Jenna Viljakainen

SUPABASEN TIETOKANTAOMINAI-SUUKSIEN JA KÄYTTÄJÄHALLINNAN YHDISTÄMINEN WEBSOVELLUKSEEN

Opinnäytetyö

Liiketalouden ammattikorkeakoulututkinto

Tietojenkäsittelyn koulutus

2024

Kaakkois-Suomen ammattikorkeakoulu

Tutkintonimike	Tradenomi (AMK)
Tekijä	Jenna Viljakainen
Työn nimi	Supabasen tietokantaominaisuuksien ja käyttäjähallinnan yhdistäminen websovellukseen
Toimeksiantaja	Kotka Svenska Barnträdgård
Vuosi	2024
Sivut	41 sivua, liitteitä 1 sivu
Työn ohjaaja	Miia Liukkonen

TIIVISTELMÄ

Tämän opinnäytetyön tavoitteena oli selvittää, mitä BaaS eli palvelinpuoli palveluna, tarkoittaa ja kuinka sellainen otetaan käyttöön websovelluksessa. BaaS-palveluksi valittiin vapaan lähdekoodin Supabase, jonka tietokanta- ja käyttäjähallintaominaisuuksia tutkittiin tarkemmin. Supabase integroitiin Next.js-websovellukseen ja jokainen vaihe käytiin läpi yksityiskohtaisesti.

Työn toimeksiantajana toimi Kotka Svenska Barnträdgård (Kotkan ruotsinkielinen päiväkoti), jonka henkilökunnalle toteutettiin ensimmäinen versio työvuorosovelluksesta, jossa työntekijät pääsevät tarkastelemaan, muokkaamaan ja suunnittelemaan työvuoroja. Päiväkodin henkilökunnalla ei ollut käytössä työvuoroihin liittyvää sovellusta, joten tällä työllä esitettiin, millainen se voisi olla. Sovelluksen avulla käytiin läpi Supabasen ominaisuuksia ja kuinka niitä yhdistetään sovellukseen.

Työn teoriaosassa käytiin läpi websovelluskehitystä yleisesti ja perehdyttiin palvelinpuoleen ja sen kehitykseen tarkemmin. Lisäksi käytiin läpi, mikä BaaS on ja vertailtiin, miten sovelluksen kehitys eroaa, kun käytetään BaaS-palvelua verrattuna perinteiseen tapaan toteuttaa sovelluksen palvelinpuoli. Teoriaosassa käytiin myös läpi Supabasea ja keskityttiin etenkin sen tietokantaominaisuuksiin ja käyttäjähallintaan

Sovelluksen avulla esiteltiin Supabasen käyttöönotto ja sen integrointi websovellukseen vaihe vaiheelta. Sovelluksen avulla käytiin läpi käyttäjähallinnasta käyttäjien rekisteröiminen sekä sisään- ja uloskirjaus. Tietokantaominaisuuksien osalta käytiin läpi taulun luominen sekä erilaisten tietokantaoperaatioiden tekeminen ja näiden reaaliaikainen seuraaminen.

Työn tuloksena syntyi ensimmäinen versio työvuorosovelluksesta, jossa on hyödynnetty Supabasen tietokantaominaisuuksia ja käyttäjähallintaa. Työtä voi hyödyntää Supabasen integroinnissa omaan sovellukseen. Toteutuksen aikana tuli selväksi, että BaaS-alustan käyttö on helppoa ja sen avulla sovelluksen saaminen valmiiksi on nopeaa ilman syvempää osaamista palvelinpuolen kehityksestä.

Asiasanat: ohjelmistokehitys, Supabase, tietokantaohjelmat, todentaminen

South-Eastern Finland University of Applied Sciences

Degree title	Bachelor of Business Administration
Author	Jenna Viljakainen
Thesis title	Integrating Supabase database features and user management into a web application
Commissioned by	Kotka Svenska Barnträdgård
Time	2023
Pages	41 pages, 1 page of appendices
Supervisor	Miia Liukkonen

ABSTRACT

The objective of this thesis was to find out what BaaS (Backend as a Service) meant and how it could be implemented in a web application. The open-source platform Supabase served as a BaaS platform, with focus on its database and user management features. Supabase was integrated into the Next.js web application and each step of the process was examined in detail.

The commissioner of the thesis was Kotka Svenska Barnträdgård (Kotka Swedish Daycare). As part of the thesis, a first version of a web app was created to allow the staff of the daycare to view, edit, and plan work shifts. Because the daycare did not have any application for managing shifts, this project presented one solution for it. With the help of the application, the features of Supabase were examined and how they could be integrated into the application.

The theorical part of the thesis covered web development in general, with a focus on backend development in more detail. Additionally, it explored what BaaS was and compared how application development differed when using BaaS versus traditional backend development. The theorical part also examined Supabase with a focus on its database and user management features.

With the help of the application development process, the deployment of Supabase and its integration to a web application were presented step by step. The application examined user management features, including user registration, login, and logout. Regarding database features, it covered the creation of database tables, various database operations and real-time tracking of these events.

The result of this thesis was the first version of a shift management application where database and user management features of Supabase were used. The thesis can be used for integrating Supabase into web applications. During the development of the application, it became clear that using BaaS was easy and it accelerated the completion of an application without deeper knowledge of backend development.

Keywords: software development, Supabase, database programs, authentication

SISÄLLYS

1 JOHDANTO

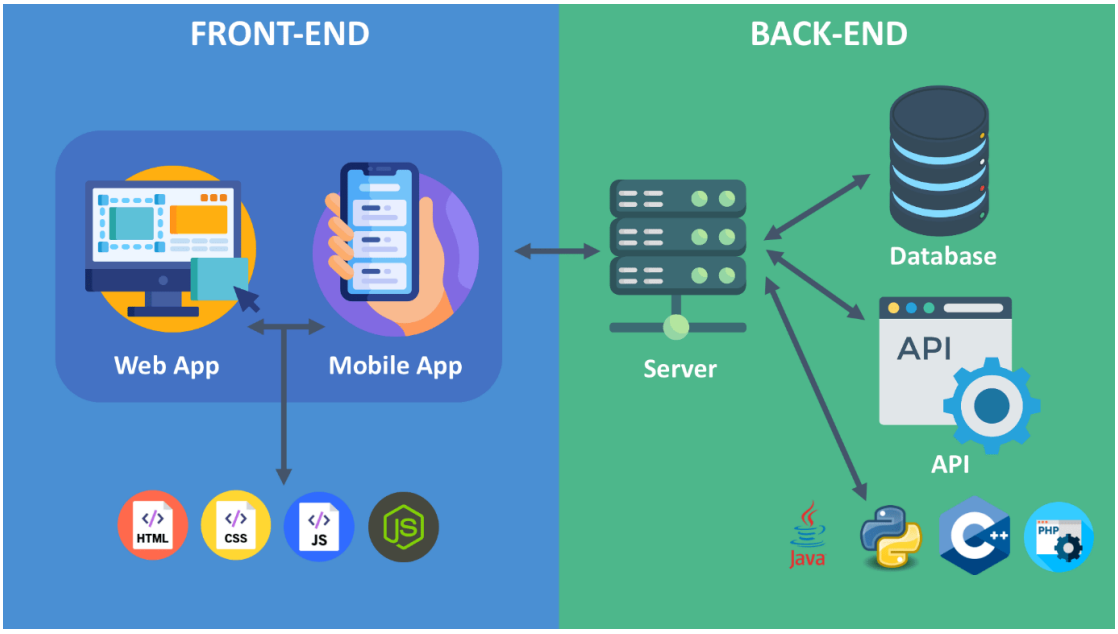
Palvelinpuoli eli backend, on olennainen osa websovelluksia ja se mahdollistaa selainpuolen eli frontendin sujuvan toiminnan, tietojen hallinnan ja tiedonsiirron. Palvelinpuolen kehittäminen vaatii paljon resursseja, minkä vuoksi on kehitetty BaaS-alustoja (Backend as a service), joissa palvelinpuoli otetaan käyttöön palveluna. Ne mahdollistavat keskittymisen selainpuolen kehitykseen nopeuttaen websovellusten kehitysprosessia markkinoille.

Opinnäytetyön tavoitteena on selvittää, mitä BaaS-palvelu tarkoittaa ja kuinka sellainen otetaan käyttöön websovelluksessa. Sovelluksen BaaS-palveluksi on valittu vuonna 2000 ilmestynyt avoimen lähdekoodin BaaS-palveluntarjoaja Supabase, jota käydään tarkemmin läpi teoriaosuudessa. Erityisesti keskitytään Supabasen tietokantaominaisuuksiin ja käyttäjähallintaan, jotka ovat keskeisimmät osat websovelluksen rakentamisessa. Valitsin BaaS-palveluksi nimenomaan Supabasen, sillä se on vielä melko uusi palvelu, eikä sitä ole vielä käsitelty paljoakaan.

Työn toimeksiantajana toimii Kotka Svenska Barnträdgård (Kotkan ruotsinkielinen päiväkoti), jonka tarpeisiin kehitetään websovellus työvuorojen lisäämiseen, katseluun ja muokkaamiseen. Tarkoituksena on luoda ensimmäinen versio sovelluksesta, jonka voisi tulevaisuudessa ottaa käyttöön päiväkodin henkilökunnalle. Opinnäytetyössä käydään läpi, kuinka Supabase otetaan websovelluksessa käyttöön ja kuinka sen ominaisuudet integroidaan kehitettävään sovellukseen. Opinnäytetyössä keskitytään ainoastaan Supabasen ominaisuuksiin, eikä tarkastella sovelluksen kehitystä muilta osin.

2 WEBKEHITYS

Webkehitys pitää sisällään web-suunnittelua ja -julkaisua, ohjelmointia sekä tietokannan hallintaa. Sillä tarkoitetaan sellaisten sovellusten luomista ja ylläpitoa, jotka toimivat internetin välityksellä käyttäjän selaimessa. Tyypillisesti webkehitys jaetaan kahteen pääalueeseen: selain- ja palvelinpuoleen (kuva 1). (Web Development 2023.)



Kuva 1. Webkehityksen jako selain- ja palvelinpuoleen (Basílio 2022)

Kuten kuvassa 1 on esitetty, on molemmilla omat tehtävänsä ja ohjelmointikielensä, joita käydään seuraavaksi läpi tarkemmin.

2.1 Selainpuoli

Selainpuoli eli frontend tarkoittaa websovelluksessa kaikkea sitä, minkä käyttäjä näkee ja jonka avulla käyttäjä on vuorovaikutuksessa sovelluksen kanssa. Sitä kutsutaan usein myös asiakaspuoleksi. Selainpuolen kehityksessä on käytössä useita teknologioita, joiden avulla käyttöliittymästä saadaan tehtyä mahdollisimman käyttäjäystävällinen ja ulkoasultaan miellyttävä mahdollisimman helposti. Perusohjelmointikieliä ovat HTML, CSS ja JavaScript, jotka pyörivät käyttäjän selaimessa luoden käyttöliittymän. (Web Development 2023.) HTML on merkintäkieli, jonka avulla määritellään verkkosivujen rakenne ja asettelu. CSS on tyylitiedosto-kieli, jonka avulla voidaan määritellä verkkosivun ulkoasua, kuten värejä, fontteja ja asetteluja. JavaScript taas on ohjelmointikieli, jonka avulla verkkosivuille voidaan lisätä interaktiivisuutta, animaatioita ja dynaamisia elementtejä. (Wilkins 2021.) Näiden avulla pystytään luomaan toimiva käyttöliittymä sovellukselle.

Selainpuolen kehitykseen on olemassa paljon erilaisia sovelluskehyksiä ja kirjastoja, joiden avulla kehitys on helpompaa. Näitä ovat esimerkiksi CSS-sovelluskehykset Tailwind CSS ja Bootstrap sekä JavaScript-kirjasto ReactJS ja JavaScript-sovelluskehys AngularJS. (Web Development 2023.) Next.js, jolla

esimerkkisovellus kehitetään, on React-pohjainen sovelluskehys, jonka avulla voi hyödyntää kaikkia React:n ominaisuuksia ja lisäksi ottaa käyttöön mm. Next.js:n automaattisen reitityksen sekä palvelinpuolen renderöinnin (Next.js 2023).

2.2 Palvelinpuoli

Palvelinpuoli eli backend, on websovelluksen osa, jota käyttäjät eivät näe tai minkä kanssa käyttäjät eivät ole suoraa vuorovaikutuksessa (Lemonaki 2022). Sen ohjelmointikieliin kuuluvat mm. Java, Python ja JavaScript (Node.js) ja myös niille on sovelluskehyksiä, jotka helpottavat ohjelmointia, kuten Spring, Django ja Express.js (Web Development 2023). Palvelinpuolen koodi suoritetaan palvelimella ja se keskustelee selainpuolen kanssa tehden dynaamisia ja interaktiivisia käyttäjäkokemuksia. Siinä keskeistä on datan käsittely, tallennus ja haku tietokannoista sekä käyttäjien syötteiden prosessointi. Tämän lisäksi palvelinpuolen kehittäjät vastaavat sovelluksen turvallisuudesta. (Le Thanh 2023.) Kun käytössä ei ole omaa palvelinta, voidaan sovelluksen isännöinti toteuttaa myös pilvipohjaisella palveluntarjoajan palvelimella, mikä vähentää tarvetta syvälliselle palvelinosaamiselle (Jones 2022).

2.2.1 Palvelin

Web-palvelin voi olla laitteisto, ohjelmisto tai näiden yhdistelmä. Palvelintietokoneessa säilytetään verkkosivuston tiedostoja ja palvelinohjelmistoa, joka käsittelee käyttäjien pyyntöjä ja tarjoaa pääsyn näihin tiedostoihin. Palvelinohjelmisto toimii vähintään http-palvelimena, joka ymmärtää verkko-osoitteita (url) ja käyttää http-protokollaa, jota selaimet käyttävät verkkosivujen näyttämiseen, tiedostojen toimittamiseen. Käyttäjät pääsevät http-palvelimelle verkkosivun verkkotunnuksen kautta, ja palvelin toimittaa pyydetyn verkkosivun sisällön käyttäjän laitteelle. (What is a web server? s.a.)

Yksinkertaisimmillaan, kun selain tarvitsee palvelimen tiedostoja, se lähettää http-pyynnön palvelintietokoneelle, jossa palvelinohjelmisto hyväksyy pyynnön, etsii tarvittavan tiedoston ja lähettää vastauksen takaisin selaimelle http:n välityksellä. Tällaista kutsutaan staattiseksi web-palvelimeksi. Dynaaminen web-palvelin taas koostuu staattisesta palvelintietokoneesta ja -ohjelmistosta,

mutta niiden lisäksi se sisältää myös ylimääräisen palvelinsovelluksen ja tietokannan. Nykyaikaiset websivut käyttävät niin sanottuja dynaamisia web-palvelimia, sillä ne koostuvat tyypillisesti muutamasta erilaisesta HTML-mallista ja tietokannasta. (What is a web server? s.a.)

2.2.2 Tietokanta

Tietokanta on organisoitu tietokokoelma, joka on tallennettu tietokonejärjestelmään ja jota hallitaan yleensä tietokannan hallintajärjestelmällä. Yleensä tietokannoissa oleva tieto on mallinnettu taulukoihin, jotka koostuvat riveistä ja sarakkeista. Tämä mahdollistaa kyselyiden ja datan käsittelyn tehokkaan suorittamisen. (What is Database? 2023.) Relaatiotietokanta on yksi tällainen tietokantatyyppi. Se käyttää rakennetta, jonka avulla voi tunnistaa ja hakea tietoa suhteessa toiseen tietokannan tietoon. Tietokannan taulukoille on ennalta määritelty sarakkeet sekä tietotyypit, jotka yhdessä koostavat taulukon tietokantakaavion (schema). Koostettua kyselykieltä, eli SQL-kieltä, käytetään usein kommunikoimaan relaatiotietokannan hallintajärjestelmään tallennetun tiedon kanssa. (Codecademy s.a.) Suosittuja tietokannan hallintajärjestelmiä ovat mm. MySQL, PostgreSQL ja SQLite, joista tarkemmin tutustutaan PostgreSQL-hallintajärjestelmään, jota Supabase hyödyntää palveluissaan.

PostgreSQL on edistyksellinen vapaan lähdekoodin relaatiotietokannan hallintajärjestelmä, joka käyttää SQL-kieltä yhdessä muiden ominaisuuksien kanssa. Sillä on vahva maine sen arkkitehtuurin, luotettavuuden ja tiedon eheyden ansiosta ja se on yksi suosituimmista vaihtoehdoista relaatiotietokannan hallintajärjestelmien joukossa. (PostgreSQL s.a.) PostgreSQL tukee sekä SQL-, että JSON-kyselyitä sekä suurinta osaa ohjelmointikielistä, kuten esimerkiksi Python, Java ja Node.js (PostgreSQL Introduction s.a.). PostgreSQL on helposti laajennettavissa ja sen avulla voi luoda esimerkiksi omia tietotyyppejä, rakentaa omia funktioita tai kirjoittaa koodia toisella kielellä kääntämättä tietokantaa uudelleen. (PostgreSQL s.a.) Supabase hyödyntää automaattisesti PostgreSQL:ää ja lisäksi käyttäjä voi halutessaan ottaa käyttöön lisää PostgreSQL-ominaisuuksia. Supabasen käyttö ei kuitenkaan vaadi mitään kokemusta PostgreSQL:stä.

2.2.3 Ohjelmointirajapinta

Ohjelmointirajapinta eli API, on joukko sääntöjä, jotka määrittelevät kuinka sovellukset tai laitteet muodostavat yhteyden ja keskustelevat keskenään. Se mahdollistaa viestinnän ja vuorovaikutuksen eri ohjelmistojärjestelmien välillä, ja sen avulla voi helposti yhdistää kolmansien osapuolien palveluita sovellukseen. (Le Thanh 2023.) Etänä toimivat ohjelmointirajapinnat on suunniteltu toimimaan tietoverkon kautta, eli ohjelmointirajapintojen käsittelemät resurssit ovat pyynnön tekevän tietokoneen ulkopuolella. Koska eniten käytetty tietoverkko on internet, on suurin osa ohjelmointirajapinnoista suunniteltu webstandardien mukaan. Ne käyttävät tyypillisesti http-pyyntöjä ja määrittelevät vastausviestien rakenteen usein joko XML- tai JSON-muodossa, sillä ne ovat helppoja käsitellä. (Red Hat 2022.)

Ohjelmointirajapintojen leviämisen myötä on kehitetty protokolla tiedonvaihdon standardoimiseksi, jota kutsutaan termillä SOAP (simple object access protocol), joka käyttää XML-muotoa http-pyyntö- ja vastausviesteissä. Suosituin ohjelmointirajapinnan määrittely on REST (representational state transfer), joka ei ole protokolla kuten SOAP, vaan arkkitehtuurityyli, joka on nykyään laajemmin käytössä, sillä se on helpompi käyttää kuin SOAP. (Red Hat 2022.) REST API käyttää http-pyyntömetodeja, kuten get, put, delete ja post, päästäkseen käsiksi tietoon ja käyttää XML- tai JSON-formaattia tiedon esittämiseen (Gillis 2020).

2.2.4 Käyttäjähallinta

Yksi sovelluksen tärkein puolustuskeino on toimiva autentikointi ja auktorisointi (Munene 2023). Sovellukset sisältävät paljon henkilökohtaista ja arkaluontoista tietoa käyttäjistä, jotka täytyy pyrkiä pitämään turvassa ulkopuolisilta. Autentikoinnilla tarkoitetaan prosessia, jossa varmistetaan käyttäjän henkilöllisyys ennen kuin hänelle myönnetään pääsy tietojärjestelmiin. Kirjautumiseen on erilaisia tapoja, kuten salasanapohjainen, biometrinen ja monivaiheinen autentikointi. (Pant ym. 2023, 782, 784.) Lisäksi rest-ohjelmointirajapinnassa täytyy tarkastaa käyttäjän henkilöllisyys aina kun käyttäjä tekee pyyntöjä ja tähän tarkoitukseen on erilaisia autentikointimenetelmiä kuten token- ja

evästepohjainen autentikointi, kolmannen osapuolen tarjoamat autentikointitavat kuten OAuth, OpenID sekä SAML (Sandip 2023).

Kolmannen osapuolen autentikointi, kuten OAuth, mahdollistaa käyttäjien kirjautumisen sovellukseen käyttämällä olemassa olevia tunnuksiaan toisilta palveluntarjoajilta kuten Facebookilta tai Googlelta, ilman tarvetta luoda uusia tunnuksia. Palveluntarjoaja myöntää tokenin sovellukselle, jonka avulla voi saada lisätietoja käyttäjästä. OpenID hyödyntää kolmannen osapuolen kirjautumista ilman salasanojen tai muiden tietojen vaihtoa. SAML (Security assertion markup language) toimii kuten OpenID, mutta on XML-pohjainen ja käyttää identiteetin tarjoajan URL-osoitetta käyttäjän kirjautumiseen ja ohjaamaan XML-vastauksen kautta takaisin sovellukseen, jossa tiedot puretaan. (Sanders 2023; Sandip 2023.)

Evästepohjaisessa autentikoinnissa palvelin tarkastaa käyttäjän kirjautumistiedot, luo tunnisteen istunnolle ja tallettaa sen palvelimelle sekä selainpuolelle evästeisiin (Sandip 2023). Evästeet ovat pieniä tiedostoja, joita palvelin luo ja lähettää selaimelle, ja selaimet säilövät niitä ennalta määritellyn ajan tai käyttäjän istunnon ajan (What are cookies… s.a.). Seuraavia pyyntöjä palvelimelle tehtäessä evästeisiin tallennettua tunnistetta verrataan palvelimelle tallennettuun tunnisteeseen ja mikäli ne täsmäävät, hyväksytään pyyntö. Ulos kirjautuessa istunnon tunnisteet poistetaan sekä palvelimelta että evästeistä. (Sanders 2023; Sandip 2023.)

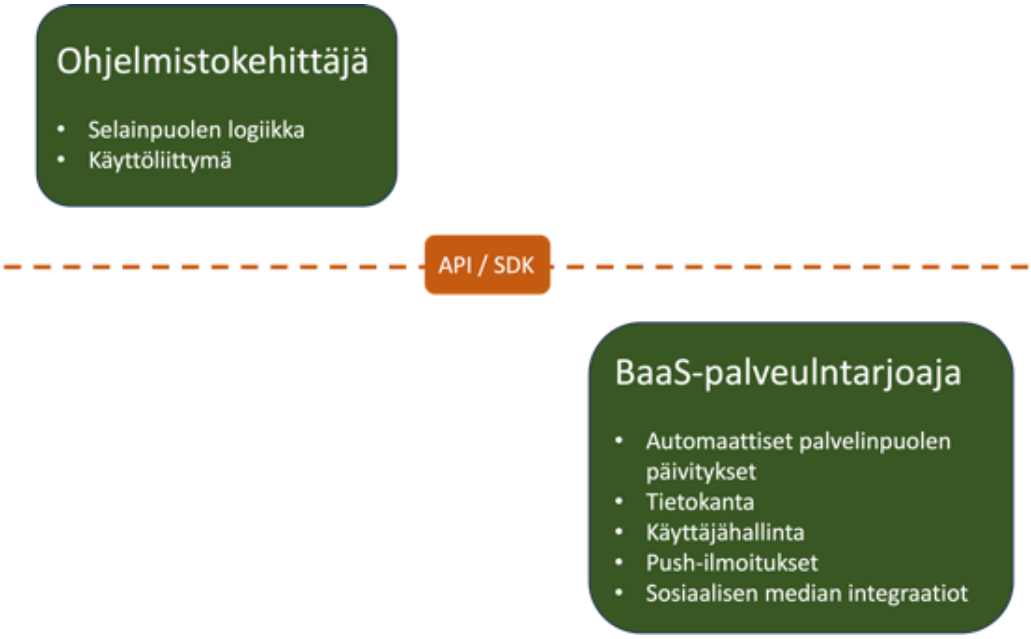
Tokenpohjaisessa autentikoinnissa käytetään hyväksi JSON Web Tokenia, eli JWT:tä, joka on yksi käytetyimmistä tavoista luoda token (Sandip 2023; Pant, Rajawat ym. 2022, 786). JWT:n avulla lähetetään tietoa JSON-objektina palvelimelta selaimeen. JWT on digitaalinen tieto, jota käytetään tietojen turvalliseen siirtoon ja se koostuu kolmesta osasta: otsikosta, joka määrittelee tokenin tyypin ja allekirjoitusmenetelmän, kuormasta, joka sisältää siirrettävät tiedot kuten käyttäjän tunnisteet sekä allekirjoituksesta, joka varmistaa tokenin aitouden. (Puton 2023.) JWT:n access token on lyhytikäinen, joten tarvitaan myös refresh tokeneita, joita käytetään luomaan uusia access tokeneita. Tämä mahdollistaa lyhytikäisten ja turvallisten access tokeneiden käytön samalla kun voidaan sallia käyttäjille pidempiaikainen autentikointi. (Kumar

2023.) Tokenpohjaisessa autentikoinnissa JWT säilötään ainoastaan selainpuolelle paikallisesti (Sandip 2023).

Auktorisointi määrittelee, mitä toimintoja käyttäjä voi suorittaa järjestelmässä autentikoinnin jälkeen, ja se perustuu tyypillisesti käyttäjän rooliin tai oikeuksiin sovelluksen sisällä. Sekä autentikointia että auktorisointia tarvitaan, jotta sovelluksen suojaus olisi riittävää. Yksi auktorisointimenetelmä on PostgreSQL:n rivitason suojaus, jonka avulla voidaan suojata tauluja kyselyiltä. Tätä varten täytyy luoda toimintatapoja, joiden mukaan annetaan pääsy käyttäjille tekemään tiettyjä kyselyitä tiettyihin tauluihin. Mikäli toimintatapoja ei ole luotu, mutta rivitason suojaus on otettu käyttöön, taulusta ei näytetä käyttäjille mitään. Toimintatavat taas eivät tule käyttöön ennen kuin rivitason suojaus on otettu käyttöön. (Row Security Policies. s.a.)

2.3 BaaS

Palvelinpuolen rakentaminen on aikaa vievää, joten etenkin uusien yritysten on taloudellisesti kannattavampaa ottaa palvelinpuoli käyttöön palveluna, kuin alkaa rakentamaan kaikkea itse alusta saakka. Ohjelmistokehittäjät voivat keskittyä enemmän selainpuolen kehitykseen, sillä BaaS-palvelun avulla koko palvelinpuolen infrastruktuuri on jo valmiina sovellukselle. (Adlakha 2023.) Kuvassa 2 on esiteltynä kehittäjän ja BaaS-palveluntarjoajien mahdollista työnjakoa. BaaS on yksi ratkaisu, kun halutaan sovellus nopeasti markkinoille ja saada käyttäjien palautetta mahdollisimman pian tai kun tiedossa ei ole huomattavaa laajennusta tulossa (Bartosz & Aleksandra 2019).



Kuva 2. Kehittäjän ja BaaS-palveluntarjoajien työnjako

BaaS on pilvipalvelun alue ja viittaa isännöityyn palvelinpuolen infrastruktuuriin, jonka avulla kehittäjät voivat perustaa nopeasti ja helposti koko palvelinpuolen (IONOS 2023). BaaS-palvelut sisältävät ohjelmointirajapintoja ja ohjelmistokehityspaketteja (SDK, software development kit), joiden avulla kehittäjät voivat integroida palvelinpuolen omaan sovellukseensa (Adlakha 2023). BaaS-palvelut sisältävät useita erilaisia ominaisuuksia, jotka helpottavat kehitystyötä ja nopeuttavat sovelluksen kehitystä. BaaS:n tärkeimpiä ominaisuuksia ovat mm. skaalautuvat tietokannat, ohjelmointirajapinnat, käyttäjähallinta, sosiaalisen median integraatiot sekä push-ilmoitukset (Gadhavi 2023). BaaS-palveluntarjoajia on useita, joista suosituimpia ovat mm. Firebase ja Supabase.

Firebase on Googlen BaaS-alusta, joka auttaa kehittäjiä rakentamaan, hallitsemaan ja kasvattamaan sovelluksia helposti. Se mahdollistaa sovellusten nopeamman ja turvallisemman kehittämisen ilman ohjelmointia Firebasen puolella. Firebase on kehittänyt palveluita Androidille, iOS:lle, verkolle sekä Unity:lle ja se käyttää pilvitallennustilaa ja NoSQL-tietokantaa tietojen tallentamiseen. Firebase sisältää paljon erilaisia ominaisuuksia, jotka voi tarvittaessa ottaa käyttöön sovelluksessa kuten käyttäjähallinnan, kaatumisraportit ja sovelluksen isännöinnin. (Firebase - Introduction 2021.) Vaikka Firebase on suo-

situin BaaS-vaihtoehto useine ominaisuuksineen, valittiin työn BaaS-palveluksi Supabase, sillä se on verrattain uusi ja tästä syystä sitä ei ole vielä käsitelty yhtä laajasti kuin Firebasea.

2.4 BaaS vs. perinteinen palvelinpuolen kehitys

Perinteisessä palvelinpuolen kehityksessä kaikki tehdään alusta alkaen itse ja kaikkea täytyy myös ylläpitää itse. Tämä voi tietyssä tilanteessa olla hyvä asia, sillä silloin kehittäjällä on enemmän valinnanvaraa, millaisen sovelluksen tekee ja minkälaista toiminnallisuutta ottaa käyttöön. Kehittäjä pystyy itse valitsemaan sovelluksessa käytettävät teknologiat omien mieltymyksiensä mukaan, mikä ei ole välttämättä mahdollista BaaS-palveluissa. Käytännössä mahdollisuudet tällöin ovat rajattomat. Kehittäjillä on tällöin myös itse täysi kontrolli sovelluksen suorituskyvystä ja kuluista, joita sovellukseen menee. Mikäli esimerkiksi kävijämäärät sovelluksessa lisääntyvät merkittävästi, saattaa sovelluksen kulut nousta BaaS-palvelussa odottamattomasti. Toisaalta tämä lähestymistapa on enemmän aikaa vievää ja siitä syystä myös taloudellisesti haastavampaa, sillä se vaatii enemmän huolellista suunnittelua, toteutusta ja ylläpitoa. (Batschinski s.a.)

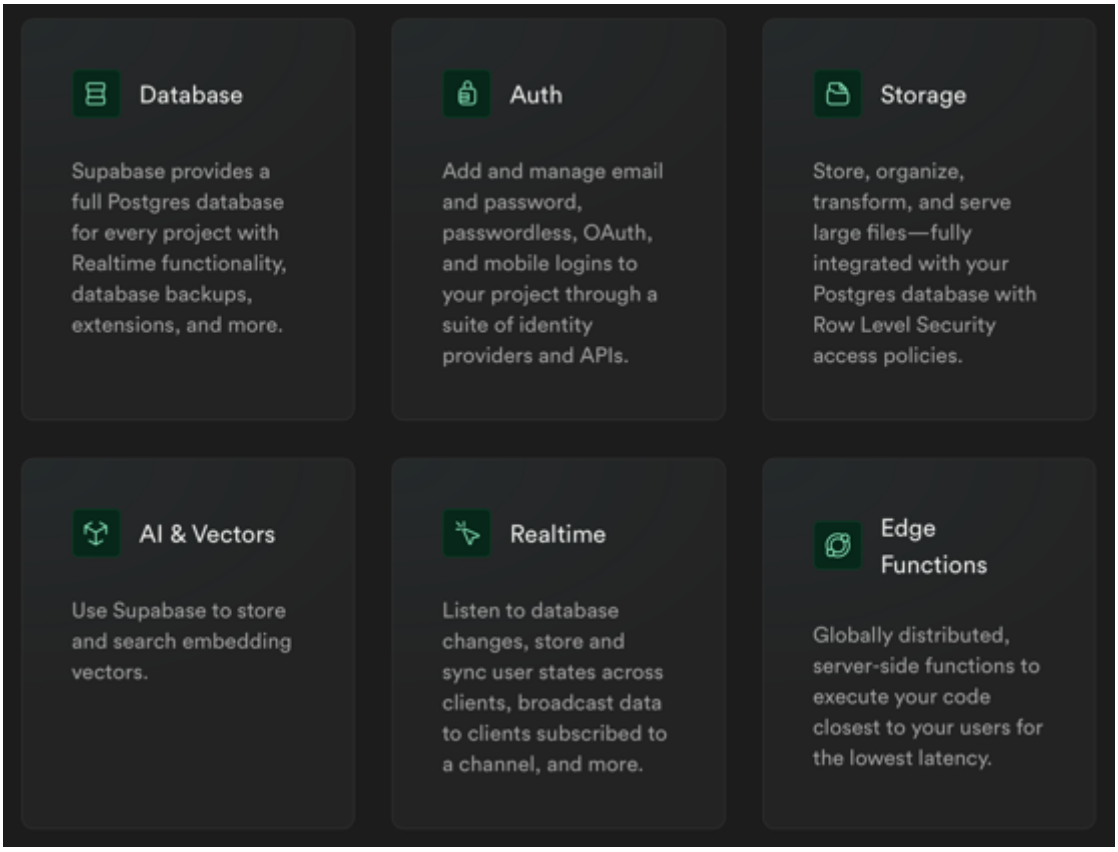
BaaS-palvelun valinta sopii etenkin silloin, kun halutaan sovellus nopeasti markkinoille. Se on halvempi tapa aloittaa sovelluksen kehitys, sillä se ei vaadi niin paljoa aikaa tai kehittäjän työpanosta kuin perinteinen tapa luoda palvelinpuoli. BaaS-palveluilla on myös runsaasti erilaisia ominaisuuksia, joita ottaa käyttöön sovellukseen, joten sovelluksen ominaisuuksista ei tarvitse välttämättä tinkiä lainkaan. Kuitenkin mikäli tarvitseekin jotain ominaisuutta, jota ei ole saatavilla BaaS-palvelussa, voi sellaisen kehittämiseen itse mennä enemmän aikaa, jolloin perinteinen tapa voi lopulta tulla halvemmaksi. Lisäksi mikäli sovelluksessa säilytetään arkaluontoista tietoa, voi perinteinen tapa luoda palvelinpuoli olla turvallisempi valinta. Toisaalta mikäli ei ole tarpeeksi tietoturvaosaamista, on tällöin BaaS parempi vaihtoehto. (Biskupski & Pytko-Włodarczyk 2019.)

Jokainen ohjelmistoprojekti on kuitenkin erilainen ja niillä on erilaiset vaatimukset ja tarpeet, jotka vaikuttavat siihen onko perinteinen tapa luoda palvelinpuoli vai BaaS-palvelun käyttö parempi vaihtoehto sovellukselle. Tästä

syystä kehittäjien on mietittävä tarkkaan mitä juuri kyseinen projekti vaatii, mitä resursseja on käytössä ja minkälainen aikataulu kehitystyölle on käytössä. Molemmille tavoille luoda sovellus on paikkansa, eikä ole olemassa selvää vastausta kumpi olisi parempi. (Biskupski & Pytko-Włodarczyk 2019.) Tähän työhön on valittu BaaS-palvelun käyttö, sillä sovellus täytyy toteuttaa nopeasti ilman rahoitusta yhden kehittäjän toimesta ja ilman syvempää osaamista tietoturvasta.

3 SUPABASE

Supabase on BaaS-palvelu, joka sisältää kehittäjille erilaisia työkaluja luomaan ja hallitsemaan palvelinpuolta (Aplyca 2023). Se perustettiin tammikuussa 2020 kilpailemaan Firebasen kanssa ja sitä markkinoidaankin Firebasen vapaan lähdekoodin vaihtoehtona (Musing 2023). Kuvassa 3 on esitelty Supabasen ominaisuuksia, joiden avulla voidaan luoda skaalautuva ja suojattu sovellus.



Kuva 3. Supabasen ominaisuudet (Supabase s.a.)

Kuten kuvassa 3 on esiteltynä, sisältää Supabase useita eri ominaisuuksia kuten tietokannan, käyttäjähallinnan ja reunatoimintoja, joita voi ottaa käyttöön

omien tarpeiden mukaan. Supabasea voi käyttää sen selaimesta löytyvän Supabase Dashboardin, eli hallintapaneelin kautta, josta löytyy kaikki asetukset ja muut tiedot, eikä sen käyttö vaadi ohjelmointia lainkaan. Supabasessa on erilaisia ohjelmistokehityspaketteja, joiden avulla sovelluksen liittäminen Supabasen toimintoihin on helppoa. Tällainen on Supabasen asiakaskirjasto, eli Client Library, jonka kautta pystytään olla yhteydessä PostgreSQL-tietokantaan ja muihin Supabasen toimintoihin. Asiakaskirjastoja on eri kielille, kuten C#, Python sekä JavaScript (Supabase Documentation s.a.). Asiakaskirjaston kautta voi mm. kuunnella tietokannan muutoksia, rakentaa käyttäjähallinnan sovellukselle sekä käsitellä suuria tiedostoja. Käytännössä siis kaikki Supabasen toiminnot voidaan toteuttaa asiakaskirjaston kautta. (JavaScript Client Library s.a.) Koska sovellus toteutetaan käyttäen Next.js-sovelluskehystä, Supabase suosittelee käytettäväksi tällaisiin palvelinpuolen sovelluskehyksiin ja kieliin lisänä @supabase/ssr-apupakettia (server side rendering), jossa käyttäjien istunnot tallennetaan evästeisiin. Näin ollen käyttäjän istunnon tiedot ovat käytössä sekä selain- että palvelinpuolella koko sovelluksessa. Käytännössä apupaketti toimii kuten asiakaskirjastot. (Setting up Server-Side… s.a.)

3.1 Käyttäjähallinta

Supabasen yksi tärkeimmistä ominaisuuksista on käyttäjähallinta, joka sisältää autentikoinnin ja auktorisoinnin. Sitä voi käyttää yksinään tai yhdessä muiden Supabasen ominaisuuksien kanssa ja myös tämä ominaisuus tulee PostgreSQL:stä, jota Supabase hyödyntää lähes kaikessa. Käyttäjän voi autentikoida usealla tavalla esimerkiksi käyttäen salasanapohjaista autentikointia, salasanatonta autentikointia tai kolmannen osapuolen tunnistautumisia. Supabase sisältää myös auktorisointiominaisuuksia kuten rivitason suojauksen, jonka avulla voidaan tarkastaa käyttäjän oikeuksia tietokantaan. Supabase hyödyntää token-pohjaista autentikointia autentikoidakseen käyttäjiä ja hallitsemaan pääsyä tietokantaoperaatioihin. (Auth s.a.)

Autentikointi tarkastaa onko käyttäjällä oikeus kirjautua sisään vai ei sekä kuka yrittää kirjautua. Yksinkertaisin tapa kirjautua sisään on käyttäen sähköpostia ja salasanaa. Toinen tapa autentikoida käyttäjä Supabasessa on käyttää salasanatonta kirjautumista, esimerkiksi kirjautumislinkki sähköpostiin tai

kertaluontoinen salasana sähköpostilla tai tekstiviestillä/WhatsApp:lla. Supabasessa on mahdollista myös käyttää kirjautumisessa kolmannen osapuolen sovellusten OAuth valtuutusta tai SAML:ää. (Auth s.a.)

Auktorisointi taas kattaa kaiken kirjautumisen jälkeen tarkastaen jokaiselta käyttäjältä, mitä heidän on lupa tehdä tietokannalle. PostgreSQL:n rivitason suojaus (RLS) on paras suojauskeino, mikäli tarvitsee yksityiskohtaisia valtuutussääntöjä ja Supabasen käyttäjähallinta onkin suunniteltu toimimaan täydellisesti rivitason suojauksen kanssa. Tämän avulla voidaan päättää, keillä käyttäjillä on oikeudet suorittaa select, insert, update ja delete-operaatioita tietylle taulun tai näkymän riville. Käytössä ovat mm. roolit anon ja authenticated, joiden avulla voi tarkastaa onko käyttäjä kirjautunut sisään. Rivitason suojaus on oletuksena päällä taulun luomisessa, mutta niille ei ole määriteltynä yhtään toimintatapaa, joten tauluun ei pääse käsiksi kukaan ennen niiden luontia. Mitään taulua ei tulisi luoda ilman rivitason suojausta, vaikka taulu olisikin saatavilla kaikille. (Auth s.a.)

3.2 Tietokanta

Jokainen Supabase-projekti sisältää vapaan lähdekoodin PostgreSQL-relaatiotietokannan (Database s.a.). Supabasen tietokantapalvelussa on kaikki PostgreSQL-ominaisuudet kuten taulukot, näkymät, funktiot ja rivitason suojaus (Yiming 2023). Supabasessa on useita tapoja päästä käsiksi tietokannan tietoihin, kuten ohjelmointirajapinnan, yhteyspoolin tai asiakaskirjaston avulla. Tässä opinnäytetyössä käytetään Supabasen JavaScript-asiakaskirjastoa yhdessä sen ssr-apupaketin kanssa, jonka avulla luodaan asiakas, joka yhdistää sovelluksen tietokantaan niin selain- kuin palvelinpuolella. (Connecting to… s.a.)

Uuden taulun lisäyksen yhteydessä luodaan sarakkeet, jotka täytyy tyypittää luontivaiheessa. Sarakkeita voi kuitenkin tarvittaessa poistaa, lisätä ja muokata myöhemmin. Sarakkeiden tyypityksessä on käytössä kaikki PostgreSQL:n oletus tyypit, kuten numerot, päivämäärät, merkkijonot ja totuusarvot. Näiden lisäksi on mahdollista luoda omia tyyppejä tai käyttää lisäosia tyypitykseen. Jokaisella taululla tulisi olla aina perusavain, jolla jokaisen taulun rivin voi tunnistaa. Tätä varten Supabasessa on aina oletuksena id-sarake, kun

uutta taulua luodaan. (Tables and Data s.a.) Kaikki taulut kuuluvat johonkin tietokantakaavioon, ja näitä käytetään yleensä turvallisuussyistä. Esimerkiksi Supabasessa uudet käyttäjät lisätään auth-tietokantakaavion users-tauluun, kun taas oletuksena uusi taulu lisätään public-tietokantakaavioon. (Managing tables… s.a.)

Supabasen ensimmäinen tarkoitus oli luoda reaaliaikainen PostgreSQL. Supabasessa onkin mahdollista laittaa päälle reaaliaikainen tietokannan seuranta, jonka ansiosta autentikoitujen käyttäjien on mahdollista seurata tietokannan muutoksia ilman viiveitä. Tämän voi ottaa käyttöön joko heti taulun luomisen yhteydessä tai myöhemmin taulun asetuksista. Kehittäjä voi päättää, mistä tauluista sekä mistä tietokannan tapahtumista lähetetään tietoja käyttäjille. (Realtime s.a.) Supabasen reaaliaikaiset toiminnot perustuvat 'Postgres Changes' -ominaisuuteen, joka on eräänlainen lisäosa tai toiminnallisuus Supabasessa. Tämä mahdollistaa tietokannan muutosten kuuntelun ja niiden lähettämisen reaaliajassa autentikoiduille käyttäjille. Tietokannan muutoksia lähetetään ainoastaan autentikoiduille käyttäjille ja tässäkin voidaan määritellä kenelle käyttäen rivitason suojausta. Lisäksi reaaliaikaominaisuuksien avulla on mahdollista lähettää nopeasti viestejä käyttäjien välillä, tarkkailla hiiren liikkeitä tai esittää käyttäjätila muille käyttäjille reaaliajassa. (Realtime Concepts s.a.)

Hallintapaneelin kautta on mahdollista käyttää SQL-editoria, jonka avulla voi tehdä SQL-kyselyitä tietokannan tauluihin. Sen avulla voi luoda myös funktioita ja laukaisimia sekä luoda tai poistaa tietokannan tauluja. Funktioiden ja laukaisimien kielenä käytetään PostgreSQL-tietokantaa varten luotua PL/pgSQL-kieltä, jonka avulla pystyy tehdä monimutkaisempia operaatioita kuin pelkällä SQL-kielellä (PL/pgSQL… s.a.). Editori mahdollistaa tietokannan hallinnan ja testaamisen suoraan hallintapaneelin kautta ilman, että tarvitsee tehdä muutoksia sovelluksen koodiin. Tämä tekee tietokannan kehittämisestä ja ylläpidosta helpompaa.

3.3 Supabasen muita ominaisuuksia

Supabasessa on myös reunatoimintoja (edge functions), jotka ovat käyttäjää lähellä toimivia palvelinpuolen TypeScript-funktioita. Niitä voidaan käyttää esimerkiksi webhookien kuunteluun tai integroimaan Supabase-projekti kolmannen osapuolen kanssa. (Edge Functions s.a.) Reunatoiminnot sopivat silloin kun vaaditaan vähäistä viivettä, mutta dataintensiivisiin operaatioihin suositellaan kuitenkin käytettäväksi tietokantafunktioita. Tämä lähestymistapa mahdollistaa optimaalisen suorituskyvyn ja joustavuuden sovelluksen eri osa-alueilla. (Edge Functions Quickstart s.a.)

Supabasen tietokantaan on mahdollista tallentaa myös esimerkiksi kuvia, videoita tai muita tiedostomuotoja. Supabasen avulla erilaisten tiedostojen lataaminen ja lähettäminen on tehty helpoksi ja siinä on lisäksi sisäänrakennettu kuvaoptimointiohjelma, jonka avulla mediatiedostojen kokoa on helppo muuttaa lennosta. Myös tiedostojen tallennuksessa käytetään hyväksi PostgreSQL:n rivitason suojausta. (Storage s.a.)

Supabasen avulla voi lisäksi kehittää tekoälysovelluksia käyttäen PostgreSQL-tietokantaa ja sen pgvector-lisäosaa. Tämä yhdistelmä mahdollistaa vektoriupotusten tehokkaan tallentamisen, indeksoinnin ja kyselyn, mikä tukee suurten datamäärien käsittelyä. Supabasen asiakaskirjastot tarjoavat helpon tavan integroida nämä toiminnot sovellukseen. Lisäksi Supabase tukee integraatioita suosittujen tekoälypalvelujen, kuten OpenAI:n ja Hugging Facen kanssa. (AI & Vectors s.a.)

4 TYÖAIKASOVELLUKSEN TOTEUTUS

Tavoitteena on luoda päiväkodin työntekijöille sovellus, jonka avulla he voivat nähdä tulevia työvuorojaan sekä muokata niitä tarpeen mukaan. Toiveena oli, että sovelluksessa pystyisi tarkastella tulevia työvuoroja, jossa näkyy kaikkien työntekijöiden vuorot siten, että työparien vuorot olisivat aina vierekkäin. Lisäksi päiväkodin johtajalla täytyisi olla mahdollisuus luoda tulevia työvuoroja helposti ja jokainen työntekijä voisi tarkastella kuukauden kokonaistyöaikaa, josta näkee työntekijän mahdolliset ylityötunnit. Työntekijöillä tulisi olla myös

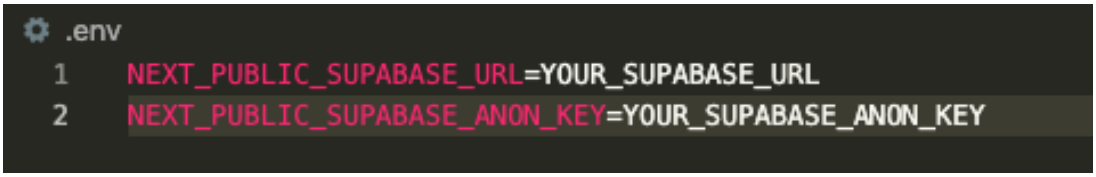
mahdollisuus muokata vuorojaan, mikäli työaika ei vastaa suunniteltua tai he vaihtavat vuoroja keskenään.

Työ toteutetaan käyttäen Next.js-sovelluskehystä ja TypeScript-kieltä. Supabasen osalta otetaan käyttöön tietokantaominaisuudet ja käyttäjähallinta, jotta saadaan toimeksiantajan vaatimusten mukainen toimiva sovellus. Käyttöön otetaan Supabasen JavaScript asiakaskirjasto ja ssr-apupaketti, joiden avulla saadaan tietokantaominaisuudet ja käyttäjähallinta käyttöön sovelluksessa. Työssä ei käsitellä sovelluksen kehitystä kuin Supabasen käytön kannalta.

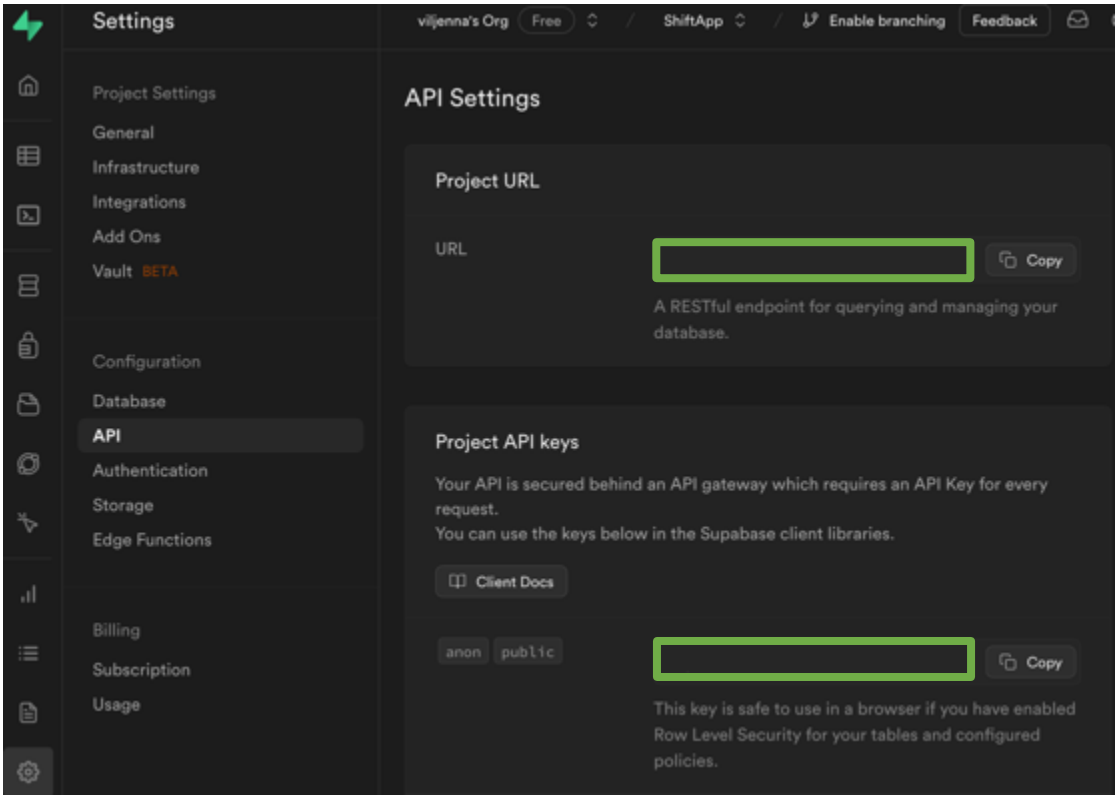
4.1 Supabase-projektin luominen

Supabasen käyttö on helppoa selaimessa toimivan hallintapaneelin avulla. Sitä kautta asetusten muokkaaminen, tietokantojen hallinta ja muut toiminnot on helppo tehdä. Ensiksi täytyy luoda tunnukset Supabase-palveluun. Tämä onnistuu suoraan Supabasen kotisivuilta www.supabase.com Start your project-napista. Tunnusten luomiseen voi käyttää kolmannen osapuolen tunnuksia kuten GitHub, jolloin erillisiä Supabase-tunnuksia ei tarvitse. Kun tunnukset on luotu ja kirjauduttu sisään, päästään hallintapaneelin etusivulle, jossa aloitetaan uusi projekti, minkä jälkeen valitaan haluttu organisaatio, jonka alle uusi projekti luodaan.

Seuraavaksi voi aloittaa Supabasen asennuksen terminaalissa. Next.js-sovelluskehyksen kanssa projektin luominen aloitetaan komennolla npx create-next-app -e with-supabase. Kun Supabase on asennettuna projektiin, täytyy ensimmäisenä lisätä .env-tiedostoon tiedot Supabasen url:stä sekä anon-key:stä merkkijonoina, joiden avulla sovellus pystytään yhdistämään juuri oikeaan Supabase-projektiin (kuva 4). Nämä tiedot löytyvät hallintapaneelista Settings/API (kuva 5).



Kuva 4. .env-tiedosto tarvittavilla Supabase-projektin tiedoilla



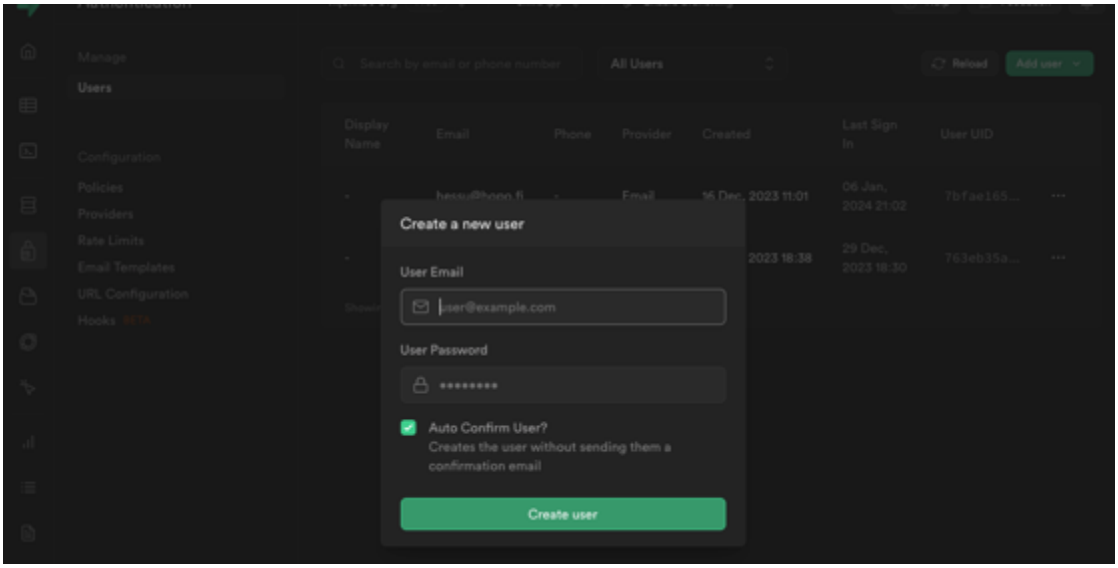
Kuva 5. Supabasen API -settings, josta löytyvät url ja anon key

Tässä vaiheessa voi edetä sovelluksen kanssa tekemään kirjautumis- ja aloitussivun, jotta voi ottaa käyttöön Supabasen käyttäjähallinnan.

4.2 Käyttäjähallinta

Tarkoituksena on tehdä sovellus päiväkodin henkilökunnalle, eli sovellus ei tule olemaan julkisesti kenen tahansa käytettävissä. Sisäänkirjautuminen toteutetaan käyttäen sähköpostia ja salasanaa, koska salasanaton autentikointi olisi hankala toteuttaa, sillä työntekijöillä on erikseen työ- ja kotisähköpostiosoitteet sekä puhelimet.

Testikäyttäjien lisääminen onnistuu helposti hallintapaneelin kautta kohdasta Authentication/Users, josta käyttäjiä voi lisätä ilman sähköpostitarkastusta (kuva 6). Mikäli lisätään oikeita käyttäjiä hallintapaneelin kautta, on järkevämpää käyttää sähköpostikutsua, jolloin käyttäjien täytyy vahvistaa osoite. Kun käyttäjä lisätään, tallennetaan se automaattisesti auth-tietokantakaavion users-tauluun.



Kuva 6. Uuden käyttäjän lisääminen Supabase-hallintapaneelissa

Evästeet ovat keskeisessä roolissa käyttäjien istuntojen hallinnassa palvelinpuolen sovelluksissa kuten Next.js ja tätä varten otetaan käyttöön Supabasen ssr-apupaketti. Paketin voi asentaa helposti JavaScript-asiakaskirjaston kanssa komennolla npm install @supabase/supabase-js @supabase/ssr, minkä jälkeen voidaan luoda Supabase-client, eli -asiakas. Next.js-sovelluksessa tarvitaan kaksi asiakasta: selainpuolen asiakas, jota käytetään selainpuolen komponenteissa, jotka suoritetaan käyttäjän selaimessa sekä palvelinpuolen asiakas, jota käytetään palvelinpuolen komponenteissa, palvelintoiminnoissa ja reittikäsittelijöissä, jotka suoritetaan palvelimella. Turvallisin tapa hallita käyttäjäistuntoja on käyttää palvelinpuolen asiakasta, jolloin tiedot tulevat palvelimelta eikä selaimesta, jossa saattaa olla vanhentunutta tietoa tallennettuna. (Setting up Server-Side… s.a.) Selainpuolen asiakasta ei tarvita käyttäjähallinnassa, mutta sitä käytetään myöhemmin tietokantaan yhdistettäessä. Koska käyttäjän tietoja tallennetaan evästeisiin, täytyy myös selainpuolen asiakas olla luotu käyttäen samaa ssr-apupakettia, jotta myös se osaa käyttää evästeitä local storagen sijaan ja näin ollen koko sovelluksessa on saatavilla käyttäjän istuntotiedot.

Selainpuolen asiakas luodaan createBrowserClient-funktiolla, joka saa parametreinä Supabasen url-osoitteen sekä anon-avaimen (kuva 7). Selainpuolen asiakkaan avulla voi olla yhteydessä Supabasen eri toimintoihin selainpuolen komponenteissa. Palvelinpuolen asiakkaan luomisessa käytetään createSer-

verClient-funktiota ja myös sille annetaan parametreinä url ja anon-avain kuten selainpuolella. Palvelinpuolen asiakas voi ainoastaan lukea evästeitä, minkä vuoksi tarvitaan middleware-tiedosto, jossa saadaan käyttäjän istunnon tietoja tarkastettua. Palvelinpuolen asiakas tarvitsee Next.js:n cookies-objektin, jotta se pystyy lukemaan ja kirjoittamaan käyttäjän istuntotietoja. Next.js antaa virheilmoituksen, kun evästeitä asetetaan tai poistetaan palvelinpuolella, mutta koska käytössä on middleware, voi niiden käsittelyn jättää huomiotta tässä kohtaa (kuva 8). (Setting up Server-Side… s.a.)

```
import { createBrowserClient } from '@supabase/ssr'

export function createClient() {
  return createBrowserClient(
    process.env.NEXT_PUBLIC_SUPABASE_URL!,
    process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!
  )
}
```

Kuva 7. Selainpuolen asiakas

```
import { createServerClient, type CookieOptions } from '@supabase/ssr'
import { type cookies } from 'next/headers'

export function createClient(cookieStore: ReturnType<typeof cookies>) {
  return createServerClient(
    process.env.NEXT_PUBLIC_SUPABASE_URL!,
    process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!,
    {
      cookies: {
        get(name: string) {
          return cookieStore.get(name)?.value
        },
        set(name: string, value: string, options: CookieOptions) {
          try {
            cookieStore.set({ name, value, ...options })
          } catch (error) {
          }
        },
        remove(name: string, options: CookieOptions) {
          try {
            cookieStore.set({ name, value: '', ...options })
          } catch (error) {
          }
        },
      },
    }
  )
}
```

Kuva 8. Palvelinpuolen asiakas

Sisäänkirjautumisessa käytetään palvelinpuolen asiakasta, joka tuodaan supabase-muuttujaan ja sille annetaan arvoksi cookies-objekti (kuva 9). Kaikki käyttäjähallintaan liittyvät metodit, kuten sisäänkirjautuminen tulee Supabasen auth-moduulista. Tämän jälkeen käyttäjä kirjataan sisään signInWithPassword-metodilla, joka saa arvoiksi käyttäjätunnuksen ja salasanan. Kirjautumisessa lähetetään tiedot Supabasen palvelimelle, joka tarkastaa käyttäjätunnus-salasana yhdistelmän ja mikäli nämä ovat oikeat, palautetaan käyttäjän tiedot.

```
export async function login(email: string, password: string) {
  const cookieStore = cookies()
  const supabase = createClient(cookieStore)

  const { error } = await supabase.auth.signInWithPassword(
    {
      email: email,
      password: password
    }
  )
```

Kuva 9. Käyttäjän sisäänkirjautuminen

Käytössä on Next.js-middleware, jonka kautta kaikki pyynnöt kulkevat ja jossa voidaan tarkastaa käyttäjien istuntoja ja sen mukaan päästää käyttäjiä sallituille sivuille. Middlewaressa otetaan myös käyttöön Supabasen palvelinpuolen asiakas, jonka avulla käsitellään käyttäjien istuntoja palvelinpuolella (ks. liite 1). Kun käyttäjä yrittää kirjautua sisään, middleware tarkastaa, onko käyttäjää löytynyt käyttäen getUser-metodia, ja mikäli käyttäjä löytyy, voidaan päästää käyttäjä eteenpäin sovelluksessa. Koska access-tokenit ovat lyhytikäisiä, täytyy niitä uusia ajoittain ja tämä tapahtuu middlewaressa, jossa voidaan myös asettaa ja poistaa evästeitä. On hyvä muistaa käyttää middleware-tiedostossa getUser- eikä getSession-metodia, kun halutaan tietoja käyttäjästä, sillä istunnon tietoja evästeissä voidaan väärentää. Metodi getUser lähettää aina pyynnön Supabasen Auth-palvelimelle, josta saa varmasti oikean käyttäjätiedon. (Setting up Server-Side… s.a.)

Uloskirjautumisessa käyttäjän istuntotiedot poistetaan evästeistä. Tällöin käytetään signOut-metodia, jonka jälkeen ohjataan käyttäjä takaisin kirjautumissivulle (kuva 10). Samalla kun pyyntö on tehty Supabaselle, ottaa middleware kiinni siitä ennen selaimelle tuloa ja poistaa remove-funktiolla evästeistä istunnon tiedot (ks. liite 1).

```
const logout = async () => {
    const cookieStore = cookies()
    const supabase = createClient(cookieStore)
    const {error} = await supabase.auth.signOut();

    if (!error) redirect("/login")
}
```

Kuva 10. Käyttäjän uloskirjaus

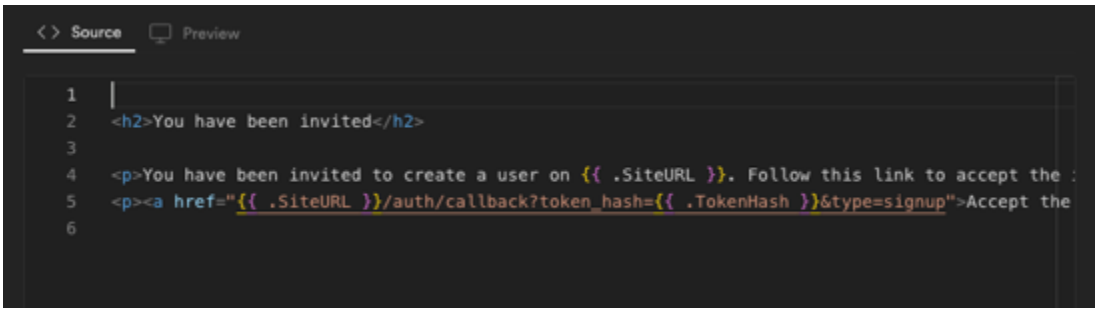
Uuden käyttäjän rekisteröimisessä sovelluksessa käyttäjä syöttää sähköpostiosoitteen ja salasanan, minkä jälkeen kutsutaan signUp-metodia kirjautumistiedoilla käyttäen palvelinpuolen asiakasta (kuva 11).

```
export async function signUp(email: string, password: string) {
  const cookieStore = cookies()
  const supabase = createClient(cookieStore)

  const { error } = await supabase.auth.signUp(
    {
      email: email,
      password: password
    }
  )
```

Kuva 11. Uuden käyttäjän rekisteröinti

Uusi käyttäjä lisätään automaattisesti Supabasen users-tauluun ja kyseiseen sähköpostiosoitteeseen lähetetään varmennuslinkki, jolla uusi käyttäjä varmistaa sähköpostin oikeellisuuden. Hallintapaneelin Authentication/Email Templates kautta sähköpostimallipohjia muokataan käyttämään palvelinpuolen autentikointia ja tässä tapauksessa Confirm signup. Linkkiä muokataan muodosta {{ .ConfirmationURL }} muotoon {{ .SiteURL }}/auth/callback?token_hash={{ .TokenHash }}&type=signup. Linkkiin täytyy vaihtaa tiedot kuten kuvassa 12,

jossa /auth/callback on reitti, jossa käsitellään käyttäjän rekisteröinti ja tämän voi muuttaa oman sovelluksen mukaiseksi. (Setting up Server-Side… s.a.)



Kuva 12. Käyttäjän rekisteröintiin oleva sähköpostimallipohja

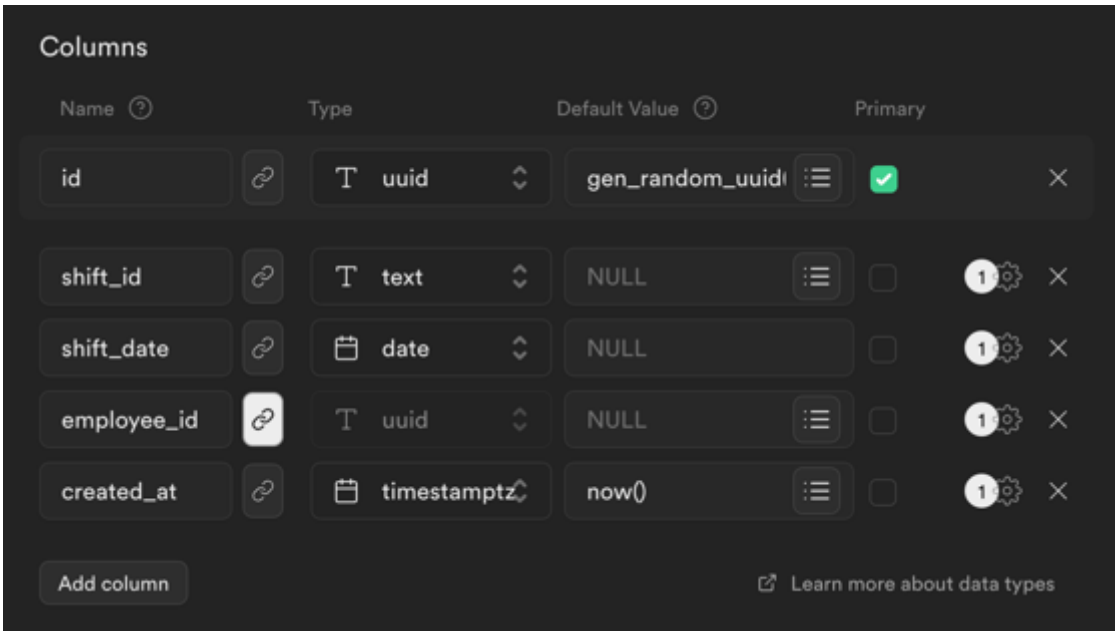
```
import { type EmailOtpType } from '@supabase/supabase-js'
import { cookies } from 'next/headers'
import { type NextRequest, NextResponse } from 'next/server'

import { createClient } from '../../../util/supabase/server'

export async function GET(request: NextRequest) {
  const cookieStore = cookies()

  const { searchParams } = new URL(request.url)
  const token_hash = searchParams.get('token_hash')
  const type = searchParams.get('type') as EmailOtpType | null
  const next = searchParams.get('next') ?? '/'

  const redirectTo = request.nextUrl.clone()
  redirectTo.pathname = next
  redirectTo.searchParams.delete('token_hash')
  redirectTo.searchParams.delete('type')

  if (token_hash && type) {
    const supabase = createClient(cookieStore)

    const { error } = await supabase.auth.verifyOtp({
      type,
      token_hash,
    })
    if (!error) {
      redirectTo.searchParams.delete('next')
      return NextResponse.redirect(redirectTo)
    }
  }
  redirectTo.pathname = '/login'
  return NextResponse.redirect(redirectTo)
}
```

Kuva 13. Reittikäsittelijä käyttäjän rekisteröintiin

Middlewaressa täytyy sallia pääsy myös autentikoimattomille käyttäjille reittiin, jossa rekisteröinti varmistetaan. Route.ts-tiedostossa jonne käyttäjä ohjataan linkistä, vaihdetaan turvakoodi auth-tokeniin ja käytetään palvelinpuolen asiakasta (kuva 13). Mikäli sivun url:sta löytyy tarvittavat token_hash- ja type-muuttujat arvoineen, lähetetään ne Supabaseen käyttäen verifyOtp-metodia. Mikäli virheitä ei tule, ohjataan käyttäjä halutulle sivulle. (Setting up Server-Side… s.a.).

4.3 Tietokannan yhdistäminen sovellukseen

Tietokannan taulujen luominen onnistuu helposti suoraan Supabasen hallintapaneelissa. Table Editor-kohdasta pystyy luomaan uusia tauluja projektia varten sekä muokata jo olemassa olevia. Sarakkeiden lisäys onnistuu luotuun tauluun samalla kun luo uutta taulua ja tauluja pystyy muokkaamaan myös jälkikäteen, kun taulussa on jo tietoa lisättynä (kuva 14). Supabase ehdottaa automaattisesti lisäämään jokaiselle taululle id:n, joka automaattisesti generoituu rivejä lisättäessä sekä created_at, joka kertoo, milloin kyseinen rivi on luotu.



Kuva 14. Uuden taulun luominen Supabasen hallintapaneelissa

Kaikissa uusissa tauluissa on automaattisesti käytössä rivitason suojaus, jolloin tietokantaan ei pääse kukaan ulkopuolinen käsiksi. Tämä on hyvä pitää mielessä, kun yrittää yhdistää sovelluksen tietokantaan, sillä vaikka url ja anon-key olisivat oikein, Supabase ei lähetä mitään tietoja taulusta ennen kuin oikeanlaiset toimintatavat on luotu. Alkuun testimielessä rivitason suojauksen

voi ottaa pois käytöstä, kun haluaa testata, toimiiko yhteys Supabaseen. Rivitason suojauksia ja toimintatapoja voi muokata hallintapaneelin kautta Authentication/Policies. On kuitenkin hyvä muistaa kytkeä se päälle jälkeenpäin ja muokata toimintatapoja omiin tarpeisiinsa, jotta kaikki tieto ei olisi kaikkien saatavilla.

Kun taulu on luotu, voidaan siihen lisätä tietoa joko hallintapaneelin tai sovelluksen kautta. Hallintapaneelin Table Editor kautta lisätään uusi rivi haluttuun tauluun, jonka kautta voi lisätä rivin tai CSV-tiedoston. Sovelluksessa muodostetaan yhteys Supabasen tauluun käyttäen joko selain- tai palvelinpuolen asiakasta sen perusteella, missä kontekstissa Next.js-sovelluksen komponentti tai sivu suoritetaan. Ensin luodaan supabase-muuttuja, johon tuodaan oikea asiakastyyppi, jonka jälkeen voidaan lisätä rivejä haluttuun tauluun kuten kuvassa 15.

```
const {error} = await supabase
.from('generated_shifts')
.insert(
    {
        employee_id: employeeId,
        shift_id: nextShiftCode,
        shift_date: dateString,
    },
)
```

Kuva 15. Rivien lisääminen Supabase-tauluun

Yllä luodaan error-muuttuja, johon Supabase tarvittaessa palauttaa tietoa, mikäli jokin menee pieleen tallennusvaiheessa. From-metodi määrittelee taulun, josta tietoja haetaan ja insert-metodi määrittelee, mitä taulun riville lisätään (kuva 15). Kuvassa 15 määriteltyjen tietojen lisäksi riville tallentuu id ja created_at, joille on taulukossa määritelty oletusarvot, jolloin niitä ei tarvitse erikseen määritellä, ellei halua asettaa omia arvoja. Kun lisätään useampia rivejä tietokantaan, tehokkain tapa on kerätä ne ensin sovelluksen sisäiseen listaan ja tehdä vain yksi lisäys Supabaseen. Tämä vähentää tarvetta monille verkkopyynnöille, parantaen suorituskykyä.

```
const { data, error } = await supabase

.from('employees')
.select('*')
.eq('id', user.id);
```

Kuva 16. Tietojen hakeminen Supabase-taulusta

Tietojen hakeminen taulusta tapahtuu lähes samalla tavalla kuin siihen tallentaminen. Kuvassa 16 luodaan data- ja error-muuttujat, joihin Supabase palauttaa vastauksen. Data-muuttujaan tallennetaan haetut taulukon rivit, jotka Supabase onnistuneesti palauttaa. Tiedot haetaan employees-taulusta, josta valitaan kaikki taulun sarakkeet. Lisäksi halutaan rajata taulukon rivit siten, että Supabase palauttaa ainoastaan ne rivit, joiden id-sarakkeen arvo on sama kuin user.id-muuttujan arvo. Tietokannan taulusta hakiessa on monia eri mahdollisuuksia rajata hakua, mutta yksinkertaisimmillaan riittää, että on määritellyt taulun nimen from-metodissa, ja valitsee kaikki sarakkeet käyttäen *-merkkiä select-metodissa. Kuten tietokannan taulusta hakeminen ja sinne lisääminen, myös poisto tapahtuu samoin, mutta käyttäen delete-metodia (kuva 17).

```
const { error } = await supabase
    .from('generated_shifts')
    .delete()
    .gte('shift_date', date)
```

Kuva 17. Rivien poistaminen tietokannan taulusta

Kun haluaa päivittää taulun riviä, käytetään update-metodia. Päivittäessä taulun riviä täytyy lisäksi määritellä tunniste, jolla kyseisen rivin tunnistaa taulusta. Id on tällaiseen hyvä, sillä se on aina uniikki jokaisella rivillä, eikä näin ollen ole vaaraa, että päivittäisi väärää riviä. Mikäli haluaa palauttaa muoka-

tun, poistetun tai lisätyn rivin tiedot onnistuneen operaation jälkeen, täytyy error-muuttujan lisäksi lisätä data-muuttuja ja loppuun vielä select-komento, jolloin data-muuttujaan tallentuu tieto rivistä (kuva 18).

```
const { data,error } = await supabase
  .from('generated_shifts')
  .update({
    new_start_time : alku,
    new_end_time: loppu,
    edited : new Date()
  })
  .eq('id', shift.id)
  .select()
```

Kuva 18. Taulun rivin päivittäminen ja päivitetyn tiedon palauttaminen

Mikäli on tarve tehdä monimutkaisempia kyselyitä tietokantaan ja haluaa esimerkiksi tietoa useasta tietokannan taulusta samalla kertaa, voi tätä varten luoda funktion, jota kutsuu sovelluksessa. Funktion luominen onnistuu SQL-editorilla ja se kirjoitetaan käyttäen PL/pgSQL-kieltä. Kuvassa 19 funktio hakee tietyn vuoden ja kuukauden mukaan työvuorot generated_shifts-taulusta. Jokaisesta työvuorosta luodaan JSON-objekti, joka sisältää tietoa myös shifts- ja employees-taulusta (kuva 19). Lopuksi kaikki objektit kootaan yhdeksi JSON-taulukoksi, joka palautetaan funktiota kutsuttaessa. Näin ei tarvitse tehdä useampaa kyselyä Supabaseen ja tiedon saa helpossa muodossa heti

sovellukselle. Funktioita voi luoda, poistaa ja muokata myös hallintapaneelissa Database/Functions.

Kuva 19. Funktion luominen Supabase-hallintapaneelin SQL-editorilla

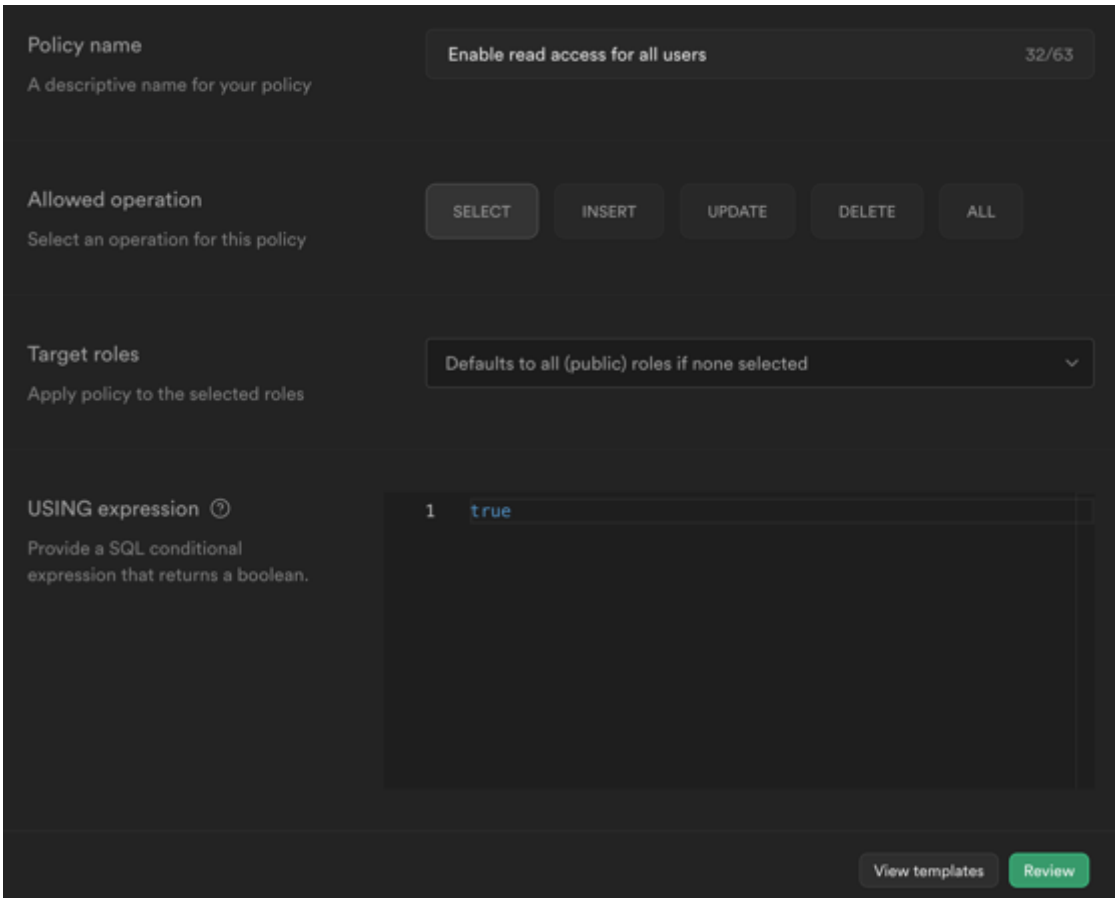
Kun halutaan kutsua funktioita, käytetään rpc-metodia, jolle annetaan arvoiksi funktion nimi ja tarvittavat parametrit (kuva 20). Onnistuneen haun jälkeen funktion palauttamat tiedot tallentuvat data-muuttujaan.

```
const { data, error } = await supabase
.rpc('get_shifts_of_month_with_info', {month, year})
```

Kuva 20. Funktion kutsu Supabase-tietokannasta

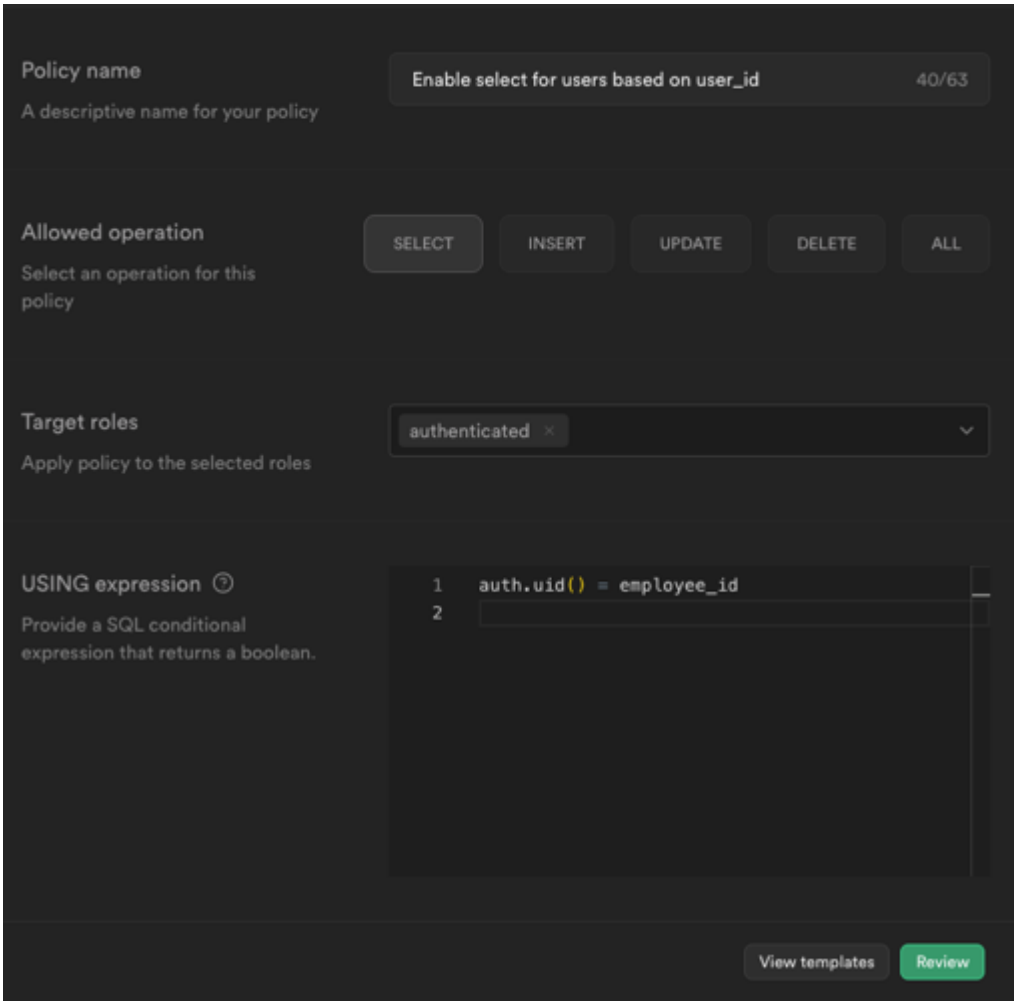
Jotta tietokannan sisältöön ei pääsisi ulkopuoliset käsiksi, täytyy kytkeä rivitason suojaus päälle, mikäli sen on aiemmin ottanut pois käytöstä. Omia toimintatapoja voi luoda hallintapaneelissa helposti joko käyttämällä valmista pohjaa tai luoden alusta alkaen itse. Yksinkertaisimmillaan toimintatapa voi olla kuten kuvassa 21, eli sallitaan lukuoikeudet vain autentikoiduille käyttäjille. Vähin-

tään tällaisen toimintatavan luominen jokaiseen tietokannan tauluun tekee tietokannasta suojatumman, sillä tällöin ulkopuoliset käyttäjät eivät pääse käsiksi tietokantaan tallennettuihin tietoihin.



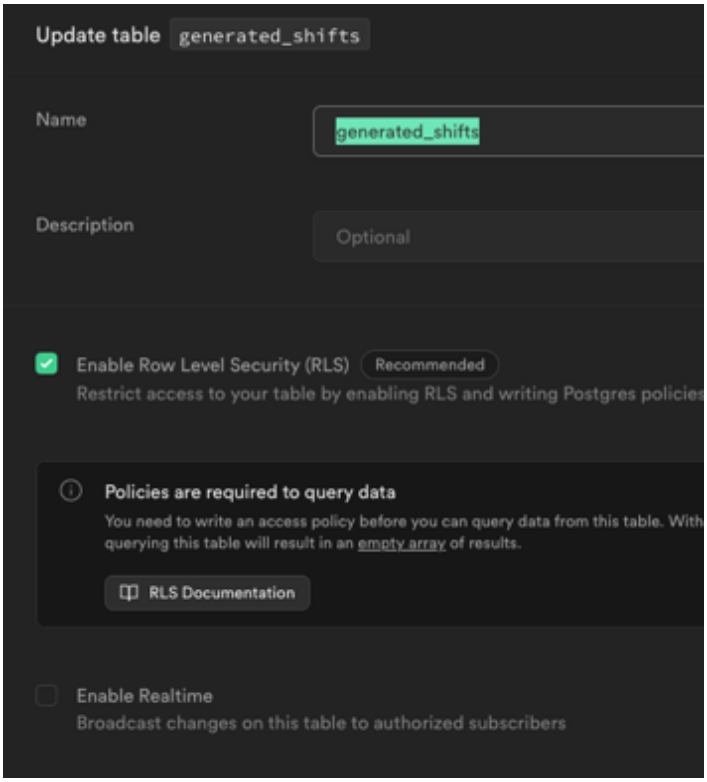
Kuva 21. Yksinkertaisen toimintatavan luominen käyttäen Supabasen hallintapaneelia

Mikäli halutaan rajata myös kirjautuneiden pääsyä kaikkiin tietoihin, voidaan luoda esimerkiksi toimintatapa, joka sallii hakemaan vain ne rivit, joiden employee_id-sarake vastaa käyttäjän omaa id:tä (kuva 22). Tämä onnistuu lisäämällä USING-kenttään rivi, joka tarkastaa käyttäjän id:n ja vertaa sitä taulun sarakkeeseen, kuten kuvassa 22.



Kuva 22. Toimintatapa, joka sallii kyselyn vain käyttäjän omiin vuoroihin

Kun sovelluksessa halutaan seurata tietokannan tauluun tehtyjä muutoksia reaaliajassa, täytyy ottaa Supabasen Realtime-ominaisuus käyttöön. Tämän voi tehdä hallintapaneelin kautta Table Editor/Edit Table, jossa voi klikata Realtime-ominaisuuden käyttöön (kuva 23). Tämän jälkeen taulun kaikkia toimintoja pystyy seuraamaan reaaliajassa. Mikäli on tarvetta rajoittaa mitä tietokantaoperaatioita voi kuunnella tauluissa, voi sen tehdä hallintapaneelissa Database/Replication.



Kuva 23. Realtime-ominaisuuden käyttöönotto Supabase-hallintapaneelissa

```
useEffect(() => {
    getTableData();
    const channel = supabase.channel('realtime shifts').on('postgres_changes',
    {
        event: '*',
        schema: 'public',
        table: 'generated_shifts'
    }, (payload) => {
        getTableData();
    }).subscribe()

    return () => {
        channel.unsubscribe();
    };

}, [supabase]);
```

Kuva 24. Supabase-taulun reaaliaikainen kuunteleminen

Sovelluksessa luodaan supabase-muuttuja, joka käyttää selainpuolen asiakasta. Taulun seuraaminen tehdään Next.js-sovelluksessa useEffect-hookin sisään, jossa annetaan ensiksi nimi kanavalle, jota halutaan kuunnella (’realtime’ ei kelpaa) (kuva 24). Kun halutaan kuunnella Supabase-tietokannan tapahtumia, joka on PostgreSQL-tietokanta, täytyy on-metodiin lisätä ’postgres_changes’. Lisäksi voidaan määritellä muita tietoja tarkentamaan mitä ja

mistä kuunnellaan. Kuvassa 24 on tarkennettu event-parametriin mitä tietokannan muutoksia kuunnellaan, schema-parametriin kuunneltava tietokantakaavio ja table-parametriin taulu, jonka muutoksia kuunnellaan. Vähimmillään riittää, että määrittelee event- ja schema-parametreiksi kaiken *-merkillä, jolloin kaikkia tietokannan muutoksia kuunnellaan. Kun muutos tapahtuu, kutsutaan funktiota, jossa voi määritellä mitä silloin tehdään. Kuvassa 24 payload on tieto siitä mitä on tehty, kun kyseiseen tauluun tehdään muutoksia ja funktion sisässä on määritelty kutsuttavaksi getTableData-funktiota. Lopuksi vielä suljetaan kanava, jolla varmistetaan, että mahdollisia tietovuotoja ei pääse syntymään käyttäen unsubscribe-metodia. (Postgres Changes s.a.)

5 JOHTOPÄÄTÖKSET

BaaS-alusta Supabasen käyttöönotto websovellukseen oli helppoa ja nopeaa. Sen avulla oli helppo luoda tietokannan tauluja ja ottaa niitä käyttöön sovellukseen. Etenkin Next.js sopi hyvin yhteen Supabasen kanssa, ja siihen löytyi hyvin dokumentaatiosta ohjeet. Käyttäjähallinta onnistui myös helposti Supabasen avulla ja sovelluksesta sai näin helposti ja nopeasti suojatumman ja kaikin puolin turvallisemman ilman syvempää osaamista tietoturvasta.

Työn lopputuloksena syntyi ensimmäinen versio työvuorosovelluksesta, jonka päiväkodin henkilökunta voisi ottaa tulevaisuudessa käyttöön. Sovelluksessa otettiin käyttöön Supabasen tietokantaominaisuuksia sekä käyttäjähallinta ja näin ollen saatiin kehitettyä toimiva ja suojattu sovellus. Toimeksiantaja oli kiinnostunut sovelluksesta, mutta käyttöliittymä vaatisi vielä useita parannuksia ennen kuin se olisi valmis käyttöön. Lisäksi sovellus vaatisi testaamista ennen käyttöönottoa.

Työn tekemistä hankaloitti se, että Supabasen ssr-apupaketti oli vasta julkaistu, eikä virallinen dokumentaatio siitä ollut vielä ajan tasalla. Virallinen ohjeistus muuttui työtä tehdessä useaan kertaan, mikä vaati sen, että sovellusta ja raporttia täytyi muuttaa aina sen mukaan. Tämä vei turhaa aikaa, sillä täytyi kaivaa tietoa ensin muista lähteistä ja yrittää kokeilemalla ottaa käyttöön kyseinen lisäosa ennen virallisen dokumentaation päivittymistä, mikä vei aikaa itse sovelluksen kehitykseltä. Muiden ominaisuuksien käyttöönotto kuitenkin

onnistui helposti, sillä Supabasen käytöstä oli jo aikaisempaa kokemusta, ja dokumentaatiokin oli näistä jo vakiintunut.

Kokonaisen sovelluksen luominen alusta alkaen vaati paljon suunnittelua ja valmistelua. Supabasessa oli helppo mm. poistaa ja tehdä uudelleen tietokantatauluja ja käyttäjiä, joten Supabasen ominaisuuksien testaaminen oli helppoa. Koska aikataulu oli lopulta melko tiukka, joutui käyttöliittymästä tinkiä, jotta toiminnallisuus säilyisi ja kaikki toimeksiantajan toivomat ominaisuudet saatiin toteutettua. Kaikki suunnitellut Supabasen ominaisuudet saatiin kuitenkin käytyä läpi työssä ja otettua käyttöön sovellukseen.

6 PÄÄTÄNTÖ

Toimeksianto tuli ehdotuksena toisen päiväkodin opinnäytetyöehdotuksen kautta. Toimeksiantaja oli hieman epävarma uuden sovelluksen tarpeesta, mutta sovimme, että tekisin ensimmäisen version mahdollisesta sovelluksesta heidän pyyntöjen ja tarpeiden pohjalta, ja mikäli se osoittautuisi hyväksi, voisi sen mahdollisesti ottaa tulevaisuudessa käyttöön. Muutamia toiveita lukuun ottamatta sain vapaat kädet sovelluksen tekemiselle. Valitsin opinnäytetyön aiheeksi Supabasen ja sen käytön sovelluksessa, sillä se sopi tähän parhaiten, kun tarve oli luoda sovellus nopeasti yhden kehittäjän toimesta. Valitsin BaaS-alustaksi nimenomaan Supabasen, sillä siitä oli aiempaa kokemusta ja sitä ei ole käsitelty juurikaan varsinkaan suomeksi. Rajasin aihetta lopulta vielä vain tärkeimpiin ja sovelluksen kannalta oleellisimpiin ominaisuuksiin eli tietokantaan ja käyttäjähallintaan. Raportoinnissa käytin hyväksi Supabasen omaa dokumentaatiota sekä useita muita alan julkaisuja ja pyrin valitsemaan vain mahdollisimman uusia lähteitä työn luotettavuuden parantamiseksi.

Supabasen ominaisuuksien esittäminen sovelluksen avulla onnistui hyvin. Sain otettua kaikki suunnitellut toiminnot käyttöön sovelluksessa ja esitettyä ne yksityiskohtaisesti raportissa. Sovelluksen testaaminen ja käyttöliittymäpuoli jäivät kesken, mutta sain toteutettua suunnitellut ominaisuudet ja työtä tehdessä tuli selväksi, miten suunnittelu ja ominaisuuksien rajaaminen on tärkeää, kun on tietty aikataulu. Työtä tehdessä pääsi myös hyvin vahvistamaan teoriatietämystä sovelluskehityksen perusasioista.

Raportin avulla pystyy ottamaan käyttöön Supabasen sekä sen tietokannan ja käyttäjähallinnan, mikäli käytössä on Next.js-sovelluskehys. Työssä on käyty läpi uusin ssr-apupaketti ja sen käyttö, jota ei ole vielä käyty läpi juuri missään, sillä se on vasta hiljattain julkaistu. Jatkokehitysehdotuksena olisi käydä läpi Supabasen muita ominaisuuksia tai sen käyttöä esimerkiksi mobiilisovelluksissa.

LÄHTEET

Adlakha, S. 2023. Backend as a Service – Future of Application Development. LinkedIn. WWW-Dokumentti. Saatavissa: https://www.linkedin.com/pulse/backend-service-future-application-development-sanjay-adlakha/ [viitattu 14.10.2023].

AI & Vectors s.a. Supabase DOCS. WWW-dokumentti. Saatavissa: https://supabase.com/docs/guides/ai [viitattu 31.1.2024].

Aplyca. 2023. Supabase: An Agile Open Source Alternative. WWW-dokumentti. Saatavissa: https://www.aplyca.com/en/blog/blog-supabase-an-agile-open-source-alternative [viitattu 29.10.2023].

Auth s.a. Supabase DOCS. WWW-dokumentti. Saatavissa: https://supabase.com/docs/guides/auth [viitattu 31.10.2023].

Batschinski, G. s.a. What is BaaS – Backend-as-a-service? back4app. WWW-dokumentti. Saatavissa: https://blog.back4app.com/backend-as-a-service-baas/ [viitattu 14.10.2023].

Biskupski, B. & Pytko-Włodarczyk, A. 2019. BackEnd as a Service (BaaS) vs. custom backend: Which one to choose and why? Merixstudio. WWW-dokumentti. Saatavissa: https://www.merixstudio.com/blog/backend-service-baas-vs-custom-backend/ [viitattu 29.10.2023].

Codecademy s.a. What is a Relational Database Management System? WWW-dokumentti. Saatavissa: https://www.codecademy.com/article/what-is-rdbms-sql [viitattu 10.1.2024].

Connecting to your database s.a. Supabase DOCS. WWW-dokumentti. Saatavissa: https://supabase.com/docs/guides/database/connecting-to-postgres [viitattu 4.11.2023].

Database s.a. Supabase DOCS. WWW-dokumentti. Saatavissa: https://supabase.com/docs/guides/database [viitattu 31.10.2023].

Edge Functions s.a. Supabase DOCS. WWW-dokumentti. Saatavissa: https://supabase.com/docs/guides/functions [viitattu 4.11.2023].

Edge Functions Quickstart s.a. Supabase DOCS. WWW-dokumentti. Saatavissa: https://supabase.com/docs/guides/functions/quickstart [viitattu 4.11.2023].

Firebase – Introduction. 2021. Geeks for Geeks. WWW-dokumentti. Saatavissa: https://www.geeksforgeeks.org/firebase-introduction/ [viitattu 12.1.2024].

Gadhavi, M. 2023. All That You Need to Know About BaaS. Radix. WWW-dokumentti. Saatavissa: https://radixweb.com/blog/backend-as-a-service-baas [viitattu 15.10.2023].

Gillis, A. 2020. REST API (RESTful API). TechTarget. WWW-dokumentti. Saatavissa: https://www.techtarget.com/searchapparchitecture/definition/RESTful-API [viitattu 14.2.2024].

IONOS. 2023. Backend as a Service (BaaS). WWW-dokumentti. Saatavissa: https://www.ionos.com/digitalguide/server/know-how/backend-as-a-service-baas/ [viitattu 14.10.2023].

JavaScript Client Library s.a. Supabase DOCS. WWW-dokumentti. Saatavissa: https://supabase.com/docs/reference/javascript/initializing [viitattu 15.1.2024].

Jones, A. 2022. I want a web application – where should I host it? Make IT Simple. Blogi. Saatavissa: https://www.makeitsimple.co.uk/blog/how-are-web-apps-hosted [viitattu 19.2.2024].

Kumar, S. 2023. JWT Authentication in Nodejs – Refresh JWT with Cookie-based Token. WWW-dokumentti. Saatavissa: https://medium.com/@tech-suneel99/jwt-authentication-in-nodejs-refresh-jwt-with-cookie-based-token-37348ff685bf [viitattu 22.1.2024].

Le Thanh, N. 2023. What is Backend Development? Medium. Blogi. Saatavissa: https://medium.com/@namtheartist95/what-is-backend-development-bcc6a15f8472 [viitattu 10.1.2024].

Lemonaki, D. 2022. Frontend VS Backend – What's the Difference? freeCodeCamp. Blogi. Saatavissa: https://www.freecodecamp.org/news/frontend-vs-backend-whats-the-difference/#beintro [viitattu 4.12.2023].

Managing tables, views, and data s.a. Supabase DOCS. WWW-dokumentti. Saatavissa: https://supabase.com/docs/guides/database/tables [viitattu 21.11.2023].

Munene, L. 2023. Best Practices to Building Secure Web Applications. LinkedIn. WWW-dokumentti. Saatavissa: https://www.linkedin.com/pulse/best-practices-building-secure-web-applications-lewis-munene [viitattu 19.2.2024].

Musing, M. 2023. How Supabase became this generation's database. Basedash. Blogi. Saatavissa: https://www.basedash.com/blog/how-supabase-became-this-generations-database [viitattu 29.10.2023].

Next.js. 2023. Geeks for Geeks. WWW-dokumentti. Päivitetty 24.5.2023. Saatavissa: https://www.geeksforgeeks.org/nextjs/ [viitattu 7.2.2024].

Pant, P., Rajawat, A., Goyal, S., Bedi, P., Verma, C., Raboaca, M. & Enescu, F. 2022. Authentication and Authorization in Modern Web Apps for Data Security Using Nodejs and Role of Dark Web. *Procedia Computer Science* 215, 781–790. Verkkolehti. Saatavissa: https://doi.org/10.1016/j.procs.2022.12.080 [viitattu 19.2.2024].

PL/pgSQL – SQL Procedural Language s.a. PostgreSQL. WWW-dokumentti. Saatavissa: https://www.postgresql.org/docs/current/plpgsql-overview.html [viitattu 2.2.2024].

Postgres Changes s.a. Supabase DOCS. WWW-dokumentti. Saatavissa: https://supabase.com/docs/guides/realtime/postgres-changes [viitattu 15.2.2024].

PostgreSQL s.a. About. WWW-dokumentti. Saatavissa: https://www.postgresql.org/about/ [viitattu 4.11.2023].

PostgreSQL Introduction s.a. W3Schools. WWW-dokumentti. Saatavissa: https://www.w3schools.com/postgresql/postgresql_intro.php [viitattu 4.11.2023].

Puton, S. 2023. API Endpoints Protection Using JWT. Blogi. Saatavissa: https://www.netguru.com/blog/api-endpoints-protection-jwt [22.1.2024].

Realtime s.a. Supabase DOCS. WWW-dokumentti. Saatavissa: https://supabase.com/docs/guides/realtime [viitattu 4.11.2023].

Realtime Consepts s.a. Supabase DOCS. WWW-dokumentti. Saatavissa: https://supabase.com/docs/guides/realtime/concepts [viitattu 28.11.2023].

Red Hat. 2022. What is an API? WWW-dokumentti. Saatavissa: https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces [viitattu 10.1.2024].

Row Security Policies s.a. PostgreSQL. WWW-dokumentti. Saatavissa: https://www.postgresql.org/docs/current/ddl-rowsecurity.html [viitattu 28.11.2023].

Sanders, M. 2023. The Ultimate Guide to Web App Authentication: Everything You Need to Know. Binmile. WWW-dokumentti. Saatavissa: https://binmile.com/blog/web-app-authentication-guide/ [viitattu 20.2.2024].

Sandip, R. 2023. Popular Authentication Methods for Web Apps. Baeldung. WWW-dokumentti. Päivitetty 27.10.2023. Saatavissa: https://www.baeldung.com/cs/authentication-web-apps [viitattu 20.2.2024].

Setting up Server-Side Auth for Next.js s.a. Supabase DOCS. WWW-dokumentti. Saatavissa: https://supabase.com/docs/guides/auth/server-side/nextjs [viitattu 29.1.2023].

Storage s.a. Supabase DOCS. WWW-dokumentti. Saatavissa: https://supabase.com/docs/guides/storage [viitattu 5.11.2023].

Supabase Documentation s.a. Supabase DOCS. WWW-dokumentti. Saatavissa: https://supabase.com/docs [viitattu 15.1.2024].

Tables and Data s.a. Supabase DOCS. WWW-dokumentti. Saatavissa: https://supabase.com/docs/guides/database/tables [viitattu 4.11.2023].

Web development. 2023. Geeks for Geeks. WWW-dokumentti. Saatavissa: https://www.geeksforgeeks.org/web-development/#front_dev [viitattu 4.12.2023].

What are cookies? | Cookies definition s.a. Cloudflare. WWW-dokumentti. Saatavissa: https://www.cloudflare.com/learning/privacy/what-are-cookies/ [viitattu 7.2.2024].

What is a web server? s.a. Mdn web docs. WWW-dokumentti. Saatavissa: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_web_server [viitattu 10.1.2024].

What is Database? 2023. Geeks for Geeks. WWW-dokumentti. Päivitetty 7.11.2023. Saatavissa: https://www.geeksforgeeks.org/what-is-database/ [viitattu 10.1.2024].

Wilkins, J. 2021. Front End Developer – What is Front End Development, Explained in Plain English. freeCodeCamp. Blogi. Päivitetty 29.11.2023. Saatavissa: https://www.freecodecamp.org/news/front-end-developer-what-is-front-end-development-explained-in-plain-english/ [viitattu 4.12.2023].

Yiming. 2023. Modern Web Architecture Without a Backend – Using Supabase. ZenStack. WWW-dokumentti. Saatavissa: https://zenstack.dev/blog/supabase [viitattu 31.10.2023].

KUVALUETTELO

SSR-PAKETTIA KÄYTTÄVÄ NEXT.JS:N MIDDLEWARE-TIEDOSTO

```
import { type CookieOptions, createServerClient } from '@supabase/ssr';
import { type NextRequest, NextResponse } from 'next/server';

export async function middleware(request: NextRequest) {
    let response = NextResponse.next({
      request: {
        headers: request.headers,
      },
    })
    const supabase = createServerClient(
      process.env.NEXT_PUBLIC_SUPABASE_URL!,
      process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!,
      {
        cookies: {
          get(name: string) {
            return request.cookies.get(name)?.value
          },
          set(name: string, value: string, options: CookieOptions) {
            request.cookies.set({
              name,
              value,
              ...options,
            })
            response = NextResponse.next({
              request: {
                headers: request.headers,
              },
            })
            response.cookies.set({
              name,
              value,
              ...options,
            })
          },
          remove(name: string, options: CookieOptions) {
            request.cookies.set({
              name,
              value: '',
              ...options,
            })
            response = NextResponse.next({
              request: {
                headers: request.headers,
              },
            })
            response.cookies.set({
              name,
              value: '',
              ...options,
            })
          },
        },
      }
    )

    const {data : {user}} = await supabase.auth.getUser();
    return response
  }

export const config = {
    matcher : [
      '/((?!_next|auth/callback).*)'
    ]
}
```