jamk

Foorumialustan kehittäminen

Joel Hämäläinen

Opinnäytetyö, AMK
Joulukuu 2024
Tieto- ja viestintätekniikka

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Kuvailulehti

Hämäläinen, Joel

Foorumialustan kehittäminen

Jyväskylä: Jyväskylän ammattikorkeakoulu. Joulukuu 2024, 30 sivua.

Tieto- ja viestintätekniikan tutkinto-ohjelma. Opinnäytetyö AMK.

Julkaisun kieli: suomi

Julkaisulupa avoimessa verkossa: kyllä

Tiivistelmä

Verkossa käydään jatkuvasti keskusteluja lukemattomista eri aiheista erilaisia sivustoja ja sovelluksia käyttäen. Verkkokeskustelulla on pitkät perinteet. Paikkoja sähköiselle keskustelulle löytyy sosiaalisen median jäteistä aina pienten piirien yhden aiheen foorumeihin. Keskustelua varten kehitetään ja on kehitetty paljon erilaisia alustoja. Verkkosovellusten kehittämistä varten on saatavilla runsaasti erilaisia teknologioita ja menetelmiä. Sovelluskehityksessä on tärkeä tuntea valittujen teknologioiden ominaisuudet ja niiden soveltuvuus haluttuun käyttötarkoitukseen.

Työssä kehitettiin Django ja React ohjelmistoja käyttäen yksinkertainen keskustelupalsta verkkosovellus, sisältäen valittuja keskustelupalstalle ominaisia toiminnallisuuksia, kuten viestien lähettäminen ja käyttäjätilille kirjautuminen. Sovelluksen avulla pyrittiin selvittämään Djangon ja Reactin vaikutusta sovelluksen kehittämiseen.

Tuloksena oli toimiva viestien lähettämiseen soveltuva verkkosovellus, jossa voidaan käydä keskusteluja useissa viestiketjuissa samanaikaisesti. Sovellukseen toteutettiin rekisteröityminen ja kirjautuminen, ja mahdollisuus käyttäjän poistaa viestejään ja ylläpitäjän poistaa kaikkien viestejä kehitettiin.

Djangon ja Reactin todettiin soveltuvan käyttötarkoitukseensa, ja että näiden teknologioiden tarjoamat työkalut olivat hyödyllisiä toiminnallisuuksien kehittämisessä. Sovelluksen yksinkertaisuuden ja pelkistettyjen toiminnallisuuksien takia todettiin, että sovelluksessa on runsaasti jatkokehitettävää voidakseen aidosti toimia julkisessa verkossa.

Avainsanat (asiasanat)

ohjelmistokehitys, keskustelupalstat, Django, React

Muut tiedot (salassa pidettävät liitteet)

jamk

Description

Hämäläinen, Joel

Forum platform development

Jyväskylä: JAMK University of Applied Sciences, December 2024, 30 pages.

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

Permission for open access publication: Yes

Language of publication: Finnish

Abstract

Discussions are constantly taking place online on countless different topics using various sites and applications. Online discussion has a long tradition. There are places for electronic discussion from social media to small circles' single-topic forums. Many different platforms are being developed and have been developed for discussion. There are many different technologies and methods available for developing web applications. In application development, it is important to know the properties of the selected technologies and their suitability for the desired purpose.

A simple discussion board web application was developed using Django and React software, including selected functionalities specific to a discussion board, such as sending messages and logging in to a user account. The application was used to investigate the impact of Django and React on the development of the application.

The result was a functional web application suitable for sending messages, where discussions can be held in several message threads simultaneously. Registration and logging in to the application was implemented, and also the option for the user to delete their messages and the administrator to delete all messages was developed.

Django and React were found to be suitable for their intended use, and the tools provided by these technologies were useful in developing the functionality. Due to the simplicity of the application and the reduced functionality, it was found that there is a lot of room for further development in the application to truly operate on the public network.

Keywords/tags (subjects)

software development, message boards, Django, React

Miscellaneous (Confidential information)

Sisältö

Kuviot

1 Johdanto

Keskusteleminen ja viestien lähettäminen verkon kautta on ollut olennainen osa internetiä sen alkuajoista saakka. Keskusteluja varten on vuosien aikana kehitetty valtava määrä alustoja ja alustojen kehittämiseen on luotu valtava määrä työkaluja. Erilaisille keskusteluaiheille musiikista ja urheilusta kalligrafiaan ja maanviljelyyn löytyy verkosta omat alustansa, joissa samoista asioista kiinnostuneet voivat kokoontua aiheensa ääreen, enemmän tai vähemmän asiallisissa merkeissä. Palstat ja sovellukset voivat olla hyvin suuria, useita aihealueita kattavia sivustoja, tai yhden asian pienen harrastajajoukon keskittymiä.

Oli keskustelualue millainen tahansa, se on kehitetty, rakennettu, käyttäen aina milloinkin sopivaksi todettuja työkaluja ja sovelluskehityksen metodeja. Oikeiden teknologioiden valitseminen on sovelluksen toiminnan, kehitettävyyden ja ylläpidon kannalta tärkeää. Erilaisten teknologioiden valtaisen määrän takia päätösten tekeminen vaatii tuntemusta sekä itse ohjelmistoista, kuin kehittäjien osaamistasosta näiden käyttämisessä. Foorumisovelluksen kohdalla on erityisen tärkeää valita oikeat tavat muun muassa tietojen säilytykseen ja käyttäjänäkymän toteutukseen.

Opinnäytetyön tarkoituksena on kehittää yksinkertainen keskustelupalstan tyylinen verkkosovellus, jota voi käyttää viestien ja viestiketjujen luomiseen. Työhön kuuluu sovelluksen teknisen toteuttamisen lisäksi myös teknologiavalintojen pohtiminen, ominaisuuksien määritteleminen ja työn toteutuksen onnistumisen arvioiminen. Lopputuloksena on kuvaus toimivan sovelluksen kehittämisestä ja sen aikana tehtävistä valinnoista. Pohdinnoilla ja teknologiavalintojen punnitsemisella on tarkoituksena osoittaa, kuinka laaja kokonaisuus sovelluksen kehittäminen voi olla ja miten selvitä valtavasta valintojen määrästä, joka lähes auttamatta on edessä tyhjästä alkavassa kehitystyössä.

Työn osana toteutettu sovellus toimii esimerkkinä valituilla teknologioilla ja ominaisuuksiensa puolesta tietyin prioriteetein toteutetusta keskustelupalstasta, jolla voidaan esitellä sovelluksen ominaisuuksia ja käytössä olevien teknologioiden toimintaa. Ensisijaiset toiminnot ovat sovelluksen yleinen toiminta, viestien lähettäminen ja poistaminen, tietojen tallentaminen ja hallinta, käyttöliittymän selkeys, käyttäjänhallinta, ylläpidettävyys ja jatkokehitettävyys. Sovelluksessa käytetään React-ohjelmistokehystä käyttöliittymään, Django-kehystä rajapintaan ja PostgreSQL-tietokantaa.

2 Tutkimusasetelma

2.1 Tutkimuskysymykset

Työssä pyritään vastaamaan kysymykseen:

1. Miten valittujen teknologioiden, eli Djangon ja Reactin, käyttäminen vaikuttavat sovelluskehityksen sujuvuuteen.

Kysymykseen pyritään vastaamaan sovelluskehittäjän näkökulmasta käyttäen esimerkkisovellusta esitellen siinä käytettäviä teknologioita ja niiden tarjoamia rakenteita ja tapoja, joilla sovellus voidaan kehittää.

2.2 Menetelmät

Työ on muodoltaan tutkimuksellinen kehittämistyö, jossa keskitytään toimivan sovelluksen rakentamiseen ja kehitysprosessin tutkimiseen. Oleellista on arvioida teknologiavalintojen vaikutusta projektin etenemisessä kuten, kuinka Djangon tarjoamat rakenteet ja valmiit komponentit sujuvoittavat kehitystyötä. Työ käyttää myös laadullisen tutkimuksen menetelmiä pyrittäessä ymmärtämään sovelluskehitystä kokonaisuutena.

tutkimuksellinen kehittämistyö on tutkimuksen ja konkreettisen asian, kuten verkkosovelluksen, kehittämisen yhdistävä työmuoto. Kehittämistyön tulos voi uusi tuote tai palvelu. Tutkimuksellisessa kehittämistyössä tutkimusta ja tiedonhankintaa ohjaa käytäntö, jota tutkimukselliset menetelmät tukevat ja toimivat apuvälineinä (Toikko & Rantanen 2009, 20-22).

Työn osana toteutetun esimerkkisovelluksen tarkoituksena oli esitellä sen ominaisuuksia ja arvioida sovelluksen kehitysprosessia, sekä Djangon ja Reactin soveltuvuutta sovelluksen kehittämiseen. Lisäksi tarkoituksena oli pohtia kehitettäväksi valittujen toiminnallisuuksien toimintaa ja osuutta Foorumialustan kokonaisuudessa, sekä mahdollisia jatkokehitysideoita. Jatkokehitysideoita voivat olla esimerkiksi toteuttamattomien toiminnallisuuksien lisääminen sovellukseen ja olemassa olevien toiminnallisuuksien parantaminen.

3 Tietoperusta

3.1 Keskustelupalstat

Foorumit eli keskustelupalstat ovat verkossa toimivia sivustoja ja sovelluksia, joiden kautta käyttäjät voivat keskustella. Keskustelupalstat tarjoavat ympäristön avoimelle sähköiselle kanssakäymiselle ja tiedonjakamiselle. Keskustelupalstan keskustelu saattaa keskittyä yhden ilmiön tai asian, kuten harrastuksen tai muun kiinnostuksen kohteen ympärille. Suuremmilla alustoilla saattaa olla useita osioita eri aiheille. Keskustelupalstoilla samasta asiasta kiinnostuneet ihmiset voivat käydä keskustelua, neuvoa muita, sekä saada itse apua aiheeseen liittyvissä asioissa, kuten harrastelaitteiden ja -välineiden käytössä.

Foorumit ovat yksi sosiaalisen median tyypeistä. Aichner ja Jacob (2015) jakavat sosiaaliset mediat kolmeentoista eri tyyppiin, jossa foorumit ovat oma ryhmänsä erottaen ne esimerkiksi blogeista, ”sosiaalisten verkostojen” alustoista, kuten Facebookista ja muista sisällönjaon alustoista. Keskustelupalstat eroavatkin näistä esimerkiksi viestien pituuden ja keskusteluiden pituuden puolesta; Foorumeilta saattaa löytää vuosia vanhoja, edelleen aktiivisia, viestiketjuja. Tältä osin keskustelupalstat erottuvat myös pikaviestimistä.

Tekniseltä toimintaperiaatteeltaan keskustelupalstat ovat samankaltaisia monien muiden web-sovellusten kanssa. Tietoja kirjoitetaan, luetaan, päivitetään ja poistetaan. Keskustelupalstan tapauksessa nämä operaatiot kohdistuvat tietokantaan ja tiedot ovat lähetettyjä viestejä ja käyttäjätietoja. Keskustelupalstan toimintoihin kuuluvat käyttäjän rekisteröiminen ja kirjautuminen, viestien lähettäminen ja mahdollisesti käyttäjäprofiilin muokkaaminen, esimerkiksi profiilikuvan lisääminen.

Koska keskustelupalstoilla sisältö on pääosin käyttäjien luomaa, palstoja tavallisesti täytyy moderoida, jottei palstoilla esiintyisi kiellettyä tai muuten epähaluttua materiaalia. Moderointi voidaan toteuttaa moderaattorien avulla. Moderaattori on henkilö, joka valvoo keskustelua. Moderaattorin tehtäviin voi kuulua keskustelujen siirtämistä palstan osiosta toiseen, keskustelujen sulkeminen, keskustelujen sekä viestien poistaminen ja käyttäjien estäminen (Damewood 2024). Palstalla voi olla käytössä jonkinlainen ilmiantomekanismi, myös automatisointia, kuten viestien suodattamista, joka estää kiellettyjä sanoja sisältävien viestien lähettämisen, voidaan käyttää.

3.2 Web-sovellukset

3.2.1 Front- ja backend

Verkon kautta toimintaansa tarjoavia palveluita kutsutaan myös web-sovelluksiksi. Web-sovelluksia ovat muun muassa verkkokaupat ja sosiaalisen median alustat. Web-sovelluksen toiminta on jaettu usein ns. frontend- ja backend-puoliin toimintojen luonteen mukaan. Frontend-osat ovat ns. asiakaspuolen komponentteja toimien käyttäjän selaimessa ja liittyen käyttäjän näkemään osaan sovelluksesta, kuten ulkonäköön. frontend-teknologioita ovat esimerkiksi HTML, CSS ja JavaScript (What is a Front-End Developer? n.d.). Backend tai palvelinpuoli liittyy sovelluksen käyttäjältä piilossa olevaan toimintaan, kuten tietojen tallentamiseen ja käsittelyyn. Kirjautuminen, yhteyden ottaminen tietokantaan ja tiedostojen käsittely ovat palvelinpuolen toimintoja (Backend n.d.).

Yksi tapa toteuttaa web-sovellus on käyttämällä web-ohjelmistokehyksiä. Ohjelmistokehykset tarjoavat valmiit työkalut, joilla toteuttaa tavallisimmat web-sovelluksen toiminnallisuudet (Server-side web frameworks n.d.). Kaikkia toimintoja ei näin tarvitse kehittää alusta alkaen tehden kehitystyöstä mahdollisesti nopeampaa ja koodista rakenteellisesti yhtenäisempää, joka voi parantaa sovelluksen ylläpidettävyyttä ja jatkokehitettävyyttä. Sekä frontend- että backend-kehitystä varten on tarjolla useita ohjelmistokehyksiä useille eri ohjelmointikielille. Sovelluksen voi halutessaan kehittää täysin yhdellä kielellä, kuten JavaScriptillä, jolle frontend-kehitys on luontevaa, mutta joka taipuu Node.js-ympäristöä käyttämällä myös backend-kehitykseen. Toisaalta kehittäjä voi valita sovelluksiin osiin eri kielet, esimerkiksi parhaan mahdollisen suorituskyvyn aikaansaamiseksi.

Front- ja Backend-teknologiat yhdistettynä muihin teknologioihin, joita ohjelmisto tai sovellus käyttää toimiakseen, kutsutaan ohjelmistopinoksi tai ratkaisupinoksi (software stack, solution stack). Ohjelmistopinon sisältö vaihtelee paitsi käytettyjen teknologioiden mukaan, myös projektin luonteen mukaan. Web-sovelluksen ohjelmistopino saattaa sisältää tietokannan, back- ja frontend-teknologiat, sekä ajoympäristön. Tällaista edustaa esimerkiksi MERN (MongoDB, Express, React, Node.js) -pino (MERN Stack Explained n.d.). Mukaan saatetaan myös lukea käytettävä tietokoneen käyttöjärjestelmä.

Perinteisen front- ja backend -jakoa noudattavien ohjelmistojen lisäksi on olemassa full-stack-kehyksiä ja kehitystyökaluja, joiden ideana on mahdollistaa koko sovelluksen rakentaminen yhtenä

pakettina. Esimerkiksi React-kehys Next.js on kehyksen dokumentaation Introduction (n.d.) -sivun mukaan tarkoitettu rakentamaan ”full-stack” web-sovelluksia. Myös opinnäyteyön osana kehitettävän Foorumialusta-sovelluksen kehityksessä käytettävä Django sisältää backend-ominaisuuksiensa lisäksi tavan genoroida HTML-dokumentteja Templates-työkalun avulla (Templates n.d.).

3.2.2 HTTP

Hypertext Transfer Protocol, eli HTTP, on dokumenttien välittämiseen verkossa käytettävä protokolla, joka on tarkoitettu verkkoselainten ja -palvelinten väliseen viestintään (HTTP 2024). HTTP on verkossa tapahtuvan tiedonsiirron peruspilareita. Palvelimet ja asiakasohjelmat, kuten selaimet, viestivät keskenään lähettämällä viestejä toisilleen. Selain lähettää palvelimelle pyynnön, esimerkiksi pyytäen jotain verkkosivua, johon palvelin vastaa lähettämällä halutun dokumentin, jos mahdollista. (An overview of HTTP 2024.)

HTTP-pyynnöt noudattavat tiettyä muotoa niiden tarkoituksen ja odotettavan vastauksen mukaan, joita kutsutaan metodeiksi. Yleisiä metodeja ovat muun muassa GET, POST ja DELETE. GET-metodilla selain pyytää saada dokumentin, ja palvelin vastauksena lähettää sen. POST-metodia käytetään dokumentin lähettämiseen selaimesta palvelimelle. Tällaista metodia voidaan käyttää esimerkiksi viestin lähettämisessä keskustelupalstalle. DELETE-metodia käytetään dokumentin poistamiseen. (HTTP request methods 2024.)

3.2.3 REST-rajapinta

Selain- ja palvelinpuolet ovat yhteydessä toisiinsa jonkinlaisen rajapinnan kautta. Ohjelmointirajapinta on mekanismi, joka mahdollistaa sovellusten välisen tiedonvaihdon. REST (Representational State Transfer) on yleisesti ohjelmointirajapinnoissa käytetty arkkitehtuurityyli. REST-rajapinnoissa kommunikointi tapahtuu esimerkiksi HTTP-kyselyiden välityksellä. Kyselyillä voidaan luoda, lukea, päivittää ja poistaa aineistoa. Asiakas, selainpuoli, tekee kyselyitä palvelinpuolelle, joka toteuttaa kyselyiden komennot ja lähettää asiankuuluvat vastauksen asiakkaalle. Tieto voi rajapinnassa liikkua lähestulkoon missä formaatissa tahansa, kuten JSON-, HTML- tai XML-muodoissa (What is a REST API? n.d.).

Vaikka REST-rajapinnat ovat yleisiä verkkosovelluksissa, ja on yleistä, että rajapinta käyttää HTTP-protokollaa, REST ei määrittele, mitä teknologioita tai protokollia rajapintojen toteuttamisessa täytyy käyttää, vaan se esittää tietynlaiset ääriviivat, miten REST-rajapinnaksi kutsuttava rajapinta toimii. Jotkin materiaalit, kuten IBM-sivuston What is a REST API (n.d.), antavat ymmärtää REST-rajapinnan käyttävää nimenomaan HTTP-protokollaa kommunikointiinsa, kun taas MDN Web docs -dokumenttien REST (2023) -artikkeli kertoo, etteivät kaikki HTTP-rajapinnat noudata REST-rajoitteita, painottaen kuitenkin asian olevan aloittelijalle melko triviaalia tietoa.

Felding (2008), joka alun perin kuvasi REST-rajapinnan, esittää blogissaan tyytymättömyytensä tapaan väärinkäyttää REST-nimitystä, koska HTTP-protokollaa käyttäviä rajapintoja saatetaan nimittää REST-rajapinnoiksi, vaikkeivät ne noudattaisivatkaan sille asetettuja rajoitteita. REST-rajapinnan ominaisuus, että rajapinnan tarjoamat tietoa sisältävät resurssit, kuten HTML-dokumentit, sisältävät hypertekstiä, eli tietokoneella esitettävää muihin dokumentteihin linkkejä sisältävää tekstiä, on Feldingin (2008) mukaan täytyttävä, jotta rajapintaa voidaan pitää REST-rajapintana.

3.2.4 Tietokannat

Keskustelupalstat ja muut web-sovellukset useimmiten tarvitsevat tietokannan, johon tallentaa sovelluksessa käsiteltävät tiedot, kuten käyttäjätilin tiedot ja luodut sisällöt, sekä mahdollisten tuotteiden tiedot. Tietokanta on yhteydessä sovelluksen palvelinosaan. Palvelinsovellus muodostaa tietokanta yhteyden ja lukee, tallentaa, päivittää tai poistaa sieltä tietoja, niin kuin käyttäjä sen käskee sen tekemään. Keskustelupalstan tapauksessa tämä voi tarkoittaa esimerkiksi viestin lähettämistä; käyttäjä lähettää kirjoittamansa viestin lähetä-painikkeella, sovelluksen selainosa lähettää viestin ja muut tarvittavat tiedot rajapinnan kautta palvelinosaan, joka tietyin askelin tallentaa viestin tietokantaan. Vastaavasti samainen viesti voidaan tietokannasta hakea palstalla esitettäväksi.

Tietokantoja on olemassa useanlaisia. Relaatiotietokannat, kuten MySQL ja opinnäytetyön osana kehitettävässä Foorumialustassa käytettävä PostgreSQL ovat suosittuja tietokantaohjelmia. Relaatiomallia toteuttamattomat, jonkin muun periaatteen mukaan toimivat tietokannat, kuten dokumenttityyppinen MongoDB ovat myös yleisiä. Tietokantajärjestelmien suosiota mittaava DB-

Engines Ranking (2024) listaa viideksi suosituimmaksi tietokannaksi jo mainittujen lisäksi Oraclen ja Microsoft SQL Serverin.

PostgreSQL on avoimen lähdekoodin relaatiotietokantajärjestelmä. Se käyttää SQL (Structured Query Language) -kieltä tietojen käsittelyyn. Järjestelmässä painottuvat skaalautuvuus, turvallisuus, datan eheys, sekä moninaisten ominaisuuksien, työkalujen ja lisäosien saatavuus. PostgreSQL on saatavilla useille käyttöjärjestelmille, kuten Windowsille, Linuxille ja MacOS:lle. (About n.d.)

3.3 React

React on Facebookin kehittämä JavaScript-kirjasto, jolla voidaan toteuttaa sovellusten käyttöliittymiä. Kirjasto käyttää käyttöliittymän, käyttäjän näkemän sivun, päivittämiseen virtual DOM:ia (Document Object Model). DOM kuvaa dokumentin, kuten verkkosivun HTML-dokumentin rakennetta puumaisesti. Virtual DOM on dokumentin varsinaisen DOM:n kanssa synkronisoitava, muistissa säilytettävä DOM-rakenne (Virtual DOM and Internals n.d.). Reactissa olennaisena osana ovat komponentit. Komponentti on käyttöliittymän osa, joka hallitsee omaa toimintaansa.

Käyttöliittymän rakentamiseen Reactissa käytetään komponentteja. React-komponentit ovat omaa toimintaansa hallitsevia, uudelleenkäytettäviä käyttöliittymän osia. Kutsuttaessa komponentit palauttavat sovelluksessa tai verkkosivulla näytettävää sisältöä, esimerkiksi kirjautumislomakkeen, joka on komponentissa esitettynä, usein JSX-muodossa, jota käsitellään myöhemmin tässä luvussa. (React Components n.d; Componentizing our React app 2023)

Komponentit esitetään koodissa joko luokkina (class) tai funktioina (function). Ennen versiota 16.8, ennen Hook-toiminnon lisäämistä, komponentit toteutettiin aina luokkien avulla (React Class Components n.d.). Nykyään funktiot ovat kuitenkin suositeltu tapa komponenttien toteuttamiseen, React-dokumentaation Component -ohje (n.d.), kuin myös MDN Web Docs -materiaalien React Resouces -sivu (2023) suosittelevat funktioiden käyttöä uudessa koodissa. Sekä luokka-, että funktiototeutuksia voi ohjelmistoissa tavata, etenkin vanhemmissa koodikannoissa (React Resources n.d.).

Hook on Reactissa käytetty toiminto, jolla voidaan muun muassa lisätä tila funktiokomponenttiin. Ne siis mahdollistavat funktioiden käytön komponenttien luomisessa luokkien sijaan. (Abramov 2019) Tätä käsiteltiin edellisessä luvussa. Hookeilla voidaan niin sanotusti tarttua tai ottaa kiinni, kuin koukku, komponentin tilasta, ja esimerkiksi käyttää tai muuttaa sitä (Hooks at a Glance n.d.). Erilaisia hookkeja löytyy React-kirjastosta valmiiksi useita, ja kehittäjä voi rakentaa niitä myös itse omiin tarpeisiinsa. Sisäänrakennettuja hookkeja on esimerkiksi useState-hook, jolla voidaan muuttaa komponentissa olevaa muuttujaa, sekä useEffect-hook, jolla voidaan vaikuttaa johonkin ulkoiseen järjestelmään, kuten selaimen DOM:iin. (Built-in React Hooks n.d.)

JSX on Reactissa käytettävä JavaScript-ohjelmointikielen jatke, joka muistuttaa XML-merkintäkielen syntaksia. Sitä voidaan käyttää luomaan ja esittämään verkkosivun käyttöliittymän elementtejä HTML-dokumentin kaltaisessa rakenteessa käyttäen HTML:stä tuttuja tunnisteita. JSX:ää ei ole pakko käyttää React-kehityksessä, mutta se on yleistä. (Writing Markup with JSX n.d.) JXS on kehitetty Reactia varten, mutta sitä voidaan käyttää myös joissain muissa ohjelmistokehyksissä, kuten Vuessa (Render Functions & JSX n.d.). Tämä luku kuitenkin käsittelee JSX:n Käyttöä Reactissa.

Sisennetty rakenne, jolla JSX esittää käyttöliittymän, antaa sivun kehittäjälle tutun ja helposti ymmärrettävän rakenteen, johon luoda käyttöliittymän elementtejä (JSX 2022). Sisäkkäisellä rakenteella tarkoitetaan, että elementit muodostavat puurakenteen, jossa ylimpänä on yksi elementti, josta muut haarautuvat, ja alemmat tasot voivat myös sisältää alielementtejä. Sisäkkäinen rakenne liittyy myös JSX:n toimintaan, sillä JSX muutetaan tavallisiksi JavaScript-objekteiksi. Rakenteella on oltava aina vain yksi ylimmän kerroksen tunniste, jonka sisällä muut tunnisteet sijaitsevat. Funktio voi palauttaa useamman objektin vain, jos ne on pakattu yhteen rakenteeseen. Tällaisena Ylimpänä tunnisteena voidaan käyttää esimerkiksi HTML:ssä käytettävää div-elementtiä. (Writing Markup with JSX n.d.)

3.4 Django

Rajapintojen luomiseen on saatavilla useita ohjelmistokehyksiä ja työkaluja lukuisille eri ohjelmointikielille. Toteutetun foorumisovelluksen palvelinosa käyttää Python-kehys Djangoa. Python-kehyksellä tarkoitetaan, että kehystä käytettäessä sovellusta kehitetään Python-ohjelmointikielellä.

Django ominaisuuksiltaan laaja ohjelmistokehys, joka sisältää paljon valmiita rakenteita. Tämän vuoksi Django voi aluksi näyttäytyä monimutkaisena, erityisesti minimalistisempiin kehyksiin, kuten FastAPI:n verratessa. Toisaalta vähempine ominaisuuksineen FastAPI voi projektin laajetessa ja ominaisuuksien lisääntyessä käydä työlääksi kehittää. Django on myös turvaominaisuuksiltaan tätä kattavampi. (Django vs Fast API: Detailed Comparison 2022.)

Rakenteellisesti Django-ohjelma jakautuu kahdelle tasolle; projektiin (project) ja sovellukseen (app). Projektissa voi olla useita sovelluksia ja sovellusta on mahdollista käyttää useissa projekteissa. Projekti sisältää kaikki sivustolla käytettävät toiminnallisuudet ja sovellus taas, toteutuksen mukaan, yhden tai useamman näistä toiminnallisuuksista, kuten käyttäjätilin luomisen tai viestien tallentamisen tietokantaan. (Writing your first Django app, part 1 n.d.)

4 Foorumialustan toteuttaminen

4.1 Toteutus ja vaatimukset

Opinnäytetyön osana kehitettiin Foorumialustaksi kutsuttu yksinkertainen verkkosovellus, joka esittelee muutamia valittuja keskustelupalstalle tyypillisiä ominaisuuksia. Toiminnallisia ominaisuuksia ovat käyttäjätunnukset ja käyttäjien roolit (ylläpitäjät ja tavalliset käyttäjät), sisään- ja uloskirjautuminen, viestien lähettäminen ja poistaminen. Tavalliset käyttäjät voivat poistaa omia viestejään ja viestiketjujaan, kun taas ylläpitäjä voi poistaa kenen tahansa lähettämää sisältöä. Ei-toiminnallisia ominaisuuksia ovat Foorumialustan selkeä ulkonäkö, helppokäyttöisyys ja jatkokehitettävyys. Foorumi on kuitenkin luonteeltaan esimerkkiprojekti, jossa toiminnalliset ominaisuudet ovat suuremmassa roolissa.

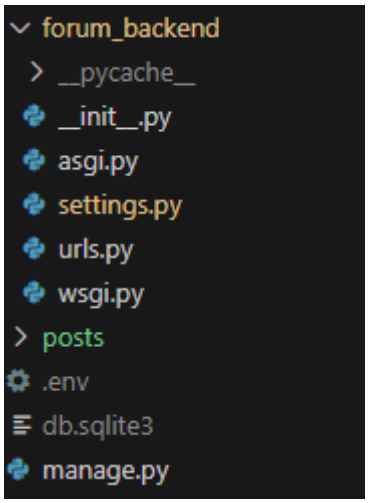
Foorumialustan toteuttaminen on pienoiskoon sovelluskehityshanke, johon kuuluvat sovelluksen palvelin- ja käyttöliittymäpuolien suunnittelu ja kehittäminen toiminnallisuuksineen sekä toimivaan kokonaisuuteen pyrkiminen niin sovelluksessa kuin kehitystyössä. Käytössä olivat Django versio 5.1 ja React versio 18.3.1.

4.2 Backend

Foorumialustan Djangolla kehitetty backend-osa koostuu projektista ja sovelluksesta. Projekti kattaa kokonaisuuden kaikkine asetuksineen ja konfiguraatioineen sekä sovelluksineen. Sovellus, joka on osa projektia, pitää sisällään tiettyjä ominaisuuksia: viestien ja viestiketjujen luomisen ja poistamisen, kirjautumisen ja rekisteröitymisen.

Forum_backend-projekti

Django-ohjelmistokehyksessä sovelluksen kehitystyö alkaa luomalla projekti ”django-admin startproject forum_backend” -komennolla. Komento luo forum_backend-nimisen projektin sisältäen tarvittavat tiedostot kehityksen aloittamiseen ja sovelluksen käynnistämiseen. Valmiit aloitustiedostot ohjaavat kehittäjää noudattamaan tietynlaista tiedostorakennetta sovelluksessa. Kuviossa 1 esitetään Foorumialustassa käytetyn projekti tiedostorakennetta. Projektiin vielä luotiin posts-niminen sovellus (app) käyttäen ”python manage.py startapp posts” -komentoa projektihakemistossa.



Kuvio 1. Forum_backend-projektin tiedostorakenne.

Projektissa käytettiin Django REST Framework -sovellusta, joka tarjoaa työkaluja REST-mallisen rajapinnan luomiseen, sekä kyseiseen sovellukseen saatavaa Simple JWT -lisäosaa käyttäjien todentamista varten JSON Web Tokeneilla. Käyttöön otettiin myös Django-cors-headers-sovellus mahdollistamaan resurssien jakaminen projektin ulkopuolelle, siis tietojen lähettäminen

Foorumialustan front- ja backendin välillä. Tällaisesta tietojen jakamisesta käytetään nimeä CORS, eli Cross-origin resource sharing.

Käytettävät sovellukset asennettiin ympäristöön ja lisättiin forum_backend-projektin settings.py-tiedoston INSTALLED_APPS-listaan, jossa luetellaan projektissa käytettävät sovellukset.

```
INSTALLED_APPS = [
    # …
    'rest_framework',
    'posts',
    'corsheaders',
    'rest_framework_simplejwt',
    'rest_framework_simplejwt.token_blacklist',
]
```

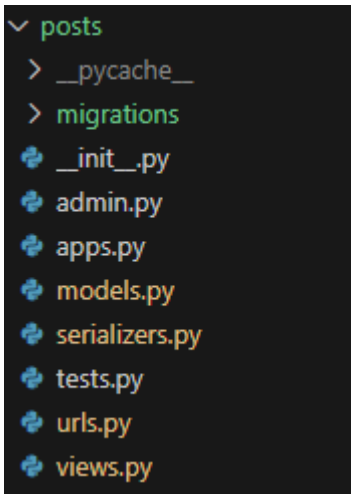
Lisäksi samaiseen tiedostoon luotiin REST_FRAMEWORK-lista, johon määritettiin todennuksessa käytettäväksi Simple JWT:tä, sekä annettiin lupa kaikkien lukea foorumin sisältöä.

```
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'rest_framework_simplejwt.authentication.JWTAuthentica-
tion',
    ),
    'DEFAULT_PERMISSION_CLASSES': (
        'rest_framework.permissions.AllowAny ',
    ),
}
```

Useita muitakin projektin asetuksia voidaan muokata ja määritellä settings.py -tiedoston kautta. Monet asetukset Foorumialustalla jätettiin oletuksiinsa. Esimerkiksi salasanojen salausasetuksiin ei koskettu. Tietokanta-asetuksia täytyi kuitenkin muuttaa. Django Käyttää oletuksena SQLite-tietokantaa, jossa tietokanta tallennetaan yhteen tiedostoon, eikä sen käyttämiseen tarvita erillistä tietokannan hallintasovellusta. PostgreSQL-tietokannan käyttämiseksi tarvittiin Psycopg2 -Python-kirjastoa ja forum_backend-projektin settings.py -tiedoston tietokantaa koskevien tietojen muuttamista. Tietokannan taulujen ja relaatioiden luominen tapahtuu Django-sovelluksen kautta Python-luokkia käyttäen, joten tietokantaa ei yhteyden muodostamisen jälkeen tarvinnut säätää.

Posts-sovellus

Foorumialustassa palvelinpuolen toiminnallisuudet ovat yhdessä sovelluksessa. Toiminnot kuten käyttäjätilit, kirjautuminen, viestien lähettäminen ja poistaminen sisällytettiin siis samaan posts-sovellukseen. Kuviossa 2 esitetään Foorumialustan posts-sovelluksen tiedostorakenteen.



Kuvio 2. Posts-sovelluksen tiedostorakenne.

Posts-sovelluksessa viestien ja viestiketjujen sisältö määritellään models.py-tiedostossa. Tiedostoon luotiin Thread (viestiketju) ja Post (viesti) -nimiset luokat, joihin määriteltiin, millaisista tiedoista ne koostuvat ja miten ne tallennetaan tietokantaan.

```
class Thread(models.Model):
    title = models.CharField(max_length=255)
    created_at = models.DateTimeField(auto_now_add=True)
    user = models.ForeignKey(User, on_delete=models.CASCADE)

class Post(models.Model):
    content = models.TextField()
    thread = models.ForeignKey(Thread, related_name="posts",
on_delete=models.CASCADE)
    created_at = models.DateTimeField(auto_now_add=True)
    user = models.ForeignKey(User, on_delete=models.CASCADE)
```

Viestiketjun tietoja ovat nimi, luontiaika ja ketjun luonut käyttäjä. Viesti sisältää lähetetyn tekstin, mihin viestiketjuun viesti kuuluu, luontiajan ja viestin luoneen käyttäjän tiedot. Luontiaika lisätään

viestiin ja ketjuun automaattisesti. User, eli käyttäjämuuttujassa on viiteavain Djangon sisäänrakennettuun User-käyttäjätietoluokkaan, jota Foorumialusta käyttää käyttäjien hallinnassa ja luomisessa. Samalla tavalla viestin thread-muuttujassa viitataan viiteavaimella Thread-luokkaan. Jos viestiketju poistetaan tietokannasta, myös ketjuun kuuluvat viestit poistetaan. Samoin jos käyttäjätili poistetaan, käyttäjän luomat viestit ja viestiketjut poistuvat.

Django luo taulut tietokantaan models.py-tiedoston luokkien mukaisesti ”python manage.py makemigrations” ja ”python manage.py migrate” -komennoilla, jotka luovat tietokannan ja sen taulut, sekä tarvittaessa tekevät niihin muutoksia. ”Makemigrations” määrittää tehtävät muutokset ja ”migrate” ajaa ne tietokantaan. Komennot tulee suorittaa tietokantaa koskevia muutoksia tehdessä. Muutokset tallentuvat python-tiedostoina sovelluksen migrations-kansioon. (Writing your first Django app, part 2. n.d.)

Tietojen serialisointia varten käytettiin Django REST Frameworkin sisältämiä serialisointityökaluja. Serialisoinnilla tarkoiteteaan tiedon muuttamista tallentamista tai lähettämistä varten sopivaan muotoon (Serialization 2023). Posts-sovellukseen luotiin serializers.py-tiedosto, jonne määritettiin serialisointiluokat. Luokat käyttävät valmista ModelSerializer-luokkaa. Esimerkkinä käyttäjien rekisteröinnissä käytettävä UserSerializer-luokka, joka käyttää Djangon User-käyttäjämallia, ja tallennettavat kentät, id-tunniste, käyttäjänimi ja salasana, ovat listana.

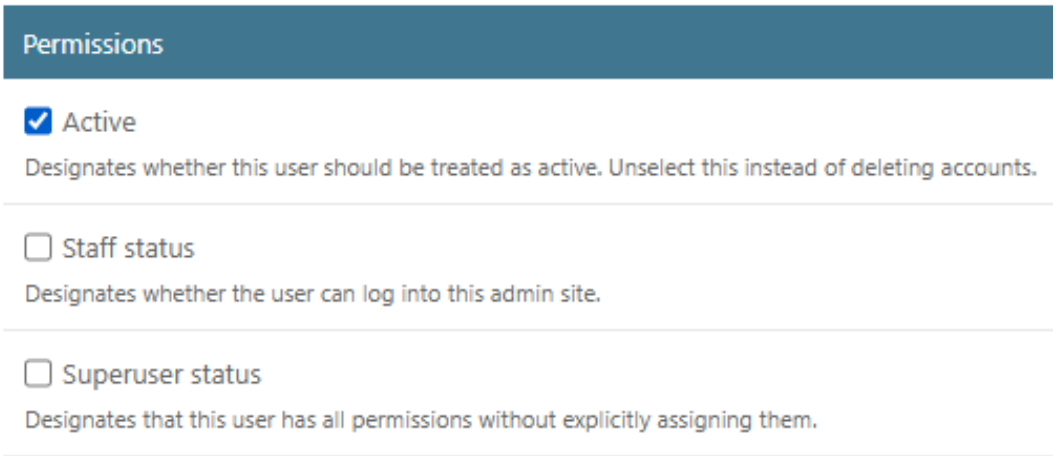
```
class UserSerializer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = ['id', 'username', 'password']
    def validate_password(self, value):
        return make_password(value)
```

Serialisointiluokkia käytetään näkymissä (views), kuten käyttäjän rekisteröityessä. Foorumialustan näkymiä ovat käyttäjän rekisteröinti, kaikki viestiketjut ja ketjun luominen, yksittäinen viestiketju ja sen poistaminen, viestit viestiketjussa ja viestin lähettäminen, sekä yksittäinen viesti ja sen poistaminen. Lisäksi käytettiin Simple JWT:n TokenObtainPairView-näkymää sisäänkirjautumiseen. Näkymiä pääsee käyttämään rajapinnan kautta; sovelluksen ja projetkin urls.py-tiedostot määrittävät rajapinnan URL-päätteet ja jokaiseen päätteeseen liittyvän näkymän. Käyttäjätilin rekisteröiminen

tapahtuu ”auth/registers/” -URL:sta, josta on pääsy RegisterUserView-näkymään, joka käyttää UserSerializer-serialisointiluokkaa. Samaa logiikkaa noudattavat muutkin toiminnot.

```
urlpatterns = [
    # …
    path('auth/register/', RegisterUserView.as_view(), name='re-
gister_user'),
]
class RegisterUserView(generics.CreateAPIView):
    queryset = User.objects.all()
    serializer_class = UserSerializer
```

Foorumialustassa on kahdenlaisia käyttäjiä: tavallisia käyttäjiä ja ylläpitäjiä. Käyttäjätilit käyttävät Djangon sisäänrakennettua käyttäjätilijärjestelmää, joten tilien osalta sovellusta ei ollut tarvetta kehittää viestien tapaan alusta asti. Myös käyttäjätyypit löytyvät valmiina. Kuviossa 3. esitetään tavallisen käyttäjän statustiedot Djangon hallintapaneelissa. Tavalliselle käyttäjälle voidaan antaa ylläpitäjän status, jolloin se voi poistaa muiden käyttäjien viestejä. Superuser-käyttäjällä on kaikki mahdolliset oikeudet ilman, että niitä tarvitsee erikseen määrittää. Staff-käyttäjälle on erikseen määritettävä, voiko se esimerkiksi poistaa muita käyttäjiä tai vaihtaa salasanoja



Kuvio 3. Käyttäjän statuksen määrittäminen Djangon hallintapaneelissa.

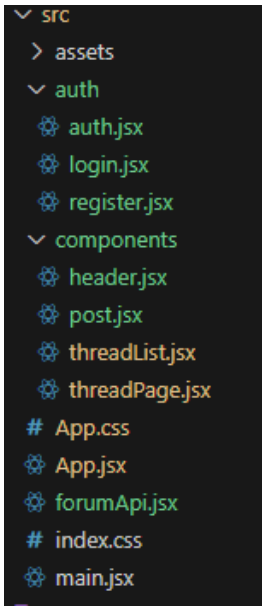
Jotta vain asianomaiset käyttäjät voisivat poistaa viestejä, sovelluksen lupajärjestelmää tarkennettiin. Sovellukseen luotiin IsOwnerOrAdmin-luokka. Niin kutsutuilla ”Custom permissioneilla” ohitetaan Django REST Frameworkin BasePermission-luokka (Permissions n.d.). Luokkaan tehtiin

has_object_permission-funktio, joka sisältää uudet käytännöt. Jos HTTP-metodi on ”turvallinen” (GET, HEAD, OPTIONS), pyyntö hyväksytään, eli kuka tahansa voi katsella sisältöä. Ylläpitäjälle sallitaan kaikki metodit. Myös, jos pyynnön kohde on oma viesti tai -ketju, metodit ovat sallittu.

```
class IsOwnerOrAdmin(permissions.BasePermission):
    def has_object_permission(self, request, view, obj):
        if request.method in permissions.SAFE_METHODS:
            return True
        if request.user.is_staff:
            return True
        return obj.user == request.user
```

4.3 Frontend

Käyttöliittymän rakentaminen Reactilla alkoi pystyttämällä projekti Vite-kehityspalvelimella komennolla ”npm create vite@latest”. Käyttäjältä kysytään projektin nimeä ja käytettäviä teknologioita. Projekti nimettiin frontendiksi ja teknologioiksi valittiin React ja Javascript. Komento loi projektikansioon valmiiksi tiedostoja kehityksen aloittamista ja ohjelman käynnistystä varten. Frontend-projektin src-nimiseen kansioon luotiin omat kansiot erilaisia toimintoja varten. Kuviosta 4. voidaan todeta, että käyttäjätiliä koskevat toiminnot on sijoitettu auth-kansioon. Sivuston yläpalkista, viesteistä ja viestiketjuista vastaavat tiedostot sijaitsevat components-kansiossa.



Kuvio 4. Frontendin src-kansion rakenne.

Sovelluksen yhteen kokoava App.jsx-tiedosto pitää sisällään App-funktion, jonka palauttama JSX-elementti sisältää sovelluksen kaikki sivut React Router -kirjaston Router-elementissä. Sivuja ovat etusivu, yksittäinen viestiketju, sisäänkirjautuminen ja rekisteröityminen. Myös sivuston yläpalkki on lisätty elementtiin, jotta se näkyy kaikilla sivuilla. Yläpalkista löytyvät painike etusivulle sekä, sen mukaan onko käyttäjä kirjautunut sisään, painikkeet sisäänkirjautumiseen ja rekisteröitymiseen, tai uloskirjautumispainike sekä ”Logged: *käyttäjänimi*” -teksti.

```
<Router>
      <div>
        <Header />
        <Routes>
          <Route path="/" exact element={< ThreadList />} />
          <Route path="/thread/:id" element={< ThreadPage />} />
          <Route path="/login" element={< Login />} />
          <Route path="/register" element={< Register />} />
        </Routes>
      </div>
</Router>
```

React Routeria käytetään sovelluksen sivujen reitityksessä, sillä Foorumialusta on toteutukseltaan yhden sivun sovellus, jossa pysytään aina yhdellä sivulla, jonka sisältö vain muuttuu. Ilman Routeria kaikki näkymät olisivat aina samassa URL-osoitteessa, eivätkä esimerkiksi selaimen navigointinäppäimet toimisi oletettavalla tavalla.

Yhteydenottoihin Django-palvelimelle käytetään forumApi.jsx-tiedostoa. HTTP-kutsujen tekemistä varten sovellukseen valittiin käyttöön Axios HTTP-asiakasohjelma. Tiedosto määrittää URL-osoitteen, jonne palvelimelle menevät kyselyt suoritetaan. Se myös liittää kyselyiden tunnisteotsakkeeseen JWT-tunnisteen, jolla käyttäjä tunnistetaan. Tiedosto myös ohjaa käyttäjän sisäänkirjautumissivulle, jos tämä koittaa mennä sivulle, johon tällä ei ole oikeutta.

Auth.jsx-tiedoston funktiot huolehtivat JWT-tunnisteiden tallentamisesta ja niiden käyttämisestä, sekä käyttäjän uloskirjautumisesta. Tunnisteet tallennetaan selaimen muistiin. Sisään kirjautuessa käyttöliittymä tallentaa tunnisteen käyttäen auth-tiedoston setTokens-funktiota. Tallennettuja tunnisteita käytetään viestien lähettämisessä sekä viestin poistopainikkeen ja uloskirjautumispainikkeen näyttämisessä sivulla.

Sisäänkirjautuminen tapahtuu syöttämällä käyttäjän nimi ja salasana asianmukaisiin sarakkeisiin sisäänkirjautumissivulla. Kirjautumislomake toteutettiin JSX-elementtinä. Login-funktio tallentaa nimen ja salasanan käyttäen Reactin useState-hookkia. Käyttäjän painaessa kirjautumispainiketta tiedot lähetetään palvelimelle ”auth/login/”-osoitteeseen. Kirjautumisen onnistuessa käyttäjä saa palvelimelta vastauksena JWT-tunnisteen, joka tallennetaan selaimeen. Rekisteröinti toteutettiin samankaltaisella menetelmällä useState-hookkeja käyttäen. Rekisteröidyttyään käyttäjä ohjataan sisäänkirjautumissivulle.

Sovelluksen etusivu näyttää olemassa olevat viestiketjut. Viestiketjujen hakeminen palvelimelta tapahtuu useEffect-hookkia käyttäen. Se suorittaa HTTP GET-pyynnön, josta saadaan vastauksena viestiketjut. Ketjut lisätään useState-hookilla sivulle. Sivulla voidaan luoda uusi viestiketju kirjoittamalla tekstikenttään. Uusi ketju lähetetään POST-pyynnöllä palvelimelle ja sivun näkymä päivittyy lisäten uuden viestiketjun sivulle. Viestiketjusta painamalla voidaan siirtyä yksittäisen viestiketjuun, jolloin nähdään ketjuun lähetetyt viestit. Yksittäisen viestiketjun näkymä on toiminnaltaan samankaltainen, kuin etusivu. Ketjun sisältä, löytyy painike ketjun poistamiselle, jos käyttäjällä on oikeus niin tehdä. Samalla tavalla toimivat viestienpoistopainikkeet. Myös viestien lähetyslomake näkyy vain kirjautuneille käyttäjille. Painikkeiden ja lomakkeiden näkyminen toteutettiin ehdollisesti. Mikäli käyttäjän on kirjautunut ja tunnistenumero on sama, kuin ketjun tai viestin luojan tunniste, tai jos käyttäjä on ylläpitäjä, poistonappi on näkyvissä.

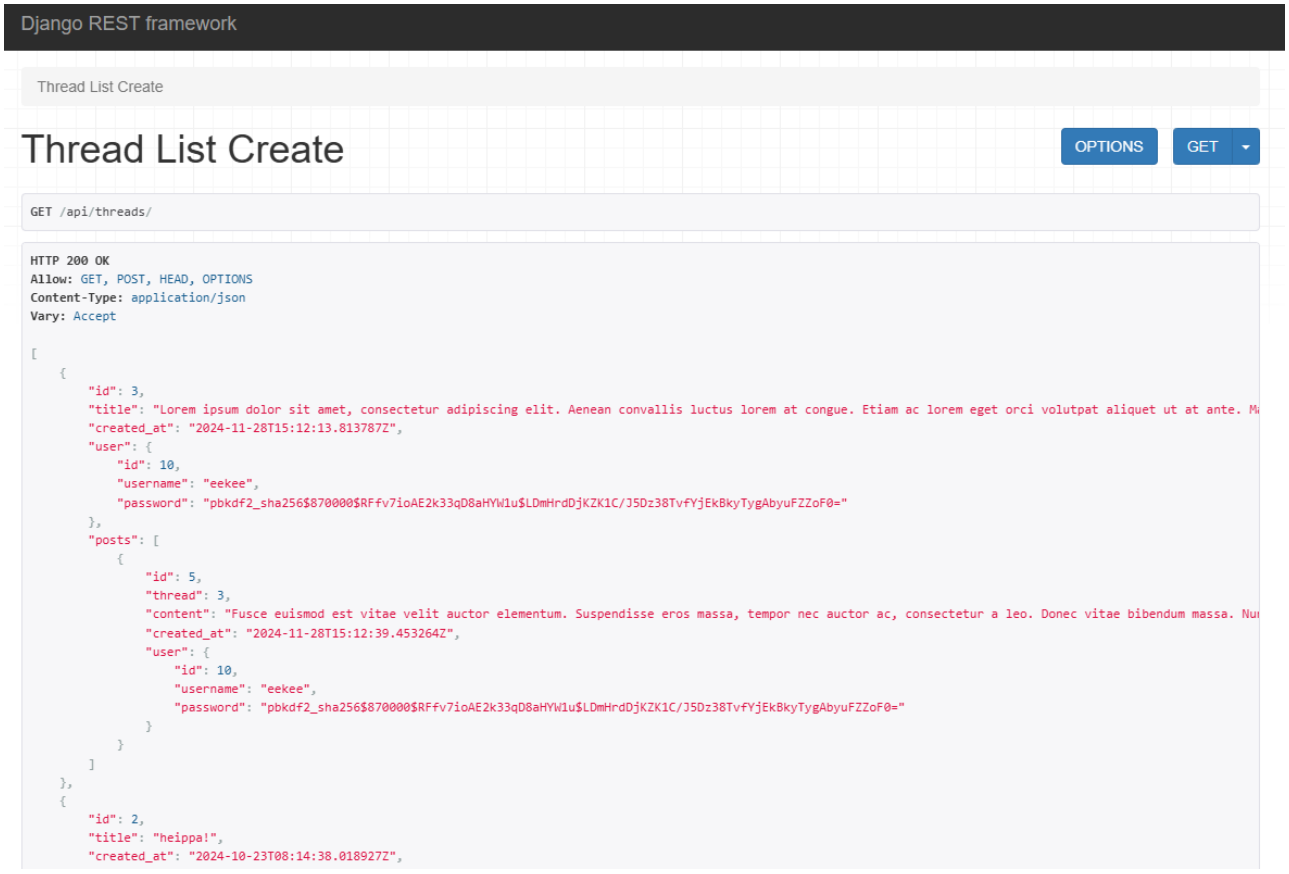
```
{(currentUser !== null && (currentUser.user_id === thread.id ||
currentUser.is_staff)) && (
          <button onClick={() => deleteThread(thread.id)}>Delete
thread</button>
)}
```

Foorumialustan ulkonäön kehittämisessä käytettiin CSS-kirjasto Bulmaa, joka otettiin käyttöön lataamalla kirjasto CSS-tiedostona ja lisäämällä tyylit className-attribuutteina JSX-elementteihin. Toiminta ei oleellisesti eroa tavallisen CSS:n käytöstä. Käytettyjä tyylejä olivat muun muassa valmiit container-, section-, ja box-attribuutit, joilla sovellukseen sai sujuvasti kehitettyä yksinkertaisen ja selkeän perusnäkymän. Käyttöliittymän pääväri on valkoinen ja viestien taustaväri on vaaleanharmaa. Ulkonäkö on hyvin pelkistetty.

Koska kehitystyön pääpainona olivat keskustelupalstoille olennaisten toiminnallisuuksien kehittäminen, käyttäjätilit ja viestit tärkeimpinä, Foorumialustaa ei valmisteltu julkaisua ja käyttöönottoa varten esimerkiksi Docker-teknologiaa käyttämällä, vaan sitä käytettiin vain paikallisessa kehitysympäristössä. Foorumialustaa ei siis siirretty palvelimelle tai pilvipalveluun, eikä siihen ollut pääsyä julkisesta verkosta.

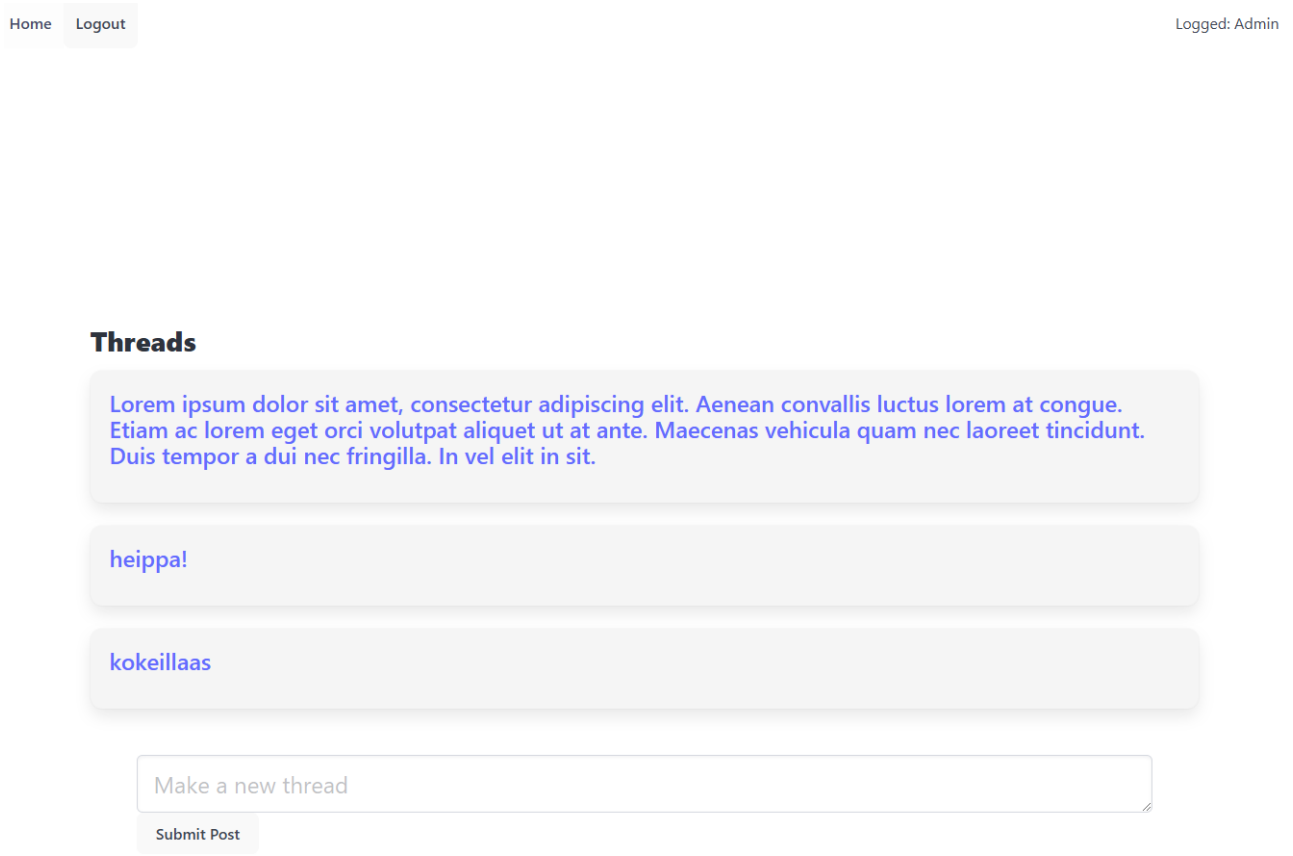
5 Tulokset

Foorumialustan kehittämisen tuloksena syntyi yksinkertainen, valittuja toimintoja sisältävä, keskustelusovellus. Toimintoja olivat käyttäjän rekisteröityminen, sisään- ja uloskirjautuminen, viestien ja viestiketjujen lähettäminen ja poistamisen mahdollistaminen niille, joilla on oikeus niin tehdä. Foorumialustan backendin osalta tuloksena oli REST-rajapinta, jonka kautta foorumin tiedot kulkivat, ja johon foorumin käyttöliittymä ottaa yhteyttä. Kuviossa 5 esitetään foorumin JSON-dataa, jollaista rajapinta tarjoilee haettaessa viestiketjujen tietoja. Tiedoista löytyvät viestiketjun tunnistenumero, teksti, luontiaika, käyttäjän tiedot, sekä ketjun sisältämät viestit.



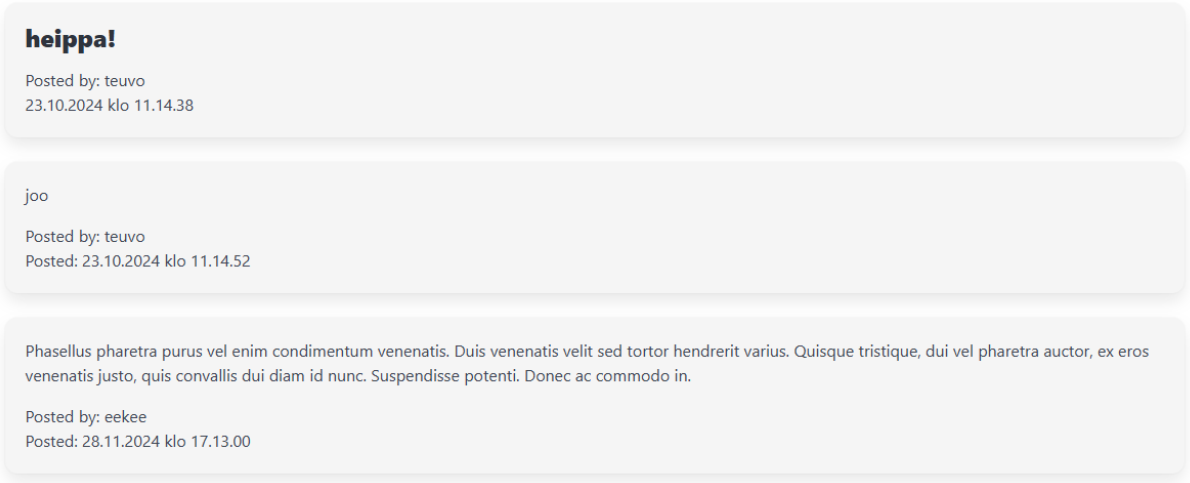
Kuvio 5. Viestiketjun tiedot Django REST Framework -näkymässä.

Reactilla kehitetyn frontendin kautta käyttäjän on mahdollista käyttää Foorumialustan toiminnallisuuksia. Käyttäjän tullessa sivustolle, ensimmäisenä hän saapuu etusivulle. Etusivu sisältää viestiketjut, sekä viestiketjun luontilomakkeen, joka on kuviossa 6 esillä, kun käyttäjä on kirjautunut. Muuten lomakkeen sijaan esitetään teksti, joka ohjeistaa kirjautumaan tai rekisteröitymään viestien lähettämiseksi. Kaikilla sivuilla on yläpalkki, joka kirjautuneena sisältää koti- ja uloskirjautumispainikkeet sekä käyttäjänimen. Ilman kirjautumista palkki sisältää kotipainikkeen lisäksi kirjautumis- ja rekisteröitymispainikkeet.



Kuvio 6. Foorumialustan etusivu.

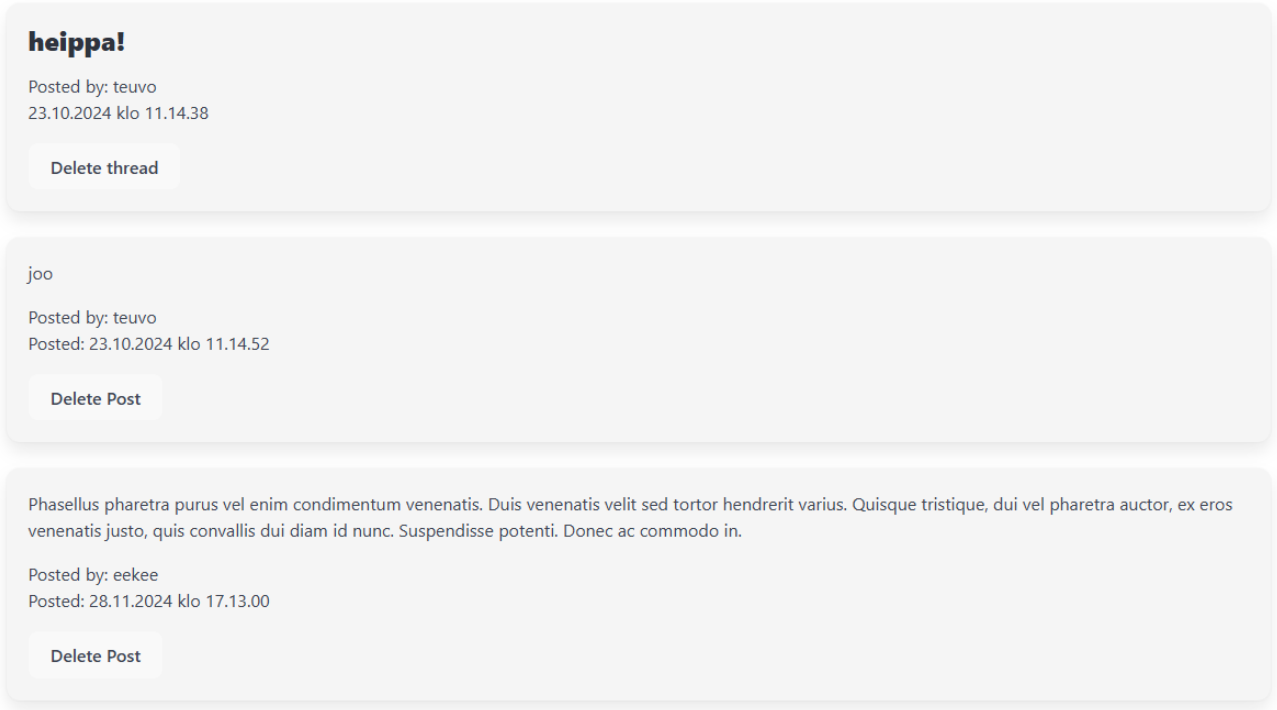
Viestiketjusivulle, joka sisältää ketjun aloituksen sekä vastausviestit, pääsee etusivulla olevien viestiketjujen linkkien kautta. Viestit sisältävät tekstin, lähettäjän nimen ja lähetysajan, jotka näkyvät kuviossa 7. Aloitusviestin teksti on suurempi, kuin vastausviestien.



Kuvio 7. Viestiketju.

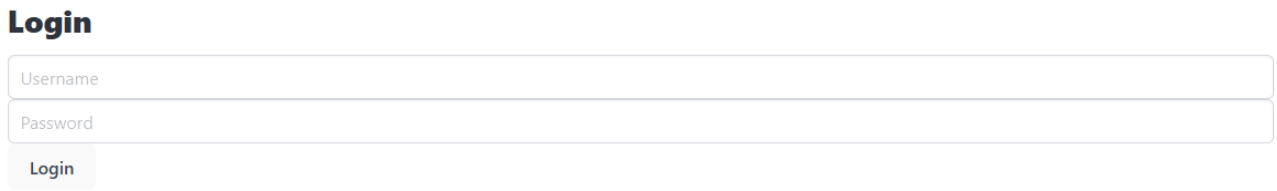
Viestien poistopainikkeet näkyvät vain, jos käyttäjä on ylläpitäjä tai viestin lähettäjä. Kuviossa 8 esiintyvässä tilanteessa eekee-niminen tavallinen käyttäjä voi poistaa oman viestinsä, mutta ei teuvo-käyttäjän viestiä. Kuviossa 9 taas ylläpitäjällä kykenee poistamaan molempien viestit, sekä viestiketjun, jossa teksti on suurempi. Viestiketjun poistaminen poistaa myös ketjun sisältämät viestit.

Kuvio 8. Käyttäjä voi poistaa oman viestinsä.



Kuvio 9. Ylläpitäjä voi poistaa viestejä.

Kirjautumis- ja rekisteröitymislomakkeet sisältävät käyttäjänimi- ja salasanakentät, sekä painikkeen kirjautumiseen tai rekisteröitymiseen. Kuvio 10 esittää kirjautumislomaketta. Rekisteröintilomake on samanlainen, joskin otsikon ja painikkeen tekstit ohjeistavat rekisteröitymään kirjautumisen sijaan.



Kuvio 10. Kirjautumislomake.

Kehitetty Foorumialusta kykenee toimimaan alustana tekstimuotoiselle keskustelulle. Alustalla voi olla käynnissä useita samanaikaisia keskusteluja. Käyttäjät voivat osallistua keskusteluihin, sekä aloittaa uusia keskusteluaiheita. Käyttäjien seassa moderaattorit voivat osallistua myös keskusteluihin samalla niitä valvoen. Tarpeen tullen moderaattorit voivat poistaa asiattomat viestit. Ylläpitäjällä on mahdollisuus Djangon hallintapaneelin kautta poistaa käyttäjiä.

6 Pohdinta

Tavoitteena työssä oli arvioida Djangon Reactin soveltuvuutta sovelluskehityksessä, erityisesti opinnäytetyön ohessa toteutetun esimerkkisovelluksen kaltaisen projektin kehitystyössä. Valittujen teknologioiden ominaisuudet ja tavat, joita toimintojen kehityksessä kyseisissä teknologioissa käytetään, esimerkiksi Djangon projekti-sovellus-jako ja Reactin Hook-tekniikat, olivat olennaisia tekijöitä kehitystyön sujuvuutta tarkastellessa; auttoivatko nämä tekijät tavoitteen saavuttamisessa? Aidosti toimivan sovelluksen kehittäminen oli sekin olennainen tavoite.

Yleisesti Djangon ja Reactin käyttäminen Foorumialustan kehittämisessä toimi hyvin ja niiden ominaisuuksilla oli myönteinen vaikutus kehitystyön kulkuun. Djangon tapa jäsennellä sovellus oli selkeä. Rajapinnan luominen, tietokantayhteys ja datan käsittely kaikkineen saatiin kehitettyä kohtuullisella työmäärä, sillä työssä ei tullut vastaan merkittäviä sudenkuoppia, joihin työskentely olisi jumittunut pitkäksi aikaa. Esimerkiksi Djangon Model-ominaisuuden käyttäminen tietokannan luomisessa osoittautui eduksi työn sujuvuuden kannalta, koska näin projektissa ei tarvinnut kirjoittaa lainkaan SQL-komentoja, vaan kaikki toiminta PostgreSQL-tietokannan kanssa hoitui Django-sovelluksessa Python-luokilla. Projektin aloittaessa automaattisesti luotavat aloitustiedostot auttoivat alkuun pääsemisessä, ja ohjelmistokehyksen dokumentaatiosta oli paljon hyötyä. Myös React-kehityksen voisi sanoa sujuneen kohtalaisesti. UseState- ja useEffect-hookit olivat Foorumialustan kehityksessä runsaassa käytössä ja ne osoittautuivat tehokkaiksi työkaluiksi käyttöliittymän luonnissa. Toiminnot, kuten viestien hakeminen rajapinnasta, toteutettiin useEffect-hookeilla, viestien listaamisessa sivulle sekä lomakkeiden toiminnassa käytettiin useState-hookkeja. JSX:n käyttäminen HTML-elementtien generoimisessa ei lopulta eroa merkittävästi tavallisten HTML-dokumenttien kirjoittamisesta. Pelkällä JavaScriptillä ilman Reactia, toimintojen kehittäminen olisi ollut työläämpää. Voisi todeta, ettei ole ihme, että React on saavuttanut suuren suosion verkkokehityksessä.

Foorumialustassa on vähän toimintoja, mutta ne kehitettiin toimiviksi. Vaikkakin yksinkertainen ja ominaisuuksiltaan vajava, sovellus on osoitus valittujen teknologioiden toimivuudesta. Yksinkertaisuus on seurausta pyrkimyksestä pitää ajankäyttö kohtuullisuuden rajoissa. Paljon toiminnallisuuksia jätettiin pois. Esimerkiksi viestien ja käyttäjäprofiilien muokkaaminen olisivat valmiille sovellukselle tärkeitä toiminallisuuksia. Ehkä syvällisemmällä alkuvaiheen suunnittelulla nämäkin

toiminnot olisi saatu sisällytettyä projektiin. Kuitenkin sovellus sisältää keskustelupalstalle ominaisia toiminnallisuuksia: käyttäjätilit, viestit ja viestiketjut. Työssä onnistuttiin kehittämään toimiva sovellus ja kehitystyössä onnistuttiin selvittämään teknologioiden vaikutuksia, jotka olivatkin työn tärkeimmät tavoitteet. Näiltä osin työssä onnistuttiin.

Työhön, prosessiin, ja tulosten arviointiin liittyi joitakin rajoitteita. Yksi työtä ja menetelmiä rajoittava tekijä oli ennestään rajallinen osaaminen erityisesti Djangon osalta, joka hidasti Foorumialustan kehitystä ja vaikeutti toiminnallisuuksien määrittämistä työn alkuvaiheessa. Esimerkiksi alussa ei ollut tiedossa, kuinka Djangossa käytetään tietokantoja. Rajoitteeksi voidaan myös laskea työn luonteen; Foorumialusta on täysin yksittäisen henkilön sovellusprojekti ilman selkeitä rajoituksia. Projekti olisi helposti voinut alkaa paisumaan täyteen epäolennaisuuksia tai kehitystyö olisi voitu toteuttaa huonoin käytäntein, ellei sovelluksen muodon hahmottuessa olisi osattu linjata toteutettavia ominaisuuksia. Tulosten luotettavuus sisältää sekin rajoitteita. Valittujen teknologioiden toimintaa ja ominaisuuksia olisi voitu enemmän verrata muihin samaan käyttöön soveltuvien teknologioiden, kuten toisten ohjelmistokehysten, kanssa esimerkiksi suorituskykymittauksin, jolloin saatavilla olisi myös numeraalista dataa. Myös Djangon ja Reactin ominaisuuksien vaikutusta sovelluksen kehitykseen arvioitiin lähinnä kehittäjän omasta näkökulmasta. Tulokset ovat kuitenkin siltä osin hyödyllisiä, että ne puoltavat teknologioiden soveltuvuutta sovelluskehitykseen. Tuloksista voi olla hyötyä esimerkiksi uuden sovelluksen suunnittelussa, kun valitaan mitä ohjelmistoja ja teknologioita aiotaan projektissa käyttää, ja kuinka tietyt toiminnallisuudet voitaisiin toteuttaa.

Koska Foorumialusta on melko pelkistetty ja toiminnallisuuksiltaan vähäinen, jatkokehitysmahdollisuuksia on runsaasti. Kehitystyössä jatkokehitettävyys otettiin huomioon muun muassa toiminnallisuuksien jakamisessa eri tiedostoihin, jatkokehitettävyyttä voi pitää ei-toiminnallisena ominaisuutena. Kehitysmahdollisuuksia löytyy. Käyttäjäprofiilien muokkaaminen, kuten nimen ja salasanan vaihtaminen tai tilin poistaminen profiilisivulla olisi hyvin tärkeä ominaisuus, samoin viestien muokkaaminen. Sähköpostiosoitteen lisääminen rekisteröintiin ja unohtuneen salasanan vaihtaminen sähköpostitse olisi myös käyttäjäkokemuksen kannalta ehdottoman tärkeää. Harmaavalkoisen minimalistisen ulkoasun voisi uudelleen kehittää näyttävämmäksi. Kuvien lisääminen viestiketjuihin toisi sovellukseen visuaalisuutta ja olisi todennäköisesti haluttu ominaisuus käyttäjien keskuudessa. Kattavat ohjelmiston testaukset sekä sovelluksen valmistelu verkkoon julkaisemista varten täytyisi suorittaa, mikäli Foorumialusta todella aiottaisiin avata käyttäjille.

Lähteet

About. n.d. PostgresSQL-tietokantajärjestelmän esittelysivu. Viitattu 13.11.2024. https://www.postgresql.org/about/

Abramov, D. 2019. React v16.8: The One With Hooks. Blogikirjoitus legacy.reactjs -sivustolla. Viitattu 5.5.2024. https://legacy.reactjs.org/blog/2019/02/06/react-v16.8.0.html

Aichner, T. & Jacob, F. 2015. Measuring the Degree of Corporate Social Media Use. researchgate.net. Viitattu 19.3.2024. https://www.researchgate.net/publication/283073224_Measuring_the_Degree_of_Corporate_Social_Media_Use

An overview of HTTP. 2024. MDN Web Docs. Viitattu 17.11.2024. https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

Backend. n.d. Bautomo-yrityksen verkkosivu. Viitattu 27.3.2024. https://bautomo.com/sanastoa/backend/

Built-in React Hooks. n.d. React-dokumentaatio. Viitattu 5.5.2024. https://react.dev/reference/react/hooks

Component. n.d. React-dokumentaatio. Viitattu 5.5.2024. https://react.dev/reference/react/Component

Componentizing our React app. 2023. MDN Web Docs. Viitattu 5.5.2024. https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_components

Damewood, C. 2024. What is a Forum Moderator? EasyTechJunkie-verkkosivusto. Viitattu 1.4.2024. https://www.easytechjunkie.com/what-is-a-forum-moderator.htm

DB-Engines Ranking. 2024. db-engines.com. Viitattu 1.4.2024. https://db-engines.com/en/ranking

Django vs Fast API: Detailed Comparison. 2022. Medium.com. Viitattu 5.5.2024. https://medium.com/@ShortHills_Tech/django-vs-fast-api-a-detailed-comparison-df8d00f3c3b2

Felding, R. 2008. REST APIs must be hypertext-driven. Roy T. Fieldingin Untangled-blogi. Viiattu 15.4.2024. https://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven

Hooks at a Glance. n.d. Legacy.reactjs-dokumentaatio. Viitattu 5.5.2024. https://legacy.reactjs.org/docs/hooks-overview.html

HTTP. 2024. MDN Web Docs. Viitattu 17.11.2024. https://developer.mozilla.org/en-US/docs/Web/HTTP

HTTP request methods. 2024. MDN Web Docs. Viitattu 17.11.2024. https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

Introduction. n.d. Next.js-ohjelmistokehyksen dokumentaatio. Viitattu. 18.11.2024. https://nextjs.org/docs

JSX. 2022. Facebookin JSX-dokumentaatio. Viitattu 4.5.2024. https://facebook.github.io/jsx/

MERN Stack Explained. n.d. MongoDB-tietokannan verkkosivut. Viitattu 18.4.2024. https://www.mongodb.com/mern-stack

Permissions. n.d. Django REST Frameworkin dokumentaatio. Viitattu 17.11.2024. https://www.django-rest-framework.org/api-guide/permissions/

Render Functions & JSX. n.d. Vue.js-ohjelmistokehyksen verkkosivut. Viitattu 4.5.2024. https://vuejs.org/guide/extras/render-function

React Class Components. n.d. W3Schools-verkkosivu. Viitattu 5.5.2024. https://www.w3schools.com/react/react_class.asp

React Components. n.d. W3Schools-verkkosivu. Viitattu 5.5.2024. https://www.w3schools.com/react/react_components.asp

React Resources. 2023. MDN Web Docs. Viitattu 5.5.2024. https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_resources

REST. 2023. MDN Web Docs. Viitattu 17.4.2024. https://developer.mozilla.org/en-US/docs/Glossary/REST

Serialization. 2023. MDN Web Docs. Viitattu 16.11.2024. https://developer.mozilla.org/en-US/docs/Glossary/Serialization

Server-side web frameworks. 2024. MDN Web Docs. Viitattu 27.3.2024. https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Web_frameworks

Templates. n.d. Django-ohjelmistokehyksen dokumentaatio. Viitattu 18.11.2024. https://docs.djangoproject.com/en/5.1/topics/templates/

Toikko, T & Rantanen, T. 2009. Tutkimuksellinen kehittämistoiminta. Tampere: Tampereen yliopistopaino.

Virtual DOM and Internals. n.d. legacy.reactjs.org -verkkosivu. Viitattu 29.3.2024. https://legacy.reactjs.org/docs/faq-internals.html

What is a Front-End Developer? n.d. w3schools-verkkosivu. Viitattu 25.3.2024. https://www.w3schools.com/whatis/whatis_frontenddev.asp

What is a REST API? n.d. IBM-teknologiayrityksen verkkosivu. Viitattu 31.3.2024. https://www.ibm.com/topics/rest-apis

Writing Markup with JSX. n.d. Reactin ohjesivut. Viitattu 4.5.2024. https://react.dev/learn/writing-markup-with-jsx

Writing your first Django app, part 1. n.d. Django-ohjelmistokehyksen dokumentaatio. Viitattu 12.11.2024. https://docs.djangoproject.com/en/5.1/intro/tutorial01/

Writing your first Django app, part 2. n.d. Django-ohjelmistokehyksen dokumentaatio. Viitattu 16.11.2024. https://docs.djangoproject.com/en/5.1/intro/tutorial02/