jamk

Sisällönhallintasovelluksen kehitys

Tokapeli Creator

Kristian Jeskanen

Opinnäytetyö, AMK
Joulukuu 2022
Tieto- ja viestintätekniikan koulutusohjelma

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Kuvailulehti

Jeskanen, Kristian

Sisällönhallintasovelluksen kehitys. Tokapeli Creator

Jyväskylä: Jyväskylän ammattikorkeakoulu. Joulukuu 2022, 50 sivua.

Tieto- ja viestintätekniikan tutkinto-ohjelma. Opinnäytetyö AMK.

Julkaisun kieli: suomi

Julkaisulupa avoimessa verkossa: kyllä

Tiivistelmä

Sisällönhallinalle on ollut tarve niin pitkään kuin ihminen on tuottanut sisältöjä, mutta tiedon siirtyessä yhä enemmän digitaalisiin ympäristöihin on tarve ratkaisuille kasvanut entisestään. Sisällönhallinalla tässä työssä viitataan toimintaan, jolla digitaalisia sisältöjä hallitaan. Erilaisia sisällönhallinnan ratkaisuja voidaan käyttää useaan tarkoitukseen kuten blogien, verkkosivujen ja sovellusten ylläpitoon.

Tehtävänä oli kehittää sisällönhallintasovellus Tokapeli-oppimisympäristön ylläpitoa varten. Tokapeli on kehitteillä oleva virtuaalinen oppimisympäristö, kriittisen lukutaidon ja luetun ymmärtämisen opettamiseen. Sen toiminta perustuu dynaamiseen arviointiin, yksinkertaisuuteen ja käyttäjien luoman sisällön esitystapaan. Sisältö voidaan esittää mm. pelillisyyden keinoin. Tokapelin toiminnallisuus nojaa täysin sisältöön, joten työkalu niiden hallintaa varten on välttämätön. Tavoitteena oli kehittää pilottiversio Tokapeli Creator -sovelluksesta, jolla käyttäjät voivat hallita ja julkaista sisältöjä, jakaa käyttö- ja lukuoikeuksia toisille käyttäjille ja tarkastella sisältöjen suorittamisesta kertynyttä dataa.

Sovelluksen toteutettiin moderneilla web-teknologioilla Full Stack -kehityksen periaatteella. Käyttöliittymä rakennettiin React.js-kirjastolla yhdessä Redux-tilanhallinkirjaston kanssa ja palvelinpuoli Django-verkkokehyksen ja PostgreSQL-tietokannan voimin.

Projekti saatettiin tavoitteiden valossa maaliin, mutta kehitettävää jäi paljon.

Avainsanat (asiasanat)

Sisällönhallinta, full stack -kehitys, full stack, web-kehitys, Django, React, Redux

Muut tiedot (salassa pidettävät liitteet)

-

jamk

Description

Jeskanen, Kristian

Development of a content management application. Comprehension Game Creator

Jyväskylä: JAMK University of Applied Sciences, December 2022, 50 pages.

Degree Programme in Information and Communication Technology. Bachelor´s thesis.

Permission for open access publication: Yes

Language of publication: Finnish

Abstract

A Content Management concept is as old as humans have been creating content, but when content is increasingly in digital environments, the need for innovative solutions is greater than ever. In the thesis content management refers to an action for managing content in digital form. Content Management solutions can be used for multiple purposes, like blogs, websites, and applications.

The task was to develop a content management application for a digital learning environment called Comprehension Game for teaching reading comprehension and critical literacy. It is a project in progress that is based on dynamic evaluation, simplicity, and content presentation. Its primary functionalities depend on the content created in it, so the application for managing that content is essential. The thesis's main goal was to develop a pilot version of the Comprehension Game Creator application that can be used for managing content and permissions and presenting content statistics.

The product was implemented as a full-stack project using modern web technologies like React.js and Redux for the frontend, and Django with PostgreSQL for the backend.

The project was finished from a thesis perspective, but it keeps on going still. A lot was implemented but there is still much to improve.

Keywords/tags (subjects)

Content management, full stack development, full stack, Django, React

Miscellaneous (Confidential information)

-

Sisältö

Kuviot

Taulukot

1. Johdanto

1.1 Taustat ja tavoitteet

Opinnäytetyössä keskitytään sisällönhallintaan ja tilastojen seurantaan käytettävän sovelluksen kehitysprosessiin moderneja web-teknologioita hyödyntäen. Toimeksiantajana on tuore Jyväskyläläinen CGWorld Learning Oy, joka on perustettu virtuaalisen oppimisympäristön kehitystyön ympärille kesäkuussa 2021. Yrityksen on perustanut Ekapelin isänäkin tunnettu Heikki Lyytinen ja tämän lisäksi työntekijöitä on 3, mukaan lukien tämän opinnäytteen antaja. Yritys kehittää uutta virtuaalista oppimisympäristöä, joka perustuu Lyytisen tekemään lukemisen tutkimukseen lähes 30 vuoden ajalta.

Tavoitteena oli siis kehittää sisällönhallintasovellus Tokapeli-oppimisympäristölle. Tokapeli on kehitteillä oleva virtuaalinen oppimisympäristö, kriittisen lukutaidon ja luetunymmärtämisen opettamiseen. Se perustuu dynaamiseen arviointiin, yksinkertaisuuteen ja käyttäjien luoman sisällön esitystapaan. Sisältö voidaan esittää mm. pelillisyyden keinoin. Tokapeli pyrkii tarjoamaan vaihtoehtoisen ja tehokkaan tavan opettaa kirjoitettua tietoa ja kehittää oppijan kriittistä lukutaitoa. Tokapeli soveltuu minkä tahansa kielisen sisällön opetukseen ja täten tähtää kansainvälisyyteen. Opetettava sisältö on kieliriippumatonta ja se koostuu metatiedosta sekä väittämistä. Sovelluksen käyttäjä suorittaa sisältöä eli opetuspaketteja käymällä läpi väittämäkokonaisuuksia, vastamaalla väittämään ”tosi” tai ”epätosi”. Opetuspaketti lasketaan läpäistyksi, kun kaikki valheelliset väittämät on tunnistettu. Oppimisympäristön toiminnallisuus nojaa täysin sen sisältöihin, jonka vuoksi sisällönhallintasovellus, Tokapeli Creator, on välttämätön.

Vaatimusmäärittely muuttui ja tarkentui Tokapelin kehittymisen myötä, mutta keskeisimmät ominaisuudet pysyvät sovelluksen toiminnan selkärankana. Tavoitteena oli, että sovelluksella voidaan luoda ja muokata Tokapelin sisältöjä, seurata käyttäjien edistymistä ja hallinnoida sisältökohtaisia oikeuksia. Yrityksen tavoitteena on kehittää Tokapeli-tuoteperheestä maailman laajuisesti käytössä oleva, ei-kielisidonnainen oppimisympäristö, mutta kehityksen alkuvaiheessa tuoteperhettä kehitetään pääsääntöisesti koulukäyttöön Suomessa ja Tokapeli Creatorin kohderyhmäksi rajattiin opettajat ja muu opetushenkilökunta. Keskeisenä kehityksessä huomioon otettuna ohjenuorana oli se, että sovelluksen käyttäjien tietotekninen osaamineen voi vaihdella suuresti.

1.2 Tutkimus

1.2.1 Tutkimuskysymykset

Tutkimuskysymyksiä ovat ”Kuinka toteuttaa vaatimusmäärittelyn mukainen tuote?” ja ”Mitä teknologioita ja työkaluja sisällönhallintasovelluksen kehittämiseen tulisi käyttää?”.

1.2.2 Tutkimusmenetelmä

Opinnäytetyö keskittyy tuotteen kehittämiseen ratkaisuksi toimeksiantajan ongelmaan ja sen tavoitteena on luoda konkreettinen tuotos, joten menetelmäksi soveltuu konstruktiivinen tutkimus. Edellä mainitulla menetelmällä pyritään käytännönläheiseen ongelmanratkaisuun, käyttämällä olemassa olevaa teoreettista tietoa. Konstruktiivisen tutkimuksen kautta toimeksiantaja saa puolueettoman ja teoreettiseen tietämykseen perustuvan ratkaisun ongelmaansa. Siinä on ominaista, että toimeksiantajan ja tutkimuksen toteuttajan välinen kommunikaatio on aktiivista ja, että toimeksiantaja on sitoutunut kehittämiseen. (Ojasalo, ym. 2015, 65–66.)

Tutkimusmenetelmän prosessi alkaa ongelman tunnistamisesta tai löytämisestä. Ennen ratkaisujen laatimista rakennetaan laaja teoreettinen tietopohja, jolla myöhemmin ratkaisun toimivuutta perustellaan. Ratkaisut pyritään testaamaan nopeasti, mahdollisesti esikokein ennen varsinaista testausta. Testauksien jälkeen osoitetaan uutuusarvo ja näytetään teoriakytkennät. Tutkimuksen lopussa tarkastellaan kehitetyn ratkaisun toimivuutta käytännössä. (Ojasalo, ym. 2015, 67.)

2. Tietoperusta

2.1 Sisällönhallinta

Sisällönhallinta (engl. Content Management) mielletään usein digitaaliajan ilmiöksi, vaikka se on ollut olemassa yhtä pitkään kuin olemme tuottaneet sisältöä, oli se sitten kirjoitettua, maalattua tai rakennettua. Aina on tarve uusille ratkaisuille sisällönhallintaan liittyen. Tässä yhteydessä sisällönhallinnalla viitataan toimintaa, jolla luodaan, hallitaan ja julkaistaan digitaalisia sisältöjä. Sisällöllä tarkoitetaan tietoa, jonka tuottaminen ja julkaisu on riippuvainen ihmisen toimista. Se voi olla esimerkiksi tekstiä, videoita tai ääntä. (Barker, 2016.)

2.2 React JS

React JS (React) on Metan (ent. Facebook) kehittämä moderni JavaScript kirjasto käyttöliittymien kehittämistä varten. React on ilmainen ja perustuu avoimeen lähdekoodiin. Sitä kirjoitetaan yleisesti JavaScript- ja JSX-syntaksilla, mutta enenemässä määrin käytössä on yleistynyt myös Microsoftin kehittämä TypeScript, joka toimii JavaScriptin yläjoukkona, lisäten mahdollisuuden staattiseen kielenkäyttöön. React käyttää deklaratiivista ohjelmointia eli selittävää ohjelmointia. Tällä ohjelmoinnin ajatusmallilla pyritään kertomaan mitä halutaan saavuttaa, kun taas vastakohtana pidetyssä imperatiivisessa ohjelmoinnissa kuvataan kuinka asiat toimivat vaihe vaiheelta. (Betrol 2017, 8; Roldan 2017, 19.)

Reactin toimintaperiaate perustuu kokonaisuuden jakamiseen pienemmiksi osiksi eli React-komponenteiksi, jotka muistuttavat paljon JavaScript-funktioita. Komponentit palauttavat React elementin, muuttumattoman objektin, joka kuvailee sen mitä Reactin tarvitsee piirtää. Komponentit ottavat vastaan ominaisuusobjekteja (props), joita käytetään tiedon jakamiseen komponenttien välillä. (Betrol 2017, 12–13; Components and Props n.d.) Komponenteille voidaan asettaa myös paikallinen tila. Tila on objekti, jota käytetään komponenttiin tehtyjen muutosten tallentamiseen. Muutokset tilassa aiheuttavat komponentin uudelleen renderöinnin. (Component State, n.d.)

React käyttää virtuaalista DOM-mallia. Se on ohjelmointi konsepti, jossa käyttöliittymä pidetään muistissa ja synkronoituna oikean DOM-puun kanssa. Käyttöliittymäkomponentin tilan muuttuessa React kokoaa muutoksen perusteella elementin ja päivittää sen virtuaaliseen DOM-puuhun. Kaikkien muutosten päivityttyä virtuaaliseen DOM-puuhun, React vertailee sitä selaimen DOM-puuhun ja päivittää ainoastaan muuttuneet kohdat. (Haberman 2014; Virtual DOM and Internals, n.d.)

React ei ole ”Model – View – Controller”-kehys (MVC), mutta yhdessä esimerkiksi Redux-kirjaston kanssa malli on hyvin saman kaltainen. (Baumgartner 2021; Hunt 2013)

2.3 React Router

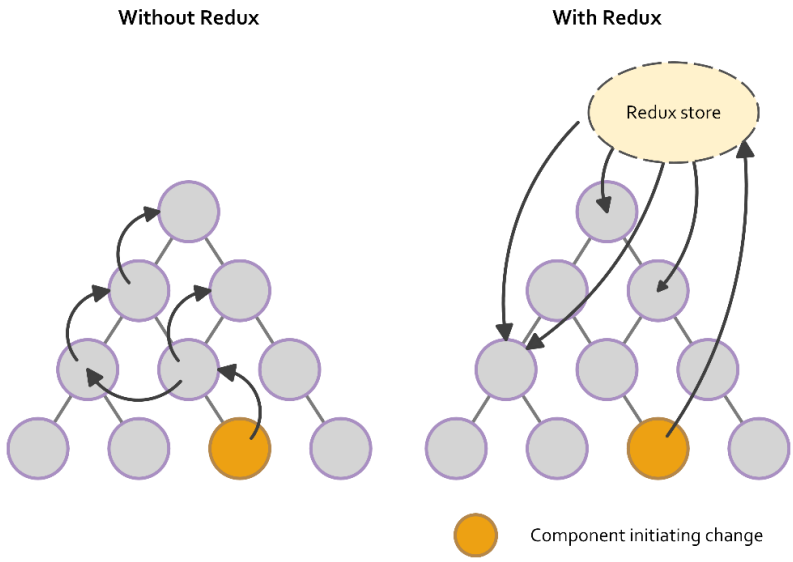
React Router on kolmannen osapuolen kehittämä laajalti käytössä oleva kirjasto, jonka avulla Reactiin saadaan ”Client-side”-reititysominaisuus. Tavallisen verkkosivun kohdalla selain pyytää

tiedoston palvelimelta, josta se renderöi sivun (HTML), CSS- ja JavaScript -määritykset huomioon ottaen. Joka kerta kun käyttäjä painaa linkkiä tai navigoi uuteen osoitteeseen, sama prosessi toistuu. ”Client side”-reitityksen myötä voidaan päivittää samaan tapaan URL-osoitetta siten, että palvelimelle ei tehdään joka kerta uutta kyselyä. (Feature overview n.d.)

2.4 Redux

Redux on tilanhallinta kirjasto JavaScript pohjaisille kirjastoille tai kehyksille kuten React, Angular tai Vue. Se koostuu tallennustilasta (*store*), toiminnoista (*actions*) ja *Reducer*-funktioista. Reduxin avulla luodaan koko sovellukselle globaali tila, joka säilötään tallennustilaan. Tila on kirjoitussuojattu, joten siihen voi tehdä muutoksi ainoastaan toimintojen avulla. Toiminnot ovat objekteja, jotka kuvaavat sitä millainen muutos tilaan tulisi tapahtua. Lopullisen muutoksen tilaan tekee toimintoja käsittelevä *Reducer*-funktio, joka ottaa argumentiksi nykyisen tilan ja palauttaa siitä, toiminnossa kuvattujen muutosten mukaisen uuden version. (Järvinen 2020.)

Kuten kohdassa 2.2 mainitaan, dataa React komponenttien välillä siirretään ominaisuusobjektien avulla ja muutokset voidaan tallentaa komponentin paikalliseen tilaan. Monimutkaisten sovellusten tilanhallinta Reactin oletuksena tarjoamilla toiminnoilla voi olla työlästä ja haastavaa. Reduxin avulla muutokset voidaan tehdä suoraan sovelluksen tilaan, jolloin kaikki komponentit pääsevät siihen käsiksi eikä ominaisuusobjektia tarvitse välttämättä käyttää. (Ks. Kuvio 1.)



Kuvio 1. Reactin tietovirta komponenttien välillä, ilman ja Reduxin kanssa. (Weck 2017.)

2.4.1 Redux Toolkit

Redux Toolkit on valinnainen työkalupakki Reduxin käyttöä varten. Se tavoittelee standardiksi ja on suositeltava tapa kirjoittaa Redux-logiikkaa. Redux Toolkit nopeuttaa Reduxin käyttöönottoa, tarjoa useita hyödyllisiä rajapintoja ja yksinkertaistaa sovellusten koodia. Se kehitettiin mm. helpottamaan tallennustilan asetusten määrittämistä ja vähentämään riippuvuuskirjastojen ja käytöstä johtuvaa kattilakoodin (engl. boiler code) määrää. Lisäksi työkalupakki tarjoaa useita helpottavia funktioita kuten *createAsyncThunk*-funktion, jonka avulla voidaan mm. kirjoittaa asynkronisten pyyntöjen palautustieto suoraan tallennustilaan. (Getting started with Redux Toolkit n.d.)

2.5 react-i18next

react-i18next on JavaScript-pohjaiseen i18next-kehykseen perustuva lokalisaatiokehys React- ja React Native -sovelluksille. Kehyksen avulla voidaan tunnistaa käyttäjän kieli, ladata sen pohjalta käännökset käyttöliittymään ja säilöä käännökset välimuistiin suorituskyvyn parantamiseksi. Lisäksi kehys tukee käännöksen hallintaan tarkoitettuja työkaluja ja palveluita kuten locize.com. Sovelluksen käännökset voidaan ladata paikallisesta JSON-tiedostosta (ks. Kuvio 2.) tai vaikka rajapintoja hyödyntäen ja koodissa *useTranslation*-hookkia käyttämällä (ks. Kuvio 3.). (Complete solution n.d; What is react-i18next? n.d.)

```
//translation file in the public/locales/en/translation.json
{
    "Welcome to React": "Tervetuloa Reactiin"
}
```

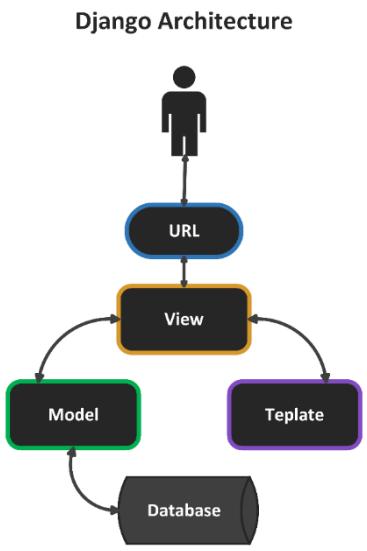
Kuvio 2. Käännöstiedosto JSON-formaatissa. (Step by step guide n.d.)

```
//i18next configuration
i18n
  .use(Backend)
  .use(LanguageDetector)
  .use(initReactI18next)
  .init({
    fallbackLng: 'en',
    debug: true,
});

//Use in the functional component.
Complexity is 3 Everything is cool!
function App() {
  const { t } = useTranslation();
  return <h2>{t('Welcome to React')}</h2>;
}
```

Kuvio 3. Käännösten käyttö komponentissa. (Step by step guide n.d.)

2.6 Django

Django on ilmainen ja avoimeen lähdekoodiin perustuva Python-pohjainen verkkokehys, joka perustuu ”Malli – Sapluuna – Näkymä”-arkkitehtuuriin (engl. Model – Template – View) (ks. Kuvio 4.). Malli viittaa Python-luokkina määritettyihin tietomalleihin eli tietokannan sisältöön. Tietokannan ja luokan välinen kommunikaatio tapahtuu Djangossa ORM- eli objektisuhdekartoitusteknologialla (engl. Object Relational Mapping). Näkymä taso määrittää sen mitä käyttäjälle esitetään ja sapluuna sen, miten. Djangon tavoitteena on tarjota helppo vaihtoehto komplekseja tietokantoja hyödyntävien verkkosivujen ja sovellusten kehittämiseen. Tämän tueksi Django tarjoaa laajan dokumentaation ja kattavasti ohjeistuksia. (Django introduction n.d; Django documentation n.d.)



Kuvio 4. Django arkkitehtuuri. (Ravindran 2018, 12.)

2.6.1 Django REST Framework

Django REST Framework (DRF) on joustava ja tehokas työkalu web-rajapintojen rakentamista varten. Se on helposti muokattava ja sisältää paljon kehittämistä helpottavia ominaisuuksia kuten Web Browsable API. Tämän ominaisuuden avulla kehittäjä pystyy helposti testaamaan rajapinnan toimintaa. Lisäksi työkalu tarjoaa mm. serialisoinnin, joka tukee tavallisia relaatiotietolähteitä ja ORM hyödyntäviä tietolähteitä. (Ravindran 2018, 167; Django REST Framework n.d.)

2.6.2 Django OAuth Toolkit

Django OAuth Toolkit on kolmannen osapuolen kehittämä kirjasto Djangolle. Sen avulla voidaan toteuttaa helposti OAuth 2.0 -standardin mukainen autorisointi eli valtuutus (Getting started n.d.). OAuth 2.0 on avoin standardi valtuutuksen toteutusta varten, jossa käyttäjä valtuuttaa kolmannen osapuolen sovelluksen käyttämään palvelimella olevia tietoja niin, että sille ei kuitenkaan toimiteta käyttäjän todennuksessa käyttämiä tietoja kuten salasanaa (What is OAuth 2.0? n.d.).

Django OAuth Toolkit tarjoaa valmiiksi tarvittavat päätepisteet, datan ja logiikan OAuth 2.0 -standardin mukaiseen autorisointiin. Kirjasto hyödyntää laajasti pythonin oauthlib-kehystä. Kirjaston avulla todennuksen voi toteuttaa usean tyypin mukaisesti, mutta dokumentaatiossaan se keskittyy kahteen yleisimpään tyyppiin, Authorization Code Grant ja Client Credential Grant. (Getting started n.d)

Authorization Code Grant -tyypillä tarkoitetaan valtuutusmalli, joka on yleisesti suosittu web- ja mobiilisovelluksissa, sen tietoturvatasosta johtuen. Mallissa rekisteröity asiakasohjelma vaihtaa väliaikaisen valtuutuskoodin (engl. Authorization code) käyttöoikeustietueeseen (engl. Access Token), valtuutuspalvelimen kanssa. Mallin etuna on se, että käyttöoikeustietue ei vaarannu vaihdon aikana. Kun käyttäjä on valtuuttanut sovelluksen, tämä ohjataan takaisin sovellukseen niin, että verkko-osoite sisältää väliaikaisen valtuutuskoodin. Sovellus vaihtaa koodin käyttöoikeustietueeseen niin, että autentikointiin käytetään asiakasohjelman Client ID:tä tai vapaaehtoista salaista koodia (engl. Client Secret). Tällöin käyttöoikeustietue ei koskaan ole näkyvissä selaimessa, eikä täten vaarassa paljastua väärinkäyttäjille. (Parecki 2018; Authorization Code Grant n.d.) Client Credential Grant -tyyppi taas on valtuutusmalli, jota käytetään silloin kun asiakasohjelman tarvitsee hakea resursseja palvelimelta ilman käyttäjän tekemää autentikointia (Client Credentials n.d.).

2.7 PostgreSQL

PostgreSQL on avoimen lähdekoodin relaatiotietokannan hallintajärjestelmä, joka käyttää SQL-kieltä ja sisältää useita ominaisuuksia datanhallinnan ja skaalaamisen helpottamiseksi. Se pohjautuu vuonna 1986, Kalifornian yliopistossa toteutettuun projektiin nimeltä POSTGRES. PostgreSQL mm. tukee useita tietotyyppejä, käyttäjän luomia tietotyyppejä ja relaatiomallista poikkeavaa dataa kuten JSON-formaattia, tarjoaa tehokkaan indeksointimetodin ja takaa datan eheyden. Yksi merkittävin ominaisuus PostgreSQL:ssä on Multiversion Concurrency Control eli MVCC, joka luo tietokantaa muokatessa käyttäjille omat tilannevedokset ja näyttää muutokset muille vasta kun tapahtuma on hyväksytysti valmis. (About n.d.)

2.8 Figma

Figma on web-grafiikan ja käyttöliittymien suunnittelutyökalu, jossa suunnittelu voidaan toteuttaa yhtäaikaisesti verkon kautta. Se on hyödyllinen työkalu koko kehitystiimille. Sen perusominaisuudet ovat ilmaiset rajallisille projektimäärille ja halutessaan ominaisuuksia saa lisää maksusta. Myös perusominaisuuksia voi laajentaa ilmaisilla lisäosilla. Figmalla käyttöliittymän voi suunnitella yksityiskohtaisesti ja maksimoida sovelluksen käyttäjäkokemusta luomalla siitä prototyypin, jonka testaaminen voidaan tehdä ilman erillistä ohjelmointia. Figma on alun perin selainpohjainen sovellus, mutta se tarjoaa myös työpöytäversion. (Husbands 2021)

3. Vaatimusmäärittely ja ominaisuudet

Tokapelin perustoiminnallisuus perustuu väittämäjaksojen esittämiseen käyttäjälle. Esitystapoja on useita, mutta sisällön formaatti pysyy samana. Sisällön suorittamisen kulku on seuraava: käyttäjät valitsevat haluamansa sisällön, halutessaan tarkastelevat tämän esitietoja ja materiaalia ja suorittavat sisällön vastamaalla sen väittämäjaksoihin. Väittämien välissä, sisällön määrityksistä riippuen, käyttäjälle näytetään selite, joka perustelee väittämän oikeaa vastausta. Käyttäjän vastausten perusteella kerätään dataa, joka kuvastaa sitä, kuinka tämä on suoriutunut.

Edellä mainitun kuvauksen pohjalta laadittu sisällönhallintasovelluksen ensimmäinen vaatimusmäärittely (ks. Taulukko 1.) oli projektin alkaessa hyvin tiivis ja suuripiirteinen, mutta se laajeni ja koki paljon muutoksia projektin aikana. Ydin vaatimusmäärittelyssä oli sisällön CRUD-toiminnot eli

sisällön lisääminen, lukeminen, päivittäminen ja poistaminen. Perustoiminnallisuuden lisäksi sovellusta tarvitaan Tokapeli-projektin tutkimusosuuden toteuttamiseen, joten siihen vaadittavat ominaisuudet päätyivät myös alkuperäiseen vaatimusmäärittelyyn. Sisältö, jota Tokapeli-oppimisympäristöön voidaan tuottaa voi vaihdella paljon aiheesta ja tekijästä riippuen, joten sen näkymistä käyttäjäkohtaisesti täytyy pystyä rajaamaan. Lisäksi ei voida olettaa, että sisältöä aina luotaisiin yksin, joten myös oikeuttaa muokata sisältöä pitää pystyä jakamaan. Toisin sanoen opetuspaketteja täytyy pystyä jakamaan.

Tokapeli tähtää kansainvälisyyteen, joten oli ilmeistä, että sovellukseen tulee toteuttaa lokalisaatio. Sisältö, jota sovelluksella hallitaan ei ole kieliriippuvaista, joten myös käyttöliittymän kielen vaihtamisen tulee olla mahdollista, joka edellyttää sen, että myös käyttöliittymän sisältö on käännettävissä valitulle kielelle.

Taulukko 1. Ensimmäinen vaatimusmäärittely

Ominaisuus	Tarkenne/Lisätiedot
Opetuspakettien lisääminen, poistaminen ja muokkaaminen	CRUD-toiminnot.
Lokalisaatio	Käyttöliittymästä voi vaihtaa kieltä, jolla sisältö näytetään.
Edistyksen seuraaminen	Tukimusta varten kerättävän tiedon näyttäminen käyttöliittymässä. Kuinka moni on pelannut? Kuinka pitkään kului opetuspaketin läpäisemisessä?
Opetuspakettien jakaminen	Kuinka käyttäjät saavat oikeuden nähdä sisältöjä Tokapelissä? Kuinka toinen käyttäjä voi muokata sitä sisällönhallintasovelluksessa?

3.1 Sisällönhallinta

Sisältö, jota Tokapeli-oppimisympäristössä tarvitsee hallita, on *opetuspaketit* eli niin sanottu oppimateriaali, jota käyttäjät oppimisympäristössä suorittavat. Sisällönhallinnan tarpeet koskevat opetuspakettien lisäämistä, poistamista, päivittämistä, tarkastelua ja kääntämistä toiselle kielelle. Lisäksi järjestelmällä hallitaan opetuspakettien luku, kirjoitus ja tarkastelu oikeuksia.

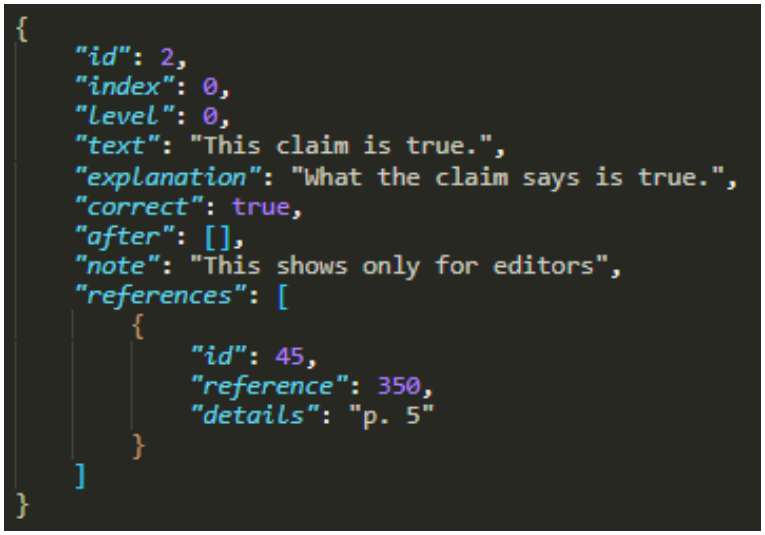
3.1.1 Opetuspaketti

Opetuspaketti on Tokapeli-oppimisympäristössä näytettävä ja käsiteltävä sisältö kokonaisuus, joka koostuu asetuksista, lähteistä, kielikäännöksistä eli kielipaketeista ja oikeuksista. Kielipaketilla tarkoitetaan opetuspaketin sisällön kielikäännöstä. Yhdellä opetuspaketilla voi olla usea kielipaketti, joka koostuu metadatasta, materiaalista ja väittämä taulusta. Teknisestä näkökulmasta opetuspaketti on JSON-formaatissa olevaa dataa (ks. Kuvio 5.), jonka mukaan Tokapeli-oppimisympäristön toimintaa ja käyttöliittymää muutetaan dynaamisesti.


```
{
    "url": "https://comprehensiongame.com/learning-packages/1/",
    "id": 1,
    "authors": [],
    "created": "2022-01-28T08:20:01.025974Z",
    "edited": "2022-04-27T08:43:00.605277Z",
    "tags": [],
    "references": [],
    "minimum_interval": 3,
    "repeat_condition": 1,
    "prune_condition": 3,
    "easy_mode_condition": 3,
    "easy_mode_length": 2,
    "explanation_condition": 3,
    "explanation_condition_correct": 1,
    "answers_end": 0,
    "rounds_end": 0,
    "correct_round_end": true,
    "consecutive_first_round": true,
    "give_immediate_feedback": true,
    "languages": [
        {
            "id": 1,
            "language": "eng",
            "title": "Title in English",
            "description": "Description in English.",
            "sentences": {
                "id": 1,
                "index": 0,
                "level": 0,
                "after": [],
                "material": null,
                "sentences": [
                    {
                        "id": 2,
                        "index": 0,
                        "level": 0,
                        "text": "This claim is true.",
                        "explanation": "What the claim says is true.",
                        "correct": true,
                        "after": [],
                        "note": "This shows only for editors",
                        "references": []
                    },
                    {
                        "id": 3,
                        "index": 1,
                        "Level": 0,
                        "text": "This claim is false.",
                        "explanation": "",
                        "correct": false,
                        "after": [],
                        "note": "This should be updated...",
                        "references": []
                    }
                ]
            }
        }
    ],
    "public_visibility": false,
    "permissions": [
        "edit",
        "view",
        "share"
    ]
}
```


Kuvio 5. Opetuspaketti JSON-formaatissa

Väittämällä (ks. Kuvio 6.) tarkoitetaan opetuspaketin aiheeseen liittyvää väitelausetta, joka on joko tosi tai epätosi. Sillä voi olla myös selite, jolla sitä voidaan halutessa tarkentaa oppijalle. Tokapelissä käyttäjä vastaa väittämiin jakso kerrallaan ja aina jakson läpäistyään tämä saavuttaa tallennuspisteen. Jaksoihin jakamiseksi väittämillä on *level*-arvo.



```
{
    "id": 2,
    "index": 0,
    "level": 0,
    "text": "This claim is true.",
    "explanation": "What the claim says is true.",
    "correct": true,
    "after": [],
    "note": "This shows only for editors",
    "references": [
        {
            "id": 45,
            "reference": 350,
            "details": "p. 5"
        }
    ]
}
```


Kuvio 6. Väittämä JSON-formaatiossa

3.2 Seuranta ja tilastot

Tuotteen kehittämisen, kannattavuuden ja konseptin todistamisen näkökulmasta on erityisen tärkeää kerätä dataa. Sovelluksesta kerättävää dataa käytetään tutkimukseen, sovelluksen kehittämiseen ja käyttäjäkokemuksen parantamiseen.

Seurannalla tarkoitetaan Tokapeli-tuotteen kohdalla käyttäjien opetuspakettikohtaista edistystä ja tilastoilla datan esittämistä käyttöliittymässä. Kerättyä dataa hyödynnetään opetuspakettien toimivuuden tarkkailussa, yksilöiden oppimisen seuraamisessa sekä kehityksen alkuvaiheessa toteutettavan tutkimustyön otantana.

3.3 Käyttötapaukset

3.3.1 Opetuspaketin luonti ja muokkaaminen

Opetuspakettien luominen ja muokkaaminen ovat sovelluksen yleisimmät käyttötapaukset. Voidakseen luoda ja/tai muokata sisältöjä käyttäjän täytyy olla rekisteröitynyt ja oikeutettu sisällön luomiseen. Pilotointi vaiheessa oikeudet jaetaan vain valituille yhteistyökumppaneille.

Muokkaamisessa käyttäjä valitsee muokattavan sisällön, tälle näytetystä listasta. Tarvittaessa sisältöä voi hakea ja/tai rajata käyttämällä hakukenttää. Kun sisältö on valittu, järjestelmä avaa muokkausnäkymän, jossa käyttäjä voi muuttaa opetuspaketin metadataa, väittämiä ja oikeuksia, muuttamalla eri kenttien arvoja. Tallentaakseen käyttäjän tulee painaa ”tallenna”-painiketta.

Uuden opetuspaketin lisääminen tapahtuu luontinäkymässä, jonka käyttäjä voi avata sovelluksen aloitusnäkymästä painamalla ”Lisää uusi”-painiketta. Järjestelmä avaa muokkausnäkymään tyhjän Opetuspaketti-mallin, jota täydentämällä voidaan luoda uusi opetuspaketti.

3.3.2 Käännöksen lisääminen

Käännös-ominaisuudella voidaan lisätä kielikäännös olemassa olevaan opetuspakettiin. Käännöksen lisäys tehdään muokkausnäkymässä. Järjestelmä ottaa alkuperäisestä kielikäännöksestä kopion, jonka metadataa ja väittämiä käyttäjä voi kääntää haluamalleen kielelle.

3.3.3 Oppimisen seuranta

Tilastot-ominaisuuden avulla käyttäjä, esimerkiksi luokan opettaja, voi seurata luomiansa opetuspaketteja suorittavien käyttäjien etenemistä ja tätä kautta arvioida paketin tehokkuutta ja yksilön kehitystä.

Tilastoja selataan valitsemalla ”Tilastot” sovelluksen valikosta. Näkymässä käyttäjä valitsee opetuspaketin, jonka tilastoja tämä tahtoo selata. Järjestelmä piirtää kaaviot valitun opetuspaketin statistiikan mukaan. Halutessaan käyttäjä voi rajata otantaa päivämäärän mukaan, valittuihin käyttäjiin ja/tai pelikertoihin. Tilastot voi tallentaa CSV-muotoon, jotta niitä voi viedä taulukkolaskentasovelluksiin.

3.3.4 Ryhmienhallinta

Ryhmät-ominaisuuden avulla voidaan luoda oikeusryhmiä, joiden avulla hallitaan kerralla useamman käyttäjän oikeuksia eri useisiin eri sisältöihin. Näkymä avataan sovelluksen valikosta kohdasta ”Ryhmät”. Käyttäjälle listataan ryhmät, johon tämä kuuluu ja joiden hallinnoija tämä on. Näkymässä voidaan muokata niitä ryhmiä, joiden hallinnoija kirjautunut käyttäjä on. Käyttäjä voi muokata ryhmän eli oikeusryhmän nimeä, kuvausta, lisätä ja poistaa hallinnoijia, jäseniä ja käyttöoikeuksia. Käyttöoikeuden kohdalla voidaan valita mitä opetuspakettia oikeus koskee ja mitkä oikeudet ryhmälle annetaan. Muokkaukset tallennetaan ”Tallenna”-painikkeesta. Uusille oikeusryhmän jäsenille lähetetään vahvistus sähköposti.

3.3.5 Asetusten muuttaminen

Asetuksista voidaan hallita omia käyttäjätietoja ja tilin alakäyttäjien tietoja. Näkymä avataan sovelluksen valikosta, kohdasta ”Asetukset”. Asetukset sivulta käyttäjä voi avata ”Tilin hallinta”-sivun, jossa voi hallinnoida omia tietojaan kuten, vaihtaa salasanaa, käyttäjätunnusta, nimeä ja sähköpostia. Lisäksi tätä kautta käyttäjä voi poistaa oman tilinsä ja ladata tästä kerätyt tiedot. Asetusten alakäyttäjien hallintaosion avulla käyttäjä voi lisätä ja poistaa alakäyttäjiä sekä muokata niiden tietoja kuten nimeä, PIN-koodia ja ajastimen näyttö asetusta.

4. Toteutus

4.1 Suunnittelu

Suunnittelu aloitettiin syksyllä 2021 Tokapelin kehityksessä heränneen tarpeen myötä. Ominaisuuksia ideoitiin useissa palavereissa ja Tokapelin sisältöjen rakenteen ollessa jo osittain tiedossa oli mahdollista luoda käyttötapaukset ja vaatimukset sisällönhallintasovelluksen pilottiversiolle. Sovellusta tultaisiin käyttämään tietokoneille, tableteilla ja jopa puhelimilla, joten nopeasti suunnittelu kääntyi Web-teknologioiden puoleen. Toteutus aloitettiin nopeasti ja sen suunnitelmat muuttuivat paljon lennosta, koska koko Tokapeli-konseptin kehitys oli vielä alkuvaiheessa. Kiireellisyyden vuoksi teknologiat päätettiin nopeasti ja ohjelmointi aloitettiin suunnittelun kanssa rinnakkain ja tavoitteena oli ottaa sovellus käyttöön heti kun perustoiminnallisuus on toteutettu. Nopealla ”testijulkaisulla” kehitystä nopeutettiin, sillä aloittaessa vaatimusmäärittely ei ollut täysin

valmis eikä kohderyhmä täysin lukossa. Lennosta testaamalla tahdottiin saada vastauksia käytettävyyteen ja ominaisuuksiin liittyviin kysymyksiin. Kohdassa 3 esitelty vaatimusmäärittely laajeni ja tarkentui mittavasti jo projektin ensimmäisten viikkojen aikana.

Sovelluksella tullaan tarkastelemaan, luomaan ja muokkaamaan dataa, joten oli ilmeistä, että sen toiminta vaatii palvelinpuolen ominaisuuksia, tietokantaa ja rajapintaa näiden välille. Sisältö tallennetaan tietokantaan, josta se haetaan ja näytetään Tokapelissä. Rajapinnan täytyy palvella siis kahta eri sovellusta. Tämän toteuttamiseksi päädyimme luomaan yhden palvelinpuolen ja käyttäjän, jonka avulla voidaan valtuutta kummatkin sovellukset käyttämään samaa rajapintaa.

Tilinhallinta lisättiin vaatimusmäärittelyyn käyttäjäkokemuksen parantamiseksi ja osittain vastaamaan EU:n tietosuoja-asetuksen vaatimukseen siitä, että käyttäjän tulee pystyä tarkistamaan hänestä tallennetut tiedot ja halutessaan poistamaan ne.

Opetuspakettien ja oikeuksien hallinta suunniteltiin toteutettavaksi dynaamisten lomakkeiden avulla. Oikeuksien myöntämisen pääteltiin olevan työlästä useampaan sisältöön ja useammalle käyttäjälle, mikäli se tehtäisiin jokaiseen haluamaansa sisältöön yksitellen, joten se suunniteltiin toteutettavaksi kahdella tavalla. Pienissä määrin oikeuksia voisi myöntää opetuspaketin lomakkeesta ja useampaan opetuspakettiin ja käyttäjille, erillisen lomakkeen kautta. Opetuspakettien muokkaamisen nopeuttamiseksi vaatimusmäärittelyyn lisättiin kopiointi ja korvaus ominaisuudet.

Väittämien tulee pohjautua faktaan ja niiden toden peräisyys tulee pystyä esittämään. Tätä varten opetuspaketteihin suunniteltiin lähdelista. Lähteisiin viitataan väittämää lisätessä.

Lokalisaation toteutus tarkentui suunnitellessa siten, että oletuksena käytettävä kieli asetetaan käyttäjän päätelaitteen kielestä. Mikäli kieltä kuitenkin tarvitsee vaihtaa sen voisi tehdä käyttöliittymästä. Toteutukseen käytettiin lokalisaatioon käytettävää kirjastoa react-i18next. Kirjaston käyttöön päädyttiin, koska sen avulla käyttäjän kielen tunnistaminen käy helposti ja käännökset voidaan toteuttaa JSON-tiedostoihin tai erillistä palvelua käyttäen.

Taulukko 2. Tarkennukset ja lisäykset vaatimusmäärittelyyn

Ominaisuus	Ominaisuuden laajennukset	Tarkenne/Lisätieto
Käyttäjät	Rekisteröinti	Käyttäjä rekisteröityy ja kirjautuu sovellukseen. Käyttäjää tarvitaan edistyksen seuraamiseen ja oikeuksien toteuttamiseen.
	Todennus	
	Käyttäjätilinhallinta	
Lokalisaatio	Käyttäjän kielen tunnistaminen	Oletuksena käytetään käyttäjän kieltä, mikäli sen käännös on olemassa. Käyttöliittymästä voi vaihtaa kieltä, jolla sisältö näytetään.
	Kielen voi vaihtaa käyttöliittymästä	
Edistyksen seuraaminen	Seurattavan sisällön valinta	Tukimusta varten kerättävä tieto voidaan piirtää kaavioiksi ja näytettävää dataa voi rajata.
	Seurattavien käyttäjien valinta	
	Kaavioiden näyttäminen käyttöliittymässä	
Opetuspakettien lisääminen ja muokkaaminen dynaamisella lomakkeella	Sisällön käännösten CRUD-toiminnot dynaamisella lomakkeella	Muokattavat sisällön osia, jotka ovat samat käännöksestä huolimatta: asetukset, oikeudet ja lähteet Käännöksen muokattavat osat: otsikko, kuvaus ja väittämät Väittämä koostuu tekstistä, selitteestä, vastauksesta ja lähdeviitteistä.
	Korvaa käännös toisella	
	Tuo käännös toisesta sisällöstä	
	Lisää sisältöön lähteitä ja viittaa niihin sisällön väittämissä	
	Kopio koko sisältö uudeksi	
	Korvaa koko sisältö toisella	
	Väittämien kopiointi uuteen tai olemassa olevaan sisältöön	
Opetuspakettien jakaminen	Sisällön oikeuksien CRUD-toiminnot	Käyttäjien sisältöjä koskevat oikeudet: luku, kirjoitus ja jakaminen Oikeuksia voi myöntää useammalla käyttäjälle ja useampaan sisältöön kerralla tai yksittäin. Alakäyttäjä luodaan osaksi pääkäyttäjä. Alakäyttäjä perii oikeudet sisältöjen käyttöön Tokapeli-sovelluksessa.
	Alakäyttäjien CRUD-toiminnot	
	Oikeusryhmien CRUD-toiminnot dynaamisella lomakkeella	

4.1.1 Teknologia valinnat

4.1.1.1 Palvelinpuoli

Tokapeliä ja Tokapeli Creatorin kehitys toteutettiin rinnakkain, joten palvelinpuoli toteutettiin palvelemaan kumpaakin sovellusta. Työssä esitellään kuitenkin ainoastaan niitä osia, jotka koskevat sisällönhallintasovelluksen toimintaa.

Palvelinpuolen teknologiaksi valittiin Django sen Python-perusteisuuden, sisäänrakennetun ylläpito käyttöliittymän ja laajojen sisällettyjen ominaisuuksien kuten todennuksen vuoksi. Sisällönhallintasovellusta kehittäessä on luontevaa käyttää REST rajapintaa, joten oli helppo valita DRF siihen tehtävään. DRF on suosittu ratkaisu rajapinnan toteuttamiseen Djangoa käyttäessä, koska se on yksinkertainen, joustava, tekee datan serialisoinnista helpompaa ja tukee sekä ORM, että non-ORM tietolähteitä.

Tietokannaksi päätyi PostgreSQL, koska se perustuu avoimeen lähdekoodiin ja se sisältää useita kehitystä helpottavia ominaisuuksia kuten tuen JSON-formaattiin ja käyttäjien luomiin tietotyyppeihin, se on skaalautuva ja tukee Multiversion Concurrency Control -ominaisuutta. Tietokannan rakenne ja tarpeet hallintajärjestelmältä eivät kehityksen alkuvaiheessa olleet vielä täysin varmoja, joten helposti skaalautuva ja muutoksia sietävä ratkaisu oli välttämätön. Kehitystiimi konsultoi myös asiantuntijaa, joka puhui teknologian puolesta.

Halusimme kokemattomina käyttää tietoturvan osalta turvalliseksi osoittautunutta menetelmää, joten päädyimme toteuttamaan valtuutuksen ja todennuksen OAuth 2.0 -standardin mukaisesti. Kehitettävä järjestelmä on web-sovellus, joiden kohdalla on yleistä käyttää edellä mainitun standardin mukaisia valtuutusvirtoja, kuten Authorization Code Grant. Tämän toteuttamiseksi Djangoon löytyi kolmannen osapuolen Django OAuth Toolkit -kirjasto. Kirjasto vähentää huomattavasti työmäärä ja on aktiivisesti ylläpidetty.

4.1.1.2 Miksi React?

Reactin käyttöön tässä projektissa päädyttiin sen joustavuuden, helppokäyttöisyyden ja laajan kehittäjäyhteisön sekä tehokkuuden vuoksi. Lisäksi kirjasto oli kehitystiimille ennestään tuttu. React

ei ole kehys vaan kirjasto, joka tekee sen opettelemisesta nopeaa ja sen käyttöön riittää pelkkä perustietopohja ohjelmoinnista. Komponenttiperusteinen lähestymistapa mahdollisti koodin uudelleen käytön sekä projektin sisäisesti, että useiden projektien välillä. Tällä periaatteella, komponenttien tilaa, ehdollista piirtämistä hyödyntäen voitiin helposti kehittää dynaamisen lomakkeen rakenne.

Prototyypin kehittämistä ajatellen on loogista, että koodista tehdään mahdollisimman hajautettua ja uudelleen käytettävää, jotta sitä voidaan hyödyntää mahdollisimman paljon myös tuotantoversiossa, vaikka sen kehitettävän sovelluksen rakenne muuttuisikin. Reactin valintaan vaikutti myös se, että sitä käytetään sen yleisyyden ja ominaisuuksien ansiosta pohjana uusille käyttöliittymäkehyksille. Tästä johtuen lähdekoodin kääntäminen tulevaisuudessa johonkin moderniin React-pohjaiseen kehykseen olisi helppoa.

4.1.1.3 Redux tilanhallinta ratkaisuna

Reduxin käyttöön päädyttiin, sillä se tarjoaa paljon ominaisuuksia ja mahdollisuuksia, joiden myötä sisällönhallintasovelluksen kehittäminen helpottuisi. Sisällönhallintasovelluksessa tietovirta voi kasvaa suureksi ja muutettavaa dataa on paljon kerrallaan. Reduxin avulla voidaan luoda globaali tila koko sovellukselle, jonne tuo data voidaan säilöä muutosten teon ajaksi ja tallentaminen tietokantaan voidaan tehdä vasta käyttäjän syötteestä tai muuten tarvittaessa. Sovelluksen tilaa ja komponentin tilaa hyödyntäen saadaan toteutettua esimerkiksi yksinkertainen muutosten vertailu.

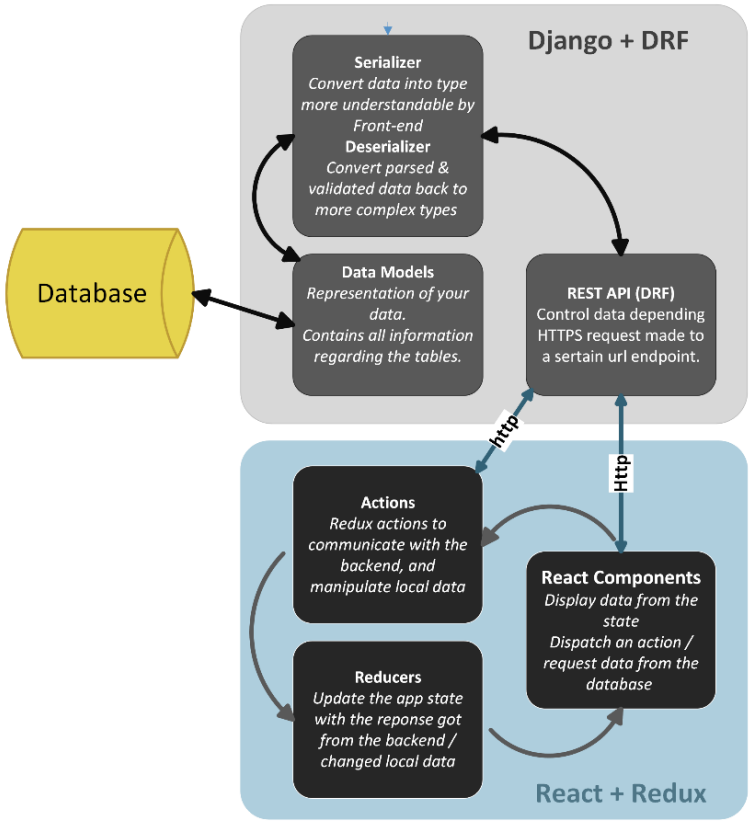
Reduxia yhdessä Redux Toolkitin kanssa käyttämällä, voidaan sovelluksen palvelinpyyntöjen logiikkaa toteuttaa suoraan siten, että palvelinpyynnöistä vastauksena saatu data voidaan tallentaa suoraan sovelluksen tilaan. Tämä voidaan toteuttaa Redux Toolkit tarjoaman *createAsyncThunk*-funktion (ks. Kuvio 7.) avulla. Tällä saadaan siirrettyä logiikkaa komponenttien ulkopuolelle ja pidettyä koodia siistinä.

```
//Create an asyncThunk
export const fetchPackages = createAsyncThunk(
  'learningPackages/fetch',
  Complexity is 9 It's time to do something...
  async ( ..., thunkAPI) => {
    try {
      const res = await tokapeliApi.listLearningPackages();
      if (!res.errorCode) {
        return res.data;
      } else {
       throw res;
      }
    } catch (e: any) {
      return thunkAPI.rejectWithValue({ errorInfo: e.response.data });
    }
});

//Create an initial state
const initialState = {
  packages: [],
} as LearningPackagesState

//Create a state slice
const learningPackagesSlice = createSlice({
  name: 'learningPackages',
  initialState,
  reducers: {
  },
  extraReducers: (builder) => {
    builder.addCase(fetchPackages.fulfilled, (state, action) => {
      state.packages = action.payload;
    })
  },
})

//Fetch packages by dispatching action generated by createAsyncThunk
dispatch(fetchPackages(1))
```

Kuvio 7. Esimerkki *createAsyncThunk*-funktion käytöstä

4.1.2 Rakenne

Tietovirta sovelluksessa suunniteltiin hyvin tavallisen web-sovelluksen tavoin (ks. Kuvio 8.). Käyttäjän syöte käyttöliittymässä laukaisee jonkun HTTP-pyynnön REST-rajapintaan, joka käsittelee kutsun ja vastaa pyynnön mukaisesti. Osa pyynnöistä tapahtuu suoraan komponentista ja osa toimintojen (action) kautta. Toimintojen kautta pyyntöjen vastauksien tuomat muutokset voidaan suoraan tarjota *Reducer*-funktiolle, joka päivittää sen sovellukseen tilaan. Näin ei tarvitse kutsua aina uutta toimintoa tilan päivittämiseksi. Ennen tiedon lähettämistä tietokannasta käyttöliittymään se serialisoidaan DRF:n tarjoamilla ominaisuuksilla eli tässä tilanteessa muunnetaan JSON-formaattiin, jotta JavaScript ymmärtää sen. Sama tehdään myös käyttöliittymästä lähetettävälle tiedolle, mutta se jäsennetään takaisin kompleksisemmiksi tietotyypeiksi eli tietokannan ymmärtämäksi objektiksi. Tätä kutsutaan deserialisoinniksi. Ennen tietokantaan kirjoittamista, käyttöliittymästä lähetettävä data validoidaan ja deserialisoidaan eheyden varmistamiseksi.



Kuvio 8. Sovelluksen tietovirta

4.2 Palvelinpuoli

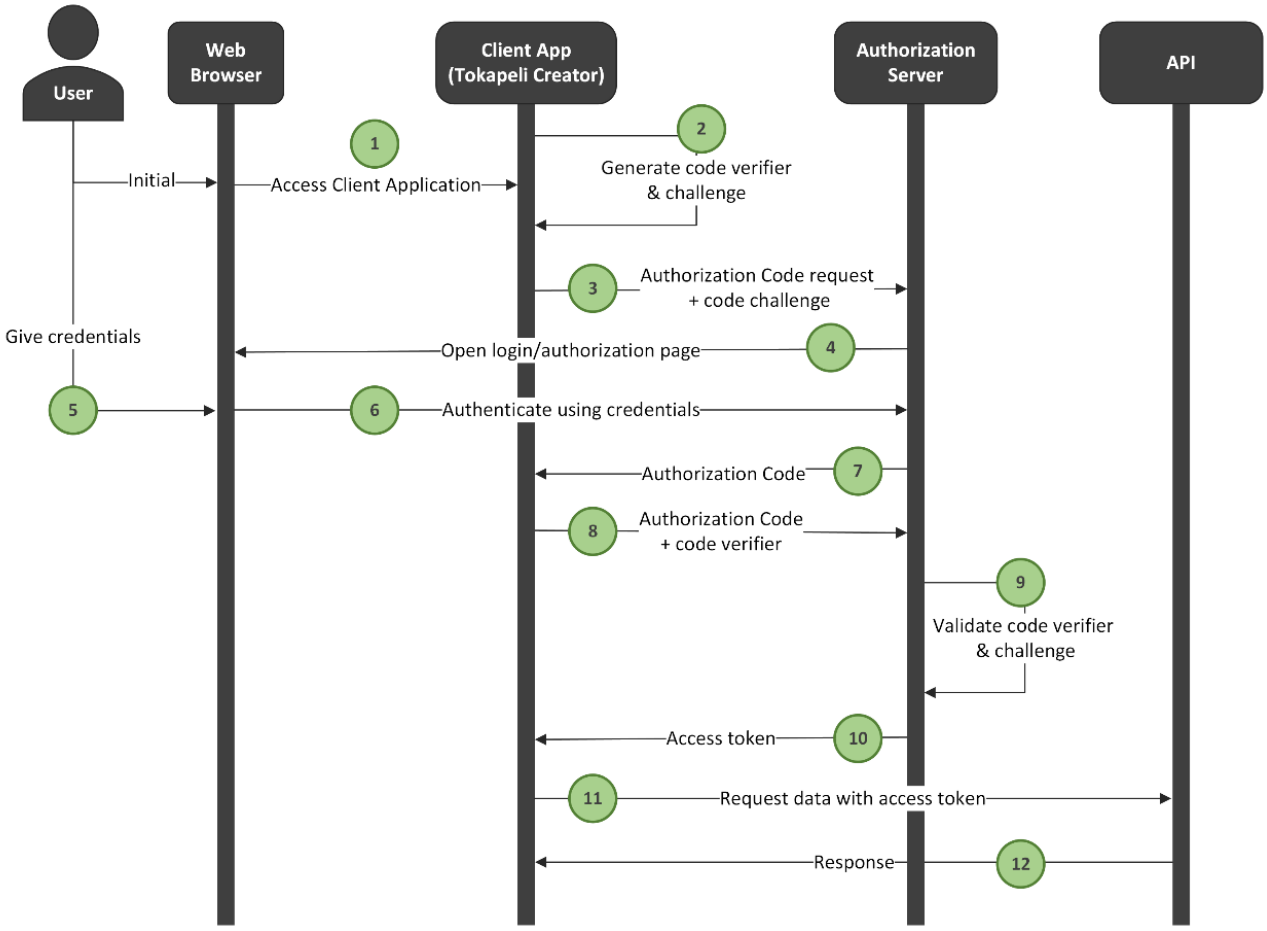
Tämä osio syventyy projektin palvelinpuoleen. Palvelinpuoli on vastuussa mm. todennuksesta ja sovelluksen ja tietokannan välisestä tietoliikenteestä eli sitä voidaan kuvata kerroksena, joka ei näy käyttäjälle. Kehityksessä palvelinpuolta ajetaan paikallisella laitteella sisäisessä verkossa, mutta tuotannossa se julkaistaan virtuaalipalvelimella yrityksen verkkotunnuksen comprehension-game.com alla.

4.2.1 Virtuaalipalvelin

Sovelluksen palvelinpuoli on julkaistu Jyväskyläläisen IT-ihme yrityksen tarjoamalla Debian-pohjaisella virtuaalipalvelimella. Virtuaalipalvelimelle on asetettu SSH-yhteys, jonka kautta tiedonsiirto ja hallinta tehdään. Django-palvelin ja sovelluksen testaukseen käytettävät tuotantoversio ovat julkaistu Apache HTTP-palvelinohjelmaa käyttäen.

4.2.2 Todennus

Todennusta suunnitellessa oli selvää, että tietoturvasta ei tingitä. Sovelluksella on tarkoituksena hallita tulevaisuudessa suuria määriä dataa, jonka vuoksi sen suojaaminen käyttöoikeuksin on tärkeää. Todennus tahdottiin toteuttaa standardien ja hyvän käytännön mukaisesti käyttämällä OAuth 2.0 -standardia. Django OAuth Toolkitin avulla saatiin helposti toteutettua standardin mukainen Authorization Code Grant -valtuutusvirta sovelluksen ja palvelimen välille (ks. Kuvio 9.). Käyttäjä ohjataan sovelluksesta valtuutuspalvelimella sijaitsevalle kirjautumissivulle, joka todennuksen onnistuttua ohjaa käyttäjän takaisin sovellukseen ja valtuuttaa sovelluksen käyttämään rajapintaa. Käyttäjän todennukseen käyttämät tiedot kuten salasana ei ole sovelluksen saatavilla. Valtuutuksella ja todennuksella saadaan mm. suojattua tietokantaa väärinkäytöksiltä.



Kuvio 9. Authorization Flow

4.2.3 REST rajapinta

REST-rajapinta luotiin todennusta, käyttäjien, opetuspakettien ja käyttöoikeuksien hallintaa varten. Django REST Frameworkin avulla rajapinnan reittien (ks. Taulukot 3, 4 ja 5) luonti onnistui helposti, sillä se tarjoaa valmiiksi kirjautumiseen käytettäviä malleja, jonka ansiosta kirjautumis- ja rekisteröintisivujen luonti oli nopeaa. Reitit toimivat päätepisteinä joihin sovelluksesta tehdään HTTP-pyyntöjä. Rajapinta käsittelee käyttöliittymästä päätepisteeseen tehdyn pyynnön, tekee tietokanta toimenpiteen ja lähettää määritysten mukaisen vastauksen taikaisin käyttöliittymään. Päätepisteissä osat, jotka on kirjoitettu <>-merkkien sisäpuolelle, kuvastavat verkkosoitteen dynaamista osaa. Tauluissa ei esitellä valtuutukseen käytettäviä päätepisteitä, sillä ne ovat Django OAuth Toolkit-kirjaston määrittämiä ja täten helposti nähtävissä kirjaston dokumentaatiossa.

Taulukko 3. Todennukseen ja käyttäjähallinnan tarvittavat päätepisteet rajapinnassa

Päätepiste ja parametrit	HTTP metodit	Käyttötarkoitus
register/	GET, POST	Käyttäjän lisääminen.
register/confirm/<str:key>/	GET	Käyttäjän sähköpostin vahvistaminen
resend_activation_email/	GET, POST	Vahvistus sähköpostin uudelleen lähettäminen
login/	GET, POST	Autentikointi ja valtuutus
logout/	GET,	Autentikoinnin ja valtuutuksen poistaminen.
password-change/	GET, POST	Salasanan vaihtaminen.
password-change/done/	GET	Ilmoitetaan salasanan vaihtamisen onnistumisesta.
Password-reset/	GET, POST	Salasanan nollaaminen, jos se on unohtunut.
password-reset/done/	GET	Ilmoitetaan salasanan resetointi sähköpostin lähettämisestä.
password-reset/<uidb64>/<token>/	GET	Todennetulle käyttäjälle salasanan vaihto lomakkeen hakeminen.
password-reset/<uidb64>/set-password	GET, POST	Uuden salasanan asettaminen.
password-reset/complete/	GET	Ilmoitetaan salasanan resetoinnin onnistumisesta.
change-email/	GET, POST	Sähköpostin vaihtaminen.
change-email/cancel/<str:key>/	GET	Sähköpostin vaihdon peruuttaminen.
change-email/confirm/<str:key>/	GET	Ilmoitetaan sähköpostin vaihdon onnistumisesta.
account/	GET, POST	Listataan käyttäjätilin hallintaan liittyvät ominaisuudet. Poistetaan tili.

Taulukko 4. Opetuspakettien ja oikeusien hallintaan tarvittavat päätepiste rajapinnassa

Päätepiste **ja parametrit**	**HTTP metodit**	**Käyttötarkoitus**
learning-packages/	GET	Opetuspakettien hakeminen.
learning-packages/<int:id>/	GET, POST, PATCH, DELETE	Opetuspaketin hakeminen, lisääminen, poistaminen ja muokkaaminen.
learning-package-users/<int:id>/	GET, POST	Opetuspakettiin oikeutettujen käyttäjätietojen hakeminen ja näiden oikeuksien muuttaminen ja poistaminen.
learning-package-permissions/	GET, POST	Opetuspaketin oikeuksien jakopyyntöjen hakeminen ja lisäys.
learning-package-permissions/<int:id>/	GET, POST	Opetuspaketin oikeuden jakopyynnön hakeminen ja poistaminen.
permission-groups/	GET, POST	Oikeusryhmien hakeminen ja lisääminen.
permission-groups/<int:id>/	GET, PATCH, DELETE	Oikeusryhmän hakeminen, muokkaus ja poistaminen.
group-invites/confirm/<str:key>/	GET	Kutsuttu käyttäjä saa linkin sähköpostiin, jonka avaamalla kutsu hyväksytään.

Taulukko 5. Tilastojen hakuun ja oikeuksien hallintaan tarvittavat päätepisteet rajapinnassa

Endpoint ja parametrit	**HTTP metodit**	**Käyttötarkoitus**
research/user/	GET	Niiden käyttäjätietojen hakeminen, jotka ovat oikeuttaneet kutsun tehneen käyttäjän.
research/games/	GET	Käyttäjien opetuspaketissa etenemiseen liittyvien tilastojen hakeminen.
research/stats/	GET	Käyttäjän opetuspakettikohtaisen tilaston hakeminen.
research/permissions/	GET, POST	Kirjautuneen käyttäjän pelidatan lukuoikeuksien omaavien käyttäjien hakeminen ja oikeuksien muokkaaminen.

Django REST Frameworkissä (DRF) päätepisteet määritetään *urls.py* tiedostoon (ks. Kuvio 10.). Päätepisteitä määritettiin kahdella tavalla, käyttämällä *DefaultRouter*-reititintä ja lisäämällä pääte piste suoraan Djangon *urlspatterns*-taulukkoon. Päätepisteille määritettiin oma *view*-funktio, joko käyttämällä Djangon *View*-luokkaa tai DRF:n *ViewSet*-luokkaa. *ViewSet*-luokkaa käyttämällä saatiin automaattisesti käyttöön *list*, *create*, *retrieve*, *update*, *partial-update* ja *destroy* toiminnot, kun taas *View*-luokka toimi ainoastaan pyyntöjen käsittelijänä. Määritetyssä *view*-funktiossa toteutettiin pyynnön vaatima logiikka ja datan serialisointi, johon käytettiin DRF:n Serializer-luokkia, koska ne hoitavat myös datan validoinnin. Esimerkki päätepisteen ”learning-packages” logiikka on kirjoitettu *LearningPackageViewSet*-luokassa (ks. Kuvio 11.), jossa serialisointi toteutetaan riippuen päätepisteeseen tehdystä pyynnöstä. Esimerkiksi GET-pyynnön kohdalla käytetään *LearningPackageListSerializer*-luokkaa (ks. Kuvio 12.).

```
router = routers.DefaultRouter()
router.register(r'learning-packages', LearningPackageViewSet, basename='learningpackage')
router.register(r'learning-package-permissions',
    LearningPackagePermissionsViewSet, basename='learningpackagepermissions'
)
router.register(r'user', UserInfoViewSet, basename='user')
router.register(r'permission-groups', OwnedGroupView, basename='permissiongroups')
router.register(r'group-invites', GroupInviteViewSet, basename='groupinvites')

urlpatterns = [
    path('login/', auth_views.LoginView.as_view(), name='login'),
    path('logout/', CustomLogoutView.as_view(), name='logout'),
    path('register/', UserRegistrationView.as_view(), name='register'),
    path('register/confirm/<str:key>/', registration_confirm_view, name='register_confirm'),
    path('resend_activation_email/',
         ResendUserActivationEmail.as_view(), name='resend_activation_email'),
    path('account/', account_view, name='account'),
    path('password-change/', CustomPasswordChangeView.as_view(), name='password_change'),
    path('password-change/done/',
         auth_views.PasswordChangeDoneView.as_view(), name='password_change_done'
    ),
    path( 'password-reset/<uidb64>/<token>/',
         auth_views.PasswordResetConfirmView.as_view(), name='password_reset_confirm'
    ),
    path('password-reset/complete/',
        auth_views.PasswordResetCompleteView.as_view(), name='password_reset_complete'
    ),
    path('password-reset/',
         auth_views.PasswordResetView.as_view(), name='password_reset'
    ),
    path('password-reset/done/',
        auth_views.PasswordResetDoneView.as_view(), name='password_reset_done'
    ),
    path('change-email/', ChangeEmailView.as_view(), name='change_email'),
    path('change-email/confirm/<str:key>/', email_confirm_view, name='email_confirm'),
    path('change-email/cancel/<str:key>/',
        email_change_cancel_view, name='email_change_cancel'
    ),
    path('learning-package-users/<int:package_id>/',
        LearningPackagePermissionGroupsView.as_view()
    ),
    path('research/stats/', ResearchStatsView.as_view()),
    path('research/games/', ResearchGamesView.as_view()),
    path('research/user/', ResearchUserView.as_view()),
    path('research/permissions/', ResearchModifyPermissionView.as_view()),
    path('groups-permissions/',
        user_groups_and_permissions_view, name='user_groups_and_permissions'
    ),
    path('group-invites/confirm/<str:key>/',
        group_invite_confirm_view, name='group_invite_confirm'
    ),
    path('', include(router.urls)),
]
```

Kuvio 10. Päätepisteiden määrittäminen

```python
class LearningPackageViewSet(AutoPermissionViewSetMixin, viewsets.ModelViewSet):
    http_method_names = ['get', 'post', 'patch', 'delete']
    serializer_class = LearningPackageListSerializer
    permission_classes = []

    def get_queryset(self):
        return get_permitted_learning_packages(
            self.request.user,
            permissions=self.request.query_params.getlist('permissions')
        )

    # show data differently when retrieving all objects or just one
    def get_serializer_class(self):
        if self.action == 'list':
            return LearningPackageListSerializer
        if self.action == 'retrieve':
            return LearningPackageRetrieveSerializer
        if self.action == 'create':
            return LearningPackagePostSerializer
        if self.action == 'partial_update':
            return LearningPackagePostSerializer
        raise MethodNotAllowed(self.request.method)

    def partial_update(self, request, *args, **kwargs):
        return self.update(request, *args, **kwargs)

    def perform_destroy(self, instance):
        delete_learning_package(instance)
```

Kuvio 11. *LearningPackageViewSet*-luokan määrittäminen

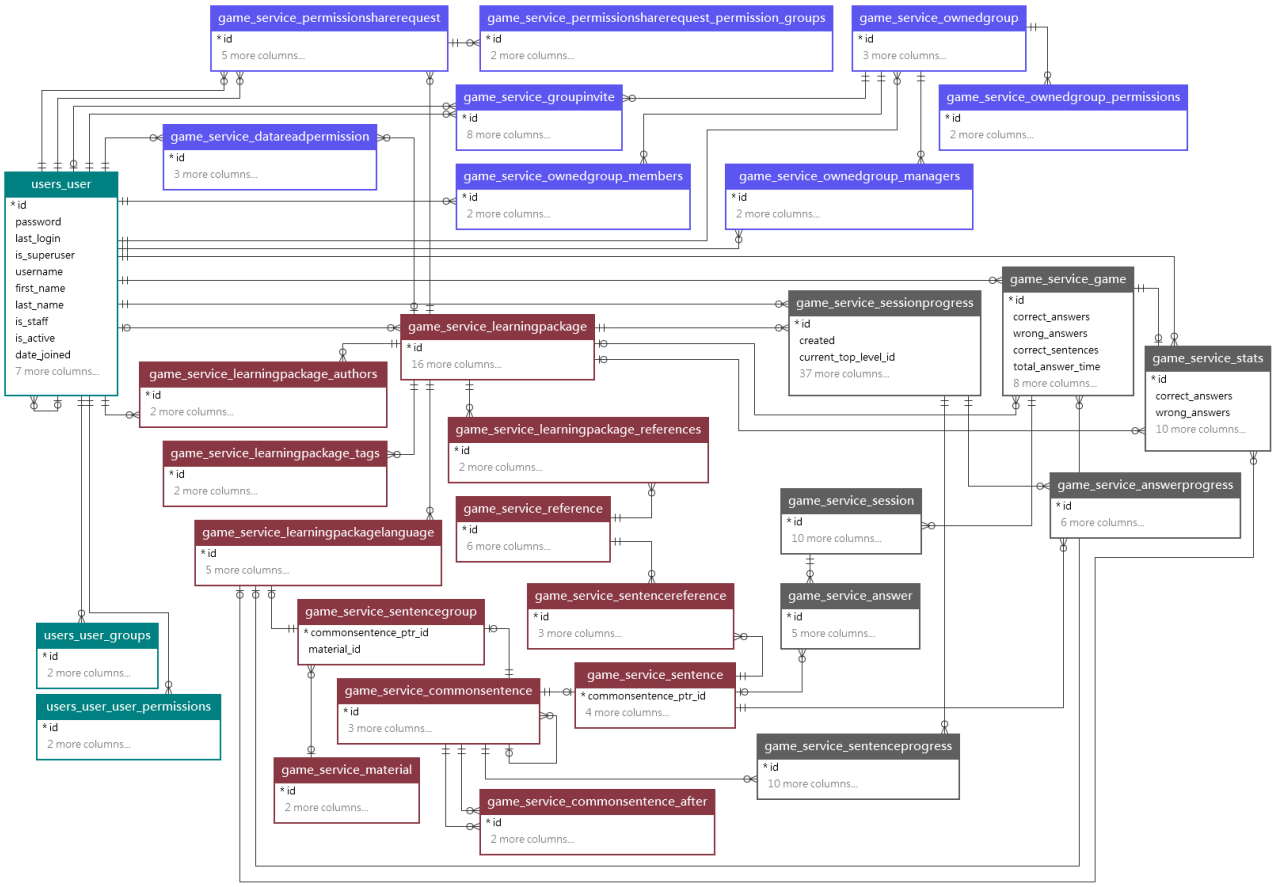
```python
class LearningPackageListSerializer(serializers.HyperlinkedModelSerializer):
    authors = UserSerializer(many=True)
    tags = TagSerializer(many=True)
    languages = LearningPackageLanguageListSerializer(many=True)

    class Meta:
        model = LearningPackage
        fields = [
            'url',
            'id',
            'authors',
            'tags',
            'created',
            'edited',
            'languages',
            'public_visibility',
        ]

    def to_representation(self, instance):
        reprensentation = super().to_representation(instance)
        reprensentation['permissions'] = get_user_permissions_for_learning_package(
            self.context['request'].user, instance.pk
        )
        return reprensentation
```

Kuvio 12. *LearningPackageListSerializer*-luokka

4.2.4 Tietokanta

Tietokanta toimii sovelluksen säilönä ja olennaisena osana koko sovelluksen päätoiminnallisuutta. Käyttäjädatan säilyttämisen lisäksi sinne tullaan tallentamaan opetuspaketit, käyttäjien pelidata ja oikeusryhmät. Tietokannan taulut määriteltiin Django *Model*-luokkina, joissa määritettiin relaatiot

ja kenttien hyväksymät tyypit. Django *Model* toimii taulun mallina, jota käytetään mm. datan validointiin. Tietokannan taulut voi karkeasti jakaa neljään osaan, joka käy hyvin ilmi relaatiokaaviosta (ks. Kuvio 13.), jossa vihreät taulut kuvastavat käyttäjää, punaiset opetuspakettia, violetit oikeuksia ja harmaat tilastoihin tarvittavaa dataa.



Kuvio 13. Tietokannan relaatiokaavio

4.3 Käyttöliittymä

Käyttöliittymän kehitys aloitettiin vaatimusmäärittelyn mukaisen rautalankamallin (ks. Kuvio 14.) toteutuksella, jonka tekemiseen käytettiin Figma-palvelua. Rautalankamallit toteutettiin mahdollisimman pelkistetysti ja niillä pyrittiin luomaan kehittäjätiimille riittävä kuva navigaatiorakenteesta ja siitä minkälaisia komponentteja sovelluksessa tulee olemaan. Figman rautalankamallia käytettiin referenssinä kehitettävälle sovellukselle ja käyttöliittymä ohjelmoitiin sen pohjalta. Lisäksi Figman avulla toteutettiin värisuunnittelua ja elementtien sommittelua.

Kuvio 14. Rautalankamalli

Käyttöliittymä toteutettiin React-kirjastolla ja sen osat layouteilla eli leiskoilla (ks. Kuvio 15.), joissa hyödynnettiin React Router-kirjaston *Outlet*-komponenttia. Outlet-komponentin avulla leiskassa päästiin käsiksi aktiivisen *Route*-komponentin lapsi elementteihin. Kuvion *MainLayout*-komponenttia käytetään koko sovelluksen pohjana silloin kun käyttäjä on kirjautunut sovellukseen. Muuten käytetään *LoginLayout*-komponenttia.

```
import { Outlet } from 'react-router-dom';
import Header from '../header';

export default function LoginLayout() {
  return (
    <div className="loginApp">
      <Header />
      <div className="page">
        <Outlet />
      </div>
    </div>
  );
}

import { useState } from 'react';
import { Outlet } from 'react-router-dom';
import Header from '../header';
import Nav from './nav';

Complexity is 3 Everything is cool!
export default function MainLayout() {
  const [showNav, setShowNav] = useState<boolean>(false);
  return (
    <div className="App bg-nav">
      <div onClick={() => setShowNav(false)} className={showNav ? 'background' : ''}></div>
      <Header disableMobileMenu={undefined} toggleNav={setShowNav} showNav={showNav} />
      <Nav showNav={showNav} onClick={() => setShowNav(false)} />
      <Outlet />
    </div>
  );
}
```

Kuvio 15. Layout-komponenttien määritys

Käyttöliittymän visuaalinen ilme sai paljon inspiraatiota mm. tunnetusta *Wordpress*-sisällönhallintajärjestelmän ohjauspaneelista, jonka käyttöliittymän rakenne koostuu ylätunnisteesta, sivupaneelista ja sisältöosiosta. Tokapeli Creator päädyttiin rakentamaan samaan tyyliin. Käyttöliittymästä toteutettiin mahdollisimman helposti laajennettava ja responsiivinen. Responsiivista rakennetta toteutettiin mm. tyylittelemällä elementtejä käyttämällä ”Media queries”-pysäytyspisteitä ja CSS-moduuleiden *flex* ja *grid* ominaisuuksia.

Vaaditun toiminnallisuuden toteutukseen vaadittavat käyttöliittymäratkaisut koostuvat paljolti listoista ja käyttäjän syötteestä muuttuvista tai laajentuvista osioista. Näiden toteutukseen käytettiin mm. silmukoita, *map*-metodia ja Reactin ehdollista piirtämistä (engl. conditional rendering).

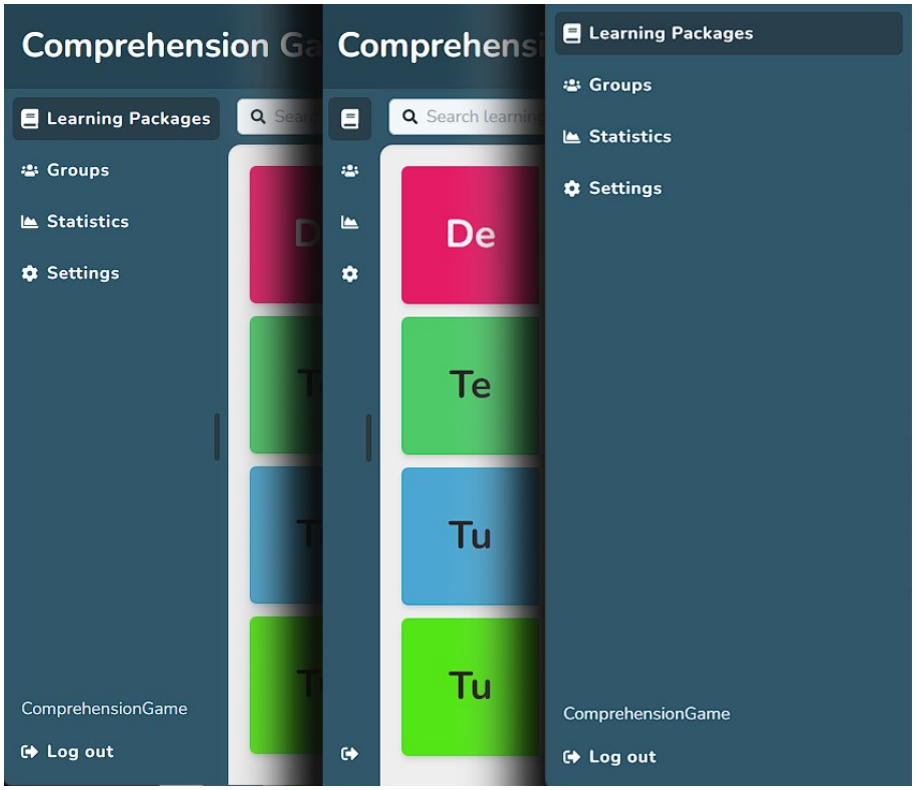
4.3.1 Navigaatio

Sovelluksen navigaatiorakenne (ks. Kuvio 16.) toteutettiin mahdollisimman yksinkertaiseksi ja helposti skaalattavaksi React Router -kirjaston *Routes*- ja *Route*-komponenteilla. Rakenne koostuu kahdesta pesitetystä *Routes*-komponentista eli sovelluksen päätepisteestä, joita käytetään riippuen siitä, onko käyttäjä kirjautunut vai ei. Pesityt päätepisteet kuvastavat sekä verkko-osoitteen, että käyttöliittymän rakennetta. Esimerkiksi käyttäjän navigoidessa päätepisteeseen ”/learning-packages”, käytetään *MainLayout*-leiskaa. Sovelluksen päätepisteet eivät ole suoraan yhteydessä rajapinnan päätepisteisiin, vaan pyyntöjä rajapintaan tehdään tarvittaessa, esimerkiksi komponenttien sivuefekteinä.

```
<Routes>
  <Route element={<LoginLayout />}>
    <Route path="/" element={<Login />} />
  </Route>
  <Route element={<MainLayout />}>
    <Route
      path="/learning-packages" ···
      }
    />
    <Route
      path="/learning-packages/:pkgId/:lngId" ···
      }
    />
    <Route
      path="/learning-packages/add" ···
      }
    />
    <Route
      path="/learning-packages/:pkgId/:lngId/edit" ···
      }
    />
    <Route
      path="/learning-packages/merge" ···
      }
    />
    <Route
      path="/groups" ···
      }
    />
    <Route
      path="/statistics" ···
      }
    />
    <Route
      path="/settings" ···
      }
    />
    <Route path="*" element={<NoMatch />} />
  </Route>
</Routes>
```

Kuvio 16. Sovelluksen navigaatiorakenne

Sovelluksen navigaatio tapahtuu vasemmalla sijaitsevan sivupalkin avulla (ks. Kuvio 17.). Navigaatiopalkissa näkyy sivuvaihtoehdot ja painike käyttäjän uloskirjaamiseksi. Lisäksi palkissa näytetään kirjautuneen käyttäjän käyttäjätunnus. Sivupalkin painikkeet on rakennettu React Router-kirjaston *NavLink*-komponentilla (ks. Kuvio 18.), joka ohjaa käyttäjän, ominaisuusobjektissa ”to” arvona annettuun sovelluksen päätepisteeseen.



Kuvio 17. Navigaatiosivupalkki eri laitteilla

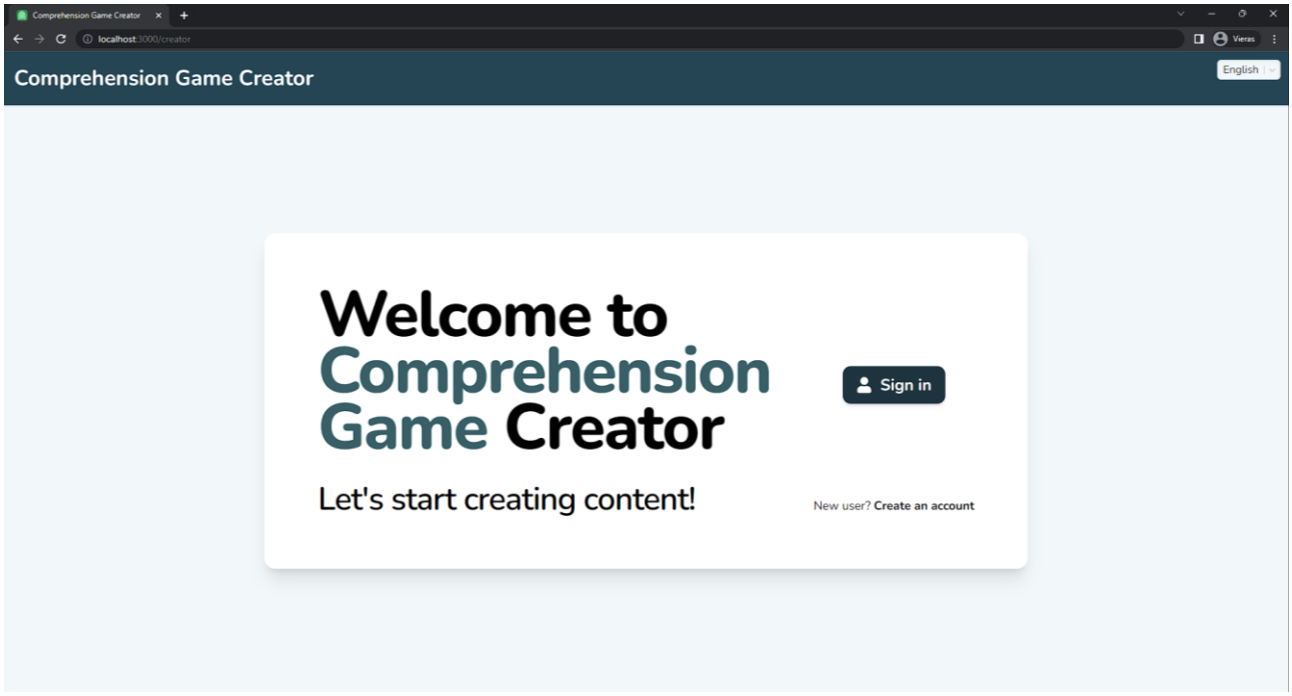
```
const NavButton = ({ children, to, ...rest }: NavLinkProps) => (
  <NavLink className={({ isActive }) => `navBtn rounded-md ${isActive ? 'isActive' : ''}`} to={to} {...rest}>
    {children}
  </NavLink>
);
```

Kuvio 18. *NavLink*-komponentti

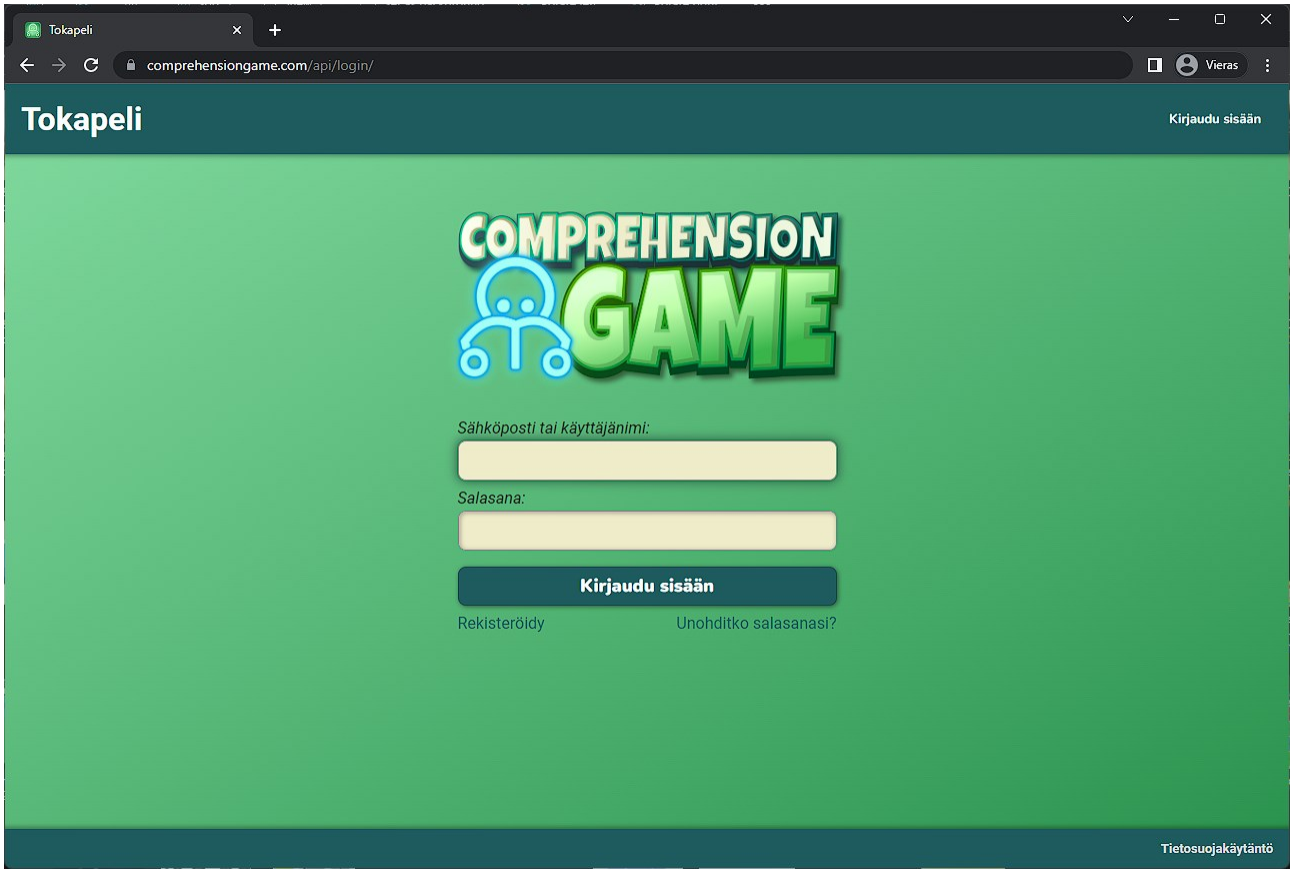
Navigaatiomalli muodostui suunnitellessa sovelluksen käyttöliittymän rakennetta. Käyttöliittymästä tahdottiin mahdollisimman yksinkertainen ja intuitiivinen, koska sen käyttäjäryhmä on taitotasoltaan niin laaja. Sovellus on siis suunnattu niin nuoremmille kuin vanhemmallekin sukupolvelle. Tänä päivänä yhä useampi laajalti käytössä oleva sovellus kuten *Wordpress*, *Youtube*, *Discord*, Microsoft *Teams* käyttävät sivupalkissa sijaitsevaa navigaatiota, joten oli helppoa päätyä ratkaisuun, joka on käyttäjille jo ennestään tuttu. Lisäksi ratkaisu mahdollisti sen, että navigaatio mobiililaitteilla ja tietokoneilla ei poikkea toisistaan liikaa kuten kuviossa 18 käy ilmi.

4.3.2 Laskeutumissivu ja kirjautuminen

Kirjautumattomalle käyttäjälle sovelluksen oletusnäkymänä toimi laskeutumissivu (ks. Kuvio 19.), josta käyttäjä voi painiketta painamalla siirtyä React-sovelluksesta erilliselle, valtuutuspalvelimella sijaitsevalle verkkosivulle, jonka kautta käyttäjä todennetaan (ks. Kuvio 20.). Käyttäjän syötteestä toteutettava pyyntö valtuutuspalvelimelle aloittaa kohdassa 4.2.2 kuvatun valtuutusvirran. Sivu haetaan rajapintaa käyttämälle, tekemällä GET-pyyntö ”login/”-päätepisteeseen.



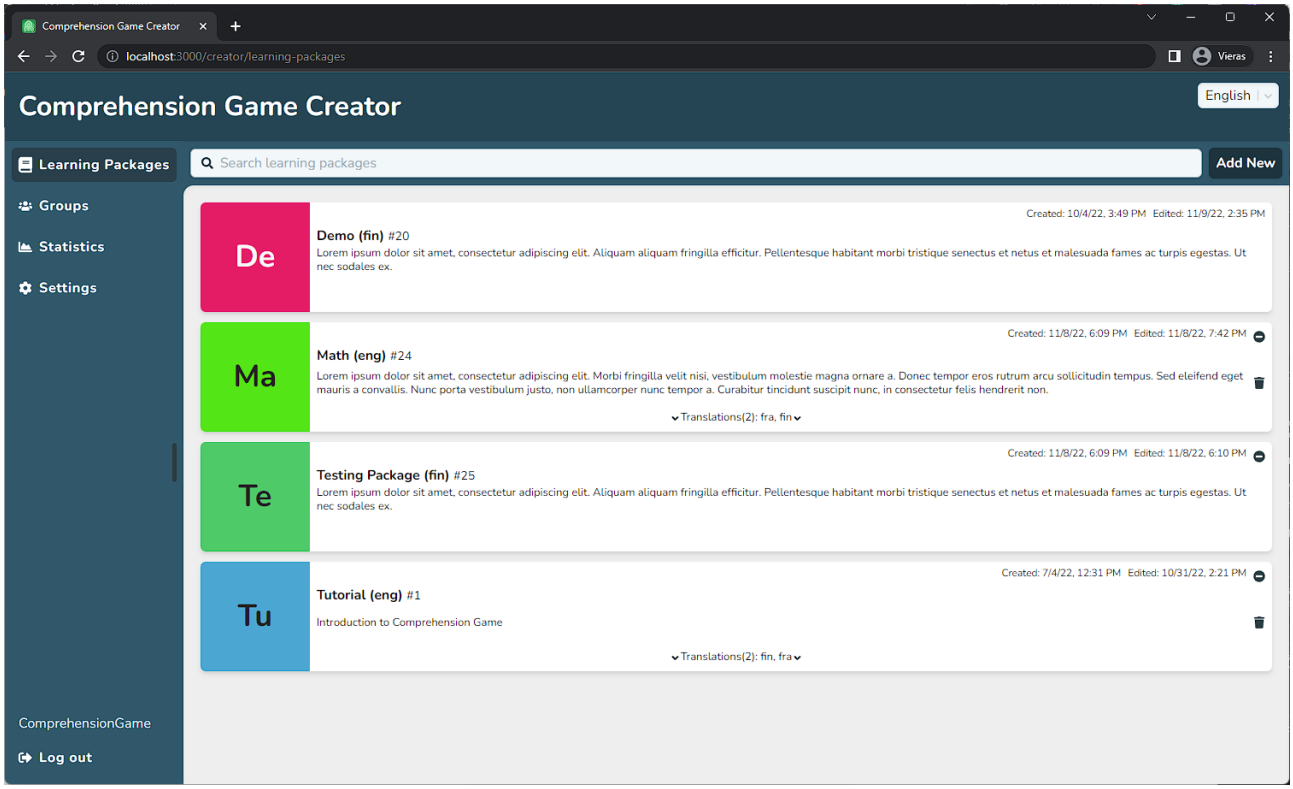
Kuvio 19. Landing page



Kuvio 20. Kirjautumissivu palvelimella

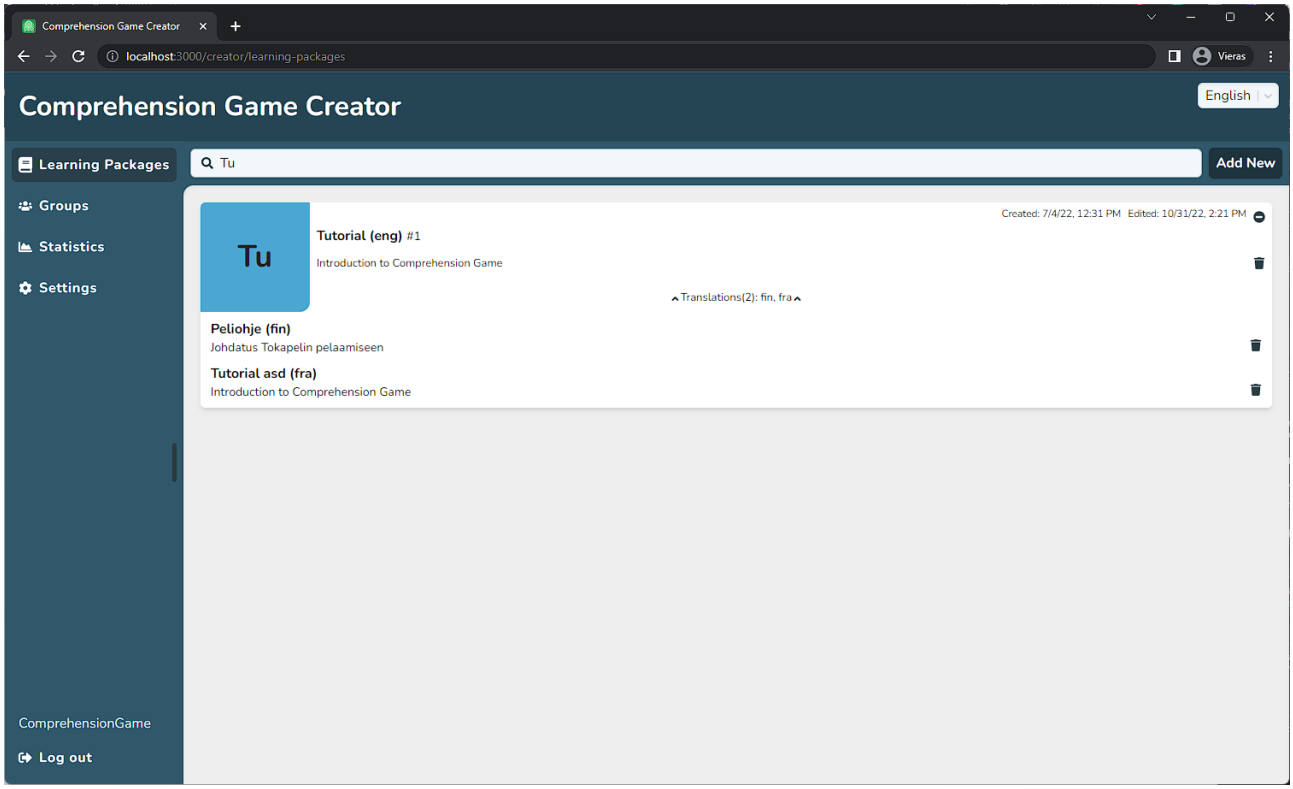
4.3.3 Opetuspakettilistanäkymä

Opetuspakettilistanäkymä (ks. Kuvio 21.) toimii sovelluksen etusivuna todennuksen jälkeen ja sen päätepiste on ”/learning-packages”. Sivulla listataan kaikki opetuspaketit, jotka ovat julkisia tai joita käyttäjällä on oikeus muokata tai katsella. Listan osioiden rajaamista varten näkymässä on hakukenttä.



Kuvio 21. Opetuspakettilista

Listan osiot on suunniteltu mahdollisimman yksinkertaisiksi ja laajennettaviksi. Ne sisältävät ainoastaan tarvittavan tiedon, jotta käyttäjä pystyy nopeasti yksilöimään osioita. Listan osiot laajentuvat käyttäjän halutessa, listaten kaikki kyseisen sisällön käännökset (ks. Kuvio 22.).



Kuvio 22. Käännösten listaaminen

Listan sisältö haetaan suorittamalla *createAsyncThunk*-funktiolla generoitu *fetchPackage*-toiminto (ks. Kuvio 23.) komponentin sivuefektinä eli Reactin *useEffect*-hookkia käyttämällä (ks. Kuvio 24.). Toiminnossa tehdään asynkroninen GET-pyyntö rajapinnan päätepisteeseen ”learning-packages/”, jonka vastauksesta saatu data kirjoitetaan tilaan sen onnistuessa.

```
function updatePackageLists(
  state: LearningPackagesState,
  packageList: (LearningPackageType | LearningPackage)[]
) {
  state.packages = packageList.filter((item) => item.permissions.includes('edit'));
  state.viewablePackages = packageList;
}

export const fetchPackages = createAsyncThunk<
  LearningPackageType[],
  void,
  {
    rejectValue: {
      errorInfo: string | undefined | unknown;
    };
  }
>('learningPackages/fetch', async (_, thunkAPI) => {
  try {
    const res = await tokapeliApi.listLearningPackages();
    if (!res.errorCode) {
      return res.data;
    } else throw res;
  } catch (e: any) {
    return thunkAPI.rejectWithValue({ errorInfo: e.response.data });
  }
});

export const learningPackagesSlice = createSlice({
  name: 'learningPackages',
  initialState,
  reducers: {···
  },
  extraReducers: (builder) => {
    builder.addCase(fetchPackages.pending, (state) => {
      console.log('Pkg fetch pending...');
      state.isLoading = true;
    });
    builder.addCase(fetchPackages.fulfilled, (state, { payload }) => {
      console.log('Pkg fetch fulfilled...', payload);
      const tempState = { ...state };
      updatePackageLists(tempState, payload);
      tempState.isLoading = false;
      return tempState;
    });
    builder.addCase(fetchPackages.rejected,  (state, { payload }) => {
      console.error('Viewable packages fetch rejected...', payload);
      state.isLoading = false;
    });
```

Kuvio 23. *fetchPackages*-toiminto

```
useEffect(() => {
  dispatch(resetResText());
  dispatch(fetchPackages());
  dispatch(setActivePackage());
}, [dispatch]);
```

Kuvio 24. *fetchPackage*-toiminto komponentin sivuefektinä

Tilaan kirjoitetut opetuspaketit ovat taulukkomuodossa, jonka alkiot piirrettiin käyttöliittymään *ListItem*-komponenttia käyttämällä (ks. Kuviot 25 ja 26). Komponentin ominaisuusobjekti muodos-

tetaan taulukon objektialkion arvojen pohjalta. Opetuspaketin käännökset näytetään komponentissa piilotettavana listana. Lista piirretään komponentin *showLanguages*-tilan mukaan, jota vaihdetaan käyttäjän syötteestä.

```
<div className="learningPackages">
  <ul className="gap-3">
    {filteredPackages.length > 0 ? (
      filteredPackages.map((lp) => {
        if (lp.languages && lp.languages.length > 0) {
          const backup = getStorageValue(createLocalStorageKey(username, lp.id));
          const localLp = backup ? combinePkgLanguages(lp, backup) : lp;
          const lng = selectDefaultLanguagePackage(localLp.languages);
          return lng !== null && lng !== undefined ? (
            <ListItem
              key={lp.id}
              lp={lp}
              lng={lng}
              localLp={localLp}
              handleDetails={handleDetails}
              handleEdit={handleEdit}
              backup={backup}
              readOnly={lp.type === 'edit' ? false : true}
            />
          ) : null;
        } else {
          console.error('Empty package detected. Something is probably wrong: id', lp.id);
          return undefined;
        }
      })
    ) : (
```

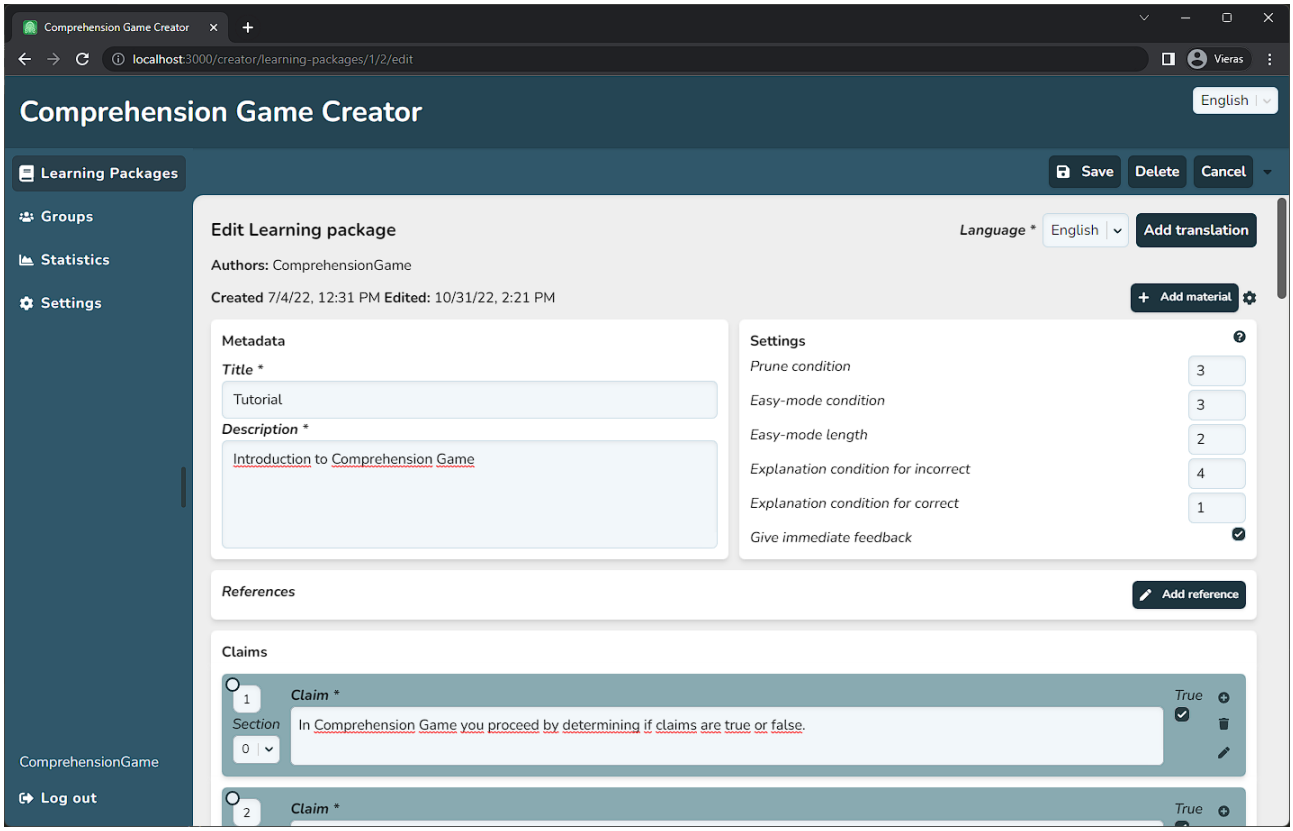
Kuvio 25. Opetuspakettien läpiajo

```
export const ListItem = ({
    lp,lng, localLp, handleDetails, handleEdit, backup, readOnly
} : LearningPackageListItem) => {
  const { t } = useTranslation();
  const [showLanguages, setShowLanguages] = useState(false);
  const dispatch = useAppDispatch();

  const languageComponents = useMemo(···
  );

  return (
    <li
      key={uuidv4()}
      className="..."
    >
      <div
        className="..."
        onClick={() => (readOnly ? handleDetails(lp.id, lng.id) : handleEdit(lp.id, lng.id))}
      >
        <div···
        </div>
        <div className="...">···
        </div>
      </div>
      {localLp.languages.length > 1 && (
        <div···
        </div>
      )}
      {showLanguages && languageComponents}
    </li>
  );
};
```

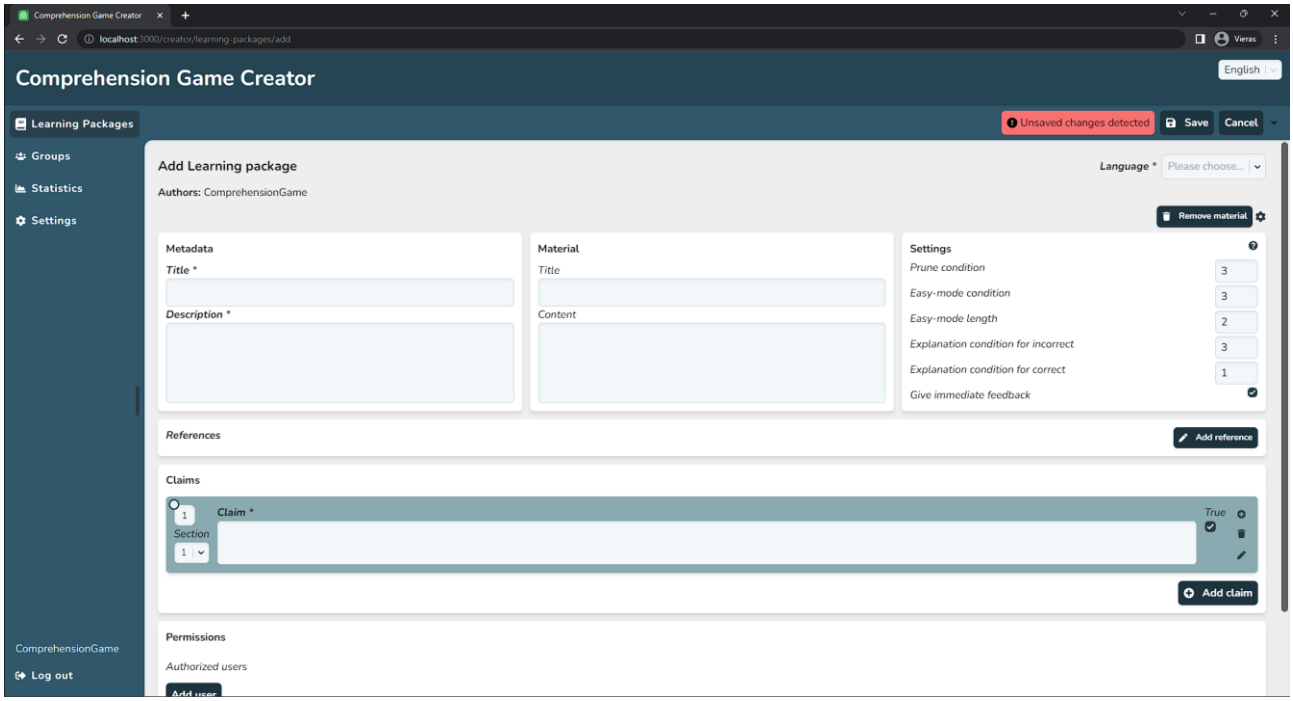
Kuvio 26. *ListItem*-komponentti

4.3.4 Opetuspaketin muokkaus- ja lisäysnäkymä

Opetuspaketin muokkaus näkymä (ks. Kuvio 27.) avautuu painamalla opetuspakettilistan osiota, mikäli käyttäjällä on tähän oikeus. Näkymä on dynaaminen lomake, jota muokkaamalla ja täyttämällä käyttäjä voi muuttaa opetus paketin sisältöä. Lisäysnäkymä (ks. Kuvio 28.) on sama kuin muokkaus näkymä, mutta lomakkeen kentät ovat siinä oletuksena tyhjät. Käännöksen lisääminen tapahtuu ”Add translation”-painikkeesta, joka avaa laajan kielilistan, josta käyttäjä voi valita haluamansa kielen. Kielen valinnan jälkeen edellisen käännöksen sisällöt kopioidaan uuteen lomakkeeseen ja asetetaan paketin kieleksi valittu kieli. Käyttäjä voi kääntää kentät, korvaamalla arvot tekemällään käännöksellä. Kääntäminen ei siis tapahdu automaattisesti.

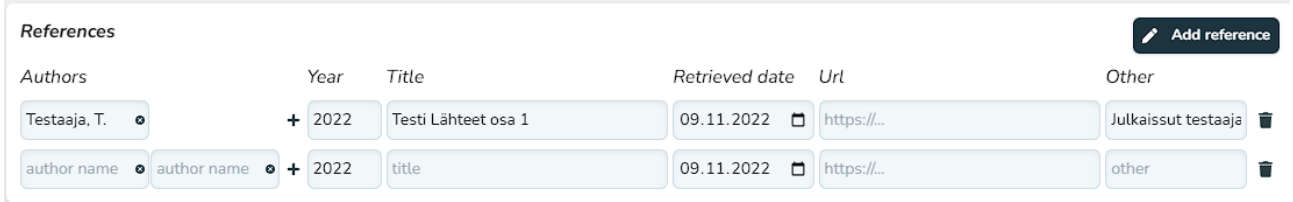


Kuvio 27. Opetuspaketin muokkausnäkymä

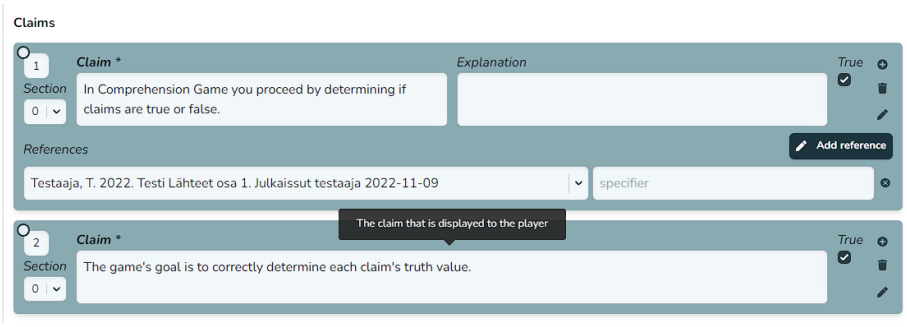


Kuvio 28. Opetuspaketin lisäysnäkymä

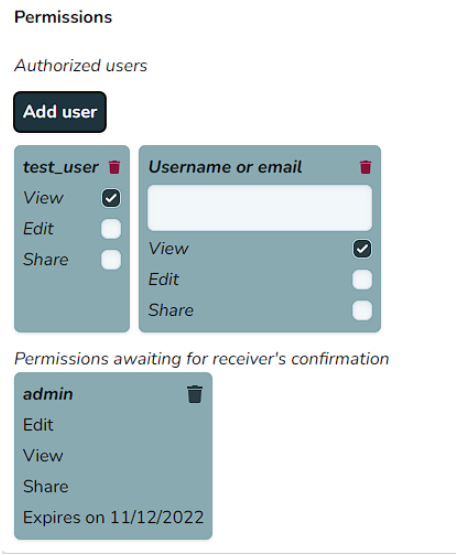
Näkymä suunniteltiin mahdollisimman selkeäksi ja itsestään selväksi, jotta suuren kokonaisuuden hallinta olisi käyttäjälle mahdollisimman helppoa. Näkymän lomake on jaettu kolmeen pääosaan: metadataan, väittämiin ja käyttöoikeuksiin. Lomaketta laajennetaan lisäämällä alkioita osioiden listoihin tarpeen mukaan. Lisättäviä alkioita ovat lähteet (ks. Kuvio 29.), väittämät (ks. Kuvio 30.) ja oikeudet (ks. Kuvio 31.). Jokainen alkio luodaan tätä kuvaavaa funktionaalista komponenttia käyttäen. Esimerkiksi väittämälistan alkiot piirretään *SentenceComponent*-komponenttia käyttäen. Laajennusominaisuus on toteutettu koodissa hyödyntämällä ehdollista piirtämistä (ks. Kuvio 32.).



Kuvio 29. Lähdelista muokkaunäkymässä



Kuvio 30. Laajennettu ja normaali väittämä.



Kuvio 31. Oikeuksienhallinta muokkausnäkymässä

```
  </div>
  {item.showOptions && (
    <div className="advancedOptions">
      <label htmlFor={'sentence_reference_' + item.id} className={`${!hasReference ? 'hidden' : ''}`}>
      </label>
      {!readOnly && (
        <LoadingButton
        </LoadingButton>
      )}
      <div className={`advancedOptions_referenceSelectionContainer${!hasReference ? ' pt-0' : ''}`}>
        {item.references.length > 0
          ? item.references.map((r: SentenceReference, sRefIndex: number) => (
              <div
              </div>
            ))
          : null}
      </div>
    </div>
  )}
</li>
```

Kuvio 32. Ehdollista piirtämistä *SentenceComponent*-komponentissa

Muokkaus näkymän päätepiste on ”/learning-packages/:pkgId/:lngId/edit” ja lisäysnäkymän ”/learning-packages /add”. Kaksoispisteellä alkavat osat päätepisteessä kuvastavat sen dynaamista osaa eli parametrejä, joita käytetään oikean opetuspaketin hakemiseen palvelimelta. Navigaatio näkymiin tehdään *ListItem*-komponentin *onClick*-tapahtumaan sidotuilla funktiolla (ks. Kuvio 33.). Muokkaus- ja tarkastelunäkymää varten opetuspaketin tarkemmat tiedot haetaan navigaation jälkeen näkymäkomponentin sivuefektinä ja asetetaan sovelluksen tilaan aktiiviseksi paketiksi (ks. Kuvio 34.).

```
const handleDetails = async (learningPkgId, lngPkgId) => {
  navigate(`/learning-packages/${learningPkgId}/${lngPkgId}`);
};

const handleEdit = async (learningPkgId, lngPkgId) => {
  navigate(`/learning-packages/${learningPkgId}/${lngPkgId}/edit`);
};

const handleAdd = () => {
  dispatch(setActivePackage());
  navigate('/learning-packages/add');
};
```

Kuvio 33. Funktiot muokkaus-, tarkastelu- ja lisäysnäkymiin

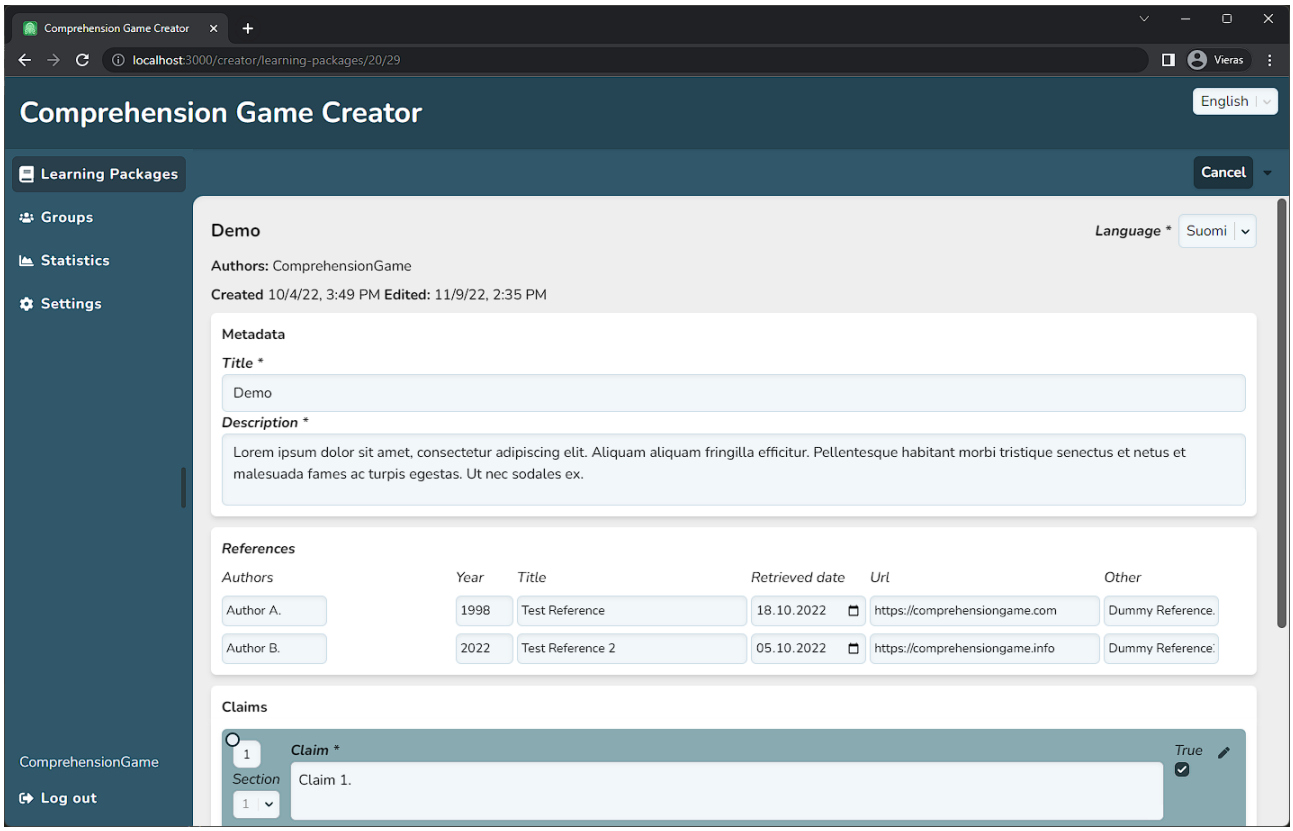
```
useEffect(() => {
  if (typeof pkgId === 'number' && pkgId !== activePackage?.id) {
    console.log('Matching active package not found. Retrieve correct package from server.');
    const fetchActivePackage = async () => {
      const result = await tokapeliApi.getLearningPackage(pkgId);
      if (!result.errorCode) {
        dispatch(setActivePackage({ pkg: result.data }));
        const retrievedBackup = retrieveBackup();
        let lpToSet;
        if (retrievedBackup) {···
        } else {···
        }
        if (selectedItems.length > 0) {···
        }
        setLp(lpToSet);
      } else {
        console.error('Failed to retrieve active package');
        navigate('/learning-packages');
      }
    };
    fetchActivePackage();
  } else if (matchAdd && lp.id !== activePackage?.id) {
    initializeAddPackage(lp);
  }
}, [
  activePackage?.id,
  addSentences,
  dispatch,
  initializeAddPackage,
  lngId,
  lp,
  lp.references,
  matchAdd,
  navigate,
  pkgId,
  retrieveBackup,
  selectedItems,
]);
```

Kuvio 34. Sivuefekti sisällön tarkempien tietojen hakemiseksi palvelimelta

Opetuspaketin tallentaminen toteutetaan lähettämällä muokatessa PATCH- ja lisätessä POST-pyyntö rajapinnan päätepisteeseen. Muokattu tai lisätty opetuspaketti asetetaan pyynnön *body*-osioon.

4.3.5 Opetuspaketin tarkastelu -näkymä

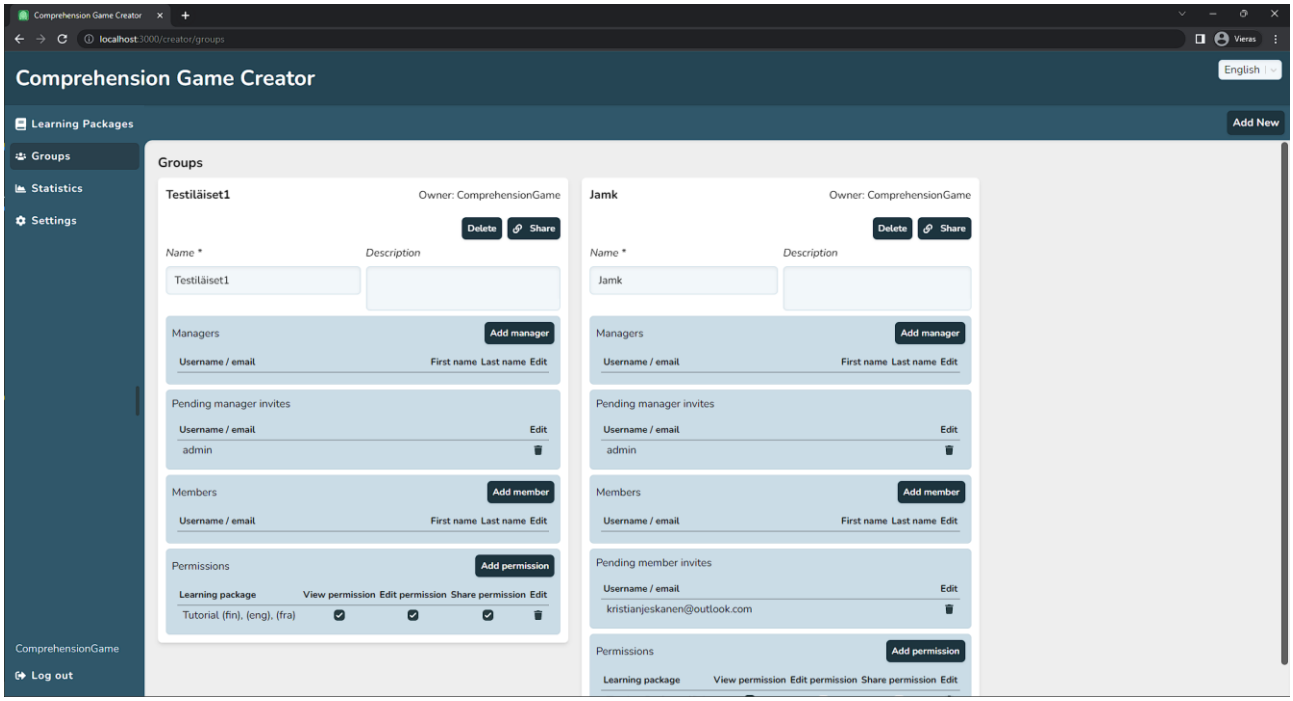
Tämä näkymä (ks. Kuvio 35.) on pelkistetympi versio muokkausnäkymästä ja se avautuu opetuskattilistan osiosta, jos käyttäjällä ei ole oikeutta muokata sisältöä. Näkymän päätepiste on ”learning-packages/ learning-packages/:pkgId/:lngId”.



Kuvio 35. Opetuspaketin tarkastelu -näkymä

4.3.6 Ryhmät-näkymä

Ryhmät-näkymässä (ks. Kuvio 36.) listataan kaikki käyttäjän luomat oikeusryhmät ja kaikki ryhmät, joihin tämä kuuluu. Listan osiot ovat ”kortti”-muodossa ja niissä näytettävä tieto on pyritty tiivistämään mahdollisimman pieneen tilaan niin, että ne toimivat myös mobiili näkymässä.

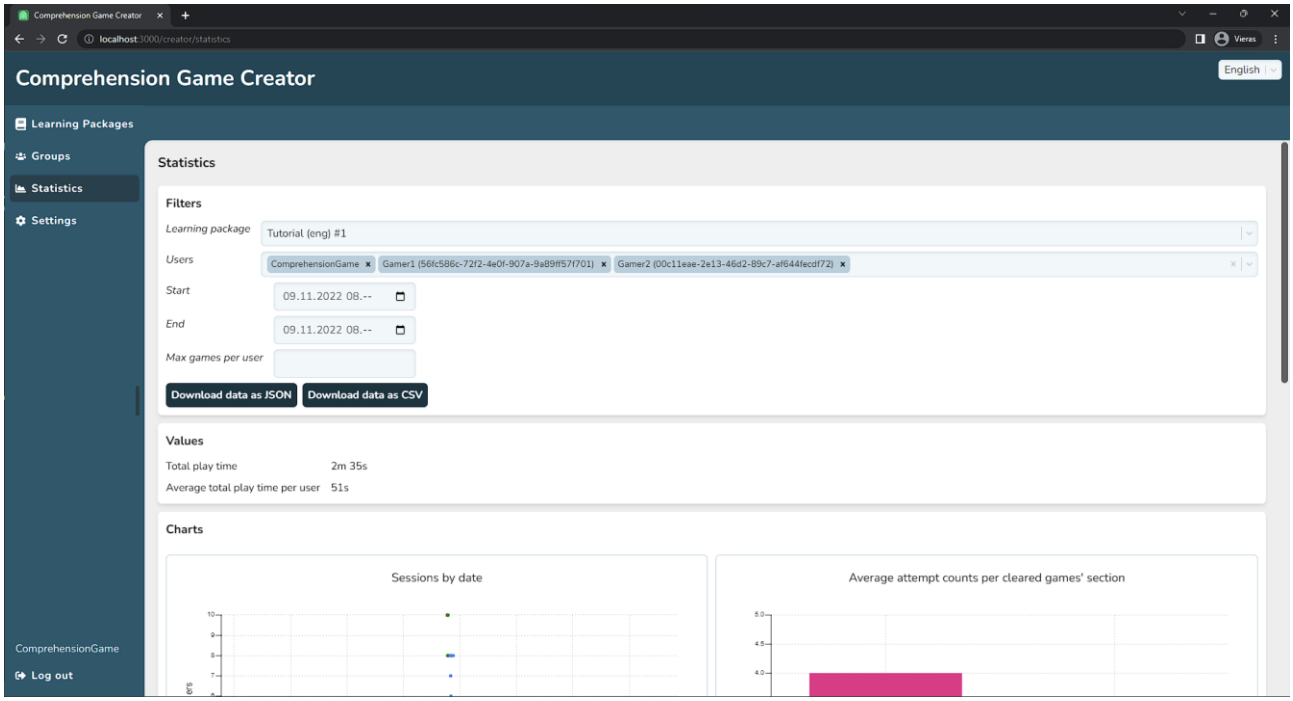


Kuvio 36. Ryhmät-näkymä

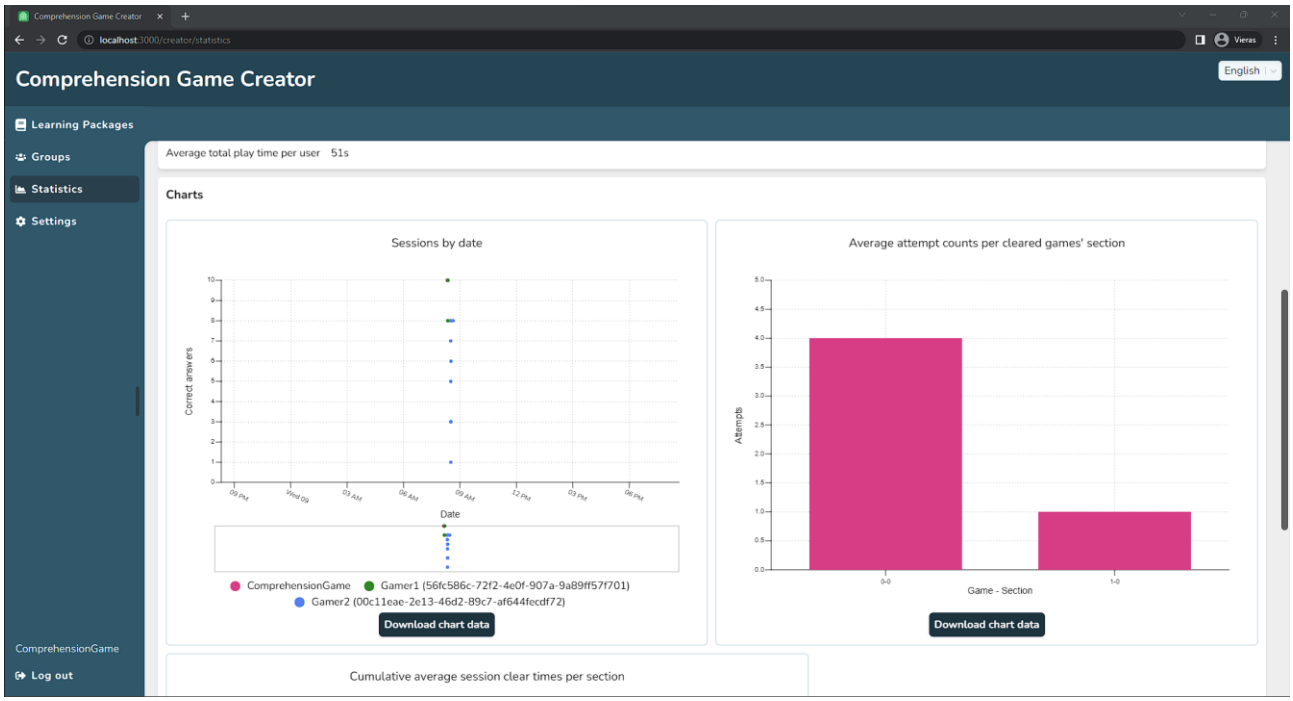
Päätepiste on ”/groups”. Näkymässä olevan listan sisältö haetaan sivuefektinä rajapinnan päätepisteestä ”permission-groups/”. Ryhmien tallennus tehdään muokatessa PATCH- ja lisätessä POST-pyynnöllä. Pyynnön *body*-osioon asetetaan päivitetty ryhmäobjekti. Pyyntö lähetetään hakua vastaavaan päätepisteeseen, lisäten siihen parametrinä ryhmän id.

4.3.7 Tilastot-näkymä

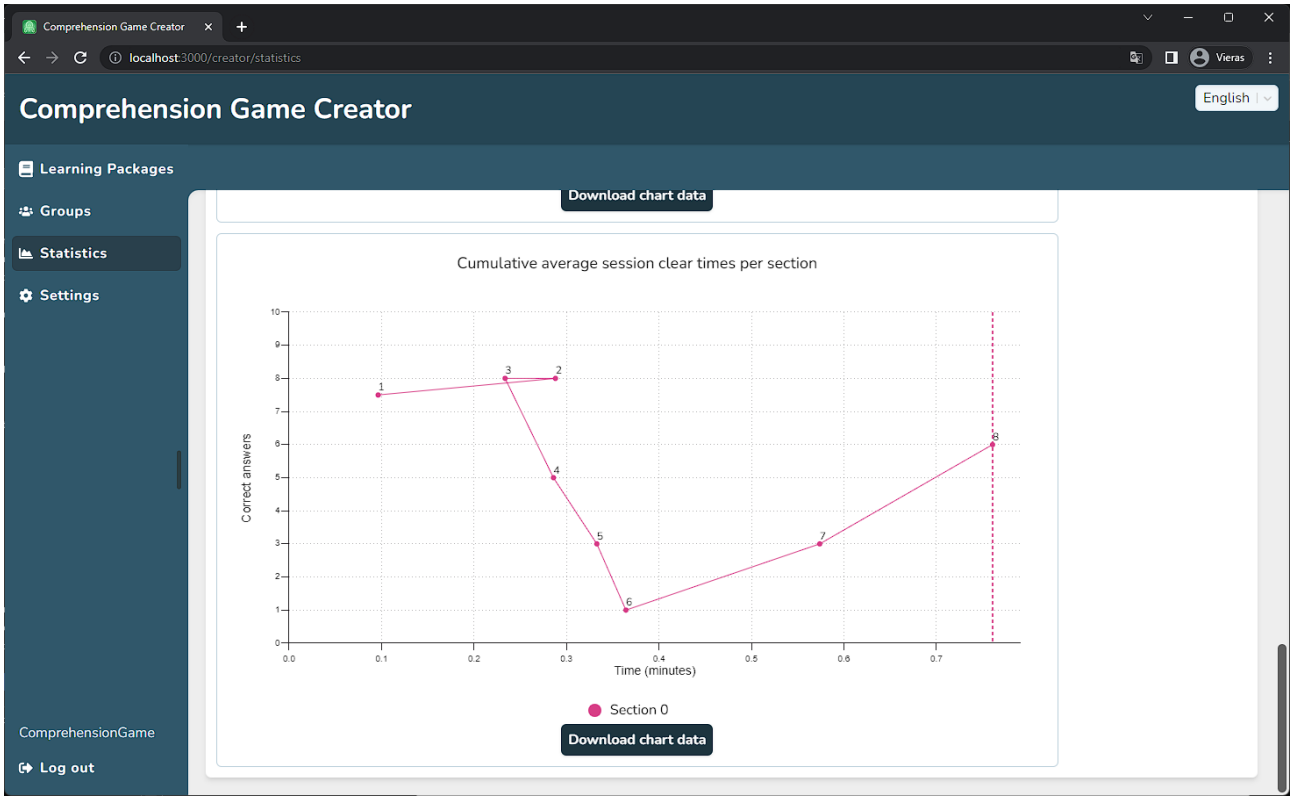
Tilastot näkymässä (ks. Kuvio 37.) näytetään tärkeitä arvoja koko paketin kohdalta ja yksityiskohtaisemmin kaavioita käyttäjän asettamien määritysten mukaisesti. Kuvioissa 38 ja 39 näkyvien kaavioiden piirtämiseen käytetään *Visx*-kirjastoa. Tilastojen piirtymiseksi käyttäjän tulee valita opetuspaketti. Oletuksena sovellus piirtää kaaviot kaikkien käyttäjien datasta, joihin niitä hakevalla käyttäjällä on oikeus. Tilastoja voi rajata käyttäjän, päivämäärän ja pelien määrän mukaan. Näkymän päätepiste on ”/statistics”.



Kuvio 37. Tilastot -sivun hakufiltterit ja tärkeät arvot



Kuvio 38. Kaaviot sessioista ja jaksojen läpäisyyn kuluneiden yritysten keskiarvosta

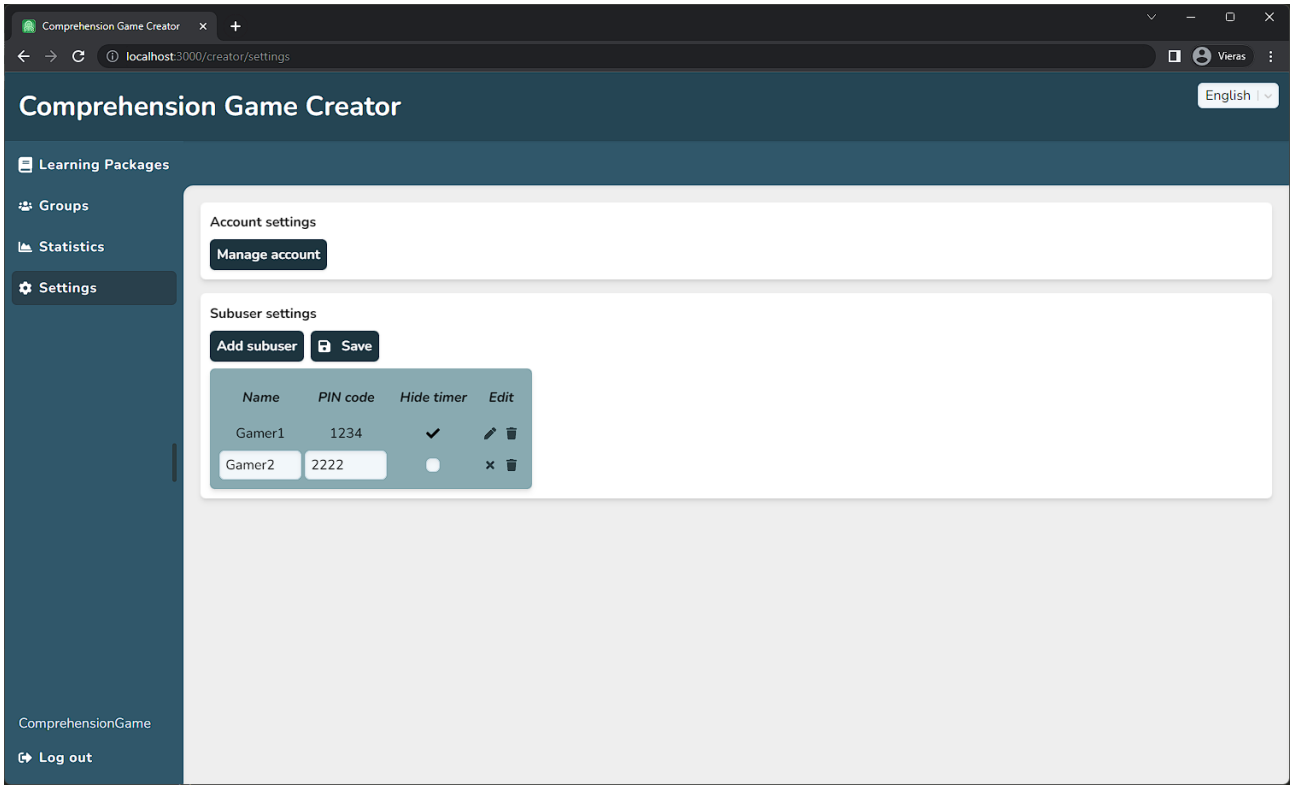


Kuvio 39. Kaavio sessioihin kuluneesta ajasta jaksoittain, kumulatiivisesti esitettynä.

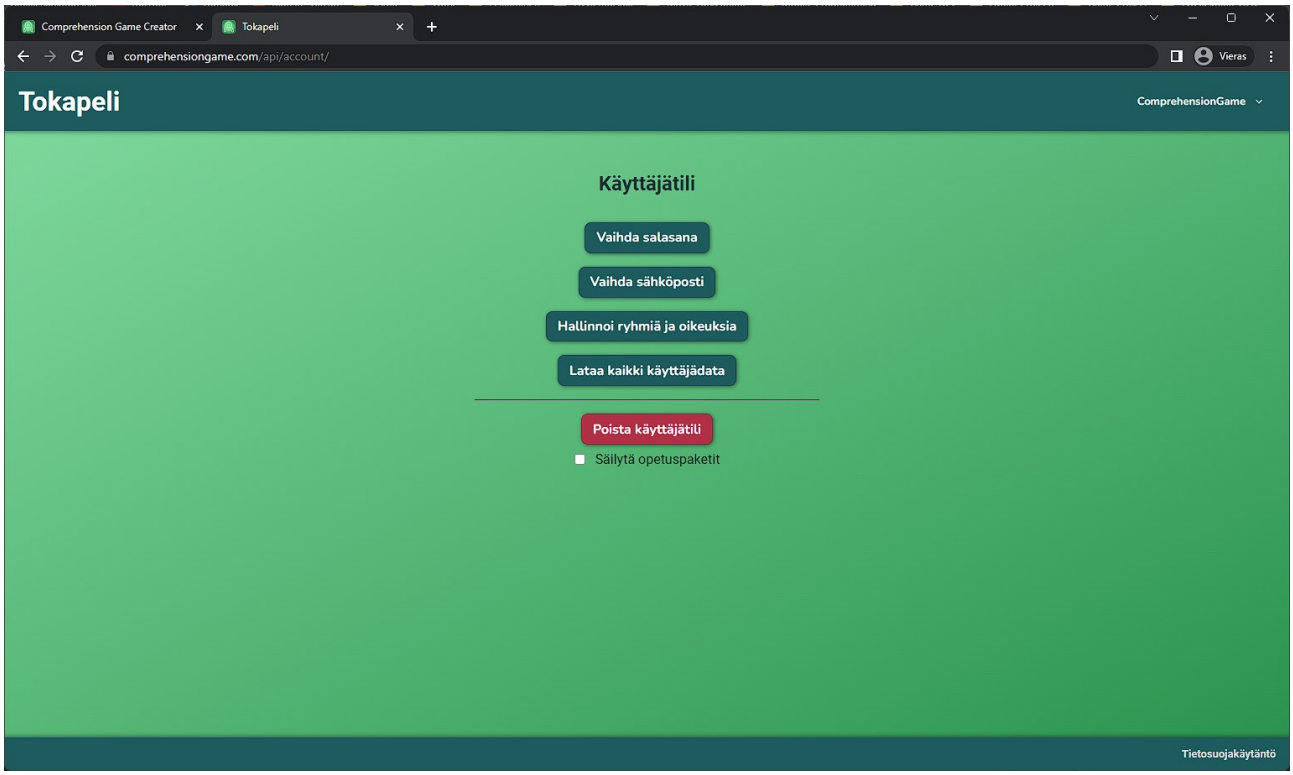
Kaavioiden piirtämiseen tarvittava data haetaan palvelimelta lähettämällä GET-pyyntö rajapinnan päätepisteeseen ”research/games/”, silloin kun käyttäjä valitsee opetuspaketin listasta tai kun rajausmääritykset muuttuvat.

4.3.8 Asetukset-näkymä ja tilinhallinta

Asetukset näkymässä (ks. Kuvio 40.) listataan asetusosiot aiheittain. Siitä pyrittiin suunnittelemaan yksinkertainen ja helposti laajennettava. Sovellusasetuksia oli vaatimusmäärittelyssä vain yksi, kielen vaihtaminen, eikä sitä haluttu piilottaa asetusten taakse vaan pitää mahdollisimman näkyvillä sovelluksen ylätunnisteessa. Asetus-kategoriaan sopivia ominaisuusvaatimuksia kuitenkin oli, joten Asetuksiin päädyttiin lisäämään ne eli polku tilinhallintasivulle (ks. Kuvio 41.) ja osio alakäyttäjien hallintaa varten. Näkymän päätepiste on ”/settings”.



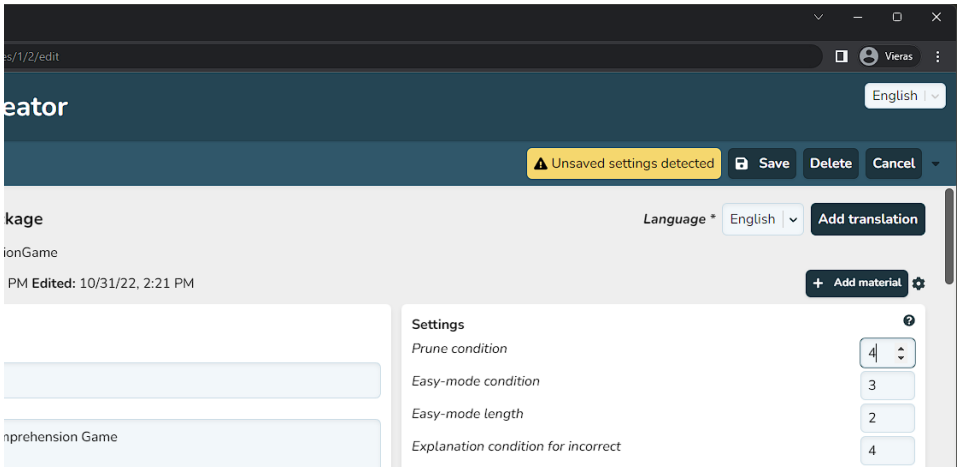
Kuvio 40. Asetukset-näkymä



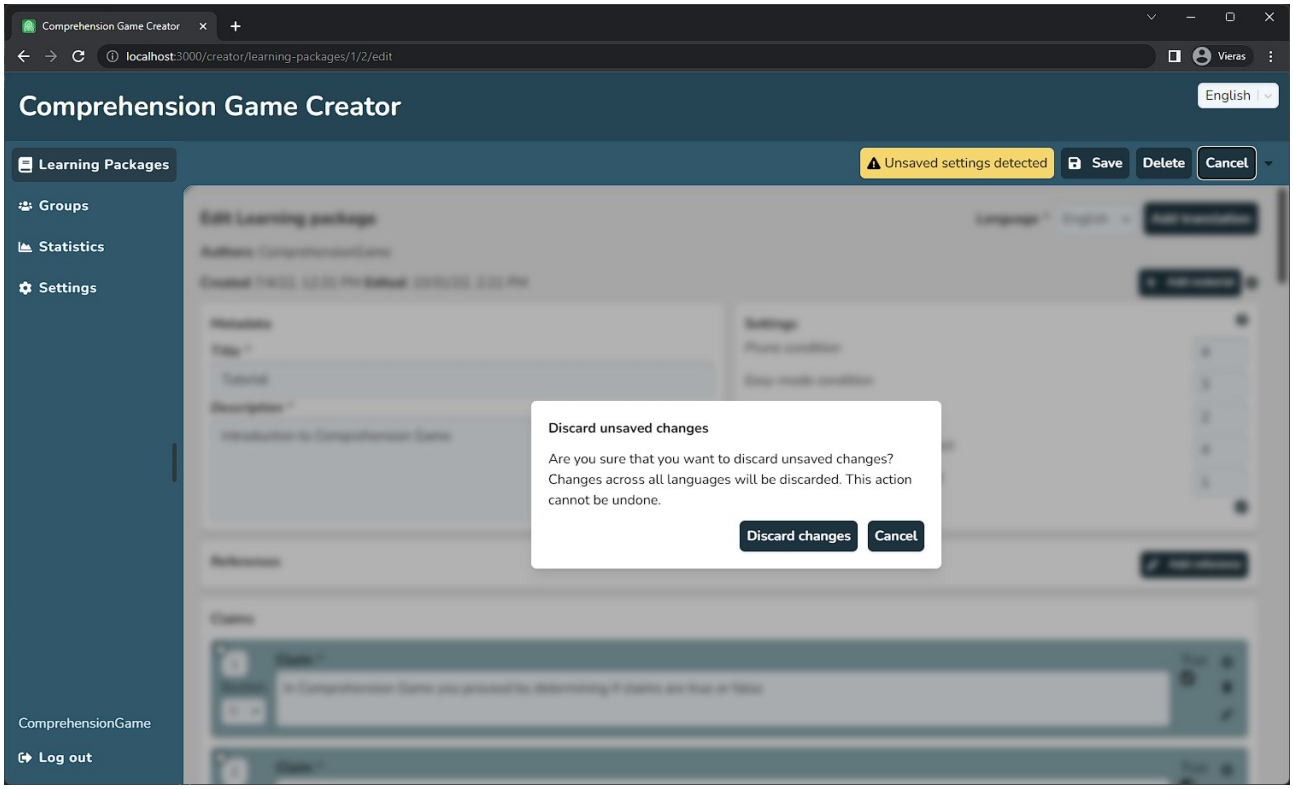
Kuvio 41. Tilinhallintasivu palvelimella

4.3.9 Modalit ja ilmoitukset

Käyttäjää pyritään informoimaan tarvittavissa tilanteissa, kuten silloin kun käyttäjän toimien vaikutus on peruuttamatonta tai kun halutaan käyttäjän vahvistus. Toiminnoista suoriutumisen tilan näyttämiseen käytetään pientä ilmoitus komponenttia, joka sijaitsee sisältöosion ylätunnisteessa (ks. Kuvio 42.). Vahvistukseen käytetään modaalia (ks. Kuvio 43.), joka täyttää sovelluksen sisältöosion ja täten pysäyttää käyttäjän toiminnan, herättäen tämän huomion. Jotta käyttäjä voi jatkaa tältä vaaditaan syöte.



Kuvio 42. *SmallAlert*-komponentilla luotu ilmoitus

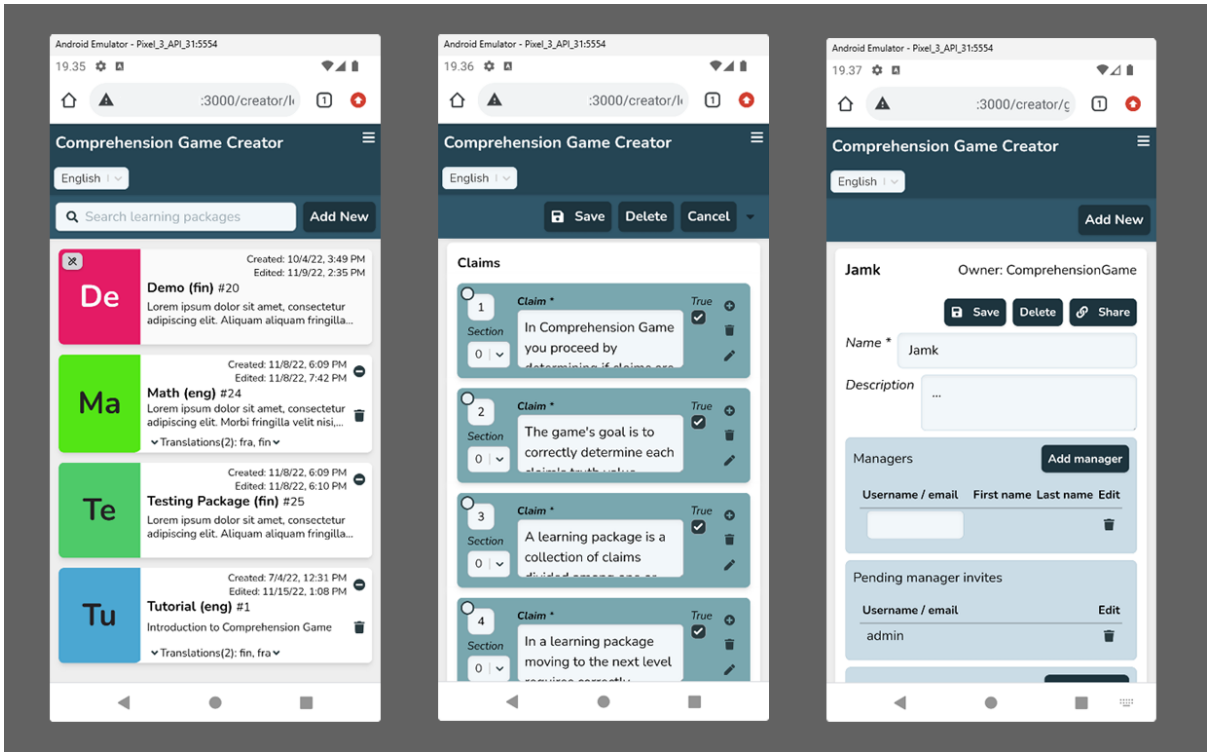


Kuvio 43. Käyttäjän vahvistusta vaadittaessa käytettävä modaali

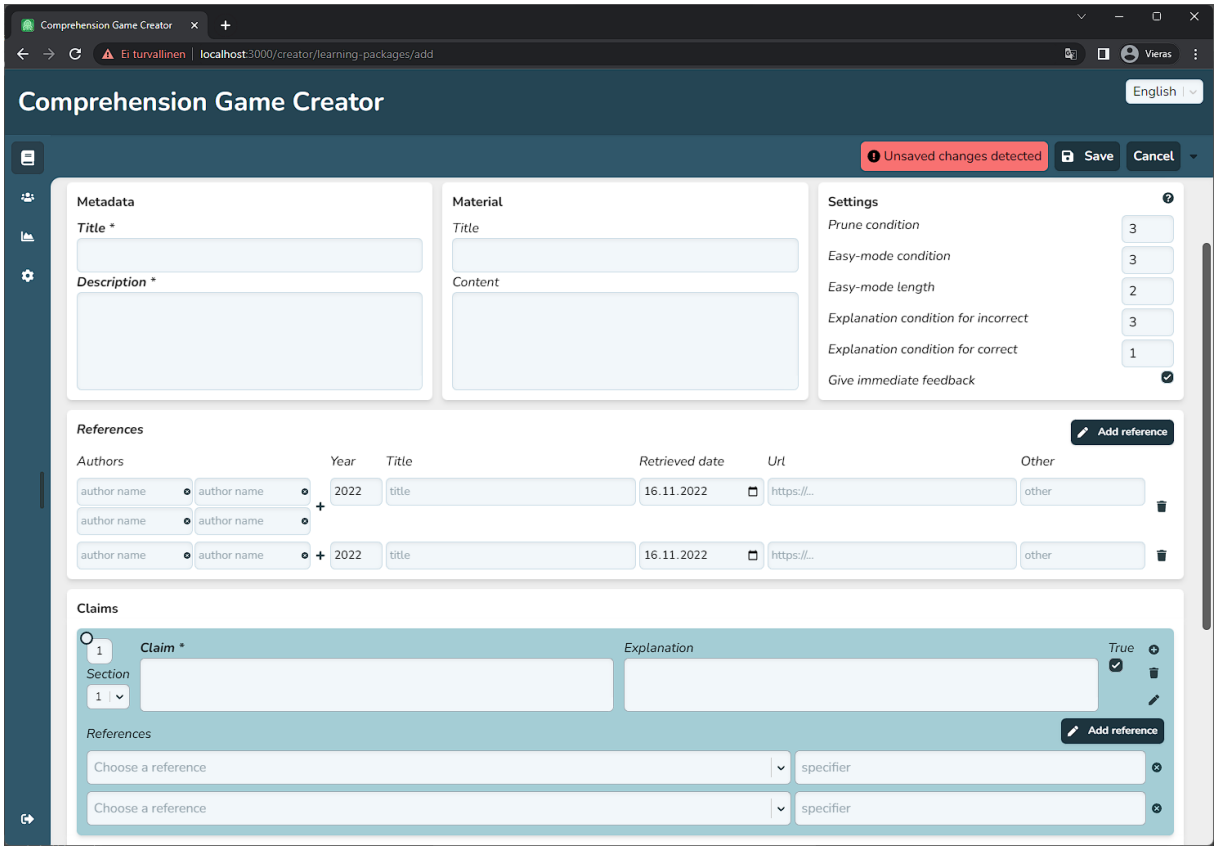
5. Tulokset

Tuloksena saatiin aikaan ensimmäinen tuotannossa oleva pilottiversio Tokapeli Creator -sisällönhallintasovelluksesta, joka sisältää ensimmäisten vaatimusten lisäksi myös muita käyttäjäkokemusta parantavia ominaisuuksia, kuten tilinhallinta. Tuotteen palvelinpuoli ja käyttöliittymä saatiin siihen vaiheeseen, että pystyttiin turvallisesti tekemään testijulkaisu yrityksen verkkotunnuksen alle. Kehitettävää on paljon ja projekti jatkuu vielä työn päättymisen jälkeen, mutta jatkokehityksen kannalta tuote toimii hyvänä runkona.

Sovellus on responsiivinen (ks. Kuvio 44.) ja se tarjoaa ne ominaisuudet mitä siihen suunniteltiinkin ja enemmän. Käyttöliittymä koostuu viidestä päänäkymästä, jotka ovat opetuspakettilista, muokkausnäkymä, ryhmät-näkymä, tilastot-näkymä ja asetukset. Sovelluksen käyttötarkoituksen huomioon ottaen merkittävin tulos työssä on kuitenkin sisällön lisäämiseen ja muokkaamiseen tarkoitettu dynaaminen lomake (ks. Kuvio 45). Se on tulevaisuutta ajatellen helposti laajennettavissa sekä käyttöliittymä-, että palvelinpuolella, joka mahdollistaa Tokapelin sisällön muutokset mahdollisimman pienellä työllä.



Kuvio 44. Käyttöliittymä mobiililaitteella



Kuvio 45. Dynaaminen lomake

Sovelluksen palvelinpuoli on laaja kokonaisuus, jota käytetään myös Tokapeli-sovelluksessa, joten kaikkea sen toiminnallisuutta ei tässä työssä avattu vaan siitä esiteltiin ainoastaan tämän sovelluksen kehitykseen liittyviä kokonaisuuksia. Merkittävimmät palvelinpuolen tulokset tämän sovelluksen näkökulmasta ovat tietokantarakenne, REST-rajapinta ja tietoturvan kannalta hyvän käytännön mukaisesti rakennetut valtuutus ja todennus.

6. Pohdinta

Projektin tavoitteena oli kehittää sovellus, jolla voidaan hallita Tokapeli-oppimisympäristön sisältöjä ja tarkastella sisällön suorittamisesta kertynyttä dataa. Teknologioiksi toteutukseen valittiin Django-verkkokehys ja React-käyttöliittymäkirjasto, joita hyödyntämällä saatiin kehitettyä tavoitteiden mukainen sovellus.

Projekti oli alusta saakka hyvin lennokas ja muutoksia tuli paljon, joka vaikutti paljon siinä käytettäviin teknisiin ratkaisuihin. Ominaisuuksia implementoitiin kiireellisellä aikataululla pitkin projektia, jonka vuoksi toteutusten lopputulos ei aina ollut parasta laatuaan. Ohjelmoijilla ei ennestään ollut kokemusta tämän mittakaavan projekteista, jonka vuoksi joitain teknologioita otettiin käyttöön niitä löytäessä.

Kohdassa 3 esitetty ensimmäinen vedos vaatimusmäärittelystä oli hyvin suuripiirteinen, eikä se ottanut kantaa vaatimusten pohjana vaadittaviin rakenteisiin kuten todennukseen, sisällön muotoon tai tilastojen esitystapaan. Nämä seikat tarkentuivat suunnittelun aikana. Vaatimusmäärittelyn muutoksiin vaikuttivat myös paljon Tokapeli-oppimisympäristön rinnakkainen kehitystyö ja siihen implementoitavat uudet ominaisuudet.

Projektin suurimpana haasteena oli selkeästi tiimin kokemattomuus, jonka vuoksia kaikki hyvät käytännöt verkkokehityksessä eivät olleet käytössä alusta saakka. Tämän seurauksena esim. sovelluksesta jäi puuttumaan testitapaukset ja projekti aloitettiin ohjelmoimaan JavaScriptillä, TypeScriptin sijaan, joka otettiin käyttöön vasta työn loppuvaiheessa. TypeScriptin käyttö olisi nopeuttanut koodin kirjoittamista, sillä sen tyyppitarkistuksen ansiosta olisi nähty minkä tyyppisiä parametrejä funktiot käyttävät ilman tarkistamista funktion dokumentaatiosta tai koodista. Lisäksi virheiden tunnistaminen jo koodia kirjoittaessa olisi vaikuttanut tuottavuuteen positiivisesti. Kokonaisuudessaan projektin aloitus oli teknologiakokonaisuudeltaan hyvin puhdas ja tavallinen, eikä

siinä ollut esimerkiksi käytössä erillisiä UI-kirjastoja tai komponenttikirjastoja, joita myöhemmin lisättiin tarpeen mukaan. Vaikka muutokset olivat laadullisesti perusteltuja, vaikutti se kuitenkin ajallisesti projektin etenemiseen, koska uusia teknologioita täytyi opetella ja ottaa käyttöön. Toisaalta jotkut muutokset olisivat itsessään nopeuttaneet projektin etenemistä, kuten UI-kirjastojen käyttö. Tiimin kokemuksen huomioon ottaen, teknologia valinnat, visuaalinen ilme ja palvelinratkaisut onnistuivat kuitenkin taitotasoon nähden hyvin.

Reactin valinta käyttöliittymän toteutukseen oli mielestäni kestävä ratkaisu. Tulevaisuudessa voi olla mahdollista, että sovellus halutaan muuttaa esim. suositun ja alati kehittyvän React-pohjaisen Next.js-verkkokehityskehyksen rakenteeseen, joka on mahdollista saman kielen ansiosta. Next.js mahdollistaisi mm. sovelluksen palvelinpuolen renderöinnin. Jatkokehitykselle on paljon mahdollisuuksia. Pari projektin aikana esiin noussutta ideaa olivat opetuspakettien versionhallinta ja reaaliaikainen ryhmämuokkaustoiminto. Käyttäjäkokemuksen ja visuaalisuuden näkökulmasta teemat, animaatioiden lisääminen ja responsiivisuuden parantaminen ovat tärkeitä jatkokehityksen kohteita. Lisäksi sovelluksen saavutettavuutta voisi parantaa lisäämällä helppokäyttötoimintoja ja parantamalla HTML-rakennetta semanttisemmaksi. Käyttöliittymän kielen kääntämisen helpottamiseksi käännöstiedostot voisi ladata palvelimelle tai tähän tarkoitettuun palveluun.

Django ja sen tukena käytetyt kirjastot toimivat mielestäni hyvin palvelinpuolen toteutukseen. Vaikka kehitystiimillä ei ollut ennestään suurempaa kokemusta verkkokehyksen käytöstä, saimme silti aikaan monipuolista palvelinpuolen toiminnallisuutta. Django perii kaikki Pythonin hyvät puolet ja mielestäni lisää niitä paljon omilla ominaisuuksillaan. Muun muassa Djangon ORM-teknologian tuoma apu tietokannan käsittelyssä helpotti ja nopeutti valtavasti kehitysprosessia. Lisäksi Djangon arkkitehtuuri mahdollisti Tokapelin ja Tokapeli Creatorin käyttäjien todennukseen ja tilinhallintaan tarvittavan logiikan ja sivujen toteuttamisen yhdessä paikassa eli palvelinpuolella, eikä näitä ei tarvinnut toteuttaa erillisillä teknologioilla.

Vaikka projektia ei käytännössä saatettu täysin loppuun saakka, se täytti tekijän henkilökohtaiset tavoitteet täysin. Tokapeli Creator -projekti opetti tekijälle paljon sovelluskehityksen eri osa-alueista, hioi ohjelmointi- ja suunnittelutaitoja sekä opetti tiimissä työskentelyä.

Lähteet

About. N.d. Artikkeli PostgreSQL sivustolla. Viitattu 9.10.2022. https://www.postgresql.org/about/.

Authorization Code Grant. N.d. OAuth-verkkodokumentaatio. Viitattu 9.10.2022. https://www.oauth.com/oauth2-servers/server-side-apps/authorization-code/.

Barker, D. 2016. Web Content Management. Systems, features, and best practices. O'Reilly Media Inc. Viitattu 5.10.2022. https://www.oreilly.com/library/view/web-content-management/9781491908112/ch01.html, O'Reilly learning platform.

Bertoli, M. 2017. React Design Patterns and Best Practices. Birmingham UK: Packt Publishing Ltd. Viitattu 7.10.2022. https://www.packtpub.com/product/react-design-patterns-and-best-practices/9781786464538. Packt.

Components State. N.d. React JS -verkkodokumentaatio. Viitattu 12.10.2022. https://reactjs.org/docs/ faq-state.html#gatsby-focus-wrapper.

Components and Props. N.d. React JS -verkkodokumentaatio. Viitattu 12.10.2022. https://reactjs.org/docs/components-and-props.html.

Complete solution. N.d. i18next-verkkodokumentaatio. Viim. muutos 1.10.2022. Viitattu 1.11.2022. https://www.i18next.com/#complete-solution.

Django introduction. N.d. Artikkeli MDN Web Docs sivustolla. Viim. muutos 9.10.2022. Viitattu 7.10.2022. https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction.

Django documentation. N.d. Django-verkkodokumentaatio. Viitattu 15.11.2022. https://docs.djangoproject.com/en/4.0/.

Django REST framework. N.d. Django REST Framework -verkkodokumentaatio. Viitattu 9.10.2022. https://www.django-rest-framework.org/.

Dughila, D. 2020. Elements of MVC in React. Julkaisu medium.com blogissa. Viitattu 7.10.2022. https://medium.com/swlh/elements-of-mvc-in-react-9382de427c09.

Feature Overview. N.d. React Router -verkkodokumentaatio. Viitattu 12.10.2022. https://reactrouter.com/en/main/start/overview.

Gaumgartner, R. 2021. How React and Redux brought back MVC and everyone loved it. Julkaisu rangle.io blogissa. Viitattu 7.10.2022. https://rangle.io/blog/how-react-and-redux-brought-back-mvc-and-everyone-loved-it/.

Getting started with Redux Toolkit. N.d. Redux Toolkit -verkkodokumentaatio. Viim. muutos 4.4.2022. Viitattu 9.10.2022. https://redux-toolkit.js.org/introduction/getting-started.

Getting started. N.d. Django OAuth Toolkit -verkkodokumentaatio. Viim. muutos 4.10.2022. Viitattu 9.10.2022. https://django-oauth-toolkit.readthedocs.io/en/latest/getting_started.html#.

Haberman, J. 2014. React Demystified. Julkaistu blogissa 22.2.2014. Viitattu 9.10.2022. https://blog.reverberate.org/2014/02/react-demystified.html.

Hunt, P. 2013. Why did we build React? Julkaisu reactjs.org blogissa. Viitattu 7.10.2022. https://reactjs.org/blog/2013/06/05/why-react.html.

Husbands, J. 2021. The complete guide to Figma and why it's the future of design. Julkaistu Bootcamp blogissa. Julkaistu 25.10.21. Viitattu 1.11.2022. https://bootcamp.uxdesign.cc/the-complete-guide-to-figma-and-why-its-the-future-of-design-a82839d3b5b9.

Järvinen, L. 2020. React Native sovellus Redux Sagoilla. Julkaistu medium.com blogissa. Viitattu 7.10.2022. https://medium.com/rnd-works/react-native-sovellus-redux-sagoilla-94e31cb76c15.

Ojasalo, K., Moilanen, T. & Ritalahti, J. 2015. Kehittämistyön menetelmät. Uudenlaista osaamista liiketoimintaan. 4. uud. p. Helsinki: Sanoma Pro Oy. Viitattu 5.10.2022. https://janet.finna.fi/Record/jamk.993548674806251, Ellibs.

Ravindran, A. 2018. Django Design Patterns and Best Practices: Industry-Standard Web Development Techniques and Solutions Using Python. 2. uud. p. Birmingham UK: Packt Publishing Ltd. Viitattu 7.10.2022. https://ebookcentral-proquest-com.ezproxy.jamk.fi:2443/lib/jypoly-ebooks/reader.action?docID=5405693&query=. Ebook Central.

Roldan, C. 2017. React 17 Design Patterns and Best Practices. 3. uud. p. Birmingham UK: Packt Publishing Ltd. Viitattu 7.10.2022. https://www.packtpub.com/product/react-17-design-patterns-and-best-practices/9781800560444. Packt.

Virtual DOM and Internals. N.d. React JS -verkkodokumentaatiossa. Viitattu 9.10.2022. https://reactjs.org/docs/faq-internals.html.

What is OAuth 2.0?. N.d. Artikkeli auth0 sivustolla. Viitattu 9.10.2022. https://auth0.com/intro-to-iam/what-is-oauth-2/.

What is react-i18next?. N.d. react-i18next -verkkodokumentaatio. Viitattu 9.19.2022. https://react.i18next.com/#what-is-react-i18next.

Weck, S. 2017. Developing modern offline apps with ReactJS, Redux and Electron – Part 3 – ReactJS + Redux. Codecentric-blogissa 3.12.2017. Viitattu 9.10.2022. https://blog.codecentric.de/developing-modern-offline-apps-reactjs-redux-electron-part-3-reactjs-redux-basics.