jamk

Mobiilisovelluskehitys React Nativella

Esa Viitanen

Opinnäytetyö, AMK
Toukokuu 2024
Ohjelmistotekniikka
Insinööri (AMK), tieto- ja viestintätekniikka

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Kuvailulehti

Viitanen, Esa

Mobiilisovelluskehitys React Nativella

Jyväskylä: Jyväskylän ammattikorkeakoulu. Toukokuu 2024, 55 sivua

Tieto- ja viestintätekniikan tutkinto-ohjelma. Opinnäytetyö AMK.

Julkaisun kieli: suomi

Julkaisulupa avoimessa verkossa: kyllä

Tiivistelmä

Tässä opinnäytetyössä tutkittiin mobiilisovellusten kehittämistä React Native -kehitysalustan avulla keskittyen erityisesti tiekuntien tarpeiden huomioimiseen. Työn tavoitteena oli selvittää, miten React Native soveltuu mobiilisovelluksen luomiseen, joka helpottaa tiekuntien hallinnollisia ja operatiivisia tehtäviä. Tutkimus aloitettiin analysoimalla markkinoita ja olemassa olevia sovelluksia, joiden pohjalta määriteltiin projektin tavoitteet ja suunnitteluvaatimukset.

Sovelluskehitysprosessissa painotettiin käytännönläheistä lähestymistapaa, yhdistäen teoriaa ja käytännön toteutusta. React Native -alustaa käytettiin sovelluksen suunnittelussa, toteutuksessa ja testauksessa. Työn aikana käytiin läpi eri vaiheet sovelluksen suunnittelusta sen toteutukseen ja testaamiseen, minkä jälkeen arvioitiin kehitetyn sovelluksen toimivuutta ja käyttäjäkokemusta.

Tutkimuksen tulokset osoittivat, että React Native soveltuu hyvin tiekuntien tarpeisiin suunnatun mobiilisovelluksen kehittämiseen. Sovellus mahdollisti tehokkaan tiedonvaihdon ja päätöksenteon tiekunnan jäsenten välillä, mikä paransi yhteisön toimintaa ja läpinäkyvyyttä. Lisäksi sovellus tarjosi ratkaisuja yleisiin tiekuntien kohtaamiin ongelmiin, kuten tiedon jakamiseen ja kunnossapitotöiden koordinointiin.

Johtopäätöksissä todettiin, että oikein suunniteltu mobiilisovellus voi merkittävästi tehostaa tiekuntien hallinnollisia toimintoja. React Native -kehitysalustan käyttö mahdollisti nopean ja kustannustehokkaan sovelluskehityksen, joka vastasi tiekuntien erityistarpeisiin. Työn tulokset tarjoavat hyödyllistä tietoa ja käytännön ohjeita vastaavanlaisten sovellusten kehittämiseen tulevaisuudessa.

Avainsanat (asiasanat)

React Native, mobiilisovelluskehitys, tiekunnat, Android

Muut tiedot (salassa pidettävät liitteet)

jamk

Description

Viitanen, Esa

Mobile Application Development with React Native

Jyväskylä: JAMK University of Applied Sciences, May 2024, 55 pages

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

Permission for open access publication: Yes

Language of publication: Finnish

Abstract

This thesis explores mobile application development using the React Native framework, focusing specifically on the needs of road maintenance associations. The aim of the study was to determine how well React Native is suited for creating a mobile application that facilitates the administrative and operational tasks of these associations. The research began with an analysis of the market and existing applications, which helped define the project's objectives and design requirements.

The application development process emphasized a practical approach, combining theory and practice. React Native was used for designing, implementing, and testing the application. Throughout the project, the various stages of application design, implementation, and testing were covered, followed by an evaluation of the developed application's functionality and user experience.

The results of the study showed that React Native is well suited for developing a mobile application tailored to the needs of road maintenance associations. The application enabled efficient information exchange and decision-making among the association members, which improved community operations and transparency. Additionally, the application provided solutions to common problems faced by road maintenance associations, such as information sharing and maintenance task coordination.

The conclusions indicated that a well-designed mobile application can significantly enhance the administrative functions of road maintenance associations. The use of the React Native framework allowed for rapid and cost-effective application development that met the specific needs of these associations. The results of this work offer valuable insights and practical guidelines for developing similar applications in the future.

Keywords/tags (subjects)

React Native, mobile application development, road maintenance associations, Android

Miscellaneous (Confidential information)

Sisältö

Kuviot

LYHENTEET JA TERMIT

Android
Android on Googlen kehittämä avoimen lähdekoodin käyttöjärjestelmä mobiililaitteille, kuten älypuhelimille ja tableteille. Se perustuu Linux-käyttöjärjestelmään ja tukee monia sovelluskehitysratkaisuja, kuten Java ja Kotlin. Androidin laajan käyttäjäkunnan ja monipuolisten ominaisuuksien ansiosta se on yksi suosituimmista mobiilikäyttöjärjestelmistä maailmassa.

API
API (Application Programming Interface) on joukko määriteltyjä protokollia ja työkaluja, joiden avulla eri ohjelmistot voivat kommunikoida keskenään. API:t mahdollistavat erilaisten sovellusten integroinnin ja toiminnallisuuksien hyödyntämisen toistensa välillä.

CSS
CSS (Cascading Style Sheets) on tyylisivu, jota käytetään verkkosivujen ulkoasun muotoiluun ja layoutin määrittämiseen. CSS erottaa sisällön (HTML) ja ulkoasun toisistaan, mikä mahdollistaa verkkosivujen visuaalisen ilmeen hallinnan keskitetysti. CSS:n avulla voidaan määrittää esimerkiksi fontteja, värejä, marginaaleja ja sijoittelua.

EXPO
Expo on kehitysalusta, joka yksinkertaistaa React Native -sovellusten kehittämistä. Se tarjoaa työkalut ja palvelut, jotka helpottavat mobiilisovellusten. Expo sisältää valmiita komponentteja, kirjastoja ja kehitysympäristöjä, jotka nopeuttavat sovelluskehitystä.

HTML
HTML (HyperText Markup Language) on verkkosivujen perusrakenteen määrittelykieli. HTML:n avulla voidaan luoda ja jäsentää verkkosivun sisältöä, kuten tekstiä, kuvia, linkkejä ja lomakkeita. HTML:n elementit kertovat selaimelle, miten sisältö tulee esittää käyttäjälle. HTML on verkkosivujen perusteknologia yhdessä CSS:n ja JavaScriptin kanssa.

JavaScript
JavaScript on ohjelmointikieli, joka on suunniteltu lisäämään vuorovaikutteisuutta ja dynaamisuutta verkkosivuille. Se mahdollistaa muun muassa käyttäjän toimintoihin reagoimisen, lomakkeiden validoinnin, animaatiot ja paljon muuta. JavaScriptiä käytetään laajalti verkkokehityksessä ja se on yksi tärkeimmistä kielistä selainpohjaisissa sovelluksissa.

JSX
JSX (JavaScript XML) on laajennus JavaScriptille, jota käytetään erityisesti React-sovelluksissa. JSX mahdollistaa HTML:n kaltaisten rakenteiden kirjoittamisen suoraan JavaScript-koodiin, mikä tekee komponenttien määrittelystä ja hallinnasta helpompaa ja intuitiivisempaa.

iOS
iOS on Applen kehittämä mobiilikäyttöjärjestelmä, jota käytetään iPhoneissa, iPadeissa ja iPod Touch -laitteissa. Sovelluksia iOS:lle kehitetään yleensä Swift- tai Objective-C -ohjelmointikielillä käyttämällä Xcode-kehitysympäristöä.

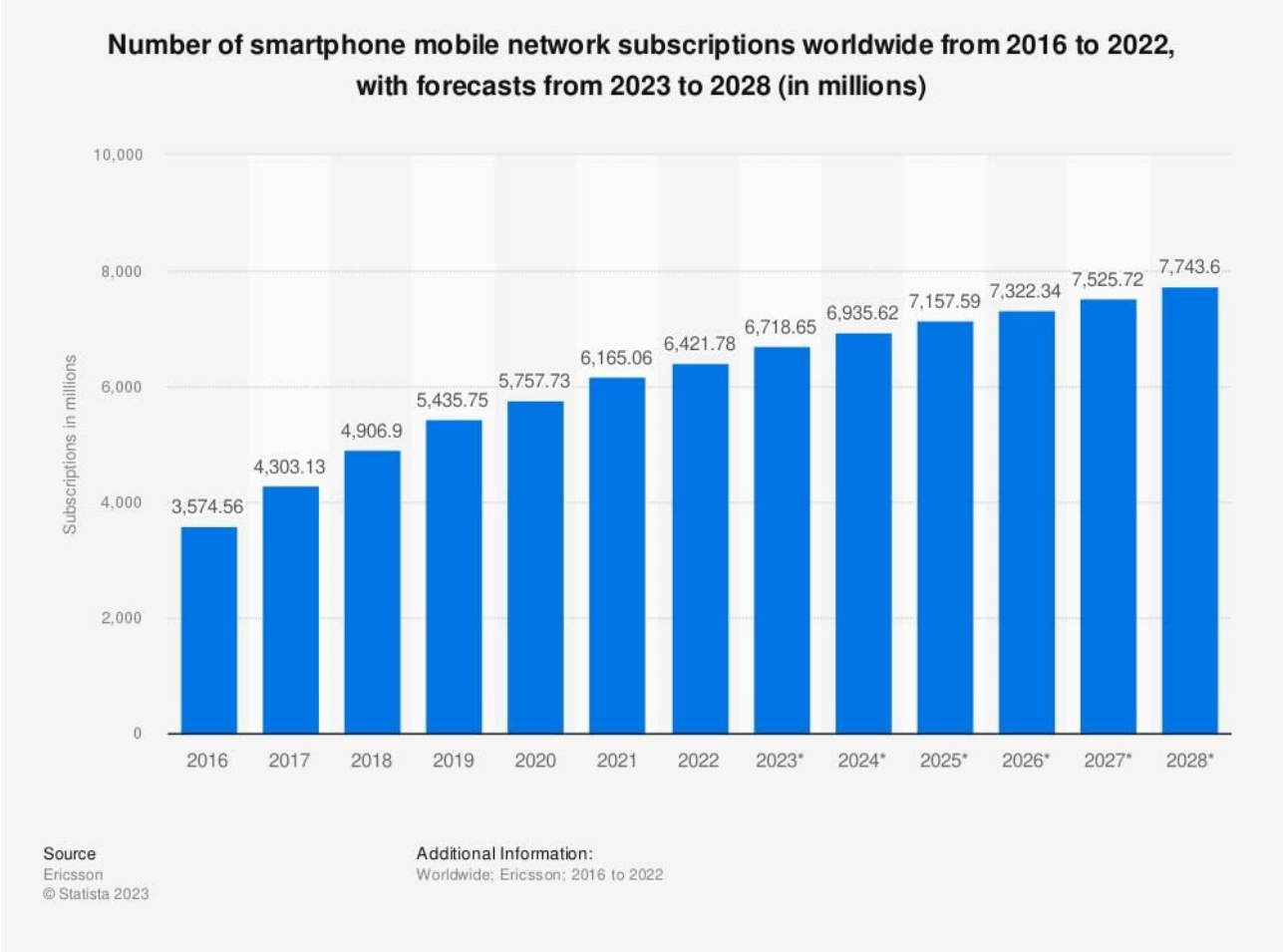
1 Johdanto

1.1 Työn lähtökohdat

Mobiilisovelluskehitys on nopeasti kehittyvä ala, joka vastaa digitalisoituvan maailman tarpeisiin tarjoamalla älypuhelimien ja tablettien käyttäjille monipuolisia sovelluksia. Viime vuosina mobiililaitteiden käyttö on kasvanut räjähdysmäisesti, mikä on luonut valtavan kysynnän innovatiivisille ja käyttäjäystävällisille mobiilisovelluksille.

Yli 78 % ihmisistä ympäri maailmaa omistaa älypuhelimia, joten näiden laitteiden pääasiallinen käyttö liittyy mobiilisovelluksiin. Nämä sovellukset kattavat erilaisia luokkia, mukaan lukien pelit, musiikki, kalenterit, viestintä ja paljon muuta. Tunnetut sovellukset, kuten WhatsApp, Spotify, Facebook ja Instagram, ovat esimerkki mobiilisovellusten laajasta suosiosta. (Blair N.d.)

Kuten nähdään seuraavasta olevasta kuviosta (ks. kuvio 1), älypuhelimien määrä on ollut jatkuvassa kasvussa vuodesta 2016 lähtien ja kasvu on ennustettu jatkuvan myös tulevina vuosina. Tämä kasvu heijastaa älypuhelimien lisääntyvää tarvetta päivittäisessä elämässä sekä mobiiliteknologian saavutettavuuden paranemista globaalisti. Kehittyvien markkinoiden kasvu, datan kulutuksen lisääntyminen ja uusien mobiiliteknologioiden käyttöönotto ovat tekijöitä, jotka ajavat tätä kasvua ylöspäin. Tämä trendi tarjoaa merkittäviä mahdollisuuksia mobiilisovellusten tekijöille innovoida ja laajentaa palveluitaan vastaamaan kasvavaa kysyntää.



Kuvio 1. Älypuhelinten määrä 2016 – 2022 ja ennustettu kehitys vuosille 2023 – 2028. (How Many Smartphone Users Are There?)

Mobiilisovelluskehitys tarkoittaa ohjelmistojen luomista ja suunnittelua älypuhelimille, tableteille ja vastaaville laitteille. Vaikka jakaa yhtäläisyyksiä muiden ohjelmistokehitysmuotojen, kuten verkkosovellusten, kanssa, tärkein ero on mobiilisovellusten ainutlaatuinen kyky hyödyntää laitteen alkuperäisiä ominaisuuksia. (Blair N.d.)

React Native on Facebookin kehittämä avoimen lähdekoodin mobiilisovelluskehys, mahdollistaa kehittäjien rakentaa tehokkaita sovelluksia natiivin suorituskyvyn kanssa käyttäen JavaScriptia, mikä tekee siitä suositun valinnan sovelluskehittäjien keskuudessa. Tämän työn lähtökohtana oli tutkia mobiilisovelluskehityksen prosesseja ja parhaita käytäntöjä React Nativella, erityisesti keskittyen sovellusten kehittämiseen, jotka palvelevat yhteisöllisiä ja hallinnollisia tarpeita. Opinnäytetyössä keskityttiin mobiilisovelluskehitykseen React Nativella, jonka avulla luodaan käytännön puhelinsovellus tiekunnille.

Suomessa on noin 50 000 yksityistietä, jotka ovat tiekuntien hallinnoimia. Lisäksi on noin 45 000 kiinteistöjen yhteistä aluetta. Näistä noin parikymmentätuhatta on järjestänyt hallintonsa ja niille on vahvistettu säännöt. Monet tahot, kuten Maanmittauslaitos, erilaiset viranomaiset sekä yksityiset kansalaiset, tarvitsevat tietoja siitä, keneen ottaa yhteyttä yksityistietä ja yhteistä aluetta koskevissa asioissa. (Tiekuntien ja yhteisten alueiden yhteystiedot voi ilmoittaa verkon kautta 2023.)

Tiekunnat vastaavat tiealueidensa ylläpidosta ja kehittämisestä. Tiekunnan jäsenet koostuvat alueen kiinteistönomistajista. Tiekunnan tehtävänä on huolehtia, että tie on turvallinen ja käyttökelpoinen ympäri vuoden. Tämä sisältää lumityöt, kunnostukset, ojien puhdistukset ja muut tarvittavat toimenpiteet.

Mobiilisovelluksen käyttöönotto voi merkittävästi parantaa tiekunnan toimintaa usealla tavalla. Useasti tiekunnan jäsenet eivät asu samalla paikkakunnalla, joten sovellus mahdollistaa reaaliaikaisen tiedonvaihdon jäsenten välillä. Sovelluksen kautta voi jakaa tietoa tien kunnosta, suunnitelluista kunnostustöistä sekä muista ajankohtaisista asioista. Lisäksi se voi helpottaa päätöksentekoa äänestyksen tai mielipidekyselyjen avulla, mikä nopeuttaa yhteisten päätösten syntymistä.

Mobiilisovelluksen käyttöönotto voi parantaa tiekunnan näkyvyyttä ja avoimuutta. Jäsenet voivat seurata toimintaa reaaliajassa sekä jakaa kokemuksia, neuvoja ja vinkkejä, mikä lisää luottamusta ja tekee tiekunnan toiminnasta läpinäkyvämpää.

1.2 Tavoite

Opinnäytetyön tavoitteena oli tutkia mobiilisovelluskehityksen periaatteita React Nativella sekä selvittää, miten tämä kehitysalusta soveltuu mobiilisovelluksen luomiseen tiekuntien tarpeeseen. Työssä käydään läpi suunnitteluvaihe, toteutus, testaus sekä sovelluksen käyttöönotto.

Lisäksi yhtenä tavoitteena oli tunnistaa ja dokumentoida parhaat käytännöt, joita voidaan soveltaa React Native -sovelluksen kehityksessä, ottaen huomioon suorituskyvyn optimoinnin, käyttöliittymän suunnittelun, tietoturvan ja käyttäjien vuorovaikutuksen.

Tämän työn kautta haluttiin myös lisätä ymmärrystä siitä, miten teknologia voi tukea ja tehostaa yhteisöjen ja organisaatioiden, kuten tiekuntien, hallinnollisia toimintoja. Tutkimme, miten mobiilisovellus voi tarjota ratkaisuja yleisiin ongelmiin, kuten tiedon jakamiseen, päätöksenteon tukemiseen ja yhteistyön parantamiseen jäsenten välillä. Tavoitteena oli osoittaa, että oikein suunniteltu ja toteutettu mobiilisovellus voi olla merkittävä apuväline yhteisöllisen hallinnon ja toiminnan tehostamisessa.

Tehokas ja toimiva tiekunnan hoito varmistaa, että kaikki tarpeelliset dokumentit ovat valmiina tiekunnan vuosikokouksiin, tilinpäätöksiin sekä vuosittain haettavien tukien hakemiseen. Mobiilisovelluksen on tarkoitus helpottaa tätä asiaa. Tämän tapainen lähestymistapa luo pohjan tehokkaalle päätöksenteolle ja varmistaa tiekunnan toiminnan jatkuvuuden.

1.3 Tutkimusasetelma

Opinnäytetyön tutkimusasetelma perustuu seuraaviin tutkimuskysymyksiin:

1. Minkälaisia asioita tulisi ottaa huomioon mobiilisovelluksen suunnittelussa?
2. Miten React Native -ohjelmointikehystä voidaan hyödyntää tiekuntien tarpeeseen suunnatun mobiilisovelluksen kehittämisessä?
3. Millaisia teknisiä ja suunnittelullisia haasteita kohdataan mobiilisovelluksen kehittämisessä React Nativella, ja miten näihin haasteisiin vastataan?

Tutkimusasetelma pyrki tarjoamaan kattavan ymmärryksen React Native -kehitysalustan vahvuuksista ja haasteista mobiilisovelluskehityksessä, keskittyen erityisesti käytännön sovelluksen kehittämiseen, joka vastaa tiekuntien erityistarpeisiin.

2 Mobiilisovelluksen kehittämisen askeleet

2.1 Idean kehittäminen

Ainutlaatuisen sovellusidean kehittäminen on usein haastavin vaihe. Tarve voi nousta henkilökohtaisista haasteista, havainnoista ympäristössä tai parantaa olemassa olevaa sovellusta. Myös sijoittajien rahoittamien projektien tai startup yrityksien tutkiminen voi herättää uusia ideoita. (The Basics of App Development: Step-by-Step Guide. 2024.)

Mobiilisovelluksen kehittäminen aloitetaan tarkastelemalla sen ensimmäistä ja perustavaa vaihetta: idean kehittämistä. Kuten yllä olevassa kappaleessa korostuu, hyvän sovellusidean syntyminen on koko kehitysprosessin kulmakivi. Idea voi syntyä monenlaisista lähteistä, kuten henkilökohtaisista kokemuksista tai havaitusta tarpeesta markkinoilla. Tämän takia on suositeltavaa kirjata ylös kaikki ideat, oli ne sitten suuria tai pieniä. On tärkeää muistaa, että menestyksekäs mobiilisovellus ei pelkästään toteuta teknistä toimintoa, vaan se ratkaisee käyttäjänsä ongelman tai täyttää tietyn tarpeen. Tästä syystä idean kehittämisen alkuvaiheeseen kuuluu olennaisesti myös perusteellinen taustatutkimus, joka auttaa ymmärtämään potentiaalisen sovelluksen kysyntää ja kilpailukenttää.

2.2 Tutki markkinoita

Markkinoiden tutkiminen on välttämätöntä, jotta ymmärtää keitä on kilpailijoina sekä kohderyhmänä. Tämä vaihe auttaa määrittämään idean toteutettavuutta, arvioimaan olemassa olevia ratkaisuja ongelmaan sekä arvioimaan sovelluksen mahdollista kysyntää. Markkinatutkimuksesta voi saada tietoa tiettyihin kehitysalueisiin. (The Basics of App Development: Step-by-Step Guide. 2024.)

Edellisessä osiossa tarkastelimme sovellusidean kehittämisen merkitystä ja prosessia, joka on ensimmäinen askel mobiilisovelluksen luomisessa. Nyt siirrymme käsittelemään toista keskeistä vaihetta, markkinatutkimusta. Tässä osiossa syvennytään syvällisemmin siihen, kuinka markkinatutkimus toimii perustana kilpailuedun saavuttamiselle, tarjoamalla katsauksen siitä, miten sovelluksen

kehittäjät voivat hyödyntää markkinatutkimusta ymmärtääkseen paremmin nykytilanteen, tunnistamaan potentiaaliset kilpailijat ja löytämään markkinoiden aukkoja, jotka tarjoavat mahdollisuuksia uusille sovelluksille.

Alkuvaiheen suunnittelussa kannattaa keskittyä erityisesti siihen, miten määritellään sovelluksen potentiaalinen kohderyhmä, olipa kyseessä sitten tietyn ikäryhmän käyttäjät, erityiset harrastukset tai elämäntilanteet. Tämän määrittelyn jälkeen korostetaan muiden, markkinoilla jo olevien samankaltaisten sovellusten tutkimisen tärkeyttä. Tavoitteenamme on antaa sovelluksen kehittäjille selkeä kuva siitä, miten tämä vaihe auttaa tunnistamaan, mitä ominaisuuksia ja toimintoja kilpailevat sovellukset tarjoavat, ja kuinka uusi sovellus voi erottautua ja täyttää olemassa olevat markkinoiden aukot.

2.3 Suunnitteluvaihe

Suunnittelu on erittäin tärkeä vaihe intuitiivisen ja käyttäjäystävällisen kokemuksen luomiseksi. Kuvittele itsesi käyttäjäksi ja visualisoi mielessäsi, kuinka käyttäisit sovellusta. Suunnittelussa kannattaa myös asettaa etusijalle käytettävyys, saavutettavuus ja toimivuus. (The Basics of App Development: Step-by-Step Guide. 2024.)

Suunnittelussa kannattaa ottaa huomioon värimaailma, kuvat, grafiikka ja muodot. Tämä vaihe sisältää toimivan prototyypin valmistamisen ilman koodin kirjoittamista. Sovellukseen kannattaa sisällyttää tässä vaiheessa jo toiminnallisuuksia kuten sisäänkirjautuminen ja navigaatio. (The Basics of App Development: Step-by-Step Guide. 2024.)

Tässä osiossa tarkastelemme suunnitteluvaiheen monitahoista luonnetta mobiilisovelluksen kehittämisprosessissa. Aluksi korostuu visuaalisen suunnittelun, kuten värimaailman, kuvien, grafiikan ja muotojen, sekä toiminnallisten prototyyppien kehittämisen ilman koodin kirjoittamista, merkitystä. Tämän jälkeinen vaihe on, miten suunnitteluvaiheessa määritellään sovelluksen toiminnot ja ulkoasu vastaamaan käyttäjien tarpeita ja odotuksia, tuoden esiin käyttäjäkeskeisen suunnittelun merkityksen.

Tässä vaiheessa on tärkeää tietää, kuinka käyttäjien tarpeiden ja odotusten syvällinen ymmärtäminen on keskeistä, korostaen käyttäjille suunnattujen kyselyiden ja haastattelujen arvoa. Lisäksi pitää painottaa prototyyppien testauksen ja käyttäjäpalautteen keräämisen tärkeyttä, jotka ovat olennaisia osia suunnitteluprosessissa. Tämä ei ainoastaan helpota tehokkaiden ja käyttäjäystävällisten sovellusten luomista, vaan myös varmistaa, että tietoturva ja käyttäjätietojen suojaus huomioidaan jo varhaisessa vaiheessa.

2.4 Aloita kehittäminen

Kehitysvaiheessa suunnitelma muutetaan konkreettiseksi sovellukseksi. Tässä vaiheessa päätetään mitä teknologioita sovellus tukee. Alustat, joissa ei tarvitse koodata itse niin voi tehdä ohjelmasta prototyypin nopeasti. (The Basics of App Development: Step-by-Step Guide. 2024.)

Kuten aiemmin suunnitteluvaiheessa käsiteltiin, prototyypin luominen muodostaa sillan ideoiden visuaalisesta konseptoinnista niiden tekniseen toteutukseen kehitysvaiheessa. Prototyyppien avulla voidaan varhaisessa vaiheessa testata käyttöliittymän suunnittelua ja käyttäjäkokemusta, mikä on välttämätöntä ennen syvällisempää kehitystyötä. Käyttämällä työkaluja kuten React Native, joka tarjoaa Hot Reload -ominaisuuden, kehittäjät voivat tehdä muutoksia reaaliajassa ja nähdä vaikutukset välittömästi, mikä tehostaa kehitysprosessia ja tekee siitä interaktiivisempaa.

Edellä mainitun suunnitteluprosessin pohjalta kehitysvaiheessa on olennaista ottaa huomioon myös käyttäjien palaute. Testaamalla prototyyppejä loppukäyttäjillä mahdollisimman paljon saadaan arvokasta tietoa, joka auttaa räätälöimään sovelluksen paremmin käyttäjien odotusten ja tarpeiden mukaiseksi. Kuten suunnittelussa korostettiin käyttäjäkeskeisyyttä, kehitysvaiheen aikana tämä aspekti konkretisoituu käyttäjäpalautteen aktiivisessa hyödyntämisessä.

Lisäksi, kuten suunnitteluvaiheessa keskityttiin käyttäjäkokemuksen intuitiivisuuteen, kehitysvaiheessa tulee kiinnittää huomiota myös sovelluksen skaalautuvuuteen. Kun sovelluksen käyttäjämäärä kasvaa, on kriittistä varmistaa, että sovellus pystyy käsittelemään kasvavaa liikennettä ilman, että sen suorituskyky heikkenee. Tämä varmistaa, että sovellus toimii luotettavasti ja tehokkaasti myös suuremmilla käyttäjämäärillä.

Yhteenvetona, tämä osio kehitysvaiheesta jatkaa luonnollisena jatkumona aiemmin kuvatulle suunnitteluvaiheelle, syventäen ymmärrystämme siitä, kuinka prototyyppien testaus ja käyttäjäpalaute integroituvat osaksi laajempaa kehitysprosessia. Tämän vaiheen huomioiminen on olennaista, jotta voimme luoda käyttäjäystävällisiä ja teknisesti toimivia sovelluksia.

2.5 Sovelluksen testaaminen

Testaus on ratkaisevan tärkeää, jotta sovellus toimii niin kuin pitäisi. Se kattaa useita eri kohtia, kuten käytettävyys, toiminnallisuus, suorituskyky, yhteensopivuus ja tietoturvallisuus. Perusteellinen testaus on välttämätöntä kehityskohteiden havaitsemiseksi varhaisessa vaiheessa. (The Basics of App Development: Step-by-Step Guide. 2024.)

Jatkaen edellä käsiteltyä kehitysvaihetta, tässä osiossa syvennymme testauksen kriittiseen rooliin sovelluskehityksessä. Testauksen tarkoituksena on varmistaa ohjelmiston laatu ja luotettavuus ennen sen julkaisua. Seuraavaksi käymme läpi viisi keskeistä testauksen osa-aluetta: käytettävyys, toiminnallisuus, suorituskyky, yhteensopivuus ja tietoturva, jotka kaikki ovat olennaisia onnistuneen sovelluksen kannalta.

Ensimmäisenä keskitymme käytettävyystestaukseen, joka on suunniteltu takaamaan sovelluksen intuitiivisuus ja helppokäyttöisyys. Tämä osa-alue kytkeytyy suoraan aiemmin mainittuun käyttäjäkeskeiseen suunnitteluun, jonka tavoitteena on vastata käyttäjien tarpeisiin ja odotuksiin. Seuraavaksi tarkastelemme toiminnallisuustestauksen merkitystä, joka varmistaa, että sovelluksen toiminnot vastaavat suunniteltuja vaatimuksia.

Kolmanneksi, suorituskykytestaus arvioi sovelluksen tehoa erityisesti haastavissa olosuhteissa, kuten suuren käyttäjämäärän aikana tai rajoitetuissa verkkoyhteyksissä, mikä on välttämätöntä sovelluksen yleisen tehokkuuden varmistamiseksi. Yhteensopivuustestaus seuraa tätä, tarkistamalla sovelluksen toimivuuden eri laitteilla ja alustoilla, mikä on ratkaisevaa laajan käyttäjäkunnan saavuttamiseksi.

Viimeisenä, tietoturvatestaus keskittyy sovelluksen mahdollisten haavoittuvuuksien tunnistamiseen ja korjaamiseen, mikä on elintärkeää käyttäjien arkaluonteisten tietojen suojaamiseksi. Näiden testausvaiheiden yhdistelmä muodostaa perustan kattavalle testausstrategialle, joka ei ainoastaan vähennä kehityskustannuksia, vaan myös parantaa merkittävästi lopputuotteen laatua.

Yhteenvetona, tässä osiossa on kerrottu, kuinka testaus integroituu osaksi laajempaa kehitysprosessia ja kuinka se auttaa varmistamaan, että sovellus ei ainoastaan täytä teknisiä vaatimuksia, vaan ylittää myös loppukäyttäjien odotukset laadun ja suorituskyvyn osalta. Tämä tarkastelu täydentää edellisiä osioita tarjoten syvällisen ymmärryksen testauksen merkityksestä kehityksen kaikissa vaiheissa.

2.6 Frontend

Frontend toimii käyttäjän rajapintana ohjelmistossa. Yleisesti frontend totetutetaan verkkoteknologioilla kuten JavaScript, HTML:llä tai CSS:llä tai esimerkiksi Javalla. Responsiivinen suunnittelu on tärkeää varmistuakseen, että käyttöliittymä toimii saumattomasti eri laitteilla. Käyttöliittymän toteutuksen tulisi perustua käytettävyyden suunnitteluun, mikä varmistaa käyttäjäkokemuksen sujuvuuden. Tärkeintä on yhdistää käyttäjän tarpeet ja ohjelmiston potentiaali suunnittelussa. Tavoitteena on luoda käyttöliittymä, joka tukee käyttäjän tavoitteita ja tarjoaa parhaan mahdollisen käyttökokemuksen. (Mitä tarkoittaa Frontend ja BackEnd ohjelmistokehityksessä? 2021.)

Edellisissä osioissa käsiteltiin sovelluksen suunnitteluvaiheita ja testauksen merkitystä ohjelmiston kehityksessä. Nyt siirrymme tutkimaan erityisesti sovelluksen frontend-kehitystä, joka toimii käyttäjän rajapintana. Frontend eli käyttöliittymä on se osa sovellusta, jonka kanssa käyttäjä vuorovaikuttaa suoraan, ja se on toteutettu usein verkkoteknologioilla kuten JavaScript, HTML ja CSS. Myös Javaa voidaan käyttää käyttöliittymän kehittämisessä, mutta yleisimmin käytetään edellä mainittuja teknologioita.

Frontend-suunnittelussa keskeistä on responsiivisuus, mikä tarkoittaa käyttöliittymän kykyä mukautua saumattomasti eri laitteiden näyttöihin. Tämä on välttämätöntä, sillä käyttäjät voivat käyttää ohjelmistoa monenlaisilla laitteilla, kuten älypuhelimilla, tableteilla ja tietokoneilla. Responsiivinen suunnittelu takaa, että käyttökokemus on laadukas riippumatta laitteesta, jolla sovellusta käytetään.

Käyttöliittymän kehityksessä on tärkeää noudattaa käytettävyyden periaatteita. Käytettävyys tarkoittaa sitä, kuinka helppoa sovelluksen käyttö on loppukäyttäjille. Sujuva käyttäjäkokemus edellyttää, että suunnittelussa huomioidaan käyttäjien tarpeet ja ohjelmiston tekniset mahdollisuudet. Tavoitteena on luoda käyttöliittymä, joka ei ainoastaan täytä teknisiä vaatimuksia, vaan myös tukee käyttäjien tavoitteita ja tarjoaa miellyttävän käyttökokemuksen.

Tämän ymmärryksen pohjalta voimme siirtyä tarkastelemaan myös backend-kehitystä, joka käsittelee sovelluksen palvelinpuolen toimintaa ja tietojenkäsittelyä, mutta joka on näkymätön loppukäyttäjille. Frontend ja backend yhdessä muodostavat täyden ohjelmistoratkaisun, jossa frontend vastaa käyttäjäkokemuksesta ja backend huolehtii tietojen prosessoinnista ja liiketoimintalogiikasta. Tämä integraatio on elintärkeä ohjelmiston toiminnallisuuden ja tehokkuuden kannalta.

2.7 Backend

Backend vastaa ohjelmiston tai palvelun toiminnasta, tiedon tarjoamisesta ja käsittelystä. Yleensä se toimii palvelimella ja voi olla toteutettu esimerkiksi Java, JavaScript tai Python -ohjelmointikielillä. Backend käsittelee frontendin lähettämiä pyyntöjä ja hallinnoi tietoa, usein käyttäen erillistä tietokantapalvelinta, joka saattaa sijaita eri palvelimella kuin itse backend. Backendin suunnittelussa on tärkeää ottaa huomioon nykyiset ja tulevat tarpeet, jotta tietorakenteet ja ohjelmistoarkkitehtuuri mahdollistavat joustavan ohjelmistokehityksen sekä nykyhetkessä että tulevaisuudessa. (Mitä tarkoittaa Frontend ja BackEnd ohjelmistokehityksessä? 2021.)

Voidaan sanoa, että backendin toteutus on ohjelmiston keskipiste, koska kaikki palvelun käyttäjät vuorovaikuttavat palvelinohjelmiston kanssa, oli frontend sitten verkkosivusto, puhelinsovellus, IoT tai jokin muu käyttöliittymä. (Mitä tarkoittaa Frontend ja BackEnd ohjelmistokehityksessä? 2021.)

Edellisissä osioissa käsittelimme frontend-kehityksen merkitystä käyttäjäkokemuksen ja käytettävyyden näkökulmasta. Nyt siirrymme tarkastelemaan backend-kehitystä, joka toimii ohjelmiston tai palvelun toiminnan ytimessä. Backend on se osa järjestelmää, joka käsittelee datan tallennusta, käsittelyä ja tiedon välittämistä frontendille. Backendin rooli ohjelmiston arkkitehtuurissa on kriittinen, sillä se varmistaa, että käyttöliittymän pyynnöt käsitellään tehokkaasti ja turvallisesti.

Backend-kehityksessä käytetään usein ohjelmointikieliä kuten Java, JavaScript ja Python. Nämä teknologiat mahdollistavat monipuolisen ja skaalautuvan palvelinpuolen toteutuksen, joka voi sisältää monimutkaisia tietorakenteita ja tietokantahallintaa. Backendin suunnittelussa on keskeistä huomioida järjestelmän nykyiset sekä tulevat tarpeet, jotta arkkitehtuuri tukee joustavaa kehitystä ja ylläpitoa pitkällä tähtäimellä.

Backend vastaa myös tietokannan hallinnasta, mikä kattaa tiedon tallennuksen, päivityksen ja kyselyn käsittelyn. Tietokantapalvelimet, kuten MySQL tai MariaDB, ovat olennaisia backendin toiminnassa, ja ne voivat sijaita samalla tai eri palvelimella kuin itse sovelluslogiikka. Tämä mahdollistaa tehokkaan datan käsittelyn ja hallinnan, mikä on välttämätöntä suurten tietomäärien ja monimutkaisten kyselyiden käsittelyssä.

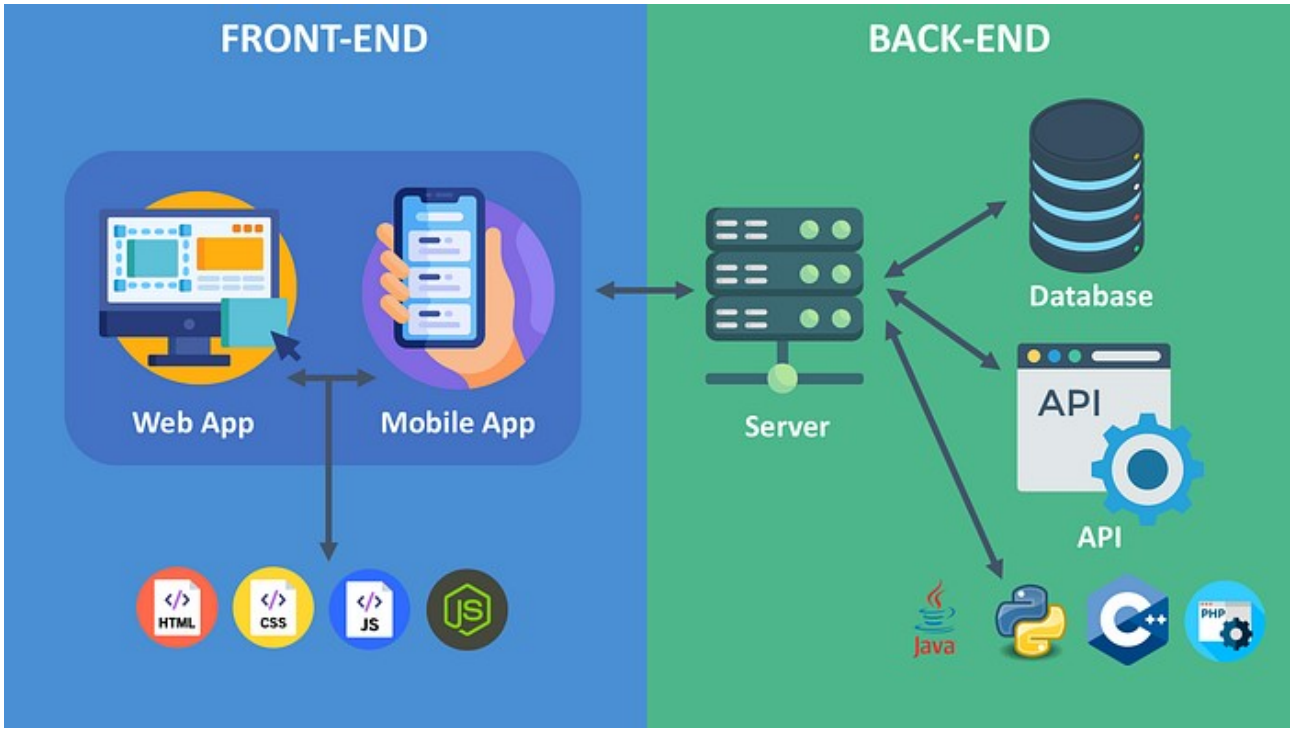
Backendin toteutuksen merkitys korostuu erityisesti siinä, miten se mahdollistaa eri käyttöliittymien, kuten verkkosivustojen, mobiilisovellusten ja IoT-laitteiden, saumattoman toiminnan. Se on ohjelmiston keskipiste, jossa kaikki käyttäjät vuorovaikuttavat palvelinohjelmiston kanssa. Tästä syystä backendin suunnitteluun ja kehitykseen tulee suhtautua yhtä huolellisesti kuin frontendin kehitykseen, varmistaen, että koko ohjelmisto toimii harmonisesti ja täyttää loppukäyttäjien tarpeet.

Kokonaisuutena voidaan sanoa, että backendin ja frontendin yhteistyö on ratkaisevaa ohjelmiston toimivuuden kannalta. Kun backend varmistaa tietojen oikean käsittelyn ja toimittamisen, frontend puolestaan takaa, että käyttäjäkokemus on sujuva ja miellyttävä. Tämä integraatio on keskeinen tekijä ohjelmiston menestyksessä, ja se edellyttää jatkuvaa yhteistyötä ja kommunikaatiota kehitystiimin sisällä.

Seuraavassa kuvassa on havainnollistettu web-kehityksen kahta perusosaa: front-end ja back-end. Front-end, joka on merkitty kuvassa vasemmalle puolelle, viittaa käyttöliittymään ja käyttäjän kanssa vuorovaikutuksessa oleviin osiin. Front-end kehityksessä käytetään teknologioita kuten HTML, CSS ja JavaScript. HTML määrittelee sivun rakenteen, CSS vastaa ulkoasusta ja JavaScript tarjoaa dynaamisia toimintoja. Näitä tekniikoita käytetään sekä web-sovelluksissa että mobiilisovelluksissa.

Back-end, oikealla puolella kuvassa, koostuu palvelimesta, tietokannasta ja sovellusrajapinnoista (API:sta), jotka yhdessä muodostavat sovelluksen palvelinpuolen logiikan. Palvelin vastaanottaa ja käsittelee pyyntöjä, tietokanta säilyttää tietoja, ja API:t mahdollistavat kommunikoinnin palvelimen ja front-endin välillä. Kuvassa on mainittu back-end ohjelmointikieliä ja teknologioita, kuten Java, Python, C++, ja PHP, joita käytetään luomaan back-end toiminnallisuutta.

Kuvassa (ks. kuvio 2) front-end ja back-end ovat yhdistetty nuolilla, mikä tarkoittaa jatkuvaa tietovirtaa näiden kahden osa-alueen välillä. Käyttäjien toiminnat front-endissa aiheuttavat dataa kulkeutuvan back-endiin käsiteltäväksi, ja back-end puolestaan välittää tietoa takaisin front-endiin näytettäväksi. Tämä jakautuminen korostaa modernin web-kehityksen modulaarista luonnetta, jossa erilliset kerrokset vastaavat eri toiminnallisuuksista.



Kuvio 2. Frontend ja Backend toimintalogiikka (Backend for Frontend Basics: A Comprehensive Guide)

3 Työssä käytettävät ohjelmistokehykset

3.1 React

3.1.1 Yleistä

React tarjoaa tehokkaan lähestymistavan käyttöliittymien rakentamiseen. Se helpottaa sivujen luomista jakamalla ne pienempiin osiin, joita kutsutaan komponenteiksi. React-komponentti on käytännössä pieni koodinpätkä, joka kuvaa tiettyä osaa sivusta. Jokainen komponentti on itse asiassa JavaScript-funktio, joka palauttaa koodin, joka kuvaa kyseistä verkkosivun osaa. React käyttää JSX-syntaksia, joka muistuttaa HTML:ää, tarjoten dynaamisen sisällön luomisen helppouden. Lisäksi Reactin komponenttiarkkitehtuuri mahdollistaa tiedonvälityksen komponenttien välillä propsien avulla, ja tilan hallinta toteutetaan erityisten mekanismien, kuten tilan ja elinkaarimetodien kautta, mikä tekee sovellusten tilanhallinnasta johdonmukaisen ja ennustettavan. (Ström, C, 2019)

3.1.2 Historia

Reactin alku juontaa juurensa vuoteen 2010, jolloin Jordan Walke, Facebookin ohjelmistoinsinööri, loi sisäisen prototyypin nimeltä ”FaxJS”. Tämä innovatiivinen ratkaisu syntyi vastaamaan Facebookin kohtaamaan erityiseen haasteeseen – tarpeeseen päivittää uutissyöte dynaamisesti reaaliajassa ilman sivun päivittämistä. FaxJS ratkaisi tämän ongelman tehokkaasti ja sai Facebookin tunnistamaan laajemman käyttöliittymäkehityspotentiaalinsa. (Kumar 2023.)

Vuonna 2011 Facebook otti Reactin käyttöön sisäisesti, ja myöhemmin kirjasto paljastettiin avoimen lähdekoodin projektina vuonna 2013. React sai nopeasti vetovoimaa ja kiehtoi kehittäjiä, jotka omaksuivat sen innokkaasti käyttöliittymien rakentamiseen. React erottui muista ohjelmointikehyksistä ainutlaatuisella kyvyllään tehdä saumattomasti reaaliaikaisia muutoksia, mikä tekee siitä erityisen dynaamisten verkkosovellusten luomiseen. (Kumar 2023.)

3.1.3 Nykypäivä

React on yksi suosituimmista ja laajimmin käytetyistä front-end kirjastoista. Vankan yhteisön tukemana suuret teknologiajättiläiset, kuten Uber, Facebook ja AirBnB, käyttävät Reactia sovelluksiinsa. Reactin oppiminen avaa ovet lukuisille työmahdollisuuksille. Ohjelmistokehittäjänä Reactin

hallitseminen on keskeinen taito. React-kehittäjien kasvava kysyntä osoittaa, että mahdollisuudet alalla kukoistavat myös tulevina vuosina. (Sikiru 2024.)

Reactin ekosysteemi on laaja ja sisältää monia kirjastoja ja työkaluja, kuten Redux, MobX ja React Router, jotka yksinkertaistavat kehitystyötä ja lisäävät toiminnallisuutta. Yhteisö on erittäin aktiivinen ja tarjoaa kattavaa dokumentaatiota, tutoriaaleja ja tukea, mikä on kriittistä sekä uusille että kokeneille kehittäjille (React Ecosystem in 2024. 2024)

3.2 React Native

3.2.1 Yleistä

React Native on suosittu avoimen lähdekoodin ohjelmointikehys mobiilisovellusten kehittämiseen. Se mahdollistaa sekä iOS- että Android-alustoille tarkoitettujen sovellusten luomisen yhdellä koodilla, hyödyntäen Reactin komponentteja. Yksi keskeinen ominaisuus on ns. hot-reloading, joka näyttää koodin muutokset välittömästi näytöllä. (Zammetti, 2018.)

3.2.2 Historia

Facebook kehitti React Nativen ja julkaistiin vuonna 2015. Se kehitettiin alun perin tukemaan iOS-alustaa, ja Android-tuki lisättiin myöhemmin samana vuonna. React Native saavutti nopeasti suosiota kehittäjien keskuudessa. Sitä käyttävät monet suuryritykset mobiilisovelluksissaan, mukaan lukien Facebook, Instagram ja Airbnb, mikä osoittaa sen luotettavuuden ja skaalautuvuuden. React Nativella on merkittävä yhteisö, joka on edistänyt sen rikasta ekosysteemiä kolmannen osapuolen liitännäisillä, kirjastoilla ja työkaluilla, jotka parantavat sen toiminnallisuutta ja käytettävyyttä. (Komperla et al., 2022.)

3.2.3 Nykypäivä

Viimeaikaiset kehityssuunnat ja projektit, jotka hyödyntävät React Nativea, ovat keskittyneet parantamaan mobiilisovellusten kehittämistä eri aloilla. Esimerkiksi Mobile Incident Management System -järjestelmä kehitettiin React Nativella parantamaan tapahtumien hallinnan ketteryyttä

mobiilialustalla (Aran & Cabañero, 2023). Lisäksi REUNIFY, työkalu, joka integroi React Native -sovellusten JavaScript- ja natiivikoodin parempaan staattiseen analyysiin, osoittaa jatkuvia edistysaskeleita kehyksen kyvyissä monimutkaisten sovelluskehitysskenaarioiden osalta (Liu et al., 2023).

4 Projektissa käytettävät työkalut

4.1 EXPO

Expo on alusta React Native-sovellusten rakentamiseen. Sen tavoitteena on tarjota yksinkertaistettu kehityskokemus React Native ohjelmien tekemiseen.

Expo tarjoaa nopean tavan aloittaa sovelluskehitys ilman, että kehittäjien on tarpeen huolehtia laitekohtaisista asetuksista tai natiivikoodiriippuvuuksista. Sen avulla kehittäjät voivat lähettää päivityksiä suoraan käyttäjien laitteisiin Over-The-Air (OTA) -menetelmällä, mikä eliminoi tarpeen ladata uusia versioita sovelluskaupoista. Lisäksi Expo sisältää laajan valikoiman valmiiksi rakennettuja kirjastoja ja API:ita, jotka helpottavat uusien ominaisuuksien, kuten push-ilmoitusten, sovelluksen sisäisten ostosten ja karttojen, lisäämistä sovelluksiin. Sovelluksen rakennusprosessi on yksinkertaistettu, sillä Expo hoitaa sen kehittäjän puolesta, piilottaen tarpeen konfiguroida työkaluja tai käsitellä natiivikoodin kääntämistä. Expo App Services (EAS) avulla kehittäjät voivat rakentaa ja julkaista sovelluksia iOS:lle ilman Macia, käyttämällä pilvipohjaista rakennus- ja julkaisuprosessia. Lisäksi Expo Go -sovelluksen avulla kehittäjät voivat yhdistää sovelluksensa fyysisiin Android- tai iOS-laitteisiin samassa verkossa ja testata niitä reaaliaikaisesti. (Softworth Solutions Private Limited, 2023)

4.2 Visual Studio Code

Visual Studio Code (VS Code) on ketterä ja vahva koodieditori, joka on yhteensopiva Windows-, macOS- ja Linux-järjestelmien kanssa. Siinä on tuki JavaScriptille, TypeScriptille ja Node.js:lle, samalla kun se tarjoaa monipuolisen valikoiman laajennuksia eri kielille, mukaan lukien muun muassa C++, C#, Java, Python, PHP, Go ja .NET. Sen monipuolisuus ja laajennettavuus tekevät siitä suositun valinnan saumatonta koodauskokemusta etsivien kehittäjien keskuudessa. (Visual Studio Getting Started, N.d)

4.3 Git ja Github

GitHub on web-pohjainen versionhallintajärjestelmä ja yhteistyöalusta ohjelmistokehitykseen. Se hyödyntää Git-versionhallintajärjestelmää, joka on avoimen lähdekoodin hajautettu versionhallintaohjelmisto, mahdollistaen kehittäjien hallita projektejaan tehokkaasti. GitHub tarjoaa keinoja ohjelmakoodin tallentamiseen, seuraamiseen ja jakamiseen, sekä yhteistyön tekemiseen muiden kehittäjien kanssa koodin parissa. GitHub on suosittu työkalu sekä yksittäisten kehittäjien että suurten organisaatioiden keskuudessa, ja se mahdollistaa monenlaiset työnkulut, kuten koodin tarkastelun, projektinhallinnan ja automatisoinnin. (HTG Staff, 2016)

4.4 Google Firebase

Google Firebasen PaaS (Platform-as-a-Service) alusta on erityisesti suunniteltu mobiilisovellusten ja progressiivisten web-sovellusten kehittämiseen. Sen tarjoama valmius sovelluskehitykseen mahdollistaa kehittäjien aloittavan uusien sovellusten luonnin nopeasti ilman tarvetta oman alustan tai infrastruktuurin asentamiseen ja ylläpitoon. Firebasen käytön helppous ja kustannustehokkuus ovat sen keskeisiä etuja. Aloittaminen on maksutonta, ja monet sen tarjoamista toiminnoista ovat jatkuvasti ilmaisia. Kun datan käyttö lisääntyy, kulut perustuvat käyttömäärään, mahdollistaen kustannustehokkaan tavan hyödyntää palvelua, joka voi parhaimmillaan pysyä täysin maksuttomana. (Hulkkonen, 2022)

5 React Native -projektin aloittaminen

5.1 Kehitysympäristön asennus

Kehitysympäristön asennus on ensimmäinen askel React Native -sovellusprojektin aloittamisessa EXPOn avulla. EXPO mahdollistaa sovellusten kehittämisen ilman, että kehittäjän tarvitsee asentaa ja konfiguroida iOS- ja Android-kehitysympäristöjä erikseen, mikä säästää merkittävästi aikaa ja vaivaa.

Ennen kuin kehittämisen aloittaa, pitää varmistaa, että koneelle on asennettuna Node.js, joka on Expo-sovelluskehityksen perusta. Node.js:n avulla voit käyttää npm-pakettienhallintaa, joka on tarpeen Expo CLI:n asentamiseksi. Expo CLI on komentorivityökalu, joka auttaa luomaan uusia projekteja, käynnistämään kehityspalvelimen ja siirtämään sovelluksen testauslaitteisiin.

Seuraavaksi asennetaan koodieditori mitä käytetään. Visual Studio Code (VS Code) on erinomainen valinta editoriksi React Native -projekteissa. Tämä kevyt, mutta tehokas koodieditori tarjoaa laajan valikoiman ominaisuuksia, jotka tekevät koodin kirjoittamisesta, debuggauksesta ja hallinnasta nopeaa ja tehokasta. VS Code tukee monia eri kieliä ja teknologioita ja sen toiminnallisuutta voidaan laajentaa lukemattomilla lisäosilla, jotka on suunniteltu helpottamaan kehittäjän työtä. Kun kehittää Expo-projekteja, Visual Studio Code yhdessä Expo-työkalujen kanssa mahdollistaa erinomaisen kehityskokemuksen.

5.2 Ensimmäisen sovelluksen luominen

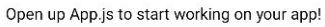
Kehitysympäristön asennuksen jälkeen avataan VS Code sisäänrakennetun komentokehotteen ja siellä aloitetaan uusi projekti käyttämällä ”npx create-expo-app sovelluksenNimi” -komentoa. Expo CLI luo projektin ja asentaa kaikki tarvittavat riippuvuudet. Tämän jälkeen mennään kansioon komennolla ”cd sovelluksenNimi”. Seuraavaksi käynnistetään projekti ”npx expo start”, mikä mahdollistaa sovelluksen esikatselun reaaliajassa joko fyysisessä laitteessa tai emulaattorissa.

Tämän jälkeen voit avata kehitteillä olevan sovelluksen puhelimella QR-koodin avulla. Laite millä sovellus avataan pitää olla samassa verkossa kuin tietokone. Seuraavassa kuviossa näet näkymän, kun olet käynnistänyt projektin (ks. kuvio 3). Kun sovellus on käynnissä ja toimii Expo Gon kautta, voit hyödyntää Expo-alustan tarjoamia etuja, joka päivittää sovelluksen näkymät välittömästi muutosten jälkeen ilman, että tarvitsee käynnistää sovellusta uudelleen.

Kuvio 3. EXPO QR-koodi, jolla sovellus avataan puhelinsovelluksella.

5.3 Perusnäkymä ja komponentit

Edellisessä kappaleessa luotiin ensimmäinen sovellus. Alla olevassa kuviossa on näkymä sovelluksen asennuksen jälkeen, kun se avataan EXPO sovelluksessa (ks. kuvio 4). Perusnäkymää pystyy muokkaamaan ”App.js” tiedostosta. Tiedostossa voimme määritellä elementtejä, kuten tekstikomponentteja, painikkeita ja kuvia, jotka yhdessä luovat rakenteen käyttöliittymälle.



Kuvio 4. React Native näkymä asennuksen jälkeen.

Edellinen kuva oli graafinen näkymä asennuksen jälkeen. Seuraavassa kuvassa näkyy, minkälainen peruskoodi on asennuksen jälkeen. Käyttöliittymän luomisessa keskeiset komponentit ovat ”View”, ”Text”, ”Image” ja ”Button”, joihon tutustaan enemmän seuraavassa luvussa. Tyylien määrittely tapahtuu Stylesheet-komponentin avulla.

```
JS App.js > ...
1   import { StatusBar } from 'expo-status-bar';
2   import { StyleSheet, Text, View } from 'react-native';
3
4   export default function App() {
5     return (
6       <View style={styles.container}>
7         <Text>Open up App.js to start working on your app!</Text>
8         <StatusBar style="auto" />
9       </View>
10    );
11  }
12
13  const styles = StyleSheet.create({
14    container: {
15      flex: 1,
16      backgroundColor: '#fff',
17      alignItems: 'center',
18      justifyContent: 'center',
19    },
20  });
21
22
```

Kuvio 5. Oletuskoodi asennuksen jälkeen.

5.4 React Native komponentit

5.4.1 Core Components

Ydinkomponentit (Core Components) ovat olennaisia osia kaikissa React Native -sovelluksissa. Ne ovat alustasta riippumattomia JavaScript-komponentteja, jotka renderöityvät sujuvasti käyttäen kunkin mobiilialustan, kuten iOS:n tai Androidin, natiiveja käyttöliittymäelementtejä. Tämä mahdollistaa JavaScript-koodin ja alustan natiivielementtien välisen yhteistyön. Ydinkomponentit tukevat alustojen yli toimivaa renderöintiä, mikä takaa johdonmukaisen käyttäjäkokemuksen eri laitteilla ja parantaa sovelluksen suorituskykyä sekä käyttäjien kokemusta sovelluksesta. (Gentile, 2023)

Seuraavassa kuvassa on esimerkkejä, joka esittelee ydinkomponenttien käyttöä, jotka löytyvät React Nativesta. View toimii yleisenä säiliönä, johon muut komponentit sijoitetaan. Teksti esitetään Text-komponentilla, kun taas Image-komponentti näyttää kuvat. ScrollView mahdollistaa sisällön vierittämisen ja TextInput antaa käyttäjille mahdollisuuden syöttää tekstiä. StyleSheet-komponentin avulla määritellään sovelluksen visuaalinen tyyli.

```
import React from 'react';
// import the core components from React Native
import { View, Text, Image, ScrollView, TextInput, StyleSheet } from 'react-native';

// core view (basic container) component
function ViewComponent(){
  // styles prop allows you to control/manipulate visual presentation
  return <View><Text>Core Component!</Text></View>;
};

// core image component
function ImageComponent(){
  return <Image source='https://via.placeholder.com/150' />;
};

// core scrollview (scrolling list) component
function ScrollViewComponent(){
  return (
    <ScrollView>
      <Text>{/* Long text content for scrolling list */}</Text>
    </ScrollView>
  );
};

// core text input (form-like) component
function TextInputComponent(){
  return <TextInput placeholder="Enter text here..." />;
};

function CoreComponentsExample(){
  return (
    <>
      <ViewComponent />
      <ImageComponent />
      <ScrollViewComponent />
      <TextInputComponent />
    </>
  );
};
```

Kuvio 6. React Native Core Components esimerkkejä. (Basics of React Native: Three Types of Components for Your Application)

Ydinkomponenttien keskeinen rajoitus React Native -kehityksessä liittyy niiden yksinkertaisuuteen ja yhdenmukaisuuteen. Ne eivät tarjoa laajaa mukautettavuutta perusominaisuuksiensa ulkopuolella. Erityisesti räätälöityjä ja optimoituja ratkaisuja varten on suositeltavaa tarkastella muita komponenttiluokkia. (Gentile, 2023)

5.4.2 Custom Components

Mukautetut natiivikomponentit ovat sovelluskohtaisesti kehitettyjä elementtejä, jotka muunnetaan natiiviksi käyttöliittymäelementeiksi iOS- ja Android-alustoilla React Native -alustan kautta. Kehittäjät voivat hyödyntää näitä komponentteja laitekohtaisiin toimintoihin, kuten kameran tai GPS:n käyttöön, jotka eivät ole valmiiksi sisällytettyjä koodikantaan. Lisäksi mukautetut komponentit tukevat kolmannen osapuolen moduulien integrointia sovellukseen, mikä edistää kehitystä ja optimointia koodinjakamisen avulla. (Gentile, 2023)

Kuviossa 7 näkyy koodi React Native -sovelluksesta, joka integroi kameratoiminnallisuuden käyttämällä react-native-camera -moduulia. Koodissa on määritelty tilanhallintaa kameran käytettävyydelle, funktiot kameran avaamiseen ja sulkemiseen sekä renderöintifunktio, joka määrittää kameran asetuksia kuten tyyppi, automaattitarkennus ja salamavalon tila. Lisäksi koodi sisältää alustakohtaisen logiikan käyttöliittymäkomponenttien renderöimiseksi riippuen siitä, suoritetaanko sovellus iOS:ssä tai Androidissa. Käyttöliittymässä on myös napit kameran avaamiseen ja sulkemiseen.

```
import React, { useState } from 'react';
import { View, Button, StyleSheet, Platform } from 'react-native';
// import library to interact with device camera
import { RNCamera } from 'react-native-camera';

// set state for camera status
function CameraComponent() {
  const [cameraOpen, setCameraOpen] = useState(false);

  // alters state of the camera toggle
  function handleOpenCamera(){
    setCameraOpen(true);
  };

  // alters state of the camera toggle
  function handleCloseCamera(){
    setCameraOpen(false);
  };

  // uses core components to build out/enhance functionality of custom component
  function renderCamera(){
    return (
      <View>
        <RNCamera
          type={RNCamera.Constants.Type.back}
          autoFocus={RNCamera.Constants.AutoFocus.on}
          flashMode={RNCamera.Constants.FlashMode.off}
          captureAudio={false}
        />
        <Button title="Close Camera" onPress={handleCloseCamera} />
      </View>
    );
  };

  return (
    // uses the module to determine what platform the app is run on
    <View>
      {Platform.OS === 'ios' && !isCameraOpen && <Button title="Open Camera" onPress={handleOpenCamera} />}
      {Platform.OS === 'ios' && isCameraOpen && renderCamera()}
    </View>
  );
}
```

Kuvio 7. React Native custom component esimerkki (Basics of React Native: Three Types of Components for Your Application)

5.4.3 Community Components

Kuten edellä mainittiin, React Native -kehittäjäyhteisö kehittää, ylläpitää ja jakaa näitä komponentteja. Ne levitetään yleisesti npm-paketteina, jotka voidaan integroida sovelluksiin, tarjoten monipuolisia toiminnallisuuksia ja käyttäjäkokemuksia. Nämä komponentit voi ajatella olevan toisten kehittäjien tai kehitystiimien luomia räätälöityjä osia. Usein tehokkaimmat koodiratkaisut syntyvät monien kehittäjien yhteistyön tuloksena pitkän ajan kuluessa. Yhteisökomponentit ovat tästä erinomainen esimerkki: ne tarjoavat keinoja yksilöille luoda ja päivittää jaettuja, räätälöityjä kom-

ponentteja, jotka edistävät koodausyhteisön etua. Tämä käytäntö ei ainoastaan edistä teknologista innovaatiota, vaan myös vahvistaa yhteisön sisäistä yhteistyötä ja tietojen jakamista. (Gentile, 2023)

Alla olevassa kuvassa esimerkki, jossa on käytetty yleistä yhteisökomponenttia ”react-navigation” näyttöjen väliseen navigointiin. Koodi sisältää kaksi näyttöä, "Screen One" ja "Screen Two", jotka on määritelty käyttäen ”StackNavigator”-komponenttia. Kummassakin näytössä on View-komponentti, jossa on teksti, joka kertoo käyttäjälle, missä näytössä hän on, sekä nappi, joka ohjaa käyttäjän toiseen näyttöön. Ensimmäisessä näytössä on nappi nimeltä "Go to Screen Two", joka siirtää käyttäjän toiseen näyttöön käyttäen navigation.navigate-funktiota. Toisessa näytössä on vastaavasti nappi nimeltä "Go back", joka palauttaa käyttäjän takaisin ensimmäiseen näyttöön navigation.goBack-funktion avulla.

```jsx
import React from 'react';
import { View, Text, Button, StyleSheet } from 'react-native';

// imported the community component
import { NavigationContainer } from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/stack';

const Stack = createStackNavigator();

function App(){
  return (
    // implementing that community component in your app
    <NavigationContainer>
      <Stack.Navigator initialRouteName="ScreenOne">
        <Stack.Screen
          name="ScreenOne"
          component={() => (
            <View style={styles.container}>
              <Text>Screen One</Text>
              <Button title="Go to Screen Two" onPress={() => navigation.navigate('ScreenTwo')} />
            </View>
          )}
        />
        <Stack.Screen
          name="ScreenTwo"
          component={() => (
            <View style={styles.container}>
              <Text>Screen Two</Text>
              <Button title="Go back" onPress={() => navigation.goBack()} />
            </View>
          )}
        />
      </Stack.Navigator>
    </NavigationContainer>
  );
};

export default App;
```

Kuvio 8. React Native yhteisökomponetti esimerkki (Basics of React Native: Three Types of Components for Your Application)

Kolmannen osapuolen kehittämien komponenttien käyttöönotossa on useita haasteita, jotka on otettava huomioon. Koska nämä komponentit luodaan ja päivitetään ulkopuolisten toimesta, niiden ylläpito ja tekninen tuki eivät välttämättä ole aina jatkuvia tai luotettavia. Lisäksi tietyt yhteisössä suositut komponentit voivat menettää suosiotaan ajan myötä, mikä saattaa johtaa odottamattomiin ongelmiin ja virheisiin sovelluksessa. Siksi on tärkeää suorittaa säännöllisiä tarkastuksia sovelluksen komponenteille, jotta voidaan varmistaa niiden toimivuus ja ajantasaisuus. Tämä edellyttää jatkuvaa valppautta ja proaktiivisuutta sovelluskehittäjiltä, jotta sovelluksen laatu ja toimintavarmuus säilyvät korkealla tasolla. (Gentile, 2023)

6 Sovelluksen suunnittelu

6.1 Sovelluksen vaatimukset

Sovelluksen kehittäminen tiekuntien tarpeisiin vaatii ymmärrystä niiden ylläpidollisista ja hallinnollisista tehtävistä. Tärkeää on rakentaa sovellus, joka tukee sekä reaaliaikaista tiedonkeruuta että -hallintaa, ottaen huomioon teiden kunnon seurannan, kunnossapitotöiden suunnittelun ja toteutuksen sekä taloushallinnon ja viestinnän. Sovelluksen tulisi mahdollistaa kunnossapitotöiden helppo aikataulutus, budjetin seuranta ja jäsenmaksujen käsittely. Lisäksi sen tulisi tarjota alustan tehokkaalle viestinnälle tiekunnan jäsenten välillä, varmistaen, että kaikki ovat ajan tasalla suunnitelluista töistä ja muista tärkeistä päivityksistä.

Raportointi- ja analytiikkatoiminnot ovat olennaisia, sillä ne auttavat ymmärtämään teiden kunnossapidon tehokkuutta ja tiekunnan taloudellista tilannetta. Tiedon analysointi auttaa tunnistamaan mahdollisia ongelma-alueita ja trendejä, mikä puolestaan mahdollistaa ennaltaehkäisevien toimenpiteiden suunnittelun ja toteutuksen.

Yksi keskeisistä vaatimuksista on yhteensopivuus eri alustojen kanssa. Tiekuntien käyttäjäkunta on monimuotoinen, sisältäen eri ikäisiä ja teknisesti eri tasolla olevia henkilöitä, jotka käyttävät sekä iOS- että Android-laitteita. Tästä syystä on välttämätöntä, että sovellus tarjoaa saumattoman ja yhdenmukaisen käyttökokemuksen kaikilla laitteilla, jotta kaikki käyttäjät voivat hyödyntää sovellusta tehokkaasti ilman teknisiä esteitä.

Tietoturva ja tietosuoja ovat myös äärimmäisen tärkeitä tekijöitä sovelluksen kehityksessä, erityisesti kun käsitellään käyttäjien henkilökohtaisia tietoja. Sovelluksen on varmistettava, että kaikki henkilötiedot ovat suojattuja ja että tietosuojakäytännöt noudattavat alueellisia ja kansainvälisiä lakeja ja määräyksiä. Tämä luo luottamusta sovelluksen käyttäjien keskuudessa ja varmistaa, että henkilökohtaiset ja herkät tiedot pysyvät turvassa.

Lisäksi käyttöliittymän selkeys ja helppokäyttöisyys ovat avainasemassa, jotta sovellus on saavutettavissa ja hyödyllinen kaikenikäisille käyttäjille. Selkeä ja intuitiivinen käyttöliittymä tekee sovelluksen opettelusta nopeaa ja vähentää käyttöön liittyvää turhautumista. Tämä on erityisen tärkeää tiekuntien kaltaisille käyttäjäryhmille, jotka saattavat kattaa laajan ikähaarukan ja erilaiset tekniset taidot. Helppokäyttöisyys tarkoittaa, että käyttäjät voivat keskittyä sovelluksen tarjoamien toimintojen tehokkaaseen hyödyntämiseen sen sijaan, että he kamppailisivat vaikeaselkoisen käyttöliittymän kanssa.

6.2 Sovelluksen toiminnallisuudet

Sovelluksen toiminnallisuudet ovat keskeisiä tiekuntien tehtävien hallinnassa ja ylläpidossa. Alla on kuvattu, miten kukin mainittu toiminnallisuus tukee tiekuntien tarpeita:

- **Tiekartta**: Tiekartta-toiminto mahdollistaa tiekunnan vastuulla olevien teiden visuaalisen esittämisen kartalla. Käyttäjät voivat tarkastella tietoja eri teiden sijainnista, kuntoarvioista ja suunnitelluista kunnossapitotoimenpiteistä. Tämä auttaa tiekuntia suunnittelemaan ja priorisoimaan teiden ylläpitoa tehokkaammin.

- **Huoltopyynnöt**: Huoltopyyntöjen hallinta mahdollistaa tiekunnan jäsenten tehdä ilmoituksia teiden kunnossapitoa vaativista ongelmista. Sovellus keskittää nämä pyynnöt helposti hallittavaksi listaksi, josta tiekunnan vastuuhenkilöt voivat seurata ja ohjata huoltotöiden toteutusta.

- **Tiedotus ja viestintä**: Tiedotus- ja viestintätyökalut ovat tärkeitä ylläpitämään avointa linjaa tiekunnan jäsenten välillä. Tämän toiminnallisuuden avulla voidaan lähettää ajankohtaisia tiedotteita, päivityksiä ja muita viestejä, mikä edistää yhteisöllisyyttä ja varmistaa, että kaikki ovat tietoisia tärkeistä asioista.

- **Budjetin seuranta**: Budjetin seurantatyökalu antaa tiekunnille mahdollisuuden hallita taloudellisia resurssejaan tehokkaammin. Se tarjoaa yksityiskohtaisen näkymän tuloista, menoista ja budjetin toteutumasta, mikä auttaa teiden ylläpidon suunnittelussa ja talouden hallinnassa.

- **Dokumenttien ja sopimusten hallinta:** Dokumenttien ja sopimusten hallintatoiminto keskittää kaikki tärkeät asiakirjat yhteen paikkaan, helpottaen niiden hallintaa ja saatavuutta. Tämä sisältää sopimukset urakoitsijoiden kanssa, tiekunnan kokousten pöytäkirjat ja muut viralliset dokumentit, jotka ovat tärkeitä tiekunnan toiminnalle.

- **Push-ilmoitukset:** Push-ilmoitukset ovat tehokas keino pitää tiekunnan jäsenet ajan tasalla tärkeistä tiedotteista ja tapahtumista. Ne varmistavat, että tärkeä informaatio saavuttaa jäsenet välittömästi, parantaen viestinnän nopeutta ja tehokkuutta.

- **Tehtävien aikataulutus:** Tehtävien aikataulutustyökalu mahdollistaa kunnossapitotöiden ja muiden tehtävien suunnittelun ja aikataulutuksen. Tämä auttaa tiekuntia varmistamaan, että kaikki tarvittavat toimet toteutetaan ajallaan ja resurssit käytetään mahdollisimman tehokkaasti.

Kukin näistä toiminnallisuuksista on suunniteltu helpottamaan tiekuntien vastuulla olevien teiden hallintaa ja ylläpitoa, tehostamaan kommunikaatiota jäsenten välillä ja parantamaan yleistä organisatorista tehokkuutta.

6.3 Käyttäjätarinat

Käyttäjätarinat ovat työkalu ohjelmistokehityksessä, joka auttaa selittämään ohjelmiston tavoitteet loppukäyttäjän perspektiivistä. Ne pyrkivät valottamaan, miksi käyttäjä tarvitsee tiettyjä ominaisuuksia ohjelmistossa ja kuinka nämä ominaisuudet tuottavat hänelle hyötyä. Käyttäjätarinat kuvaavat käyttäjän toiveita ja tarpeita selkeästi ja yksinkertaisesti. Nämä tarinat auttavat ohjelmistokehittäjiä ja suunnittelijoita ymmärtämään käyttäjien vaatimukset ja mitä arvoa ohjelmiston tulee tuottaa. Käyttäjätarinan avulla kehitystiimi tietää, mitä heidän on tarkoitus rakentaa ja minkä vuoksi.

Käyttäjätarinoita ilmaistaan usein yksinkertaisella rakenteella: “Käyttäjänä X, haluan Y, jotta Z”. Esimerkiksi: “Kurssin opiskelijana haluan saada ilmoituksia sähköpostitse aina, kun uusi materiaali tai tehtävä on julkaistu kurssialustalle, jotta kurssien tärkeät päivitykset eivät mene ohi ja pysyn ajan tasalla kurssin sisällöstä.”

On tärkeää, että käyttäjätarina ei ole liian epämääräinen tai liian tekninen. Yleiset kuvaukset eivät anna tarpeeksi yksityiskohtia tarvittavista toiminnoista, kun taas liian spesifit tarinat saattavat rajoittaa kehitystiimin luovuutta. Ihanteellinen käyttäjätarina tarjoaa riittävästi tietoa, jotta kehitystiimi voi tehdä itsenäisiä päätöksiä parhaan lopputuloksen saavuttamiseksi.

7 Sovelluksen kehitysvaihe

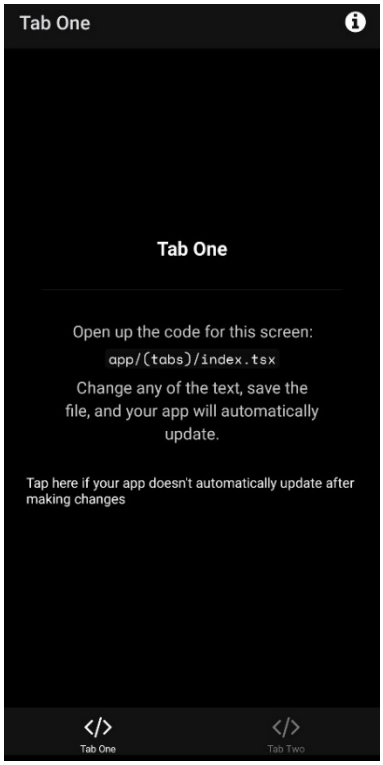
7.1 Käyttöliittymä

Mobiilisovellusten käyttöliittymän suunnittelussa on tärkeää noudattaa käyttäjäkeskeistä lähestymistapaa, joka huomioi mobiilikäyttäjien erityistarpeet, kuten rajoitetun näyttökoon ja vaihtelevan internet-yhteyden. Tämä lähestymistapa auttaa luomaan intuitiivisia ja tehokkaita käyttöliittymiä (Braun et al., 2015). Lisäksi visuaalisen suunnittelun tehokkuus mobiilisovelluksissa korostuu nykyaikaisten mobiililaitteiden kehittyneiden graafisten kykyjen ansiosta. Suunnittelijoiden on otettava huomioon uudet haasteet, kuten käyttäjien lyhyempi keskittymisaika ja korkeammat vuorovaikutuskustannukset (Chittaro, 2011).

Käyttämällä valmiiksi määriteltyjä suunnittelumalleja, jotka tarjoavat ratkaisuja yleisiin ongelmiin, suunnitteluprosessia voidaan tehostaa ja sovellusten käytettävyyttä parantaa (Nilsson, 2009). Kvantitatiivinen analyysi auttaa ymmärtämään suunnitteluelementtien ja käyttäjätyytyväisyyden välisiä yhteyksiä, mikä ohjaa käyttöliittymän parannuksia (Jiang et al., 2019). Mukautettavat ja personoidut käyttöliittymät parantavat käyttökokemusta räätälöimällä käyttöliittymää yksittäisten käyttäjien tarpeiden mukaan, mikä lisää sitoutumista ja tehokkuutta (Braham et al., 2019). Tämä kokonaisvaltainen lähestymistapa on keskeinen tehokkaiden ja käyttäjäystävällisten mobiilisovellusten suunnittelussa.

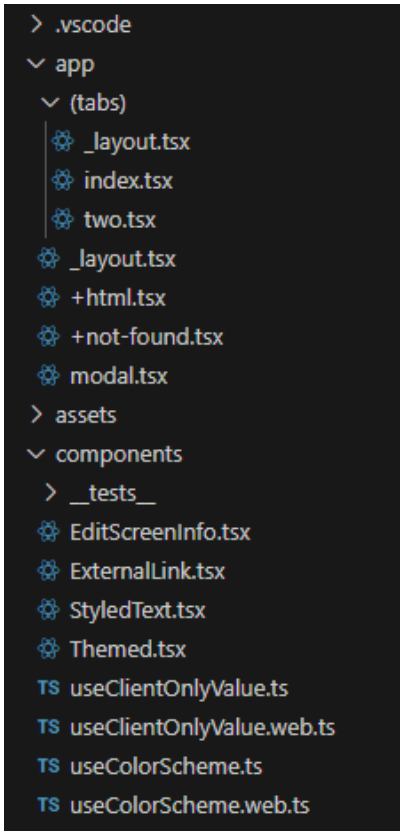
Sovelluksen kehityksessä käytettiin EXPO Tabs -pohjaa käyttöliittymän selkeyden vuoksi. Alhaalla olevat valikot luovat loppukäyttäjälle helpon pääsyn tärkeimpiin resursseihin. Seuraavasta kuviosta (ks. kuvio 9) on nähtävissä sovelluksen oletuskäyttöliittymä asennuksen jälkeen, jossa on valmiina kaksi välilehteä, ”Tab One” ja ”Tab Two”. Ensimmäisessä välilehdessä on myös ohjeet siitä, miten sovelluksen koodia voi oletuksena muokata. EXPO tabs pohja asennetaan komennolla ”npx

create-expo-app tiekunta -t tabs”. Tämän jälkeen sovelluksen voi avata EXPO ohjelmalla puhelimessa.



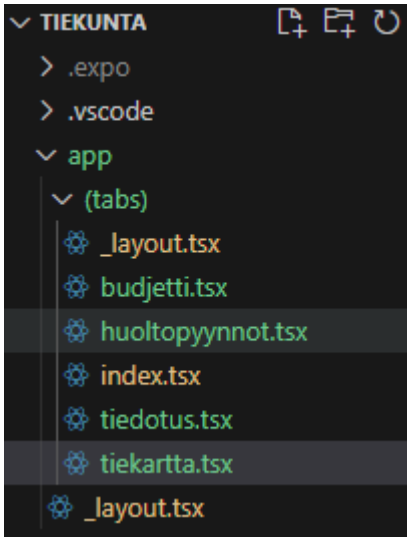
Kuvio 9. EXPO Tabs oletusnäkymä asennuksen jälkeen.

Asennuksen jälkeen poistettiin muutamia asennuksessa tulevia tiedostoja ja lähdettiin melkein puhtaalta pöydältä kehittämään sovellusta. Tiedostot ”(tabs)”- sekä ”components” -kansioista poistettiin ja jätettiin vain (tabs) kansiossa oleva ”_layout.tsx” sekä ”index.tsx”.



Kuvio 10. Asennuksessa tulevat tiedostot

Tiedostojen poiston jälkeen tehtiin omat tiedostot jokaiselle valikon kohteelle. Kuten seuraavasta kuviosta näkyy, että kaikille valikon kohteille on luotuna omat tiedostot. Esimerkiksi, jos halutaan päivittää tietyn valikon kohteen käyttöliittymää, kehittäjä voi tehdä tarvittavat muutokset suoraan kyseisen kohteen tiedostoon. Tämä vähentää virheiden riskiä ja tekee testaamisesta tehokkaampaa, koska muutosten vaikutusalue on rajattu.



Kuvio 11. Itse tehdyt tiedostot valikoille.

Seuraavaksi muokattiin _layout.tsx tiedostoa alla olevan kuvion mukaisesti (ks. kuvio 12.). Tiedostossa käytettiin ”expo-router”-kirjastoa tabien hallintaan. Tiedostoon lisättiin aluksi vain alla oleva koodi ilman mitään muuta sisältöä.

```
import React from 'react';
import { View, Text } from 'react-native';
import { Tabs } from 'expo-router'

export default function Layout() {
  return (
    <Tabs>

    </Tabs>
  );
}
```

Kuvio 12. _layout.tsx tiedosto muokkaamisen jälkeen.

Edellisessä kuviossa lisättiin _layout.tsx tiedostoon koodi, jossa käytettiin ”expo-router”-kirjastoa, ilman mitään muuta sisältöä. Vain tämän koodin lisäämällä saadaan esille kuviossa 13 tehdyt omat tiedostot jokaiselle valikon kohteelle (ks. kuvio 13).

Kuvio 13. Alavalikon kohteet tiedostojen lisäämisen jälkeen.

7.2 Valikkojen muokkaaminen

Edellisen luvun kuviossa 13 tehtiin alavalikot kuntoon. Seuraavaksi valikon kohteille laitettiin ikonit sekä muokattiin valikkotekstiä sopivaksi. Oletuksena valikkojen nimet tulevat tiedoston nimistä mitä on tehty sekä niissä ei ole kohteen kuvaavaa ikonia.

Ikonit on valittu ilmaisista lähteistä, kuten FontAwesome ja Ionicons. Näistä löytyy laaja valikoima ilmaisia ikoneita, jotka kattavat monia eri tarpeita ja käyttötarkoituksia. FontAwesome tarjoaa

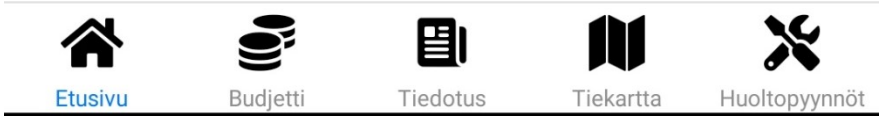
sekä perinteisiä että modernimpia ikoneita, joista löytyy sopivia vaihtoehtoja monenlaisiin projekteihin. Ionicons puolestaan keskittyy erityisesti mobiilikäyttöliittymiin, tarjoten selkeitä ja helposti tunnistettavia ikoneita, jotka parantavat sovellusten käyttökokemusta.

Tabs.Screen:ssä määriteltiin reitiksi "index", joka on tiedoston nimi, käyttämällä name-attribuuttia. Näytön asetukset määriteltiin options-objektissa, jossa välilehden otsikoksi asetettiin "Etusivu" ja ikoniksi FontAwesome-komponentti, jonka nimeksi asetettiin "home" ja koko saatiin dynaamisesti size-prop:ista(Ks kuvio 14).

```
import React from 'react';
import { View, Text } from 'react-native';
import { Tabs } from 'expo-router'
import { FontAwesome } from '@expo/vector-icons';

export default function Layout() {
  return (
    <Tabs>
      <Tabs.Screen
        name="index"
        options={{
          tabBarLabel: 'Etusivu',
          tabBarIcon: ({ size }) => <FontAwesome name="home" size={ size } />
        }}
      />
    </Tabs>
  );
}

```

Kuvio 14. Ensimmäisen valikkokohteen sisältämä koodi.

Tämän jälkeen sama prosessi toistettiin kaikille alavalikon kohteille. Jokaisen alavalikon kohteen kohdalla muutettiin name-attribuutti sopivaksi kyseiselle näytölle. Lisäksi päivitettiin tabBarLabel-attribuutti vastaamaan uuden välilehden otsikkoa.



Kuvio 15. Valmiit alavalikon kohteet.

7.3 Näkymien toiminnallisuudet

7.3.1 Etusivu

Etusivulle laitettiin näkymä (ks. kuvio 16), joka tarjoaa sää- ja yhteystietoja sekä nykyisen päivämäärän. Näiden alla on tiekunnan yhteyshenkilöiden yhteystiedot. Sovellus käyttää OpenWeatherMap API:ta saadakseen ajankohtaiset säätiedot ja Axios-kirjastoa HTTP-pyyntöjen suorittamiseen.

Kuvio 16. Mobiilisovelluksen etusivu.

Koodissa käytettiin Reactin ”useState”-hookia määrittelemään ”weather”-tilan, johon tallennetaan haettu säädata. Aluksi tämä tila on ”null”. ”useEffect”-hookia käytetään suorittamaan säädatan haku, kun komponentti renderöidään ensimmäisen kerran. Tämä tapahtuu ”fetchWeather”-funktiolla, joka suorittaa HTTP GET -pyynnön ”axios.get”-metodilla aiemmin määriteltyyn URL-osoitteeseen. Jos pyyntö onnistuu, saatu data tallennetaan ”weather”-tilaan ”setWeather”-metodin avulla. Jos pyyntö epäonnistuu, virhe tulostetaan konsoliin.

Nykyisen päivämäärän näyttämiseksi koodissa on ”getCurrentDate”-funktio, joka luo uuden ”Date”-objektin ja muuntaa sen paikalliseen päivämäärämuotoon. Tämä tapahtuu käyttämällä ”toLocaleDateString”-metodia, jossa määritellään suomenkielinen (fi-FI) päivämäärämuotoilu, joka näyttää viikonpäivän, vuoden, kuukauden ja päivän (ks. kuvio 17.).

```
const weatherApiKey = [redacted]'; // Replace with your OpenWeatherMap API key
const weatherUrl = `https://api.openweathermap.org/data/2.5/weather?q=Kankaanpää,fi&units=metric&appid=${weatherApiKey}`;

const contacts = [
  { id: '1', name: 'Esa Viitanen', phone: '+35812 345 67 89' },
];

const App: React.FC = () => {
  const [weather, setWeather] = useState<any>(null);

  useEffect(() => {
    const fetchWeather = async () => {
      try {
        const response = await axios.get(weatherUrl);
        setWeather(response.data);
      } catch (error) {
        console.error('Error fetching weather data:', error);
      }
    };

    fetchWeather();
  }, []);

  const getCurrentDate = () => {
    const date = new Date();
    return date.toLocaleDateString('fi-FI', {
      weekday: 'long',
      year: 'numeric',
      month: 'long',
      day: 'numeric'
    });
};
```

Kuvio 17. Mobiilisovelluksen etusivun koodi.

7.3.2 Budjetti

Kuviossa 18 on sovelluksen budjettiosion käyttöliittymä. Se näyttää tulot ja menot sekä laskee näiden perusteella kokonaisbudjetin saldon. Käyttöliittymä sisältää syöttökentät, joihin käyttäjä voi kirjoittaa uuden tulon tai menon kuvauksen ja määrän. Kun käyttäjä on täyttänyt nämä kentät, hän voi lisätä tulon tai menon painamalla vastaavaa painiketta. Tämä toiminto päivittää listan tulot- ja menot-kohdissa sekä laskee automaattisesti uuden saldon.

Käyttäjä voi myös poistaa aiemmin lisätyn tulon tai menon painamalla poistopainiketta kyseisen kohteen vieressä. Tämä toiminto poistaa valitun kohteen listalta ja päivittää kokonaisbudjetin saldon sen mukaisesti. Näin tiekunnan jäsenet voivat ylläpitää ajantasaista ja tarkkaa kirjanpitoa omista tuloistaan ja menoistaan.

Kuvio 18. Mobiilisovelluksen budjettisivun näkymä.

Koodissa on neljä funktiota (ks. kuvio 19.), jotka hallinnoivat tulojen ja menojen lisäämistä sekä poistamista:

”handleAddIncome”-funktio lisää uuden tulon listaan. Funktio tarkistaa ensin, että ”title” ja ”amount” eivät ole tyhjiä. Jos molemmat kentät sisältävät arvoja, uusi tulo lisätään ”incomes”-tilaan. ”setIncomes”-funktiota käytetään päivittämään ”incomes”-tila lisäämällä uusi tulo nykyisten tulojen listaan. Uusi tulo lisätään objektina, jossa on ”title” ja ”amount”, joka muunnetaan merkkijonosta numeroksi ”parseFloat”-funktiolla. Lopuksi ”title” ja ”amount” -kentät tyhjennetään.

”handleAddExpense”-funktio lisää uuden menon listaan. Funktio toimii samalla periaatteella kuin ”handleAddIncome”, mutta se päivittää ”expenses”-tilaa. Tarkistuksen jälkeen uusi meno lisätään ”expenses”-tilaan. ”setExpenses”-funktiota käytetään päivittämään ”expenses”-tila lisäämällä uusi meno nykyisten menojen listaan. ”title” ja ”amount” tyhjennetään lisäyksen jälkeen.

”handleRemoveIncome”- funktio poistaa tulon listasta. Funktio ottaa parametrina poistettavan tulon indeksin. Se luo kopion nykyisestä ”incomes”-tilasta ”updatedIncomes”-muuttujaan. ”splice”-

metodilla poistetaan tulo listasta annetun indeksin kohdalta. ”setIncomes”-funktiota käytetään päivittämään ”incomes”-tila poistamisen jälkeen.

”handleRemoveExpense”-funktio poistaa menon listasta. Funktio toimii samalla periaatteella kuin ”handleRemoveIncome”, mutta se päivittää ”expenses”-tilaa. Parametrina annettu indeksi määrittää poistettavan menon. Luodaan kopio ”expenses”-tilasta ”updatedExpenses”-muuttujaan. ”splice”-metodilla poistetaan meno listasta annetun indeksin kohdalta. ”setExpenses”-funktiota käytetään päivittämään ”expenses”-tila poistamisen jälkeen.

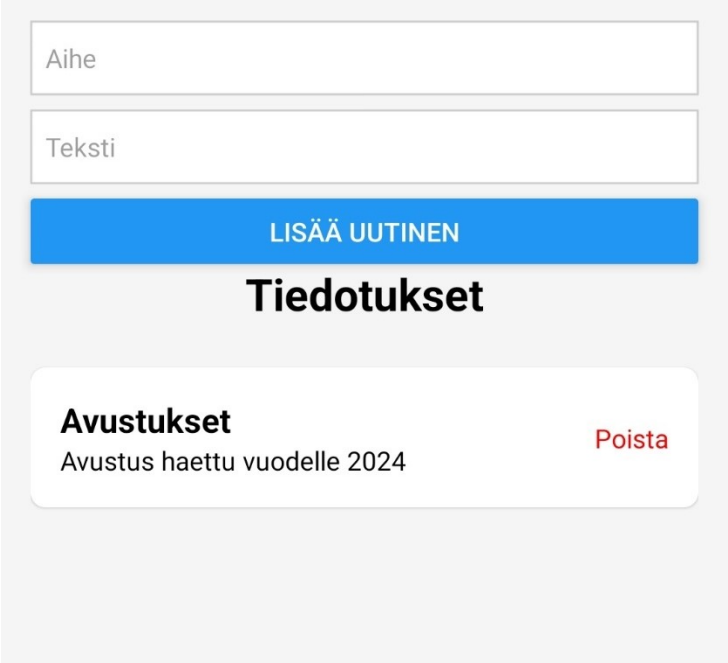
```
const handleAddIncome = () => {
  if (title && amount) {
    setIncomes([...incomes, { title, amount: parseFloat(amount) }]);
    setTitle('');
    setAmount('');
  }
};

const handleAddExpense = () => {
  if (title && amount) {
    setExpenses([...expenses, { title, amount: parseFloat(amount) }]);
    setTitle('');
    setAmount('');
  }
};

const handleRemoveIncome = (index: number) => {
  const updatedIncomes = [...incomes];
  updatedIncomes.splice(index, 1);
  setIncomes(updatedIncomes);
};

const handleRemoveExpense = (index: number) => {
  const updatedExpenses = [...expenses];
  updatedExpenses.splice(index, 1);
  setExpenses(updatedExpenses);
};
```

Kuvio 19. Budjettisivun koodia.

7.3.3 Tiedotus

Kuviossa 20 on sovelluksen tiedotusosion käyttöliittymä. Sovelluksessa on kentät uuden tiedotteen luomiselle sekä lista olemassa olevista tiedotteista. Käyttöliittymässä on kaksi tekstikenttää,

jotka ovat "Aihe" ja "Teksti". Näihin kenttiin käyttäjä voi syöttää uuden tiedotteen otsikon ja sisällön. Kun tiedot ovat syötetty, käyttäjä voi painaa "Lisää uutinen" -painiketta, jolloin uusi tiedote lisätään tiedotteiden listaan. Alaosassa näkyy esimerkki tiedotteesta. Tiedotteella on otsikko sekä käyttäjän kirjoittama teksti. Tiedotteen oikealla puolella on "Poista" -painike, jota painamalla käyttäjä voi poistaa kyseisen tiedotteen.



Kuvio 20. Tiedotussivun näkymä.

Koodissa on kaksi funktiota (ks. kuvio 21.), jotka hallinnoivat tiedotteiden lisäämistä ja poistamista sovelluksessa. ”addNews”-funktio vastaa uuden tiedotteen lisäämisestä. Se tarkistaa ensin, että sekä otsikko ”title” että sisältö ”story” on täytetty. Jos ne ovat, sovellus luo uuden tiedotteen objektina, joka sisältää yksilöllisen id:n, otsikon ja sisällön. Tämä uusi tiedote lisätään nykyisten tiedotteiden ”news” listaan, joka päivitetään ”setNews”-funktiolla. Tämän jälkeen päivitetty lista tallennetaan ”saveNews”-funktiolla, jonka jälkeen syöttökentät tyhjennetään.

”deleteNews”-funktio puolestaan poistaa tiedotteen listalta sen id:n perusteella. Se suodattaa pois poistettavan tiedotteen käyttämällä ”filter”-metodia ja päivittää listan ”setNews”-funktiolla. Päivitetty lista tallennetaan myös ”saveNews”-funktiolla. Näiden toimintojen avulla käyttäjät voivat lisätä ja poistaa tiedotteita helposti.

```
const addNews = () => {
  if (title && story) {
    const newNews = { id: Date.now().toString(), title, story };
    const updatedNews = [...news, newNews];
    setNews(updatedNews);
    saveNews(updatedNews);
    setTitle('');
    setStory('');
  }
};

const deleteNews = (id: string) => {
  const updatedNews = news.filter((item) => item.id !== id);
  setNews(updatedNews);
  saveNews(updatedNews);
};
```

Kuvio 21. Tiedotussivun uutisten lisäämisen ja poistamisen toteuttava koodi.

7.3.4 Tiekartta

Kuviossa 22 on mobiilisovelluksen tiekarttaosio, johon on punaisella korostettu tiekunnan ylläpitämä tie. Kartan yläosassa on sininen painike, jossa lukee "MENE TÄSTÄ TAKAISIN KOTITIELLE". Tämä painike palauttaa käyttäjän takaisin karttanäkymän omalle tielle. Käyttäjä voi navigoida helposti takaisin alkuun painamalla tätä painiketta, mikäli käy katsomassa muita teitä kartalla.

Kuvio 22. Tiekartta sivun näkymä.

Koodissa käytetään ”react-native-maps”- ja ”expo-location”-kirjastoja, jolla näytetään kartan ja käyttäjän sijainti. Sovellus pyytää lupaa paikannustietojen käyttöön ja saa käyttäjän nykyisen sijainnin. Kun sijainti on saatu, se tallennetaan tilaan. Jos lupaa ei myönnetä, näytetään virheilmoitus.

Sovellus määrittää kartalle alkuperäisen alueen (ks. kuvio 23.) ”initialRegion”, joka keskittyy tietyille koordinaateille. Sovellus piirtää punaisen reitin tiettyjen koordinaattien perusteella.

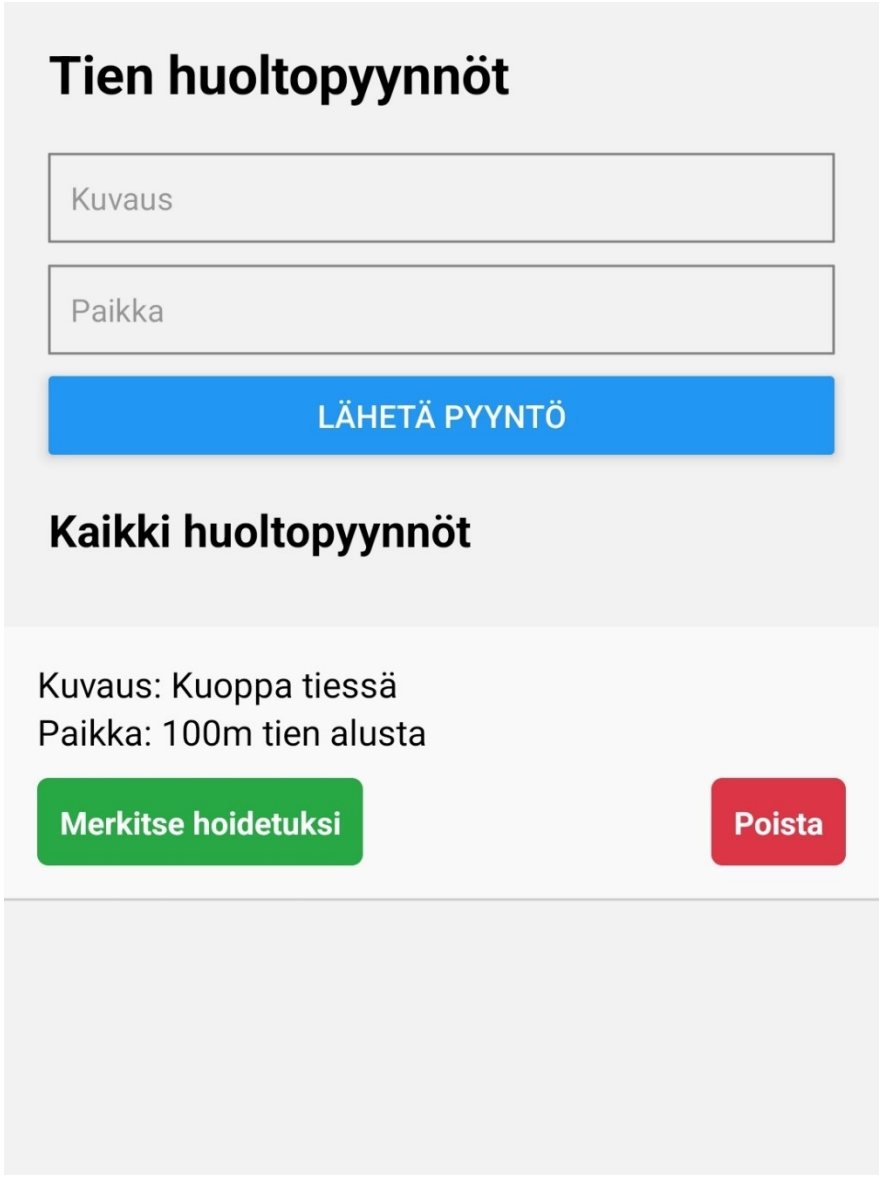
”goToHighlightedRoad”-funktio vastaa karttanäkymän siirtämisestä tiettyyn alueeseen. Funktio käyttää kartan viittausta ”mapRef”-komponentin ”animateToRegion”-metodia. Tarkoituksena on animoida karttanäkymä käyttäjän painaman painikkeen vaikutuksesta, jotta se keskittyy tiettyyn alueeseen, tässä tapauksessa alkuperäiseen määriteltyyn alueeseen ”initialRegion”.

```
const roadCoordinates = [
  { latitude: 61.78979024991171, longitude: 22.24922355110472 },
  { latitude: 61.79118903769705, longitude: 22.247022222331687 },
  { latitude: 61.79341502600575, longitude: 22.246895916582414},
  { latitude: 61.794088760795, longitude: 22.245326116555745},
  { latitude: 61.794080232598816, longitude: 22.242673695821026},
  { latitude: 61.79430196492995, longitude: 22.242006079717726},
  { latitude: 61.79428490865357, longitude: 22.241753468219184}
];

const initialRegion: Region = {
  latitude: roadCoordinates[0].latitude,
  longitude: roadCoordinates[0].longitude,
  latitudeDelta: 0.01,
  longitudeDelta: 0.01,
};

const goToHighlightedRoad = () => {
  if (mapRef.current) {
    mapRef.current.animateToRegion(initialRegion, 1000);
  }
};
```

Kuvio 23. Tien karttanäkymän toiminnan koodi.

7.3.5 Huoltopyynnöt

Tien huoltopyyntö-välilehdessä on kaksi tekstikenttää, jotka on merkitty "Kuvaus" ja "Paikka" (ks. kuvio 24.). Käyttäjä voi syöttää näihin kenttiin huoltopyynnön kuvauksen ja sen paikan. Näiden kenttien alla on "Lähetä pyyntö" -painike, jota painamalla käyttäjä voi lähettää huoltopyynnön. Tämä lisäys päivittyy luetteloon alla. Alapuolella näkyy lista jo tehdyistä huoltopyynnöistä. Huoltopyynnön alla on kaksi painiketta: ”Merkitse hoidetuksi" ja "Poista". Ensimmäinen painike on tarkoitettu merkkaamaan huoltopyynnön hoidetuksi, jolloin se merkitään hoidetuksi. Toinen painike, "Poista", on tarkoitettu poistamaan huoltopyyntö listalta kokonaan.



Kuvio 24. Tien huoltopyynnöt sivun näkymä.

Tässä koodissa on kolme funktiota (ks. kuvio 25.), jotka hallinnoivat huoltopyyntöjä React Native -sovelluksessa. Näitä funktioita käytetään lisäämään uusia huoltopyyntöjä, poistamaan olemassa olevia pyyntöjä ja merkitsemään pyynnöt hoidetuiksi.

”handleAddRequest”-funktio vastaa uuden huoltopyynnön lisäämisestä. Tämä funktio on määritelty käyttämällä ”useCallback”-hookia, mikä parantaa suorituskykyä muistamalla tämän funktion ja estää komponentin renderöimisen uudelleen, mikäli parametrit eivät ole muuttuneet. Ensimmäinen tarkistus varmistaa, että sekä kuvaus ”descriptionRef.current” että sijainti ”location-Ref.current” ovat täytetty. Jos jokin kentistä on tyhjä, näytetään virheilmoitus.

Uusi pyyntö lisätään olemassa olevien pyyntöjen listaan ”setRequests”-funktion avulla, joka päivittää tilan. Lopuksi tyhjennetään syöttökentät, jotta ne ovat valmiit seuraavaa syöttöä varten.

”handleRemoveRequest”-funktio poistaa huoltopyynnön listasta ”id”:n perusteella. Tämä funktio ottaa parametrina poistettavan pyynnön tunnisteen ja käyttää ”setRequests”-funktiota suodattaakseen pois poistettavan pyynnön. Tämä tapahtuu käyttämällä ”filter”-metodia, joka palauttaa kaikki muut pyynnöt paitsi poistettavan.

”handleSolveRequest”-funktio merkitsee huoltopyynnön ratkaistuksi sen ”id”:n perusteella. Tämä funktio käyttää ”setRequests”-funktiota ja ”map”-metodia päivittääkseen tietyn pyynnön tilan. Se tarkistaa jokaisen pyynnön ja jos pyynnön id vastaa annettua ”id”:tä, se päivittää kyseisen pyynnön ”solved”-arvon ”true”:ksi. Muut pyynnöt säilyvät ennallaan.

```
const handleAddRequest = useCallback(() => {
  if (!descriptionRef.current || !locationRef.current) {
    Alert.alert('Error', 'Please fill in all fields.');
    return;
  }

  const newRequest: MaintenanceRequest = {
    id: requests.length > 0 ? requests[requests.length - 1].id + 1 : 1,
    description: descriptionRef.current,
    location: locationRef.current,
    solved: false,
  };

  setRequests(prevRequests => [...prevRequests, newRequest]);
  descriptionRef.current = '';
  locationRef.current = '';
}, [requests]);

const handleRemoveRequest = (id: number) => {
  setRequests(prevRequests => prevRequests.filter(request => request.id !== id));
};

const handleSolveRequest = (id: number) => {
  setRequests(prevRequests => prevRequests.map(request => request.id === id ? { ...request, solved: true } : request));
};
```

Kuvio 25. Tien huoltopyynnön sivuston koodi.

7.4 Sovelluksen jatkokehitys

7.4.1 Tyylit

Sovelluksen käyttöliittymän yhtenäistämiseksi otetaan käyttöön keskitetty tyylien hallinta. Tämä helpottaa visuaalisten elementtien kuten nappien, tekstikenttien, taustavärien ja fonttien määritystä ja hallinnointia yhdestä tyylitiedostosta.

7.4.2 Huoltopyyntöjen, tiedotuksien ja budjetin kohteiden poiston varmistus

Lisätään vahvistuspyyntö kaikille poisto-operaatioille, jotka koskevat huoltopyyntöjä, tiedotuksia ja budjetin kohteita. Kun käyttäjä yrittää poistaa jonkin näistä kohteista, sovellus näyttää varmistusviestin, jossa käyttäjää pyydetään vahvistamaan poistotoimenpide. Tällä estetään poistot, jotka tehdään tahattomasti.

7.4.3 Budjetti tallennus vuosittain Firebase-palvelun avulla

Budjettitietojen tallennusta kehitetään siten, että jokaisen vuoden budjetti tallennetaan erikseen Firebase-tietokantaan. Tällä mahdollistetaan budjettitietojen arkistoinnin ja hakemisen eri vuosilta, mikäli tarvetta.

7.4.4 Budjettiin päivämääräkenttä

Jokaiseen budjettikohteeseen lisätään kenttä, joka tallentaa kohteen lisäyspäivämäärän. Päivämäärä luodaan automaattisesti, kun käyttäjä luo uuden budjettikohteen. Tämä auttaa seuraamaan milloin kohteita on lisätty.

7.4.5 Huoltopyyntöihin päivämäärä milloin lisätty sekä ratkaistu

Huoltopyyntöihin lisätään kaksi uutta kenttää: lisäyspäivämäärä ja ratkaisupäivämäärä. Lisäyspäivämäärä luodaan automaattisesti, kun uusi huoltopyyntö tehdään, ja ratkaisupäivämäärä tallennetaan, kun huoltopyyntö merkitään ratkaistuksi. Tämä mahdollistaa huoltopyyntöjen elinkaaren seurannan.

7.4.6 Kirjautuminen

Käyttäjäkirjautuminen integroidaan Clerk-palvelun avulla. Clerk tarjoaa turvallisen ja helppokäyttöisen tavan hallinnoida käyttäjien kirjautumista ja rekisteröitymistä. Clerk-integraatio varmistaa käyttäjätietojen turvallisuuden ja yksinkertaistaa kirjautumisprosessia, parantaen samalla sovelluksen käyttäjäkokemusta.

7.4.7 Lokien katsomismahdollisuus

Sovellukseen lisätään lokien hallintatoiminto, joka tallentaa käyttäjien tekemät toimenpiteet. Näitä lokitietoja voidaan käyttää jälkikäteen tarkistamaan, kuka on tehnyt mitäkin muutoksia sovelluksessa. Tämä parantaa sovelluksen läpinäkyvyyttä ja helpottaa ongelmatilanteiden selvittämistä.

8 Tulokset

Opinnäytetyö perustui alla oleviin tutkimuskysymyksiin. Käydään niiden kautta läpi saavutettuja tuloksia.

1. Minkälaisia asioita tulisi ottaa huomioon mobiilisovelluksen suunnittelussa?

Suunnittelussa on tärkeää ottaa huomioon useita keskeisiä tekijöitä, jotka vaikuttavat sovelluksen toimivuuteen ja käyttäjäkokemukseen. Suunnittelun tulee olla käyttäjäkeskeistä, sillä käytettävyys ja käyttäjäkokemus ovat erittäin tärkeitä varmistaen sovelluksen helppokäyttöisyyden ja intuitiivisuuden. Käyttöliittymän tulee olla selkeä ja looginen. Sovelluksen toimivuus eri laitteilla kannattaa ottaa huomioon, koska markkinoilla on monia eri laitteita eri käyttöjärjestelmillä. Tietoturva on myös tärkeä osa-alue, jotta käyttäjien tiedot ovat suojassa. Sovelluksen suorituskyvyn kannattaa kiinnittää huomiota, jotta käyttäjäkokemus pysyy miellyttävänä, mikä edellyttää optimoitua koodia ja tehokasta resurssienhallintaa. Lisäksi sovelluksen tulee olla saavutettava kaikille käyttäjille, mukaan lukien ne, joilla on erilaisia toimintarajoitteita. Käyttäjäpalautteen kerääminen ja sen pohjalta tehtävät parannukset ovat oleellisia. Iteratiivinen suunnittelu, jossa sovellusta kehitetään jatkuvasti käyttäjäpalautteen perusteella, auttaa varmistamaan, että sovellus vastaa käyttäjien tarpeisiin ja odotuksiin mahdollisimman hyvin.

2. Miten React Native -ohjelmointikehystä voidaan hyödyntää tiekuntien tarpeeseen suunnatun mobiilisovelluksen kehittämisessä?

React Native -ohjelmointikehyksen hyödyntäminen tiekuntien tarpeisiin suunnatun mobiilisovelluksen kehittämisessä tarjoaa monia etuja. Ensinnäkin, React Native mahdollistaa sovellusten kehittämisen sekä iOS- että Android-alustoille yhdellä koodipohjalla, mikä säästää aikaa ja resursseja. React Native tarjoaa myös laajan valikoiman valmiita komponentteja ja kirjastoja, mikä helpottaa eri toiminnallisuuksien, kuten push-ilmoitusten, karttojen ja tiedotusten, toteutusta. Suuri kehittäjäyhteisö tarjoaa runsaasti tukea, dokumentaatiota ja valmiita ratkaisuja, mikä nopeuttaa ongelmien ratkaisua ja uusien ominaisuuksien implementointia. Kaiken kaikkiaan React Native tarjoaa tiekunnille kattavan, kustannustehokkaan ja joustavan ratkaisun mobiilisovellusten kehittämiseen, joka vastaa sekä nykyisiin että tuleviin tarpeisiin.

3. Millaisia teknisiä ja suunnittelullisia haasteita kohdataan mobiilisovelluksen kehittämisessä React Nativella, ja miten näihin haasteisiin vastataan?

React Native -sovelluksen kehittämisessä voi kohdata useita teknisiä ja suunnittelullisia haasteita. Suorituskykyongelmat ovat yksi yleisimmistä, monimutkaisissa sovelluksissa saattaa esiintyä suori-

tuskykyongelmia, jotka voidaan ratkaista optimoimalla koodia, vähentämällä renderöintien määrää ja hyödyntämällä natiiviin koodiin siirtymistä kriittisissä osioissa. Yhteensopivuusongelmat eri laitteiden ja käyttöjärjestelmäversioiden välillä voivat myös olla haasteellisia, ja tämä vaatii kattavaa testausta eri laitteilla ja versioilla sekä mahdollisesti erillisiä ratkaisuja tietyille alustoille. Kolmannen osapuolen komponenttien ja kirjastojen käyttö voi aiheuttaa riippuvuusongelmia ja ylläpitokysymyksiä. Tämä voidaan ratkaista valitsemalla aktiivisesti ylläpidettyjä ja yhteisön tukemia kirjastoja sekä säännöllisellä riippuvuuksien päivittämisellä. Käyttäjäystävällisen käyttöliittymän suunnittelu vaatii huolellista suunnittelua ja käyttäjäpalautteen hyödyntämistä, ja prototyyppien testaaminen käyttäjillä ja iteratiivinen suunnittelu ovat avainasemassa. Näihin haasteisiin voidaan vastata huolellisella suunnittelulla, testaamisella ja jatkuvalla kehityksellä sekä hyödyntämällä React Nativen tarjoamia työkaluja ja yhteisön tukea.

9 Pohdinta

Tässä opinnäytetyössä tutkittiin mobiilisovellusten kehittämistä React Native -kehitysalustan avulla, erityisesti keskittyen tiekuntien tarpeisiin. Työn tavoitteena oli luoda tiekunnille sovellus, joka helpottaa heidän hallinnollisia ja operatiivisia tehtäviään. Tämä tutkimus ja kehitysprosessi tarjosivat useita oivalluksia ja oppimiskokemuksia, jotka ovat merkityksellisiä sekä sovelluskehittäjille että tiekunnille.

Yksi keskeisistä asioista oli käyttäjäkeskeisen suunnittelun tärkeys. Mobiilisovelluksen menestys riippuu pitkälti sen käytettävyydestä ja siitä, kuinka hyvin se vastaa käyttäjiensä tarpeisiin. Tiekuntien jäsenet eivät välttämättä ole teknisesti suuntautuneita, joten sovelluksen helppokäyttöisyys on tärkeää. Käyttäjäpalautteen kerääminen on erittäin hyödyllinen tapa kehittää sovelluksen toiminnallisuuksia ja parantaa käytettävyyttä.

React Native -kehitysalustan valinta osoittautui oikeaksi ratkaisuksi, sillä sen avulla pystyttiin kehittämään monialustainen sovellus tehokkaasti. Yhden koodipohjan käyttäminen sekä iOS- että Android-sovellusten luomiseen säästi merkittävästi aikaa ja resursseja. React Native -yhteisön tarjoamat valmiit komponentit ja kirjastot nopeuttivat kehitystyötä ja mahdollistivat monipuolisten ominaisuuksien, kuten karttojen helpon integroinnin.

Expo-työkalun käyttö yksinkertaisti kehitysprosessia ja mahdollisti sujuvamman julkaisun sekä ylläpidon. Expo tarjosi valmiita ratkaisuja moniin yleisiin kehitysongelmiin, mikä teki sovelluksen kehittämisestä nopeampaa ja vähemmän monimutkaista.

Kehitysprosessin aikana saatiin arvokasta tietoa siitä, miten teknologia voi tukea ja tehostaa yhteisöjen ja organisaatioiden, kuten tiekuntien, hallinnollisia toimintoja. Mobiilisovellus, joka on suunniteltu oikein ja vastaa käyttäjien tarpeisiin, voi merkittävästi parantaa tiedonvaihtoa, päätöksenteon tehokkuutta ja yhteisön jäsenten välistä yhteistyötä.

Tulevaisuuden kehityssuunnitelmiin kuuluu sovelluksen jatkuva parantaminen käyttäjäpalautteen perusteella sekä uusien ominaisuuksien lisääminen vastaamaan tiekuntien muuttuvia tarpeita. Lisäksi sovelluksen skaalautuvuus ja ylläpidettävyys varmistavat, että se pysyy tarpeellisena ja hyödyllisenä pitkään.

Kokonaisuudessaan tämä projekti osoitti, että React Native on tehokas ja joustava työkalu mobiilisovellusten kehittämiseen, ja että hyvin suunniteltu mobiilisovellus voi tarjota merkittäviä hyötyjä yhteisöjen hallinnollisiin tarpeisiin. Tiekuntien kaltaiset pienet organisaatiot voivat hyödyntää teknologiaa parantaakseen toimintaansa ja viestintäänsä, mikä luo pohjan entistä tehokkaammalle ja läpinäkyvämmälle hallinnolle.

Lähteet

Blair, I. N.d. A Complete Guide to Mobile App Development. https://buildfire.com/understanding-mobile-app-development-lifecycle/

Braham, A., Buendía, F., Khemaja, M., & Gargouri, F. 2019. Generation of Adaptive Mobile Applications Based on Design Patterns for User Interfaces. https://www.mdpi.com/2504-3900/31/1/19

Braun, S., Hess, S., Lenhart, T., Magin, D., & Naab, M. 2015. Mobile Business Applications: Designing User Interface and Architecture. 2015 2nd ACM International Conference on Mobile Software Engineering and Systems. https://ieeexplore.ieee.org/document/7283043

Chittaro, L. 2011. Designing visual user interfaces for mobile applications. https://dl.acm.org/doi/10.1145/1996461.1996550

Gentile, A. 2023. Basics of React Native: Three Types of Components for Your Application. https://angelogentileiii.medium.com/basics-of-react-native-three-types-of-components-for-your-application-bf1c239697ad

GilPress. 2024. How Many People Own Smartphones? (2024-2029) https://whatsthebigdata.com/smartphone-stats/

HTG Staff. 2016. What Is GitHub, and What Is It Used For?. https://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/

Hulkkonen, T. 2022. Miksi aina ei kannata rakentaa Kubernetesin päälle? Esittelyssä Google Firebase. https://www.gappsgroup.com/fi/blogi/miksi-aina-ei-kannata-rakentaa-kubernetesin-paalle-esittelyssa-google-firebase/

Jiang, Z., Yin, H., Luo, Y., Gong, J., Yang, Y., & Lin, M. 2019. Quantitative Analysis of Mobile Application User Interface Design. 2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC). https://ieeexplore.ieee.org/document/8958722

Kumar, R. 2023. History of React JS. Viitattu 10.2024. https://www.linkedin.com/pulse/history-react-js-ranjith-kumar/

Käyttäjätarinat ohjelmistokehityksessä. 2023 https://www.haltu.fi/blogi/kayttajatarinat

Mitä tarkoittaa Frontend ja BackEnd ohjelmistokehityksessä?. 2021. Artikkeli Mectalent Oy sivustolla. Julkaistu 29.9.2021. Viitattu 1.3.2024. https://www.mectalent.com/fi/ajankohtaista/mita-tarkoittaa-frontend-ja-back-end

mobileLive. 2023. Backend for Frontend Basics: A Comprehensive Guide. https://mobilelive.medium.com/backend-for-frontend-basics-a-comprehensive-guide-37768062e55a

Nilsson, E. G. 2009. Design patterns for user interface for mobile applications. Adv. Eng. Softw., 40, 1318-1328. https://www.sciencedirect.com/science/article/abs/pii/S0965997809000428?via%3Dihub

React Ecosystem in 2024. 2024. Viitattu 14.4.2024https://refine.dev/blog/react-js-ecosystem-in-2024/#introduction

sikiru. 2024, Why I'm Learning React in 2024. https://medium.com/@sikirus81/why-im-learning-react-in-2024-a2567a179def

Softworth Solutions Private Limited. 2023. Expo vs React Native CLI. https://medium.com/@softworthsolutionspvtltd/expo-vs-react-native-cli-7e47c7630039

Ström, C. 2019. React pähkinänkuoressa. Viitattu 10.2.2024. https://joinex.fi/react-pahkinankuoressa/

The Basics of App Development: Step-by-Step Guide. 2024. Artikkeli Kissflow Inc sivustolla. Julkaistu 18.3.2024. Viitattu 1.4.2024. https://kissflow.com/application-development/app-development-basics-explanied/

Tiekuntien ja yhteisten alueiden yhteystiedot voi ilmoittaa verkon kautta. 2023. Tieyhdistyksen verkkosivut. Viitattu 24.2.2024. https://www.tieyhdistys.fi/yksityistiet/ajankohtaista-yksityis-tieasiaa/

Visual Studio Code Getting Started. N.d. Visual Studio Code verkkosivut. https://code.visualstu-dio.com/docs