<Korona-App>

Tekijä: Tuomas Salomäki

Kuvaus ohjelmasta

Korona-App on android-sovellus, jonka avulla käyttäjä pääsee käsiksi THL:n rajapintaan, ja THL:n tarjoamaan avoimeen dataan, jota on tuotettu Suomessa korona virukseen liittyen. Sovellus hyödyntää datan rajaamisessa THL:n tarjoamia rajausehtoja, kuten sijainti, ikä ym. Sovelluksen avulla käyttäjä pystyy piirtämään kuvaajia, tai tarkastelemaan tartuntamääriä tietyillä menneillä ajanjaksoilla, ja rajaamaan hakua tietyillä ehdoilla. Käyttäjä pystyy tallentamaan kuvaajia sovellukseen, ja tarkastelemaan niitä jälkikäteen.

Ohjelma sisältää viisi päänäkymää: Etusivu, tilastot, kuvaajan piirtäminen, sekä kuvaajan avaaminen. Pääsivulla on näkyvillä lyhyt kuvaus ohjelman toimintaperiaatteesta. Tilastot-sivulla käyttäjä pystyy tarkastelemaan tartuntamääriä, tai muun valitsemansa mittarin dataa ennalta määritellyillä aikajaksoilla nykyhetkestä taaksepäin. Kuvaaja-sivulla käyttäjä pystyy luomaan viivakuvaajan, sekä halutessaan tallentamaan sen. Käyttäjällä on käytössään samat rajausehdot, kuin tilastot-näkymässä, mutta tämän lisäksi käyttäjä pystyy vapaasti valitsemaan ajanjakson, jolta dataa haetaan. Asetukset-sivulla käyttäjä pystyy vaihtamaan sovelluksen kielen. Tallennetut kuvaajat -kohdasta käyttäjä pystyy aukaisemaan aiemmin tallentamiaan kuvaajia.

Ohjelman toteutus

Ohjelman minimivaatimus on API23: Android 6.0, ja sovellus toimii 94 %:lla laitteista. Kuvaajan piirtämisessä käytettiin MPAndroidChart -kirjastoa, joka mahdollistaa erittäin monipuolisen datan esittämisen. Tässä työssä oli ajan puutteen vuoksi mahdollisuus toteuttaa vain viivadiagrammin piirtäminen. JSON-tiedostojen tallentamiseen käytettiin Gson-kirjastoa, mikä mahdollistaa tiedon tallentamisen erittäin helposti. Tiedostojen lukemisessa käytettiin sekä googlen Gson- että org-son Json- kirjastoja. Sovellus toteutettiin Andoid-Studio -ohjelmalla. Testauksessa käytettiin Android-Studion työkaluja, kuten debuggeria, ja tietojen tulostusta. Ryhmätyökaluna käytettiin GitHubia.

Luokkakaaviot

Kaaviossa UML-1 on esitetty ohjelman pääluokan, eli MainActivity-luokan alle sijoittuvat fragmenttien pääluokat. Luokat sisältävät niin paljon toiminnallisuutta ja alaluokkia, että vain osa tiedoista on luokkakaaviossa mukana. Tekstikentät ja muut komponentit on esitetty vain yhtenä muuttujana per luokka, vaikka niitä olisi useampia, jotta kaaviota saadaan selkeytettyä.

Kaaviossa UML-2 on esitetty ThlObjects-luokka, jonka avulla THL:n tarjoama JSON-data muutetaan olioiksi. ThlObjects on ns. pääluokka, joka voi säilöä sisälleen usita ThlObject-olioita. ThlObjec -luokka on toteutettu pääluokan sisällä. Se sisältää id-tunnisteen, sekä label-kentän, joka sisältää jonkin nimen dataluokalle. Lisäksi se sisältää ArrayListin, joka koostuu Children-olioista. Children-luokka on toteutettu ThlObject-luokan sisällä. Children olio sisältää kaikki arvot, joita eri olioilla THL:n datassa on. On myös mahdollista, että Children-luokka sisältää Children-luokan, eli luokka sisältää itsensä. Tällöin Children-luokka kutsuu itseään. THL:n data voi sisältää useita sisäkkäisiä kerroksia, mutta luokan kutsuessa itseään se osaa ottaa huomioon myös kaikki sisemmätkin luokat.

Toteutetut ominaisuudet

Ominaisuus	Perustelut	Pisteet
Olio-ohjelmoitu	Pakollinen	Pakollinen
Väh 5 luokkaa	Pakollinen	Pakollinen
API	THL rajapinta	Pakollinen
JSON-formaattiin tallennus	Kuvaajien tallentaminen JSON-muodossa	Pakollinen
Tiedostojen lukeminen	Käyttäjä voi tarkastella tallentamiaan kuvaajia	Pakollinen
UI-komponentit	Ohjelmassa on 5 päänäkymää, jotka kukin sisältävät lisänäkymiä	5
Asynkronisten HTTP-kutsujen käyttö dataa haettaessa	Ohjelma hakee dataa useilla eri http-kutsuilla saman aikaisesti, jolloin asynkronisilla kutsuilla nopeutetaan ohjelman toimintaa	2
Fragmenttien käyttö	Kaikki ohjelman eri näkymät on toteutettu fragmenttien avulla	2

Kielen valinta	Käyttäjä voi vaihtaa sovelluksen kielen asetuksista	2
Kuvaajan piirtäminen	Kuvaajadatan piirtämisessä hyödynnettiin MPAndroidChart -kirjastoa. X-akselille sijoitetaan aina aikaväli, jonka käyttäjä valitsee kalenterikomponentista. Y-akselille sijoitetaan jokin THL:n rajapinnasta löytyvistä mittareista, kuten tartuntamäärät. Lisäksi dataa on mahdollista rajata muiden ehtojen, kuten sukupuolen tai iän mukaan.	5
THL-rajapinnan dimensiot ja mittarit	THL-rajapinta tarjoaa datan rajaukseen tietyt työkalut, joita ovat esimerkiksi sijainti, ikä, sekä tartuntamäärät. Kaikki käytettävissä olevat rajausehdot ovat saatavilla THL:n sivuilla. Sovellus hakee kaikki rajausehdot rajapinnasta, ja luo näistä oliot. Näitä olioita hyödynnetään esimerkiksi spinner-komponenteissa, joiden avulla määritellään haettava data. THL:n JSON-data sisältää useita identtisiä, mutta sisäkkäisiä luokkia, joten ohjelmassa luokat toteutettiin niin että ne kutsuvat tarvittaessa itseään.	5
Kuvaajadatan tallennus recyclerview komponenttiin	Kuvaajia on mahdollista tallentaa recyclerview komponentteihin, jolloin niiden selaaminen on helppoa. Käyttäjä voi antaa tallennettavalle datalle nimen, joka näkyy kortin otsikkokentässä. Lisäksi kortissa näkyy päivämäärä, jolloin kuvaaja on tehty.	3
Koronatilastot	Ohjelman avulla on mahdollista tarkastella ennalta määritettyjen aikavälien tartuntamääriä tilastot-välilehdellä. Aikavälit ovat 7, 14, 30 ja 100 päivää. Käyttäjä voi THL:n rajausehtojen mukaan rajata dataa.	3
Summa		27+13=40

Työmäärät

Tekijä	Tehtävät	Tunnit
	THL:n datan rajaukseen käytettävät oliot	25
	THL:n datan muuttaminen olioiksi	20
	Tilastonäkymä	10
	Kuvaajanäkymä	15
	Tietojen tallennus ja lukeminen	10
	Kieliasetukset	3
	Recycler-view	7
Summa		90

Mitä opin harjoitustyöstä?

- THL:n käyttämä datan rakenne, ja http-pyyntöjen rajaukset olivat aluksi erittäin vaikeaselkoisia, mutta ajan myötä rupesivat selkiytymään. Erittäin paljon työtunteja kului muutenkin JSON-datan käsittelyyn. Tämän työn myötä JSON datan käsittely on kohtalaisen hyvin hallussa.
- Ohjelman toteuttaminen ja datan käsittely olio-ohjelmointia käyttäen opetti lähestymään ohjelmointiongelmia entistä paremmin olio-ohjelmointiin kuuluvalla ajattelutavalla, jossa tehtävät ja vastuut jaetaan pieniin yksiköihin. Tässä vaiheessa voisi toki löytää työstä useita kohtia, joissa olio-ohjelmointia olisi voitu hyödyntää vielä tehokkaammin.
- Ohjelman laajentuessa kommentoinnista alkaa olla selkeää hyötyä
- Ohjelman rakennetta kannattaa harkita tarkkaan, jotta pysyy itse kärryillä ohjelman toiminnasta, tästä syystä myös luokkakaaviot ovat hyvä työkalu

Palaute harjoitustyöstä (vapaaehtoinen)

- Harjoitustyö oli ainakin yksin toteutettuna valtavan aikaa vievä
- Haastavin osuus oli ymmärtää kuinka THL:n rajapinta oikein toimii, kuinka dataa tulkitaan, ja miten se saadaan muutettua olioiksi.
- Kaikkiaan erittäin opettavainen projekti