Dokumentasjon

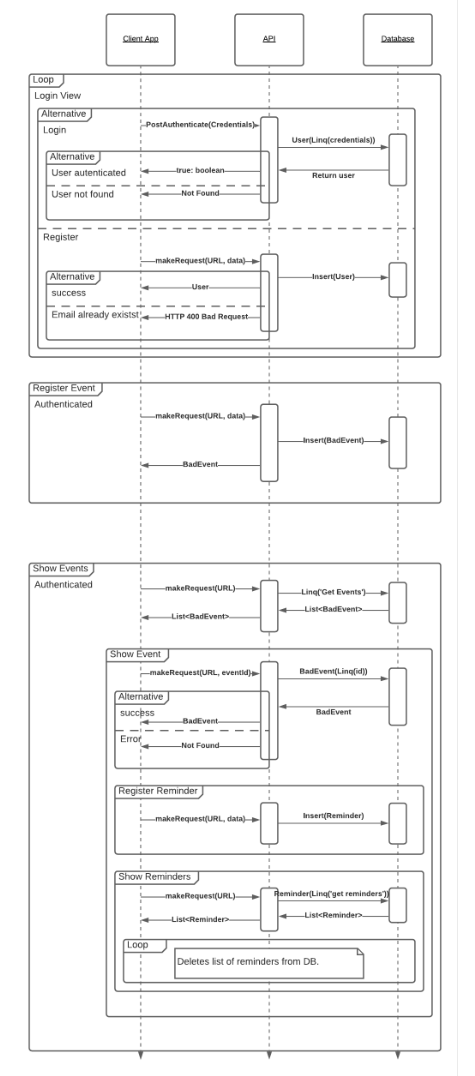
Gruppe 5

Rest server API

Oppgave 4

Daniel Eli, Iuliia Ukraynets, Eskil Hardy Johnsen

# UML Sekvensdiagram



Dette diagrammet viser hvordan applikasjonen vår fungerer generelt, viser skjematisk det koordinerte klasserarbeid, metoder og funksjoner. Viser hvordan data overføres fra klient til databasen og hvilken rolle spiller API. Vi bestemte oss å bruke i vår applikasjonen klient – server arkitektur fordi det gjør det enkelt å løse de angitte oppgavene, og utfører funksjonene. Siden serveren husker ingenting om brukeren som bruker API, så all nødvendig informasjon for å behandle forespørselen må leveres av klienten på hver forespørsel. Klient-server-arkitektur gjør det også mulig å sikre pålitelighet og sikkerhet, samtidig når øker belastningen, skalere serveren og holde applikasjonen vår tilgjengelig.

Denne arkitekturen gir oss mulighet til å legge til nye objekter og funksjoner uten å bryte programmets hovedstruktur. Dette lar oss mulighet legge til eller endre funksjonalitet etter behov. Dette diagrammet viser hvordan hovedfunksjonene til klient-server applikasjonen fungerer. Hver blokk tilsvarer en bestemt klasse og hvordan databevegelse foregår i denne klassen. Hvordan funksjonene er involvert, og hva klienten mottar som svar på hvert enkelt skjermaklikk. I det siste bør bare understrekes at det ikke er helt rett tegnet opp nederst. Det er fordi nettsiden som var brukt ikke tillater flere 'piler' uten å betale for 'programmet'.

# 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.

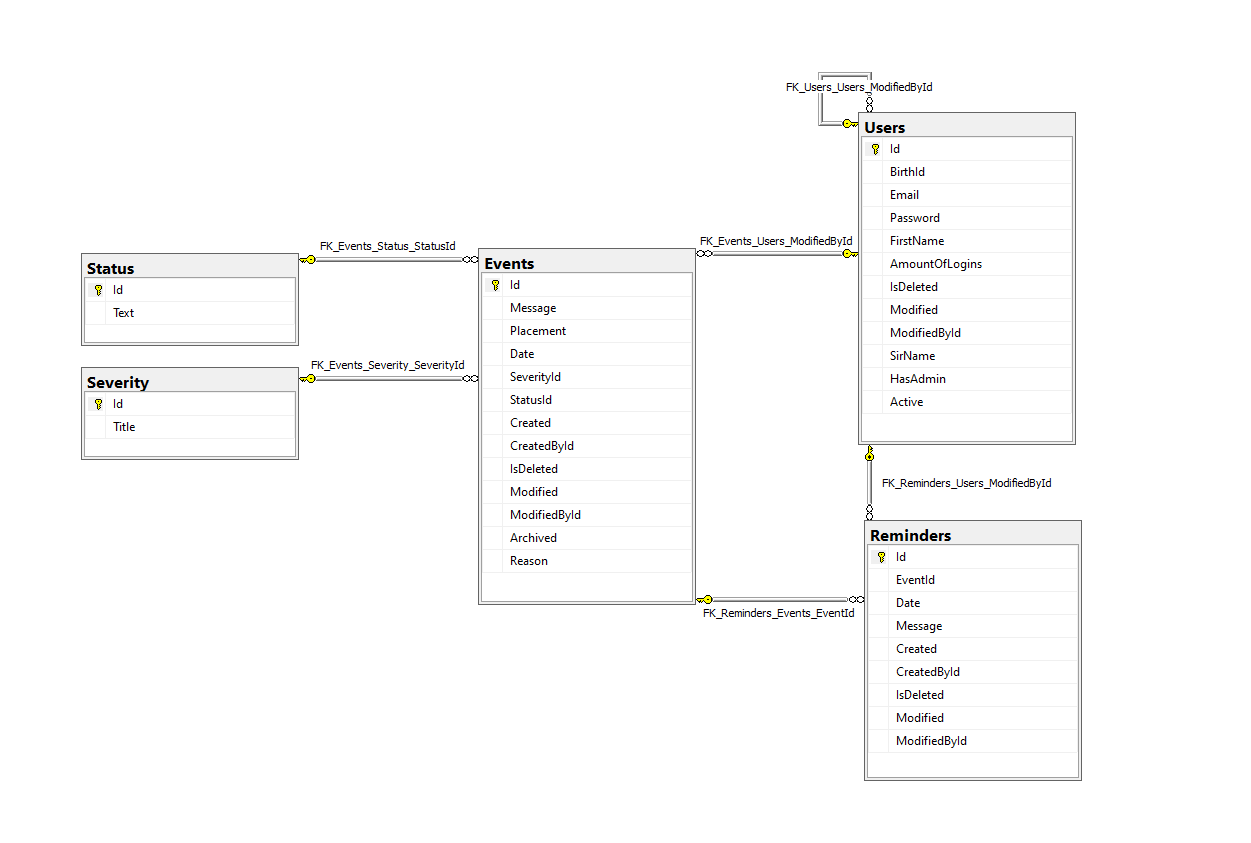
CORS på serveren er satt opp til å godta alle forespørsler, Dette er gjort for å enklere sette det opp. I en reell sitasjon burde nok disse være satt opp slik at serveren kun responderte på ønskede http-metodene som er satt opp ellers i løsningen.

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.

Tilkoblingene mot databasen er satt opp med ConnectionStrings som er spesifisert i konfigurasjonsfilen til API-et. Her kan man også legge på brukernavn og passord på selve databasen slik at man kan sette opp sikrere tilkobling mot databasen. ConnectionString til SQL databasen ligger ved slik at man kan benytte seg av denne koden for å hente ut data registrert her fra tidligere løsning.

# Databasen

Databasen er satt opp slik som dette.

Her vil man kunne spore siste endringer om noe ikke skulle stemme. ModifiedById legges til på alle endringer. Status og Severity er begge hjelpetabeller for å kunne ha begrensninger på typer slik at ikke de blir fritekst. Det er også oppgitt et felt på hendelsene som beskriver om det er arkivert eller ikke for å kunne begrense antallet som vises til sluttbruker. Seed for Apache Cassandra opprettelse av databasen ligger [vedlagt](#_Server_seed).

# 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.

# Vedlegg

## Server seed

CREATE KEYSPACE restserver WITH replication = {'class': 'SimpleStrategy', 'replication\_factor': '1'} AND durable\_writes = true;

CREATE TABLE restserver.status (

id int PRIMARY KEY,

text text

) WITH bloom\_filter\_fp\_chance = 0.01

AND caching = {'keys': 'ALL', 'rows\_per\_partition': 'NONE'}

AND comment = ''

AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max\_threshold': '32', 'min\_threshold': '4'}

AND compression = {'chunk\_length\_in\_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}

AND crc\_check\_chance = 1.0

AND dclocal\_read\_repair\_chance = 0.1

AND default\_time\_to\_live = 0

AND gc\_grace\_seconds = 864000

AND max\_index\_interval = 2048

AND memtable\_flush\_period\_in\_ms = 0

AND min\_index\_interval = 128

AND read\_repair\_chance = 0.0

AND speculative\_retry = '99PERCENTILE';

CREATE TABLE restserver.reminder (

id int PRIMARY KEY,

created timestamp,

createdbyid int,

date timestamp,

eventid int,

isdeleted boolean,

message text,

modified timestamp,

modifiedbyid int

) WITH bloom\_filter\_fp\_chance = 0.01

AND caching = {'keys': 'ALL', 'rows\_per\_partition': 'NONE'}

AND comment = ''

AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max\_threshold': '32', 'min\_threshold': '4'}

AND compression = {'chunk\_length\_in\_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}

AND crc\_check\_chance = 1.0

AND dclocal\_read\_repair\_chance = 0.1

AND default\_time\_to\_live = 0

AND gc\_grace\_seconds = 864000

AND max\_index\_interval = 2048

AND memtable\_flush\_period\_in\_ms = 0

AND min\_index\_interval = 128

AND read\_repair\_chance = 0.0

AND speculative\_retry = '99PERCENTILE';

CREATE TABLE restserver.badevent (

id int PRIMARY KEY,

archived boolean,

created timestamp,

createdbyid int,

date timestamp,

isdeleted boolean,

message text,

modified timestamp,

modifiedbyid int,

placement text,

reason text,

severityid int,

statusid int

) WITH bloom\_filter\_fp\_chance = 0.01

AND caching = {'keys': 'ALL', 'rows\_per\_partition': 'NONE'}

AND comment = ''

AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max\_threshold': '32', 'min\_threshold': '4'}

AND compression = {'chunk\_length\_in\_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}

AND crc\_check\_chance = 1.0

AND dclocal\_read\_repair\_chance = 0.1

AND default\_time\_to\_live = 0

AND gc\_grace\_seconds = 864000

AND max\_index\_interval = 2048

AND memtable\_flush\_period\_in\_ms = 0

AND min\_index\_interval = 128

AND read\_repair\_chance = 0.0

AND speculative\_retry = '99PERCENTILE';

CREATE TABLE restserver.severity (

id int PRIMARY KEY,

title text

) WITH bloom\_filter\_fp\_chance = 0.01

AND caching = {'keys': 'ALL', 'rows\_per\_partition': 'NONE'}

AND comment = ''

AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max\_threshold': '32', 'min\_threshold': '4'}

AND compression = {'chunk\_length\_in\_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}

AND crc\_check\_chance = 1.0

AND dclocal\_read\_repair\_chance = 0.1

AND default\_time\_to\_live = 0

AND gc\_grace\_seconds = 864000

AND max\_index\_interval = 2048

AND memtable\_flush\_period\_in\_ms = 0

AND min\_index\_interval = 128

AND read\_repair\_chance = 0.0

AND speculative\_retry = '99PERCENTILE';

CREATE TABLE restserver.user (

id int PRIMARY KEY,

active boolean,

amountoflogins int,

birthid text,

email text,

firstname text,

hasadmin boolean,

isdeleted boolean,

modified timestamp,

modifiedbyid int,

password text,

sirname text

) WITH bloom\_filter\_fp\_chance = 0.01

AND caching = {'keys': 'ALL', 'rows\_per\_partition': 'NONE'}

AND comment = ''

AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max\_threshold': '32', 'min\_threshold': '4'}

AND compression = {'chunk\_length\_in\_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}

AND crc\_check\_chance = 1.0

AND dclocal\_read\_repair\_chance = 0.1

AND default\_time\_to\_live = 0

AND gc\_grace\_seconds = 864000

AND max\_index\_interval = 2048

AND memtable\_flush\_period\_in\_ms = 0

AND min\_index\_interval = 128

AND read\_repair\_chance = 0.0

AND speculative\_retry = '99PERCENTILE';

insert into status (id, text) values (1, 'Planning');

insert into status (id, text) values (2, 'Executing');

insert into status (id, text) values (3, 'Finished');

insert into severity (id, title) values (1, 'None');

insert into severity (id, title) values (2, 'Some');

insert into severity (id, title) values (3, 'Take action');

insert into severity (id, title) values (4, 'Medium minus');

insert into severity (id, title) values (5, 'Medium');

insert into severity (id, title) values (6, 'Medium pluss');

insert into severity (id, title) values (7, 'Take immediate action');

insert into severity (id, title) values (8, 'High minus');

insert into severity (id, title) values (9, 'High');

insert into severity (id, title) values (10, 'Very high');