Elisa Rajaniemi

Kaupunkipyörien vuokrausominaisuus mobiilisovellukseen

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Insinöörityö

19.3.2018

Metropolia

Tekijä Otsikko Sivumäärä Aika	Elisa Rajaniemi Kaupunkipyörien vuokrausominaisuus mobiilisovellukseen 34 sivua 19.3.2018
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	tieto- ja viestintätekniikka
Ammatillinen pääaine	Mobile Solutions
Ohjaaja	lehtori Peter Hjort

Insinöörityön tarkoituksena oli kehittää matkustusmobiilisovellukseen kaupunkipyörien vuokraamisen mahdollistava ominaisuus. Ominaisuus kehitettiin Android-mobiilisovellukseen. Ominaisuuden tarkoitus oli tarjota käyttäjille mahdollisuus vuokrata kaupunkipyöriä ja seurata pyörävarauksien tilaa sovelluksesta käsin.

Kaupunkipyöräominaisuus oli tarkoitus julkaista ensin Englannin Birminghamissa, ja sen jälkeen sitä oli tarkoitus pystyä käyttämään myös muiden maiden ja kaupunkien kaupunkipyörien vuokraamisessa. Insinöörityössä ominaisuus kehitettiin tukemaan Birminghamissa kaupunkipyöräpalveluntarjoajan pyörien vuokrausta.

Birminghamissa ei ollut sovellusta kehitettäessä vielä kaupunkipyöriä, eikä oikeaa dataa pyöristä tai pyöräasemista ollut saatavilla. Tämän takia jouduttiin tekemään olettamuksia siitä, miten kaupunkipyörien vuokraaminen tulisi toimimaan sovelluksesta käsin. Ennen kuin ohjelmointirajapinnan testiympäristö oli valmis, ominaisuuden kehityksessä käytettiin kovakoodattua testidataa. Insinöörityöprojektin edetessä saatiin lisätietoa käytössä olevista kaupunkipyöristä ja pyöräasemista, minkä perusteella tehtiin muutoksia ominaisuuden ulkoasuun ja logiikkaan.

Ominaisuus kehitettiin Android Studio -kehitysympäristössä käyttäen ohjelmointikielenä Javaa RxJava 2.0 -kirjastoa hyödyntäen.

Insinöörityöprojektin päätyttyä kaupunkipyörien vuokraus toimi odotetulla tavalla testiympäristössä ja oli valmis testaukseen tuotannossa Englannissa.

Avainsanat	Android, mobiilisovellus, kaupunkipyöräpalvelut

Author Title Number of Pages Date	Elisa Rajaniemi City bike rental feature for a mobile application 34 pages 19 March 2018
Degree	Bachelor of Engineering
Degree Programme	Information and Communication Technology
Professional Major	Mobile Solutions
Instructor	Peter Hjort, Senior lecturer

The purpose of this thesis was to develop a feature to book city bikes with a mobile application. The thesis focuses on developing the feature for an Android application. The purpose of the feature was to let users book city bikes and follow the state of the bookings with the application.

The city bike feature was planned to be published first in Birmingham in England. After that the feature was supposed to support city bike services in other cities and countries. In this thesis the feature was developed to support city bike rental in Birmingham.

There were no city bikes in Birmingham while creating the feature so there was no real data of the bikes or the bike stations. The first versions of the feature were made based on assumptions of how the city bike rental would work in Birmingham. Hardcoded test data was used to create the basic structure of the feature before getting data from API's test environment. While creating the feature more information about the bikes and bike stations started to be available. With new information changes were done to the design and the logic of the feature.

The feature was developed in Android Studio development environment with Java programming language supported by RxJava 2.0 library.

At the end of the project the feature works in test environment as intended and is ready to be tested in production in England.

Keywords	Android, mobile application, city bike services

Sisällys

Lyhenteet

1 Johdanto 1

2 Liikkuminen palveluna (MaaS) 2

2.1 MaaS Global ja Whim-sovellus 2
2.2 Muita MaaS-palveluja 3

3 Kaupunkipyöräpalvelut 4

3.1 Yleistä 4
3.2 Helsingin ja Milton Keynesin kaupunkipyöräpalvelut 5

4 Kaupunkipyöräominaisuuden kehitys 7

4.1 Projektin eteneminen 7
4.2 Käytetyt kirjastot ja mallit 9
4.3 Apuna käytetyt ohjelmat 16
4.4 Yhteys ohjelmointirajapintoihin 19
4.5 Testaus 20
4.6 Valmis ominaisuus 21
4.7 Ulkoasu 22
4.8 Ominaisuuden yhteensopivuus muiden kaupunkipyöräpalveluiden kanssa 29

5 Yhteenveto 30

Lähteet 32

Lyhenteet ja käsitteet

MaaS	Mobility as a Service. Konsepti, jolla pyritään luomaan käyttäjälle mahdollisuus käyttää liikkumispalveluja ekologisesti ja tehokkaasti teknologian avulla.
MVP	Model View Presenter. Sovelluksen kehitysmalli, jossa logiikka ja ulkoasun luominen erotetaan toisistaan.
REST	REpresentational State Transfer. Ohjelma-arkkitehtuurimalli ohjelmistorajapinnoille, joka määrittelee millaisia operaatioita palvelimelle voi tehdä.
DAO	Data Access Object. Rajapintaluokka, joka määrittelee metodimallit, joilla tietokantaan ollaan yhteydessä.
HSL	Helsingin seudun liikenne. Kuntayhtymä, joka vastaa Helsingin seudun joukkoliikennejärjestelmästä.

1 Johdanto

Kaupunkipyörät ovat ekologinen, vaivaton ja usein nopea tapa liikkua kaupungeissa. Kaupunkipyöristä on tullut osa monien kaupunkien katukuvaa, ja teknologian kehittyessä pyörien vuokraamisesta on tullut mutkattomampaa. Monissa kaupungeissa pyöriä voi vuokrata kaupunkipyöräpalveluntarjoajan mobiilisovelluksen kautta. Käyttäjillä saattaa olla monia eri mobiilisovelluksia eri liikkumispalveluihin, kuten taksin tilaamiseen ja julkisen liikenteen matkalippujen ostamiseen.

Insinöörityön tavoitteena oli kehittää työn asiakasyrityksen, MaaS Globalin, eri liikkumispalveluja yhdistävään Whim-mobiilisovellukseen kaupunkipyörien vuokraamisen mahdollistava ominaisuus. Helmikuussa 2018 Whimin kautta pystyi ostamaan julkisen liikenteen lippuja, tilaamaan takseja ja vuokraamaan autoja. Kaupunkipyöräominaisuus oli tarkoitus julkaista keväällä 2018 Englannissa Birminghamissa tarjoamalla käyttäjille käyttöön Nextbike-kaupunkipyöriä. Myöhemmin ominaisuuden tulisi tukea myös Helsingin HSL-kaupunkipyöriä sekä muiden kaupunkien ja maiden kaupunkipyöriä, joihin Whimin palvelut tulevaisuudessa laajentuvat.

Insinöörityöraportissa esitellään kaupunkipyöräominaisuuden lisääminen Whimin Android-sovellukseen. Raportissa käydään läpi projektin kehityksen vaiheita, projektissa käytettyjä teknologioita sekä työn tulokset.

2 Liikkuminen palveluna (MaaS)

2.1 MaaS Global ja Whim-sovellus

Kaupungistuminen aiheuttaa erilaisia ongelmia katukuvassa. Iso ongelma on yksityisautoilu, joka saastuttaa ja aiheuttaa liikenneruuhkia. Lisäksi isoissa kaupungeissa yksityisautoilu ei aina ole kannattavaa sen hitauden sekä usein kalliiden ja vähäisten pysäköintipaikkojen takia.

MaaS (Mobility as a Service eli liikkuminen palveluna) on konsepti, joka pyrkii vastaamaan kaupungistumisen liikenteeseen liittyviin haasteisiin tarjoamalla palveluita, jotka ovat kannattavia sekä ympäristölle että käyttäjälle. MaaS-palvelut pyrkivät tarjoamaan teknologian avulla monipuolisesti erilaisia liikkumispalveluja, kuten julkisen liikenteen palveluja, takseja, kaupunkipyöriä sekä autovuokrausta auton omistamisen vaihtoehdoksi. MaaS-palveluiksi lasketaan kaikki ne palvelut, jotka tarjoavat teknologian avulla ratkaisuja kaupungistumisen liikenteeseen liittyviin haasteisiin. (1.)

MaaS-palveluihin kuuluvat esimerkiksi myös ostosten ja ravintola-annosten kuljetuspalvelut, sillä niiden avulla jokaisen kuluttajan ei tarvitse kulkea palveluiden luokse, mikä vähentää liikennettä (1).

MaaS Global on suomalainen vuonna 2015 perustettu yritys, jonka tavoite on tarjota Whim-sovelluksensa avulla vaihtoehto auton omistamiselle. MaaS Global pyrkii vähentämään käyttäjien päivittäiseen liikkumiseen liittyviä haasteita ja samalla auttamaan käyttäjiä valitsemaan ekologisempia kulkemisvaihtoehtoja. Jokaista uutta ominaisuutta Whimiin luotaessa mietitään ensin, poistaako se käyttäjän liikkumistottumuksista turhaa vaivaa. MaaS Globalin tavoitteena on tarjota käyttäjille helposti ja edullisesti toimivia liikkumispalveluja, jotta mahdollisimman monen käyttäjän olisi madollista vähentää yksityisautoilua tai luopua siitä kokonaan. (2.)

MaaS Globalin Whim-sovellus on maailman ensimmäinen kansainvälisiä MaaS-liikkumispalveluja tarjoava mobiilisovellus. Whimin iOS-sovellus julkaistiin syksyllä 2016 ja Android-sovellus syksyllä 2017. Helmikuussa 2018 Whim tarjosi palveluita Helsingissä ja Birminghamissa, mutta sen on tulevaisuudessa tarkoitus tarjota palveluitaan ympäri maailmaa. (2.)

Whim tarjoaa käyttäjille erilaisia kuukausitilauksia. Ilmaisen kuukausitilauksen avulla voi ostaa yksittäisiä matkalippuja, tilata takseja ja vuokrata autoja. Maksullisiin tilauksiin kuuluu erilaisia etuja, kuten julkisen liikenteen matkalippuja sekä edullisia tai ilmaisia vuokra-autoja ja taksimatkoja.

Rekisteröityessään palveluun käyttäjä valitsee alueen, jonka kuukausitilauksia hän haluaa käyttää. Kuukausitilaukset ovat sidonnaisia paikallisiin palveluihin, joten eri maissa ja kaupungeissa on eri kuukausitilausvaihtoehdot. Esimerkiksi Helsingin alueeksi valinneet käyttäjät saavat maksullisissa tilauksissa käyttää rajattomasti HSL-matkalippuja. Tilauksen alueesta riippumatta käyttäjät voivat käyttää Whimin palveluja valitsemansa alueen ulkopuolella. Esimerkiksi Helsinkiin rekisteröitynyt käyttäjä voi tilata Whimillä takseja ja ostaa matkalippuja Birminghamissa. Whim seuraa käyttäjän sijaintia tietääkseen, minkä alueen palveluja käyttäjälle tulee milläkin hetkellä tarjota.

2.2 Muita MaaS-palveluja

Uber

Uber on mobiilisovellus, jonka avulla käyttäjät voivat tilata autokyytejä. Sovelluksen kartalta voi seurata tilatun kyydin lähestymistä. Maksu matkoista menee suoraan käyttäjän palveluun rekisteröidyltä kortilta tai sen voi jakaa muiden matkustajien kanssa sovelluksen kautta. Uberin suosio perustuu sen helppouteen ja usein takseja halvempaan hintaan. Uber toimii yli 600 kaupungissa ympäri maailmaa. (3.)

Uber-kuljettajat ovat tavallisia ihmisiä, jotka käyttävät kuljettamiseen usein omaa autoaan. Koska Uber-kuljettajaksi pääsemiseen ei vaadita ammattiajolupaa, Uberin palvelut eivät vielä ole laillisia Suomessa. (4.)

Wolt

Wolt-sovelluksen avulla voi tilata ruokaa ravintoloista ja kahviloista, joiden kanssa Woltilla on sopimus. Woltilla on oma kuljetuspalvelu, joka kuljettaa ruoka-annoksia ravintolan lähialueelle. Woltia voi käyttää mobiilisovelluksena tai selaimesta käsin. (5.)

3 Kaupunkipyöräpalvelut

3.1 Yleistä

Kaupunkipyöräpalveluja on tarjolla monissa eri kaupungeissa ympäri maailmaa (kuva 1). Yleensä kaupungit tukevat kaupunkipyöräpalvelujaan, mutta usein palvelun rahoitusta saadaan myös sponsoreilta, mainostuloista ja käyttömaksuista. (6.)

Kuva 1. Toiminnassa olevia, korjauksen tai kehityksen alla olevia sekä poistuneita kaupunkipyöräpalveluita maailmalla (6).

Pyörien vuokraustavat, pyörät ja pyörätelineet vaihtelevat palveluntarjoajien välillä. Esimerkiksi Nextbike-kaupunkipyörät tulee palauttaa aina Nextbike-pyöräasemalle, kun taas Mobike-kaupunkipyörän voi jättää mihin tahansa julkiseen pyörätelineeseen (7; 8). Nextbike- ja Mobike-kaupunkipyöräpalveluntarjoajat toimivat eri kaupungeissa ympäri maailmaa, mutta on myös kaupunkeja, kuten Helsinki, joilla on omat kaupunkipyöränsä.

Usein kaupunkipyörinä käytetään älypyöriä, joiden sijaintia ja käyttöä pystytään seuraamaan palvelimelta käsin. Erityisesti kaupungeissa, joissa pyörän voi jättää mihin vain julkiseen pyörätelineeseen, sijaintitieto on tärkeä. Vapaat pyörät näytetään kaupunkipyöräsovelluksessa kartalla, jotta käyttäjät löytävät ne uudestaan käyttöönsä (7). Sijaintitietoa käytetään hyväksi myös tiedon saamiseksi siitä, onko pyörä palautettu pyöräasemalle, jos pyörä tulee pysäköidä sinne tai sen läheisyyteen. Joissain kaupungeissa, kuten Varsovassa, käytetään älypyörätelineitä, jotka välittävät palvelimelle tiedon vapaiden pyörien ja telinepaikkojen määrästä (8).

Yleensä pyörien lukot avataan koodilla, jonka syöttämisen jälkeen palvelin saa tiedon matkan alkamisesta. Kun pyörä on palautettu asianmukaisesti, voidaan matkan kokonaishinta laskea ja veloittaa käyttäjältä. (7; 8.)

3.2 Helsingin ja Milton Keynesin kaupunkipyöräpalvelut

Kirjoitushetkellä helmikuussa 2018 Birminghamissa ei ollut vielä kaupunkipyöriä, joten kaikista Birminghamin Nextbike-kaupunkipyöräpalvelun piirteistä ei ollut vielä varmuutta. Tästä syystä tässä luvussa kerrotaan Milton Keynesin Santander Cycles -kaupunkipyöristä, joiden vuokraamisen pitäisi toimia samalla tavalla kuin Birminghamiin tulevien kaupunkipyörien vuokraaminen. Nextbike tuottaa Santander Cycles -kaupunkipyöräpalvelun välineet ja ohjelmiston.

HSL-kaupunkipyöräpalvelu

Helsingin kaupunkipyörät ovat älypyöriä, joita voi vuokrata kahdella eri tavalla. Ensimmäinen tapa on syöttää pyörään oma käyttäjätunnus ja PIN-koodi. Toinen tapa vuokrata HSL-pyörä on HSL-matkakortin ja PIN-koodin avulla, jolloin matkakortti toimii käyttäjätunnuksena. Tunnukset käyttäjä saa rekisteröitymällä Helsingin seudun liikenteen (HSL) verkkosivuilla. Ensimmäisellä vuokrauskerralla käyttäjän täytyy käyttää tunnuksia, mutta matkakortilla tunnistamisen voi ottaa käyttöön heti seuraavalla kerralla. Tunnukset ovat voimassa valitun kauden ajan, ja sen hinta riippuu kauden pituudesta.

Helsingin kaupunkipyöriä voi käyttää 30 minuuttia ilmaiseksi, minkä jälkeen käyttö maksaa euron jokaiselta puolelta tunnilta. Maksut veloitetaan käyttäjän rekisteröimältä maksukortilta.

HSL-kaupunkipyörät palautetaan niille tarkoitetuille pyöräasemille. Jos pyörätelineessä ei ole tilaa, pyörän voi lukita myös telineen viereen. Pyörän lukitseminen päättää varauksen automaattisesti. (9.)

Santander Cycles -kaupunkipyöräpalvelu

Milton Keynesin kaupunkipyörien käyttäjäksi tulee rekisteröityä, ennen kuin pyöriä voi vuokrata, jotta lisäveloitus on mahdollista. Rekisteröityminen onnistuu kolmella eri

tavalla: Santander Cycles MK -mobiilisovelluksella, täyttämällä lomake verkossa tai soittamalla Santander Cyclesin asiakaspalveluun.

Milton Keynesin kaupunkipyöräpalvelua voi käyttää vuosijäsenyydellä tai ilman sitä. Jäsenyyden ostaneille ensimmäiset 30 minuuttia ovat ilmaisia. 30 minuutin jälkeen jokainen puoli tuntia maksaa 0,5 puntaa, mutta vuorokauden maksimihinta on aina 5 puntaa. Ilman jäsenyyttä jokainen 30 minuuttia maksaa käyttäjälle yhden punnan ja vuorokauden maksimihinta on 10 puntaa.

Helsingin pyörien tapaan Milton Keynesin kaupunkipyörät ovat älypyöriä. Niitä voi vuokrata kolmella eri tavalla. Ensimmäinen tapa on vuokrata pyörä Santander Cycles MK -mobiilisovelluksen avulla. Nextbike-kaupunkipyörissä lukee pyörän oma tunnusluku, joka syötetään sovellukseen. Käyttäjä vastaanottaa lukon avauskoodin, joka syötetään pyörän tietokoneeseen, minkä jälkeen lukko avautuu.

Toinen tapa on vuokrata pyörä suoraan pyörän tietokoneelta. Pyörän tietokoneeseen syötetään käyttäjän rekisteröimä puhelinnumero ja käyttäjäkohtainen PIN-koodi. Puhelinnumeron sijaan tunnistus onnistuu myös Santander Cycles -asiakaskortilla, jonka voi tilata rekisteröitymisen yhteydessä.

Kolmas tapa on soittaa Santander Cyclesin asiakaspalveluun, josta saa ohjeita noudattamalla lukon avauskoodin, joka syötetään pyörän tietokoneeseen.

Milton Keynesissä kaupunkipyörät tulee palauttaa niille tarkoitetulle pyöräasemalle. Jos teline on täynnä, pyörän voi lukita pyöräasemalle telineen viereen. Kun pyörä on pysäköity, käyttäjä painaa pyörän tietokoneen näytöltä ”OK”-painiketta, mikä päättää varauksen. (10.)

4 Kaupunkipyöräominaisuuden kehitys

4.1 Projektin eteneminen

Lähtötilanne

Kaupunkipyöräominaisuuden kehitys alkoi syksyllä 2017, kun MaaS Global alkoi laajentaa Whimin palveluja Birminghamin Nextbike-kaupunkipyöriin. Nextbike-pyörien oli tarkoitus tulla käyttöön Birminghamissa keväällä 2018.

Sovellukseen ominaisuutta lähdettiin kehittämään Helsingin kaupunkipyörille tarkoitettujen ulkoasusuunnitelmien pohjalta. MaaS Globalin taustajärjestelmäkehittäjä teki samaan aikaa yhteistyötä Nextbiken taustajärjestelmäkehittäjien kanssa luodakseen Whimin ohjelmointirajapintaan eli API:iin tuen Nextbike-pyörien varaamiselle. Siihen asti kunnes Whimin API:sta sai Nextbike-pyöriin ja -asemiin liittyvää oikeaa dataa, käytettiin kovakoodattua testidataa, jotta projekti saatiin käyntiin.

Testidatan avulla pystyttiin luomaan pohja pyörä-, asema- ja telineluokille, joiden avulla voitiin testata, miten dataa tulee käsitellä ja miten se tulee esittää käyttöliittymässä. Esimerkiksi pyöräasemien karttapinnit ja lisätietonäkymät pystyi luomaan testidatan avulla.

Yhteyksien kehittäminen

Kun ensimmäiset yhteydet oli luotu Whimin ohjelmointirajapinnasta Nextbiken ohjelmointirajapintaan, sovellukseen saatiin dataa Varsovan kaupunkipyöristä. Nextbikellä ei ollut tarjota käyttöön testiympäristöä, minkä takia kaikki saatu data oli oikeaa dataa. Pyörien varaamista ei siis voinut testata, koska kukaan ei olisi palauttamassa pyörää asemalle ja näin päättämässä pyörävarausta. Sekin, miten pyörien palauttaminen käytännössä tapahtuisi, oli vielä epäselvää. MaaS Globalin Nextbike-pyöräominaisuuden parissa työskentelevällä taustajärjestelmäkehittäjällä oli kokemusta Nextbike-palveluista Puolassa, ja tämän kokemuksen avulla pystyttiin luomaan jonkinlaisia olettamuksia siitä, miten vuokraus Birminghamissa voisi mahdollisesti toimia.

Pari kuukautta myöhemmin Nextbikellä oli saanut valmiiksi testiympäristön, josta saatiin testidataa kolmesta testipyöräasemasta Birminghamissa. Tämän jälkeen pystyttiin testaamaan pyörien vuokraamista ja palauttamista. Ensimmäiseksi Whimin

ohjelmointirajapinnan kautta pystyi ainoastaan vuokraamaan pyöriä, minkä takia sovellukseen jouduttiin tekemään väliaikainen yhteys suoraan Nextbiken ohjelmointirajapintaan, jotta pyörät saatiin myös palautettua. Lopulta pyörät saatiin palautettua myös Whimin ohjelmointirajapinnan kautta painamalla palautuspainiketta kaupunkipyöränäkymässä.

Projektin edetessä saatiin lisätietoja siitä, millaisia pyöriä ja asemia Birminghamiin oli tulossa. Opittiin, että Nextbike-pyörät Birminghamissa tulisivat olemaan älypyöriä, mutta pyörätelineet eivät olisi älytelineitä. Osaa alkuperäisistä suunnitelmista ei pystytty toteuttamaan ilman älytelineitä, minkä takia ulkoasusuunnitelmiin ja Whimin rajapintaan tehtiin muutoksia. Muun muassa pyörän palauttamiseen tarkoitettu painike poistettiin, koska Birminghamin kaupunkipyörien palauttamiseen ei tarvitsisi varmistusta sovelluksen kautta. Kaupunkipyörä välittäisi tiedon lukitsemisesta Nextbiken ohjelmointirajapintaan, joka välittäisi tiedon Whimin ohjelmointirajapintaan, josta sovellus saisi tiedon pyörävarauksen tilan muuttumisesta. Testausta helpottaakseni tein ajanottonäkymään testipainikkeen, jolla pyörän pystyi palauttamaan suoraan näkymästä.

Lopullinen kaupunkipyörien vuokraamisen tekninen kulku

Jotta pyörävarausvaihtoehtoja ja pyöräasemien dataa voi käsitellä sovelluksessa, ne tulee ensin hakea ohjelmointirajapinnasta. Pyörävaihtoehdot tallennetaan listoihin pyörävaihtoehtojen mukana tulevien pyöräasematunnusten perusteella. Pyörävaihtoehtolistat yhdistetään pyöräasematunnusta vastaavan pyöräaseman kanssa BikeStationItem-olioksi, joita jokaista varten luodaan oma karttapinni, johon olio kiinnitetään. Pyörävarausvaihtoehdot haetaan uudestaan aina, kun käyttäjä liikuttaa karttaa tai kun käyttäjän sijainti kartalla vaihtuu. Karttapinniä painamalla käyttäjä pääsee käsiksi pyöräaseman lisätietosivulla sijaitsevaan pyörävaihtoehtolistaan, josta hän valitsee haluamansa pyörän.

Varausvaihtoehto sisältää ajan, jolloin vaihtoehto on haettu ohjelmointirajapinnasta. Tätä varausvaihtoehdon aikaa käytetään varausta tehdessä myös varauksen aloitusaikana. Jotta vältytään luomasta varauksia menneisyyteen, haetaan vaihtoehdot uudestaan ja valitaan valittua pyörää vastaava vaihtoehto uusista vaihtoehdoista. Jos vaihtoehto ei ole enää saatavilla, siitä ilmoitetaan käyttäjälle, aseman lisätietosivu suljetaan ja varausvaihtoehdot haetaan uudestaan.

Ennen varauksen luomista käyttäjän täytyy hyväksyä kaupunkipyörävarauksen ehdot, ja ensimmäinen puoli tuntia veloitetaan käyttäjän maksukortilta.

Varauksen luomiseksi valittu pyörävarausvaihtoehto lähetetään ohjelmointirajapintaan, joka ilmoittaa, onko varauksen tekeminen onnistunut. Kun varaus on tehty onnistuneesti, sovellus alkaa lähettää viiden sekunnin välein kyselyjä ohjelmointirajapintaan varauksen tilan seuraamiseksi. Kysely palauttaa JSON-elementin, josta luodaan varausolio, joka sisältää varauksen viimeisimmät tiedot.

Kun käyttäjä on palauttanut pyörän Nextbike-asemalle, kysely palauttaa tiedon varauksen tilan vaihtumisesta EXPIRED-tilaan. EXPIRED-tilan mukana saadaan tieto siitä, miltä asemalta pyörä on haettu ja mille asemalle se on palautettu sekä varauksen kokonaiskesto. Varauksen vaihtuessa EXPIRED-tilaan ohjelmointirajapinta laskee varauksen kokonaishinnan ja veloittaa sen käyttäjältä.

Kun ohjelmointirajapinta on saanut laskettua varauksen hinnan, sovellus saa kyselyllä tiedon siitä, että varauksen tila on vaihtunut FINISHED-tilaan, jonka mukana sovellus saa varauksen kokonaishinnan. Hinta näytetään käyttäjälle lopetusnäkymässä muiden varauksen tietojen lisäksi.

Kommunikaatio

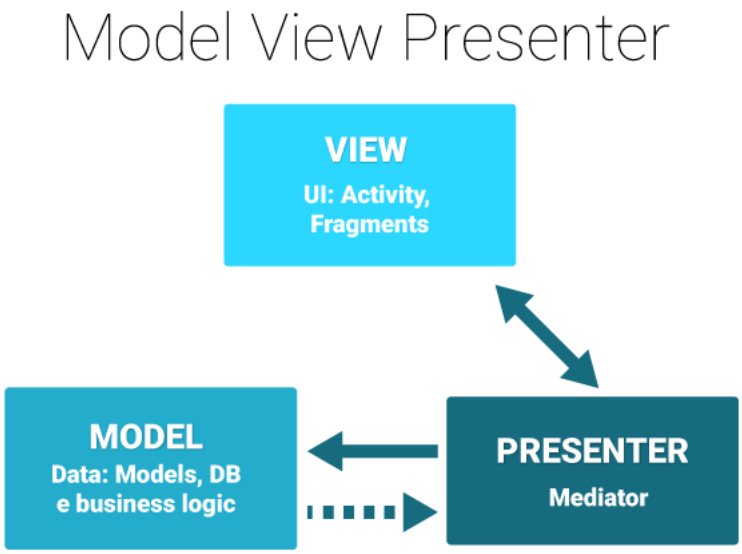
Aluksi MaaS Globalin taustajärjestelmäkehittäjä oli ainut, joka oli yhteydessä Nextbiken ohjelmoijiin, ja hän jakoi saamansa tiedon muille kehittäjille. Kommunikoinnin helpottamiseksi perustettiin WhatsApp-ryhmä, johon kuului kehittäjiä ja markkinointihenkilökuntaa sekä MaaS Globalilta että Nextbikeltä. Sovelluskehittäjät pystyivät keskusteluryhmän avulla kysymään asioita suoraan Nextbiken kehittäjiltä, mikä nopeutti vastausten saantia ja vapautti taustajärjestelmäkehittäjän tietolähteen roolista. Keskusteluryhmässä jaettiin tietoa pyöristä, asemista, projektin etenemisestä ja koodiin liittyvistä asioista.

4.2 Käytetyt kirjastot ja mallit

MVP

MVP (Model View Presenter) on sovelluksen kehitysmalli, joka on johdettu Android-kehityksessä usein käytetystä mallista MVC (Model View Controller). MVP jakaa sovelluksen käyttöliittymän kahteen kerrokseen, näkymään ja logiikkaan. MVP pitää siis erillään sen, miten käyttöliittymä toimii ja sen, mitä käyttäjälle näytetään. Yleensä Androidissa aktiviteetit sisältävät molemmat. Näkymän erottaminen logiikasta auttaa sovelluksen

ylläpidossa. Jos esimerkiksi logiikkaan vaaditaan muutoksia, ei näkymän metodeihin tarvitse välttämättä koskea. Pyöräominaisuutta kehittäessä pyörävarausten tilan seuraamiseen tarvitsi tehdä useita muokkauksia, kun rajapinnasta saatava data muuttui, mutta ulkoasuun vaikuttaviin metodeihin ei tarvinnut tehdä muutoksia. (11.) (Kuva 2.)



Kuva 2. Mallin, näkymän ja esittelijän väliset suhteet (12).

Model View Presenter -mallissa mallilla (Model) tarkoitetaan kaikkea datan käsittelyyn liittyvää: sen hakemista ohjelmointirajapinnasta, tallentamista ja tietokantojen käsittelyä. Näkymän (View) ainoa tehtävä on esitellä data käyttäjälle. Näkymällä tarkoitetaan aktiviteettien ja fragmenttien näkymiä. Esittelijä (Presenter) välittää datan mallin ja näkymän välillä. (13.)

Esimerkiksi, jos käyttäjä painaa näkymässä kaupunkipyöräpainiketta, näkymä välittää tiedon painalluksesta esittelijälle. Esittelijä välittää tiedon mallille, joka hakee kaupunkipyöriä ohjelmointirajapinnasta. Malli muuttaa saadun JSON-pyörädatan pyöräolioiksi ja välittää pyöräoliot esittelijälle. Esittelijä kertoo näkymälle, miten pyörät tulee näyttää käyttäjälle, ja näkymä näyttää pyörät ohjeiden mukaan.

RxJava 2.0

Whimissä käytetään paljon dataa, joka on riippuvaista muusta datasta. Esimerkiksi käyttöliittymässä näytetään käyttäjälle tieto siitä, onko hänellä voimassa oleva pyörävaraus

vai ei. Tätä varten haetaan palvelimelta tiedot aktiivisista pyörävarauksista. Jos pyörävaraus on aktiivinen, täytyy palvelimelle lähettää tasaisesti toistuvia kyselyjä varauksen tilasta, jotta sovellus tietää, milloin pyörävaraus on päättynyt ja käyttäjälle tulee näyttää matkan kesto ja hinta. (Esimerkkikoodi 1.)

Jotta tiedettäisiin, täytyykö pyörävarauksen tilaan liittyvien kyselyjen lähettäminen aloittaa, täytyy ensin saada tieto siitä, onko aktiivisia pyörävarauksia. RxJava 2.0 -kirjaston avulla pystytään seuraamaan, milloin tieto on haettu onnistuneesti, ja luomaan metodiketjuja. RxJava 2.0 mahdollistaa siis sen, että kyselymetodi aloitetaan vasta, kun tieto aktiivisesta pyörävarauksesta on varmistunut. (Esimerkkikoodi 1.)

```
private void getBikeBookingsByState() {
    Timber.d("getBikeBookingsByState()");
    final String[] bookingStates = new String[]{
        BookingState.PAID,
        BookingState.ACTIVATED
    };

    disposables.add(bookingService.getBookingsByState(bookingStates)
        .flattenAsObservable(routeBookings -> routeBookings)
        .filter(routeBooking -> TransportMode.BICYCLE.equalsIgnoreCase(route-
Booking.getLeg().getMode()))
        .toList()
        .subscribe(routeBookings -> {
          Timber.d("getBikeBookingsByState():: onSuccess():: routeBook-
ings.size(): " + routeBookings.size());

          if (!routeBookings.isEmpty()) {
            final RouteBooking routeBooking = routeBookings.get(0);
            Timber.d("getBikeBookingsByState():: onSuccess():: ongoing: " +
routeBooking);

            setOnGoingBikeBooking(true);
            setCurrentBooking(routeBooking);
            pollBookingById(routeBooking.getBookingId());
            setRemainingTimeInterval(routeBooking.getLeg().getStartTime());

            view.showOngoingBooking(routeBooking);
          } else {
            setOnGoingBikeBooking(false);
            setBikeLockCodeShown(false);

          }
        }, e -> Timber.e(e, "getBikeBookingsByState():: onError")));
  }
```

Esimerkkikoodi 1. RxJava 2.0 -kirjastoa hyödyntävä metodi, jolla haetaan aktiiviset pyörävaraukset ja päätetään, aloitetaanko tilan muutosten kysely ohjelmointirajapintaan.

RxJava 2.0 on kirjasto, jonka avulla voi kehittää reaktiivista Java-koodia. Reaktiivisella koodilla tarkoitetaan tietovuon (englanniksi ”data flow”) ympärille rakennettua ohjelmointimallia, jossa muuttujien muutokset välittyvät eteenpäin. Reaktiivisella ohjelmoinnilla

pyritään luomaan sovelluksia, jotka ovat joustavia ja nopeita ja käsittelevät virhetilanteita hallitusti. (14.)

RxJava 2.0:n perusta on observablet, observerit, operaatiot ja schedulerit. Observablet ovat asioita, joita halutaan seurata. Edellistä esimerkkiä käyttäen pyörävaraus on asia, jota halutaan seurata. Kun observablea seurataan, sille asetetaan yksi tai useampia seuraajia eli observereita. Observer kertoo, onko observablen välittämä data saatu onnistuneesti, mitä data on tai onko datan hakeminen epäonnistunut. Operaatioiden avulla voidaan muokata haettavaa dataa, esimerkiksi suodattamalla pois osia, joita ei tarvita, tai muuttamalla dataolio toiseksi olioksi. Esimerkiksi käyttäjän varauksia käsiteltäessä voidaan suodattaa pois kaikki muut varaukset kuin pyörävaraukset (esimerkkikoodi 1). Schedulerit määrittelevät, missä säikeessä metodi suoritetaan (scheduleOn()) ja mitä säiettä seurataan (observeOn()) (esimerkkikoodi 2). (15.)

```
public Single<List<RouteBooking>>
  getBookingsByState(@NonNull final String[] bookingStates) {
    return api.getBookingsByState(bookingStates)
        .flatMap(applySingle())
        .map(jsonObject -> mapJsonToObjectArray(jsonObject.get("bookings"),
            RouteBooking.class))
        .subscribeOn(Schedulers.io())
        .observeOn(AndroidSchedulers.mainThread());
  }
```

Esimerkkikoodi 2. Metodi, jolla haetaan käyttäjän varaukset niiden tilojen mukaan ja määritellään, missä säikeessä haku suoritetaan ja seurataan.

Observablen sijaan RxJava 2.0:ssa voi käyttää halutessaan Singleä, joka välittää vain yhden arvon tai tiedon tapahtuneesta virheestä. Singleä kannattaa käyttää esimerkiksi API-kutsumetodeissa, joiden odotetaan palauttavan vain yksi tulos, kuten lista kaupunkipyörävarausvaihtoehtoja. (Esimerkkikoodi 2.) (15.)

Retrofit ja Gson

Retrofit on kirjasto, jonka avulla voidaan tehdä kyselyjä REST-pohjaisen palvelimen rajapintaan. REST-pohjaiseen ohjelmointirajapintaan kyselyt lähetetään käyttämällä HTTP-operaatioita, kuten GET, PUT, POST ja DELETE. Retrofit-kirjastoa käytettäessä luodaan rajapintaluokkia, jotka sisältävät metodeina edellä mainittuja HTTP-operaatioita. Jokainen metodi on oma API-kutsunsa, jonka täytyy sisältää merkintä siitä, minkälainen operaatio on kyseessä (esimerkiksi POST tai GET). (16.) (Esimerkkikoodi 3.)

```
@GET("bookings/options/{agencyId}")
```

```
  Single<Response<JsonElement>> getBikeBookingOptions(
      @Path("agencyId") String agencyId,
      @Query("from") String fromLatLng,
      @Query("fromRadius") int fromRadius,
      @Query("to") String toLatLng,
      @Query("startTime") long startTime,
      @Query("mode") String mode);
```

Esimerkkikoodi 3. API-kutsu pyörävarausvaihtoehtojen hakuun Retrofitillä.

Retrofit on nopea tapa olla yhteydessä ohjelmointirajapintaan ja koodikanta näyttää siistimmältä, verrattuna esimerkiksi API-kutsujen luomiseen AsyncTaskilla (esimerkkikoodi 4) (17). Retrofit vaatii myös ”Retrofit.Builder”-luokan, jossa määritellään URL, johon API-kutsut lähetetään (16).

```
@Override
protected String doInBackground(String... strings) {
    String weather = "UNDEFINED";
    try {
        URL url = new URL(strings[0]);
        HttpURLConnection urlConnection = (HttpURLConnection) url.openConnec-
tion();

        InputStream stream = new BufferedInputStream(urlConnection.get-
InputStream());
        BufferedReader bufferedReader = new BufferedReader(new In-
putStreamReader(stream));
        StringBuilder builder = new StringBuilder();

        String inputString;
        while ((inputString = bufferedReader.readLine()) != null) {
            builder.append(inputString);
        }

        JSONObject topLevel = new JSONObject(builder.toString());
        JSONObject main = topLevel.getJSONObject("main");
        weather = String.valueOf(main.getDouble("temp"));

        urlConnection.disconnect();
    } catch (IOException | JSONException e) {
        e.printStackTrace();
    }
    return weather;
}
```

Esimerkkikoodi 4. API-kutsu säätietojen hakemiseen AsyncTaskin avulla (18).

Whimin API:sta saatava data on JSON-tyyppistä. Gson on Googlen kirjasto, joka yksinkertaistaa JSON-tyyppisen datan muuttamisen Java-olioiksi ja toisinpäin (19) (esimerkkikoodi 5).

```
final Option option = gson.fromJson(jsonElem, Option.class);
```

Esimerkkikoodi 5. JsonElementin muuttaminen Option-luokan olioksi Gsonia käyttäen.

Butter Knife

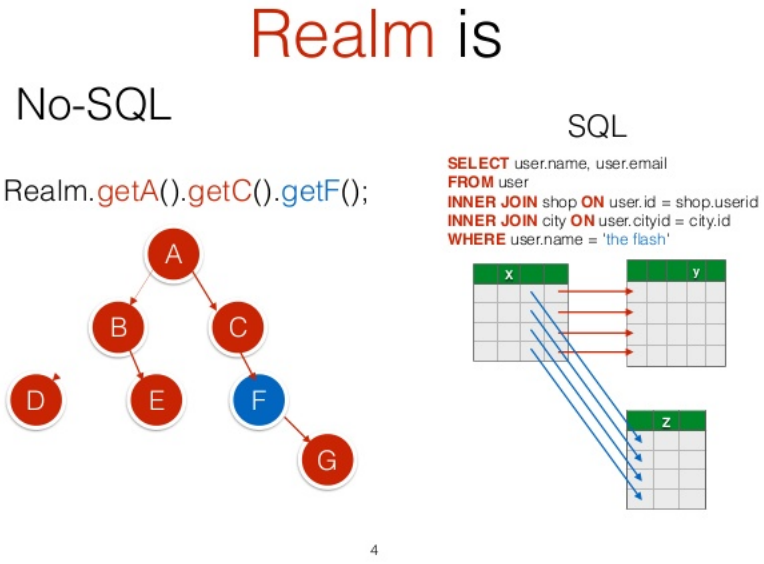
Whim käyttää Butter Knife -kirjastoa voidakseen sitoa näkymiä Android-komponentteihin vaivattomasti ja tilaa säästäen. Sitominen tehdään metodien ulkopuolella, ja sidotut komponentit ovat käytettävissä luokan kaikissa metodeissa. (20.) (Esimerkkikoodi 6.)

```
@BindView(R.id.bike_station_name)
TextView bikeStationNameText;
```

Esimerkkikoodi 6. Tekstinäkymän sitominen Android-komponenttiin Butter Knifen avulla.

Realm

Realm on tietokanta, johon data tallennetaan ja josta data haetaan olioina (kuva 3). Jotta Java-olion voi tallentaa Realmiin, sen täytyy periä RealmObject-luokka (21). Esimerkiksi käyttäjän päättyneet varaukset tallennetaan Realm-tietokantaan, jotta niitä varten ei tarvitse joka kerta luoda uutta API-kutsua, mikä vaatii internetyhteyden ja on hitaampaa kuin puhelimen muistista hakeminen.



Kuva 3. Esimerkki datan hakemisesta Realm-tietokannasta ja SQL-tietokannasta (22).

Ennen kuin Realm-tietokannan dataa voi muokata, täytyy avata yhteys Realmiin. Yhteyden ollessa auki Realmiin voi esimerkiksi tallentaa olioita tai sieltä voi poistaa olioita. Kun halutut haut tai muutokset on suoritettu, täytyy Realm-yhteys sulkea. Pelkkään datan hakemiseen ei tarvitse avata yhteyttä (esimerkkikoodi 7).

```
final RealmResults<User> users = realm.where(User.class).lessThan("age", max-
Age).findAll();
```

Esimerkkikoodi 7. Alle 50-vuotiaiden käyttäjien hakeminen Realm-tietokannasta.

Roomin käyttö ei vaadi yhteyden avaamista, mutta Roomia varten tulee luoda DAO-rajapintaluokkia, jotka sisältävät tarvittavat komennot tietokannan käsittelyyn, kuten tietokantaan olioiden tallentaminen, tietokannasta olioiden poistaminen tai olioiden hakeminen tietokannasta eri kriteereillä (esimerkkikoodi 8). (23.)

```
@Query("SELECT * FROM user WHERE age < :maxAge")
    public User[] loadAllUsersYoungerThan(int maxAge);
```

Esimerkkikoodi 8. Alle 50-vuotiaiden käyttäjien hakeminen SQLite-tietokannasta Roomin avulla.

Roomin ja Realmin isoin ero on, että Realm on kokonaan oma tietokantansa, kun taas Room on SQLite-tietokannan päälle rakennettu taso, joka helpottaa SQLite-tietokannan käsittelyä pukemalla SQL-komennot helposti luettaviksi ja käytettäviksi metodeiksi. SQLite-tietokanta on valmiiksi sisäänrakennettuna Android-laitteissa, minkä takia Roomin käyttö vie paljon vähemmän tilaa kuin Realmin. Realm-kirjaston koko on 3–4 megabittiä, kun taas Room vie tilaa vain 50 kilotavun verran. (23.)

Suurin syy siihen, miksi Whimin Android-sovelluksessa käytetään Realm-tietokantaa Room-tietokannan sijaan, on se, että RxJava 2.0 otettiin projektissa käyttöön vasta tammikuussa 2018. Room-tietokannan käyttö RxJavan kanssa vaatii android.arch.persistence.room:rxjava2:1.0.0-kirjaston asentamista, eikä sen käyttö ollut mahdollista RxJava 1.0:n kanssa. RxJava-kirjastopäivityksen myötä Room-tietokannan käyttöönottoa voidaan harkita uudestaan.

Google Maps

Kaupunkipyöränäkymässä näytetään pyöräasemat kartalla. Kartta tehtiin käyttämällä Google Maps Android API Utility -kirjastoa, jota käytetään kaikissa Whimin karttanäkymissä. Karttanäkymän lisäksi kirjasto tarjoaa erilaisia karttaan liittyviä toimintoja, kuten karttapinnien lisääminen, niiden koskettamiseen liittyviä toimintoja sekä käyttäjän sijainnin näyttäminen kartalla. (24.)

Barcode API

Barcode API on osa Google Vision API -kirjastoa. Barcode API:a käytettiin QR-koodiskannerin luomisessa. Barcode API käyttää mobiililaitteen kameraa skannatakseen QR-koodin. Osa Barcode API:n koodista oli vanhentunutta, minkä vuoksi sen mukana tulevat taskulamppuominaisuus ja automaattinen kohdentaminen eivät toimineet. Automaattista kohdentamista ei tarvittu, joten siihen liittyvät osat poistettiin, mutta taskulamppuun liittyvät kohdat korjattiin käyttäen Androidin omaa kamerakirjastoa. (25.)

Google Maps Directions API

Google Maps Directions API:a käytettiin kartalla käyttäjän sijainnin ja pyöräaseman välille luotuun reittiehdotukseen. Reittiehdotuksen saamiseksi API-kutsuun täytyy asettaa lähtö- ja määränpääkoordinaatit. Tarkempaa hakua varten voi asettaa vaihtoehtoisia määritteitä, kuten haluttu kulkumuoto tai tullien välttäminen. API-kutsu palauttaa sovellukseen DirectionsResult-olion. DirectionsResult-olio sisältää listan DirectionsRoute-olioita, joista jokainen on oma reittiehdotuksensa. Reittiehdotukset sisältävät muun muassa reitin, mahdollisia varoituksia (esimerkiksi tietyömaista) sekä mahdollisen hinnan, jos reitin kulkeminen vaatii kulkuvälineitä. (26.)

4.3 Apuna käytetyt ohjelmat

Valesijaintisovellukset

Whimin palvelutarjonta on sidoksissa käyttäjän sijaintiin. Käyttäjä näkee sovelluksesta vain lähialueensa palveluntarjoajien palvelut, minkä takia Nextbike-palvelut ovat näkyvillä vain kaupungeissa, joissa Whim tarjoaa Nextbike-kaupunkipyöriä. Jotta Birminghamin pyörädataan pääsi käsiksi ilman, että joutuisi matkustamaan Birminghamiin, täytyi

puhelimessa käyttää valesijaintia. Whimissä ei ole mahdollisuutta vaihtaa oikeaa sijaintia valesijainniksi, minkä takia käytettiin valmiita valesijaintisovelluksia.

Käytin ominaisuutta kehittäessä kahta eri valesijaintisovellusta. Ensimmäiseksi käytin Fake GPS GO Location Spoofer Free -sovellusta. Sovelluksen toimivuus oli vaihtelevaa, ja usein paikan nimellä hakeminen ei johtanut mihinkään. Vaihdellessani sijaintia Varsovan, Birminghamin ja Helsingin välillä, jouduin usein etsimään sijainnit suoraan kartalta, mikä hidasti työntekoa.

Fake GPS GO Location Spoofer Free -sovelluksella sijaintien etsiminen kartalta oli työlästä, minkä vuoksi vaihdoin lopulta työkaverini käyttämään Fake GPS Location -valesijaintisovellukseen. Aluksi kaupunkien hakeminen nimillä toimi hyvin, mutta viikon käytön jälkeen hakuominaisuuden toiminnassa alkoi olla häiriöitä. Haettu sijainti löytyi ja ilmestyi valikkoon, mutta sen painaminen ei aina vaihtanut sijaintia sovelluksessa. Tallensin kaikki käyttämäni valesijainnit suosikkeihin, mutta niidenkään kautta sijainti ei toisinaan vaihtunut. Päädyin taas usein hakemaan kaikki etsimäni kaupungit suoraan kartalta.

Ainoa asia, jonka takia Fake GPS Location oli mielestäni parempi kuin Fake GPS GO Location Spoofer Free, oli sen mukana tuleva widget eli pienoisohjelma, jonka avulla pystyi siirtämään valesijaintia toisen sovelluksen ollessa esillä. Tämän avulla pystyin simuloimaan käyttäjän liikkumista kaupunkipyöränäkymän kartalla.

Postman

Postman on ohjelmointirajapintakehitykseen tarkoitettu ohjelma, jonka avulla voi testata erilaisia API-kutsuja. Kaupunkipyöräominaisuutta kehitettäessä Postmania käytettiin pyörävarausvaihtoehtojen hakua, varausta ja palautusta testattaessa. API-kutsuja voi testata sovelluksestakin käsin, mutta se on hitaampaa, kun sovellus täytyy aina pakata ja käynnistää uudestaan uusien ominaisuuksien testaamiseksi. Postmanin käyttö nopeuttaa työskentelyä etenkin kokeiltaessa, miten eri yksittäiset arvot, kuten sijainti, vaikuttavat saatuun dataan.

Postman oli erittäin käytännöllinen, kun halusi tarkkailla ohjelmointirajapinnasta saatua dataa. Esimerkiksi aina, kun ohjelmointirajapinnasta saatuun dataan tuli muutoksia, ne sai nopeasti haettua Postmanin avulla ilman, että sovellusta tarvitsi käynnistää. Postman

esittää datan jäsenneltynä helposti luettavassa muodossa, mikä oli avuksi luotaessa olioluokkia, joina dataa käsiteltiin sovelluksessa.

Ohjelma oli myös avuksi, kun haluttiin seurata, miten varauksen tilan vaihtuminen vaikutti sovelluksen toimintaan. Ohjelman kautta pystyi muuttamaan varauksen tilaa ja samalla seuraamaan, tapahtuiko kaikki, mitä piti esimerkiksi, kun pyörävarauksen tila vaihtui päättyneeksi. (27.)

Sketch

Sketch on ulkoasusuunnitteluun tarkoitettu työkalu, jolla suurin osa Whimin ulkoasusuunnitelmista tehdään. Sketch tallentaa suunnitelmat vektorigrafiikkana omassa kuvatiedostomuodossaan Sketch-tiedostoina. Yleensä MaaS Globalin ulkoasusuunnittelijat suunnittelevat sovelluksen ominaisuuksia näkymä kerrallaan ja tallentavat ne Sketch-tiedostoina.

Suurimman osan ulkoasuelementeistä voi luoda sovellukseen ohjelmoimalla, mutta esimerkiksi sovellusta varten tehdyt kuvakkeet otetaan suoraan ulkoasusuunnitelmien tiedostoista. Sketch-tiedostoja pystyy käsittelemään lähes ainoastaan Sketchin avulla, minkä takia kehittäjienkin kannattaa asentaa Sketch. Sketch-tiedostoista pystyy valitsemaan haluamiaan elementtejä ja tallentamaan ne kuvatiedostomuotoihin, joita Android Studio tukee. Sketchin käyttöä varten on hankittava lisenssi, joka helmikuussa 2018 maksoi 99 Yhdysvaltojen dollaria vuodessa. (28.)

Zeplin

Zeplin on kehittäjien ja suunnittelijoiden yhteistyön avustamiseen tarkoitettu työkalu. Zepliniin voi ladata kuvatiedostoja eri kuvankäsittelyohjelmista, kuten Sketchistä tai Photoshopista. Sketch-tiedostoja pystyy siis käsittelemään myös Zeplinin avulla. Zepliniä voi käyttää tietokoneen työpöydältä sekä selaimessa. Suunnittelijat tallentavat sovelluksen ulkoasusuunnitelmat Zepliniin saman projektin alle ja kutsuvat osallistujiksi kaikki ne, jotka tarvitsevat pääsyn tiedostoihin. Osallistujat pääsevät katselemaan suunnitelmia ja tallentamaan suunnitelmien elementtejä. Suunnitelmien esikatselu on Zeplinillä nopeampaa kuin Sketchillä, koska Sketch-tiedostoja ei tarvitse erikseen ladata sinne. Zeplin lataa tehdyt muutokset automaattisesti projekteihin. (29.)

Ennen kuin Zepliniin luodaan uusi projekti, täytyy määritellä, mille käyttöjärjestelmälle projekti on tarkoitettu, jotta kuvatiedostoista tallennettavien elementtien kuvatiedostomuodot ja koot olisivat valmiiksi yhteensopivia projektissa käytetyn käyttöjärjestelmän kanssa. Whimin Zeplin-projekti on tarkoitettu iOS-käyttöjärjestelmälle, minkä takia siitä saatavat kuvaelementit eivät sovi yhteen Android-käyttöjärjestelmän kanssa. iOS-projektien kuvakokoyksikkö on pt (point), joka on 1/72 tuumaa, ja Android-projektien kuvakokoyksikkö on dp (density independence) (30). Dp on joustava yksikkö, joka skaalaa elementit näyttöön sopiviksi (31). Tämän vuoksi samat kuvaelementit eivät näytä hyviltä molemmissa käyttöjärjestelmissä. MaaS Globalilla työskentelevät Android-kehittäjät käyttävät Zepliniä enimmäkseen suunnitelmien esikatseluun ja tallentavat kuvaelementit Sketchin avulla.

Zeplinin käyttömaksu riippuu projektien määrästä. Helmikuussa 2018 hinnat vaihtelivat ilmaisesta 26:een Yhdysvaltojen dollariin kuukaudessa. (32.)

4.4 Yhteys ohjelmointirajapintoihin

Voidakseen tarjota kaupunkipyöriä käyttäjille Whimin täytyy olla yhteydessä palveluntarjoajan ohjelmointirajapintaan, josta saadaan muun muassa tieto vapaiden pyörien määrästä ja pyöräasemien sijainneista. Koska pyörien varaustilanne vaihtelee jatkuvasti, ei ole järkevää säilyttää pyöriin liittyvää dataa Whimin palvelimella, vaan data haetaan aina suoraan palveluntarjoajalta.

Sovellus on aina yhteydessä vain Whimin omaan ohjelmointirajapintaan, joka taas on yhteydessä eri palveluntarjoajiin. Tällä tavalla sovellus ei tarvitse jokaista palveluntarjoajaa varten omia API-kutsuja. Esimerkiksi Whimin ohjelmointirajapinnasta voi hakea erilaisia varausvaihtoehtoja lisäämällä yleisen varausvaihtoehtoja hakevan API-kutsun perään varauksen tyypin (esimerkiksi pyörä). Lisäksi API-kutsuun tulee lisätä käyttäjän sijainti, jotta pystytään tarjoamaan mahdollisimman lähellä sijaitsevia palveluja, sekä varauksen ajankohta.

Whimin taustajärjestelmä vastaanottaa API-kutsun ja lähettää puolestaan API-kutsun palveluntarjoajalle. Saatuaan vastauksen palveluntarjoajan ohjelmointirajapinnasta Whimin taustajärjestelmä välittää vastauksen (esimerkiksi pyörävaihtoehdot) sovellukseen, jossa ne näytetään käyttäjälle.

4.5 Testaus

Kun tiimiin saatiin laaduntarkkailija, testausprosessi muuttui järjestelmällisemmäksi. Aikaisemmin testaus tehtiin pyytämällä muita kehittäjiä kokeilemaan uudet tai korjatut ominaisuudet. Tämän tavan huono puoli on se, että kehittäjät ovat tottuneet sovellukseen ja tietävät, miten sitä tulee käyttää, joten kaikkia mahdollisia virheitä ei aina huomata ajoissa. Laaduntarkkailija lähestyy ominaisuuksia käyttäjän näkökulmasta ja kokeilee asioita, joita kehittäjä ei välttämättä keksi edes kokeilla.

Ennen kuin uudet tai korjatut ominaisuudet annetaan laaduntarkkailijalle testattavaksi, ominaisuuden tulee läpäistä muiden kehittäjien arviointi. Jokaisesta ominaisuudesta tehdään oma pull request, jonka muut kehittäjät käyvät läpi. Pull requestia voi kommentoida ja siihen voi jättää korjausehdotuksia, jotka kehittäjän tulee käydä läpi ja korjata, ennen kuin pull request hyväksytään.

Muiden kehittäjien arvioinnin lisäksi pull requestin tulee läpäistä Githubin automaatiotestausohjelman Travis CI:n testit. Travis CI käy läpi projektiin tehdyt muutokset sekä mahdolliset yksikkötestit ja testaa ne. Vaikka muut kehittäjät hyväksyisivät pull requestin, sitä ei voi liittää projektiin, jos Travis CI on löytänyt virheen. Vasta, kun yksi kehittäjistä on hyväksynyt muutokset ja Travis CI:n testit ovat menneet läpi, ominaisuus voidaan liittää osaksi projektia.

Whimin Android-sovelluksen kehityksessä ei ole vielä käytetty paljon yksikkötestejä, mutta niitä pyritään tekemään enemmän jatkossa, kun projekti laajenee.

MaaS Globalilla kaikki tehtävät merkitään joko Github-verkkopalveluun tai Jira-projektinhallintaohjelmaan. Tehtävissä selostetaan tehtävän sisältö, määritellään sen tärkeys ja kuka tehtävän tekee sekä tehtävän tila (”ei aloitettu”, ”työn alla” ja niin edelleen). Kun kehittäjä saa tehtävän valmiiksi, hän kirjoittaa laaduntarkkailijalle ohjeen siitä, mitä ominaisuutta tulee testata. Tehtävän tekijäksi vaihdetaan laaduntarkkailija ja tilaksi ”testauksessa”. Kun laaduntarkkailija on testannut ominaisuuden, hän joko muuttaa tehtävän tilan valmiiksi tai palauttaa sen takaisin kehittäjälle jatkokehitystä varten.

4.6 Valmis ominaisuus

MaaS Globalin Whim-sovelluksessa on valikko, josta käyttäjä näkee kulloisenkin alueensa palveluntarjoajat. Kaupunkipyöräpainikkeen löytää valikosta silloin, kun alueella on kaupunkipyöräpalveluja, joiden kanssa MaaS Global tekee yhteistyötä.

Kun kaupunkipyöräpalvelu on saatavilla, käyttäjä pääsee valikosta kaupunkipyöränäkymään. Näkymässä on kartta, johon on merkitty käyttäjän sijainti sekä lähialueen pyöräasemat karttapinneinä, joissa näkyy aseman vapaiden pyörien määrä. Koskettamalla pinniä, avautuu näkymä, jossa on lisätietoa pyöräasemasta sekä lista pyörävaihtoehdoista, joita painamalla käyttäjä varaa pyörän itselleen. Käyttäjä voi myös halutessaan varata pyörän skannaamalla pyörässä olevan QR-koodin. Varauksen yhteydessä käyttäjä maksaa ensimmäisen puolen tunnin hinnan. Kun varaus on vastaanotettu ja hyväksytty, näytölle ilmestyy koodi, jolla käyttäjä saa avattua valitun pyörän lukon.

Whim näyttää ajanottonäkymää, kunnes käyttäjä on palauttanut pyörän valitsemalleen asemalle, jotta käyttäjä tietää, kuinka kauan varaus on kestänyt ja kuinka paljon käyttö tulee maksamaan. Matkan tiedot näkyvät myös sovelluksen pääsivulla sekä erillisellä lippusivulla.

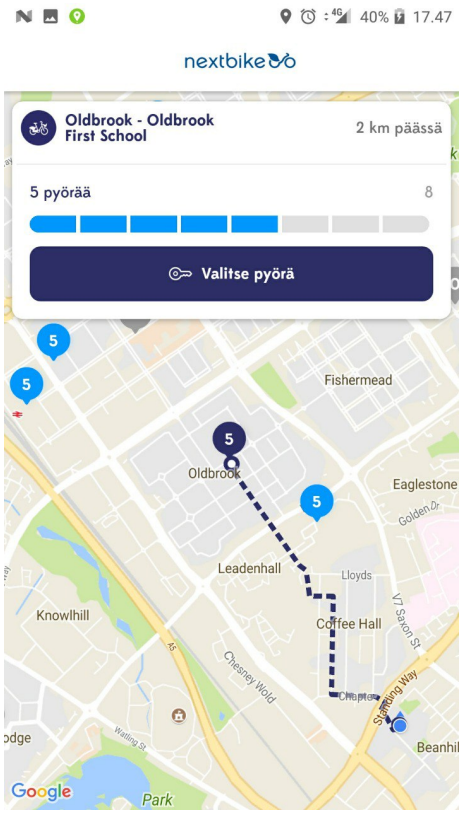
Kun pyörä on palautettu, palvelimelta saadaan tieto siitä, kuinka kauan matka kesti ja paljonko se maksoi. Jos matka kesti enemmän kuin 30 minuuttia, lisähinta veloitetaan käyttäjän tietoihin tallennetulta maksukortilta. Käyttäjä ei pysty vuokraamaan kaupunkipyöriä ilman tallennettua maksukorttia.

4.7 Ulkoasu

Kaupunkipyöränäkymän ulkoasu muuttuu riippuen siitä, onko käyttäjällä meneillään oleva pyörävaraus vai ei. Pyörävarauksen tila vaikuttaa muun muassa karttapinnien ulkoasuun, pyöräasemien lisätietosivujen ulkoasuun ja siihen, näytetäänkö käyttäjälle ajanottonäkymä.

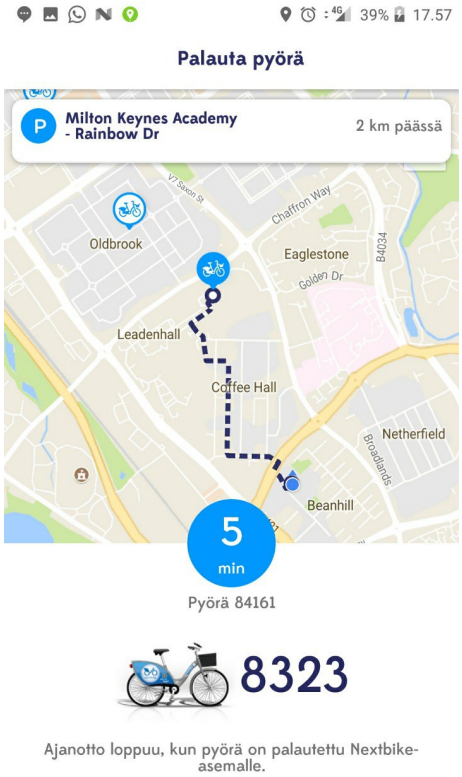
Karttapinnit

Jos käyttäjällä ei ole varattua pyörää, kartalla näytettävissä pyöräasemapinneissä näytetään vapaiden pyörien määrä sinisellä pohjalla. Google Maps -karttapinneillä on oletuksena punainen ulkoasu, minkä takia haluttua lopputulosta varten pinneille täytyi tehdä omat kuvakkeet ja korvata oletuskuvakkeet. Pyöräasemapinnin valitseminen muuttaa pinnin väriä tummemmaksi, mikä tehtiin vaihtamalla pinnin kuvaketta. (Kuva 4.)



Kuva 4. Karttapinnit, reittiehdotus ja pyöräaseman lisätietosivu ilman aktiivista varausta.

Kun käyttäjällä on voimassa oleva pyörävaraus, karttapinnien kuvakkeet vaihdetaan kuvakkeisiin, joissa on pieni sininen pyörän kuva valkoisella pohjalla ja valittuna valkoinen pyörä sinisellä pohjalla (kuva 5). Alun perin tarkoitus oli näyttää pyöräaseman vapaiden paikkojen määrä, mutta koska Birminghamin Nexbike-asemille pyörän voi jättää telineen ulkopuolelle eikä tietoa vapaista paikoista ollut saatavilla, päädyttiin edellä mainittuun ratkaisuun.



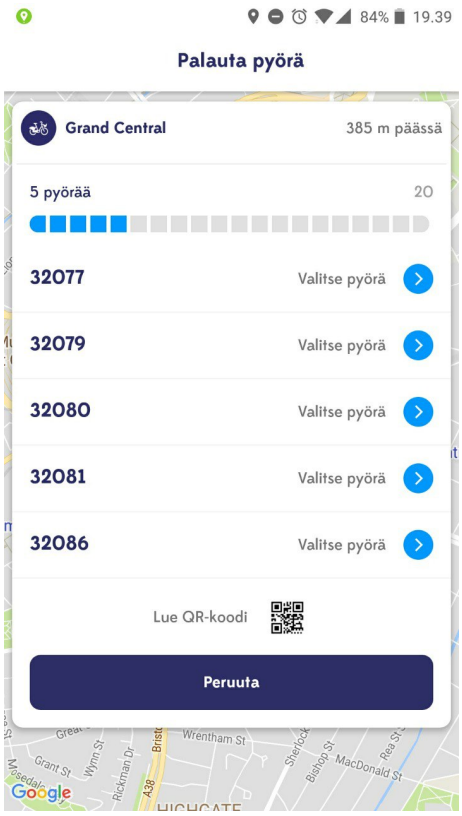
Kuva 5. Karttapinnit, reittiehdotus, pyöräasema ja ajanottonäkymä, kun käyttäjällä on aktiivinen pyörävaraus.

Reittiehdotukset

Karttapinnejä koskettamalla kartalle ilmestyy reittiehdotus, joka näkyy sinisenä katkoviivana käyttäjän sijainnista valitulle pyöräasemalle. Kun käyttäjällä on aktiivinen pyörävaraus, näkymän alareunassa on ajanottonäkymä, joka piilottaa osan kartasta. Aktiivisen pyörävarauksen aikana reittiehdotusta nostetaan hieman ylemmäksi, jotta käyttäjä näkee siitä tarpeeksi ison osan. (Kuva 4.)

Pyöräaseman lisätietosivu

Pyöräaseman lisätietosivu toteutettiin näyttämällä pyöränäkymäaktiviteetin päällä pieni fragmentti, joka avautuu painamalla pyöräasemakarttapinniä. Kun käyttäjällä ei ole aktiivista pyörävarausta, lisätietosivulla näytetään aseman nimi, etäisyys käyttäjästä, pyörätelineen paikkojen määrä, vapaiden pyörien määrä ja lista, jossa näkyy pyörävalikoima sekä painike, joka avaa QR-koodiskannerin. Lista on näkyvillä vain, jos käyttäjä on painanut ”Valitse pyörä” -painiketta. Vapaiden pyörien ja telineen paikkojen määrä näytetään sekä lukuina että horisontaalisena kuvaajana, jossa vapaat pyörät on merkitty sinisellä. (Kuva 6.)



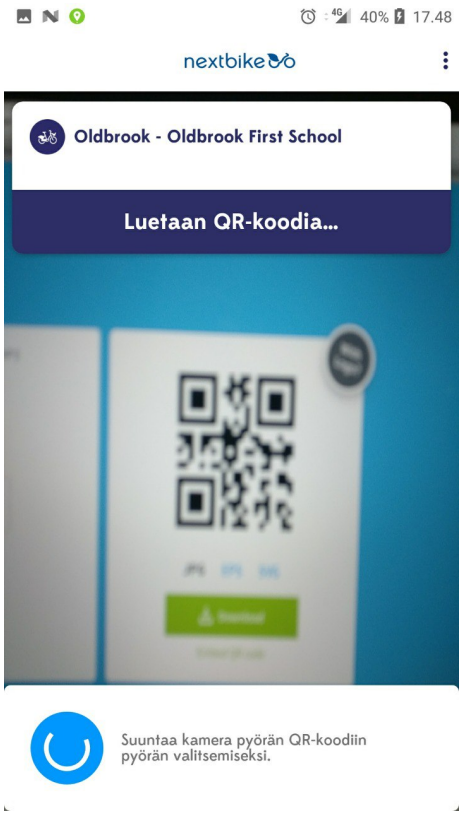
Kuva 6. Pyöräaseman lisätietonäkymä, kun pyörävalikoimalista on näkyvillä.

Kun käyttäjällä on aktiivinen pyörävaraus, sivulta piilotetaan pyörävalikoimalista ja vapaiden pyörien määrä. Birminghamissa pyöräasemista ei saa tietoa siitä, ovatko aseman pyörät telineessä vai sen vieressä, joten vapaiden telinepaikkojen näyttäminen ei ole mahdollista. Esimerkiksi tilanne, jossa pyörätelineessä on kymmenen paikkaa ja asemalla on kymmenen pyörää, mutta yhtäkään ei ole pysäköity telineeseen, on mahdollinen. Jos käyttäjälle näytettäisiin tällaisessa tilanteessa tieto, että asemalla ei ole yhtään vapaata paikkaa, tieto olisi harhaanjohtava ja kaiken lisäksi turha, koska pyörän voi jättää

asemalle telineen viereen, vaikka kaikki telinepaikat olisivatkin täynnä. Siksi päädyttiin ratkaisuun, että lisätietosivulla ei näytetä aseman pyörätelineen tietoja silloin, kun pyörävaraus on aktiivinen. (Kuva 5.)

QR-koodiskanneri

Käyttäjä voi varata kaupunkipyörän joko painamalla pyörävarausvaihtoehtoa listassa tai skannaamalla pyörässä olevan QR-koodin. QR-koodin skannausvaihtoehto näytetään pyörälistan viimeisenä elementtinä. Sitä painamalla käyttäjä pääsee QR-skannerinäkymään, jossa näkyy valitun pyöräaseman nimi ja lyhyet ohjeet QR-koodin skannaamiseen. (Kuva 7).



Kuva 7. QR-koodiskannerinäkymä.

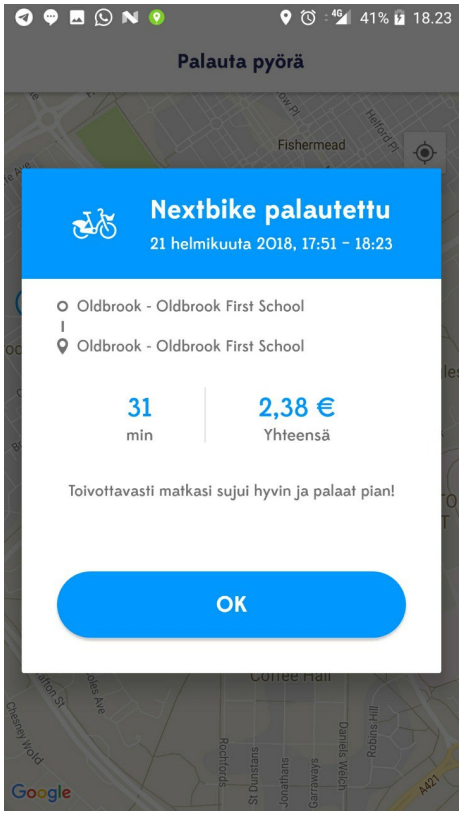
Ajanottonäkymä

Kun käyttäjä on varannut pyörän, kaupunkipyöränäkymän alaosaan tuodaan esille näkymä, jossa näkyy pyörän tunniste, lukon avauskoodi sekä sininen ympyrä, jonka keskellä näkyy ajanotto. Näkymä on esillä ja ajanotto on käynnissä niin kauan, kunnes

ohjelmointirajapinnasta saadaan tieto siitä, että pyörä on palautettu Nextbike-asemalle. (Kuva 5.)

Lopetusnäkymä

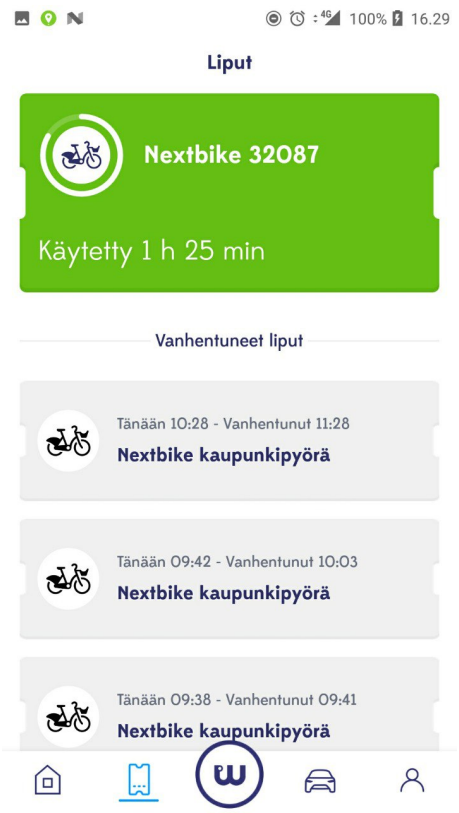
Pyörävarauksen päätyttyä kaupunkipyöränäkymään avautuu dialogi, jossa näkyy koottuna varauksen tiedot: lähtöasema, palautusasema, ajankohta, kesto ja veloitettu hinta. Lopetusnäkymästä täytyi tehdä monikäyttöinen, koska sitä tultaisiin käyttämään muidenkin kuin pyörävarausten päätyttyä. (Kuva 8.)



Kuva 8. Lopetusnäkymä, kun pyörä on palautettu samalle asemalle, kuin mistä se on otettu.

Lippusivu

Whimissä on sivu erilaisia matkalippuja varten. Sivulla näytetään käyttäjän aktiiviset ja vuorokauden sisällä vanhentuneet julkisen liikenteen liput sekä taksimatkojen liput. Taksimatkoille luotiin omat liput, sillä käyttäjä voi tilata päällekkäisiä taksimatkoja, mutta käyttäjän täytyy päästä näkemään jokaisen taksimatkan tiedot erikseen. Lippujen avulla käyttäjä pystyy seuraamaan useampaa taksitilausta samanaikaisesti. (Kuva 9.)



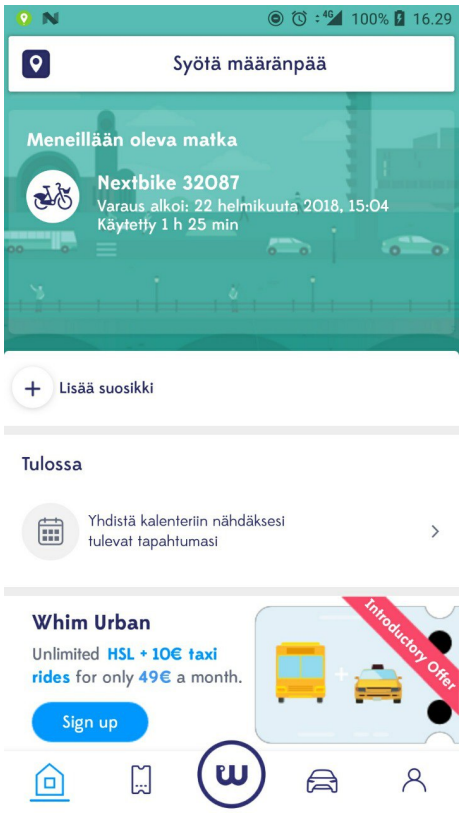
Kuva 9. Aktiivinen pyörälippu ja vanhentuneita pyörälippuja.

Pyörävaraukset näytetään lippusivulla, koska suunnitelmissa oli, että pyörävarauksen ensimmäiset 30 minuuttia olisivat ilmaisia maksullisen kuukausitilauksen omistaville. Lipussa näytetään käytetty aika sekä numeroina että renkaana, joka värittyy vähitellen, kunnes 30 minuuttia on kulunut. Kun 30 minuuttia on kulunut, renkaan väritys alkaa alusta. Painamalla aktiivista pyörälippua käyttäjä pääsee kaupunkipyöränäkymään. Vanhentuneen lipun painaminen vie käyttäjän myös kaupunkipyöränäkymään, mutta avaa siellä lopetusnäkymän. Näin käyttäjä pääsee näkemään vuorokauden sisällä päättyneiden pyörävarausten lisätietoja. Kaikki kuukauden sisällä päättyneet varaukset löytyvät myös profiilisivun ”Matkani”-painikkeen kautta.

Etusivu

Jotta käyttäjän ei tarvitse pysyä kaupunkipyöränäkymässä tai lippusivulla nähdäkseen pyörävarauksensa keston, aktiivinen pyörävaraus näkyy myös etusivulla. Whimin etusivun yläosassa on kohta, jossa näytetään aktiiviset matkat. Kohdassa on tilaa vain yhdelle aktiiviselle matkalle tai varaukselle. Pyörävarauksesta siinä näytetään varatun pyörän tunniste, varauksen alkamisajankohta ja kesto. (Kuva 10.)

Koska pyörävarauksen tietoja pääsee seuramaan myös lippusivulta ja kaupunkipyöränäkymästä, päädyttiin siihen, että jos käyttäjällä on pyörävarauksen kanssa samanaikaisesti aktiivisia matkoja, ne näytetään ensisijaisesti etusivulla.



Kuva 10. Aktiivisen kaupunkipyörävarauksen tiedot Whimin etusivulla.

4.8 Ominaisuuden yhteensopivuus muiden kaupunkipyöräpalveluiden kanssa

Yksi tärkeä tavoite oli, että kaupunkipyöräominaisuutta voitaisiin tulevaisuudessa käyttää myös muiden kaupunkipyöräpalveluiden kanssa Birminghamin Nextbike-kaupunkipyörien lisäksi. Yhteensopivuutta varmistettiin välttämällä Nextbiken mainitsemista suoraan Strings-tiedostoissa (tiedostot, jotka sisältävät näkymissä näytetyt tekstit), vaan lisäämällä palveluntarjoajan nimi koodillisesti. Kaikkiin kohtiin, joissa näytetään Nextbiken logo, on mahdollista asettaa minkä tahansa kaupunkipyöräpalvelun logo.

Asia, johon joudutaan varmasti tekemään muutoksia Helsingin kaupunkipyöriä varten, on pyörien varaus. HSL-kaupunkipyöriä ei varata pyöräkohtaisesti, vaan käyttäjä voi saamallaan lukon avauskoodilla ottaa pyörätelineestä minkä pyörän tahansa. Käyttäjälle ei näytetä listaa pyörävaihtoehdoista, ja varaus tehdään todennäköisesti muuttamalla Selaa pyöriä-painikkeen kautta. Tämä vaatii ainakin pieniä ulkoasumuutoksia.

Jotkut kaupunkipyöräpalvelut vaativat, että käyttäjä päättää itse pyörävarauksensa sovelluksen kautta. Tämä vaatii todennäköisesti palautuspainikkeen näyttämistä ajanottonäkymässä. Nämä tarvittavat muutokset voidaan tarpeen tullen tehdä, mutta ne eivät ole toistaiseksi korkealla tärkeysjärjestyksessä.

5 Yhteenveto

Insinöörityössä kehitettiin Whim-mobiilisovelluksen Android-versioon kaupunkipyörien vuokraamisominaisuus. Ominaisuuden oli tarkoitus tulla osaksi Whimin tarjoamia liikkumisvaihtoehtoja julkisen liikenteen, taksien ja vuokra-autojen lisäksi. Ensimmäinen tavoite oli tarjota käyttäjille Nextbike-kaupunkipyörien vuokrausmahdollisuus Birminghamissa keväällä 2018, kun pyöräasemat ja pyörät saapuvat Birminghamiin.

Toinen tavoite oli tehdä ominaisuudesta mahdollisimman ”kierrätyksen kestävä”, eli sen tuli toimia myös muiden kaupunkien kaupunkipyörien vuokraamisessa. Kaupunkipyöräominaisuuden avulla oli tarkoitus nähdä pyöräasemien sijainnit ja vapaiden pyörien määrä kartalla, pystyä vuokraamaan kaupunkipyöriä, seurata varauksen kestoa sekä nähdä varauksen yhteenveto sen päätyttyä.

Kaupunkipyöräominaisuuden kehitys on ollut isoin projekti, josta olen ollut päävastuussa MaaS Globalilla. Aloitin ominaisuuden kehittämisen syksyllä 2017 valmistelemalla karttanäkymän ja pyöräasemanäkymän, ja päädyin lopulta ottamaan vastuun muistakin ominaisuuksista. Sain tukea tiimiltäni kaikessa, mihin tarvitsin apua. Projektin loppusuoralla alkoi olla kova kiire, minkä vuoksi pyysin tiimikaveriani auttamaan näkymissä tapahtuvien muutosten animoinnissa.

Whimin iOS-versio oli julkaistu noin vuosi ennen Android-versiota, minkä vuoksi iso osa ominaisuuksista, joita olen aikaisemmin kehittänyt Whimin Android-versioon, oli jo valmiiksi olemassa iOS-versiossa. Tämän vuoksi tieto siitä, että kaikki tarvittava data saadaan ohjelmointirajapinnasta, oli varmaa. Kaupunkipyöräominaisuus oli uusi ominaisuus molemmille versioille ja Whimin ohjelmointirajapinnalle, minkä takia kaikkea tarvittavaa dataa ei saanut automaattisesti. Ohjelmointirajapinta ja ulkoasusuunnitelmat kehittyivät projektin edetessä, minkä takia yhteydenpito kaikkien projektiin osallistuneiden välillä oli tärkeää. Kehittäjien ja suunnittelijoiden tuli pysyä ajan tasalla, jotta ominaisuus saatiin toimimaan järkevällä tavalla.

Kaupunkipyöräominaisuus on valmis testaamiseen. Testaaminen aloitetaan Milton Keynesissä ennen Birminghamin kaupunkipyörien saapumista. Ominaisuuden kaikki vaaditut osat toimivat testiympäristössä, mutta oikeiden pyörien vuokraamisen yhteydessä löydetään mahdollisesti jotain parannettavaa. Koodiin tullaan muutenkin tekemään hienosäätöä, ja osia siitä muutetaan tulevaisuudessa Javasta Kotliniksi. Samalla, kun

mahdollisia muutoksia tehdään testaamisesta saadun palautteen jälkeen, aletaan valmistautua Helsingin kaupunkipyörien vuokraamisen mahdollistamiseen sovelluksesta käsin.

Lähteet

1 MaaS-konsepti ja MAASiFiE-projektiesittely. Verkkoaineisto. VTT. <http://www.vtt.fi/sites/maasifie/fi/finnish>. Luettu 16.2.2018.

2 Meistä. Verkkoaineisto. MaaS Global. <https://whimapp.com/fi/>. Luettu 16.2.2018.

3 Kuinka Uber toimii. Verrkoaineisto. Uber. <https://www.uber.com/fi/ride/how-uber-works/>. Luettu 20.2.2018.

4 Kuljettajan vaatimukset. Verkkoaineisto. Uber. <https://www.uber.com/fi/drive/requirements/>. Luettu 20.2.2018.

5 Wolt is the one app for discovering and getting great food. Verkkoaineisto. Wolt. <https://wolt.com>. Luettu 20.2.2018.

6 The Bike-sharing World Map. Verkkoaineisto. Google. <https://www.google.com/maps/d/u/0/viwer?mid=1UxYw9YrwT_R3SGsktJU3D-2GpMU&hl=en&ll=-23.014306864542263%2C138.1453712&z=2>. Luettu 18.2.2018.

7 FAQ. Verkkoaineisto. Mobike. <https://mobike.com/global/faq>. Luettu 18.2.2018.

8 How It Works. 2018. Verkkoaineisto. Nextbike. <https://nextbike.com>. Luettu 18.2.2018.

9 Näin käytät kaupunkipyöriä. Verkkoaineisto. HSL. <https://www.hsl.fi/kaupunkipyorat>. Luettu 18.2.2018.

10 How it works. 2018. Verkkoaineisto. Santander Cycles. <https://santandercyclesmk.co.uk/how-it-works/>. Luettu 20.2.2018.

11 Leiva, Antonio. 2014. MVP for Android: how to organize the presentation layer. 2014. Verkkoaineisto. <https://antonioleiva.com/mvp-android/>. Luettu 16.2.2018.

12 Fadhel, Ahmed Ben. MVP for Android. Verkkoaineisto. <http://ahmedbenfadhel.com/2014/04/15/mvp_for_android/>. Luettu 24.2.2018.

13 Model-View-Presenter: Android guidelines. 2017. Verkkoaineisto. Medium. <https://medium.com/@cervonefrancesco/model-view-presenter-android-guidelines-94970b430ddf>. Luettu 24.2.2018.

14 Reaktiivisen sovelluksen ohjelmointi. Verkkoaineisto. Hellas, Arto; Agile Education Research. <https://web-palvelinohjelmointi.github.io/osa7.html#chapter13>. Luettu 5.3.2018.

15 Reactive Extensions (Rx). 2016. Verkkoaineisto. Medium. <https://medium.com/@kevalpatel2106/what-is-reactive-programming-da37c1611382>. Luettu 17.2.2018.

16 Retrofit. Verkkoaineisto. 2017. Vogella. <http://www.vogella.com/tutorials/Retrofit/article.html>. Luettu 17.2.2018.

17 Android Async HTTP Clients: Volley vs Retrofit. 2013. Verkkoaineisto. Instructure. <http://instructure.github.io/blog/2013/12/09/volley-vs-retrofit/>. Luettu 5.3.2018.

18 Android Networking Tutorial with AsyncTask. 2016. Verkkoaineisto. Zenva. <https://androidkennel.org/android-networking-tutorial-with-asynctask/>. Luettu 5.3.2018.

19 How to convert Java object to / from JSON (Gson). 2016. Verkkoaineisto. Mkyong.com. <https://www.mkyong.com/java/how-do-convert-java-object-to-from-json-format-gson-api/>. Luettu 17.2.2018.

20 ButterKnife—A viewbinding library for Android (Beginner guide). 2017. Verkkoaineisto. Medium. <https://medium.com/@pranaypatel/butterknife-a-viewbinding-library-for-android-beginner-guide-fd92caf8e505>. Luettu 17.2.2018.

21 Introducing Realm. 2014. Verkkoaineisto. Realm. <https://realm.io/blog/introducing-realm/>. Luettu 17.2.2018.

22 Phạm, Khắc. 2016. Realm Presentation. Verkkoaineisto. <https://www.slideshare.net/khacpv/realm-presentation>. Luettu 25.2.2018.

23 Realm, ObjectBox or Room. Which one is for you? 2017. Verkkoaineisto. Dev Labs. <https://notes.devlabs.bg/realm-objectbox-or-room-which-one-is-for-you-3a552234fd6e>. Luettu 6.3.2018.

24 Google Maps Android API Utility Library. 2017. Verkkoaineisto. Google Developers. <https://developers.google.com/maps/documentation/android-api/utility/>. Luettu 17.2.2018.

25 Barcode API Overview. 2017. Verkkoaineisto. Google Developers. <https://developers.google.com/vision/android/barcodes-overview>. Luettu 17.2.2018.

26 Developer's Guide. 2018. Verkkoaineisto. Google Developers. <https://developers.google.com/maps/documentation/directions/intro>. Luettu 24.2.2018.

27 Postman Supports Every Stage of the API Lifecycle. 2018. Verkkoaineisto. Postman. <https://www.getpostman.com>. Luettu 23.2.2018.

28 The digital design toolkit. 2018. Verkkoaineisto. Sketch. <https://www.sketchapp.com>. Luettu 24.2.2018.

29 Features. Verkkoaineisto. Zeplin. <https://zeplin.io/features>. Luettu 24.2.2018.

30 Points & Picas. Verkkoaineisto. Fonts.com. <https://www.fonts.com/content/learning/fontology/level-1/type-anatomy/points-and-picas>. Luettu 24.2.2018.

31 Units & measurements. Verkkoaineisto. Material Design. <https://material.io/guidelines/layout/units-measurements.html#>. Luettu 24.2.2018.

32 Pricing. Verkkoaineisto, Zeplin. <https://zeplin.io/pricing>. Luettu 24.2.2018.