Pelien arvostelusovelluksenkehitys: Full Stack - Ratkaisu

Karri Helokumpu

Haaga-Helia ammattikorkeakoulu

Tradenomin tutkinto

Amk-opinnäytetyö

2025

Tiivistelmä

Tekijä(t)
Karri Helokumpu

Tutkinto
Tradenomi

Raportin/Opinnäytetyön nimi
Pelien arvostelusovelluksenkehitys: Full Stack - Ratkaisu

Sivu- ja liitesivumäärä
19

Opinnäytetyöni perustana on Full-Stack-sovellus, jonka suunnittelu, kehittäminen ja ylläpito muodostavat työn keskeisen sisällön. Sovelluksen tarkoituksena on tarjota käyttäjille alusta, jonka avulla he voivat arvioida erilaisia videopelejä ja muodostaa mielipiteitä pelien laadusta muiden käyttäjien arvostelujen perusteella. Sovellus mahdollistaa myös vuorovaikutuksen käyttäjien välillä, mikä lisää yhteisöllisyyttä ja antaa arvokasta tietoa pelien suosiosta ja vastaanotosta.

Kvalitatiivinen tutkimukseni perustuu omiin empiirisiin havaintoihini sovelluksen kehittämisprosessin aikana. Dokumentoin koko kehitysprosessin ideoinnista toteutukseen sekä ylläpitovaiheeseen asti. Erityistä huomiota kiinnitin omiin havaintoihini, käyttöliittymän selkeyteen sekä teknisten ratkaisujen toimivuuteen.

Tutkimusmenetelmäni oli käytännönläheinen ja iteratiivinen. Kehitin sovellusta jatkuvasti omien havaintojeni perusteella. Jokainen kehitysvaihe tuotti uutta tietoa sovelluksen toiminnallisuuksista, käytettävyydestä ja teknisestä vakaudesta, jotka kaikki vaikuttivat jatkokehityksen suuntaan.

Sovelluksen kehittäminen tarjosi samalla mahdollisuuden syventää omaa osaamistani ohjelmistokehityksen eri osa-alueilla sekä havainnoida, miten eri komponentit ja teknologiat yhdistyvät toimivaksi kokonaisuudeksi. Näiden kokemusten pohjalta muodostin kokonaiskuvan siitä, mitä Full-Stack-kehittäminen käytännössä vaatii ja millaisia haasteita ja mahdollisuuksia siihen liittyy.

Asiasanat
full-stack-kehitys, ohjelmistokehitys, web-kehitys, frontend, backend, tietokannat.

Sisällys

1 Johdanto

Aiheenani on tehdä videopeliarvostelusivusto, ja olen kiinnostunut laajentamaan osaamistani niillä taidoilla, jotka jo hallitsen. Haluan toteuttaa full stack -projektin, jonka avulla voin syventää osaamistani paitsi ohjelmoinnissa myös siinä, millä periaatteilla sovellus toteutetaan. Tavoitteenani on kerätä tietoa full stack -projektin luomisesta yleisesti.

Oman osaamisen kehittämiseksi hyödynnän seuraavia konkreettisia keinoja:

1. Ongelmanratkaisu: Pyrin ensin ratkaisemaan haasteet itsenäisesti ja vasta sen jälkeen kysyn apua. Analysoin myös vaihtoehtoisia ratkaisuja ja sovellan erilaisia työskentelymenetelmiä.
2. Kirjallinen ilmaisu: Harjoittelen akateemista kirjoittamista sekä pyydän tarvittaessa palautetta ohjaajalta.
3. Tiedonhaku: Etsin aktiivisesti ajankohtaista tietoa ja tutkimuksia, jotka tukevat projektin toteuttamista ja oman ymmärryksen syventämistä.

Sovellus julkaistaan tuotantoympäristöön, ja se on käytettävissä loppukäyttäjille. Käytän Git-versionhallintaa systemaattisesti, mukaan lukien haarat (branches), commit-viestit ja pull request -käytännöt. Kehitän sovelluksen niin, että se täyttää määritetyt toiminnallisuudet ja toimii odotusten mukaisesti. Sovellukseni avulla käyttäjät voivat arvioida pelejä ja muodostaa mielipiteen niiden laadusta muiden arvostelujen perusteella.

- ✓ **Uusimmat pelit tulevat heti näkyviin** – Pelit ilmestyvät sovellukseen reaaliaikaisesti IGDB:n päivitysten mukaan.
- ✓ **Kaikki Peli Arvostelut Näkyvät Helposti** - Kotisivulla näkyvät kaikki pelit, joille on jätetty arvioita, ja jokaisen pelin kohdalta pääset tarkastelemaan kaikki kommentit.
- ✓ **Yksinkertainen Arvostelujen Tekeminen –** Etsi ja valitse peli listalta, jonka jälkeen voit lisätä oman arviosi nimellä, kommentilla ja arvosanalla.

Vastuullisuuden osalta varmistan koodin selkeyden, testattavuuden ja tietoturvan (esim. käyttäjätietojen suojaaminen). Dokumentoin sovelluksen ja sen kehitysprosessin täsmällisesti, jotta muut saavat hyödyntää sitä tulevaisuudessa. Pyrin parantamaan sovelluksen kaikkia osa alueita.

1.1 Käytetyt ohjelmistot ja sovellukset

React on JavaScript-kirjasto käyttöliittymien suunnittelemiseen. Spring Boot on Java-pohjainen backend-kehys, joka auttaa palvelinsovelluksien kehittämisessä. REST API (Representational

State Transfer) on Arkkitehtuurimalli, jossa frontend ja backend keskustelevat HTTP-pyyntöjen kautta.

Projektini front-end käyttää Reactia, backend taas käyttää Spring Boottia ja Javaa, Tietokanta käyttää MySQL. Rajaan lopputyön ulkopuolelle React Native ja Flutter koska projektini keskittyy ainoastaan web-sovelluksen kehittämiseen. Tarvittava osaaminen tälle työlle on Java ja JavaScript-koodikielten osaaminen. Tällä hetkellä osaamiseni on se, että pystyn tekemään toimivan full-stack-projektin.

2 Tietoperusta

2.1 Backend teknologiat

Spring Framework on sovelluskehys Java-kielelle, joka tarjoaa kattavan ohjelmointi- ja konfigurointimallin Java-pohjaisten sovellusten kehittämiseen. Se yksinkertaistaa Java-sovellusten kehittämistä käsittelemällä monimutkaisia kokoonpanoja ja tehtäviä, kuten kohteiden elinkaaren hallintaa, riippuvuuksien injektointia (dependency injection) ja transaktioiden hallintaa (docs.spring.io). Spring Framework on myös modulaarinen, mikä mahdollistaa vain tarvittavien ominaisuuksien käytön ilman, että koko kehystä tarvitsee ottaa käyttöön (docs.spring.io).

Spring Boot on Spring Frameworkin päälle rakennettu laajennus, joka helpottaa itsenäisten, tuotantovalmiiden Spring-pohjaisten sovellusten luomista. Se yksinkertaistaa Spring-sovellusten kehittämistä poistamalla tarpeen laajoille kokoonpanoille ja tarjoamalla valmiita malleja sekä käytäntöjä (spring.io). Tämän ansiosta kehittäjät voivat keskittyä sovelluksen toiminnallisuuksien rakentamiseen ilman huolta vakiokoodista tai monimutkaisista asetuksista (spring.io).

2.2 Frontend teknologiat

React on JavaScript-kirjasto, jolla luodaan käyttöliittymiä. Sen on luonut ja sitä ylläpitää Meta (Entinen Facebook) (Wikipedia React). Reactia käytetään ensisijaisesti yksisivuisten sovellusten (Single Page Application) ja dynaamisten käyttöliittymien rakentamiseen verkkosivustoille ja mobiilisovelluksille (reactjs.org).

Reactilla on helppo luoda dynaamisia sovelluksia, koska se vaatii vähemmän koodia ja tarjoaa enemmän toiminnallisuutta verrattuna pelkkään JavaScriptiin, jossa koodaaminen voi helposti muuttua monimutkaiseksi (reactjs.org).

2.3 Menetelmät

SCRUM on johtamiskehys, jota tiimit käyttävät itseorganisoitumiseen ja yhteisen tavoitteen saavuttamiseen. Se määrittelee joukon kokouksia, työkaluja ja rooleja tehokkaan projektin toteuttamiseksi.

Ohjelmoijatiimit käyttävät Scrumia monimutkaisten ongelmien ratkaisemiseen kustannustehokkaasti ja kestävästi (aws.amazon). Scrum-tiimit hyödyntävät niin kutsuttuja Scrum-artefakteja ongelmien ratkaisemiseen ja projektinhallintaan. Scrum-artefaktit tarjoavat kriittistä suunnittelu- ja tehtävätietoa tiimin jäsenille ja sidosryhmille.

Ensisijaisia artefakteja on kolme: Tuotteen kehitysjono, Sprint-kehitysjono ja Kasvu. Tuotteen kehitysjono on dynaaminen luettelo ominaisuuksista, vaatimuksista, parannuksista ja korjauksista, jotka on suoritettava loppuun projektin onnistumiseksi. Se on pohjimmiltaan tiimin tehtävälista, jota tarkastellaan jatkuvasti uudelleen ja priorisoidaan uudelleen, jotta se voi mukautua markkinoiden muutoksiin.

Tuotteen omistaja ylläpitää ja päivittää luetteloa poistamalla tarpeettomia kohteita tai lisäämällä uusia pyyntöjä asiakkailta (aws.amazon). Scrum-tiimi koostuu kolmesta erityisestä roolista: tuotteen omistajasta, Scrum-johtajasta ja kehitystiimistä.

Tuotteen omistaja keskittyy varmistamaan, että kehitystiimi tuottaa yritykselle eniten lisäarvoa. Hän ymmärtää ja priorisoi loppukäyttäjien ja asiakkaiden muuttuvat tarpeet. Tehokkaat tuotteen omistajat antavat tiimille selkeät ohjeet siitä, mitkä ominaisuudet toimitetaan seuraavaksi. He yhdistävät sen, mitä yritys haluaa, ja sen, mitä tiimi ymmärtää, sekä päättävät, kuinka usein julkaisuja tulisi tapahtua. Scrum-johtaja toimii tiimin valmentajana ja on vastuussa Scrum-tiimin tehokkuudesta. Hän tukee tiimejä, tuotteen omistajia ja yritystä parantamaan Scrum-prosesseja ja optimoimaan toimituksia. Scrum-johtaja huolehtii myös siitä, että jokaiselle sprintille varataan tarvittavat resurssit, sekä helpottaa Sprint-tapahtumia ja tiimikokouksia. Lisäksi hän johtaa digitaalista muutosta tiimissä, tukee uusien teknologioiden käyttöönottoa, järjestää koulutuksia sekä kommunikoi ulkopuolisten ryhmien kanssa ratkaistakseen tiimin kohtaamat haasteet.

2.4 Sovelluskehityksen työkaluja

Visual Studio Code (VS Code) on sovellus, jolla voi editoida kaikenlaisia eri tiedostomuotoja, ja se tunnistaa automaattisesti eri ohjelmointikielet, kun sille annetaan koodia (Microsoft, Visual Studio Code).

GitHubin saa myös integroitua Eclipseen lataamalla Eclipsen sovelluskaupasta EGit-nimisen lisäosan. Kun lisäosa on asennettu, voi vain klikata hiiren oikealla nykyistä projektia, mennä "Team"-nimiseen välilehteen ja sieltä löytyvät kaikki GitHub-ominaisuudet, kuten push ja commit (Eclipse Foundation, EGit).

2.5 Sovelluksen julkaisuprosessi

Yleensä kun sovellusta luodaan ja julkaistaan siitä, on useita eri versioita, on Alfa (Alpha) joka on kehittäjien sisäinen testaus, joka yleensä sisältää bugeja, Beeta (Beta) joka julkaistaan yleensä pienelle määrälle käyttäjiä (beta testaajat) jotta ne voivat tunnistaa bugit ja kerätä palautetta, tämä versio on stabiilimpi kuin alpha versio mutta ei ole valmis tuotantokäyttöön ja julkaisuun. Julkaisuehdokas (RC eli Release Candidate) joka on versio mistä voi tulla viimeinen versio, jos vakavia

bugeja ei pahemmin löydy, viimeinen vaihe ennen tuotantokäyttöä. Yleinen Saatavuus (GA eli General Availability) joka on viimeinen stabiili versio mikä julkaistaan julkisuuteen. Se on valmis tuotantokäyttöön (Darrel Ince: Software Engineering, A Practitioners Approach).

3 Seurantajakson raportointi viikkoanalyyseineen

Backendissä olen käyttänyt tälläisiä riippuvuuksia: Spring Web, Spring Data JPA, MYSQL Driver, Spring Security, Spring Boot DevTools, Validation, Lombok. Backend tietokantana käytän MariaDB. Frontendin vaatimukset: frontendissä näkee kaikki pelit ja näkee yksittäisen pelin tarkemmat tiedot.

3.1 Sprint 1 (5.11 – 19.11.2024)

Tavoitteena on saada toimiva backend missä tietoa voidaan lähettää Rest API:n avulla tietokantaan hyödyntäen CRUD operaatioita.

Tein sprint1 aikana uuden REST ohjelmointirajapinnan, joka sisältää Review nimisen luokan, jossa on kaikki muuttujat käyttäen Spring Boottia. integroin MariaDB tietokanta palvelimeni Review taulukolla, johon laitoin arvostelu tiedot.

Lisäsin CRUD operaatiot (Create Update Delete) ja eri endpointit:

GET /api/reviews saadakseni kaikki arvostelut

POST /api/reviews lisätäkseni uuden arvostelun.

PUT /api/reviews/id päivittääkseni arvostelua.

DELETE /api/reviews/id poistaakseni arvostelun.

GET /api/reviews/rating/rating suodattaakseni arvostelut arvosanan perusteella.

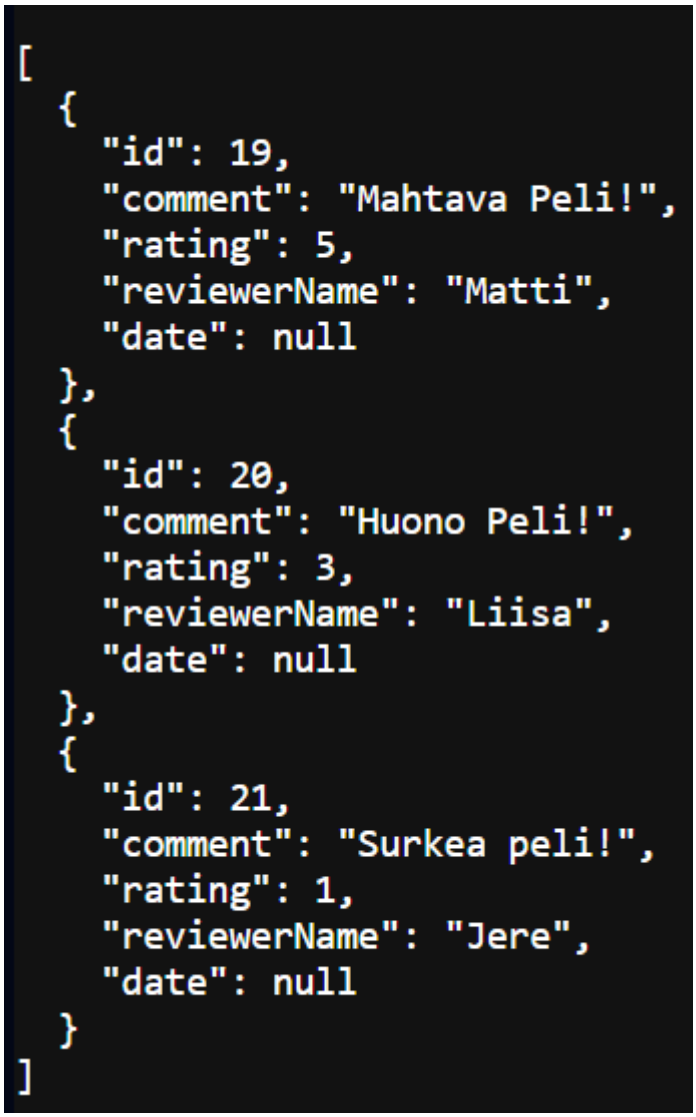


Figure 1 Näkymä /api/reviews endpointista

Käytin Spring Data JPA yhdistääkseni MariaDB tietokanta palvelimeeni ja testasin tietokanta yhteyksiä Postmanilla. Integroin MariaDB (MYSQL Workbenchillä) tietokannan palvelimeksi backendiini. Laitoin Kaikki nämä tuonne application.properties tiedostoon:

```
spring.application.name=PortfolioBackend
spring.datasource.url=jdbc:mariadb://localhost:3306/portfolio_db
spring.datasource.username=portfolio_user
spring.datasource.password=qwerty12345678
spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
spring.jpa.show-sql=true
spring.jpa.database-platform=org.hibernate.dialect.MariaDBDialect
spring.jpa.hibernate.ddl-auto=update
spring.security.user.name=admin
spring.security.user.password=admin
spring.security.user.roles=USER
logging.level.org.springframework.security=DEBUG
```

Määritin yhteyden MariaDB tietokanta palvelimeen.

Lisäsin projektin alussa projektiin mukaan Spring Security:n joka vaati tunnistautumisen ohjelmointirajapinnan käyttöä varten ja kohtasin haasteita sen kanssa juuri tämän takia. Lisäsin applications.properties tiedostoon `spring.security.user.name=admin spring.security.user.password=admin` jotta ei tarvitsisi joka kerta valita konsolista uusi generoitu salasana, joka täytyy laittaa Postman:iin. Tämän muutoksen jälkeen kaikki GET komennot toimivat Postmanilla tehdyillä testeillä mutta kaikki muut CRUD operaatiot aiheuttivat 401 unauthorized virheen.

Tietokannassa tuli ongelmia, kun yritin tyhjentää kaikki rivit taulukosta (MYSQL Error 1175) (Safe mode). Korjasin tämän ongelman antamalla tietokannalle komennon:

`SET SQL_SAFE_UPDATES = 0;` joka poistaa Safe Updates käytöstä mikä estää poistamasta taulukosta liikaa rivejä, jos niitä on paljon siellä. Tämän jälkeen on mahdollista poistaa kaikki rivit taulukosta mutta on hyvä laittaa se takaisin päälle `SET SQL_SAFE_UPDATES = 1;` komennolla jotta tulevaisuudessa ei vahingossa poista kaikkia rivejä taulukosta.

Missä projekti on nyt: Minun CRUD Rest ohjelmointirajapinta toimii ilman Spring Securityä, minä poistin sen pom.xml tiedostosta. Minun backend toimii MariaDB tietokannan kanssa onnistuneesti.

3.2 Sprint 2 (19.12 - 6.1.2024)

SUUNNITELMA: Sprintin 2 aikana luon frontendin, joka hakee pelejä Reactin kautta ohjelmointirajapinnasta. Teen perussivun, jossa voi lisätä pelejä, ja näytän lisätyt pelit. Tallennan pelien ID:t omaan tietokantaan. Käytän Reactia, MUI:ta ja Figmaa käyttöliittymän suunnitteluun.

Tein Sprint2 aikana React frontend nettisivun, sivulta pystyy näkemään pelejä ja pelien arvosteluja. Arvostelut tulevat omasta tietokannastani ja tallentuvat sinne mutta itse pelit ja niiden kansikuvat tulisivat ohjelmointirajapinnan kautta. Mietin käyttää Internet Games Database ohjelmointirajapintaa (IGDB) niillä on todella laaja valikoima eri videopelejä sekä kuvia ja videoita eri videopeleistä. Sen omistaa Amazon ja sitä pääsääntöisesti käytetään Twitch striimauspalvelussa, mutta sitä voi käyttää sen ulkopuolella myös. Käyttäjä tulisi itse laittaa kotisivulla oma ohjelmointirajapinta avain, jonka saa, kun rekisteröityy niiden palveluun ja se ei tallennu ollenkaan minun omaan tietokantaani

pitäen sen turvassa. Tietenkin jos tämä ei toimi ja menee liian monimutkaiseksi tämän osalta, niin laitan itse omia tietoja sinne sivulle. Kuitenkin olen saanut aikaiseksi perusyhteyden React Frontendin ja Spring Boot backendin kanssa.

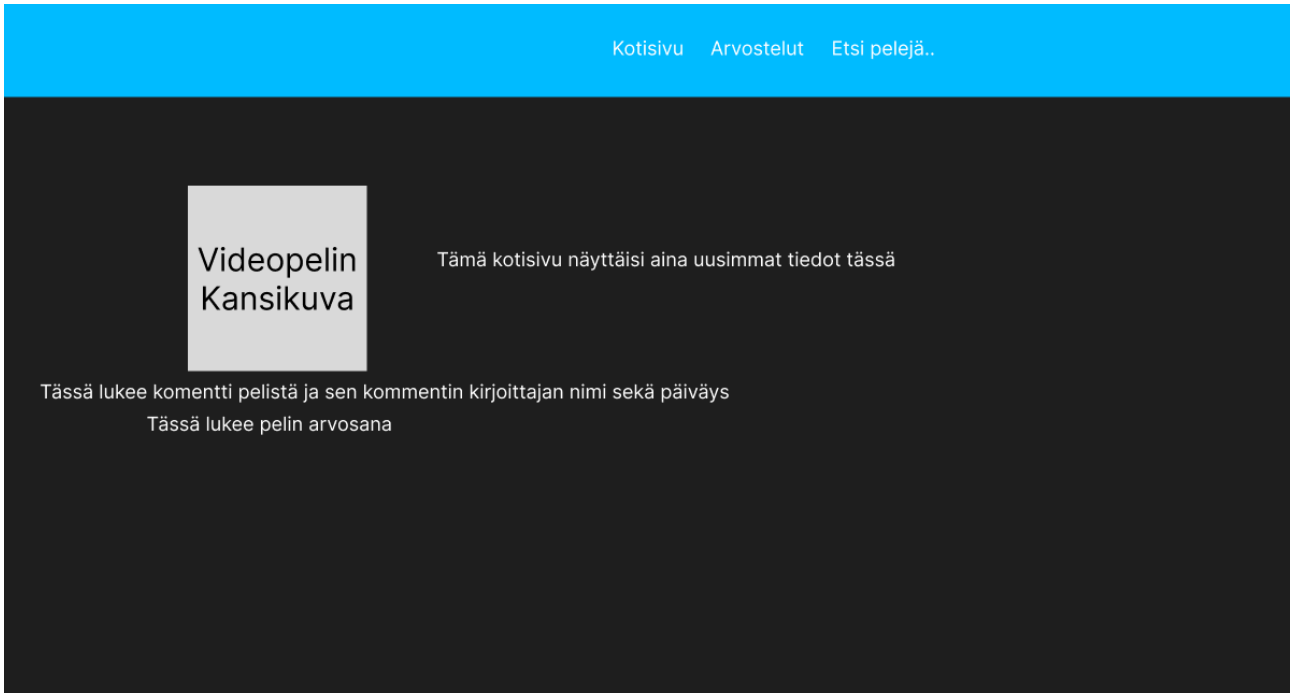


Figure 2 Figmalla tehty näkymä mahdollisesta Kotisivusta, jonka tulen viimeistelemään React Material UI:lla.

Tällä hetkellä kotisivu saa IGDB tietokannan tiedot ja ne näkyvät Material UI:n AG Gridissä, Omat backend tiedot näkyvät siinä gridissä.

3.3 Sprint 3 (kesti kalenteriajassa noin 6 viikkoa. 3 viikkoa ajasta meni kipeänä oltaessa, 1.2.2025 – 22.3.2025)

Tässä Sprintissä oli vain kaksi tavoitetta. Tämän sprintin ensimmäinen tavoite oli saada toimiva nettisivu, josta löytyy kaikki perusominaisuudet, kuten arvostelujen lisääminen, pelien etsiminen, arvosteltujen pelien näyttäminen kotisivulla, mahdollisuus lisätä useampi arvostelu yhdelle pelille sekä nähdä kotisivun kautta kaikki arvostelut, jotka on jätetty kyseiselle pelille.

Toinen tavoite oli tehdä muutoksia omaan MariaDB-tietokantaani ja lisätä puuttuvat ominaisuudet backend-koodiini. Joka löytyy backend koodini luokasta Review.java

private Long reviewid;

private Long igdbId;

private String comment;

private int rating;

private String reviewerName;

private LocalDateTime date;

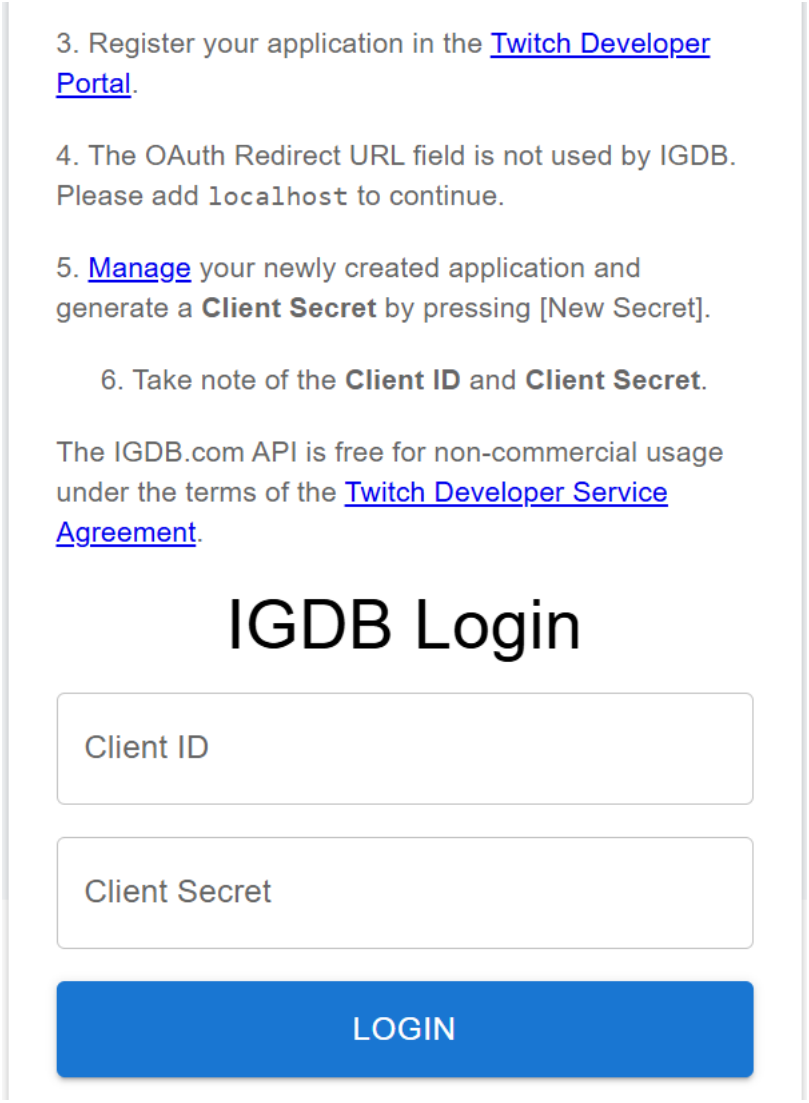
igdbId on kokonaislukuarvo, johon tallennetaan pelin tunniste (ID) IGDB-rajapinnasta. Kun tämä tunniste on haettu, voin yhdistää oman tietokantani arvostelu tiedot IGDB:stä haettuihin pelitietoihin tunnisteen avulla. Lisäsin tämän kentän myös MySQL-tietokantaani ja huomasin, että olin määrittänyt igdbId:lle uniikkiavaimen. Tämä tarkoitti sitä, etten voinut tallentaa useampaa kuin yhden arvostelun per peli. Käytin komentoa: ALTER TABLE reviews DROP INDEX igdb_id; Tämä poisti igdbId-kentältä uniikkiavaimen, minkä jälkeen pystyin tallentamaan useamman arvostelun samalle peli-ID:lle. Lisäsin myös kaksi uutta endpointtia backend-koodiini sekä tämän rivin ReviewRepository nimiseen luokkaan: List<Review> findByIgdbId(Long igdbId); Näiden avulla frontend osaa löytää oikeat sivut. Polun /add kautta on mahdollista lisätä arvostelu frontendin kautta. Toinen endpoint taas mahdollistaa sen, että kun avaan pelin kaikki arvostelut etusivulta, se osaa hakea oikean ID:n ja avata sivun, joka kuuluu kyseiselle pelille.

```
@PostMapping("/add")
        public ResponseEntity<Review> addReview(@RequestBody Review review) {
        Review savedreview = reviewRepository.save(review);
        return ResponseEntity.ok(savedreview);
        }

        @GetMapping("/game/{igdbId}")
        public List<Review> getReviewsByGameId(@PathVariable Long igdbId) {
        return reviewRepository.findByIgdbId(igdbId);
        }
```

3.4 Valmiin Sovelluksen Kuvaus

Jotta on mahdollista käyttää peliarvostelusivua ja sen palveluita, Internet Games Database (IGDB) ohjelmistorajapinta edellyttää sisäänkirjautumista Asiakastunnuksella (Client ID) ja Asiakassalaisuudella (Client Secret), jotka saa kun rekisteröityy Twitch-palveluun ja luo pääsytunnuksen (access token) Twitch Developers -sivulla. Kirjautumisen avulla saamme luvan hakea pelitietoja IGDB:stä – kuten pelin nimen ja pelin kuvauksen – ja yhdistää ne omaan tietokantaan.

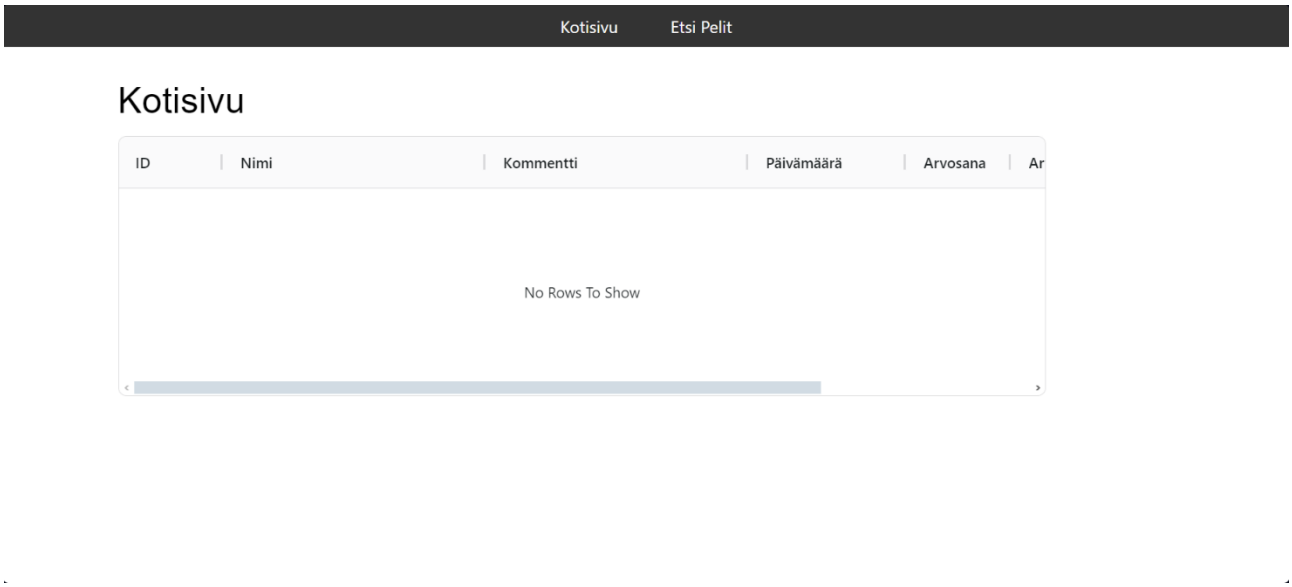


Kuva 1 *Tämä on ensimmäinen sivu joka tulee esiin kun käyttää peli arvostelu palvelua. Sivusto vaatii Asiakastunnuksen (Client ID) ja Asiakassalaisuuden (Client Secret) jotka saa vain kun rekisteröityy Twitch palveluun ja luo pääsytunnuksen (access token).*

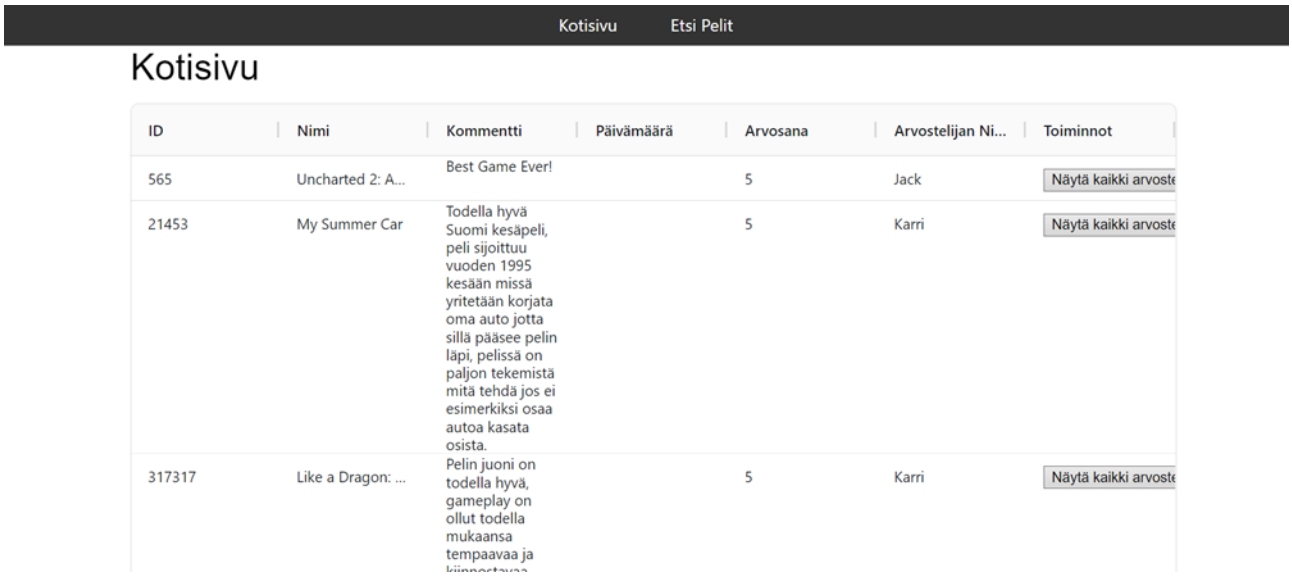
Kirjautumissivu tarkistaa automaattisesti, että syötetyt Client ID ja Client Secret ovat oikein ja toimivat. Jos tunnukset ovat virheelliset sivu ilmoittaa tästä virheilmoituksella. Tällöin käyttäjälle ei anneta pääsyä palvelun muihin osiin, eikä navigointipalkki ole käytettävissä ennen kuin oikeat tunnukset on syötetty.

Jos asiakastunnukset ovat oikein ja toimivat, pääset eteenpäin ja pääset kirjautumaan sisään. Oikeiden tunnusten syöttämisen jälkeen navigointipalkki aktivoituu ja saat pääsyn kaikkiin peliarvostelusivun toimintoihin. Kirjautuminen on voimassa ainoastaan niin kauan kuin välilehti on auki. Jos välilehti suljetaan ja sivu avataan uudelleen, käyttäjältä pyydetään kirjautumista uudestaan, koska uusi välilehti ei muista aiempaa sisäänkirjautumista.

Kun tunnukset on luotu ja kirjautuminen on onnistunut, pääset käyttämään kaikkia sivun ominaisuuksia, kuten pelitietojen hakemista, arvostelujen jättämistä ja muiden käyttäjien arvostelujen lukemista.



Kuva 2 *Tässä on React etusivu minun omasta peliarvostelu sivusta (peli lista on tyhjä kun ei ole arvosteluja lisätty ja tämä näyttää vain arvostellut pelit).*



Kuva 3 *Tässä etusivulla näkyy pelejä, joilta löytyy arvosteluja. Samalla näkyy nappi mistä painamalla pääsee näkemään kaikki arvostelut, jotka on jätetty kyseiselle pelille (jos sillä on useampia).*

Kotisivulla käyttäjät voivat tarkastella kaikkia videopelejä, joita on arvosteltu. Sivulla näkyy AG Grid -komponentti lista joka esittää listan peleistä, joille on jätetty arvosteluja. Jokaiselle pelille on oma rivi, joka sisältää pelin ID:n, nimen, kommentin, arvosanan ja arvostelijan nimen. Jos pelille on jätetty arvosteluja, ne tulevat näkyviin tässä listassa.

Jos arvosteluita ei ole jätetty, lista on tyhjä eikä pelistä ole näkyvillä lisätietoja. Tämä voi johtua siitä, että peliä ei ole vielä arvosteltu tai se ei ole saatavilla arvostelujen puuttumisen tai nettiongelmien takia.

Jokaiselle pelille näkyy myös toimintopainike ("Näytä kaikki arvostelut"), joka vie käyttäjän erilliselle sivulle, jossa voi tarkastella tarkemmin pelille jätettyjä arvosteluja. Tällä sivulla on listattu kaikki pelille annetut arvostelut, ja käyttäjä voi nähdä, kuinka monta arvostelua peli on saanut. Tämä sivu tarjoaa kattavamman näkymän pelin saamaan palautteeseen, ja siellä voi lukea muiden pelaajien

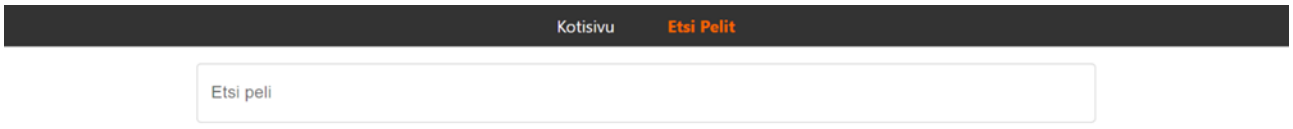
kokemuksia pelistä. Näin käyttäjät voivat helposti tutustua eri pelien arvosteluihin, vertailla niitä ja saada paremman käsityksen pelien laadusta ja pelattavuudesta ennen omaa arviointiaan

Kuva 4 *Tässä näkyy kaikki arvostelut pelille mitä sille on mahdollista antaa (kuvassa pelille on tosin annettu vain 1 arvosana)*

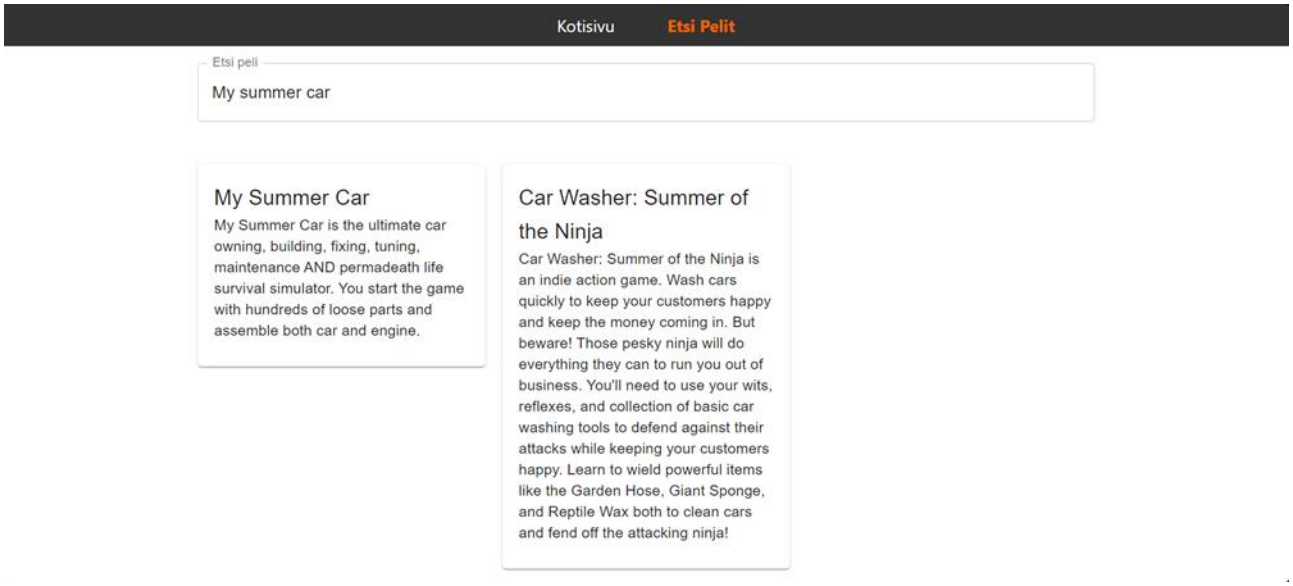
Jos peliin on jätetty arvosteluja, ne näkyvät tällä sivulla. Voit selata muiden käyttäjien näkemyksiä ja kommentteja pelistä, ja saat näin paremman käsityksen pelin sisällöstä ja laadusta. Arvostelut voivat käsitellä mitä tahansa pelin eri osa-alueita, kuten grafiikkaa, pelattavuutta, tarinan kerrontaa, musiikkia ja pelin laatua että löytyykö sieltä paljon bugeja, jotka saattavat vaikuttaa pelikokemukseen.

Tämä auttaa käyttäjää arvioimaan, vastaako peli odotuksiansa tai onko se sellainen, jota hän haluaisi kokeilla tai välttää kokonaan. Arvostelujen kautta voi myös nähdä, kuinka peli on otettu vastaan eri käyttäjien keskuudessa ja onko se saanut positiivista vai negatiivista palautetta.

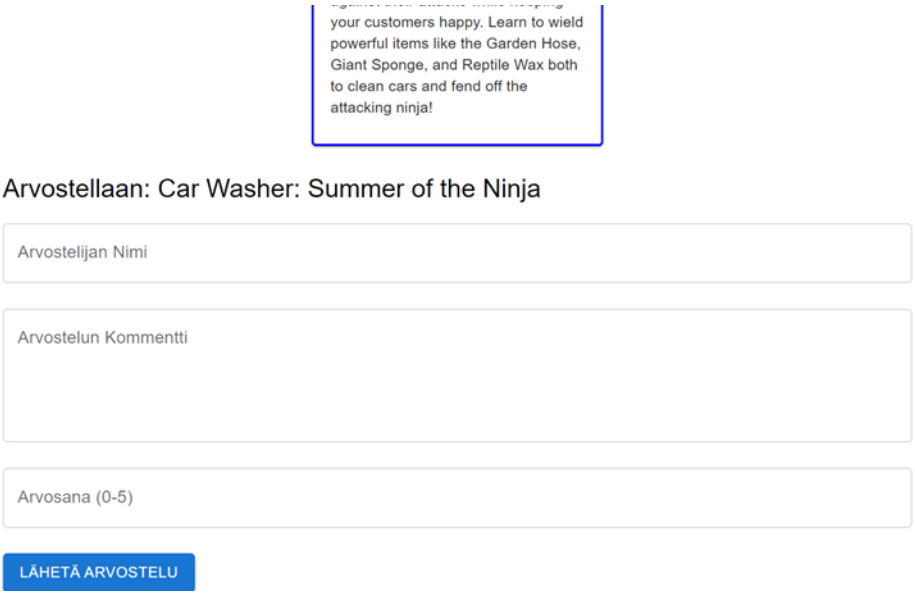
Sivulta voit myös nähdä, onko pelillä erityisiä vahvuuksia tai heikkouksia, jotka eivät välttämättä ilmene pelin virallisista tiedoista. Kommentit voivat sisältää henkilökohtaisia kokemuksia siitä, miten peli on vaikuttanut pelaajiin, joko nostalgisesti tai uusilla tavoilla. Tämä kaikki auttaa sinua päättämään, onko peli se, mitä etsit, ja voitko nauttia siitä samalla tavalla kuin muut pelaajat.



Kuva 5 *Etsi Pelit sivulla pystyy etsimään pelejä ja se sitten näyttää ne, kun etsii (tässä ei ole vielä etsitty peliä).*



Kuva 6 *Tässä näkyy kuinka etsi toiminto toimii kun se löytää pelin tai pelejä.*



Kuva 7 *Kun peli on valittu sille voi jättää arvostelun, arvostelun voi jättää ainoastaan jos joku peli on valittu. (arvosanan voi antaa vain 0–5 ja jos antaa yli tämä tietokanta ei salli sitä.*

Tällä sivulla käyttäjä voi etsiä videopelejä, jotka häntä kiinnostavat. Etsiminen tapahtuu helposti pelin nimellä, ja kun rajapinta löytää hakutulokset, ne esitetään visuaalisessa muodossa. Hakutuloksista luodaan materiaalipelikortteja, joissa näkyy pelin nimi ja lyhyt kuvaus pelistä tai sen sisällöstä, kuten mitä pelissä tehdään ja mikä tekee siitä erityisen.

Kun käyttäjä valitsee jonkin peli kortin, peli kortin ympärille ilmestyy tummansininen reuna, joka kuvastaa, että peli on valittu. Tämä visuaalinen merkki auttaa käyttäjää tunnistamaan, minkä pelin hän on valinnut. Valitun kortin alapuolelle ilmestyy kentät, joihin käyttäjä voi jättää oman nimimerkkinsä, arvosanansa ja kommenttinsa. Kommentissa voi esimerkiksi kertoa, kuinka kyseinen peli oli hänen lapsuutensa suosikki tai miten sen pelaaminen muutti hänen elämänsä.

Arvostelut voivat sisältää pelin parhaat ja huonoimmat puolet, pelattavuuden, tarinan ja visuaaliset elementit, joten muut käyttäjät voivat saada kattavan käsityksen pelistä. Jos peli on jollain tavalla erityinen, käyttäjä voi jakaa henkilökohtaisia kokemuksiaan ja tarinoita siitä, miksi peli on jäänyt mieleen. Tämä auttaa luomaan yhteisön, jossa pelaajat voivat jakaa omia muistojaan ja mielipiteitään sekä löytää samankaltaisia pelikokemuksia muiden käyttäjien arvosteluista.

3.5 Github Linkit

https://github.com/PortfolioOrg/PortfolioBE - backend linkki.

https://github.com/PortfolioOrg/PortfolioFE - frontend linkki.

4 Pohdinta

Alun perin ajattelin, että sivusta tulisi hyvin yksinkertainen, mutta huomasin pian, että se vaatii paljon enemmän työtä. Olen kehittynyt projektin aikana, erityisesti oppimalla yhdistämään tietokantoja ja toteuttamaan full-stack-ratkaisuja.

Opin yhdistämään kahden eri tietokannan tiedot, kuten ID:t, ja yhdistämään tietoja omasta tietokannastani toiseen tietokantaan ohjelmointirajapinnan kautta. Lisäksi opin luomaan oman MySQL-tietokannan ja hakemaan siihen sisältöä React-frontendin kautta.

Huomasin, miten ohjelmointirajapinnat toimivat eri tavoin: jotkut saattavat rajoittaa pyyntöjen määrää ja antaa HTML 429 -virheilmoituksen, kun tämä raja ylittyy. Virheilmoitus tarkoittaa, että pyyntöjä on lähetetty liikaa (HTML 429: Liian monta pyyntöä). Kun tämä tapahtuu, ohjelmointirajapinnan käytöstä riippuen voi kestää tunti tai useita tunteja ennen kuin se sallii uusia pyyntöjä. Tutkin asiaa ja selvitin, että käyttämäni Internet Games Database -ohjelmointirajapinta hyväksyy vain 4 pyyntöä kerrallaan. Jos tämä raja ylittyy, tulee 429-virhe. Tämän vuoksi loin koodin, joka seuraa lähetettyjen pyyntöjen määrää. Jos se ylittää neljän pyynnön rajan, ohjelma antaa ilmoituksen rajan saavutuksesta ja odottaa 1 sekunnin ajan ennen kuin se lähettää uusia pyyntöjä, näin välttäen 429-virheilmoituksen.

Voisin kehittää itseäni tulevaisuudessa, sillä tavalla, että jos ja kun joudun käyttämään IGDB:n kaltaista ohjelmointirajapintaa, luon siihen oman välimuistin palvelimen (Cross-Origin Resource Sharing, CORS Proxy) eri alkuperää olevien resurssien yhdistämistä varten. Tämä sallisi sen, että voisin päästä käsiksi IGDB:n tietokantaan ilman ongelmia. Normaalisti CORS estää pääsyn IGDB:n tietoihin suoraan selaimella, mutta käyttämällä CORS Anywhere -demoa voi väliaikaisesti pyytää luvan käyttää sen toimintoja. Tämä on kuitenkin vähemmän tehokas vaihtoehto verrattuna oman CORS Proxy:n luomiseen.

Lähteet

Internet Games Database. Luettavissa: https://www.igdb.com/ Luettu: 20.1.2025.

IGDB Alkuun pääseminen. Luettavissa: https://api-docs.igdb.com/#getting-started Luettu: 20.1.2025.

Java Spring Boot Tietoa. Luettavissa: https://www.ibm.com/topics/java-spring-boot Luettu: 10.1.2025.

ReactJS Mikä se on. Luettavissa: https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs Luettu: 10.1.2025.

Mikä on Scrum. Luettavissa: https://aws.amazon.com/what-is/scrum/ Luettu: 10.1.2025.

Roger S. Pressman, 2000, Darrel Ince: Software Engineering, A Practitioners Approach. London: McGraw-Hill, cop. 2000. London.

AG Grid. Luettavissa: https://www.ag-grid.com/javascript-data-grid/reference/ Luettu: 24.3.2025.

HTML 429 Liian Monta Pyyntöä Virhe. Luettavissa: https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/429 Luettu: 23.3.2025.

CORS Anywhere Demo. Luettavissa: https://cors-anywhere.herokuapp.com/corsdemo Luettu: 24.3.2025.