**Pathfinder**

TTOW0630 Mobile Project

Miika Avela  
Janne Möttölä

Harjoitustyö

Joulukuu 2016

Ohjelmistotekniikka

Tekniikan ja liikenteen ala

Sisällysluettelo

[1 Johdanto 1](#_Toc468733434)

[2 Toimeksianto 1](#_Toc468733435)

[3 Tavoitteet 1](#_Toc468733436)

[4 Käyttötapaukset ja kohderyhmä 1](#_Toc468733437)

[5 Prioriteetit 2](#_Toc468733438)

[6 Välineet ja teknologiat 2](#_Toc468733439)

[6.1 Julkiset rajapinnat 3](#_Toc468733440)

[6.1.1 Google 3](#_Toc468733441)

[6.1.2 DigiTraffic 3](#_Toc468733442)

[7 Käyttö 4](#_Toc468733443)

[8 Tietoa ohjelmasta 5](#_Toc468733444)

[8.1 Pathfinderin rakenne 5](#_Toc468733445)

[8.1.1 Suhteet 5](#_Toc468733446)

[8.1.2 Ohjelman looginen kulku 6](#_Toc468733447)

[8.2 Luokat ja niiden tarkoitukset 7](#_Toc468733448)

[8.2.1 Aktiviteetit 7](#_Toc468733449)

[8.2.2 Data 11](#_Toc468733450)

[8.2.3 Adapterit 13](#_Toc468733451)

[8.2.4 Rajapinnat 13](#_Toc468733452)

[8.2.5 Objektit 14](#_Toc468733453)

[8.3 Toteutettu toiminnallisuus 15](#_Toc468733454)

[8.4 Toteuttamaton toiminnallisuus 15](#_Toc468733455)

[8.5 Luokkien eroavaisuudet suunniteltuihin 15](#_Toc468733456)

[9 Laadunvarmistus ja testaus 16](#_Toc468733457)

[10 Ongelmat 16](#_Toc468733458)

[10.1 Ratkaistut ongelmat kehityksen aikana 16](#_Toc468733459)

[10.2 Yhä olemassalevat ongelmat ja niiden ratkaiseminen 17](#_Toc468733460)

[10.3 Tunnetut virhetilanteet 18](#_Toc468733461)

[11 Analyysi 18](#_Toc468733462)

[11.1 Mitä opittiin 18](#_Toc468733463)

[11.2 Mitä tulisi opetella lisää 18](#_Toc468733464)

[11.3 Suurimmat haasteet 19](#_Toc468733465)

[11.4 Mikä olisi pitänyt tehdä toisin 19](#_Toc468733466)

[11.5 Jatkokehitys 19](#_Toc468733467)

[11.6 Kommunikointi 20](#_Toc468733468)

[11.7 Vastuunjako 20](#_Toc468733469)

[11.8 Toteutukseen kuluneet resurssit 21](#_Toc468733470)

[12 Arviointi 24](#_Toc468733471)

# Johdanto

Tämä harjoitustyö on TTOW0610 Mobiiliprojekti –kurssin arviointiperusteena.

Harjoitustyön aiheena on ohjelma, joka tulkitsee Suomen julkisista junaverkkorajapinnoista dataa ja piirtää sen avulla käyttäjän päätelaitteelle reitin toivottuun määränpäähän hyödyntäen julkisia kulkuneuvoja.

# Toimeksianto

Toimeksiantona oli rakentaa Android-käyttöjärjestelmälle ohjelma vapaavalintaiseen tarkoitukseen.

# Tavoitteet

Tavoitteena oli tuottaa toimiva ohjelma, joka kykenee hakemaan ja näyttämään käyttäjälleen useita reittivaihtoehtoja eri aikatauluilla, mitä noudattamalla hän pääsee määränpäähänsä. Reittien tuli rakentua junaliikenteen ympärille.

Jokainen reittivaihtoehto oli tarkoitus näyttää yksityiskohtaisesti ja pilkottuna aikataulujen, junanumeroiden ja muun relevantin informaation kera.

Googlen rajapintoja hyödyntämällä saatiin kävelyreitit juna-asemille ja -asemilta, kun taas Digitrafficin rajapinnasta kerättiin junien aikatauluihin liittyvä informaatio joista parsittiin käyttäjälle keskeinen data näkyviin.

# Käyttötapaukset ja kohderyhmä

Ohjelma on suunniteltu kenelle tahansa joka haluaa matkustaa suomessa pidempiä matkoja. Käyttäjille, jotka eivät tunne reittiä juna-asemalle on tehty käyttäjäystävällinen käyttöliittymä, jota seuraamalla pääsee myös lähimmälle asemalle.

# Prioriteetit

|  |  |
| --- | --- |
| Prioriteetti | Ominaisuus |
| 1 | Lomake jossa määritetään mistä lähdetään ja minne mennään |
| 1 | Kartta josta näkee kokonaisreitin haluttuun päämäärään |
| 1 | Ohjelma pilkottu eri aktiviteetteihin |
| 1 | Reitti pilkottuna etappeihin ja ohjeisiin |
| 1 | Junien integroiminen reittien laskentaan ja määrittelyyn |
| 2 | Vaihtoehtoisten reittien tarjoaminen |
| 2 | Kaukoliikenteen linja-autojen integroiminen reittilaskentaan ja määrittelyyn |
| 2 | Lähiliikenteen linja-autojen integroiminen reittien laskentaan ja määrittelyyn |
| 3 | Lentokoneiden integroiminen reittien laskentaan ja määrittelyyn |
| 4 | Laajentaminen ulkomaille |

# Välineet ja teknologiat

Koska tehtävänantona oli tuottaa Android-järjestelmälle ohjelma, on luonnollista että ohjelman kehittämiseen käytettiin Android-laitteita.

Käytössämme oli:

* Huawei P8 Lite, Android 6.0
* OnePlus 3, Android 6.0.1

Ohjelmointiympäristönä projektin aikana käytimme Android Studiota, mikä on Androidin virallinen ohjelmointiympäristö.

Minimi API-taso ohjelmalle oli 19 – KitKat ja kohdetaso 24 – Nougat.

## Julkiset rajapinnat

Koska ohjelmalle on kriittistä kyetä tutkimaan sijainteja, koordinaatteja ja liikennettä, se tarvitsee tietoa koskien edellämainittuja. Nämä tiedot on saatavilla julkisista rajapinnoista.

### Google

Kyseisellä IT-maailman jättiläisellä on tarjottavana valtava määrä rajapintoja, joista ohjelma käyttää kolmea. Googlen rajapinnat vaativat API-avaimen, jonka perusteella Google rajoittaa hakujen määrää.

#### Google Maps

Android-laitteella on sisäinen maailmankartta olemassa, mutta se on liian karkea ohjelman käyttätarkoituksiin. Siispä käyttöön otettiin GoogleMaps API joka tarjoaa tarkemman ja selkeämmän kartan johon voi merkitä reittejä ja kohteita käyttäjää varten.

#### Google Directions API

Jotta saadaan reittiohjeet piirrettyä kartalle, Directions API:a tarvitaan kävelyreittitietojen hakemiseen annettujen koordinaattien perusteella. Directions API palauttaa pyytäessä JSON- tai XML-muodossa http-kyselyn tulokset reitistä josta Pathfinder parsii itselleen keskeiset tiedot käyttöön. Pathfinder pyytää JSON-formaattia kyselyssä.

#### Google Geolocation API

Käyttömukavuuden säilyttämiseksi ohjelmalle kelpaa koordinaattien lisäksi myös osoitteet. Jotta osoitteet saadaan muutettua koordinaateiksi, ohjelman sisäisiä laskutoimituksia varten, tarvitaan Geolocation API:a joka kääntää osoitteita koordinaateiksi tai toisinpäin. Geolocations API toimii http-kutsulla ja palauttaa XML- tai JSON-formaatissa vastauksen, joka voidaan määrittää kyselyssä. Pathfinderissa käytetään JSON-formaattia.

### DigiTraffic

Digitraffic on Liikenneviraston palvelu, minkä kautta on saatavissa avoimesti ajantasaista liikennetietoa Suomen tieverkolta ja rautatieliikenteestä. Tietolähteenään palvelu käyttää *ratakapasiteetin hallinnan LIIKE-järjestelmää*, minkä ylläpidosta vastaavat useammat rautateiden ympäristössä toimivat tahot. Palvelu ei vaadi rekisteröitymistä eikä erillistä API -avainta.

Ohjelmassa käytetään Digitrafficin rautatieliikenteen REST -rajapintaa, mistä haetaan reitinhakuun vaadittavaa dataa hakuparametrien avulla. Datapyynnöt tapahtuvat http-kutsuilla ja vastauksen paluuarvot tulevat JSON-formaatissa. Lisätietoa rajapinnan käytöstä ja hakuparametreista <http://rata.digitraffic.fi>

# Käyttö

Ohjelmiston käynnistyttyä, alkuruudussa (ks. 8.2.1.2 MainActivity) on yksinkertainen näkymä jossa käyttäjältä kysytään mistä lähdetään ja minne on tarkoitus mennä, sekä milloin lähdetään liikkeelle. Käyttäjä saa vaihtoehtoisesti valita oman laitteensa sijainnin lähtöpisteeksi, jolloin ohjelma varmistaa että laitteen seuranta-asetukset ovat kunnossa ja pyytää käyttäjää muuttamaan asetuksia mikäli ne eivät ole.

Kun tiedot on syötetty ja painetaan Hae reitit –nappia, niin ohjelma vie toiseen aktiviteettiin (ks. 8.2.1.3 RoutePresenter) jossa käyttäjälle listataan reittivaihtoehdot aikatauluineen. Selkeästi epämieluisia vaihtoehtoja voi poistaa listasta pyyhkäisemällä vasemmalle, kun taas kiinnostavia vaihtoehtoja voi näyttää kartassa kokonaisuudessaan klikkaamalla, sekä halutun reitin voi valita tarkempaan tarkasteluun pyyhkäisemällä oikealle.

Kun reitin tarkempi tarkastelu valitaan, niin aukeaa jälleen uusi aktiviteetti (ks. 8.2.1.4 segmentPresenter). Siellä on samankaltainen näkymä, paitsi että käyttäjälle listataan kokonaisten reittien sijaan valittu reitti pilkottuna korttinäkymään etapeiksi, joista jokainen voidaan klikkaamalla näyttää kartassa jota on helppo käännellä, tarkentaa, loitontaa tai tarkastella. Reitin listattua etappia klikkaamalla ohjelma tarkentaa kartan näyttämään sen. Mikäli laitteen seuranta on aktiivinen, voidaan kartan oikeassa yläkulmassa olevasta napista etsiä käyttäjän oma sijainti kartalla.

Reitin palasia seuraamalla käyttäjä pääsee määränpäähänsä. Mikäli tutkittava reitti ei ollutkaan mieluisa, niin edelliseen näkymään pääsee painamalla androidin Back-painiketta, jossa edelliset tulokset ovat yhä listattuna. Uuden haun voi tehdä perääntymällä ohjelman aloitusruutuun.

# Tietoa ohjelmasta

Tässä osiossa käsitellään ohjelman rakennetta ja toimintaa.

## Pathfinderin rakenne

### Suhteet

Pathfinderissa on reilusti luokkia ja niiden välisiä suhteita. Aktiviteetit tarvitsevat paljon dataa näytettäväkseen, joka vaatii paljon parsintaa ja vertailua saavuttaakseen oikean muodon (ks. Figure 1). Tätä varten Pathfinderissa on luokkia jotka on omistettu datan parsinnalle, joita aktiviteetit voivat käyttää tarpeisiinsa.

Luokkien tarkemmat määritelmät ovat kohdassa 8.2.

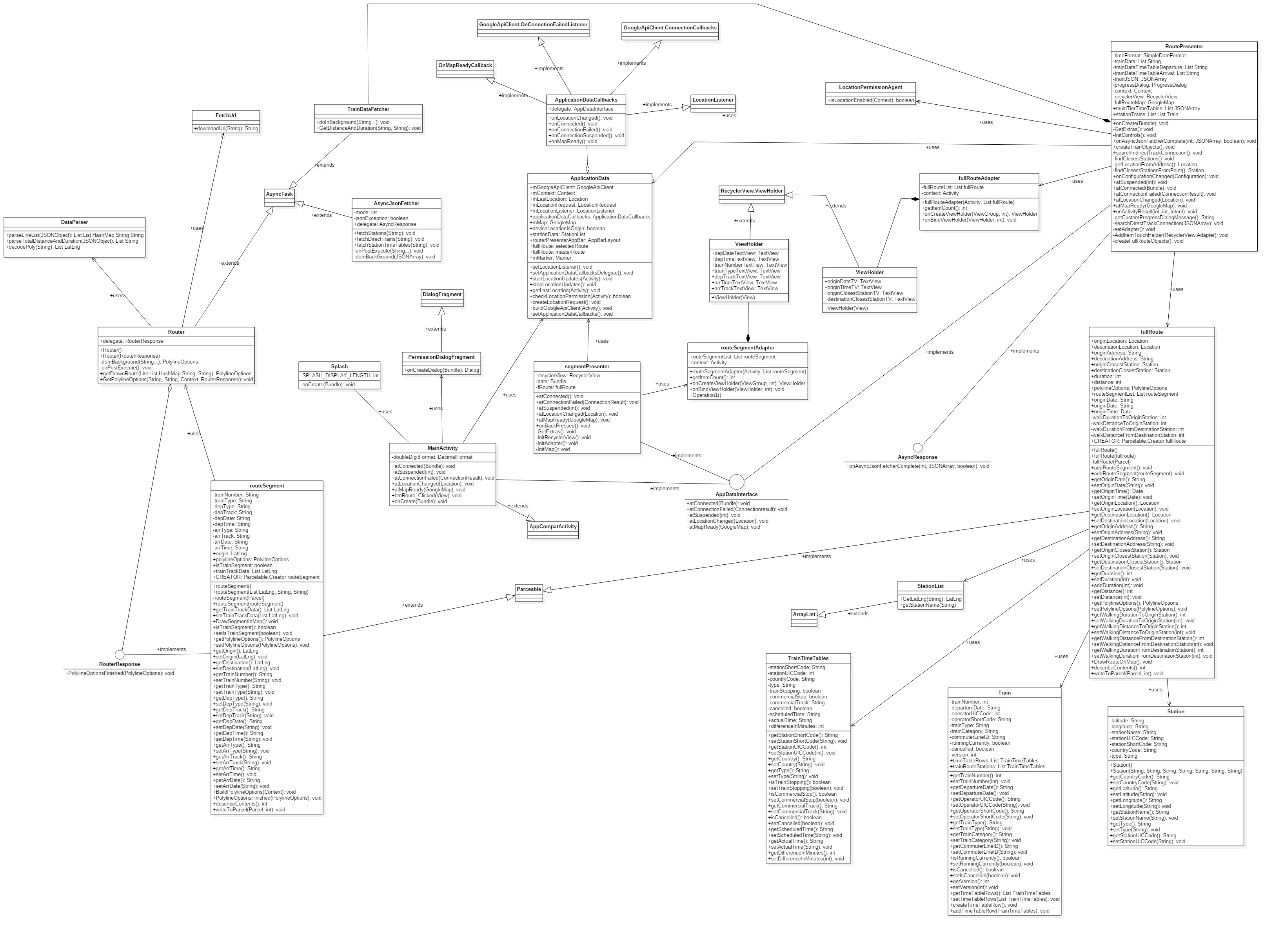


Figure 1: Luokkakaavio

Kaavio on myös projektin liitetiedostona, josta katselmointi on helpompaa.

### Ohjelman looginen kulku

Ohjelma hakee käyttäjän antamia osoitteita vastaavat koordinaatit ja juna-asemien tiedot Suomesta, joista vertailemalla saadaan selville lähtöpaikkaa ja kohdetta lähimmät juna-asemat. Kun ne tiedetään, ohjelma kyselee Digitrafficilta junatiedot joilla kyseisten asemien väli päästäisiin kulkemaan ja parsii niistä listan käyttäjälle mikäli niitä löytyy.

Junatiedot saatuaan ohjelma listaa ne käyttäjälle ja hakee asynkronisesti Google Directions rajapinnasta kävelyreitit lähtöpisteestä lähtöasemalle ja kohdeasemalta kohteeseen, jotka kiinnitetään valmistuessaan omille paikoilleen reitteihin.

Tässä vaiheessa ohjelma on kerännyt kaiken keskeisen datan ja voi näyttää reittivaihtoehtoja annetulle välille. Kierros alkaa alusta kun käyttäjä painaa aloitusruudussa ’Hae reitit’ -nappia.

Ohjelman pääpiirteinen toimintakaavio on nähtävissä alla (ks. Figure 2, Aktiviteettikaavio).

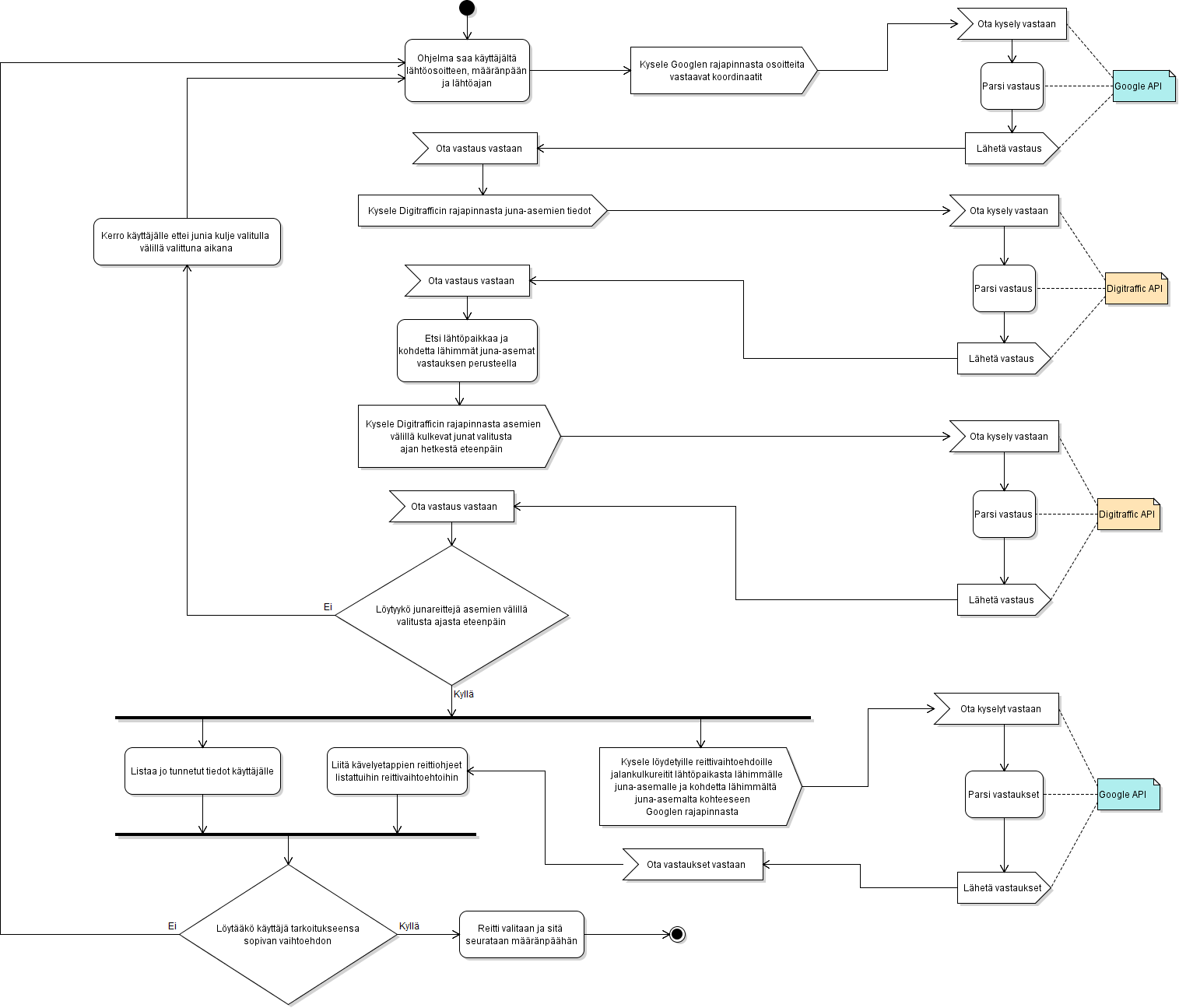


Figure 2: Aktiviteettikaavio

Aktiviteettikaavio on myös projektin liitetiedostona, josta katselmointi on helpompaa.

## Luokat ja niiden tarkoitukset

### Aktiviteetit

Aktiviteetit ovat luokkia joihin kiinnitetään käyttäjälle näkyvät layoutit.

#### SplashActivity

Jotta ohjelma ei näyttäisi niin askeettiselle, lisättiin aktiviteetti jonka ainoa tarkoitus on näyttää käyttäjälle ohjelman lataus-/esittelyruutua ja avata MainActivity muutaman sekunnin kuluttua.

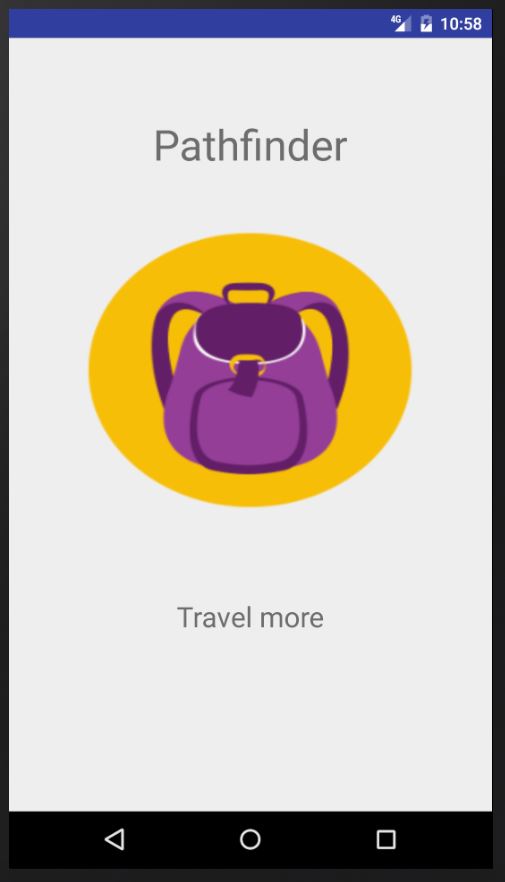


Figure 3:SplashActivity

#### MainActivity

MainActivity sisältää näkymän hakutietojen syöttämiseen. Sen käyttöliittymä sisältää yksinkertaisia tekstikenttiä ja toggleja käyttäjää varten, joilla määritellään haun parametrit. Se initialisoi keskeiset elementit (ks. 8.2.2.1 ApplicationData) jotka käsittelevät laitteen sijaintia ja ovat vastuussa reittihaun käynnistämisestä.

Mikäli käyttäjä aktivoi moodin jossa lähtösijaintina käytetään laitteen sijaintia, Pathfinder tutkii onko laitteen paikannus päällä. Mikäli se ei ole, niin käyttäjä ohjataan aktivoimaan seuranta (ks. 8.2.2.7 PermissionDialogFragment), jonka jälkeen moodi voidaan laittaa päälle.

MainActivity perii AppCompatActivityn ja implementoi AppDataInterfacen.

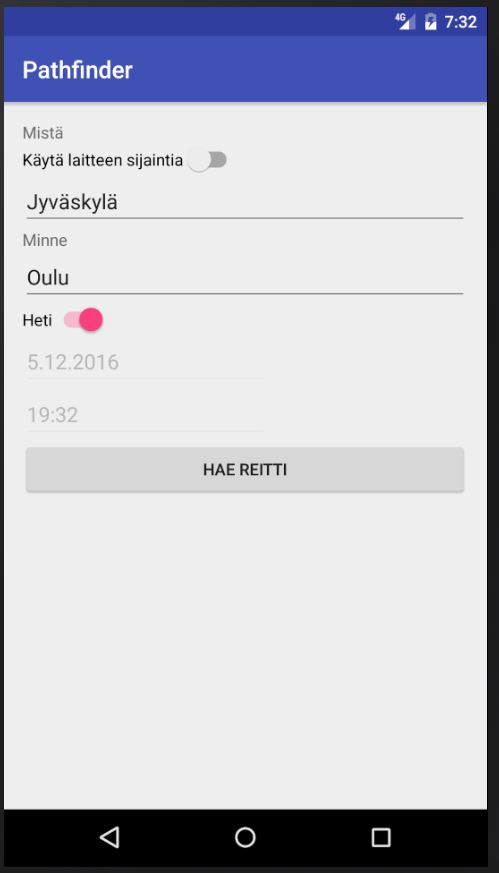


Figure 4:MainActivity

#### RoutePresenter

RoutePresenterin vastuulla on listata käyttäjälle reittivaihtoehdot joista valita. Sen käyttöliittymä sisältää RecyclerViewin, johon listataan reittivaihtoehdot fullRouteAdapterin (ks. 8.2.3.1 fullRouteAdapter) avulla valikoitavaksi. RoutePresenterin käyttöliittymään kuuluu myös romahtava toolbar, joka sisältää kartan reitin näyttämistä varten. RoutePresenterissä on suurin osa ohjelman logiikasta.

RoutePresenter perii AppCompatActivityn ja implementoi AsyncResponsen sekä AppDataInterfacen.

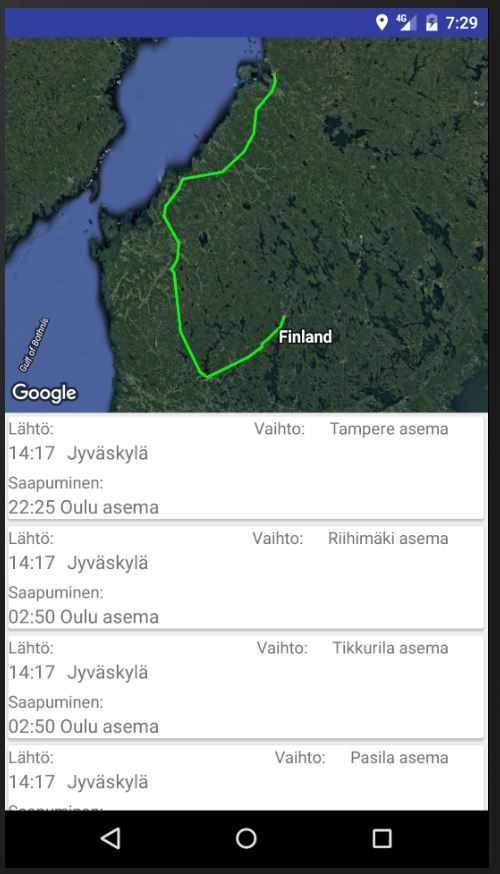


Figure 5: RoutePresenter

#### segmentPresenter

segmentPresenter on ohjelman viimeinen aktiviteetti. Sen vastuuseen kuuluu näyttää RoutePresenterissä (ks. 8.2.1.3 RoutePresenter) valitun reitin etapit RecyclerViewissä ja tarjota mahdollisuus käyttäjälle tarkastella reitin etappeja tarkemmin. segmentPresenter kertoo junaraiteen numeron, junan koodin ja aikataulut joita tulee noudattaa ehtiäkseen kyseiseen junaan.

segmentPresenter perii AppCompatActivityn ja implementoi AppDataInterfacen.

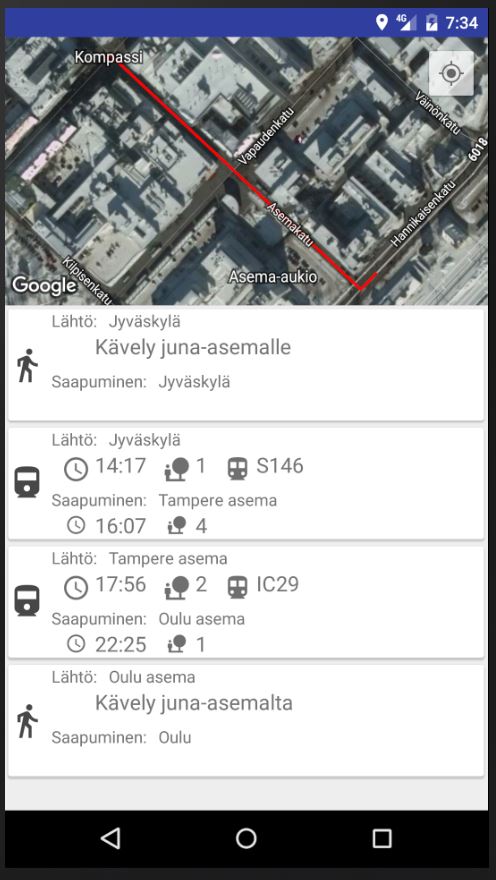


Figure 6: segmentPresenter

### Data

#### ApplicationData

ApplicationData syntyi tarpeesta siirtää isoja määriä tietoa aktiviteettien välillä. Kaikki luokan sisällä olevat muuttujat ja funktiot ovat staattisia jotta niihin pääsee käsiksi mistä vaan luomatta luokasta instanssia.

ApplicationDatan tehtävä on pitää sisällään ohjelmalle keskeistä informaatiota, joten luonnollisesti siihen oli järkevä kiinnittää myös laitteen sijainninseurantatoiminnallisuus ja keskeiset tiedot sijainnista. ApplicationDatan avulla saatiin rajattua sijaintia määrittävien yhteyksien määrä yhteen usean sijaintia tarvitsevan aktiviteetin ohjelmassa.

Luokkaan sisällytettiin muuttujaksi ApplicationDataCallbacks-luokan instanssi (ks. 8.2.2.2 ApplicationDataCallbacks), jonka avulla saadaan siirrettyä mille tahansa aktiviteetille callbackit asynkronisista sijaintiin liittyvistä muutoksista.

ApplicationData perii luokan Application.

#### ApplicationDataCallbacks

ApplicationDataCallbacks -luokan ainoa tarkoitus on toimia välikappaleena asynkronisten callbackien välityksessä aktiviteeteille. Se implementoi AppDataInterfacen (ks. 8.2.4.1 AppDataInterface), jolla callbackit voidaan välittää eteenpäin. Ilman tätä luokkaa jokainen luokka, joka haluaisi sijainteihin liittyviä callbackeja, joutuisi implementoimaan liudan erinäisiä luokkia saadakseen tarpeellisen toiminnallisuuden.

ApplicationDataCallbacks implementoi seuraavat luokat: GoogleApiClient.ConnectionCallbacks, LocationListener, GoogleApiClient.OnConnectionFailedListener, OnMapReadyCallback

#### AsyncJsonFetcher

AsyncJsonFetcher on luokka, mikä perii luokan AsyncTask ja siten ajetaan aina kutsuttaessa asynkronisesti välttyäkseen häiritsemästä varsinaisen käyttöliittymän toimintaa ja käyttömukavuutta.

Luokan tarkoitus on noutaa JSONArray mistä tahansa sellaisen palauttavasta rajapinnasta. Luokka sisältää funktiot asemien, suorien junayhteyksien sekä asemien aikataulutietojen noutamiselle. Kutsuttaessa parametreina annetaan haluttu funktio sekä kohderajapinnan osoite. Funktiot toteuttavat keskenään samanlaiset tiedonhaut, mutta noudon valmistuessa välitetään tieto jatkotoimenpiteistä JSONArrayn ohessa pääohjelmalle rajapinnan (ks. 8.2.4.2 AsyncResponse) kautta.

#### DataParser

DataParserin tehtävä on parsia kartassa näytettävä reitti rajapinnoista saaduista vastauksista.

#### FetchUrl

Luokan yksinkertainen tarkoitus on hakea annetusta osoitteesta http-kyselyllä vastaus ja palauttaa se.

#### LocationPermissionAgent

Luokan tarkoitus on toimia rajapintana jolta voidaan kysellä onko aktiviteetilla lupa toteuttaa aiottua toimenpidettä liittyen sijainteihin.

#### PermissionDialogFragment

Luokan vastuulla on tuottaa dialogi käyttäjälle siinä tapauksessa että pois päältä olevaa sijaintitoiminnallisuutta laitteessa ollaan käyttämässä. Dialogi ohjaa käyttäjän aktivoimaan sijaintipalvelut (ks. Figure 7).

PermissionDialogFragment perii luokan DialogFragment.

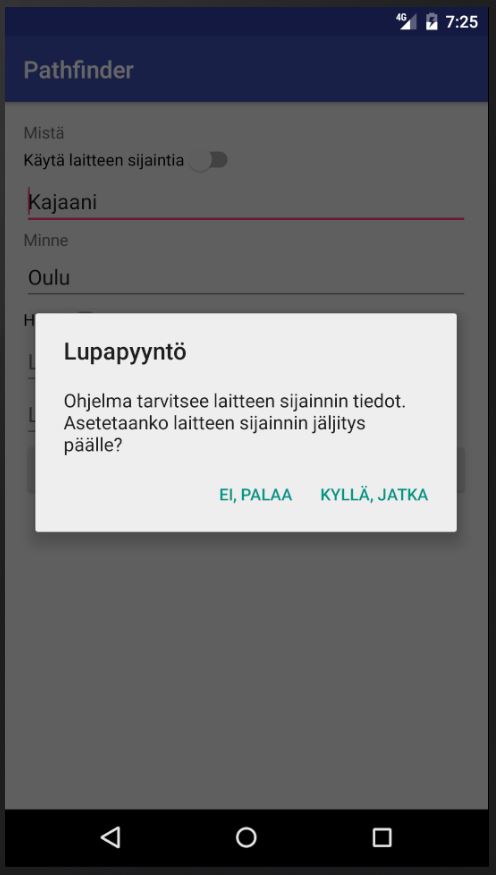


Figure 7: DialogFragment

#### Router

Router on vastuussa reittiohjeiden toimittamisesta kävelyreitteihin. Jokaisen reitin etappiobjektin, eli routeSegmentin, luonnin yhteydessä luodaan Router-objekti jolle annetaan tehtäväksi hakea asynkronisesti PolylineOptions-objekti routeSegmentille. Tieto saadaan perille RouterResponse-rajapinnan avulla.

Router perii luokan AsyncTask ja implementoi rajapinnan RouterResponse (ks. 8.2.4.3 RouterResponse).

### Adapterit

#### fullRouteAdapter

fullRouteAdapter on komponentti mikä ohjaa halutun datan, tässä tapauksessa valmiit reitit (ks. 8.2.5.5 fullRoute), käyttöliittymään korttinäkymään nähtäväksi. valmiit reitit sisältävät kaiken tarvittavan datan reittietappikorteille (ks. 8.2.3.2 routeSegmentAdapter), mitkä tulevat näkyviin käyttäjän valitessa jonkin piirretyistä korteista.

#### routeSegmentAdapter

routeSegmentAdapter piirtää valmiilta reiteiltä tulevan (ks. 8.2.3.1 fullRouteAdapter) datan valitun yksittäisen reitin etapeista käyttöliittymään korttinäkymään, missä käyttäjä voi tarkastella valitsemaansa reittiä vaihe-vaiheelta.

### Rajapinnat

#### AppDataInterface

AppDataInterface –rajapinta toimii ohjurina sijaintiin liittyville callbackeille. Rajapinnan avulla callbackit saadaan niitä kulloinkin tarvitsevalle aktiviteetille.

#### AsyncResponse

AsyncResponse -rajapinnan kautta palautetaan kaikki AsyncJsonFetcherin (ks. 8.2.2.3 AsyncJsonFetcher) noutamat JSONArrayt (suorat reittiyhteydet ja asemakohtaiset juna-aikataulut) routePresenter-aktiviteetille.

#### RouterResponse

RouterResponsen –rajapinnan avulla Routerin (ks. 8.2.2.8 Router) asynkronisesti hakemat tiedot saadaan ohjattua oikeille routeSegment-objekteille (ks. 8.2.5.6 routeSegment).

### Objektit

Suurin osa ohjelman noutamasta datasta avoimilta rajapinnoilta käännettiin objekteiksi ja niistä listaobjekteiksi, helpottaaksemme datan käsittelyä ja selkeyttä.

#### Station

Station on luokka jonka avulla saadaan talletettua asemilta saatavat tiedot selkeisiin rakenteisiin. Asemien tietoja tarvitaan reittien parsinnassa.

Sisältää tyypilliset data-objektin getterit ja setterit.

#### StationList

StationList-luokka luotiin tarpeesta selkiyttää parsintaa. Luokan kaksi toimintoa on palauttaa aseman koordinaatit syötettyä aseman koodia vastaan, sekä palauttaa aseman täysi nimi syötettyä aseman koodia vastaan.

StationList perii luokan Station (ks. 8.2.5.1 Station) ja toiminnot perustuvat vertailujen tekemiseen sisältämälleen Station-objektilistalle.

#### Train

Train -luokka muodostetaan juna-asemilta haetuista kulkevien junien tiedoista, jotta näistä tiedoista voidaan myöhemmin tehdä epäsuorien reittien haku. Luokkaan tallennetaan kaikki Digitrafficilta saatavissa oleva tieto yksittäisestä junasta. Luokka sisältää myös listan TrainTimeTables-objektista (ks. 8.2.5.4 TrainTimeTables), mikä on primäärinen vertailuperuste epäsuorille reiteille.

Luokka sisältää tyypilliset data-objektin getterit ja setterit.

#### TrainTimeTables

TrainTimeTables -luokka sisältää kaikki Digitrafficilta saatavat yksittäisen junapysäkin tiedot junan kulkemalla reitillä. Luokkaa käytetään listana Train -luokan (ks. 8.2.5.3 Train) sisällä.

Sisältää tyypilliset data-objektin getterit ja setterit.

#### fullRoute

fullRoute on luokka jonka on tarkoitus vastata reittiä aloituspaikasta määränpäähän. fullRoute sisältää listan routeSegmenttejä ja luokkaa käytetäänkin valmiiden reittien piirtämiseen (ks. 8.2.3.1 fullRouteAdapter) korttinäkymään.

Sisältää tyypilliset data-objektin getterit ja setterit.

#### routeSegment

routeSegment on luokka joka mallintaa etappia, eli reitin osaa. Se tietää määränpäänsä, lähtöpisteensä ja objektin jonka avulla reitti voidaan piirtää kartalle. routeSegmentillä on myös metodi piirtämistä varten.

Sisältää tyypilliset data-objektin getterit ja setterit.

## Toteutettu toiminnallisuus

Pathfinder tekee sitä mihin se suunniteltiin. Siihen voi syöttää osoitteet ja lähtöajan, joiden mukaan ohjelma hakee listan junatietoja hyödyntäviä reittejä käyttäjälle nähtäväksi.

Kaikki ensimmäisen prioriteettiluokan toiminnallisuudet on toteutettu ja ohjelma toimittaa asiansa joistain virhetilanteista tai puutteista huolimatta.

Toisen prioriteettiluokan toiminnallisuuksista saatiin mukaan vaihtoehtoisten reittien tarjoaminen ja kirsikkana kakun päällä on ohjelman tuki junan vaihtoihin kesken matkan määränpäähän pääsemiseksi.

Kooditeknisesti ohjelma on melko laaja suhteessa osaamiseen jakson alussa. Pathfinderiin on käytetty suurin osa kurssilla käydyistä asioista.

## Toteuttamaton toiminnallisuus

Suunnitelluista pois jäi linja-autojen ja lentokoneiden integroiminen reittihakuun sekä ohjelman toiminnan laajentaminen ulkomaiseen liikenteeseen. Lisää näistä aiheista kohdassa 11.5 Jatkokehitys.

## Luokkien eroavaisuudet suunniteltuihin

Ohjelma oli alunperin tarkoitus tuottaa demoksi TTOW0620 – Android Application Dev -kurssille vaadittavaa tutkimustyötä varten ja oli alunperin suunniteltu huomattavasti yksinkertaisemmaksi. Idean lopputuotteesta kehittyessä ja tavoitteesta laajentaa kehitystä TTOW0630 – Mobile Project -kurssille ohjelmassa muokattiin tai uudelleenrakennettiin monta asiaa.

Mobile Project -kurssin alussa oli vaikea nähdä kuinka demosta paranneltava ohjelma toimisi lopussa, joten ohjelmaa rakennettiin pala kerrallaan ja suunniteltiin ketterästi lyhyiden sprintien muodossa mitä tarvitaan seuraavaksi.

Ohjelmalle ei tehty kovinkaan kattavaa suunnitelmaa, joten erittely eroavaisuuksista lopputuleman ja suunnitellun välillä ei ole järkevää. Suuremmat muutostyöt kumpusivat ongelmista, joten niitä käydään läpi kohdassa 10.

# Laadunvarmistus ja testaus

Pathfinderin laadunvarmistus oli sisäisesti rakentunut kehittäjiensä haluun suorittaa kurssi kunnialla ja intoon koodata. Erillisiä laatuvastaavia ei projektiin määritelty.

Ohjelman testaamiseen ei tehty testausautomaatiota, sillä ohjelma itsessään oli mittakaavassaan valtava kahdelle opiskelijalle, joten testausautomaation pystyttämiseen ei haluttu käyttää resursseja. Pathfinderia siis ajettiin manuaalisesti muutosten jälkeen ja testattiin puhelimella tai emulaattorilla toimintaa.

# Ongelmat

Pathfinderin kehitys ei suinkaan ollut ongelmatonta.

## Ratkaistut ongelmat kehityksen aikana

Ongelmatilanteita oli useita. Alusta alkaen oli vaikea hahmottaa logiikkaa millä järjestelmä tuottaisi selkeän listan reittejä käyttäjälle. Kehityksen aikana päädyttiin myös erinäisten ympäristön rajoitteiden takia tekemään suuriakin muutostöitä.

Ensimmäinen ongelma oli se, että kurssilla käytiin osa materiaaleista läpi paljon myöhemmin kuin niitä olisi tarvittu projektissa. Asynkroniset haut julkisista rajapinnoista oli ensimmäisiä asioita mitä projektin eteneminen vaati. Tarvittiin myös Googlen karttarajapintaa, jonka läpikäynti kurssilla tapahtui myös useita viikkoja tarpeen syntymisen jälkeen. Ongelma ei ollut vakava, sillä internet oli pullollaan esimerkkejä ja ohjeita toteutusta varten - ongelmat ratkaistiin opiskelemalla.

Toinen ongelma ilmeni kun tehtiin useampia aktiviteetteja, joista useamman täytyi olla tietoinen puhelimen sijainnista ja toimia kartan kanssa. Tällaisenaan ohjelmalla olisi useita yhteyksiä sijaintipalveluun, koska jokainen aktiviteetti pitäisi omaa lukua laitteen sijainnista. Se vaati myös duplikaatteja callback-funktioita joita oli jo implementoitu toisiin aktiviteetteihin. Ei varsinainen ongelma koodin toiminnassa, mutta kiusallinen särö eleganssissa ja tehokkuudessa.

Kun ilmeni että ohjelman täytyy kyetä siirtämään suuria määriä objekteja ja dataa toisilleen eikä intentin bundleen voinut sisällyttää kerrallaan useita objekteja kätevästi, ryhdyttiin muutostöihin.

Syntyi ApplicationData staattisilla muuttujilla, funktioilla ja rajapinnan avulla aktiviteeteille uudelleenohjatuilla callbackeilla. Näin saatiin rajattua sijaintitietoja ylläpitävä yhteys yhteen kappaleeseen ja aktiviteettien välillä oli vaivatonta kuljettaa halutunlaista dataa ja ratkaistiin ongelmat yhteyksien ja datansiirron kanssa.

## Yhä olemassalevat ongelmat ja niiden ratkaiseminen

Googlen rajapintoihin on 2500 kyselyn päivärajoite ilmaiskäytössä. Pathfinder tekee nykyisessä muodossaan kyselyitä löytyneiden reittien määrän kerrottuna kahdella, sillä jokaiselle reitille haetaan kävelyosuus lähtöasemalle ja toinen kohdeasemalta kohteeseen, jolloin kyselyjä voi tulla pahimmassa tapauksessa satoja. Ongelma ei yhden ihmisen käytössä ilmene, mutta jos ohjelman laittaa levitykseen niin käyttäjäkunnan kasvaessa syntyy nopeasti tilanne missä raja tulee vastaan. Pathfinderiin voisi laittaa mainokset joilla rahoittaa suurempi kyselykatto, mutta ongelmaa tulisi ensin lähestyä optimoimalla jo haetun datan käyttöä. Ohjelma voisi tutkia onko kyseinen reitti jo haettu ja käyttää kerran haettua reittiä uudelleen.

Collapsing toolbar ja sen sisäinen kartta vaikutti ensin hyvältä idealta, kun listattujen reittien määrä oli pieni. Myöhemmin reittejä alkoi syntymään enemmän epäsuorien reittien implementoinnin myötä ja piilotettavassa kartassa ilmeni kiusallinen ongelma käyttäjälle, jossa klikattu reitti saattaa työntyä näkymättömiin kun toolbar laajentuu. Collapsing toolbar luultavasti tullaan hävittämään ja ohjelmassa siirrytään staattisen karttafragmenttin käyttöön, mutta toistaiseksi collapsing toolbar jätettiin toimintaan arvioinnin ajaksi, jotta tehty työ näkyisi.

Aikarajoitteiden takia osa ohjelman logiikasta jäi pilkkomatta järkevästi. Reittiobjektien luonti ja varsinainen reitinetsintä on puuroutunut keskenään, eivätkä funktiot kuvasta välttämättä oikein niiden koko toimintaa. Pilkkominen olisi täysin toteutettavissa, mutta aikaavievää eikä sitä ehditä toteuttaa ennen harjoitustyön palautusta.

Välillä, kahden kehittäjän tehdessä innokkaasti töitä projektin eteen, huomattiin että oltiin tehty samoja asioita toimittavia funktioita ja luokkia. Ongelma eleganssissa, mutta yhdentekevää koodin toimivuuden kannalta. Korjaustyö olisi suhteellisen helppoa, mutta kuitekin aikaavievää toteuttaa ja se rajattiin pois rajallisen harjoitustyöhön käytettävän ajan takia.

Ohjelma tuottaa usein duplikaatteja reittivaihtoehtoja, joita ei olla vielä saatu hävitettyä. Tehtyjä reittiobjekteja voitaisiin vertailla ja parsia kopiot pois valmiiden reittien listasta, mutta on haluttavampaa selvittää ensin miksi niitä syntyy ja sitten päättää mikä on paras tapa lähestyä ongelmaa.

Jostain syystä ohjelma ei osaa määrittää laitteen sijaintia aina oikein sisätiloissa. Ulkona sijainti löytyy kuitenkin nopeasti. Vika on selkeästi wifiverkkojen perusteella toimivassa sijainnin päivittämisessä ja se tullaan korjaamaan.

## Tunnetut virhetilanteet

Käyttäjän valitsemasta lähtöajankohdasta puuttuu syötteen validointi. Mikäli käyttäjä syöttää haluamansa päivämäärän tai kellonajan väärässä formaatissa, ohjelma ei osaa tulkita sitä ja kaatuu. Validointi olisi nopea tehdä, mutta prioriteetiltaan hyvin matala.

Digitrafficin suorien junayhteyksien haun rajapinta on kaatunut tai ajettu alas satunnaisesti projektin aikana. Tällöin rajapinnasta saatu vastaus on joko tyhjä JSONArray, tai jonkinlainen virheviesti. Ohjelma osaa huomata tämän ja hylätä suorien reittien vaihtoehdot. Epäsuorien reittien vertailua varten käytettävä juna-aseman aikataulutietorajapinta toimi koko projektin ajan moitteettomasti, joten virhetilanteita luotiin antamalla rajapinnalle virheellisiä parametreja. Virheellisen vastauksen perusteella osasimme tehdä ohjelmasta tällaisia tilanteita sietävän.

# Analyysi

Mitä opittiin, mitä tulisi opetella lisää, suurimmat haasteet, mikä olisi pitänyt tehdä toisin, jatkokehitys, kommunikointi, vastuunjako ja toteutukseen kuluneet resurssit.

## Mitä opittiin

Kummallekkaan ei ollut Android-ympäristö tuttu jakson alussa ja siitä on tultu pitkälle. Joitain asioita, mitä harjoitustyöhön sisällytettiin, ei käyty kurssilla ollenkaan vaan ne opeteltiin itse ja implementoitiin ohjelman arkkitehtuurin ja rakenteen parantamiseksi.

Asynkroniset funktiot, interfacet ja callbackit olivat aikaisemmin tuntemattomia tai vähemmän tunnettuja asioita ja niiden toiminta opeteltiin harjoitustyön aikana.

## Mitä tulisi opetella lisää

Android-ohjelmien arkkitehtuurin näkemys ja suunnittelu kaipaisi hiomista. Ympäristö asettaa joitain rajoitteita joita Androidille uutena kehittäjänä ei osannut odottaa, jolloinka ongelmien kiertäminen saattoi vaatia suuriakin muutostöitä.

Jotta säästettäisiin aikaa jatkossa, olisi oleellista saavuttaa selkeämpi kuva järkevistä arkkitehtuuriratkaisuista. Tätä ei kuitenkaan ole järkevää yrittää opetella erikseen seuraavaa projektia varten, vaan on yksinkertaisesti toteutettava uusia projekteja ja hahmotettava suuri kuva koostamalla se useista pienistä. Osaaminen syntyy tekemisestä ja suuri osaaminen suuresta määrästä tekemistä.

## Suurimmat haasteet

Aika suhteessa työn määrään oli lyhyt, sillä molemmilla Pathfinderin kehittäjillä oli 44:n opintopisteen edestä kursseja jakson aikana. Projektin loppuunsaattaminen tavoitteiden puitteissa vaati ponnistelua.

Koska kehityksen alussa ei ollut selkeää kuvaa kuinka ohjelma toimisi, ohjelmaa ja samalla mielikuvaa sen toiminnasta rakennettiin pala palalta. Tämä johti arkkitehtuurillisiin ongelmiin ja korjaustöihin joihin ei ollut aina varaa käyttää aikaa.

## Mikä olisi pitänyt tehdä toisin

Usean eri aktiviteetin sijaan olisi kannattanut käyttää fragmentteja, jotta ohjelma olisi ollut kevyempi. Tähän ratkaisuun törmättiin myöhässä, joten katsottiin että asiaa koskevat muutostyöt rajataan toistaiseksi pois, kirjataan ylös ja toteutetaan myöhemmin kun siihen on aikaa.

## Jatkokehitys

Toteuttamaton suunniteltu toiminnallisuus on hyvä suunta projektin jatkokehitykselle. Lisäksi kaikki yhä olemassa olevat ongelmat -osiossa mainitut aiheet tulisi ratkaista, arkkitehtuuria kannattaisi hioa selkeämmäksi sekä nimetä ja pilkkoa funktioita paremmin.

Reittihaku kykenee tällä hetkellä hakemaan vain suorat reitit tai yhden vaihdon reitit. Jotta Pathfinder voisi kasvaa täyteen potentiaaliinsa, sen on kyettävä hakemaan kaikki reitit riippumatta vaihtojen määrästä ja parsimaan niistä relevantit reitit käyttäjälle.

Ohjelman aktiviteettien määrä tullaan laskemaan yhteen. Erilaiset näkymät voitaisiin hoitaa fragmenteilla jotka ovat huomattavasti kevyempiä ajaa. Tällöin ei myöskään tarvittaisi erillistä luokkaa siirtämään dataa aktiviteettien välillä tai hallinnoimaan yhteyttä GoogleMaps-rajapintaan, jolloin arkkitehtuuri yksinkertaistuu ja selkeytyy.

Käyttöliittymää voi parannella lukuisilla muutoksilla. Esimerkiksi tarkasteluun valittu, mutta ei kuitenkaan vielä seuraavaan aktiviteettiin viety reittivaihtoehto voisi värjääntyä eri väriseksi, jotta käyttäjälle näkyisi jatkuvasti mikä reitti on se joka näkyy kartalla.

Kun nykyiset ongelmat on korjattu ja toimintaa hiottu, ohjelmaa pyritään laajentamaan myös muihin kulkuvälineisiin kuten laivoihin, juniin, lentokoneisiin ja ehkä myös majoitukseen. Käyttäjämaksuista ei olla vielä päätetty, mutta rahoitus mainonnalla vaikuttaa vahvalta optiolta.

## Kommunikointi

Pääosin ohjelmiston kehityksenaikainen kommunikointi hoidettiin Mumblella, avoimen lähdekoodin alustariippumattomalla VoIP-ohjelmistolla, sekä Steam-ohjelmistoon sisäänrakennetulla viestintäalustalla. Pidimme myös lähipalavereita ja suunnitteluhetkiä lähes jokaisena tiistai- ja keskiviikkoaamuna TTOW0620- ja TTOW0630 -kurssien kontaktitunneilla, sekä erikseen sovittaessa myös muina aikoina projektitiloissa.

## Vastuunjako

Tarkkaa vastuunjakoa ohjelman kehityksessä on vaikea eritellä ja rajata, koska ohjelma on tiukasti verkostoitunut ja moni luokka/funktio riippuvainen useasta muusta luokasta/funktiosta samaan aikaan. Tämä synnytti tarpeen työstää useita luokkia ja funktioita yhteisvoimin, jonain päivinä vuorotellen kehitysvuoroa ja jonain parikoodaten. Listattakoon kuitenkin karkeasti jaoteltuja aihealueita joiden kanssa kehittäjät työskentelivät.

Miika:

* Pääohjelman latausruutu.
* Käyttäjän syötteen välitys.
* Ajankohtaisen asemalistan nouto.
* Asynkroniset funktiot JSON-datan noudolle Digitrafficin rajapinnasta, rajapinta datan palautukselle.
* Handler kuuntelemaan kaikkea threadeista palaavaa dataa.
* Noudetun JSON-datan parsiminen ja muunto Train- ja TrainTimeTables -objekteiksi.
* Suorien reittien etsintä ja vertailu, tallennus täydelliseksi reittiobjektiksi etappilistaobjekteineen.
* Epäsuorien reittien etsintä ja vertailu, tallennus täydelliseksi reittiobjektiksi etappeineen.
* Valmiiden, löydettyjen täydellisten reittien ja reittietappien rakenteen suunnittelu ja kehittäminen objekteiksi.
* Kokonaisreittien korttinäkymän layout ja adapter.
* Reittietappien korttinäkymän layout ja adapter.
* ProgressDialog ja sattumanvaraisten tekstien palautus dialogiin.
* Loppudokumentaatiossa avustaminen
* Yleinen debuggaus

Janne:

* Datan siirtäminen aktiviteettien välillä ApplicationDatan avulla
* Aktiviteettien sijainninseurannan keskittäminen yhteen yhteyteen
* Arkkitehtuurin suunnittelu ja muokkaus kehityksen aikana
* Kartat ja niiden toiminnallisuus
* Laitteen paikannus
* Junareittien koordinaattien tuotto ja piirto kartalle
* fullRoutePresenterin RecyclerViewin ItemTouchHelper ja sweep-toiminnot
* Lupien tarkistus ajon aikana
* Dialogi laitteen seurantaluvan pyytämiselle
* Asynkroninen datan haku Googlen rajapinnoista
* Romahtava ActionBar routePresenterissä
* Loppudokumentaatio
* Käyttöliittymän hiominen
* Yleinen debuggaus

## Toteutukseen kuluneet resurssit

Miika:

|  |  |  |
| --- | --- | --- |
| Pvm | Tunnit | Kuvaus |
| 13.09 | 6h | JSON-parsinnan testaus ja datan tallennus, Asemadatan nouto Digitrafficin rajapinnasta. Layout-kehitys. |
| 15.09 | 7h | Asemadatan muutokset, AsyncTaskin luominen JSON-datan noudolle. Junayhteyksien selvitystyö. |
| 16.09 | 8h | Junayhteyksien nouto Digitrafficilta, yhteyksien Bundlaus, napinpainalluksella asetettava nykyinen päivämäärä ja kellonaika. |
| 17.09 | 5h | Päivämäärän konversio käyttäjälle sopivammaksi ja SimpleDateFormatiin. Aikataululistojen siistiminen. Käyttäjän valitseman ajan yhdistämistä aikatauluihin. Koodin siistintä. |
| 18.09 | 5h | Android Calendar, puutteiden korjaus. Koodin siistintä. Valitun ajan yhdistämisen aikatauluihin käyttöönotto. Ohjelmiston rakenteen uudelleenjärjestelyä. Aikataululistojen tallennuksen muutoksia. |
| 20.09 | 4h | Ohjelmiston rakenteen suunnittelua ja UML-kaavioita. |
| 05.10 | 3h | AsyncTask monitoriksi muille AsyncTaskeille, seuraamaan muiden taskien ajonaikaista edistymistä (Myöhemmin muutettu Handleriksi). Funktion luonti joka hakee lähimmän aseman sijainnin annettujen koordinaattien suhteen. |
| 06.10 | 4h | Ohjelmiston toiminnallisuuden suunnittelua ja UML-kaavioita. |
| 11.10 | 4h | Projektipalaveri, Erillisten branchien merge yhdeksi. |
| 12.10 | 2h | Lisää mergeä, ongelmia. |
| 23.10 | 3h | Lisää mergeä, ongelmia. |
| 26.10 | 3h | Lisää mergeä. AsyncTaskeissa ilmenneiden puutteiden korjaus. ProgressDialogin luominen ja käyttöönotto sattumanvaraisine teksteineen. |
| 28.10 | 0,5h | Kortin luonti kokonaisreiteille. |
| 01.11 | 5h | Kokonaisreittikortin layout, rakenteen suunnittelua tukemaan reittietappeja. RecyclerView. |
| 02.11 | 0h | TODO-listan luonti. |
| 15.11 | 6h | Reittiobjektin suunnittelu ja luonti, noudettujen reittien muutos ArrayListista Listaobjekteiksi. |
| 16.11 | 9h | Jatkettu reittiobjektin kehitystä ja muutosta. Kokonaisreittiobjektin luonti + parcelointi, reittietappiobjektin luonti + parcelointi + listaaminen kokonaisreittiobjektin sisään. RoutePresenterin rakennemuutoksia. Adapteri reittietappien korttinäkymälle + korttien layout + aktiviteetti näiden esittämiselle. Koodin siistiminen. Branchien merge. |
| 18.11 | 4h | Yhteistä suunnittelua, koodin debuggausta. |
| 22.11 | 6h | Juna-objekti. Aikatauluobjekti ja listaus junaobjektiin. AsyncTaskin paluuarvojen tulkinta turhan datan karsimiseksi ja datan tallentaminen junaobjekteiksi sisäisine aikatauluineen. |
| 23.11 | 8h | Epäsuorien reittien vertailun luomista. |
| 24.11 | 7h | Kopiointiongelman korjaus fullRouteListissa. |
| 28.11 | 0h | Kadonneiden UML-kaavioiden palautus Githubiin. |
| 30.11 | 9h | Epäsuorien reittien vertailun kehitys ja piirto korttinäkymään. Epäsuorien listojen noudon rajapinnasta aikaistus. |
| 01.12 | 3,5h | Loppudokumentointia ja ongelmanratkaisua. |
| 02.12 | 8h | Loppudokumentointia, splash screen, korttinäkymän layoutin parannukset. |
| 03.12 | 3h | Korttinäkymän layoutin parannukset. |
| 04.12 | 2h | Layoutmuutoksia. |
| 05.12 | 3h | Dokumentointia. |
|  | 128h | Yhteensä |

Figure 8: Miikan työtuntitaulukko

Janne:

|  |  |  |
| --- | --- | --- |
| Pvm | Tunnit | Kuvaus |
| 14.09 | 4h | JSON:n tiedoston hakua ja parsintaa |
| 14.09 | 7h | Routen kehitystä |
| 16.09 | 6h | Reitin piirron kehittämistä ja asynctaskien rakentelua |
| 18.09 | 4h | Laitteen lokaation liittäminen ohjelmistoon |
| 19.09 | 3h | Laitteen lokaation liittäminen ohjelmistoon |
| 05.10 | 6h | Arkkitehtuurin muokkaus |
| 11.10 | 2h | Projektipalaveri, erillisten branchien merge yhdeksi. |
| 28.10 | 10h | ApplicationData- ja ApplicationDataCallbacks -luokkien luonti ja integrointi |
| 29.10 | 3h | MapFragmentin käyttöönotto ja collapsingtoolbar |
| 29.10 | 5h | RoutePresenterin layoutin korjailu ja ApplicationDatan tehokkaampi hyödyntäminen |
| 30.10 | 9h | Reittiobjektien luonti ja piirtäminen kartalle demoa varten |
| 08.11 | 2h | Aktiviteetin sulkeutuminen dialogissa ja nullpointer googleapiclientissä korjattu |
| 19.11 | 6h | Reittien piirto kartalle |
| 20.11 | 9h | Swipe-toiminnallisuus recyclerviewiin, bugfixejä |
| 22.11 | 4h | Laitteen reagointi orientaatiomuutokseen, korjailtu reittien näyttämistä kartalla |
| 28.11 | 5h | Loppudokumentaatio ja luokkakaavio |
| 29.11 | 5h | Loppudokumentaatio ja aktiviteettikaavio |
| 30.11 | 2h | Monipolvisten reittien debuggaus |
| 30.11 | 6h | Loppudokumentointi |
| 31.11 | 2h | Loppudokumentointi |
| 01.12 | 9h | Epäsuorien junareittien polylinet |
| 02.12 | 2h | Epäsuorien junareittien polylinet |
| 03.12 | 8h | Epäsuorien junareittien reitinhaku ja UI |
| 04.12 | 1h | Koodin siivous |
| 04.12 | 4h | Loppudokumentointi |
| 05.12 | 6h | Loppudokumentointi, ohjelman hiominen ja powerpoint |
|  | 130h | Yhteensä |

Figure 9: Jannen työtuntitaulukko

# Arviointi

Harjoitustyön idea oli hyvä, toteutus näyttää siistiltä ja sen käyttäminen on selkeää. Dokumentointi on myös kattava ja helposti ymmärrettävä kenelle tahansa ohjelmointiin vihkiytyneelle. Aihealue oli uusi molemmille kehittäjille ja siihen suhteutettuna tuotos on erinomainen, joten poissulkien joitain ongelmia arkkitehtuurissa ja tehokkuudessa on turvallista sanoa että siihen ollaan tyytyväisiä.