest sensitive av data er lagret kryptert i databasen. Krypteringen som er satt opp er SHA-512, optimalt kunne man også legge på salting på denne krypteringen i databasen, kun laget med en sammenkobling på brukeren. Saltet vil da være lagret på brukeren i databasen, men kun benyttet av DAO klassen, dermed ikke bli sendt videre. Denne kan da godt krypteres med en privat nøkkel uten en publisert offentlig nøkkel for bedre sikkerhet om noen skal få ugyldig tilgang til databasen.

# Autentisering

I database tabellen for en bruker er det en kolonne som brukes for JWT-token og en annen som inneholder et Timestamp. Måten disse brukes på er når bruker logger inn, ved å sende fødselsnummer og passord, responderer API med en JWT-token til Android applikasjonen. Denne JWT-token blir også lagret i bruker tabellen med Timestamp for innloggings tidspunkt.

Alle forespørsler som klienten (Android applikasjonen) nå sender til API inneholder denne JWT-token i headeren. API bruker så denne for å autentisere alle forespørsler ved å sjekke om JWT-token eksisterer i databasen, og at timestampet ikke er gått lengere enn 30 minutter. Hvis klienten ikke lengre er autentisert vil Android applikasjonen sende bruker tilbake til innloggings skjermen for re-autentisering. Om brukeren fremdeles er autentisert ved forespørsler oppdateres timestampet for å tillate 30 ytterlige minutter med inaktivitet før re-autentisering blir nødvendig.