OAMK OULUN AMMATTIKORKEAKOULU

TURNAUKSENHALLINTASOVELLUKSEN KEHITYS

Teemu Väänänen & Niko Tuulaniemi
Opinnäytetyö AMK
Syksy 2025
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma
Ohjelmistokehityksen suuntautumisvaihtoehto

Tekijä(t): Niko Tuulaniemi, Teemu Väänänen
Opinnäytetyön otsikko: Turnauksenhallintasovelluksen kehitys

Työn valmistumislukukausi ja -vuosi: syksy 2025
Sivumäärä: 51

Opinnäytetyömme tavoitteena oli kehittää web-sovellus video- ja korttipelien turnauksien hallintaan. Sovelluksen avulla käyttäjät pystyvät luomaan ja hallitsemaan turnauksia helposti ja vaivattomasti, sekä osallistujat pystyvät samalla sovelluksella seuraamaan turnauksien kulkua. Opinnäytetyön idea oli opiskelijoiden kehittämä.

Opinnäytetyö toteutettiin kehityspainotteisena työnä, jossa käytiin läpi koko kehityskaari suunnittelusta toteutukseen. Testaaminen suoritettiin kehitystyön lomassa manuaalitestauksena. Viimeisteltyä versiota ei keretty julkaista aikataulullisten haasteiden vuoksi. Opinnäytetyö jakautuu Teknologian Kartoitus- Suunnittelu- ja toteutusosuuksiin. Teknologian Kartoitus-osuudessa käymme läpi valittuja teknologioita, sekä taustaa siitä, miksi näihin päädyttiin. Suunnittelussa perustellaan projektin käyttöliittymän näkyvien elementtien kanssa tehtyjä valintoja, sekä tietokannan rakenteen suunnittelua. Toteutuksessa käymme läpi, kuinka sovelluksen eri näkymät on toteutettu, sekä kuinka tietokanta ja itse turnauslogiikka on toteutettu. Lopussa on vielä pohdintojamme projektista ja kuinka siitä suoriuduimme.

Työn lopuksi meille jäi hyvä pohja lähteä jatkokehittämään sovellusta.

ABSTRACT

Oulu University of Applied Sciences
Degree Program in Information technology
Option of Software development

Author(s): Niko Tuulaniemi, Teemu Väänänen
Title of thesis: Development of tournament management application
Term and year when the thesis was submitted: fall 2025
Number of pages: 51

The aim of our thesis was to develop a web application for managing video- and card game tournaments. The application allows users to create and manage tournaments easily and effortlessly, while participants can use the same application to follow the progress of the tournaments. The idea for the thesis was developed by the students.

The thesis was carried out as a development-oriented project, covering the entire development cycle from design to implementation. Testing was carried out manually during the development work. The final version was not published due to scheduling challenges. The thesis is divided into Technology Mapping, Design, and Implementation sections. In the Technology Mapping section, we review the selected technologies and the background for why they were chosen. The design section justifies the choices made with the visible elements of the project's user interface and the design of the database structure. In the implementation section, we go through how the different views of the application have been implemented and how the database and the tournament logic itself have been implemented. At the end, we share our thoughts on the project and how we performed.

At the end of the work, we were left with a good foundation for further development of the application.

SISÄLLYS

TIIVISTELMÄ 2
ABSTRACT 3
SISÄLLYS 4
SANASTO 5
1 JOHDANTO 7
2 TEKNOLOGIAN KARTOITUS 8
2.1 Next.Js 9
2.2 Zod 9
2.3 Tailwind CSS 10
2.4 Supabase 10
2.5 Github 11
2.6 Insomnia 11
3 SUUNNITTELU 12
3.1 Frontend 12
3.2 Tietokanta 14
4 TOTEUTUS 16
4.1 Projektin- ja versionhallinta 16
4.2 Frontend 17
4.2.1 File-based Routing 20
4.2.2 Komponentit 20
4.3 Tietokanta 24
4.4 Bisneslogiikka 29
4.5 Testaus 36
5 POHDINTA 48
LÄHTEET 50

SANASTO

API	Application Programming Interface, eli ohjelmointirajapinta
Backend	Sovelluksen taustajärjestelmä
Backlog	Teollisuudessa käytetty termi listalle asioista, jotka halutaan toteuttaa, mutta ei ole vielä toteutettu
Button	Nappikomponentti, suorittaa klikatessa sille määritellyn asian, yleensä linkin tai funktion
CSS	Cascading Style Sheets, Web-kehityksessä käytetty kieli, jonka kautta verkkosivulle saadaan tyylitykset
Framework	Sovelluskehys, joka tarjoaa kehittäjälle pohjan, työkalut ja ohjeet kehitykseen.
Frontend	Sovelluksen käyttöliittymä
Funktio	Kutsuttavissa oleva ohjelmiston osa, jota voidaan kutsua toisesta osasta. Toteuttaa tarkasti määritellyn osan ohjelmaa.
GSL	Global Starcraft League, Turnausformaatti, joka on ensiksi ollut käytössä samannimisessä turnaussarjassa
Header	Verkkosivun yläreunassa oleva siirtymisiä helpottava elementti
Kanban	Japanilainen termi, joka tarkoittaa ”näkyvää taulua”, ohjelmistokehityksessä laajalti käytetty tapa hallita tehtäviä

Swiss	Turnausformaatti, jossa pelaajat sijoitetaan otteluihin sijoituksensa mukaan.

1 JOHDANTO

Työn aiheena oli kehittää turnauksenhallintasovellus. Aihe valikoitui tällaiseksi, koska toinen kirjoittajista on innokas Magic: the Gathering – korttipelin turnauspelaaja, ja toinen pitää pelaamisesta yleensä. Sovelluksella on mahdollista tällä hetkellä luoda swiss-formaatin turnauksia. Formaatti toimii käytännössä niin, että pelaajat paritetaan joka kierroksella mahdollisuuksien mukaan samassa pistemäärässä tai mahdollisimman lähellä olevaa pelaajaa vastaan. Tämä muistuttaa hieman round-robinia, jossa pelaajat pelaavat kaikkia toisia pelaajia vastaan. Swiss-formaatissa ei kuitenkaan välttämättä käy näin, vaan pelataan ennalta määritelty määrä kierroksia, yleensä pelaajamäärän mukaan. (Chess.com 2025.)

Jatkokehitysmahdollisuutena on kaavailtu myös GSL-formaatin turnauksia, joissa pelataan ensin lohkovaiheen neljän hengen tai joukkueen lohkoissa, ja niistä voittajat ja toiseksi sijoittuneet pääsevät seuraavaan vaiheeseen.

Turnauksenhallintasovelluksia on olemassa muutamia, mutta ne ovat miltei poikkeuksetta maksullisia, ja niistä yleensä puuttuu turnaushistorian seuraaminen. Tästä syystä päädyimme kehittämään sovelluksen ja siihen taustalle tietokannan. Myös se, ettei tulevaisuudessa ei tarvitse maksaa tällaisesta sovelluksesta on motivaattori projektille.

Käytettyihin teknologioihin päädyttiin koska molemmilla oli hieman kokemusta Javascript-pohjaisista kielistä, sekä halusimme tulevaisuutta silmällä pitäen kehittyä web-kehityksessä. Tietokannaksi valittiin Supabase, sillä siihen saatiin tuttavalta suositus, ja työkalu vaikutti mielenkiintoiselta ja sopivalta esimerkiksi muihin omien ideoiden toteutukseen. Muut valitut teknologiat ovat luontevat tälläiselle teknologiapinolle (tech stack).

2 TEKNOLOGIAN KARTOITUS

Web-sovellus on sovellus, jota käytetään internet -selaimen kautta. Tämä mahdollistaa sovelluksen käyttämisen laitteella, ilman että sitä tarvitsee erikseen asentaa, joka tarjoaa nopeamman ja helpomman käyttökokemuksen käyttäjälle. Web-sovellukset ovat nykypäivänä erittäin suosittuja tapoja tehdä pienempiä sovelluksia, jotka eivät teknisesti vaadi erillistä asennusta.

Web-sovelluksen juuret juontavat 90-luvun alkupuolelle, jolloin Tim Burners-Lee kehitti Web 1.0:n. Web 1.0-aikakautta kutsutaan ”vain luku” -verkoksi, sillä käyttäjä pystyi vain hakemaan ja lukemaan sisältöä hyvin vähäisellä vuorovaikutuksella. Web 1.0:n keskeisiä teknologioita olivat HTML-merkintäkieli, HTTP-protokolla ja URI-osoitejärjestelmä. Lisäksi käyttöön otettiin uudempia kuvauskieliä, kuten XML, XHTML ja tyylitiedostoihin perustuva CSS. Nämä teknologiat mahdollistivat muunmuassa staattisten sivujen kehityksen. (Kariyawasam. 2020.) Staattinen sivu palautetaan sellaisenaan serveriltä, eikä sitä voida prosessoida serverillä ennen käyttäjälle lähetystä. (Geeks for Geeks. 2025s)

Web 2.0:n kehitys toi mukanaan aiempaa enemmän vuorovaikutusta käyttäjän ja verkkosivun välille, minkä vuoksi sitä kutsutaan usein “luku ja kirjoitus” -verkoksi (read–write web). Tällöin käyttäjät pystyivät selaamisen ja lukemisen lisäksi myös luomaan, muokkaamaan ja jakamaan sisältöä. Tämän kehityksen myötä syntyivät monet nykyäänkin tunnetut web-sovellukset, kuten Facebook, Twitter, YouTube ja MySpace. (Kariyawasam. 2020.)

Web 2.0 toi myös mukanaan monipuolisempia ja dynaamisempia sovelluksia, mutta samalla se loi haasteita suorituskyvyn ja hakukoneoptimoinnin kannalta. Sivut alkoivat rakentua yhä enemmän selaimessa (client-side rendering), jolloin ensimmäinen lataus saattoi olla hidas ja sisältö vaikeammin indeksoitavissa. Näiden ongelmien ratkaisemiseksi palvelinpuolinen renderöinti (server-side rendering, SSR) teki paluun moderniin web-kehitykseen. SSR:n etuna on, että sivun HTML generoidaan jo palvelimella, jolloin käyttäjä näkee sisällön välittömästi ja hakukoneet pystyvät indeksoimaan sen tehokkaammin. Tämä parantaa

suorituskykyä, käyttökokemusta ja saavutettavuutta. (Cloudinary. 2025.) Nykyiset kehykset, kuten Next.js, ovat palauttaneet SSR:n keskeiseksi osaksi web-kehitystä.

2.1 Next.Js

NextJs on yhdysvaltalaisen Vercel -yrityksen kehittämä avoimen lähdekoodin React-pohjainen web-sovelluskehys. Se kehiteltiin nopeuttamaan ja helpottamaan web-sovelluksen kehitystä Reactilla, tarjoamalla valmiita ominaisuuksia esimerkiksi reititykseen, palvelinpuolen renderöintiin ja staattisten sivujen generointiin, jotka Reactissa vaatisivat erillisen koodikirjaston käytön tai tyhjästä rakentamisen. (Next.js. 2025a.) Next.js mahdollistaa myös React-pohjaisen sovelluksen rakentamisen niin, että kehittäjä voi yhdistää sekä SSR- että staattisen sivugeneroinnin (SSG) parhaat puolet samassa projektissa. Tämä tekee siitä luonnollisen valinnan modernille web-sovellukselle, jossa suorituskyky ja käyttäjäkokemus ovat keskiössä.

Next.js valittiin projektiin sen modernien ominaisuuksien, helppokäyttöisyyden ja oppimisen halun vuoksi. Tekijöillä oli ennestään kokemusta Reactista, mutta halu oppia uudempaa ja hieman moderninpaa teknologiaa sai valitsemmaan Next.js:n.

2.2 Zod

Zod on avoimen lähdekoodin validointikirjasto Typescriptille ja Javascriptille. Se tarjoaa ominaisuuksia datan validointiin ja yhtenäistämiseen, kuten mallien määrityksen (Schema defining), joita vasten palautuva data voidaan validoida. (Zod. 2025.)

Zod valittiin projektiin, sillä projektissa päätettiin kirjoittaa Next.js:ää Javascriptillä, josta tunnetusti puuttuu tyypitys. Zodin avulla voidaan varmistaa, että sovelluksen lähettämä ja saatu tieto tietokannasta on oikean tyyppistä.

2.3 Tailwind CSS

Tailwind CSS (myöh. Tailwind) on avoimen lähdekoodin framework, jonka on kehittänyt Tailwind Labs. Tailwindin ensimmäinen julkaisu tapahtui 13.5.2019.

Tailwind tarjoaa nopeamman käyttöliittymän rakennusprosessin, antamalla mahdollisuuden käyttää tyyliluokkia suoraan niiden HTML-koodissa, eliminoiden tarpeen mukautetuille tyyliluokille. (Geeks for Geeks, 22.7.2025)

Tailwind otettiin käyttöön projektin alussa lisäämällä paketti projektiin asennuskomennolla: ”npm install tailwind”. Seuraavaksi CSS-tiedostoomme täytyi injektoida Tailwindin base, components, ja utilities – kirjastot riveillä ”@tailwind base;”, ”@tailwind components;” ja ”@tailwind utilities;”. Tailwind luo HTML-koodissa annettujen className – tyyliparametrien mukaan CSS- tiedoston, sekä automaattisesti poistaa käyttämättömät tyylit sieltä. Tämä parantaa verkkosivun suorituskykyä nopeammilla latausajoilla. (Geeks for Geeks, 22.7.2025)

2.4 Supabase

Supabase on Googlen Firebasen innoittama avoimen lähdekoodin BaaS(Backend as a service) -palvelu, eli se tarjoaa mm. tietokannan, rajapinnan, autentikoinnin, auktorisoinnin ja pilvitallennuksen valmiina palveluna, käyttäen apuna toisia avoimen lähdekoodin projekteja. (Supabase, 14.10.2025)

Supabase käyttää tietokantaratkaisun pohjalla Postgres SQL:ää, joka on avoimen lähdekoodin laaja relaationaalinen tietokanta. (PostgreSQL, 14.10.2025).

Supabase soveltuu sovellukseen sen monipuolisen palvelinpuolen palveluiden ja yksinkertaisen käytettävyytensä ansiosta. Supabasen avulla voidaan suoraan aloittaa tietokannan rakentaminen, sisällyttää autentikointi ja käyttää tietokannan tietoja helposti sovelluksessa.

2.5 Github

Github on Microsoftin omistama palvelu, joka käyttä avoimen lähdekoodin Git -versionhallintatyökalua. Githubia käytetään laajalti erityisesti avoimen lähdekoodin projekteissa. Se tarjoaa nykyisin myös projektinhallintatyökaluja, kuten kanban-taulun, jonka kortit voidaan yhdistää suoraan versiohallinnan brancheihin ja comitteihin. Github on helppo valinta tämänkaltaisille projekteille, sillä se on kattava ja pääosin ilmainen version- ja projektinhallintatyökalu.

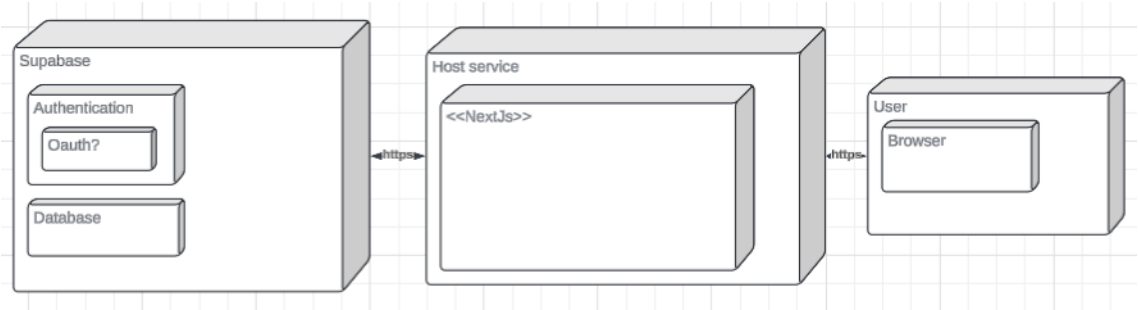
2.6 Insomnia

Insomnia on avoimen lähdekoodin ohjelma rajapintojen testaukseen. Se muistuttaa hyvin paljon suosittua Postman-työkalua, mutta on tietokoneen resurssien kannalta kevyempi ja käyttöliittymältään yksinkertaisempi. (TechTarget 24.7.2025)

3 SUUNNITTELU

Suunnittelu aloitettiin arkkitehtuurista (kuva 1), jossa pyöriteltiin eri vaihtoehtoja ja pyrimme huomioimaan molempien aikaisempaa osaamista. Pääteknologioiksi valittiin Next.js ja Supabase, joista kummastakaan tekijöillä ei ollut varsinaista kokemusta, mutta ne olivat lähellä muita tuttuja teknologioita kuten React ja Firebase.

Sovellus myös suunniteltiin toteutettavaksi palvelimettomalla(serverless)-arkkitehtuurilla, joka tarkoittaa, että sovellukseen ei tulisi perinteisiä backend ja frontend puolia, vaan backend kutsut tapahtuvat käyttöliittymän server-komponenteista. Tämä vähentää sovelluksen palvelintarvetta, sillä sovellus voidaan isännöidä yhdessä paikkaa.

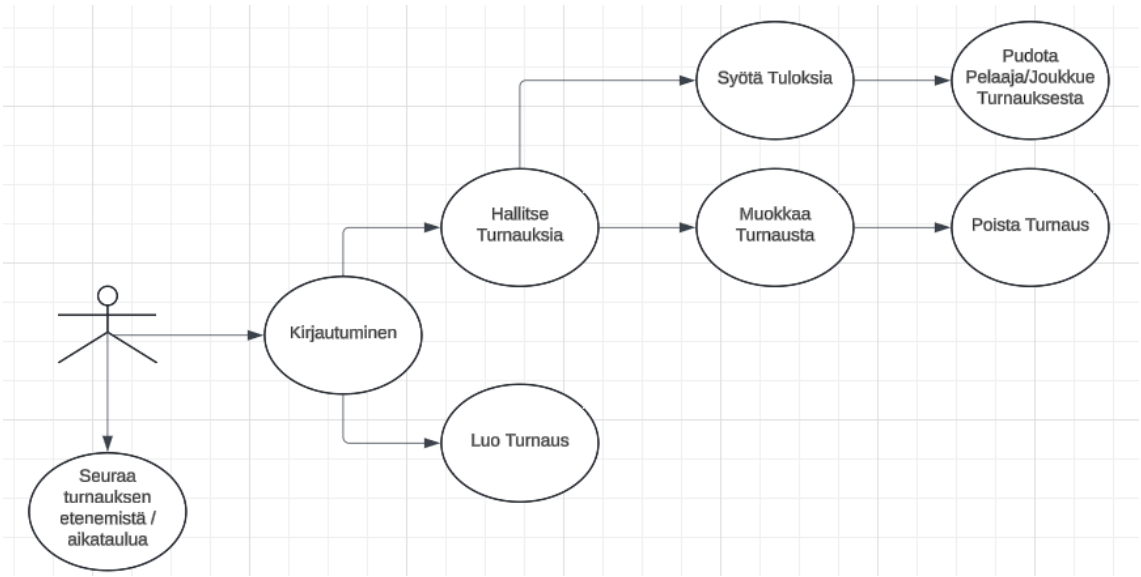


KUVA 1. Arkkitehtuurikaavio suunnitteluvaiheessa

3.1 Frontend

Käyttöliittymän suunnittelussa pyrittiin saamaan aikaan yksinkertaisen, mutta toimivan kokonaisuuden. Värimaailma valittiin rauhallisuus ja yksinkertaisuus mielessä.

Sovellusta suunnitellessa luotiin käyttötapauskaavio, josta selviää siihen halutut toiminnallisuudet (kuva 2).



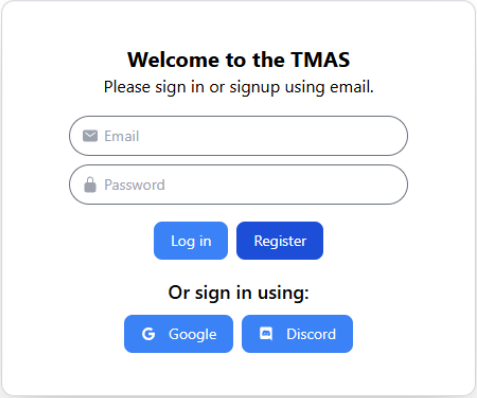
KUVA 2. Käyttötapauskaavio

Komponentit ovat React-pohjaisten sovellusten UI:n rakennuspalikoita. Ne helpottavat paljon käytettyjen asioiden, kuten buttonien hallintaa ja uusiokäyttöä. (Geeks for Geeks 28.8.2025n.)

Jo suunnitteluvaiheessa päätettiin luoda omat komponentit projektiin. Tämä teki helpommaksi muokata sivujen tyyliä, kun muutokset kaikkiin komponenttiluokkiin sai tehtyä muokkaamalla yhtä koodisivua. Tämä vähensi koodirivien toistoja, kun esim. Button-komponentin saa lisättyä sivulle tuomalla sen omasta tiedostostaan import-komennolla ja julistamalla sen koodissa komennolla <Button />.

Turnauksen hallintaan otettiin aluksi tavoitteeksi saada mahdollisuudet lisätä turnaukseen pelaajia, syöttää tulos, muokata turnauksen alkuaikaa, pelin nimeä sekä turnauksen nimeä.

Väärinkäytösten ehkäisemiseksi sekä valvomiseksi määriteltiin myös, että verkkosivulle vaaditaan sisäänkirjautuminen (kuva 3), jos halutaan luoda turnaus sinne. Sisäänkirjautumisen vaatiminen on myös elintärkeä, jossa käyttäjä haluaa muokata jo luomaansa turnausta. Olisi vastuutonta antaa kenen tahansa muokata jonkun toisen luomaa turnausta.



KUVA 3. Sisäänkirjautumiskomponentti

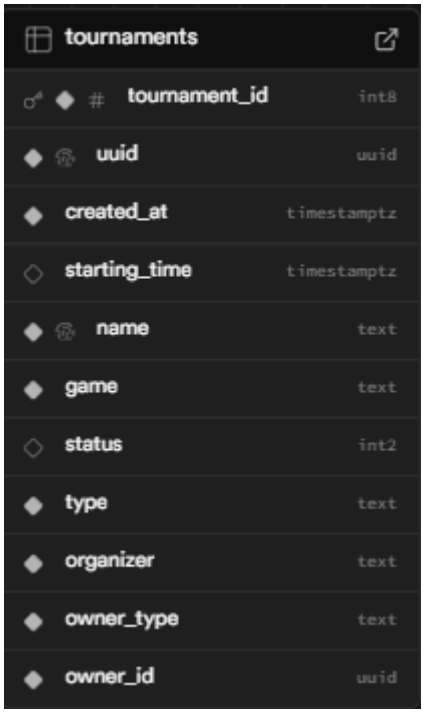
3.2 Tietokanta

Tietokannan suunnittelussa pyrimme saamaan tietokanasta helposti skaalautuvan, mutta ei tarpeettoman laajaa tai monimutkaista relaatiotietokantaa.

Skaalautuvan, mutta yksinkertaisen ja suppean tietokannan saavutamme hyödyntämällä relaatiokannan ominaisuuksia ja välitauluratkaisuja. Esimerkiksi turnauksen omistajuus: tournaments-tauluun suunniteltiin owner_id -kenttä, johon omistajan uniikki tunniste syötettäisiin (kuva 4). Turnauksen kuitenkin voisi tulevaisuudessa omistaa organisaatio, joka maholistaisi useamman hallitsijan yhdelle turnaukselle. Tätä mielessä pitäen suunnittelimme tauluun owner_type -kentän, joka sisältää tekstimuodossa joko ”user” tai ”organization”. Tämän avulla käyttävä taho pystyy päättelemään, onko omistajan tunniste käyttäjän vai organisaation. Tämä mahdollistaa myös ehdollisten tietojen näyttämisen, esimerkiksi organisaation tietoja, jos niitä on saatavilla. Tämä ei kokoluokkamme toteutuksessa olisi tarpeellista, mutta se antaa hyvän skaalautuvan pohjan, jos sovellus joskus kasvaisi.

Supabase tarjoaa myös mahdollisuuden tietokantafunktioille ja Edge -funktioille, joita suunnittelimme alusta alkaen käytettäviksi. Tietokantafunktiot Supabasessa ovat kätevä tapa saada esimerkiksi pelaajatiedot suoraan luettavassa muodossa

käyttöliittymälle, sillä raakadata etenkin välitauluissa sisältää yleensä vain uniikkeja tunnisteita.



KUVA 4. Tietokannan tournaments-taulu

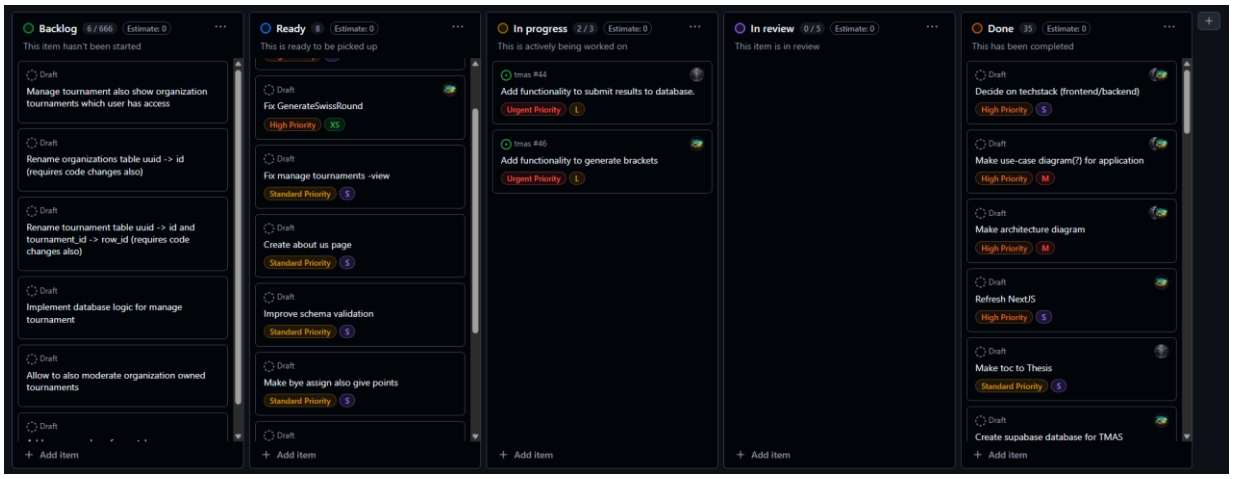
Edge-funktiot ovat tietokantafunktioita monipuolisempia serverless TypeScript-funktioita, jotka ajetaan kutsuvaa käyttäjää lähimpänä olevalla palvelimella. Näin saavutetaan pieni latenssi, sillä kutsu ei matkusta fyysisesti pidemmällä olevalle tietokantapalvelimelle.

4 TOTEUTUS

4.1 Projektin- ja versionhallinta

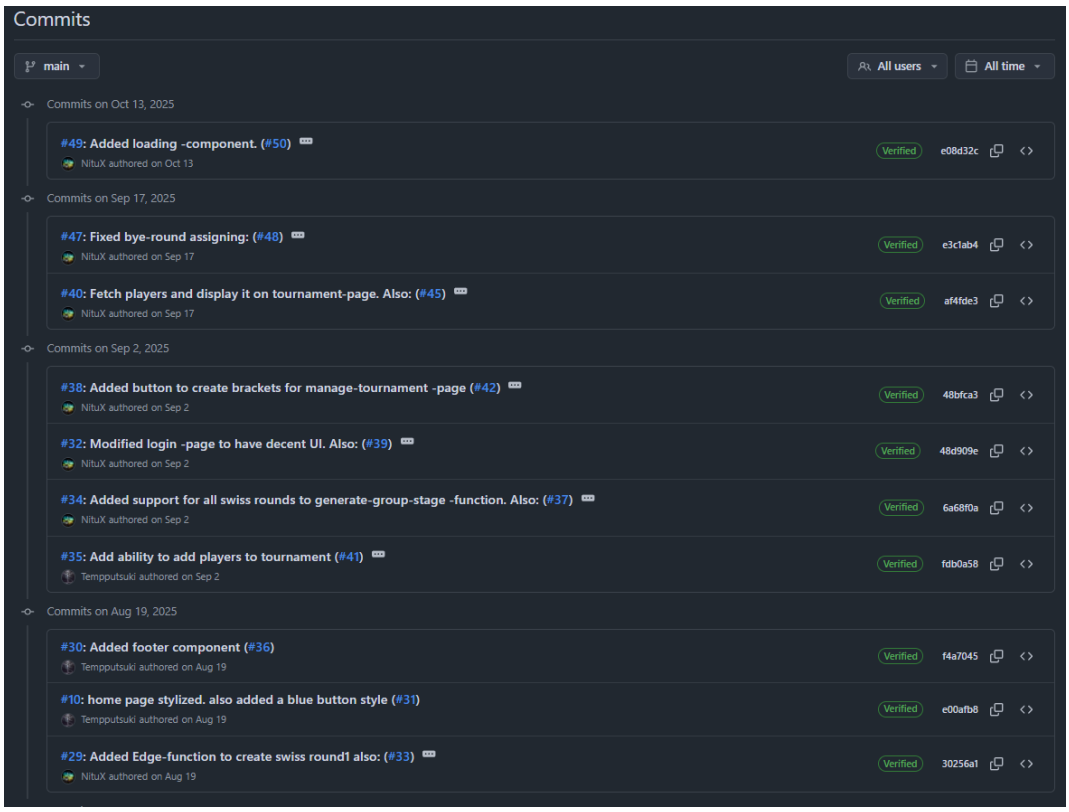
Projektinhallintaan käytettiin suunnitelmien mukaan GitHubin kanban -taulua (kuva 5), jonne lisättiin projektin edetessä aina seuraavaksi suunniteltuja ja tarpeellisia ominaisuuksia backlogille. Backlogilta valittiin seuraavia työnalle otettavia ominaisuuksia palavereissa, joita pidettiin 2–4 viikon välein, tai kun työnalla olevat ominaisuudet saatiin valmiiksi. Kanban-taulussamme oli 5 tilaa: työjono (Backlog), valmis toteutettavaksi (Ready), työn alla (In Progress), katselmoinnissa (In review) ja valmis (Done).

Backlogille listattiin ominaisuuksia hyvin pienellä kynnyksellä, jotta ideat ja kehitystarpeet pysyisivät dokumentoituna, vaikkei niitä toteutettaisikaan projektin aikana. Kun palavereissa valittiin ominaisuuksia mitä seuraavaksi totetutetaan, siirrettiin ne Ready-tilaan, joka tarkoittaa sitä, että ne ovat valmiina kehitykseen. Kun ominaisuutta alettiin toteuttamaan, siirrettiin se In progress-tilaan ja muutettiin tiketiksi(issue). Issue mahdollistaa GitHubissa branchin luomisen suoraan kanban-taululta, joka helpottaa gitin käyttöä hieman. Kun ominaisuus valmistui, luotiin siitä pull request ja siirrettiin issue kanban-taululla In review-tilaan. Kun pull request hyväksytysti katselmoitiin, testattiin ja mergettiin mainiin, GitHub siirsi issuen automaattisesti Done-tilaan.



KUVA 5. GitHubin kanban-taulu

Versionhallinnassa noudatettiin ominaisuuskohtainen muutos (feature-based commit) -periaatetta. Tämä tarkoittaa, että yhdestä tehdystä ominaisuudesta tehdään vain yksi git commit. Näin ollen versiohistoria pysyy siistinä ja luettavana (kuva 6), sekä käytettävyys on yksinkertaista. Myös pull requesteissa muutosten katselmoinnit ovat yksinkertaisempia, kun ei tarvitse katselmoida montaa committia, kun tämän kokoluokan projektissa koodin määrä on muutoinkin hyvin vähäistä.

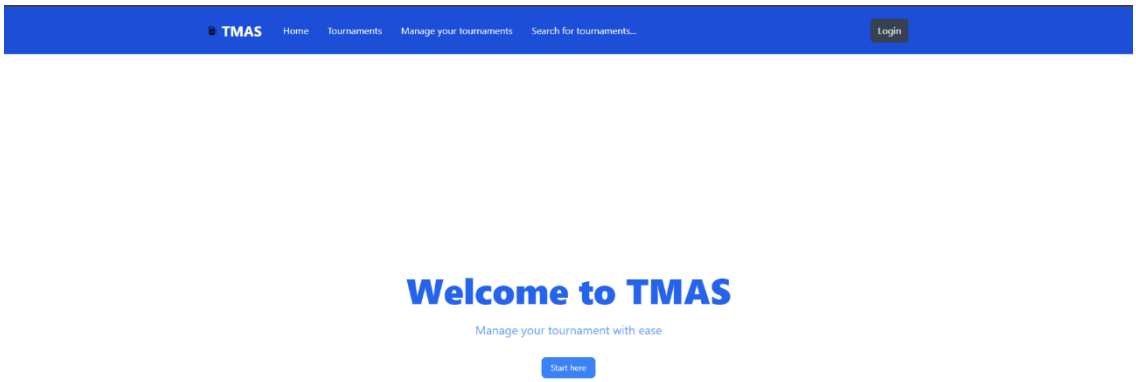


KUVA 6. GitHub versiohistoria

4.2 Frontend

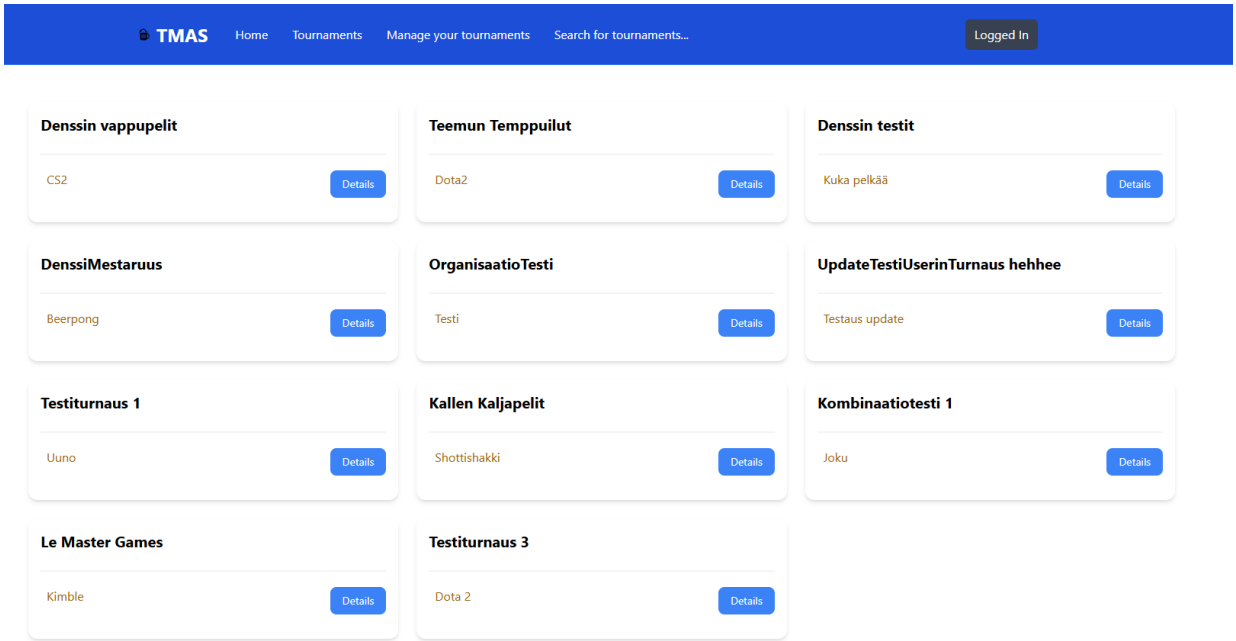
Sovelluskehitys alkoi verkkosivun toteutuksella. Sen rakenne on kutakuinkin seuraavanlainen:

Etusivulta (kuva 7) löytyy yksinkertainen header, jonka kautta voi siirtyä haluamalleen sivulle. Siitä löytyy myös profiilisivu, jolla käyttäjä voi rekisteröityä, kirjautua sisään tai muokata omaa nimeään. Sivulle voi rekisteröityä tällä hetkellä vain Googlen tai Discordin tunnuksella.



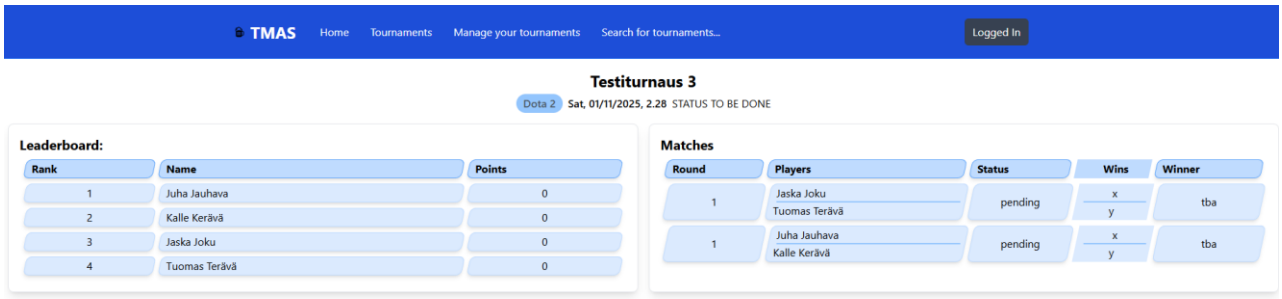
KUVA 7. Etusivu

Tournaments-sivulta (kuva 8) löytyy listaus olemassa olevista turnauksista, joita käyttäjä voi selata.



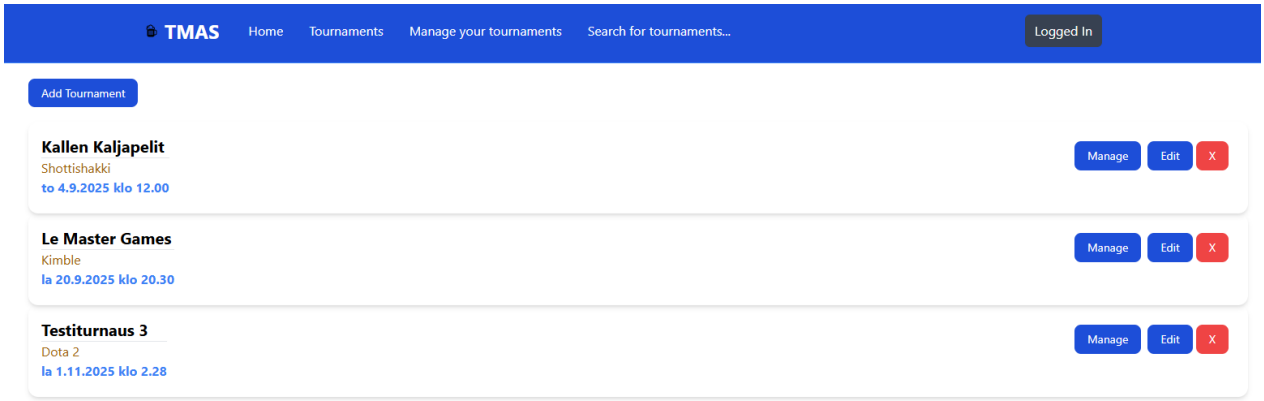
KUVA 8. Tournaments-sivu

Klikkaamalla details-nappia aukeaa tournament-sivu (kuva 9), jolta käyttäjä näkee turnauksen tulostaulukon ja turnauksen ottelut.



KUVA 9. Tournament-sivu

Manage your tournaments-sivulta (kuva 10) löytyy listaus käyttäjän hallinnoimista turnauksista. Käyttäjä voi tällä sivulla muokata, hallita tai poistaa omia turnauksiaan. Sivulla voi myös halutessaan luoda uuden turnauksen.



KUVA 10. Manage your tournaments-sivu

Edit-nappia painamalla voi muokata turnauksen tietoja, kuten nimeä, alkamisajankohtaa, pelin nimeä sekä turnauksen formaattia. Sovellus tehtiin tukemaan swiss-formaattia, jossa pelaajat laitetaan pelaamaan toisiaan vastaan ansaittujen pisteiden mukaan.

Manage-nappia painamalla käyttäjä voi aloittaa turnauksen ja luoda ottelukierrokset. Ottelukierrosten luonti tapahtuu sivulta löytyvällä napilla Generate Round. Tätä nappia painamalla luodut ottelut näytetään sivulla omissa elementeissään. Yksittäi4tä ottelua klikkaamalla elementtiin avautuu Manage Matches-nappi, josta avautuu Manage-komponentti (kuva 11), mistä käyttäjä voi syöttää yksittäisen pelin tuloksen järjestelmään. Manage-komponentti suunniteltiin tukemaan

Bo3 – otteluformaattia, eli jokaista ottelua kohden pelataan niin pitkään, että jompikumpi pelaaja voittaa joko kaksi peliä, tai kierroksen aika loppuu kesken. Komponentissa voi siis syöttää yksittäisen pelin tuloksen enintään kolmesti.

KUVA 11. Manage-komponentti

4.2.1 File-based Routing

Next.js käyttää file-based routingia, eli tiedostot, jotka lisätään pages-kansioon tulevat automaattisesti käytettäviksi reittinä. Käytännössä tämä näkyy siinä, että jos laitetaan index-sivulle eli etusivulle linkki jostain toisesta sivusta, joka on luotu pages-kansioon (esim. tournaments-sivu), voidaan kirjoittaa linkkiin osoitteeksi vain /tournaments ja sovellus osaa automaattisesti ohjata tälle kyseiselle sivulle.

4.2.2 Komponentit

Button

Button-komponentit on eniten käytetty komponentti projektissa, joten on tärkeää, että niiden muokkaaminen on mahdollisimman vaivatonta ja yksinkertaista. Tästä syystä Button-komponentille luotiin oma tyylitaulukko, jota voidaan käyttää antamalla luodulle komponentille parametrinä haluttu tyyli taulukosta. Komponentti ottaa argumenttina myös mm. otsikon, joka näkyy tekstinä luodussa napissa. Näiden lisäksi napille voidaan antaa parametrinä myös funktioita ajettavaksi nappia painamalla. Nämä funktiot täytyy antaa onClick- tapahtumakäsitteljiän kautta (kuva 12).

```
<div className="flex gap-2.5 mb-3">
  <Button onClick={handleSubmit} style="green" title="Add player" />
  <Button onClick={handleClear} style="red" title="Clear" />
</div>
```

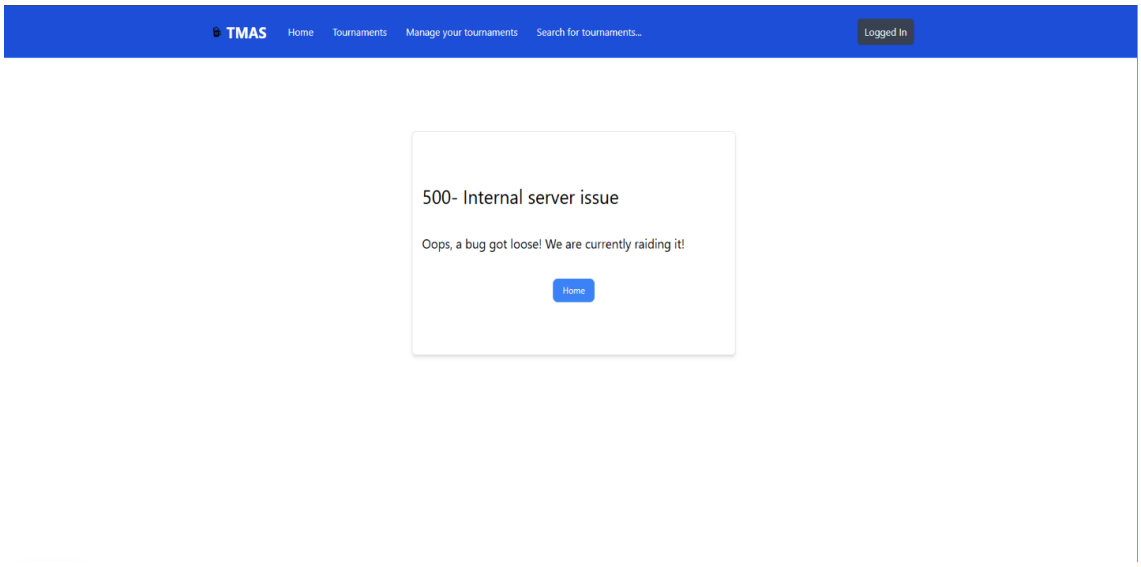
KUVA 12. Button-komponenttien luonti onClick-funktion kanssa

CustomInputField

CustomInputField on komponentti, joka määrittää tekstikenttien reunojen paksuuden ja värin. Sille voidaan myös parametrinä antaa esimerkkiteksti eli placeholder kenttään. Tästä hyvänä esimerkkinä toimivat kuvan 1 Email ja Password-tekstikentät.

Custom error

Projektiin luotiin myös omia virhesivuja (kuva 13), jotta ne näyttäisivät yhteneviltä muiden sivujen kanssa. Next.js luo projektiinsa myös oletuksena virhesivun, mutta se on teemaltaan Next.js:än oma värimaailma.

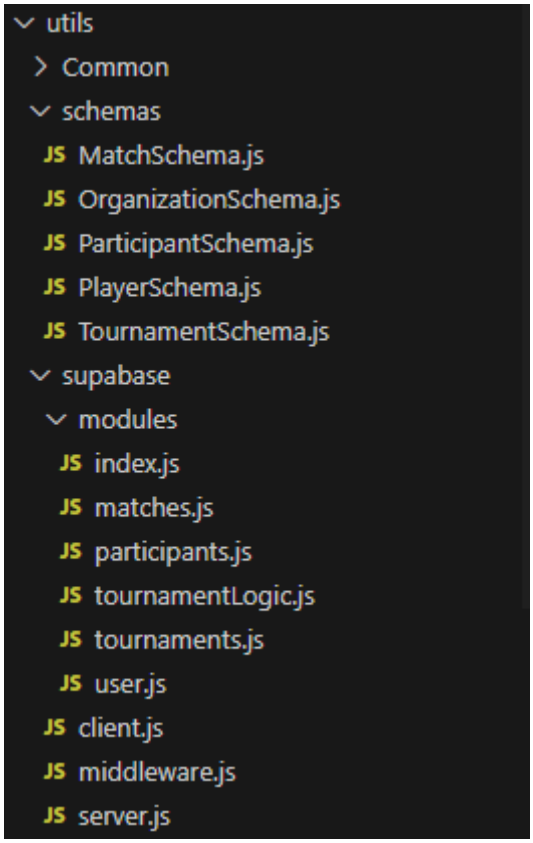


KUVA 13. 500-internal server error-sivu

Tietokantakutsut

Tietokantakutsut toteutettiin projektiin erillisinä palvelinpuolessa ajettavina funktioina, joita kutsutaan käyttöliittymästä. Näinollen selaimessa ajettavan javascript-ohjelman ei tarvitse tietää tietokantasalaisuuksia, eikä tietokantakutsut

näy loppukäyttäjälle. Funktiot koostuvat kolmesta osasta, malli(schema), ohjain (controller tai modules) ja mallien oheen tehdyistä validoinneista (kuva 14).



KUVA 14. Tietokantakutsujen tiedosto

Schemassa on määritelty haettavalle tiedolle malli, jossa muodossa se halutaan käyttöliittymään. Tätä mallia käytetään Zodin validoinneissa, jotka toteutettiin ihan mallien rinnalle samaan tiedostoon (kuva 15).

```
1  import { z } from "zod";
2  import { PlayerShortSchema } from "./PlayerSchema";
3
4  const MatchRawSchema = z.object({
5    match_id: z.string().uuid(),
6    round_num: z.number(),
7    match_status: z.enum(["pending", "ongoing", "finished"]),
8    player1: PlayerShortSchema,
9    player2: PlayerShortSchema,
10   winner: PlayerShortSchema,
11 });
12
13 const CustomMatchSchema = z.array(MatchRawSchema);
14
15 export const validateCustomMatches = (data) => {
16   const result = CustomMatchSchema.safeParse(data);
17
18   if (!result.success) {
19     console.error("Validation failed for match:", result.error.errors);
20     return null;
21   }
22
23   return result.data;
24 };
```

KUVA 15. Matches-schemat ja CustomMatches -validointi

Controllerissa tapahtuu itse kutsu (kuva 16). Funktiolle annetaan parametrina mahdollinen filtteri, yleensä jokin uuid, jonka perusteella funktio voi hakea esimerkiksi tietyn turnauksen kaikki matsit. Funktio kutsuu joko tietokantafunktiota, tai supabasen automaattista APIa, joka palauttaa JSON-muodossa tiedon funktiolle. Jos kutsu onnistuu ja funktio saa tietoa, tarkistetaan tiedon oikeellisuus, vertaamalla sitä vastaavaan schemaan. Jos validointi onnistuu, palautetaan tieto käyttöliittymään. Jos jokin menee vikaan missä tahansa vaiheessa kutsua, palautetaan käyttöliittymään virhe, joka kertoo missä meni vikaan.

```
1   "use server";
2
3   import { createClient } from "@/utils/supabase/server";
4   import { validateCustomMatches } from "@/utils/schemas/MatchSchema";
5   import { cookies } from "next/headers";
6
7   // gets matches by tournament uuid
8   export async function getMatchesByTournamentId(tournamentGuid) {
9     const supabase = await createClient();
10    try {
11      const { data, error, status } = await supabase.rpc(
12        "fetch_tournament_matches",
13        {
14          tournament_id: tournamentGuid,
15        }
16      );
17
18      if (error && status !== 406) {
19        console.warn(`Error fetching data (status: ${status})`);
20        throw new Error(`Error fetching data: ${error.message}`);
21      }
22
23      if (!data || data.length === 0) {
24        return [];
25      }
26
27      const validatedMatchData = validateCustomMatches(data);
28
29      if (validatedMatchData === null) {
30        throw new Error("Match data failed validation");
31      }
32
33      return validatedMatchData;
34
35      // Catch and rethrow any error that occurs during fetching or validation
36    } catch (error) {
37      console.error("Failed to fetch or validate match", error);
38      throw error;
39    }
40  }
```

KUVA 16. Matches-controller

4.3 Tietokanta

Tietokannan toteutus aloitettiin luomalla uusi organisaatio supabasen verkkosivuilta, jonka jälkeen organisaatioon lisättiin kehittäjät. Seuraavaksi organisaatiolle luotiin projekti, joka esimerkiksi MySql:ssä vastaisi yhtä tietokantaa. Luontiprosessi oli yksinkertainen, ja Supabasen oma dokumentaatio tarjosi tähän erittäin selkeät ohjeet.

Tournaments-taulu

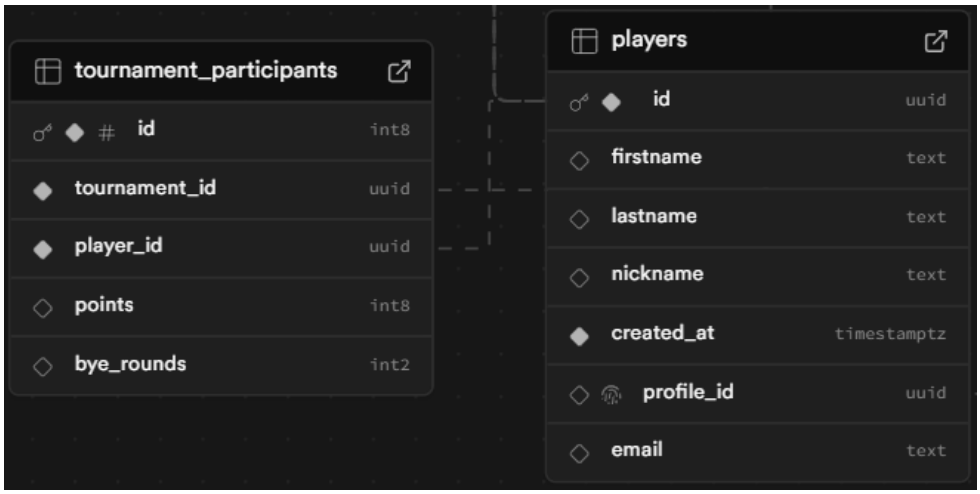
Taulujen luonti tapahtui Supabasen verkkosivuilla projektin dashboardilta. Ensimmäiseksi luotiin projektin perusta, tournaments-taulu (kuva 4). Tournaments-taulu sisältää turnauksen perustiedot: juoksevan id:n(id), uniikin tunnisteen(uuid), luontiajan(creation_datetime), turnauksen aloitusajan(starting_datetime), nimen(name), pelin(game), tilan(status), tyypin(type), organisoijan(organization), organisoijan tyypin(organization_type) sekä omistajan uniikin tunnisteen(owner_id).

Players-taulu

Seuraavaksi luotiin players-taulu (kuva 17), jossa sijaitsee pelaajien tiedot, kuten etu- ja sukunimi, pelaajanimi, sähköposti ja uniikki tunniste. Supabase luo myös automaattisesti autentikointia lisätessä profiles-taulun, jonne se kerää autentikointiin tarvittavat tiedot, kuten käyttäjänimen, sähköpostin ja käyttäjän uniikin tunnisteen. Emme kuitenkaan halunneet pelaajatietoja ja kirjautumistietoja samaan tauluun, jotta pelaajakortin voi myös lisätä pelaajalle, jolla ei ole käyttäjätunnusta sovellukseen.

Tournament_participants-taulu

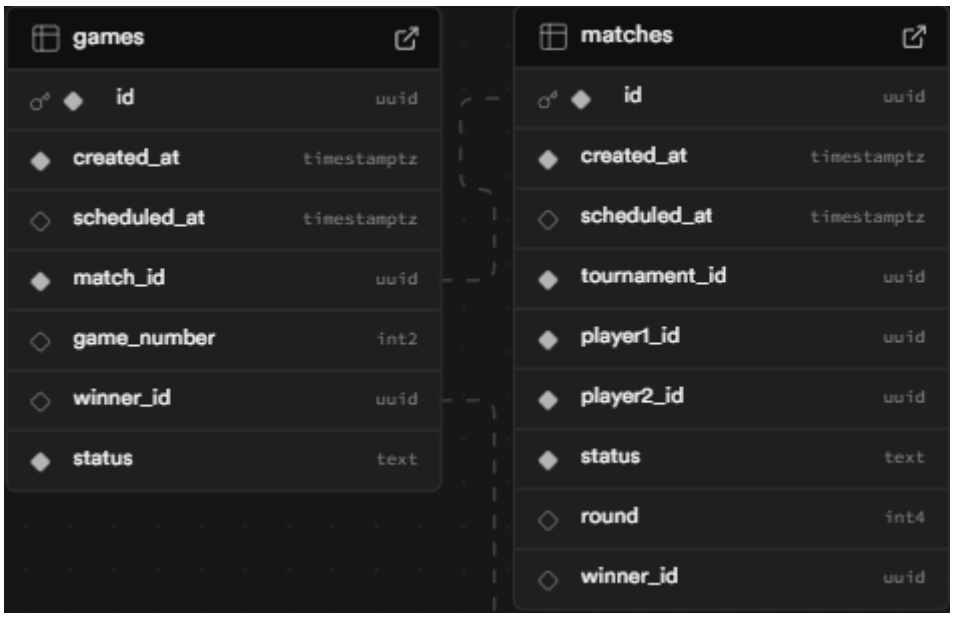
Jotta pelaajat voidaan lisätä turnauksiin, tarvitaan tournament_participants -välitaulu käyttäen yhteystyyppiä monen suhde moneen (kuva 17). Tämä avaa mahdollisuuden tallentaa yhden pelaajan tiedot vain kerran tietokantaan, ja käyttää samoja pelaajan perustietoja useammassa turnauksessa. Myös pelaajan turnauskohtaiset pisteet ja bye:t tallennetaan tähän välitauluratkaisuun, jotta turnauskohtainen pisteidenlasku onnistuisi helposti saman pelaajan pelatessa useammassa eri turnauksessa.



KUVA 17. Tietokannan tournament_participants- ja players -taulut

Matches- ja games-taulut

Jotta pelikohtaisia tietoja voidaan käsitellä, tarvittiin seuraavaksi ottelu ja pelitaulut, matches ja games (kuva 18). Matches-taulu sisältää ottelun tiedot, kuten uniikin tunisteen, turnauksen unikin tunnisteen, kierroksen, vastakkain olevien pelaajien uniikit tunnisteet, tilan ja voittajan unikin tunnisteen. Games-taulu sisältää pelien tiedot, kuten pelin uniikin tunnisteen, ottelun uniikin tunnisteen, pelin järjestysnumeron, tilan sekä voittajan uniikin tunisteen.



KUVA 18. Tietokannan games- ja matches -taulut

Tietokantafunktiot

Tietokantaan toteutettiin useita pienempiä funktioita, joilla yksinkertaistettiin tietojen hakua, lisäämistä ja muokkaamista. Esimerkiksi kun haetaan turnaussivulle tiedot turnauksesta, tietoa halutaan useasta eri taulusta: turnaustiedot tournaments-taulusta, pelaajatiedot tournament_participants-taulusta sekä players-taulusta, matsi- ja pelitiedot matches ja games -tauluista. Usean erillisen kyselyn sijasta luotiin yksi tietokantafunktio (kuva 19), jota kutsumalla sovellus voi hakea kaikki tarvittavat tiedot tietokannasta vain yhdellä tietokantakutsulla. Funktio palauttaa kaikki tarvittavat tiedot suoraan sovelluksen käyttämässä muodossa, jolloin erillisiä tietokantakutsuja ei tarvita, ja tiedot voidaan näyttää sellaisenaan sovelluksessa.

```sql
CREATE OR REPLACE FUNCTION public.get_matches_with_games_by_tournament(p_tournament_id uuid)
  RETURNS TABLE (
  match_id uuid,
  round_num int,
  status text,
  player_1 public.player_info,
  player_2 public.player_info,
  winner public.player_info,
  games public.game_info[]
)
LANGUAGE sql
STABLE
AS $$
    SELECT
        m.id AS match_id,
        m.round AS round_num,
        m.status AS status,
        (p1.id, p1.firstname, p1.lastname, p1.nickname)::public.player_info AS player1,
        (p2.id, p2.firstname, p2.lastname, p2.nickname)::public.player_info AS player2,
        (w.id, w.firstname, w.lastname, w.nickname)::public.player_info AS winner,
        COALESCE((
          SELECT array_agg(
            (
           g.id,
            g.game_number::smallint,
            g.status,
            g.winner_id
            )::public.game_info
        ORDER BY g.game_number
        )
        FROM public.games g
        WHERE g.match_id = m.id
        ), ARRAY[]::public.game_info[]) AS games
    FROM public.matches m
    LEFT JOIN public.players p1 ON p1.id = m.player1_id
    LEFT JOIN public.players p2 ON p2.id = m.player2_id
    LEFT JOIN public.players w ON w.id = m.winner_id
    WHERE m.tournament_id = p_tournament_id
    ORDER BY m.id;
$$;
```

KUVA 19. get_matches_with_games_by_tournament -tietokantafunktio

Jos tiedot haettaisiin ilman funktioita, esimerkiksi turnauksen osallistujalista sisältäisi pelkästään pelaajien uuid:t, joiden perusteella täytyisi sovelluksesta hakea tietokannan players-taulusta erikseen pelaajan tiedot, jotta pelaajien nimet voidaan näyttää tulostaulukossa.

Turvallisuus ja autentikointi

Supabase tarjosi lähes valmiita ominaisuuksia turvallisuuteen ja autentikointiin. Kirjautumisvaihtoehtoina sovelluksessa käytetään Supabasen tarjoamia kolmannen osapuolen kirjautumisvaihtoehtoja: Discord ja Google, sekä perinteistä sähköposti ja salasana -yhdistelmää.

Supabase docs tarjosi valmiit komponentit kirjautumiselle, jotka liitettiin lähes sellaisenaan sovellukseen. Kolmannen osapuolen kirjautumisvaihtoehdoille lisättiin Supabasen asetuksiin API-keyt, jotka täytyi käydä luomassa erikseen Discordin ja Googlen sovelluskehitys-sivuilta.

Sovelluksen turvallisuus huomioitiin Supabasen Row Level Security:n (RLS) avulla. Tietokantaan määriteltiin taulukohtaisesti, mitä eri tietoja käyttäjät saavat lukea, kirjoittaa ja muokata. Esimerkiksi turnaustaulun tietoja pääsee kirjautumaton käyttäjä katsomaan, mutta ei lisäämään, muokkaamaan tai poistamaan. Näille ominaisuuksille täytyy erikseen lisätä sääntö, jotta tietyt kirjautuneet käyttäjät pääsevät tietoja muokkaamaan. Kuten turnauksen tietoja voi muokata vain turnauksen omistaja, jonka uuid löytyy tournaments-taulun owner_id-kentästä.

Sovelluksen ja tietokannan välinen viestintä tapahtuu supabasen sisäisen API:n kautta, joka käyttää sisäänrakennetusti JsonWebToken:ia(JWT) käyttäjän ja istunnon tunnistamiseen.

4.4 Bisneslogiikka

Varsinainen turnauslogiikka tapahtuu Supabasen edge-funktio-ominaisuudella, jolle annetaan parametriksi vain turnauksen ID ja kirjautunut käyttäjä. Funktio hakee sisäisesti tietokannasta tarvittavat turnauksen tiedot, osallistujat pisteineen ja edelliset pelit. Näiden avulla funktio parittaa pelaajat edellisen kierroksen perusteella, ja jos kyseessä on ensimmäinen kierros, arpoo satunnaiset parit. Parituksen jälkeen funktio luo matsit ja pelit tietokantaan, sekä lähettää paluuviestin sovellukselle.

Parametrien validointi

Funktiota kutsuttaessa tarkistetaan ensin HTTP-metodin ja ympäristömuuttujien oikeellisuus, haetaan käyttäjän UUID tietokannasta Supabase-istunnon avulla ja tarkistetaan pakollisten parametrien olemassaolo. (kuva 20)

```
12    // 1. Validate HTTP method
13    if (req.method !== "POST") {
14      return errorJson("Method Not Allowed", 405);
15    }
16
17    try {
18      // 2. Validate environment variables
19      const supabaseUrl = Deno.env.get("SUPABASE_URL");
20      const supabaseKey = Deno.env.get("SUPABASE_SERVICE_ROLE_KEY");
21      if (!supabaseUrl || !supabaseKey) {
22        throw errorJson("Supabase environment variables are not set.", 500);
23      }
24
25      // 3. Validate Authorization header
26      const authHeader = req.headers.get("Authorization");
27      if (!authHeader) {
28        return errorJson("Unauthorized: Missing Authorization header", 401);
29      }
30
31      // 4. Create supabase client
32      const supabase = createClient(supabaseUrl, supabaseKey, {
33        global: { headers: { Authorization: authHeader } },
34      });
35
36      // 5. Fetch user id for checking tournament rights
37      const {
38        data: { user },
39        error: userError,
40      } = await supabase.auth.getUser();
41      if (!user) {
42        return errorJson("Unauthorized", 401);
43      }
44      const user_uuid = user.id;
45
46      // 6. Parse and validate request body
47      const { tournament_uuid } = await req.json();
48      if (!tournament_uuid) {
49        return errorJson("Missing required parameters", 400);
50      }
```

KUVA 20. Bisneslogiikan kutsun validointi

Tarvittavien tietojen haku

Vaadittujen tietojen validoinnin jälkeen tarkistetaan käyttäjän oikeus kyseessä olevaan turnaukseen, sekä haetaan turnauksen tiedot, osallistujat sekä mahdolliset edelliset kierrokset tietokannasta. Samalla alustetaan kierrosluvuksi 1, joka muokataan varsinaiseksi kierrosluvuksi myöhemmin, mikäli edellisiä kierroksia löytyy (kuva 21). Kierrosluvun päättelyssä otetaan huomioon maksimikierrokset, mitä turnauksen pelaajamäärällä pystytään pelaamaan. Maksimikierroksiksi on

määritelty round robin, eli tilanne, jolloin kaikki pelaajat ovat jo pelanneet toisiaan vastaan kerran.

```
61    // 8. Fetch tournaments participants
62    const { data: participants, error: participantsError } = await supabase
63      .from("tournament_participants")
64      .select("id, player_id, points, bye_rounds")
65      .eq("tournament_id", tournament_uuid);
66
67    if (participantsError || !participants || participants.length === 0) {
68      return errorJson("No participants found for the tournament", 404);
69    }
70
71    let roundParticipants = participants;
72
73    // 9. Fetch previous matches
74    let previousMatches = [];
75    let roundNumber = 1; // Default to round 1 if no matches exist
76    const { data: matchesData, error: previousMatchesError } = await supabase
77      .from("matches")
78      .select("round, player1_id, player2_id")
79      .eq("tournament_id", tournament_uuid);
80
81    if (previousMatchesError) {
82      return errorJson(
83        `Error fetching previous matches: ${previousMatchesError.message}`,
84        500
85      );
86    }
87
88    // 10. Determine round number
89    if (matchesData && matchesData.length > 0) {
90      previousMatches = matchesData;
91      roundNumber = Math.max(...matchesData.map((match) => match.round)) + 1;
92
93      // Check if all possible rounds have been played (round robin)
94      const playerCount = participants.length;
95      const maxRounds = playerCount % 2 === 0 ? playerCount - 1 : playerCount;
96      if (roundNumber > maxRounds) {
97        return errorJson(
98          "All possible rounds have already been generated for this tournament.",
99          400
100         );
101       }
102   }
```

KUVA 21. Turnauksen tietojen haku ja kierrosluvun päättely

Pelaajien paritus

Tietojen haun jälkeen funktio aloittaa pelaajien parituksen. Ensimmäisellä kierroksella sekoitetaan pelaajalista, muutoin pelaajat järjestetään pisteiden mukaan suuruusjärjestykseen. Mikäli pelaajia on pariton määrä, annetaan viimeisenä

listalta löytyvälle pelaajalle vapaakortti(bye) sillä ehdolla, että pelaaja ei ole aikaisemmilla kierroksilla saanut byetä.

Parituksen valmistelun jälkeen funktio kutsuu apufunktiota generateSwissRound (kuva 22), joka lopulta parittaa pelaajat. Alkuun funktio luo setin jo pelanneista pareista, jotta voidaan parittaessa varmistua, etteivät pelaajat ole jo pelanneet toisiaan vastaan. Apufunktio luo myös toisen setin pelaajista, jotka on jo parittanut luotavalle kierrokselle. Näin apufunktio pystyy valitsemaan Swiss-logiikan näkökulmasta parhaat pelaajaparit.

Seuraavaksi apufunktio lähtee kokeilemaan pareja for-loopissa, apufunktio aloittaa suurimmat pisteet omaavasta pelaajasta, ja koittaa löytää sille parin mahdollisimman samoissa pisteissä olevasta pelaajasta, jota vastaan hän ei ole vielä pelannut. Kun pari löytyy, lisätään se matches-arrayhyn, joka sisältää matches - tietokantataulun pakolliset kentät, eli kierroksen(round), pelaajan 1 uuid:n, pelaajan 2 uuid:n, turnauksen uuid:n ja statuksen. Status saa arvokseen ”pending”, sillä peliä ei vielä ole pelattu. Paritetut pelaajat myös lisätään used-settiin, jotta vältytään lisäämästä pelaajaa useaan saman kierroksen peliin. Apufunktio käy läpi koko participants-taulukon varmistaen aina ensin used-setistä, ettei pelaajaa ole vielä tälle kierrokselle paritettu.

Jos pelaajaa ei jostain syystä voitu parittaa kenenkään kierroksen pelaajan kanssa, lisätään se unpaired-arrayhyn. Jos parituksen jälkkeen unpaired-array sisältää yhden tai useamman pelaajan, palauttaa apufunktio virheen sekä listan pelaajista, joita ei voitu parittaa. Tällöin apufunktio ei luo tietokantaan saakka kierrosta, vaan kierroksen lisäys keskeytyy. Muussa tapauksessa apufunktio palauttaa varsinaiselle funktiolle valmiin listan matseista, joita kierroksella pelataan.

```
41  //Generate swiss round, pair participants based on order they're given to function.
42  export function generateSwissRound(
43    participants: Participant[],
44    round: number,
45    tournament_uuid: string,
46    previousMatches: any[]
47  ): Match[] {
48    const matches = [];
49    const playedPairs = new Set();
50
51    // Build set of already played pairs
52    if (previousMatches) {
53      previousMatches.forEach((m) => {
54        playedPairs.add(`${m.player1_id}-${m.player2_id}`);
55        playedPairs.add(`${m.player2_id}-${m.player1_id}`);
56      });
57    }
58
59    const used = new Set();
60    let unpaired = [];
61
62    for (let i = 0; i < participants.length; i++) {
63      if (used.has(participants[i].player_id)) continue;
64      let paired = false;
65      for (let j = i + 1; j < participants.length; j++) {
66        if (used.has(participants[j].player_id)) continue;
67        const pairKey = `${participants[i].player_id}-${participants[j].player_id}`;
68        if (!playedPairs.has(pairKey)) {
69          matches.push({
70            round,
71            player1_id: participants[i].player_id,
72            player2_id: participants[j].player_id,
73            tournament_id: tournament_uuid,
74            status: "pending",
75          });
76          used.add(participants[i].player_id);
77          used.add(participants[j].player_id);
78          paired = true;
79          break;
80        }
81      }
82      if (!paired) {
83        unpaired.push(participants[i].player_id);
84        used.add(participants[i].player_id);
85      }
86    }
87
88    if (unpaired.length > 0) {
89      throw (
90        (new Error(
91          `Could not pair all players. Unpaired player IDs: ${unpaired.join(
92            ", "
93          )}`
94        ),
95        400)
96      );
97    }
98
99    return matches;
100 }
```

KUVA 22. GenerateSwissRound-apufunktio

Matsien ja pelien luonti

Mikäli pelaajat saadaan onnistuneesti paritettua, funktio luo tietokantaan matches-taulun rivit GenerateSwissRound-funktion luoman matches-arrayn perusteella. Lisäksi funktio kutsuu toista apufunktiota generateGamesForMatches (kuvat 23 ja 24), joka luo kolme peliä jokaista luotua matsia kohden. Pelit lisätään games-tauluun ja jokainen games-taulun rivi linkitetään matsiin vierasavaimen match_uuid:n avulla.

```
160    // 15. Insert matches
161    const { data: insertedMatches, error: insertMatchesError } = await supabase
162      .from("matches")
163      .insert(matches)
164      .select();
165
166    if (insertMatchesError) {
167      return errorJson(
168        `Error inserting matches: ${insertMatchesError.message}`,
169        500
170      );
171    }
172
173    // 16. Generate and insert games
174    const games = generateGamesForMatches(insertedMatches);
175
176    const { error: insertGamesError } = await supabase
177      .from("games")
178      .insert(games);
179
180    if (insertGamesError) {
181      return errorJson(
182        `Error inserting games: ${insertGamesError.message}`,
183        500
184      );
185    }
```

KUVA 23. Matsien ja pelien luonti

```
102  //Generate 3 games for each match of the round
103  export function generateGamesForMatches(matches: Match[]): Game[] {
104    const games = [];
105    matches.forEach((match) => {
106      for (let game_number = 1; game_number <= 3; game_number++) {
107        games.push({
108          match_id: match.id,
109          game_number,
110          status: "pending",
111        });
112      }
113    });
114    return games;
115  }
```

KUVA 24. generateGamesForMatches-apufunktio

Paluuviesti

Kun matsit ja pelit luodaan onnistuneesti, palauttaa funktio sovellukselle vastauksen onnistumisesta, joka sisältää viestin luotujen matsien ja pelien määrästä (kuva 25).

```
187      // 17. Return success/error response
188      return new Response(
189        JSON.stringify({
190          success: true,
191          message: `${matches.length} matches and ${games.length} games created for round ${roundNumber}`,
192          roundNumber,
193        }),
194        { headers: { "Content-Type": "application/json" } }
195      );
196    } catch (err) {
197      const errorMessage = err instanceof Error ? err.message : String(err);
198      return errorJson(errorMessage, 500);
199    }
200  });
```

KUVA 25. Paluuviesti

Mikäli funktio jossain kohtaa prosessia kohtaa virheen, kutsutaan apufunktiota errorJson, jolle annetaan viesti sekä http-status. Apufunktio muuttaa viestin ja statuksen JSON-muotoon, lisää HTTP-headerin ja palauttaa takaisin sovellukselle (kuva 26).

```
1 export function errorJson(message: string, status: number = 500): Response {
2   return new Response(JSON.stringify({ error: message }), {
3     status,
4     headers: { "Content-Type": "application/json" },
5   });
6 }
```

KUVA 26. errorJson-apufunktio

Tämä tarkoittaa sitä, että tulostaulukon kärjessä olevasta lähtien pelaaja paritetaan mahdollisimman lähellä pisteissä olevaan seuraavaan pelaajaan, jota vastaan pelaaja ei ole vielä pelannut. Tämä ei päde vielä ensimmäisellä kierroksella, vaan silloin pelaajat paritetaan satunnaisesti toisiaan vastaan.

4.5 Testaus

Sovellusta tehdessä testaus tapahtui pääsääntöisesti tekijän toimesta, jonka vastuulla oli manuaalisesti toteuttaa yksikkö ja integraatiotestaus ennen pull requestin tekemistä. Kaikki pull requestit katselmoitiin toisen tekijän toimesta, joka katselmoinnin yhteydessä myös toteutti yksikkö- ja integraatiotestausta. Aika-ajoin toteutettiin myös satunnaista end to end -testausta, jossa testattiin suurempia kokonaisuuksia ja kokonaisia toiminnallisuuksia, kuten prosessi turnauksen luonnista, sen hallittavuuteen ja näkyvyyteen muilla sovelluksen sivuilla.

Kaikkien turnausten haku

Tarkoituksena testata, että haettaessa kaikkia turnauksia, kaikkien turnauksen tiedot palautuvat, ja virhetilanteessa näytetään selkeä virhe. Seuraavassa taulussa on kuvattu yksikkö- ja integraatiotestit, jotka toteutettiin manuaalisena testauksena suoraan sovelluksen käyttöliittymän ja konsolin kautta.

TAULUKKO 1. Kaikkien turnausten tietokantahaun testitulokset.

Odotettu tulos	**Lopputulos**
Onnistui, kaikki turnaukset palautuvat	Ongelma: Alkuun turnaukset eivät palautuneet ollenkaan, tämä johtui tietokannan (RPC-asetuksista Row Level Security), joka esti taulun tietojenhaun vakiona kaikilta. Tämä korjattiin antamalla anon-roolille(kaikille), oikeus lukea taulun tietoja. Korjauksen jälkeen testi onnistui.
Onnistui, vastauksen rakenne on oikein	Onnistui, verrattiin tietokantarakenteeseen.
Onnistui, turnausten tiedot ovat oikein	Onnistui, verrattiin tietokantarakenteeseen.
Onnistui, turnauksen tiedot näkyvät sovelluksessa oikein.	Ongelma: Alkuun sovellus kaatui, eikä turnauksen tietoja saatu näkymään sivulla, vaikka tiedot palautuivatkin konsoliin oikein. Tämä johtui siitä, että tiedot haetaan asynkronisesti, ja tietojen paluuta ei odotettu sovelluksessa oikein, ja sovellus yritti renderöidä null:ia. Tämä korjattiin lisäämällä sivulle state ja ehto, jossa näytetään joko latauskomponentti, tai palautunut data, kunhan se oli vasta anotettu. Korjauksen jälkeen testi onnistui.

Odotettu tulos	Lopputulos
Virhe, jos tietokannan tunnistautumistiedot väärin.	Onnistui, testattiin muokkaamalla tietokannan tunnistautumistiedot vääriksi käyttäjäsalaisuuksiin.
Virhe, jos tietokantayhteyttä ei voitu muodostaa.	Onnistui, testattiin syöttämällä tietokannan osoitetiedot väärin käyttäjäsalaisuuksiin.

Yhden turnauksen haku käyttäen turnauksen uniikkia tunnistetta

Tarkoituksena testata, että haettaessa yhtä turnausta sen uniikin tunnisteen avulla, kyseisen turnauksen kaikki halutut tiedot palautuvat, tiedot ovat oikein sekä virhetilanteessa näytetään selkeä virhe. Seuraavassa taulukossa on kuvattu yksikkö- ja integraatiotestit, jotka toteutettiin manuaalisena testauksena sovelluksen käyttöliittymän ja konsolin kautta.

TAULUKKO 2. Yhden turnauksen haku käyttäen turnauksen uniikkia tunnistetta testitulokset.

Odotettu tulos	Lopputulos
Onnistui, odotettu turnaus palautui	Onnistui, varmistettiin konsolitulosteesta turnauksen uniikin tunnisteen ja muiden tietojen avulla.
Onnistui, vastauksen rakenne on oikein	Onnistui, verrattiin tietokantarakenteeseen.
Onnistui, turnauksen tiedot ovat oikein.	Onnistui, testattu sovelluksen käyttöliittymän ja konsolin kautta.
Onnistui, turnauksen tiedot näkyvät sovelluksessa odotetusti	Onnistui, verrattiin tietokantarakenteeseen.

Odotettu tulos	Lopputulos
Onnistui, sovelluksessa näytetään tyhjä sivu, mikäli uniikilla tunnisteella ei löytynyt yhtään turnausta.	Onnistui, sovelluksessa näkyi tyhjä sivu, kun haettiin olemattomalla tunnisteella turnausta.
Virhe, mikäli uniikki tunniste ei ole oikeassa muodossa, näytetään sovelluksessa toast, sekä siirrytään error-sivulle.	Onnistui, testattiin sovelluksessa.
Virhe, jos tietokannan tunnistautumistiedot väärin.	Onnistui, testattiin syöttämällä käyttäjäsalaisuuksiin virheelliset tunnistautumistiedot.
Virhe, jos tietokantayhteyttä ei voitu muodostaa.	Onnistui, testattiin syöttämällä käyttäjäsalaisuuksiin virheellinen osoite.

Yhden käyttäjän omistamien kaikkien turnauksen haku

Tarkoituksena testata, että haettaessa kaikkia kirjautuneen käyttäjän omistamia turnauksia, kaikkien käyttäjän omistamien turnausten tiedot palautuvat, tiedot ovat oikein, sekä virhetilanteessa näytetään selkeä virhe. Seuraavassa taulukossa on kuvattu yksikkö- ja integraatiotestit, jotka toteutettiin manuaalisena taestauksena sovelluksen käyttöliittymän ja konsolin kautta.

TAULUKKO 3. Yhden käyttäjän omistamien kaikkien turnausten haun testitulokset.

Odotettu tulos	Lopputulos
Onnistui, kaikki käyttäjän turnaukset palautuvat.	Onnistui, verrattiin tietokantarakenteeseen.
Onnistui, vastauksen rakenne on oikein.	Onnistui, varmistettiin selaimen konsolista.

Odotettu tulos	Lopputulos
Onnistui, turnausten tiedot ovat oikein.	Onnistui, verrattiin selaimen konsolisyötettä tietokantarakenteeseen.
Onnistui, annetulla käyttäjällä ei ole turnauksia.	Onnistui, verrattiin selaimen konsolisyötettä tietokantarakenteeseen.
Onnistui, turnausten tiedot näkyvät sovelluksessa odotetusti, jos turnauksia löytyi.	Onnistui, verrattiin tietokantarakenteeseen.
Onnistui, sovelluksessa näytetään teksti, ettei turnauksia ole, mikäli käyttäjällä ei löytynyt yhtään turnausta.	Onnistui, testattiin sovelluksessa.
Virhe, jos tietokannan tunnistautumistiedot väärin.	Onnistui, testattiin muokkaamalla tietokannan tunnistautumistiedot vääriksi käyttäjäsalaisuuksiin.
Virhe, jos tietokantayhteyttä ei voitu muodostaa.	Onnistui, testattiin syöttämällä tietokannan osoitetiedot väärin käyttäjäsalaisuuksiin.

Turnauksen lisäys

Testattiin, että turnauksen lisäys toimii oikein, ja tiedot siirtyvät sovelluksesta oikein tietokantaan oikeille ID:ille.

TAULUKKO 4. Turnauksen lisäyksen testitulokset.

Odotettu tulos	Lopputulos
Onnistui, turnaus luotu tietokantaan	Onnistui, verrattiin tietokantarakenteeseen.

Odotettu tulos	Lopputulos
Onnistui, oikeat tiedot löytyvät luodusta turnauksesta	Onnistui, varmistettiin tietokannasta sekä käyttöliittymästä.
Virhe, turnauksen nimi - kenttä on tyhjä	Onnistui, kehotetaan käyttäjää täyttämään tämä kenttä.
Virhe, jos tietokannan tunnistautumistiedot väärin.	Onnistui, testattiin muokkaamalla tietokannan tunnistautumistiedot vääriksi käyttäjäsalaisuuksiin.
Virhe, jos tietokantayhteyttä ei voitu muodostaa.	Onnistui, testattiin syöttämällä tietokannan osoitetiedot väärin käyttäjäsalaisuuksiin.

Turnauksen muokkaus

Tarkoituksena testata, kun syötetään muokkauslomakkeeseen uusia arvoja ja lähetetään se, että muutokset tulevat näkymään tietokannan lisäksi myös oikealle turnaussivulle.

TAULUKKO 5. Turnauksen muokkauksen testitulokset.

Odotettu tulos	Lopputulos
Onnistui, päivämäärän muutos	Onnistui, varmistettiin sekä tietokannasta, että turnaussivulta.
Onnistui, nimen muutos	Onnistui, varmistettiin sekä tietokannasta, että turnaussivulta.
Onnistui, formaatin muutos	Onnistui, varmistettiin sekä tietokannasta, että turnaussivulta.

Odotettu tulos	Lopputulos
Onnistui, pelin muutos	Onnistui, varmistettiin sekä tietokannasta, että turnaussivulta.
Virhe, jos tietokannan tunnistautumistiedot väärin.	Onnistui, testattiin muokkaamalla tietokannan tunnistautumistiedot vääriksi käyttäjäsalaisuuksiin.
Virhe, jos tietokantayhteyttä ei voitu muodostaa.	Onnistui, testattiin syöttämällä tietokannan osoitetiedot väärin käyttäjäsalaisuuksiin.

Turnauksen osallistujien haku

Tarkoituksena testata, kun haetaan funktiolla fetch_tournament_participants tietyllä turnauksen UUID:llä osallistujia, että tiedot tulevat oikein, ja tietokantaa vastaavina sovellukseen.

TAULUKKO 6. Turnauksen osallistujien haun testitulokset.

Odotettu tulos	Lopputulos
Onnistui, oikean turnausUUID:n pelaajat palautuivat	Onnistui, varmistettiin SQL-konsolista.
Onnistui, oikeat pelaajat palautuivat sovellukseen	Onnistui, varmistettiin selaimen konsolista.
Onnistui, tiedot näkyvät sovelluksessa oikein	Onnistui, tiedot näkyivat sovelluksessa oikein.
Virhe, jos tietokannan tunnistautumistiedot väärin.	Onnistui, testattiin muokkaamalla tietokannan tunnistautumistiedot vääriksi käyttäjäsalaisuuksiin.

Odotettu tulos	Lopputulos
Virhe, jos tietokantayhteyttä ei voitu muodostaa.	Onnistui, testattiin syöttämällä tietokannan osoitetiedot väärin käyttäjäsalaisuuksiin.

Pelaajan lisäys turnaukseen

Tarkoituksena testata, että pelaajan rekisteröityessä pelaajan tiedot menevät oikein sekä players-tauluun ja tournament_participants-tauluun. Testattiin myös, että tournament_participants-tauluun menee oikean turnauksen UUID. Testaus toteutettiin manuaalitestauksena luomalla uusia pelaajia, ja seuraamalla tietokannasta, että tiedot menevät oikein kumpaankin tauluun.

TAULUKKO 7. Pelaajan turnaukseen lisäämisen testitulokset.

Odotettu tulos	Lopputulos
Onnistui, tiedot löytyvät molemmista tauluista	Onnistui, varmistettiin kannasta.
Onnistui, turnausta vastaava UUID löytyy tournament_participants-taulusta	Onnistui, varmistettiin kannasta.
Virhe, jos tietokantayhteyttä ei voitu muodostaa	Onnistui, testattiin syöttämällä tietokannan osoitetiedot väärin käyttäjäsalaisuuksiin.

Turnauksen matsien haku

Tarkoituksena testata, että turnauksen matsien haku käyttämällä turnauksen UUID:tä palauttaa oikean turnauksen matsit, matsien tiedot ovat oikein, tiedot näkyvät oikein käyttöliittymässä, ja että virhetilanteessa näytetään selkeä virhe. Seuraavassa taulukossa on kuvattu yksikkö- ja integraatiotestit, jotka toteutettiin manuaalisena testauksena käyttäen Supabasen komentoriviä, sovelluksen konsolia ja sovelluksen käyttöliittymää.

TAULUKKO 8. Turnauksen matsien haun testitulokset.

Odotettu tulos	**Lopputulos**
Onnistui, turnauksen kaikki matsit palautuvat	Onnistui, varmistettu sekä Supabasen SQL-komentorivissä sekä sovelluksen konsolissa ja käyttöliittymässä.
Onnistui, matsien tiedot ovat oikein	Onnistui, varmistettu sekä Supabasen SQL-komentorivissä sekä sovelluksen konsolissa ja käyttöliittymässä.
Onnistui, matsien tiedot näkyvät sovelluksessa oikein	Onnistui, varmistettu käyttöliittymässä.
Onnistui, turnauksessa ei ole yhtään matsia	Onnistui, varmistettu käyttöliittymässä.
Virhe, jos tietokannan tunnistautumistiedot väärin.	Onnistui, testattiin muokkaamalla tietokannan tunnistautumistiedot vääriksi käyttäjäsalaisuuksiin.
Virhe, jos tietokantayhteyttä ei voitu muodostaa.	Onnistui, testattiin syöttämällä tietokannan osoitetiedot väärin käyttäjäsalaisuuksiin.

Matsien ja pelien generointi kierrokselle

Tarkoituksena testata, että matsien ja pelien luonti, sekä paritus toimivat oikein. Matsit tulee luotua jokaiselle parille, jokaiselle matsille luodaan kolme peliä, byet jaetaan oikein, ja virhetilanteet palauttavat selkeän virheen. Testaus toteutettiin manuaalitestauksena ensin Insomnia-työkalun avulla, ja varmistettiin toiminta uudelleen käyttöliittymän kautta, tietojen oikeellisuus katsottiin tietokannasta Supabasen työpöydän kautta.

TAULUKKO 9. Matsien ja pelien generoinnin kierrokselle testitulokset.

Odotettu tulos	**Lopputulos**
Onnistui, kaikki pelaajat paritettu onnistuneesti	Onnistui, testattiin käyttäen Insomnia-työkalua, tarkastelemalla Supabasesta tietokannan tietoja, sekä käyttöliittymän kautta.
Onnistui, pelaajat paritettu onnistuneesti, mutta yhdelle pelaajalle annetaan bye	Onnistui, testattiin käyttäen Insomnia-työkalua, tarkastelemalla Supabasesta tietokannan tietoja, sekä käyttöliittymän kautta.
Onnistui, onnistunut kutsu johtaa matsin ja kolmen pelin generointiin tietokantaan jokaiselle parille oikeilla tiedoilla	Onnistui, testattiin käyttäen Insomnia-työkalua, tarkastelemalla Supabasesta tietokannan tietoja, sekä käyttöliittymän kautta.
Onnistui, onnistuminen palauttaa sanoman, jossa kerrotaan luotujen matsien ja pelien määrä, sekä kierroksen numero	Ongelma: onnistumisviesti oli alkuun haastava saada käyttöliittymään käytettävässä muodossa. Tämä saatiin kuitenkin korjattua palauttamalla viesti JSON-muodossa. Lopulta onnistui, testattiin käyttäen Insomnia-työkalua ja käyttöliittymää.
Virhe, jos metodi ei ole HTTP POST	Onnistui, testattu käyttäen Insomnia-työkalua.

Odotettu tulos	Lopputulos
Virhe, jos Supabasen ympäristömuuttujat eivät ole oikein	Onnistui, testattu muokkaamalla edge-funktiolle annettuja supabasen ympäristömuuttujia, ja ajamalla edge-funktio Insomia-työkalun kautta.
Virhe, jos authorization tokenia ei ole saatavilla	Onnistui, testattu käyttämällä Insomnia-työkalua.
Virhe, jos authorization token ei ole validi	Onnistui, testattu käyttämällä Insomnia-työkalua.
Virhe, jos pakollinen parametri tournament_uuid puuttuu	Onnistui, testattu käyttämällä Insomnia-työkalua.
Virhe, jos käyttäjä ei ole turnauksen omistaja, tai turnausta ei löytynyt parametrilla	Onnistui, testattu käyttämällä Insomnia-työkalua.
Virhe, jos turnaukselle ei löydetty osallistujia	Onnistui, testattu käyttämällä Insomnia-työkalua.
Virhe, jos edellisten matsien haussa tapahtui virhe tietokannassa	Onnistui, testattu käyttämällä Insomnia-työkalua.
Virhe, jos kaikki mahdolliset kierrokset on jo luotu turnaukselle (round robin)	Onnistui, testattu käyttämällä Insomnia-työkalua.
Virhe, jos pelaajalle ei voitu antaa bye:tä(jos kaikki pelaajat ovat jo saaneet byen)	Onnistui, testattu käyttämällä Insomnia-työkalua.
Virhe, jos ei voitu parittaa kaikkia pelaajia	Onnistui, testattu käyttämällä Insomnia-työkalua.

Odotettu tulos	**Lopputulos**
Virhe, jos ei voitu luoda bye-matsia kierrokselle	Onnistui, testattu käyttämällä Insomnia-työkalua.
Virhe, jos matsien lisäyksessä tapahtui odottamaton virhe	Onnistui, testattu käyttämällä Insomnia-työkalua.
Virhe, jos pelien lisäyksessä tapahtui odottamaton virhe	Onnistui, testattu käyttämällä Insomnia-työkalua.
Virheen sattuessa virheviesti näytetään käyttäjälle käyttöliittymässä.	Ongelma: Tässäkin oli vaikeuksia saada paluuviesti muotoiltua käyttöliittymän käytettäväksi. Tämäkin saatiin ratkaistua muuttamalla paluuviesti JSON-muotoon. Lopulta onnistui, testattiin käyttöliittymän kautta.

5 POHDINTA

Projektin idea oli mielestämme hyvä ja kiinnostava, ja suunnittelimmekin alkuun hyvin innokkaasti ominaisuuksia ja mahdollisia jatkokehitysmahdollisuuksia. Hyvin pian alkuhuuman jälkeen työstäminen alkoi kuitenkin hidastua, kun ei täyden työpäivän päätteeksi jaksaminen ja aika tahtonut riittää sekä palautumiseen että projektin edistämiseen. Projekti edistyi kuitenkin pala palalta aina, kun aikaa ja jaksamista löytyi, kuitenkin hitaammin kuin olisi toivottu.

Toteutus kuitenkin sujui pääsääntöisesti sujuvasti ilman isompia kompastuskiviä, ja matka on ollut erittäin opettavainen. Olemme oppineet paljon käytetystä teknologiapinosta, josta molemmilla oli ennen projektia vaihtelevaa kokemusta. Next.js koettiin mielenkiintoisena vaihtoehtona, jonka ominaisuuksista pidimme erityisesti file-based routingista, koska se tuo selkeyttä projektin rakenteeseen, ja pakottaa pilkkomaan kokonaisuuksia pienempiin ja selkeisiin komponentteihin. Next.js:ssä taasen hitusen päänvaivaa aiheuttivat client- ja server -componenttien erot ja käyttötapaukset. Muutoin Next.js:llä kehittäminen tuntui melko samalta kuin Reactilla, josta molemmilla oli jonkin verran kokemusta aiemmin.

Supabase oli molemmille uusi tuttavuus, johon päädyttiin erään työkaverin suosituksen ansiosta. Supabase osoittautui hyväksi välimaastoksi, sillä molemmilla oli aiempaa kokemusta Googlen firebasesta tai MongoDB:stä. Molempia edellämainittuja siis käytetään frontendissä samaan tyyliin tietokantapalvelun tarjoaman automaattisen rajapinnan kautta kuten Supabasea, mutta niiden taustalla vaikuttaa document -based tietokanta, kun taas Supabase käyttää SQL:ää. Hieman vaikeuksia tietokannassa aiheuttikin nimenomaan SQL, jota täytyi opiskella ja debugata tehdessä melko paljon, vaikka peruskäyttö onkin toisen tekijän työn puolesta tuttua.

Git on mielestämme hieman vajavaisesti ja väärin käytetty edellisissä kouluprojekteissa, ja sen käyttö on tuntunut hieman komentorivin kautta käytetyltä pilvitallennustilalta. Git on kuitenkin toiselle tullut tutuksi työkokemuksesta, ja on auttanutkin molempia oppimaan sen ominaisuuksista erittäin paljon. Toisen kokemus on koettu tuovan sopivasti vaativuustasoa gitin käyttöön, sekä apu ja ohjeistus

on ollut helposti saatavilla. Feature-based commit -ratkaisu on ollut oppimisen kannalta yksinkertaisempaa ja selkeämpää, ja on auttanut ymmärtämään kokonaisuuden sekä eri komentojen tarkoitusta paremmin.

Koodaustyö lopetettiin hieman tavoiteltua aikaa jäljessä, ja hieman alkuperäisiä tavoitteita suppeampana. Alkuperäisenä tavoitteena oli saada swiss-turnauksen koko toiminnallisuus valmiiksi, mutta projektin aikataulun vuoksi päätimme katkaista koodaamisen päätoiminnallisuuksiin. Suunnitelluista ominaisuuksista toteuttamatta jäi top bracket-mahdollisuus. Etusivu jäi myös hieman yksinkertaiseksi, sille oli suunniteltu toteuttaa esimerkiksi suosituimpien turnausten esittelyä.

Jatkokehitysmahdollisuuksina on myös mietitty muun muassa formaattien kuten alussa mainitun GSL:n lisäämistä. Kirjautumattoman käyttäjän puolelle on myös mietitty parempaa liveseurantaa, jotta hän voisi seurata yksittäisten pelien tilannetta, kun tällä hetkellä sivulta näkee vain ottelun lopputuloksen. Olemme keskustelleet mahdollisuudesta jatkaa sovelluksen kehitystä myös tämän projektin jälkeen.

Jos sovellusta jatkokehitetään, ja se jossain vaiheessa saadaan yleiseen jakeluun valmiiksi, tulee tarve miettiä myös kustannuksia. Näitä tulee Supabasesta, sekä valitusta nettisivun isännöintipalvelusta esimerkiksi AWS tai Vercel.

Projektin tuloksena syntyi nettisivu, jota on hyvä lähteä jatkokehittämään. Sen aikana tuli opittua paljon Next.js:stä sekä Supabasesta. Gitin käytön kertaaminen ja syvempi käyttäminen oli myös mielekästä.

LÄHTEET

Chess.com 2025. What is a Swiss tournament? Luettavissa: https://support.chess.com/en/articles/8558054-what-is-a-swiss-tournament Luettu: 12.11.2025

Cloudinary.com 24.10.2025. What is server-side rendering? Luettavissa: https://cloudinary.com/guides/automatic-image-cropping/server-side-rendering-benefits-use-cases-and-best-practices Luettu: 10.11.2025.

Geeks for Geeks 22.7.2025i. Introduction to Tailwind CSS Luettavissa: https://www.geeksforgeeks.org/css/introduction-to-tailwind-css/ Luettu: 13.10.2025.

Geeks for Geeks 11.7.2025s. Static VS Dynamic Website Luettavissa: https://www.geeksforgeeks.org/websites-apps/static-vs-dynamic-website Luettu: 8.11.2025.

Geeks for Geeks 28.8.2025n. Next.js Components Luettavissa:https://www.geeksforgeeks.org/reactjs/next-js-components/ Luettu:11.11.2025.

TechTarget 24.7.2025. Insomnia vs. Postman: Comparing API management tools. Luettavissa: https://www.techtarget.com/searchapparchitecture/tip/A-quick-breakdown-of-Postman-vs-Insomnia. Luettu 4.12.2025.

Next.js 2025a. About React and Next.js. Luettavissa: https://nextjs.org/learn/react-foundations/what-is-react-and-nextjs. Luettu: 14.10.2025.

Next.js 2025b. Pages and Layouts Luettavissa: https://nextjs.org/docs/pages/building-your-application/routing/pages-and-layouts. Luettu: 3.11.2025.

Postgresql docs 14.10.2025. What Is PostgreSQL? Luettavissa: https://www.postgresql.org/docs/current/intro-whatis.html Luettu: 14.10.2025.

Supabase 14.10.2025. Architecture Luettavissa: https://supabase.com/docs/guides/getting-started/architecture Luettu: 14.10.2025.

Zod 2025. Intro Luettavissa: https://zod.dev/ Luettu 9.11.2025.