Valentina Popova

Sovelluksen kehittäminen ilmaisten pysäköintipaikkojen löytämiseen

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Insinöörityö

29.4.2024

Tiivistelmä

Tekijä:	Valentina Popova
Otsikko:	Sovelluksen kehittäminen ilmaisten pysäköintipaikkojen löytämiseen
Sivumäärä:	43 sivua + 2 liitettä
Aika:	29.4.2024
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikan tutkinto-ohjelma
Ammatillinen pääaine:	Ohjelmistokehitys
Ohjaajat:	Janne Salonen

Tässä työssä kehitetään verkkosovellus, jonka avulla voidaan löytää ilmaisia pysäköintipaikkoja kaupunkiympäristössä. Parkkipaikkojen löytämisen ongelma on tärkeä monille kaupungeille, ja tämä sovellus on suunniteltu auttamaan kuljettajia löytämään pysäköintipaikkoja nopeasti ja kätevästi. Tällaisen sovelluksen kehittämisellä on suuri potentiaali parantaa liikkuvuutta kaupungeissa.

Työn pääkohdat ovat sovelluksen kehittämisen päämäärien ja tavoitteiden määrittely, verkkosovelluksen suunnittelun ja kehitysvaiheiden kuvaaminen mukaan lukien toiminnalliset vaatimukset, käyttöliittymä ja arkkitehtuuri. Lisäksi työssä luodaan yleiskatsaus käytettyihin tekniikoihin, mukaan lukien asiakas- ja palvelinosat, sekä geolokaatio- ja kartografiamenetelmät.

Työssä kuvataan sovelluksen pääkomponentit, niiden välinen vuorovaikutus sekä käyttöliittymän sopeutumiskyky ja suorituskyvyn optimointi.

Yhteenveto sisälsi työn tulokset, niiden analyysin ja ehdotuksia sovelluksen jatkokehittämistä varten.

Työn tuloksena on valmis verkkosovellus, joka on tehokas työkalu pysäköintipaikkojen löytämiseen kaupunkiympäristössä. Sovelluksella on potentiaalia edelleen kehittämiseen ja toiminnallisuuden laajentamiseen käyttäjien tarpeet ja kaupunki-infrastruktuurin vaatimukset huomioiden.

Avainsanat: JavaScript, React, NodeJS, MongoDB, Express

Abstract

Author:	Valentina Popova
Title:	Development of an application for finding toll-free parking spaces
Number of Pages:	43 pages + 2 appendices
Date:	29 April 2024
Degree:	Bachelor of Engineering
Degree Programme:	Degree Programme in Information and Communication Technology
Professional Major:	Software development
Supervisors:	Janne Salonen

This project is devoted to the development of a web application for finding toll-free parking places in the urban environment. The problem of finding parking spaces is relevant for many cities, and this application is designed to help drivers quickly and conveniently find parking spaces. The development of such an application has great potential to improve mobility in cities.

The main aspects of the work include defining the goals and objectives for developing the application and describing the stages of the design and development phases of the web application, including functional requirements, user interface, and architecture. The paper also covers an overview of the technologies used, including the client and server side, as well as geolocation and cartography methods. The work describes the main components of the application, the interaction between them, as well as the adaptability of the interface and performance optimization. The conclusion was a summary of the results of the work, an analysis of the data obtained, and proposals for ideas for further development of the application.

The result of the work was a complete web application, which is an effective tool for solving the problem of finding parking spaces in an urban environment. The application has the potential for further development and expansion of functionality, taking into account the needs of users and the requirements of urban infrastructure.

Keywords: JavaScript, React, NodeJS, MongoDB, Express

Sisällys

Lyhenteet

API: *Application Programming Interface*. Sovellusohjelmointi liittymä.

CSS: *Cascading Style Sheets*.

DOM: *Document Object Model*.

HTML: *HyperText Markup Language*. Hypertekstin merkintäkieli.

IDE: *Integrated development environment*. Integroitu ohjelmointiympäristö.

REST: *REpresentational State Transfer*.

UI: *User Interface*. Käyttöliittymä.

UX: *User Experience*. Käyttäjäkokemus.

VCS: *Version Control System*. Versionhallintajärjestelmä.

1 Johdanto

Nykyaikaisissa kaupungeissa pysäköintipaikkojen löytämiseen liittyvät ongelmat ovat kärjistyneet erityisesti kaupungin keskustassa ja tiheimmin asutuilla alueilla. Pysäköintipaikkojen rajoitettu määrä, korkeat hinnat ja paikan etsintään käytetty aika voivat todella aiheuttaa vakavaa harmia ja stressiä autoilijoille. Tämä ongelma ei vaikuta pelkästään kuljettajien mukavuuteen vaan myös kaupungin infrastruktuurin tehokkaaseen käyttöön. Yhteiskunnan digitalisoitumisen ja mobiililaitteiden käytön lisääntymisen yhteydessä ilmaisen pysäköinnin etsintään tarkoitetun verkkosovelluksen kehittäminen näyttää olevan relevantti ja merkittävä askel kohti tämän ongelman ratkaisemista.

1.1 Ongelman merkityksellisyys

Pysäköintipaikkojen ongelmien merkityksellisyys johtuu autokannan kasvusta, kaupunkilaisten määrän kasvusta ja pysäköintipaikkojen riittämättömyydestä. Tämä johtaa ylimääräisiin liikenneruuhkiin, ympäristön saastumiseen ja kuljettajien lisääntyneeseen stressiin (Gössling 2020). Verkkosovelluksen kehittäminen ilmaisten pysäköintipaikkojen etsintään on tulossa kiireelliseksi tehtäväksi kaupunkilaisten liikkuvuuden ja viihtyvyyden parantamiseksi. Ilmaisten pysäköintipaikkojen löytämiseen tarkoitetun verkkosovelluksen kehittäminen on kiireellinen tehtävä, jolla pyritään parantamaan kaupunkilaisten liikkuvuutta ja mukavuutta.

1.2 Työn tavoitteet ja tehtävät

Työn aiheena on sovelluksen kehittäminen ilmaisten pysäköintipaikkojen etsintään.

Verkkosovelluksen kehittämisen tavoitteet ilmaisen pysäköinnin etsintään:

1. Kuljettajakokemuksen parannus. Sovelluskehityksen tavoitteena on työkalun luominen, joka auttaa kuljettajia löytämään nopeasti ja helposti ilmaisia pysäköintipaikkoja kaupungista, mikä vähentää pysäköinnin etsintään kuluvaa aikaa ja stressiä.

2. Pysäköintiresurssien käytön optimointi. Sovelluksen tavoitteena on pysäköintipaikkojen käytön parannus kaupungissa, jolloin käyttäjät voivat löytää nopeasti ilmaisia paikkoja ja välttää ruuhkia.

3. Ympäristön kestävyyden edistäminen. Pysäköintipaikkojen helppo saatavuus auttaa vähentämään ajoneuvoliikennettä, kiertoliikennettä ja päästöjä.

4. Julkisen tilan parannus. Sovelluksen luominen parantaa kaupunkiympäristön viihtyisyyttä ja houkuttelevuutta tehden sen yksinkertaisemmaksi ja helpommin saavutettavaksi kaupungin asukkaille ja vieraille.

Ilmaisen online-pysäköintihakusovelluksen kehittämiseen liittyy useita tekijöitä:

- Autojen määrän kasvu: kun autojen määrä kaupungeissa kasvaa, tarvitaan pysäköintiresurssien tehokasta käyttöä asukkaiden mukavan liikkumisen ja asumisen varmistamiseksi.
- Pysäköintiongelma: pysäköintipaikkojen rajoitus ja kaupungin katujen ruuhkaisuus vaikeuttavat pysäköinnin löytämistä, mikä aiheuttaa ruuhkaa ja ajanhukkaa.
- Teknologinen kehitys: mobiililaitteiden ja paikannusteknologioiden kehittämisen myötä voidaan luoda innovatiivisia ratkaisuja pysäköintipaikkojen etsinnän optimointiin.

Sovelluksen kehittämisen taustalla on tarve ratkaista nykyinen kaupunki-infrastruktuurin ongelma ja ajoneuvojen käytön mukavuuden ja tehokkuuden parannus.

Organisaation tavoitteet:

- Yksinkertaistaa pysäköintipaikkojen etsintää: tarjoaa kuljettajille kätevän ja tehokkaan työkalun, jonka avulla he voivat nopeasti löytää ilmaisia pysäköintipaikkoja kaupunkiympäristössä.
- Kaupunkitilan houkuttelevuuden lisääminen: pysäköintimahdollisuuksien tarjoaminen ilman ajanhukkaa lisää kaupunkien infrastruktuurin houkuttelevuutta asukkaiden ja vierailijoiden kannalta.

Käyttäjien tarpeet:

- Tehokas pysäköintipaikan etsintä: Käyttäjät etsivät keinoja löytääkseen nopeasti ja helposti ilmaisen pysäköintipaikan alueilla, joilla paikkojen määrä on rajattu.
- Säästä aikaa ja vältä stressiä: Kuljettajat haluavat vähentää pysäköinnin etsintään kuluvan ajan ja välttää ympäriajon aiheuttaman stressin ilmaista paikkaa etsiessään.

Ilmaisten pysäköintipaikkojen etsintään tarkoitetun verkkosovelluksen kehittämisellä pyritään vastaamaan käyttäjien tarpeisiin tehokkaaseen ja kätevään pysäköintipaikan etsintään sekä ratkaisemaan kaupunkiympäristön rajoitettuun pysäköintipaikkoihin liittyviä ongelmia.

Tämä työ on omistettu ilmaisen pysäköintipaikan etsintään tarkoitetun verkkosovelluksen toteuttamiselle, joka tarjoaa käyttäjille työkalun tällaisten paikkojen nopeaan ja kätevään etsintään kaupunkiympäristöstä. Työssä tarkastellaan sovelluksen arkkitehtuuria, valittuja teknologioita, päätoimintoja ja toimintaperiaatteita sekä esitellään kehittämisen tulokset ja jatkokehitysnäkymät.

2 Sovelluksen suunnittelu ja kehittäminen

Tässä luvussa esitellään verkkosovelluksen suunnittelu- ja kehitysprosessi ilmaisten pysäköintipaikkojen etsintään. Tarkastelemme sovelluksen toiminnallisia vaatimuksia, käyttöliittymäkuvausta ja käyttötapauskaaviota, joka näyttää käyttäjän ja sovelluksen tärkeimmät vuorovaikutustilanteet (Melnyk, Djahel, Nait-Abdesselam 2019).

2.1 Toiminnalliset vaatimukset

Tässä luvun osassa käsittellään toiminnallisia vaatimuksia kehitettävän verkkosovelluksen ilmaisten pysäköintipaikkojen etsintään. Nämä vaatimukset määrittelevät sovelluksen ydinominaisuudet ja -ominaisuudet, jotka on otettava käyttöön käyttäjien tarpeiden ja suorituskykytavoitteiden saavuttamiseksi.

Verkkosovelluksen toiminnallisten vaatimusten määrittämiseksi analysoitiin käyttäjien tarpeet ja tärkeimmät tehtävät, joita he suorittavat sovellusta käyttäessään. Toiminnallisia perusvaatimuksia ovat mm:

1. Pysäköintipaikkojen etsintä:

    - Käyttäjien pitäisi pystyä etsimään ilmaisia pysäköintipaikkoja.
    - Käyttäjien pitäisi pystyä hakemaan tiettyä sijaintia.
    - Sovelluksessa on annettava tiedot pysäköintipaikoista.

2. Tulosten näyttäminen kartalla:

    - Etsinnän tulokset tulisi näyttää interaktiivisella kartalla, joka auttaa kuljettajaa helposti löytämään jokaisen pysäköintialueen.
    - Kartasta tulee kertoa lähimmät ilmaiset pysäköintipaikat ja niiden sijainti.

3. Pysäköintipaikkojen lisääminen:

    - Käyttäjien pitäisi pystyä lisäämään pysäköintitilaa.
    - Pysäköintipaikkaa lisättäessä tulee olla mahdollista lisätä tietoa tästä paikasta.

4. Mukautuva muotoilu:

    - Sovelluksessa tulee olla mukautuva muotoilu, jotta varmistetaan optimaalinen näyttö ja käytettävyys useilla eri laitteilla, kuten älypuhelimilla, tableteilla ja tietokoneilla.

Näillä toiminnallisilla vaatimuksilla luodaan käyttäjäkokemus, joka vastaa käyttäjien odotuksia ja tekee sovelluksesta tehokkaan työkalun ilmaisten pysäköintipaikkojen etsintään.

2.2 Käyttöliittymä

Tässä luvun osassa tarkastellaan ilmaisten pysäköintipaikkojen etsintään kehitettävän verkkosovelluksen käyttöliittymää (UI). Käyttöliittymällä on ratkaiseva rooli sovelluksen käyttökokemuksessa, joten sen suunnittelu vaatii erityistä huomiota yksityiskohtiin ja käyttäjien käytettävyyteen.

Käyttöliittymän (UI) on oltava intuitiivinen, käyttäjäystävällinen ja esteettisesti miellyttävä houkutellakseen käyttäjiä ja tarjotakseen heille miellyttävän käyttökokemuksen. Verkkosovellus ilmaisten pysäköintipaikkojen etsintään on yksisivuinen sovellus, joka koostuu eri komponenteista. Katsotaanpa verkkosovelluksen käyttöliittymän ilmaisten pysäköintipaikkojen etsintään peruselementtejä ja periaatteita.

1. Pääosa:

    - Sovelluksen pääosa on käyttäjän ja sovelluksen välisen vuorovaikutuksen keskus. Sen tulee sisältää ominaisuuksia, kuten kartan mistä voi nähdä eri pysäköintipaikat ja hakukentän.
    - Kartta on tärkeä käyttöliittymän osa, jonka avulla käyttäjät voivat arvioida visuaalisesti pysäköintipaikkojen sijainnin ja lukumäärän.

- Kartan tulee näyttää nykyinen sijainti, pysäköintialueet ja liikkua tiettyä sijaintia haettaessa.
- Kartasta tulee kertoa lähimmät pysäköintipaikat, niiden lukumäärä ja sijainti, mikä auttaa käyttäjiä tekemään tietoisemman päätöksen pysäköintipaikan valinnassa.
- Hakukentän tulee olla yksinkertainen ja helposti ymmärrettävä, jota käyttäjät voivat käyttää helposti. Hakijoiden tulee saada tieto onnistuneesta syötöstä ja hausta.

2. Tuloksia sisältävä osa:

- Tulosten tulee näkyä hakemussivulla. Eli pysäköintipaikoista täytyy olla luettelo tiedoilla.
- On tärkeää varmistaa, että tulokset näkyvät kätevästi, jotta käyttäjä ymmärtää ja sovellusta käyttäessään.

3. Pysäköintipaikan lisäämisen osa:

- Käyttäjillä on myös mahdollisuus lisätä pysäköintipaikkaa. Paikkaa lisättäessä tulee olla mahdollista lisätä osoite, sallitut pysäköintiajat ja pysäköintiajan määrä.

Näillä käyttöliittymäosilla varmistetaan, että ilmaisia pysäköintipaikkoja etsivää verkkosovellusta on helppoa ja tehokasta käyttää, mikä tekee siitä houkuttelevan käyttäjille ja lisää heidän käyttötyytyväisyyttään (Tidwell, Brewer, Valencia 2020). Verkkoselainsivun käyttöliittymän asettelu on esitetty kuvassa 1.

Kuva 1. Käyttöliittymän kaavio.

2.3 Käyttötapauskaavio

Tässä luvun osassa tarkastellaan käyttötapauskaaviota, joka on tärkeä työkalu suunniteltaessa web-sovellusta ilmaisten pysäköintipaikkojen etsintään. Käyttötapauskaavio auttaa kuvaamaan pääskenaarioita, joissa käyttäjät ovat vuorovaikutuksessa sovelluksen kanssa, ja tunnistaa järjestelmän keskeiset toiminnalliset vaatimukset. Käymme läpi sovelluksemme käyttötapauskaavion rakentamisen perusnäkökohdat ja periaatteet sekä sen merkityksen suunnittelu- ja kehitysprosessissa.

Käyttötapauskaavio on graafinen kuvaus päätilanteista, joissa käyttäjä on vuorovaikutuksessa sovelluksen kanssa. Se kuvaa käyttötapausryhmien ja prosessiin osallistuvien toimijoiden välisiä suhteita ja riippuvuuksia (Ambler 2010).

Käyttötapaukset edustavat järjestelmän toiminnallisia vaatimuksia toimijoiden ja niiden suorittamiensa toimien muodossa.

On tärkeää ymmärtää, että käyttötapauskaavioita ei ole tarkoitettu edustamaan suunnittelua, eivätkä ne voi kuvata järjestelmän sisäistä toimintaa. Käyttötapauskaaviot on tarkoitettu yksinkertaistamaan vuorovaikutusta järjestelmän tulevien käyttäjien, asiakkaiden kanssa ja ovat erityisen hyödyllisiä määritettäessä järjestelmän vaadittavat ominaisuudet. Toisin sanoen käyttötapauskaaviosta kerrotaan, mitä järjestelmän pitäisi olla menetelmien määrittelemättä.

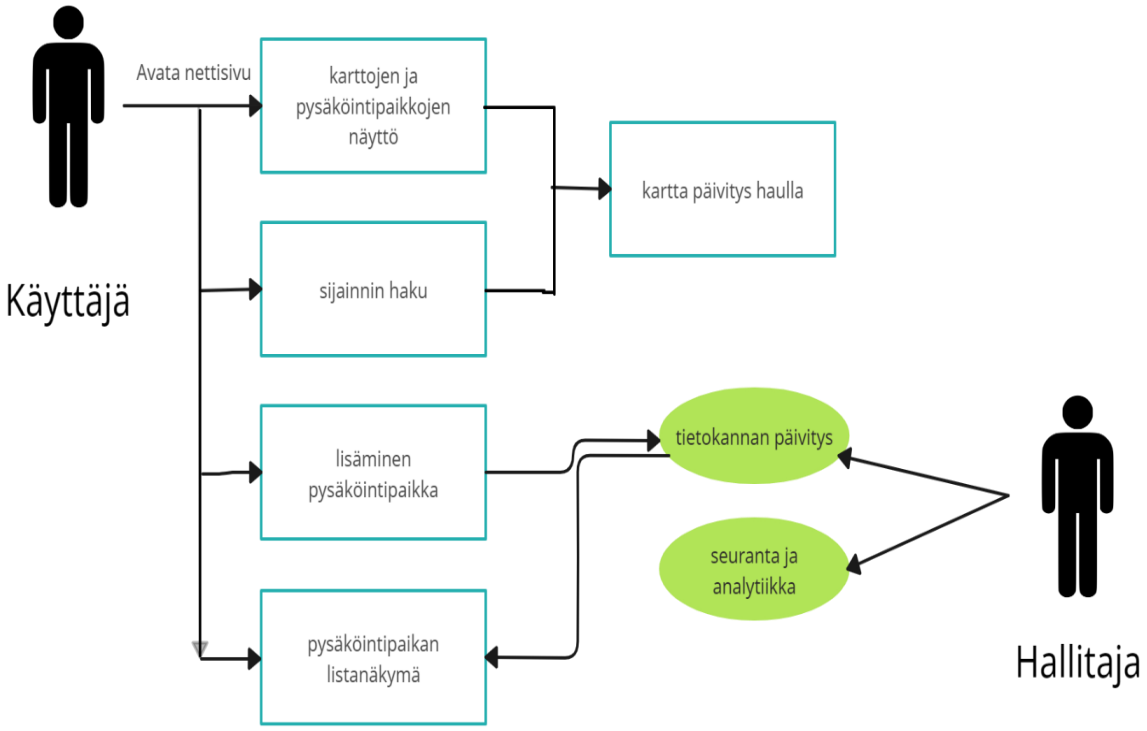
Katsotaanpa peruselementtejä ja periaatteita käyttötapauskaavion peruselementeistä ja periaatteista ilmaisen pysäköintipaikan etsintään verkkosovelluksemme, joka on esitetty kuvassa 2:

Näyttelijä:

- Käyttäjä: Järjestelmän päätoimija, joka käyttää sovellusta ilmaisten pysäköintipaikkojen etsintään.
- Järjestelmänvalvoja: toimija, joka huolehtii järjestelmän hallinnasta ja hallinnasta, mukaan lukien tietokannan hallinta, seuranta ja analytiikka.

Käyttövaihtoehdot:

- Pysäköintipaikkojen etsintä: käyttäjä käynnistää ilmaisten pysäköintipaikkojen etsinnän syöttämällä vaaditun osoitteen tai katsomalla nykyisen sijainnin lähellä olevia pysäköintipaikkoja.
- Hakutuloksen katseleminen: Haun jälkeen käyttäjä voi tarkastella tuloksia kartalla ja luettelossa.
- Pysäköintipaikkojen lisääminen: käyttäjät voivat lisätä pysäköintipaikkoja täyttämällä pysäköintipaikkakortin.



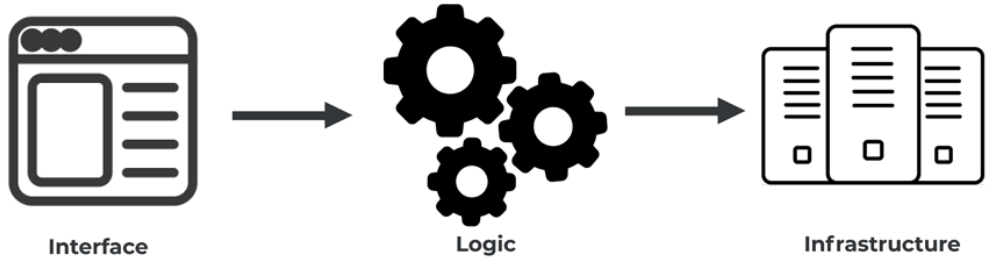
Kuva 2. Proektin käyttötapauskaavio.

Käyttötapauskaavion avulla voi visualisoida tärkeimmät skenaariot käyttäjien vuorovaikutuksesta järjestelmän kanssa ja se on pohjana sovelluksen toiminnallisuuden jatkosuunnittelulle ja kehittämiselle.

3 Sovelluskehitysteknologiat

Kehityksen teknologiset näkökohdat ovat ratkaisevassa roolissa luotaessa web-sovellusta ilmaisten pysäköintipaikkojen löytämiseen. Tässä luvussa käsitellään tärkeimmät tekniikat ja työkalut, joita käytettiin sovelluksen kehittämisessä. Verkkosovelluksen kehittäminen on monimutkainen prosessi, joka vaatii huolellista tekniikoiden valintaa korkean suorituskyvyn, turvallisuuden ja käytettävyyden varmistamiseksi loppukäyttäjälle. Se tunnistaa ohjelmointikielet, puitteet, tietokannat ja muut käytetyt tekniikat sekä perusteet tavoitteiden saavuttamiseksi ja sovellusvaatimukset.

Kuvassa 3 on esimerkki arkkitehtuurikaaviosta tyypillisestä sovelluksesta.



Kuva 3. Kaavio tyypillisestä sovelluksesta.

3.1 Verkkosovelluksen asiakaspuolen tekniikat

Verkkosovelluksen asiakaspuoli on avaintekijä, joka määrää visuaalisen esityksen ja käyttökokemuksen. Tässä alakohdassa käsitellään teknologioita, joilla kehitetään verkkosovelluksemme asiakaspuolta ilmaisten pysäköintipaikkojen löytämiseen.

3.1.1 HTML ja CSS

HTML (HyperText Markup Language) ja CSS (Cascading Style Sheets) ovat tärkeimmät tekniikat, joita käytetään web-kehityksessä verkkosivujen rakenteen ja visuaalisen suunnittelun luomiseen. Ne tarjoavat puitteet sisällön näyttämiselle ja sen muotoilulle käyttäjän selaimessa.

HTML on sivunkuvauskieli, jota käytetään määrittämään verkkosivun rakenne. Se auttaa kehittäjiä määrittelemään osia, kuten otsikoita, kappaleita, luetteloita, kuvia, linkkejä ja paljon muuta. Se tarjoaa joukon tunnisteita, jotka määrittelevät sisällön semantiikan ja sen suhteet.

CSS on tyylisivukieli, joka määrittelee verkkosivun osien ulkoasun ja suunnittelun. Sen avulla kehittäjät voivat hallita sisällön visuaalisen esityksen värejä, fontteja, kokoa, pehmusteita, kohdistusta ja muita näkökohtia. CSS:n

perusajatuksena on, että tyylejä voidaan soveltaa elementteihin eri sivun paikoissa valitsimien ja tyylisääntöjen avulla.

HTML:n ja CSS:n valinta käyttöliittymäkehityksen ydinteknologioiksi johtuu niiden laajasta tuesta verkkoselaimissa ja helppokäyttöisyydestä. HTML tarjoaa selkeän rakenteen sisällön järjestämiseen, kun taas CSS:n avulla voi luoda esteettisesti miellyttäviä ja luettavia malleja. Tämä yksinkertaistaa reagoivan ja kauniin käyttöliittymän luomista, mikä puolestaan parantaa käyttökokemusta (Bowers, Synodinos, Sumner 2011).

3.1.2 JavaScript ja React

JavaScript on tehokas ohjelmointikieli, jota käytetään laajasti dynaamisten ja interaktiivisten verkkosovellusten luomiseen. Sen avulla voi lisätä asiakaspuolen toimintoja, olla vuorovaikutuksessa käyttäjän kanssa ja hallita verkkosivun toimintaa (Flanagan 2020). Kaikki nykyaikaiset selaimet tukevat JavaScriptiä, ja sitä käytetään laajasti verkkokehityksen eri alueilla, kuten animaatioiden luomisessa, lomakkeiden käsittelyssä, pyyntöjen lähettämisessä palvelimelle, DOM:n (Document Object Model) manipuloinnissa ja paljon muuta.

React on Facebookin kehittämä käyttöliittymien luomiseen tarkoitettu JavaScript-kirjasto. Se tarjoaa tehokkaita työkaluja uudelleenkäytettävien ja helppohoitoisten käyttöliittymäkomponenttien luomiseen. Reactin ydinperiaatteet ovat komponenttilähestymistapa ja virtuaalinen DOM, jonka avulla voit luoda dynaamisia ja nopeita verkkosovelluksia. React tarjoaa myös sovelluksen tilan reitityksen ja tapahtumien käsittelyn hallintaan käteviä työkaluja.

DOM (Document Object Model) on dokumenttirakenne objektipuun muodossa, joka edustaa kaikkia web-sivun osia ja niiden attribuutteja. Jokainen DOM:n osa on objekti, jota voidaan manipuloida JavaScriptin avulla muuttaaksesi sen sisältöä, tyyliä ja käyttäytymistä.

DOM tarjoaa sivun sisällön dynaamiseen päivittämiseen tehokkaita työkaluja ilman uudelleenkäynnistystä. JavaScriptin avulla voi lisätä, poistaa ja muokata DOM-elementtejä, käsitellä käyttäjätapahtumia, animoida osia ja paljon muuta. Tämä tekee DOM:sta tärkeän työkalun interaktiivisten verkkosovellusten kehittämisessä.

React käyttää virtuaalista DOM:ia, todellisen DOM:n abstraktiota, jonka avulla voi päivittää sivun sisältöä tehokkaasti sovelluksen tilan muuttuessa. React luo virtuaalisen DOM-puun, jota verrataan todelliseen DOM:iin ja vain muuttuneet osat päivitetään sivulla. Tämä parantaa sovelluksen suorituskykyä ja vähentää selaimen kuormitusta.

DOM:n ja virtuaalisen DOM:n käyttö yhdessä JavaScriptin ja Reactin kanssa antaa kehittäjille mahdollisuuden luoda interaktiivisia ja nopeita verkkosovelluksia, jotka tarjoavat hyvän käyttökokemuksen.

JavaScript on vakioohjelmointikieli verkkokehityksessä, joka tarjoaa korkean suorituskyvyn ja joustavuuden luotaessa asiakaspuolen toimintoja. React puolestaan tarjoaa tyylikkäitä ratkaisuja käyttöliittymien luomiseen, mikä yksinkertaistaa verkkosovelluksen kehittämistä ja ylläpitoa. Reactin käytön ansiosta voimme luoda modulaarisia ja skaalautuvia rajapintakomponentteja, mikä lisää kehittämisen tehokkuutta ja varmistaa korkealaatuisen lopputuotteen.

JavaScriptin ja Reactin valinta etupään verkkosovelluskehityksen pääteknologioiksi johtuu niiden tehokkaista ominaisuuksista ja laajasta suosiosta kehittäjäyhteisössä.

3.1.3 API

API (Application Programming Interface) on menetelmien ja käytäntöjen joukko, jotka määrittelevät kuinka eri ohjelmistokomponentit voivat olla vuorovaikutuksessa keskenään.

Verkkokehityksen yhteydessä API:ta käytetään tyypillisesti määrittämään, kuinka sovelluksen asiakas- ja palvelinpuolet ovat vuorovaikutuksessa.

Ilmaisten pysäköintipaikkojen etsintään tarkoitettu verkkosovellus käyttää myös API:ta tiedon vaihtamiseen asiakkaan (käyttäjän verkkoselaimen) ja palvelimen välillä. Sovellusliittymiä on periaatteessa kahdenlaisia:

- RESTful API: REST (Representational State Transfer) on arkkitehtoninen tyyli, joka määrittelee joukon periaatteita ja käytäntöjä hajautettujen järjestelmien rakentamiseen. RESTful API käyttää tavallisia HTTP-menetelmiä (GET, POST, PUT, DELETE) suorittaakseen toimintoja URL-osoitteina esitetyille resursseille. Esimerkiksi pysäköintipaikkojen luettelon saamiseksi voidaan käyttää HTTP GET -pyyntöä osoitteeseen /parking-spots.
- GraphQL API: GraphQL on Facebookin kehittämä kyselykieli ja ajonaika palvelinpuolen API:ille. Sen avulla asiakkaat voivat pyytää vain tarvitsemaansa tietoa ja tarjoaa joustavamman ja tehokkaamman tavan olla vuorovaikutuksessa palvelimen kanssa. Toisin kuin REST, jossa jokainen päätepiste edustaa tiettyä resurssia, GraphQL:ssä asiakkaat määrittelevät pyydettyjen tietojen rakenteen.

Molempien API-tyyppien tarjoamaan vuorovaikutus sovelluksen asiakas- ja palvelinosien välillä, jolloin voi siirtää tietoja ja suorittaa tarvittavat toiminnot. Ilmaisten pysäköintipaikkojen etsintään tarkoitettu verkkosovellus käyttää API-liittymiä kyselyihin käytettävissä olevista pysäköintipaikoista, lähettää arvosteluja ja arvioita, todentaa käyttäjiä ja muita tärkeitä toimintoja. Päätin käyttää RESTful-sovellusliittymää pääasiallisena API-tyyppinä tälle sovellukselle.

RESTful API tarjoaa tehokkaan ja joustavan viestinnän sovelluksen asiakas- ja palvelinosien välillä, mikä yksinkertaistaa sovelluksen kehittämistä ja ylläpitoa. Sen avulla voi siirtää tietoja eri järjestelmäkomponenttien välillä ja suorittaa tarvittavat toiminnot resurssien avulla (Gough, Bryant, Auburn 2022).

API-dokumentaatioon ja testaukseen käytetään usein erikoistyökaluja, kuten Swagger tai Postman. Nämä työkalut auttavat kehittäjiä ymmärtämään sovellusliittymien rakennetta, tekemään pyyntöjä ja testaamaan niiden toimivuutta. Kehityksen aikana käytettiin Postman-työkalua.

3.2 Palvelintekniikat

Verkkosovelluksen palvelinpuolella on keskeinen rooli asiakkaiden pyyntöjen käsittelyssä, tietojen hallinnassa ja liiketoimintalogiikan tarjoamisessa sovellukselle. Tässä alaluvussa käsitellään perusteknologiat, joita käytetään verkkosovelluksemme taustalla ilmaisten pysäköintipaikkojen löytämiseen.

Node.js on JavaScript-ajonympäristö, joka perustuu Google Chromen V8-moottoriin. Sen avulla voi käynnistää JavaScriptiä palvelimella, joten Node.js on palvelinpuolen sovellusten ihanteellinen valinta kehittämiseen. Yksi Node.js:n tärkeimmistä eduista on sen asynkroninen ja tapahtumaohjattu arkkitehtuuri, joka tarjoaa korkean suorituskyvyn ja skaalautuvuuden. Node.js:ssä on myös laaja moduulikirjasto, jonka avulla kehittäjät voivat helposti laajentaa sovellustensa toimintoja.

Express on minimalistinen ja joustava verkkokehys Node.js:lle, jonka avulla on helppo luoda tehokkaita ja skaalautuvia verkkosovelluksia. Se tarjoaa käteviä työkaluja reitin käsittelyyn, mallinnukseen, istunnonhallintaan ja muihin web-palvelimen perustoimintoihin. Express tukee myös monia lisämoduuleja ja laajennuksia, jotka laajentavat sen toimintoja ja yksinkertaistavat kehitystä (Pasquali, Faaborg 2017).

Valitsimme Node.js:n ja Expressin kehittämään sovelluksen taustaa, koska ne tarjoavat nopean ja tehokkaan kehityksen. Lisäksi kehittäjäyhteisön laajan tukeman.

Kuten asiakaspuolen teknologioiden kuvauksessa jo mainittiin, REST määrittelee joukon periaatteita ja käytäntöjä hajautettujen järjestelmien rakentamiseen. REST API:n valinta johtuu sen helppokäyttöisyydestä, yleisyydestä ja kyvystä olla tehokkaasti vuorovaikutuksessa sovelluksen asiakasosan kanssa.

Node.js ja Express valittiin niiden korkean suorituskyvyn, helppokäyttöisyyden ja kehittäjäyhteisön laajan tuen vuoksi. Node.js mahdollistaa JavaScriptin toimimisen tehokkaasti palvelimella, jolloin voit käyttää yhtä ohjelmointikieltä

sekä sovelluksesi asiakas- että palvelinpuolella. Tämä yksinkertaistaa koodin johdonmukaisuutta ja parantaa kehitystyön tuottavuutta.

Express tarjoaa puolestaan käteviä työkaluja verkkosovellusten luomiseen, jolloin kehittäjät voivat keskittyä liiketoimintalogiikan toteuttamiseen ja samalla minimoida infrastruktuurikustannukset. Minimalistisen rakenteensa ja joustavuuden ansiosta Express on ihanteellinen RESTful-sovellusliittymien ja verkkosovellusten kehittämiseen, joiden monimutkaisuus vaihtelee.

3.3 Geolocation and kartografiamenetelmät

Paikannus- ja kartoitusteknologeilla on tärkeä rooli verkkosovelluksessa ilmaisten pysäköintipaikkojen löytämisessä, jolloin käyttäjät voivat löytää nopeasti ja kätevästi pysäköintipaikat ja navigoida kartalla. Tässä alakohdassa tarkastellaan valittuja tekniikoita paikannus- ja kartoitustoimintojen toteuttamiseen verkkosovelluksessa.

3.3.1 Kartan API -valinta

Maps API:n valinta verkkosovellukselle on avainasemassa sovelluksen interaktiivisuuden ja käytettävyyden varmistamisessa. Karttasovellusliittymää valittaessa on tärkeää ottaa huomioon helppokäyttöisyyteen, mukauttamiseen joustavuuteen, tietojen saatavuuteen ja kehittäjäyhteisön tukeen. Tässä yhteydessä Leaflet erottuu yhtenä johtajista seuraavien etujen ansiosta:

Helppokäyttöisyys: Leaflet tarjoaa yksinkertaisen ja intuitiivisen käyttöliittymän karttojen käsittelyyn verkkosovelluksissa. Sen API on helppoa oppia jopa aloitteleville kehittäjille sen intuitiivisen rakenteen ja laajan dokumentaation ansiosta.

Räätälöinnin joustavuus: Leafletissä on laajat vaihtoehdot kartan ulkoasun ja toimivuuden mukauttamiseen. Voimme muuttaa helposti tyylejä, lisätä mukautettuja säätimiä, integroida lisätasoja ja laajennuksia.

Laajennettavuus: Leaflet tukee laajennusta laajennusten avulla, jolloin voi lisätä verkkosovellukseesi lisätoimintoja. Tästä tulee Leafletiä joustavana ja skaalautuvana työkaluna erilaisten karttasovellusten kehittämiseen.

Erilaisten tietolähteiden tuki: Leaflet tukee monenlaisia tietolähteitä, sisältäen OpenStreetMap, Mapbox, Google Maps -laattakartat sekä mukautetut laatat ja paikannustiedot. Tämä antaa meille mahdollisuuden valita sopivimman tietolähteen sovelluksen tarpeiden mukaan.

Active Developer Community: Leafletillä on suuri ja aktiivinen kehittäjäyhteisö, joka toimii jatkuvasti parantaakseen ja kehittääkseen kirjastoa. Se tarjoaa tukea ja päivityksiä sekä kokemusten jakamista ja ongelmien ratkaisemista foorumien ja tietovarastojen kautta.

Näiden etujen perusteella Leafletin valitseminen verkkosovellusta varten on looginen päätös. Leaflet tarjoaa helppokäyttöisyyden, joustavan räätälöinnin ja laajat kartoitusominaisuudet, joiden avulla voidaan luoda kätevä ja toimiva sovellus ilmaisten pysäköintipaikkojen löytämiseen.

3.3.2 Leaflet-react kirjasto

Leaflet-react on kirjasto, jonka avulla on helppoa integroida Leaflet React-sovelluksiin tarjoamalla käteviä komponentteja ja työkaluja karttojen käsittelyyn Reactissa. Katsotaanpa Leaflet-reactin tärkeimpiä etuja ja ominaisuuksia:

Kätevät komponentit: Leaflet-react sisältää useita valmiita komponentteja, joiden avulla on helppoa luoda ja hallita interaktiivisia karttoja React-sovelluksessa. Esimerkiksi `<Map>`-komponenttia käytetään näyttämään kartta, `<Marker>`-komponenttia käytetään merkintöjen lisäämiseen ja `<Popup>`- ja `<Tooltip>`-komponentteja käytetään ponnahdusikkunoiden ja työkaluvihjeiden näyttämiseen.

Integrointi React-komponenttien kanssa: Leaflet-react integroituu saumattomasti React-ekosysteemiin, jonka avulla voi käyttää tuttua JSX-syntaksia ja hallita komponenttien tilaa Reactin kautta. Tämä varmistaa helpon käytön ja yhteensopivuuden muiden sovelluskomponenttien kanssa. Tuo Leaflet-kirjasto kuvan 4 projektiin.

```
import L from 'leaflet';
import {MapContainer, Marker, Popup, TileLayer, useMapEvents} from "react-leaflet";
import 'leaflet/dist/leaflet.css';
```

Kuva 4. Leaflet-kirjaston import.

DOM-vuorovaikutuksen tuki: Leaflet-react tarjoaa mahdollisuuden olla vuorovaikutuksessa React-sovelluksen DOM-elementtien kanssa, mikä helpottaa karttojen integrointia muiden käyttöliittymäelementtien kanssa ja niiden käyttäytymisen ohjaamista React-tapahtumien ja -tilojen kautta.

Laajennuksien ja lisäominaisuuksien tuki: Leaflet-reactin avulla voi käyttää kaikkia Leafletin ominaisuuksia, sisältäen laajennukset ja lisäkomponentit. Tämä antaa kehittäjille runsaasti mahdollisuuksia laajentaa kartoitussovellustensa toimintoja.

Aktiivinen yhteisö ja tuki: Leaflet-reactilla on aktiivinen kehittäjäyhteisö, joka tukee ja kehittää kirjastoa. Siinä on laaja dokumentaatio, koodiesimerkkejä ja opetusohjelmia, joiden avulla kehittäjät oppivat nopeasti Leaflet-reactin käytön perusteet.

Käyttämällä Leaflet-reactia verkkosovelluksessa voimme helposti integroida Leafletin Reactiin, jolloin voimme luoda interaktiivisia karttoja ilman suuria ponnistuksia. ja hallita niitä kätevästi Reactin komponenttien ja tilojen kautta. Tästä tulee Leaflet-reactia ihanteellisena vaihtoehdona karttatoimintojen kehittämiseen sovelluksessamme ilmaisten pysäköintipaikkojen löytämiseksi.

3.4 Tietokannan käyttäminen

Tietokannan käyttö on verkkosovelluksien olennainen osa. Erityisesti ne, jotka käsittelevät suuria tietomääriä, kuten pysäköintipaikkojen maantieteellisiä tietoja. Tässä alakohdassa käsitellään tietokannan valintaa verkkosovellusta varten ilmaisten pysäköintipaikkojen löytämiseksi ja perustelut MongoDB:n valinnalle.

Tarkastellaan useita tietokantavaihtoehtoja:

1. Relaatiotietokannat (esimerkiksi PostgreSQL, MySQL):

    - Edut:
        - On levitty laajasti ja on tutkittu hyvin.
        - Säilyttää tapahtumien ja tietojen eheyden.
    - Puutteet:
        - Ne voivat osoittautua tarpeettomiksi tietylle sovellukselle, koska ne vaativat tiukan tietoskeeman, mikä voi olla hankalaa geodatan kanssa työskennellessä.
        - Skaalaaminen voi olla rajoitettua ja vaatii lisäponnistusta.

2. NoSQL-tietokannat (esim. MongoDB, Couchbase):

    - Edut:
        - Joustava tietoskeema, joka on kätevä, kun työskennellään eri muotoisten paikannustietojen kanssa.
        - Tuki paikannusindekseille ja paikkatietokyselyille varmistaa geodatan tehokkaan käsittelyn.
        - Skaalataan helposti vaakasuunnassa, mikä on tärkeää suurten tietomäärien käsittelyssä.
    - Puutteet:
        - Saattaa kuluttaa enemmän resursseja kuin relaatiotietokannat tietyntyyppisten kyselyjen aikana.

3. Graafitietokannat (esim. Neo4j, Amazon Neptune):

- Edut:
  - Ihanteellinen kohteiden välisten suhteiden mallintamiseen, mikä voi olla hyödyllistä analysoitaessa pysäköintipaikkoja koskevia maantieteellisiä tietoja.
- Puutteet:
  - Saattaa olla ylivoimaista tämän projektin tarkoituksiin, koska sovelluksen ei tarvitse mallintaa monimutkaisia graafisia suhteita pysäköintipaikkojen välillä.

MongoDB on dokumenttipohjainen tietokanta, joka sopii erinomaisesti geodatan ja suurten tietomäärien käsittelyyn. Tietokantayhteys projektiin kuvassa 5.

```
const mongoose :   = require('mongoose');

const MONGODB_URI : string  = 'mongodb://localhost:27017/parking-db';
mongoose.connect(MONGODB_URI,  options: {
    useNewUrlParser: true,
    useUnifiedTopology: true,
    useCreateIndex: true,
});

const db : Connection  = mongoose.connection;
```

Kuva 5. Tietokantayhteys projektiin.

MongoDB on ihanteellinen valinta tälle verkkosovellukselle useista syistä:

- Joustava tietoskeema: MongoDB ei vaadi tiukkaa tietoskeemaa, minkä ansiosta tietorakennetta on helppoa mukauttaa tarpeisiimme. Tämä on erityisen tärkeää työskenneltäessä geodatan kanssa, jolla voi olla erilaisia muotoja ja tyyppejä.
- Paikannusindeksien ja paikkatietokyselyjen tuki: MongoDB tarjoaa integroidut indeksointi- ja kyselyominaisuudet maantieteellisille

tiedoille, mikä antaa mahdollisuuden pysäköintipaikan geodatan tehokkaan käsittelyn.

- Skaalautuvuus ja suorituskyky: MongoDB skaalautuu vaakasuunnassa helposti ja tarjoaa korkean suorituskyvyn suuria tietomääriä käsiteltäessä. Näin voi käsitellä käyttäjien pyyntöjä tehokkaasti ja saada nopeasti tietoa pysäköintipaikoista.
- Aktiivinen yhteisö ja tuki: MongoDB:llä on suuri ja aktiivinen kehittäjäyhteisö, joka työskentelee jatkuvasti parantaakseen ja kehittääkseen tietokantaa. Tämä tarjoaa kehittäjille tukea, päivityksiä ja pääsyn resursseihin.

Näitten syitten takia MongoDB:n valitseminen tämän verkkosovelluksen tietokannaksi on looginen päätös. MongoDB tarjoaa joustavuutta, suorituskykyä ja skaalautuvuutta, joita tarvitaan sovelluksesi sijaintiin perustuvien pysäköintitilatietojen tehokkaaseen työskentelyyn (Chodorow 2013).

3.5 Kehitysympäristö

Tehokkaalla kehitysympäristöllä on iso rooli verkkosovelluksen luomisprosessissa. Tässä alakohdassa tarkastellaan kehitysympäristömme pääkomponentteja, sisältäen verkkoselain, IDE/tekstieditori, virtuaaliympäristö ja versionhallinta.

3.5.1 Nettiselain

Verkkoselain on yksi tärkeimmistä työkaluista verkkosovellusten kehittämiseen, koska se tarjoaa mahdollisuuden tarkastella ja korjata luotavaa sisältöä. Kun otetaan huomioon tärkeitä näkökohtia verkkoselaimen valinnassa, voi valita välttämättömän vaihtoehdon.

Markkinoilla on monia verkkoselaimia, joista jokaisella on omat ominaisuudet ja edut. Jotkut niistä ovat:

1. Google Chrome:

    - Edut:

- Laaja tuki: Chromella on suuri markkinaosuus, ja käyttäjät käyttävät sitä laajasti, mikä varmistaa yhteensopivuuden useiden alustojen ja laitteiden välillä.
- Tehokkaat kehittäjätyökalut: Chrome tarjoaa runsaasti kehittäjätyökaluja, kuten elementtitarkastajan, JavaScript-konsolin ja virheenkorjaajan, mikä tekee virheenkorjausprosessista tehokkaamman.
- Nykyaikaisten teknologioiden tuki: Chrome tukee aktiivisesti uusimpia verkkokehitysstandardeja ja esittelee nopeasti uusia ominaisuuksia kuten WebAssembly ja Progressive Web Apps.

● Puutteet:

- Resurssiintensiivisyys: Chrome voi kuluttaa enemmän tietokoneresursseja kuin jotkin muut selaimet, varsinkin kun käytössä on suuri määrä välilehtiä ja monimutkaisia verkkosovelluksia.

2. Mozilla Firefox:

● Edut:

- Open Source: Firefox on avoimen lähdekoodin projekti, joka herättää kiinnostusta kehittäjiltä ja tarjoaa läpinäkyvyyttä kehitykseen ja päivityksiin.
- Yksityisyyden suoja: Firefox on vahva käyttäjien yksityisyyden suojaamisen puolestapuhuja ja tarjoaa työkaluja tietojen seurannan ja hallinnan estämiseen.
- Laajennustuki: Firefoxilla on laaja laajennuskirjasto, joka parantaa selaimen toimintoja ja antaa käyttäjille mahdollisuuden muokata sitä tarpeidensa mukaan.

● Puutteet:

- Pienin markkinaosuus: Firefoxilla on pienempi markkinaosuus kuin Chromella, mikä saattaa heikentää yhteensopivuutta joidenkin verkkosivustojen ja sovellusten kanssa.

Ottaen huomioon verkkosovelluksen vaatimukset ilmaisten pysäköintipaikkojen löytämiseksi Chrome on paras valinta. Tärkein syy sen valitsemiseen on tehokkaat kehittäjätyökalut. Chrome tarjoaa laajan valikoiman kehittäjätyökaluja, kuten elementtitarkastaja ja virheenkorjaaja, mikä helpottaa kehitystä tätä verkkoselainta käytettäessä, koska elementtitarkastajan ja virheenkorjaajan avulla voi löytää virheet heti ja korjata ne. Kehitettäessä on hyödyllinen myös

Chromen ominaisuus, jonka avulla voi tarkistaa verkkosovelluksen soveltuvuuden eri laitteisiin.

3.5.2 IDE/tekstieditori

On olemassa monia IDE:itä (integroitu kehitysympäristö) ja tekstieditoreja, joista jokaisella on omat ominaisuudet ja edut. Tarkastellaan kahta suosittua vaihtoehtoa - WebStorm ja Visual Studio Code:

1. WebStorm:

    - Edut:
        - Erikoistuminen: WebStorm on erikoistunut verkkosovellusten kehittämiseen ja tarjoaa laajan valikoiman työkaluja ja ominaisuuksia prosessin yksinkertaistamiseksi.
        - Tehokkaat ominaisuudet: WebStorm tarjoaa tehokkaita ominaisuuksia, kuten koodin viimeistely, syntaksin tarkistus, JavaScript-virheenkorjaus ja erilaisten kehyksien tuki, muun muassa React.
        - Versionhallintajärjestelmien Integrointi: WebStormissa on versionhallintajärjestelmien sisäänrakennettu tuki, kuten Git, mikä helpottaa lähdekoodin käyttöä.
        - On mahdollista sen muokkaus tarpeiden mukaan.
    - Puutteet:
        - Maksaminen: WebStorm on kaupallinen tuote, ja sen kaikkien ominaisuuksien käyttäminen edellyttää lisenssin ostamista.

2. Visual Studio koodi:

    - Edut:
        - Ilmainen ja avoin lähdekoodi: Visual Studio Code on ilmainen ja kaikkien käyttäjien saatavilla, mikä tekee siitä houkuttelevan monille kehittäjille.

  - Laaja tuki: Visual Studio Codessa on laaja käyttäjäyhteisö ja monia laajennuksia, joiden avulla voi mukauttaa sitä erilaisiin tarpeisiin.
  - Helppokäyttöisyys: Visual Studio Codessa on yksinkertainen ja intuitiivinen käyttöliittymä, joten se on uusien web-kehityskäyttäjien käytettävissä.
- Puutteet:
  - Pienin ominaisuusjoukko: Visual Studio Codessa ei välttämättä ole yhtä laajaa ominaisuusjoukkoa kuin WebStorm, etenkään erikoistuneiden verkkokehitystyökalujen yhteydessä.

Ottaen huomioon verkkosovelluksemme erityispiirteet ilmaisten pysäköintipaikkojen löytämiseen, WebStorm on suositeltava valinta.

WebStorm on integroitu kehitysympäristö (IDE), joka on erikoistunut verkkoteknologioiden käyttöön. Valitsemme WebStormin sen tehokkaiden ominaisuuksien, kuten koodin viimeistely, versionhallinnan integrointi ja erinomaisen JavaScriptin ja Reactin tuen vuoksi.

3.5.3 Virtuaalinen ympäristö

Virtuaaliympäristön avulla voi luoda erillisiä ympäristöjä sovellusten kehittämiseen ja suorittamiseen, mikä varmistaa johdonmukaisuuden ja luotettavuuden useissa ympäristöissä (Sbarski, Kroonenburg 2017).

Vertaamalla virtuaaliympäristöä voidaan paljastaa seuraavaa:

1. Virtuaalikoneet (esimerkiksi VirtualBox):

   - Edut:

- Täydellinen eristys: Virtuaalikoneet tarjoavat täydellisen ympäristön sisältäen käyttöjärjestelmän ja kaikki riippuvuudet, eristämisen, joten ne ovat ihanteellinen valinta sovellusten kehittämiseen useissa ympäristöissä.
- Erilaisten käyttöjärjestelmien tuki: virtuaalikoneiden avulla voi käynnistää sovelluksia eri käyttöjärjestelmissä, mikä on hyödyllistä yhteensopivuustestauksessa.

● Puutteet:

- Resurssikustannukset: virtuaalikoneet vaativat enemmän tietokoneresursseja (muistia, CPU-aikaa) kuin säiliöt, mikä voi johtaa suorituskyvyn heikkenemiseen.

2. Kontit (esim. Docker):

● Edut:

- Kevyys: säiliöt jakavat isäntäjärjestelmän resurssit ja eristävät vain tarvittavat komponentit, mikä tekee niistä resurssitehokkaampia.
- Nopea käyttöönotto: säiliöt voidaan käynnistää ja pysäyttää erittäin nopeasti, mikä yksinkertaistaa kehitys- ja testausprosessia.
- Siirrettävyys: Säiliöitä voidaan helposti siirtää ja ottaa käyttöön missä tahansa ympäristössä, mikä tekee niistä ihanteellisen valinnan sellaisten sovellusten kehittämiseen, joita voidaan ottaa käyttöön useilla alustoilla.

● Puutteet:

- Vähemmän täydellinen eristys: Toisin kuin virtuaalikoneet, säilöt eivät ole täysin eristettyjä, mikä voi aiheuttaa turvallisuusriskin joissakin käyttötapauksissa.

Ottaen huomioon verkkosovelluksemme vaatimukset ilmaisten pysäköintipaikkojen löytämiselle ja virtuaalinen ympäristö on suositeltava vaihtoehto MongoDB:n käyttöönottamiseksi tässä sovelluksessa, joten Docker on paras valinta seuraavista syistä:

1. Kevyys ja tehokas ressursien käyttö: Docker antaa kevyyttä ja tehokasta resurssien käyttöä, mikä on tärkeää verkkosovellusten paikallisessa kehittämisessä ja palvelimien skaalautuvuuden kannalta.

2. Siirrettävyys ja käyttöönoton helppous: Docker-säiliöt voidaan kuljettaa helposti ja ottaa käyttöön missä tahansa isännässä, mikä tekee

sovellusten kehittämisestä ja testaamisesta helppoa eri ympäristöissä ilman tarvetta ympäristön määrittää manuaalisesti.

3. Eristetty ympäristö: Docker tarjoaa eristetyn ympäristön MongoDB:n suorittamiseen, mikä estää riippuvuuksien väliset ristiriidat ja varmistaa tietokannan vakaan toiminnan verkkosovelluksessa.

3.5.4 Versionhallinta

Versionhallinta (VCS) on ohjelmistokehitysprosessin olennainen osa, jonka avulla kehittäjät voivat seurata koodin muutoksia, hallita projektin versioita ja tehdä yhteistyötä sen parissa.

Vertailuversion hallinta:

1. Paikalliset versionhallintajärjestelmät (esimerkiksi Git:n työ on kuvassa 6):

    - Edut:
        - Helppokäyttöisyys: Paikalliset versionhallintajärjestelmät, kuten Git, tarjoavat yksinkertaisen ja intuitiivisen käyttöliittymän koodiversioiden hallintaan ja muutosten seurantaan.
        - Offline-tilan työ: Gitin avulla kehittäjät voivat työskennellä offline-tilassa ja tehdä muutoksia paikallisesti myös offline-tilassa.
    - Puutteet:
        - Rajoitetut yhteistyöominaisuudet: Paikalliset versionhallintajärjestelmät voivat olla vähemmän tehokkaita, kun projektissa tehdään yhteistyötä useiden kehittäjien tai hajautettujen tiimien kanssa.

2. Keskitetyt versionhallintajärjestelmät (esimerkiksi SVN):

    - Edut:
        - Keskitetty tallennus: Keskitetyt versionhallintajärjestelmät tarjoavat yhden, keskitetyn arkiston projektille, mikä helpottaa versionhallintaa ja yhteistyötä.
        - Kehittynyt kulunvalvonta: jotkut keskitetyt versionhallintajärjestelmät tarjoavat edistyneen tietovaraston

pääsynhallinnan, joka on hyödyllinen suurten kehittäjäryhmien järjestämisessä.

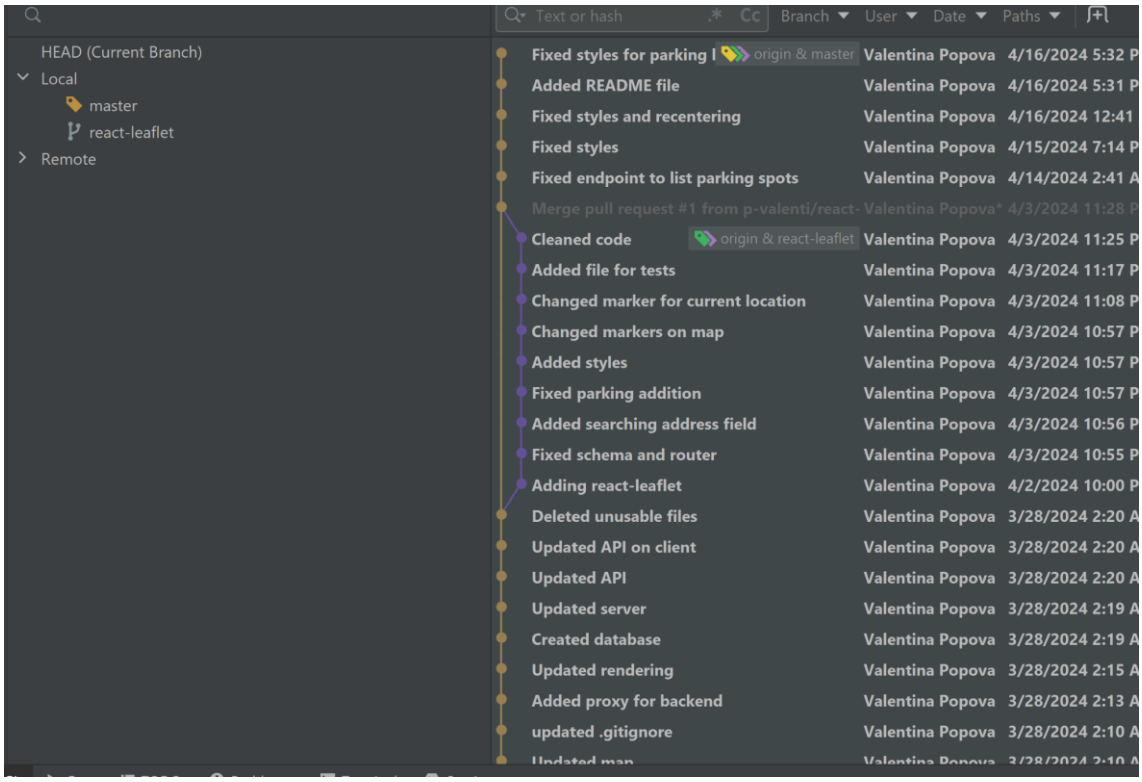
- Puutteet:
  - Yksi vikakohta: keskitetyt versionhallintajärjestelmät voivat muodostua pullonkaulaksi ja luoda yksittäisen vikakohdan, mikä voi johtaa projektiin pääsyongelmiin palvelimen kaatuessa.

Ottaen huomioon verkkosovelluksen vaatimukset ilmaisten parkkipaikkojen löytämisessä sekä tehokkaan yhteistyön ja versionhallinnan tarve, GitHub on paras valinta seuraavista syistä:

1. On levitty laajasti ja suosittu: GitHub on yksi suosituimmista alustoista projektien isännöintiin ja yhteistyöhön Git-versionhallintajärjestelmää käyttäen. Sen laaja leviäminen varmistaa saavutettavuuden ja helpon vuorovaikutuksen muiden kehittäjien ja yhteisön kanssa.

2. Integrointi muihin palveluihin: GitHub tarjoaa monia integraatioita muihin palveluihin ja kehitystyökaluihin, kuten CI/CD-järjestelmiin, ongelmanseurantajärjestelmiin sekä jatkuvaan integrointi- ja käyttöönottotyökaluihin.

3. Tehokkaat versionhallintatyökalut: GitHub tarjoaa laajan joukon versionhallintatyökaluja, kuten haaroitus-, yhdistämis- ja vetopyyntöjä.



Kuva 6. Projektin git.

Näiden tekijöiden perusteella GitHub on paras valinta versionhallintaan ja verkkosovelluksen yhteistyöhön ilmaisten pysäköintipaikkojen löytämiseksi (Skoulikari 2023).

3.6 Sovelluksen testaus

Sovelluksen toimivuuden testaamiseen käytettiin sekä manuaalista että automaattista testausta työkaluilla, kuten Jest ja Mocha, sisältäen pysäköintipaikkojen etsinnän ja tulosten näyttäminen kartalla.

Sovelluksen korkean suorituskyvyn varmistamiseksi työtä testattiin kuormituksen alla arvioiden palvelimen vasteaikaa, sivun latausnopeutta ja muita parametreja (Veenendaal, Black 2019).

Huolellisen teknisen toteutuksen ja testauksen tuloksena valmistui luotettava ja toimiva verkkosovellus ilmaisten pysäköintipaikkojen löytämiseen, joka tarjoaa käyttäjilleen mukavuutta ja käytettävyyttä.

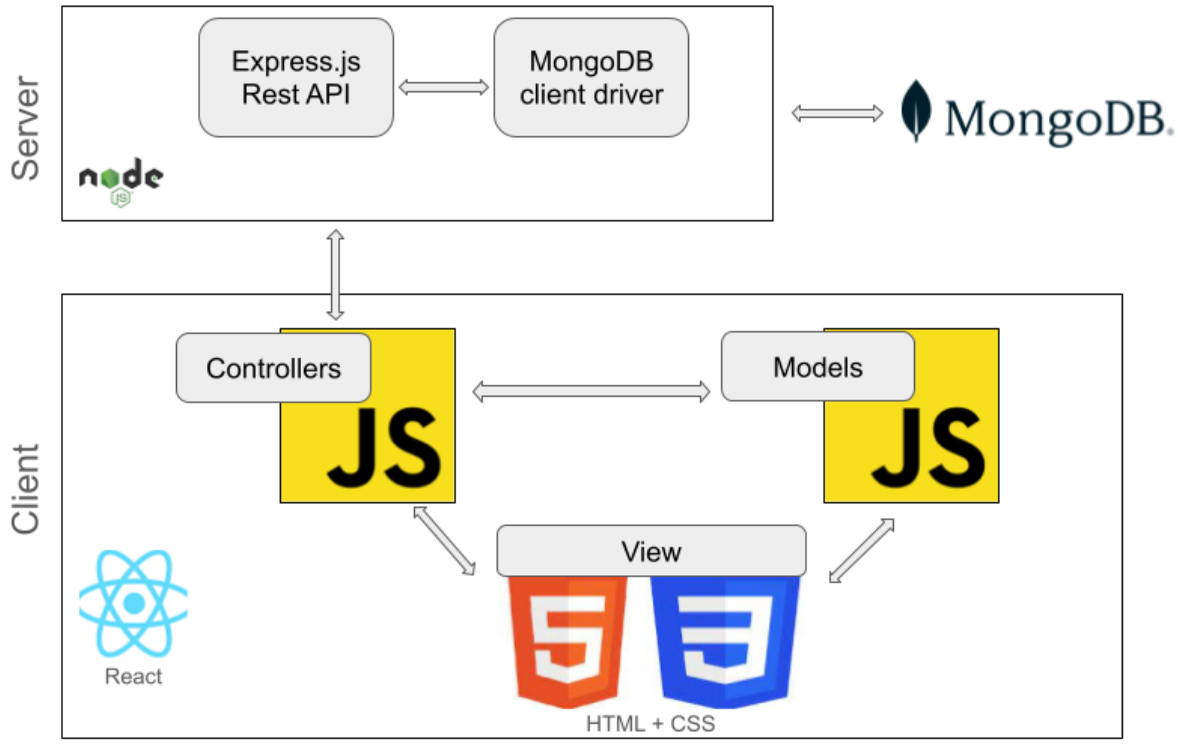
4 Sovelluksen toteuttaminen

Tämä luku on avainvaihe kehitettäessä verkkosovellusta ilmaisten pysäköintipaikkojen löytämiseksi. Tässä käsitellään yksityiskohtaisesti sovelluksen luomisprosessia sen komponenttien kuvauksesta verkkosivuston responsiivisuuteen ja käyttövaiheisiin.

Luvussa käsitellään monenlaisia teknisiä ja suunnitteluun liittyviä näkökohtia, jotka ratkaisevat sovelluksen onnistumisen. Siinä tarkastellaan sekä etu- että taustakomponentteja, niiden vuorovaikutusta sekä sovelluksen käytettävyyteen ja soveltuvuuteen eri laitteisiin ja näyttöihin liittyviä kysymyksiä.

Tämä luku kattaa myös sovelluskehityksen keskeiset vaiheet alusta suunnittelusta lopulliseen toteutukseen ja tarjoaa käyttäjille tehokkaita työkaluja ilmaisten pysäköintipaikkojen löytämiseen ja käyttökokemuksen parantamiseen.

Sovellusarkkitehtuuri on esitetty kaavamaisesti kuvassa 7.



Kuva 7. Sovellusarkkitehtuurikaavio.

4.1 Sovelluskomponentit

Verkkosovelluksessa on useita avainkomponentteja ilmaisten pysäköintipaikkojen etsimiseen, joista jokainen suorittaa oman tehtävänsä ja on vuorovaikutuksessa muiden kanssa tarjotakseen sovelluksen täyden toiminnallisuuden (Sbarski, Kroonenburg 2017). Jokaisen komponentin tarkempi kuvaus:

1. Frontend (asiakasosa):

    - Verkkokäyttöliittymä: Tämä on käyttöliittymä, jonka kautta käyttäjä on vuorovaikutuksessa sovelluksen kanssa. Sisältää säätimet, hakupalkin, tiedonsyöttölomakkeet, tulosnäytön ja kartan pysäköintipaikoilla.
    - Käyttöliittymäkomponentit: Syöttölomakkeet, painikkeet, luettelot ja muut elementit, jotka luovat käyttäjäystävällisen käyttökokemuksen sovellukselle.

2. Backend (palvelinosa):

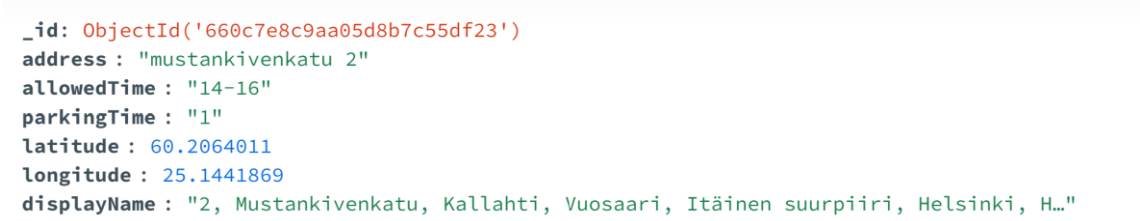
   - Palvelinsovellus: Tämä on ohjelmisto, joka käsittelee asiakkaalta tulevat pyynnöt ja hallitsee kaikkea sovelluslogiikkaa. Palvelinsovellus vastaa pysäköintipaikkojen etsintäpyyntöjen käsittelystä, vuorovaikutuksesta tietokannan kanssa ja tietojen lähettämisestä takaisin asiakkaalle.
   - API (Application Programming Interface): API määrittelee joukon sisäänkäyntipisteitä, joiden kautta asiakassovellus voi kommunikoida palvelimen kanssa. Tässä sovelluksessa se on RESTfull API.

3. Tietokanta (MongoDB):

   - Tietojen tallennus: MongoDB:tä käytetään tallentamaan tietoja käytettävissä olevista pysäköintipaikoista, niiden koordinaateista ja muista asiaan liittyvistä tiedoista. Tietokanta tarjoaa nopean pääsyn tietoihin ja tehokkaan hakukyselyn suorittamisen. Tietokantaskeeman koodi näkyy kuvassa 8. Esimerkki pysäköintipaikan syötöstä näkyy kuvassa 9.

```
const parkingSpotSchema : Schema<Document, Model<...>, undefined, {...}> = new mongoose.Schema( definition: {
    address: {
        type: String,
        required: true
    },
    allowedTime: {
        type: String,
        required: true
    },
    parkingTime: {
        type: String,
        required: true
    },
    latitude: {
        type: Number,
        required: true
    },
    longitude: {
        type: Number,
        required: true
    },
    displayName: {
        type: String,
        required: true
    }
});
```

Kuva 8.Tietokantaskeeman koodi.



Kuva 9.Pysäköintiipaikan tallentaminen tietokantaan.

Jokaisella näistä komponenteista on tärkeä rooli sovelluksen toiminnassa. Käyttöliittymä tarjoaa käyttöliittymän ja käyttökokemuksen, taustaosa hallitsee sovelluslogiikkaa ja pyyntöjen käsittelyä ja tietokanta tallentaa tiedot ja tarjoaa pääsyn niihin. Ne kaikki ovat vuorovaikutuksessa toistensa kanssa tarjotakseen käyttäjälle saumattoman ja tehokkaan kokemuksen, kun he käyttävät sovellusta parkkipaikkojen etsimiseen.

4.2 Komponenttien vuorovaikutuksen kuvaus

Ilmaisten pysäköintipaikkojen etsimiseen tarkoitetun verkkosovelluksen osien välinen vuorovaikutus on avainasemassa sen täydellisen toimivuuden ja tehokkaan toiminnan varmistamisessa. Vuorovaikutusprosessin tarkempi kuvaus:

1. Frontend <-> Backend:

    - Kun käyttäjä pyytää näyttöä tai pysäköintipaikkojen luetteloa verkkokäyttöliittymän kautta, käyttöliittymä välittää pyynnön taustajärjestelmälle API:n kautta.
    - Backend vastaanottaa pyynnön, käsittelee sen ja etsii pysäköintipaikkoja tietokannasta.
    - Kun haku on valmis, backend luo vastauksen, joka sisältää tulokset ja lähettää sen takaisin käyttöliittymään API:n kautta (Gough, Bryant, Auburn 2022).
    - Käyttöliittymä vastaanottaa tulokset taustajärjestelmästä ja näyttää ne verkkokäyttöliittymässä käyttäjälle.

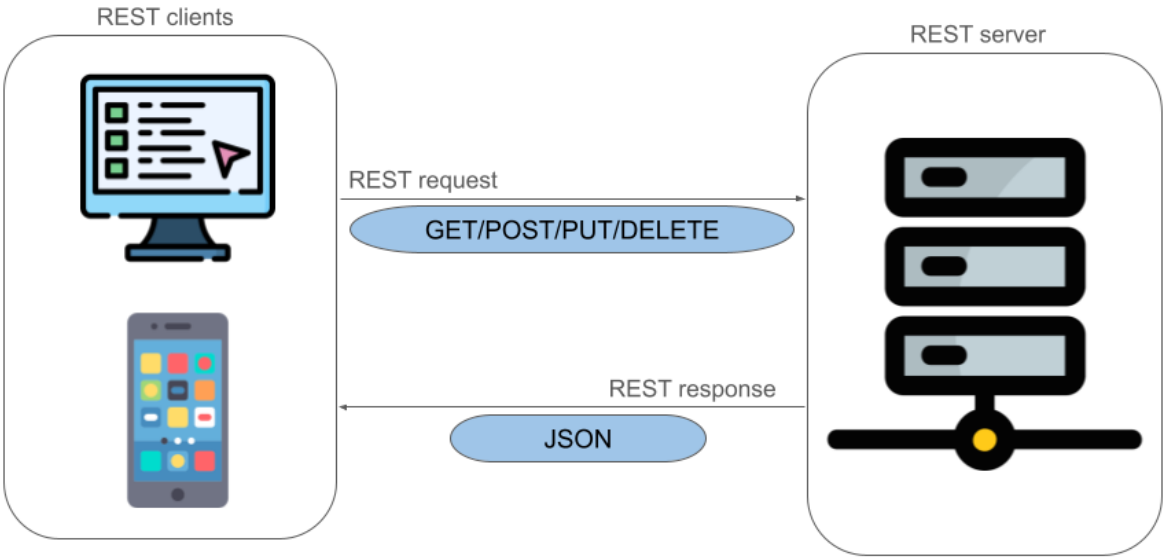
2. Backend <-> Tietokanta:

- Kun taustajärjestelmä vastaanottaa pysäköintipaikkapyynnön, se käyttää MongoDB-tietokantaa (Chodorow 2013).
- Tietokanta suorittaa kyselyn indeksien ja optimoitujen algoritmien avulla löytääkseen nopeasti vastaavat tietueet.
- Kun haku on valmis, tietokanta palauttaa tulokset taustajärjestelmään.
- Taustaohjelma käsittelee tulokset ja luo vastauksen, joka lähetetään takaisin asiakkaalle.

3. Käyttäjän vuorovaikutus:

- Käyttöliittymä tarjoaa käyttöliittymän, jonka avulla käyttäjä voi kirjoittaa hakukyselyitä, valita sijainnin kartalla ja tarkastella hakutuloksia sekä lisätä pysäköintipaikkoja.
- Käyttäjä on vuorovaikutuksessa verkkokäyttöliittymän säätimien kanssa syöttämällä tietoja.
- Saatuaan tulokset käyttäjä voi tarkastella tietoja löydetyistä pysäköintipaikoista ja valita tarpeisiinsa sopivan.

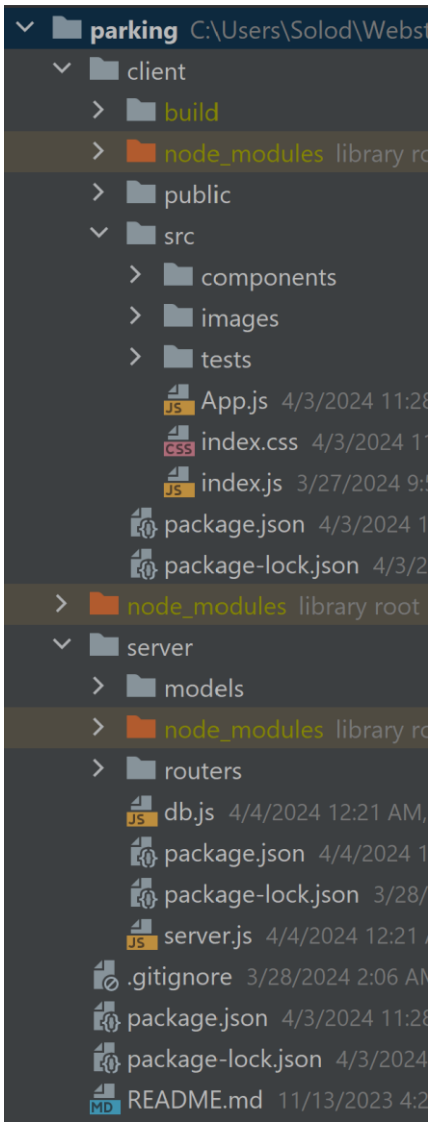
Komponenttien vuorovaikutus API:n kautta näkyy kuvassa 10 esitetyssä REST API -mallissa. Reitti GET koodista kuvassa 11.



Kuva 10. REST API malli.

Kuva 11. Route GET.

Komponenttien tehokas vuorovaikutus takaa sovelluksen nopean ja luotettavan toiminnan, jonka ansiosta käyttäjät voivat löytää nopeasti ilmaisia pysäköintipaikkoja ja parantaa autokokemustaan. Sovellusprojektin rakenne on esitetty kuvassa 12.



Kuva 12. Sovellusprojektin rakenne.

4.3 Verkkosivuston mukautuvuus ja käyttäjäkokemus

Verkkosivuston mukautuvuudella on tärkeä rooli sovelluksen käytettävyyden varmistamisessa eri laitteiden ja näyttöjen käyttäjille. Ilmaisten pysäköintipaikkojen etsimiseen tarkoitetun sovelluksen yhteydessä käyttäjäkokemus on ratkaisevan tärkeää, jotta voidaan varmistaa myönteinen käyttäjäkokemus. Tarkastellaan näitä näkökohtia tarkemmin:

1. Mukautuva suunnittelu:

- Verkkosovellus on kehitetty responsiivisella suunnittelulla, jonka avulla voit automaattisesti säätää sisällön näyttöä eri näytön resoluutioille.
- Tämä varmistaa, että sovellus näkyy optimaalisesti erikokoisissa laitteissa, kuten tietokoneissa, kannettavissa tietokoneissa, tableteissa ja älypuhelimissa.
- Käyttäjät voivat käyttää sovellusta mukavasti riippumatta siitä, millä laitteilla he käyttävät, mikä parantaa sen käytettävyyttä.

2. Intuitiivinen käyttöliittymä:

- Sovelluksen verkkokäyttöliittymä on suunniteltu UX/UI (käyttäjäkokemus/käyttöliittymä) -suunnittelun periaatteet huomioon ottaen, mikä tarjoaa intuitiivisen ja helposti opittavan käyttöliittymän.
- Tämä sisältää selkeän navigoinnin, selkeät ja ymmärrettävät säätimet sekä informatiiviset käyttäjävinkit ja -ohjeet.
- Käyttäjät voivat nopeasti navigoida sovelluksessa ja suorittaa tarvittavat toiminnot ilman tarpeetonta vaivaa, mikä lisää heidän tyytyväisyyttään sovellukseen.

3. Optimoitu suorituskyky:

- Sovellus on optimoitu nopeaan lataukseen ja reagointikykyyn, mikä tarjoaa sujuvan ja mukavan käyttökokemuksen.
- Tämä sisältää resurssien pakkaamisen, tietojen välimuistin, asynkronisen latauksen ja muiden optimointitekniikoiden käytön, jotka lyhentävät latausaikoja ja parantavat sovelluksen suorituskykyä.
- Käyttäjät pääsevät nopeasti käsiksi tarvitsemiinsa tietoihin ja suorittamaan toimintoja odottamatta, mikä parantaa merkittävästi heidän tyytyväisyyttään sovellukseen.

Verkkosivuston mukautuvuus ja käyttökokemus ovat avainasemassa sovelluksen onnistumisessa ja sen hyväksymisessä käyttäjien keskuudessa. Varmistamalla helppokäyttöisyys ja optimaalinen käyttökokemus luovat olosuhteet käyttäjien ongelmien tehokkaalle ratkaisemiselle ja liiketoiminnan tavoitteiden saavuttamiselle (Tidwell, Brewer, Valencia 2020).

4.4 Verkkosovelluksen käytön vaiheet

Ilmaisten pysäköintipaikkojen etsimiseen tarkoitettu sovellus käy läpi useita käyttövaiheita, jotka heijastavat erilaisia skenaarioita käyttäjän vuorovaikutuksesta sovelluksen kanssa. Vaiheiden tarkempi kuvaus:

1. Karttanäyttö:

    - Tämä vaihe alkaa siitä hetkestä, kun käyttäjä avaa sovelluksen. Avattaessa sovelluksen logo tulee näkyviin (kuva 13).
    - Käyttäjä voi skaalata karttaa.
    - Käyttäjä voi tarkastella pysäköintipaikan tietoja kelluvasta merkistä (kuva 14). Tämä merkki laukeaa, kun napsautat pysäköintikuvaketta kartalla.

2. Sijaintihaku:

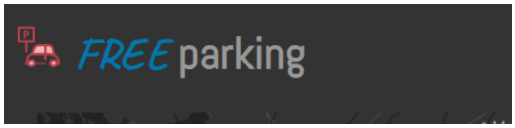
    - Tämä vaihe alkaa siitä hetkestä, kun käyttäjä syöttää vaaditun osoitteen hakupalkkiin (kuva 15).
    - Haun syöttämisen jälkeen sovellus siirtyy kartalla sujuvasti haluttuun paikkaan.
    - Liikkumisen jälkeen keskikartalle ilmestyy merkki halutulle sijainnille kartalla (kuva 16).

3. Pysäköintipaikan lisääminen:

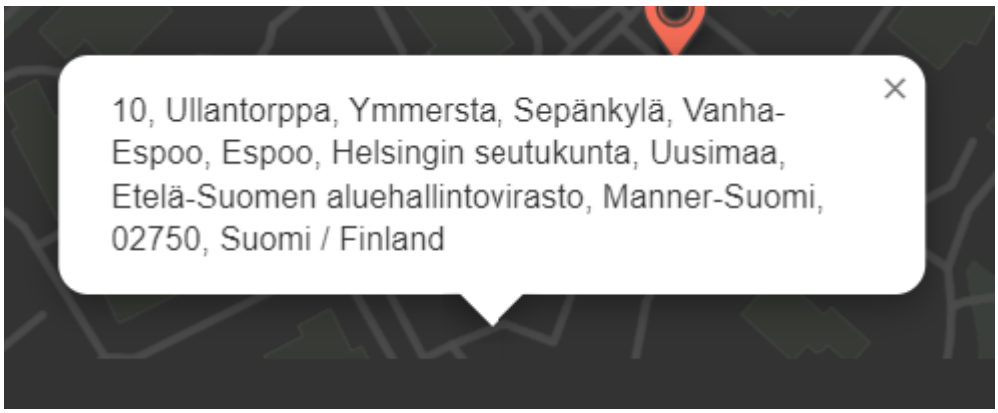
    - Tässä vaiheessa käyttäjän on täytettävä lomake pysäköintipaikan lisäämiseksi (kuva 17).
    - Parkkipaikan lisääminen tapahtuu "Lisää"-painikkeen painamisen jälkeen.

4. Tarkastele pysäköintipaikan tietoja:

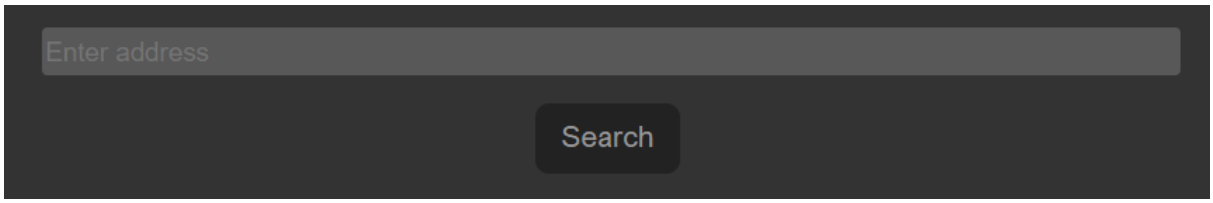
    - Käyttäjä näkee pysäköintiluettelosta kuvauksen parkkipaikasta, mukaan lukien sen sijainnin kartalla, aukioloajat ja muut hyödylliset tiedot.



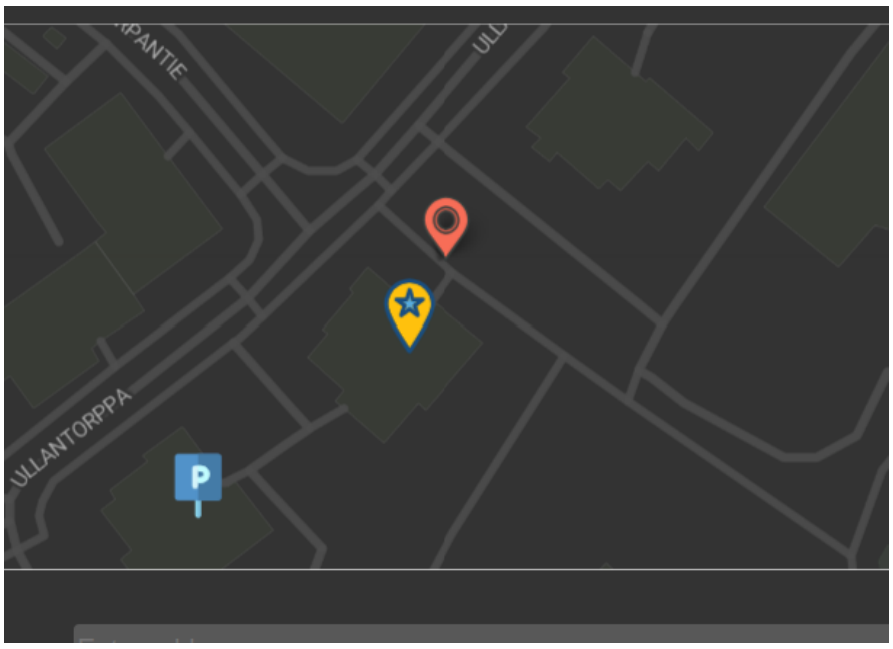
Kuva 13. Sovelluksen logo.



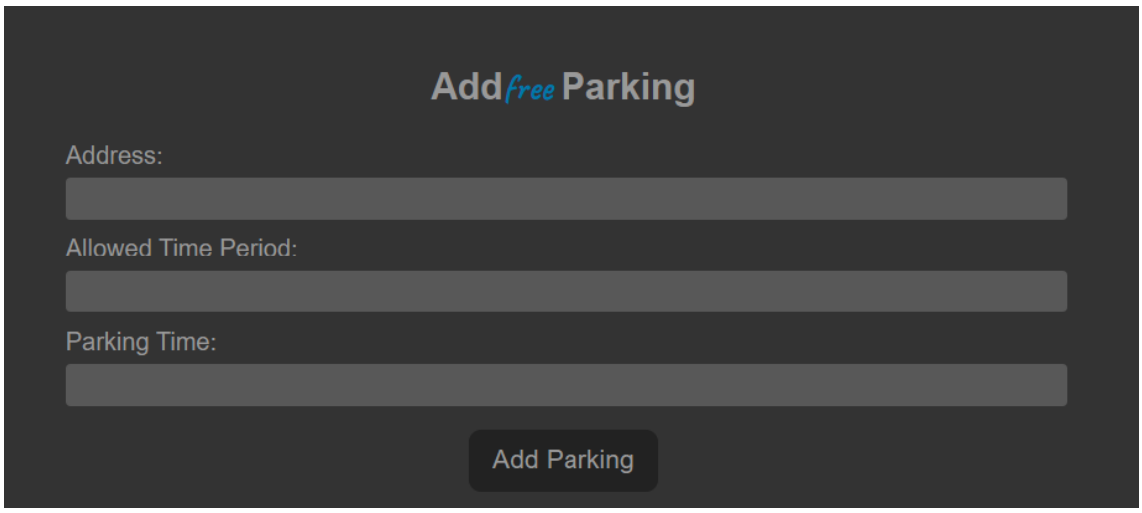
Kuva 14. Pysäköintipaikan tietoja.



Kuva 15. Hakukenttä



Kuva 16. Sijaintimerkki kartalla.



Kuva 17. Lisäminen pysäköintipaikka.

Jokainen näistä vaiheista on tärkeä osa käyttökokemusta ja vuorovaikutusta sovelluksen kanssa. Varmistamalla, että jokainen vaihe suoritetaan tehokkaasti ja saumattomasti, varmistamme, että käyttäjien tarpeet täyttyvät ja sovelluksen tavoitteet saavutetaan.

5 Päätelmät

Tässä työssä kehitettiin ilmaisten pysäköintipaikkojen etsimiseen web-sovellus, joka on työkalu käyttökokemuksen ja auton käytön tehokkuuden parantamiseen. Käydään lyhyesti työn tuloksia ja tarjotaan ideoita projektin jatkokehittämiseksi.

5.1 Yhteenveto tuloksista

Ilmaisten pysäköintipaikkojen etsimiseen tarkoitettua verkkosovellusta kehitettäessä saavutettiin seuraavat tulokset:

1. Monipuolinen sovellus: On kehitetty verkkosovellus, jossa on kaikki tarvittavat toiminnot parkkipaikkojen tehokkaaseen etsimiseen.

2. Mukautuva suunnittelu: Sovellusrajapinta on sovitettu erilaisiin laitteisiin ja näyttöihin, mikä varmistaa helppokäyttöisyyden sekä tietokoneilla että mobiililaitteilla.

3. Palvelinosa: Sovelluksen palvelinosa toteutettiin Node.js:llä ja Expressillä, joka mahdollistaa asiakkaan pyyntöjen käsittelyn ja vuorovaikutuksen MongoDB-tietokannan kanssa.

4. Tietokanta: Luotu MongoDB-tietokanta pysäköintipaikkatietojen ja muiden asiaan liittyvien tietojen tallentamiseen tehokkaan tietojen tallennuksen ja käsittelyn mahdollistamiseksi.

5. Helppokäyttöisyys: Käyttäjät voivat nopeasti ja helposti löytää pysäköintipaikkoja ja tarkastella niitä koskevia lisätietoja.

6. Suorituskyky: Sovelluksen suorituskyky on optimoitu, mikä varmistaa nopean latauksen ja reagoivan käyttöliittymän.

Kaiken kaikkiaan kehitetty sovellus on täydellinen parkkipaikkojen hakua ja varausta helpottava työkalu, jota käyttäjät voivat käyttää laajasti kaupunkiympäristössä.

5.2 Ideoita jatkokehitykseen

Vaikka kehitetty verkkosovellus on jo täydellinen väline ilmaisten parkkipaikkojen etsimiseen, on olemassa useita ideoita, joita voidaan toteuttaa sen toimivuuden parantamiseksi ja sen houkuttelevuuden lisäämiseksi käyttäjille.

Tärkeimmät tehtävät ilmaisten pysäköintipaikkojen etsimiseen tarkoitetun verkkosovelluksen jatkokehityksessä ovat toteutus, uusien toimintojen lisääminen ja käyttökokemuksen parantaminen.

Ensimmäinen prioriteetti on ottaa sovellus käyttöön palvelimella niin, että se on useiden käyttäjien käytettävissä. Tätä varten sinun on valittava sovellukselle luotettava isännöinti.

Seuraava vaihe on lisätä uusia ominaisuuksia ja parantaa käyttökokemusta. Interaktiivisemman ja informatiivisemman kokemuksen tarjoamiseksi käyttäjille tulisi lisätä kommentointiominaisuus. Käyttäjät voivat jakaa arvostelujaan ja suosituksiaan pysäköintipaikoista, mikä auttaa muita käyttäjiä valitsemaan sopivimmat pysäköintipaikat. Palautteen lisääminen sovellukseesi auttaa parantamaan sitä.

Sovelluksen käytettävyyden parantamiseksi on tarkoitus toteuttaa toiminto, jossa näytetään vain käyttäjän nykyistä sijaintia lähimpänä olevat ilmaiset pysäköintipaikat. Näin käyttäjät voivat löytää nopeasti parkkipaikan läheltä nykyistä sijaintiaan.

Lisäksi optimoitu sovelluksen suorituskyky takaa nopean vasteen ja vakaan suorituskyvyn kaikissa olosuhteissa.

Suunnitelmien toteuttaminen parantaa merkittävästi verkkosovelluksen toimivuutta ja käyttökokemusta tehden siitä entistä hyödyllisemmän ja kätevämmän työkalun ilmaisten pysäköintipaikkojen etsimiseen.

6 Johtopäätökset

Verkkosovelluksen kehittäminen ilmaisten parkkipaikkojen etsimiseen on merkittävä askel kohti kaupunkiliikenteen parantamista ja autojen käytön helpottamista. Tämän sovelluksen kehittäminen asetti itselleen useita tärkeitä tehtäviä.

Aluksi tärkein motivaatio sovelluksen kehittämiselle oli tarve ratkaista ongelma, joka liittyy ilmaisten pysäköintipaikkojen löytämiseen kaupunkiympäristössä. Käytettävissä olevan tiedon puute ja pysäköintiresurssien epäoptimaalinen käyttö aiheutti haittaa käyttäjille ja vaikutti kaupunkiliikenteen tehokkuuteen.

Projektin tavoitteena oli luoda työkalu, jonka avulla käyttäjät löytävät nopeasti ja kätevästi ilmaisia parkkipaikkoja omalta alueelta ja tehdä tämän ilman turhaa vaivaa ja aikaa. Tämän tavoitteen saavuttamiseksi asetettiin useita erityistehtäviä, mukaan lukien hakutoimintojen kehittäminen, tulosten visualisointi kartalla, käyttöliittymän mukautuvuuden varmistaminen eri laitteille ja sovellusten suorituskyvyn varmistaminen.

Hanketyön aikana panostettiin merkittävästi tavoitteiden saavuttamiseen ja ongelmien ratkaisemiseen. Tuloksena oli täysin toimiva verkkosovellus, joka on intuitiivinen ja kätevä työkalu pysäköintipaikkojen etsimiseen.

Lähteet

1. Ambler, S. 2010. The elements of UML 2.0 Style. Cambridge University Press.

2. Bowers, M., Synodinos, D., Sumner, V. 2011. Pro HTML5 and CSS3 Design Patterns. APress.

3. Chodorow, K. 2013. MongoDB: The Definitive Guide: Powerful and Scalable Data Storage. O'Reilly Media, Inc.

4. Flanagan, D. 2020. JavaScript: The Definitive Guide. O'Reilly Media, Inc.

5. Gough, J., Bryant, D., Auburn, M. 2022. Mastering API Architecture: Design, Operate, and Evolve API-Based Systems. O'Reilly Media, Inc.

6. Gössling, S. 2020. Why cities need to take road space from cars - and how this could be done. Viitattu 26.4.2024. https://www.tandfonline.com/doi/full/10.1080/13574809.2020.1727318

7. Melnyk, P., Djahel, S., Nait-Abdesselam, F. 2019. Towards a Smart Parking Management System for Smart Cities. Viitattu 26.4.2024. https://www.researchgate.net/publication/335974165_Towards_a_Smart_Parking_Management_System_for_Smart_Cities

8. Pasquali, S., Faaborg, K. 2017. Mastering Node.js : Build robust and scalable real-time server-side web applications efficiently. Packt Publishing.

9. Sbarski, P., Kroonenburg, S. 2017. Serverless Architectures on AWS. Manning Publications.

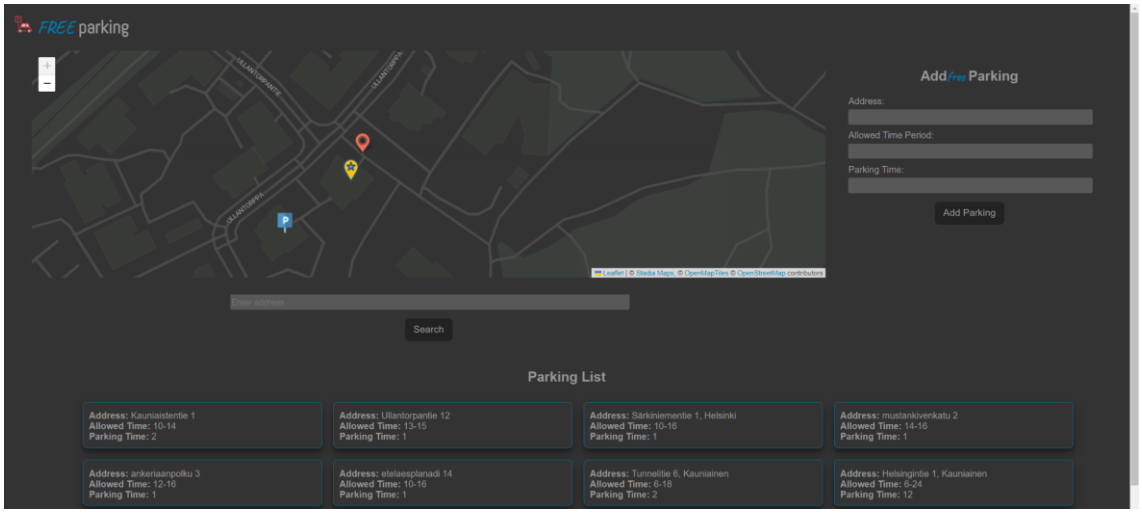
10. Skoulikari, A. 2023. Learning Git. O'Reilly Media, Inc.

11. Tidwell, J., Brewer, C., Valencia, A. 2020. Designing Interfaces. O'Reilly Media, Inc.

12. Veenendaal, E., Black, R. 2019. Foundations of Software Testing ISTQB Certification. Cengage Learning EMEA

Sovelluksen käyttöliittymä

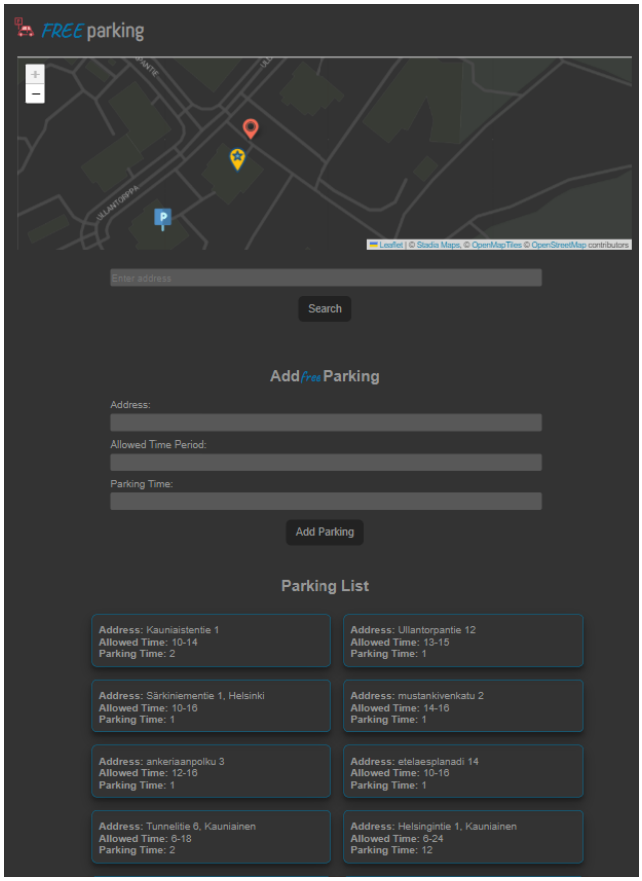
Tämä liite kuvassa 1 näyttää verkkosovelluksesta ilmaisten pysäköintipaikkojen etsimiseen. Sovelluksen käyttöliittymä sisältää perustoiminnot, kuten parkkipaikkojen näyttämisen sekä parkkipaikkojen lisäämisen.



Kuva 1. Sovelluksen käyttöliittymä.

Mobiiliversion sovellusliittymä

Kuvan 2 liite näyttää verkkosovelluksen mobiiliversion ilmaisten pysäköintipaikkojen etsimiseen. Sovelluksen käyttöliittymä sisältää perusominaisuudet, kuten parkkipaikkojen näyttämisen ja parkkipaikkojen lisäämisen.



Kuva 2. Mobiiliversion sovellusliittymä.