Vuokranseuranta postinumeroittain Vue 3 -ohjelmistokehyksellä

Eetu Parkkonen

Opinnäytetyö, AMK
Joulukuu 2024
Tieto- ja viestintätekniikan tutkinto-ohjelma (AMK)

Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Kuvailulehti

Parkkonen, Eetu

Vuokranseuranta postinumeroittain Vue 3 -ohjelmistokehyksellä

Jyväskylä: Jyväskylän ammattikorkeakoulu. Joulukuu 2024, 25 sivua.

Tieto- ja viestintätekniikan tutkinto-ohjelma. Opinnäytetyö AMK.

Julkaisun kieli: suomi

Julkaisulupa avoimessa verkossa: Kyllä

Tiivistelmä

Opinnäytetyön tavoitteena oli suunnitella ja toteuttaa moderni vuokranseurantaan käytettävä sovellus, joka mahdollistaisi seurata postinumeroittain tiettyjen alueiden vuokrakehitystä taulukko- ja viiva-kuvaajien avulla. Sovellukseen valittiin moderneja teknologioita, jotta käyttäjä saisi parhaan käyttökokemuksen, sekä sovellus saisi pitkän käyttöiän. Vuokranseurantaan keskittyvä sovellus palvelee vuokramarkkinoista kiinnostuneita henkilöitä, kuten sijoittajia.

Vuokranseuranta-sovellus on Vue 3 -sovelluskehyksellä rakennettu verkkosovellus, joka hakee Axios-paketin avulla tilastokeskuksen rajapinnasta vuokrahintoja neliömetreittäin. Käyttäjä voi hakea eri kokoisten, sekä eri alueiden asuntojen hintoja käyttämällä Vuetify-käyttöliittymäkirjaston syöttökenttiä. Palautettua JSON-muotoista dataa näytetään HTML-taulukossa, sekä ApexCharts.js-viivakuvaajassa, jotta käyttäjä voi tarkastella vuokrahintoja selkeästi. Käytön sujuvuutta varten dataa säilytettiin local storagessa, mikä vähentää toistuvien verkkopyyntöjen tarvetta.

Opinnäytetyö toteutettiin soveltavana tutkimuksena ja lopputuloksena syntyi selkeä ja toimiva sovellus, joka helpottaa vuokrien seuraamista. Käyttäjä voi tarkastella postinumeroalueittain vuokrahintoja nopeasti ja vaivattomasti.

Tuloksista päätellen vuokran seurannasta onnistuttiin tekemään helppoa yksinkertaisella, sekä nopealla käyttäjäkokemuksella. Jatkossa sovellusta voidaan laajentaa monipuolistamalla sen ominaisuuksia, kuten ottamalla käyttöön erilaisia kuvaajia, sekä hakemalla monipuolisempaa dataa.

Avainsanat (asiasanat)

ApexCharts.js, TypeScript, Vite, Vue.js, Web-sovellus

Muut tiedot (salassa pidettävät liitteet)

-

jamk

Description

Parkkonen, Eetu

Rental tracking by postal code with Vue 3 framework

Jyväskylä: JAMK University of Applied Sciences, December 2024, 25 pages.

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

Permission for open access publication: Yes

Language of publication: Finnish

Abstract

The purpose of the thesis was to design and implement a modern web application for rental tracking, enabling users to monitor rental price trends in specific areas by postal code using table and line charts. Modern technologies were chosen for the application to ensure the best user experience and a long lifespan for the application. The rental tracking application serves individuals interested in the rental market, such as investors.

The rental tracking application is a web-based solution built with the Vue 3 framework. The app retrieves rental prices per square meter from the StatFin API using the axios package. It retrieves rental prices of apartments of different sizes and regions using Vuetify's user interface input fields. The returned JSON data is displayed in an HTML table or in a line chart created with ApexCharts.js, allowing users to clearly examine rental prices. To improve usability, data is stored in local storage, reducing the need for repeated network requests.

The thesis was conducted as applied research, resulting in a clear and functional application that simplifies rental tracking. Users can quickly and effortlessly review rental prices by postal code area.

The results indicate that rental tracking was successfully made with a simple and fast user experience. In the future, the application could be expanded by adding more features, such as implementing different types of charts or retrieving data from different sources as well.

Keywords/tags (subjects)

ApexCharts.js, TypeScript, Vite, Vue.js, Web application

Miscellaneous (Confidential information)

-

Sisältö

Kuviot

Taulukot

1 Johdanto

1.1 Työn lähtökohdat ja tavoite

Tämän opinnäytetyön aiheena oli toteuttaa käyttäjäystävällinen verkossa toimiva vuokrien seurantaan tarkoitettu sovellus käyttäen moderneja web-teknologioita. Sovelluksen avulla pystyisi seuraamaan vuokra-asuntojen hintakehitystä postinumeroittain valitulla ajanjaksolla. Datan visualisointiin käytettäisiin taulukko- ja viivagraafeja. Sovellusta oli tarkoitus käyttää opinnäytetyön tekijän henkilökohtaisena työkaluna asuntosijoittamiseen liittyvässä analyysissä muiden sovellusten ohella.

1.2 Tutkimusmenetelmä

Opinnäytetyön projektissa käytettiin soveltavaa tutkimusmenetelmää. Soveltavassa tutkimuksessa käytetään aikaisempaa tietoa ja taitoa, joiden avulla rakennetaan jotain uutta, tai parannetaan jotain vanhaa. Tämän opinnäytetyön tapauksessa kehitettiin käytännön sovellus.

Opinnäytetyössä haettiin vastausta seuraaviin tutkimuskysymyksiin:

1. Kuinka verkkosivun käytön voi tehdä mahdollisimman sujuvaksi ja nopeaksi?
2. Miten tehdä vuokran seurannasta helppoa käyttäjälle?
3. Kuinka visualisoida monen vuoden vuokranseurannan tulokset?

2 Teknologiat

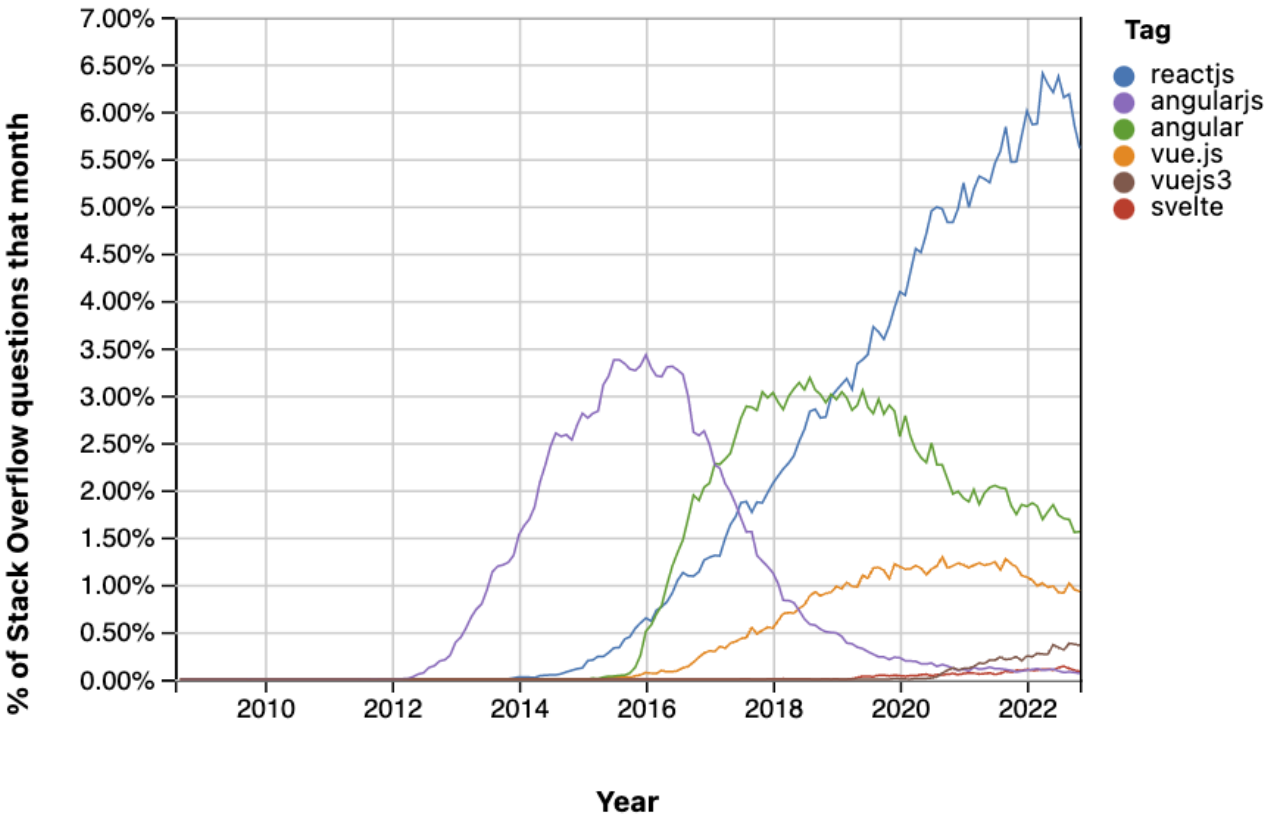
2.1 Teknologioiden valintaperusteet

Teknologioiden valintoihin vaikutti niiden käytettävyys, suosio ja niiden aikaisempi käyttö, koska kehittäjällä oli aikaisempaa kokemusta kyseisistä teknologioista. Myös paketit ja teknologiat, joita tässä opinnäytetyössä käytetään ovat hyvin ylläpidettyjä, joten ne ovat todennäköisesti tulevaisuudessakin vielä toimivia. TypeScriptin valinta oli koodin luettavuuden helpoittamiseksi, koska tyypitettyä koodia on helpompi lukea kuin dynaamista.

2.2 Vue 3

2.2.1 Yleistä

Vue on JavaScript-sovelluskehys, jolla rakennetaan nettisivuja. Vue perustuu HTML:än, CSS:än ja JavaScriptin yhdistelmään, mahdollistaen dynaamisten ja reaktiivisten käyttöliittymien kehittämisen. Vue on tällä hetkellä kolmanneksi suosituin frontend-sovelluskehys. (Ks. Kuvio 1.) Sovelluskehyksen käytössä on monia hyötyjä, koska esimerkiksi dynaamisten muuttujien ja reaktiivisuuden eli JavaScriptin muuttujien tilojen muutosta on paljon helpompi seurata ja muuttaa DOM:ia samalla. Vuella pystyy tekemään pelkkiä komponentteja, joita upotetaan toisille verkkosivuille, tai sillä pystyy tekemään esimerkiksi SPA:n. Vue on avoimen lähdekoodin ohjelmistokehys, sen on kehittänyt Evan You ja se on julkaistu vuonna 2014. (You 2014.)



Kuvio 1. Stack Overflowssa web-sovelluskehysten suosio (Benny Lane 2024.)

2.2.2 Vuetify

Vuetify on voimakas UI-kirjasto, jonka avulla voi helposti rakentaa tyyliteltyjä komponentteja Vuen kanssa. Se on avoimen lähdekoodin kirjasto ja se on ilmainen käyttää. Vuetifylla voidaan luoda yleiskäyttöisiä komponentteja, joita voidaan hyödyntää monella sivulla. Lisäksi teemat pystytään määrittämään dynaamisesti vuetifyn avulla, mikä vaikuttaa sivun värimaailmaan. (Introduction N.d.)

2.2.3 DOM

Document Object Model on ohjelmointirajapinta web-asiakirjoille. DOM sisältää solmuja ja objekteja, joiden avulla ohjelmointikielet voivat olla vuorovaikutuksessa sivun kanssa. JavaScript on yleisin ohjelmointikieli, jolla web-asiakirjaa muokataan Document Object Modelin avulla. DOM ei itsessään ole ohjelmointikieli, mutta ilman sitä ei JavaScript-kielellä olisi käsitystä verkkosivusta ja sen osista. DOM on suunniteltu olemaan riippumaton ohjelmointikielistä, joten sitä pystyy käyttämään millä tahansa ohjelmointikielellä, mutta yleisimmin sitä käytetään vain JavaScript-kielen kanssa. (Introduction to the DOM N.d.)

2.3 ApexCharts

ApexCharts on avoimen lähdekoodin kirjasto, jonka voi ladata käyttöön omaan projektiin. Kirjasto on moderni ja helppokäyttöinen. Kuviossa 2 on näkyvissä esimerkki hallintapaneelista, jolla on ApexCharts-graafeja. ApexChartsilla saa vaivattomasti luotua kauniita visualisointeja datalle. Kirjasto on myös hyvin dokumentoitu ja ApexCharts tukee suoraan Vue-sovelluskehystä. (APEXCHARTS.JS 2024.)

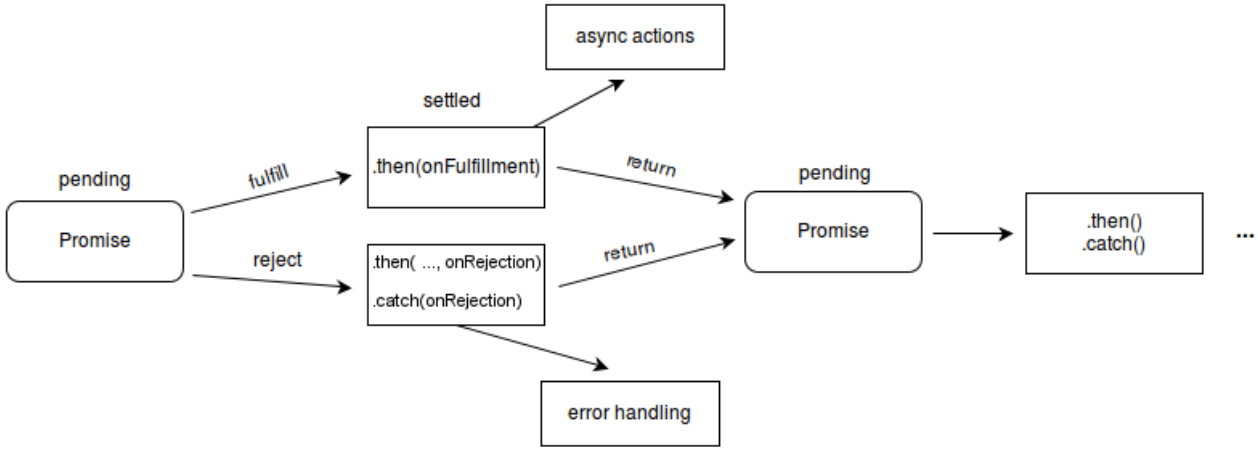
Kuvio 2. ApexCharts-hallintapaneeli (APEXCHARTS.JS. 2024.)

2.4 JavaScript-verkkopyynnöt

2.4.1 JavaScript Promise

JavaScriptissä on Promise-objekti, joka kuvastaa asynkronisen operaation tulosta. Promisea käytetään verkkopyyntöjen kanssa. Promisen arvoa ei tiedetä sen tekohetkellä, vaan JavaScript odottaa sen valmistumista. Promisella on kolme tilaa, se on joko ”pending” eli vireillä, ”fulfilled” eli täytetty tai ”rejected” eli hylätty (Ks. Kuvio 3.)

Kun Promise tehdään, niin se on ensin aina vireillä tilassa, kunnes se täytetään tai hylätään. Promisen täytyttyä kutsutaan sen then-käsittelijää, joka voi tehdä Promisen jälkeistä käsittelyä kun esimerkiksi dataa tulee. (Promise N.d.)



Kuvio 3. Promisen elinkaari (Promise N.d.)

2.4.2 Axios

Axios on npm-paketti, jolla pystytään tekemään HTTPS- eli verkkopyyntöjä. Axios on isomorfinen eli sitä voidaan käyttää selaimessa ja Node.js-palvelimella samalla koodikannalla. (Getting started N.d.)

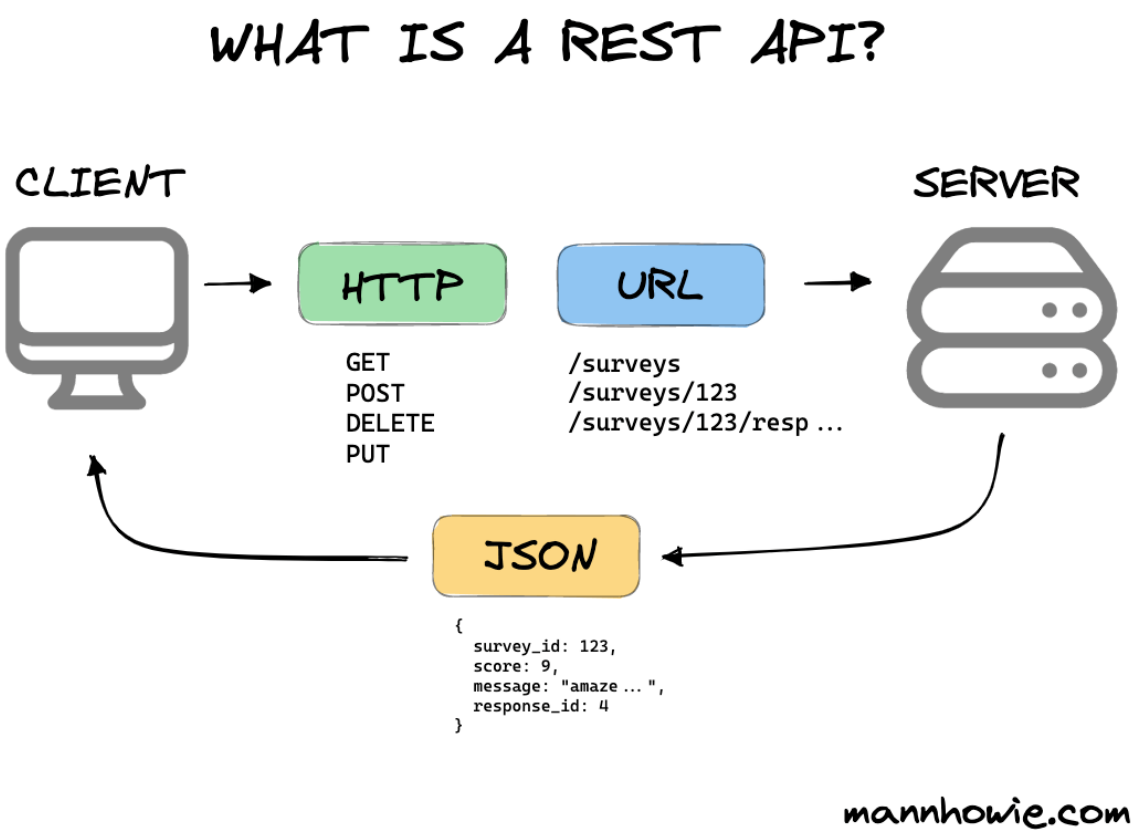
JavaScriptissä on sisäänrakennettu Fetch API, jolla pystytään tekemään verkkopyyntöjä ilman Axios-pakettia. Axios tukee suoraan monia asioita, kuten HTTP-sieppaajia ja sillä saa tehtyä esimerkiksi automaattista JSON-datan muunnosta. Axiosissa on automaattinen virheen keruu toiminto, kun taas Fetch:ssä joutuu itse tekemään virheen käsittelyn. Toisin sanoen Axios on nopeampi käyttöönottaa. Fetch tukee paremmin kustomisointia, koska se on JavaScriptin natiivi tapa tehdä verkkopyyntöjä. (Ashley Innocent 2024.)

2.5 API

API on ohjelmointirajapinta, jonka avulla ohjelmat pystyvät keskustelemaan toisilleen protokollien ja ohjeiden mukaisesti. API:t ovat ratkaisevia ohjelmien tekijöitä, joita ilman ei voitaisi tehdä useita asioita. Esimerkiksi YouTube-rajapinnan avulla voit lisätä videoita omalle verkkosivustollesi. (Coursera Staff 2024.)

Courseran henkilökunnan mukaan rajapintoja on monia erin tyyppisiä, kuten avoimia rajapintoja, joita kuka vain pystyy käyttämään tai integroimaan sovelluksen toimimaan niiden kanssa. Rajapintoja on myös yksityiä, joka tarkoittaa sitä, että esimerkiksi yrityksellä voi olla yksityinen rajapinta jota vain heidän ohjelmansa voi käyttää. (Coursera Staff 2024.)

Toisaalta nykypäivänä suosituin rajapintatyyppi on REST API, joka on monipuolinen käyttöliittymä esimerkiksi JSON-datan hakemiseen ja lähettämiseen HTTP-pyyntöjen avulla. Rest-rajapinnat sisältävät yleisesti GET-, POST-, PUT- ja DELETE-toiminnot, joilla voidaan muuttaa pinnan alla olevaa tietorakennetta. Visuaalinen esimerkki Rest-rajapinnasta kuviossa 4. (Axandria Shepard 2023.)



Kuvio 4. Esimerkki Rest-rajapinnasta (Howie Mann 2023.)

2.6 SCSS

CSS eli Cascading Style Sheet on verkkosivujen tyylittelyn kieli. CSS:tä on luotu kehittyneempi versio joka on SCSS eli Sassy Cascading Style Sheets. Eroja CSS ja SCSS välillä kuvataan taulukossa 1. SCSS myös tukee muuttujien luontia sekä luokkien sisäkkäistä kirjoittamista. (Difference Between CSS and SCSS N.d.)

Taulukko 1. CSS ja SCSS eroja (Yoshitaka Shioutsu 2022.)

SCSS	CSS
SCSS on Sass-esisuorittimen syntaksi	CSS on tyylitiedostojen kieli
Tukee pitkään käytössä olleita muuttujia, joita voi käyttää missä tahansa tyylitiedostossa	Natiivimuuttujien tuki on melko uusi ominaisuus ja niitä voidaan käyttää vain arvojen ja käyttöliittymän tokenien tallentamiseen
Tukee sisäkkäistä syntaksia, jossa voi saumattomasti sisäkkäistää sekä ominaisuuksia että muita valitsimia	Tukee sisäkkäisyyttä vain yksittäisien valitsimien ominaisuuksille, mutta ei muille valitsimille
Tukee mixinejä	Ei tue mixinejä

2.7 Vite

Vite tulee ranskan kielestä ja se tarkoittaa nopeaa. Vite on työkalu, joka rakentaa ohjelman sekä pyörittää sitä palvelimena. Vite on lisäksi hyvin laajennettava JavaScript- ja Plugin-rajapinnan avulla. (Getting started N.d.)

Viteä on optimoitu monella tavalla. Ensimmäiseksi siihen on implementoitu moduulien laiska-lataus, joka tarkoittaa sitä, että koodia ladataan käyttöön vain silloin kun sitä oikeasti käytetään. Tämän ansiosta myös pakettikoot ovat pienempiä. Toiseksi Vitessä on käytössä niin sanottu tree-shaking eli puun ravistaminen, jonka tarkoitus on eliminoida kuollut koodi ohjelmasta. Kolmanneksi viteen sisäänrakennettu kehittämispalvelin on optimoitu moduulien korvaamisella ja nopealla uudelleenlatauksella. Näiden ansiosta kun ohjelmaan tekee muutoksia, niin vite osaa näyttää muutokset ilman, että tarvitsee ladata uudelleen koko sivua. (Tim Davidson 2023.)

Kuviossa 5 on esimerkki kuinka Vite-komentoja määritetään ohjelmassa.

```json
{
  "scripts": {
    "dev": "vite", // start dev server, aliases: `vite dev`, `vite serve`
    "build": "vite build", // build for production
    "preview": "vite preview" // locally preview production build
  }
}
```

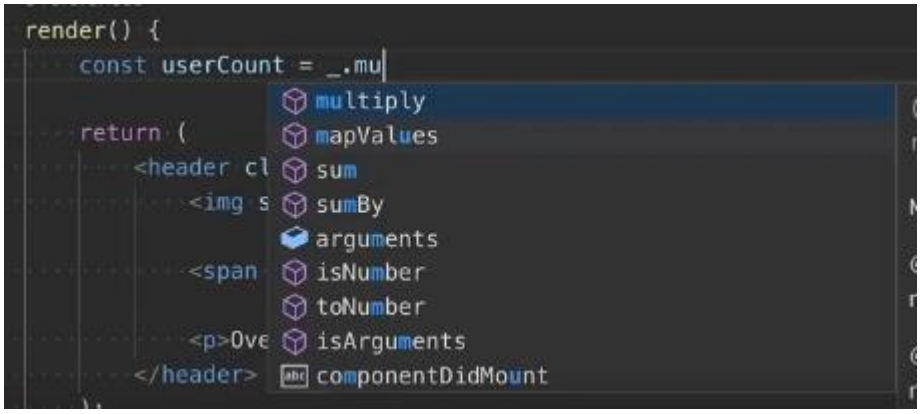
Kuvio 5. Viten palvelimen käynnistyskomento ja rakentaminen (Getting started N.d.)

2.8 Local storage

Local storage on JavaScriptin sisäänrakennettu ominaisuuus, joka antaa JavaScriptin tallentaa selaimeen avain-arvo-pareja. Local storagea käytetään tiedon tallentamiseen paikallisesti, toisaalta sillä ei kannata tallentaa suuria määriä tietoa, jotta data ei katoa jos käyttäjä sulkee selaimen. Storagea voidaan käyttää hyväksi vaikkei käyttäjällä olisi internet yhteyttä, koska data tallennetaan selaimeen paikallisesti. (Noba Obaseki 2024.)

2.9 Visual Studio Code

Visual Studio Code eli VSCode on ohjelmointiin tarkoitettu koodieditori. Siihen on sisäänrakennettu JavaScript IntelliSense, formatointi ja paljon muuta. IntelliSense on systeemi, joka osaa kertoa mitä kaikkea koodia kielellä voi kirjoittaa ja se osaa ehdottaa oikeita metodeita. Kuviossa 6 havainnollistava esimerkki intellisensestä. VSCode sisältää myös suoraan snippettejä JavaScriptille, joiden avulla editori osaa kirjoittaa esimerkiksi for loopin vain kirjoittamalla siihen for. (JavaScript in Visual Studio Code 30.09.2024)



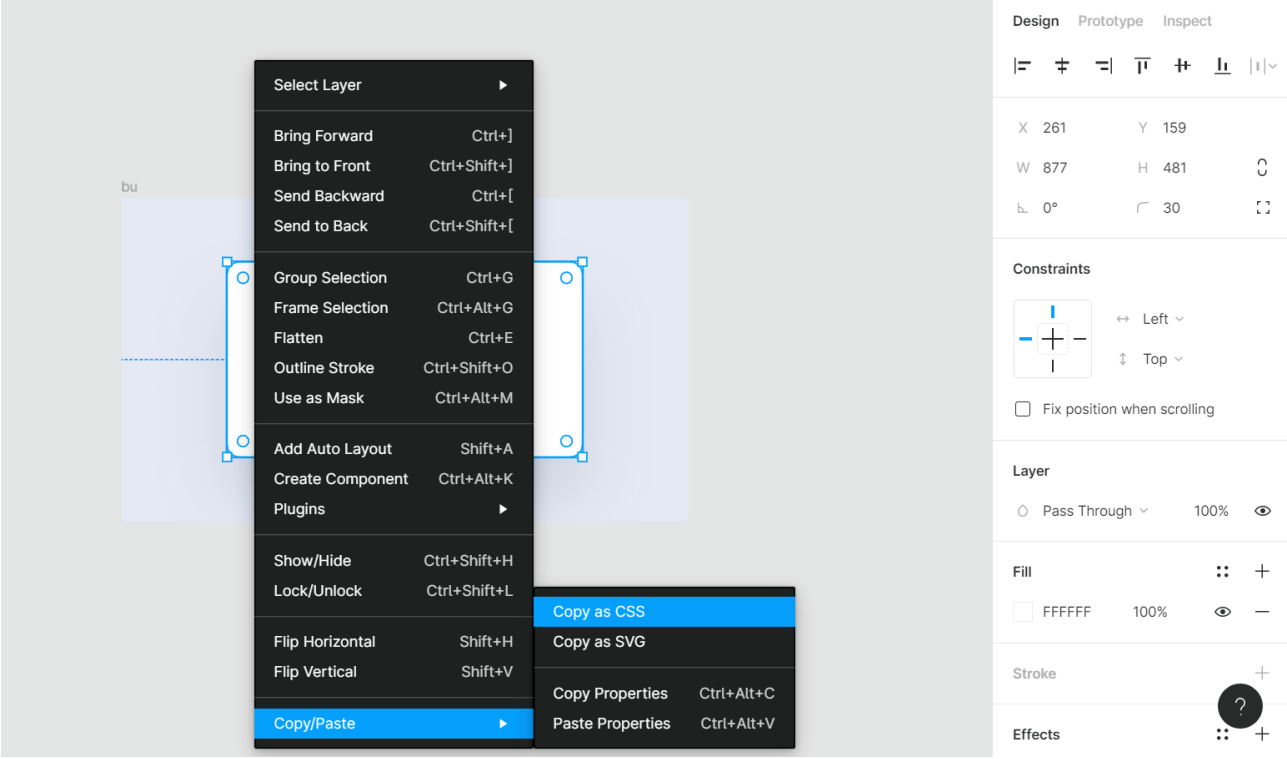
Kuvio 6. Kuvankaappaus IntelliSensestä (JavaScript in Visual Studio Code 30.09.2024)

2.10 Figma

Figma on ilmainen verkossa toimiva nopeakäyttöinen ohjelmistojen suunnittelutyökalu. Figman suosio on kasvanut paljon viime vuosina, koska kuka vain pääsee katsomaan suunnitteluja tiimin sisällä verkosta, eikä kenenkään tarvitse ladata ohjelmaa omalle tietokoneellen. Figma on suunniteltu helpoksi ottaa yrityksen käyttöön, sekä se on hyvin kustannustehokas. (Jurn van Wissen 2020.)

Figmassa saa tehtyä helposti komponentteja. Käyttäjä voi rakentaa esimerkiksi taulukon Figmassa ja taulukosta saa kopioitua tai vietyä kaikki CSS-tyylit suoraan koodiin. (Exporting CSS Code N.d.)

Esimerkki Figmassa CSS-tyylin kopioimisesta kuviossa 7.



Kuvio 7. Esimerkki CSS:n kopioimisesta figmassa (Exporting CSS Code N.d.)

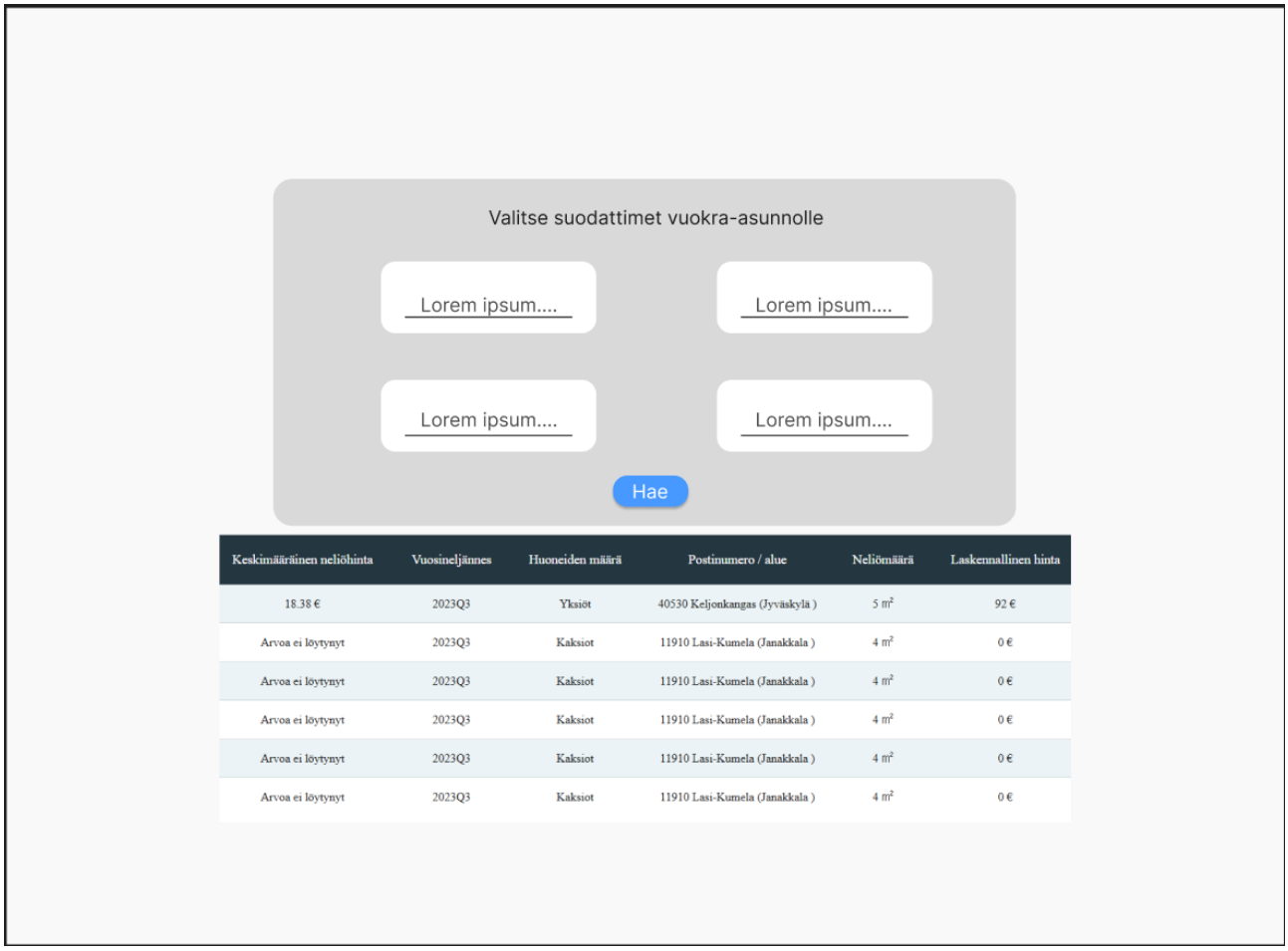
3 Vuokranseuranta-sovellus

3.1 Suunnittelu

Ohjelman suunnittelu alkoi tutkimalla tilastokeskuksen tarjoamia rajapintoja, joista saisi mahdollisimman monipuolista dataa. Sovelluksen tietorakenteiden ja datan käsittelyn suunnittelussa keskityttiin siihen, että rajapinnasta haettava data saataisiin esitettyä käyttäjälle

selkeästi. Tilastokeskuksen tietokanta-aineiston data haetaan JSON-muodossa, joka tallennetaan myös local storageen, jotta rajapintaan ei tarvitsisi tehdä jatkuvasti hakupyyntöjä.

Suunnittelussa otettiin huomioon sovelluksen käyttötarkoitus, joten siitä haluttiin tehdä mahdollisimman yksinkertainen. Suunnittelun aikana käytettiin Figmaa, jolla tehtiin useita verkkosivuun liittyviä mockupeja. Figma on suunnitteluun tehty työkalu, jolla voi helposti piirtää erilaisia verkkosivuihin liittyviä komponentteja, tai vaikka koko sivun mockupin. Kuviossa 8 on esimerkki koko sivun mockupista.



Kuvio 8. Sovelluksen näkymä Figmassa

Sovelluksen suunnittelussa kävi myös ilmi, että dataa pitäisi näyttää aikajärjestyksessä, johon sopi hyvin viivagraafi. Viivagraafin tekoon tarvitsi jotakin JavaScript-ohjelmistokehystä, josta mieleisin tekijälle oli ApexCharts.js. Esimerkki ApexCharts.js viivagraafista kuviossa 9.

Kuvio 9. Esimerkki viivagraafista. (Line Chart N.d.)

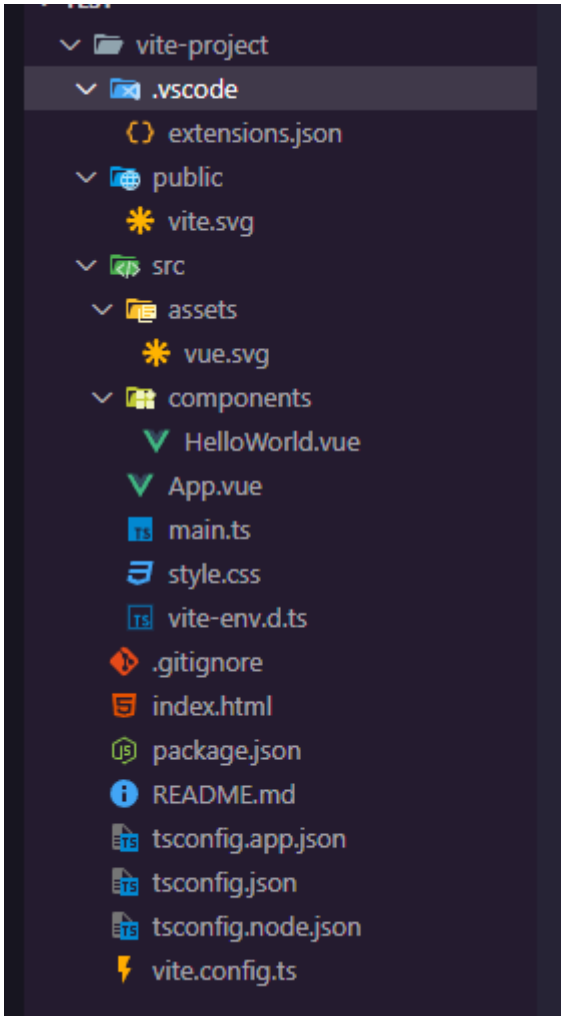
Sovelluksen pääasialliset värit on valittu suunnitteluprosessin aikana ja ne on esitetty kuviossa 10. Värivalinnoissa painotettiin sinertävää sävyä, jotta käyttöliittymästä välittyisi rauhallinen ja miellyttävä vaikutelma. ApexCharts.js-viivagraafi myös tuo esiin datan tyylikkäästi sinertävällä värillä.

```
colors: {
  primary: '#81E9EF',
  secondary: '#4799FF',
  tertiary: '#EBF3F6',
  light: '#BEFFF7',
  background: '#FFFFFF',
  surface: '#FFFFFF',
  error: '#B00020',
  info: '#2196F3',
  success: '#4CAF50',
  warning: '#FB8C00',
},
```

Kuvio 10. Sovellukseen valitut värit

3.2 Toteutus

Uuden Vue-projektin voi aloittaa Viten avulla, käyttäen npm-komentoa npm create vite@latest. Tämä komento luo projektin vakio-kansiorakenteella, mikä nopeuttaa alkuun pääsemistä. Esimerkki uuden projektin kansiorakenteesta on näkyvissä kuviossa 11.



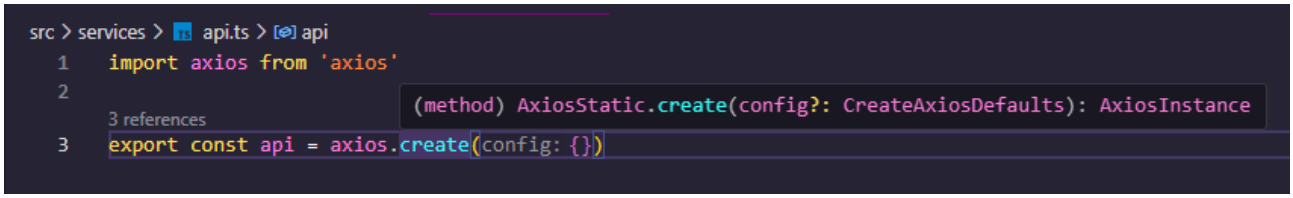
Kuvio 11. Uuden projektin kansiorakenne

Uuteen projektiin sisältyy valmiiksi useita npm-paketteja, joita tarvitaan kehitystyön eri vaiheissa. Ennen kehittämistyön aloittamista sovellukseen asennettiin myös uusia npm-paketteja, koska sovelluksen toteutuksessa tarvittavat paketit olivat jo tiedossa. Paketit asennettiin erotellen kriittiset paketit, joita sovellus tarvitsee toimiakseen, sekä kehitystä tukevat paketit, jotka eivät ole

suoraan sovelluksen toiminnan kannalta kriittisiä. Kuviossa 12 on esimerkki projektiin asennetuista npm-paketeista ja niiden erottelusta.

Kuvio 12. Sovellukseen asennetut npm-paketit

Ohjelmoinnin alottaessa ensimmäisenä toimivaksi laitettiin rajapinta-pyynnöt. Käytössä oli Axios, jonka avulla luotiin yleinen Axios-instanssi, joka hoitaisi kaikki verkkopyynnöt. Käyttäen yleistä Axios-instanssia kaikki verkkopyynnöt käyttävät samaa peruskonfiguraatiota, jolloinka koodia tarvitsee toistaa vähemmän ja sen ylläpidettävyys on helpompaa. Kuviossa 13 on kuva luodusta Axios-instanssista.



Kuvio 13. Yleinen Axios-instanssi

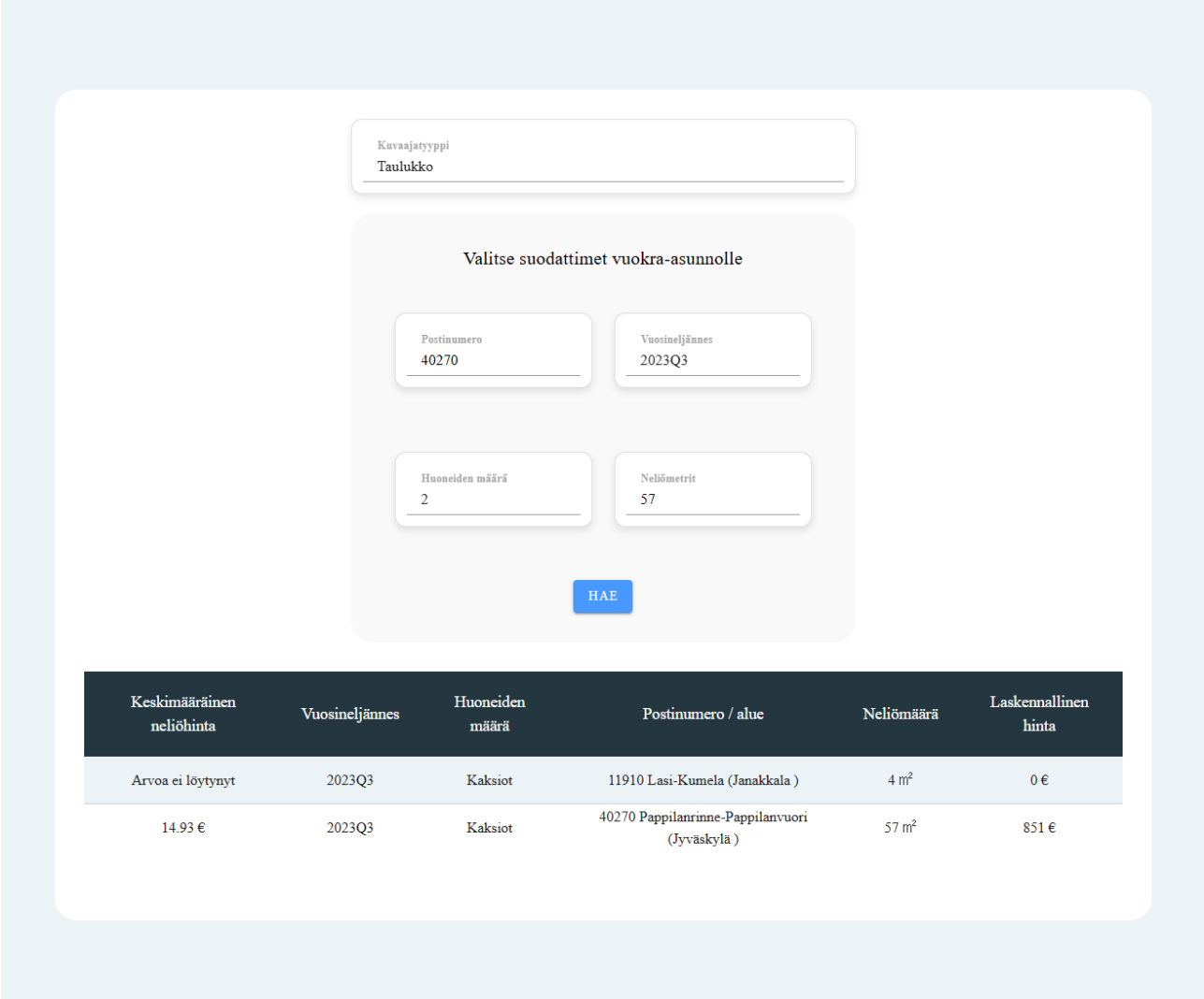
Yleisen instanssin jälkeen tehtiin kovakoodattu nappi, jota painamalla sovellus hakisi JSON-datan rajapinnasta instanssin avulla. JSON-datan haku rajapinnasta oli omituinen, koska se piti tehdä POST-pyynnöllä, kun normaalisti rajapinnoista JSON-dataa haetaan GET-pyynnöillä. Funktio joka haki verkkopyynnön avulla JSON-datan otti vastaan parametreinä vuosineljännen, postinumeron, huonemäärän ja neliömetrien määrän. Näiden tietojen avulla luodaan verkkopyyntö, jota voi katsoa kuviosta 14.

```
export const getRentalPricesBySqM = async (params: InputTypes): Promise<Root> => {
  const { data } = await api.post(url: 'https://pxdata.stat.fi:443/PxWeb/api/v1/fi/StatFin/asvu/statfin_asvu_pxt_13eb.px', data: {
    query: [
      {
        code: 'Vuosineljännes',
        selection: {
          filter: 'item',
          values: [params.quarter],
        },
      },
      {
        code: 'Postinumero',
        selection: {
          filter: 'item',
          values: [params.postal],
        },
      },
      {
        code: 'Huoneluku',
        selection: {
          filter: 'item',
          values: [params.rooms],
        },
      },
      {
        code: 'Tiedot',
        selection: {
          filter: 'item',
          values: ['keskivuokra'],
        },
      },
    ],
    response: {
      format: 'json-stat2',
    },
  });

  return data;
};
```

Kuvio 14. Funktio JSON-datan hakemista varten

Verkkopyyntöjen onnistumisen jälkeen katsottiin sovelluksen yleisilmettä ja koodattiin verkkopyynnön parametrit toimimaan käyttäjän syöttämien arvojen mukaan. Käyttäjä pystyisi siis taulukkokuvaajan tapauksessa syöttämään laatikkoihin postinumeron, vuosineljänneksen, huoneiden määrän, sekä neliömetrit. Rajapinta palautti vain keskimääräisen neliöhinnan, joten oikea hinta piti laskea neliömetrien ja keskimääräisen neliöhinnan avulla. Kuviossa 15 näkee

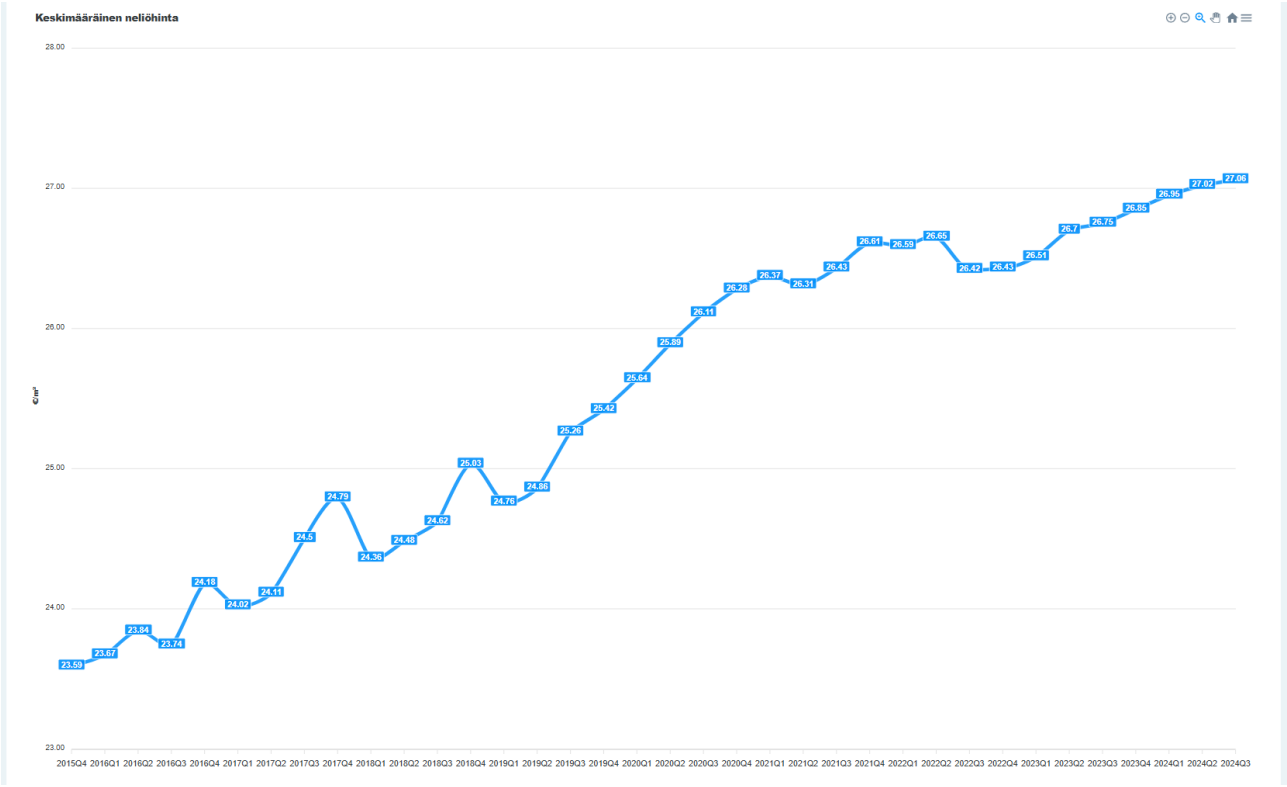
kuinka käyttäjä pystyy syöttämään laatikkoihin arvoja ja kuinka taulukkokuvaaja näyttää rajapinnasta tulleen vastauksen.



Keskimääräinen neliöhinta	Vuosineljännes	Huoneiden määrä	Postinumero / alue	Neliömäärä	Laskennallinen hinta
Arvoa ei löytynyt	2023Q3	Kaksiot	11910 Lasi-Kumela (Janakkala)	4 m²	0 €
14.93 €	2023Q3	Kaksiot	40270 Pappilanrinne-Pappilanvuori (Jyväskylä)	57 m²	851 €

Kuvio 15. Sovelluksen taulukkokuvaaja ja syöttölaatikot

Taulukon valmistuessa mietittiin kuinka viivakuvaaja olisi järkevin tehdä käyttäen ApexCharts.js kirjastoa. Tultiin siihen lopputulokseen, että järkivintä olisi hakea jokainen vuosineljännes rajapinnasta 2015-vuodesta eteenpäin, jotta viivakuvaajaan tulisi tarpeeksi datapisteitä. Kuviossa 16 on sovelluksen viivakuvaaja, jossa on paljon datapisteitä.



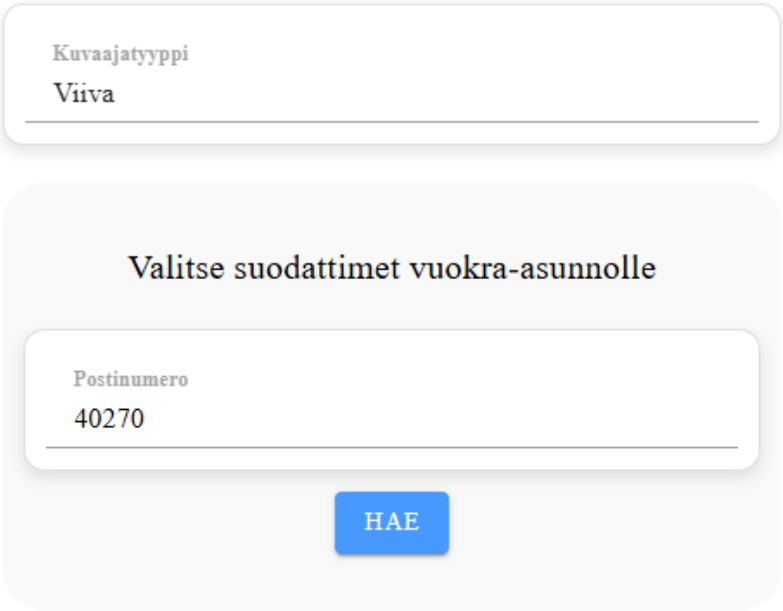
Kuvio 16. Sovelluksen viivagraafi

Datapisteiden hakua varten jouduttiin tekemään funktio, joka luo tietystä vuosineljänneksestä eteenpäin dynaamisesti jokaisen vuosineljänneksen JavaScript-listaan, koska verkkopyynnöt piti tehdä vuosineljännes kerrallaan. Luotu funktio on kuviossa 17, joka luo nykyhetkeen asti vuosineljännekset while loopin avulla. Viivagraafilla haluttiin nähdä vain hinnanseurannan kehitystä tietyllä aikajaksolla, joten käyttäjältä poistettiin näkyvistä kaikki muut valintalaatikot paitsi postinumero.

```
3 references
export function generateQuarterlyDates(startYear: number, startQuarter: number) {
  const quarterItems = [];
  const currentDate = new Date();

  // Get current year and quarter
  const currentYear = currentDate.getFullYear();
  const currentMonth = currentDate.getMonth() + 1; // getMonth() is zero-indexed
  const currentQuarter = Math.ceil(x: currentMonth / 3);

  let year = startYear;
  let quarter = startQuarter;

  // Loop until reaching the current quarter
  while (year < currentYear || (year === currentYear && quarter <= currentQuarter)) {
    quarterItems.push(...items: `${year}Q${quarter}`);

    // Increment the quarter
    quarter++;

    // If quarter exceeds 4, move to the next year
    if (quarter > 4) {
      quarter = 1;
      year++;
    }
  }

  return quarterItems;
}
```

Kuvio 17. Vuosineljänneksien generointi

Kuviossa 18 on näkyvissä kuinka käyttäjä voi valita parametrit kun kuvaajatyypiksi on valittuna viiva. Viivagraafin dataa varten käytettiin kovakoodattua arvoa neliöille ja huoneiden määrälle, jotta hintaseuranta toimisi aina samalla tavalla.



Kuvio 18. Viivagraafin valintaparametrit

4 Tulokset

Opinnäytetyön tuloksena suunniteltiin ja toteutettiin vuokranseuranta-sovellus opinnäytetyön tekijälle yksityiseen käyttöön. Sovellus käyttää moderneja teknologioita, sekä Vue 3-sovelluskehystä. Sovellus tekee asiat, jotka sen suunniteltiin tekevän ja se on helppokäyttöinen ja kevyt sovellus. Kevyen ja helppokäyttöisen sovelluksesta tekee se, että sovellus on täysin yhdellä sivulla toimiva ja siinä ei ole minkäänlaista autentikaatiota. Opinnäytetyön tutkimuskysymyksistä ensimmäiseen kysymykseen vastaus tuli suunnitteluvaiheessa kun kartoitettiin API-pyyntöjä, ulkoasua ja tekniikoita joita sovellus käyttäisi. Kaikki valinnat olivat kevyitä ja nopeakäyttöisiä.

Opinnäytetyön tekijä tunsi useimmat työkalut entuudestaan, joita sovelluksessa käytettiin, mutta sovelluksen suunnittelu tarjosi mahdollisuuden tutustua myös uuteen työkaluun, Figmaan, jota kehittäjä ei ollut aikaisemmin käyttänyt suunnittelussa. Figma osoittautui hyväksi valinnaksi, koska sillä sai vaivattomasti tehtyä mockupeja, sekä käyttöliittymiä. Figmassa suunniteltu selkeä

käyttöliittymä nopeutti myös ohjelmistokehitys-vaihetta, koska tyylejä sai kopioitua suoraan Figmasta.

Sovelluksessa hyödynnetty rajapinta oli monipuolinen, mutta siinä ilmeni myös rajoituksia. Esimerkiksi tietyistä postinumeroista ei ollut lainkaan saatavilla JSON-dataa, mikä vaikutti sovelluksen kattavuuteen. Tulevaisuudessa sovellusta voitaisiin parantaa etsimällä toinen rajapinta, jota voisi käyttää mikäli ensimmäisestä rajapinnasta ei palautuisi tarvittavaa tietoa. Lisäksi sovellukseen voisi lisätä uusia kuvaajatyyppejä, kuten pylväs- tai donitsikaavion.

Toiseen tutkimuskysymykseen vastaus tuli sovelluksen ohjelmoinnin ja testauksen aikana. Vuokran seurannasta voi tehdä helppoa käyttäjälle muun muassa varmistamalla, että sovellus toimii nopeasti. Tämä saavutettiin sillä, että käytettiin moderneja teknologioita, sekä hyödyntämällä rajapinnan vastausten tallentamista selaimen local storageen. Sovelluksen käyttöliittymä myös suunniteltiin käyttäjäystävälliseksi tarjoamalla kaikki keskeiset toiminnot yhdellä selkeällä sivulla, joka teki sovelluksen käytöstä sulavaa.

Viimeiseen tutkimuskysymykseen saatiin vastaus myös sovelluksen ohjelmoinnin aikana, kun taulukkokuvaaja oli toteutettu. Testausvaiheessa huomattiin, että monen vuoden tietojen vertailu oli taulukkokuvaajassa haastavaa, koska manuaalinen lukujen vertailu oli työlästä. Viivagraafi osoittautui tässä tilanteessa huomattavasti selkeämmäksi, koska se mahdollisti pitkän aikavälin muutosten hahmoittamisen yhdellä silmäyksellä.

Loppujen lopuksi työ oli tekijän mielestä onnistunut, sillä se saavutti asetetut tavoitteet. Työn aikana opittiin moderneista teknologioista uutta, kuten Vue 3:sta, jota voi hyödyntää tulevaisuuden projekteissakin.

Lähteet

APEXCHARTS.JS. 2024. Apexcharts.com-verkkosivu. Viitattu 07.04.2024. https://apexcharts.com/

Ashley Innocent. 2024. Axios vs Fetch: Which is best for HTTP requests in 2024? Viitattu 10.04.2024. https://apidog.com/blog/axios-vs-fetch/

Axandria Shepard. 2023. Types of APIs and Use Cases. Viitattu 12.05.2024. https://konghq.com/blog/learning-center/different-api-types-and-use-cases

Benny Lane. 2024. My Go To Tech Stack For Building SaaS Products. Viitattu 10.04.2024. https://medium.com/@bennylaneok/my-go-to-tech-stack-for-building-saas-products-26d0bebb85f6

Coursera Staff. 2024. What Is an API? (+ How Do They Work?). Viitattu 10.04.2024. https://www.coursera.org/articles/what-is-an-api

Difference Between CSS and SCSS. N.d. Byjus.com-verkkosivu. Viitattu 11.05.2024. https://byjus.com/gate/difference-between-css-and-scss/

Evan You. 2014. First week of launching Vue.js. Viitattu 11.05.2023. https://blog.evanyou.me/2014/02/11/first-week-of-launching-an-oss-project/

Exporting CSS Code. N.d. Designcode.io-verkkosivu. Viitattu 05.10.2024. https://design-code.io/figma-handbook-exporting-css-codes

Getting Started. N.d. Axios-http.com-verkkosivu. Viitattu 08.04.2024. https://axios-http.com/docs/intro

Getting Started. N.d. Vitejs.dev-verkkosivu. Viitattu 12.05.2024. https://vitejs.dev/guide/

Howie Mann. 2023. REST API Basics – 4 Things you Need to Know. Viitattu 16.11.2024. https://mannhowie.com/rest-api

Introduction. N.d. Vuetifyjs.com-verkkosivu. Viitattu 11.05.2023. https://vuetifyjs.com/en/introduction/why-vuetify/

Introduction to the DOM. N.d. Developer.mozilla.org-verkkosivu. Viitattu 01.04.2024. https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

JavaScript in Visual Studio Code. 09.05.2024. code.visualstudio.com-verkkosivu. Viitattu 30.09.2024. https://code.visualstudio.com/docs/languages/javascript

Jurn van Wissen. 2020. Everything Developers Need To Know About Figma. Viitattu 05.10.2024. https://www.smashingmagazine.com/2020/09/figma-developers-guide/

Line Chart. N.d. Apexcharts.com-verkkosivu. Viitattu 30.09.2024. https://apexcharts.com/docs/chart-types/line-chart/

Nosa Obaseki. 2024. localStorage in JavaScript: A complete guide. Viitattu 12.05.2024. https://blog.logrocket.com/localstorage-javascript-complete-guide/

Promise. N.d. Developer.mozilla.org-verkkosivu. Viitattu 08.04.2024. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise

Tim Davidson. 2023. What is Vite, And Why Is It Awesome? Viitattu 12.05.2024. https://cleancommit.io/blog/what-is-vite/

Yoshitaka Shiotsu. 2022. What Is SCSS? Learn How To Use SCSS To Style HTML. Viitattu 11.05.2024. https://www.upwork.com/resources/what-is-scss