**Dionys App**

Android TTOW0620 ja Mobile Project TTOW0630 Projekti

Aleksi Tommila

Atro Lähdemäksi

Jälkiraportti

Joulukuu 2016

Ohjelmistotekniikan koulutusohjelma

Contents

[1 Esittely 1](#_Toc468921924)

[2 Tavoitteet 1](#_Toc468921925)

[3 Saavutukset 2](#_Toc468921926)

[4 Ongelmat 3](#_Toc468921927)

[5 Suunnittelu, työnjako ja aikataulutus 4](#_Toc468921928)

[6 User storyt 6](#_Toc468921929)

[7 Käyttötapaukset 7](#_Toc468921930)

[8 Käyttöpolku 8](#_Toc468921931)

[9 Käyttöliittymämockit 9](#_Toc468921932)

[10 OOP 12](#_Toc468921933)

[11 UML Luokkarakenne 12](#_Toc468921934)

[12 Tietokantaratkaisu 14](#_Toc468921935)

[13 Loppusanoma 15](#_Toc468921936)

# Esittely

Kreikkalaisen viinin ja juhlien mukaan kehitysnimen saanut Dionys applikaatio lähti Android TTOW0620 kurssin tutkimusaiheenvalinnasta. Tarkoituksena oli siis perehtyä miten voidaan android-applikaatio liittää ulkopuoliseen tietokantaan ja miten voidaan hakea ja tallentaa tietokantaan dataa GPS-sijainnin perusteella.

Tutkimuksen ideaksi käytännössä, tuli applikaatio, joka listaa paikallisen pitäjän julkisia kohteita, kuten esimerkiksi yökerhoja, ja niissä olevia rekisteröityneitä käyttäjiä. Applikaatio vertaa henkilön tämänhetkistä sijaintia etätietokannasta tulevien kohteiden koordinaatteihin ja lisää itsensä tietokannassa tässä kohteessa oleviin henkilöihin jos koordinaatit ovat tarpeeksi lähellä toisiaan.

Koska androidissa ei ole sisäänrakennettuja funktioita yhteyden muodostamiseen etätietokantaan, otimme ASP.NET kurssin harjoitustyöksi kirjoittaa android projektiamme palvelemaan restful api-toteutuksen tietokannan ja android applikaation väliseksi rajapinnaksi.

Etätietokanta saatiin Azuren pilvestä opiskelijalisenssillä, samoin rest-apin web applikaatioslotti.

# Tavoitteet

Aplikaation idea lähti Android kurssilla siitä, miten mielenkiintoista voisi olla tutkia käyttäjän liikkeitä ja hallinnoida etätietokantaa pelkästään GPS datan pohjalta. Päätimme kuitenkin muotoilla ja tehdä aiheesta sellaisen, jolla on potentiaalia ominaisuuksiltaan olla mobiiliyhteisöä kiinnostava palvelu. Tämä tarkoittaa käyttännössä sitä, että ohjelman valmistumien edelltyttää opintojakson kestoa ja laajuutta paljon suuremman työpanoksen ja pidemmän kehityskaaren. Tästä syystä suunnittelimme projektista vain välttämättömimmät core-ominaisuudet puristettavaksi opintojakson mittaan ja valmistuviksi demoon mennessä. Rajasimme koko aplikaation kehityskaaren kannalta vähemmän oleelliset asiat viimeiseksi ja jos-ehtii tyyppisiksi.

Asiat joita halusimme saada demoon mennessä valmiiksi, oli kohteiden haku etätietokannasta rest rajapinnan läpi, käyttäjän gps sijainnin haku ja vertaaminen tietokannasta haettuiden kohteiden koordinaatteihin, käyttäjän lisääminen tiettyyn kohteeseen jos koordinaatit olivat tarpeeksi lähellä toisiaan ja kohteissa olevien käyttäjien selaaminen omasta sijainnista riippumatta. Nämä kaikki saatiin toteutettutua.

Jos-ehtii ominaisuuksiin luettiin pikaviestintä, UI-elementtien koristelu ja datan näyttö loogisesti ja käyttäjäystävällisesti, rest-apin api-key-implementaation hyödyntäminen eli restin käyttöoikeuden verifiointi clientti tasolla, sekä käyttäjätilien rekisteröinti ja verifiointi.

# Saavutukset

Presentaatio ja palautuspäivää varten ennalta-asetetut tavoitteet saavutettiin hienosti. Core-ominaisuudet tutkimus ja toteutusteemoista toimivat niinkuin pitää. Tietokannasta haetaan kohteita, niissä sijaitsevat käyttäjät löydetään oikein, GPS paikannus ja vertailu toimii ja lisää onnistuneesti käyttäjän oikeisiin kohteisiin. UI-koreilua datan näyttämiseen ehdittiin tekemään sen verran yli varsinaisten tavoitteiden, että kohteet tietoineen listataan varsinaisessa korttinäkymässä, lisättiin gesture-pohjainen navigointi, sekä fragmenttipohjainen käyttöliittymä.

Presentaation kannalta applikaatio oli tietysti laihaa viihdettä UI:n matalasta prioriteesti johtuvan keskeneräisyyden vuoksi. Projektiryhmänä olimme keskittyneet enemmän varsinaisen toiminnallisuuden dynaamiseksi ja mahdollisimman joustavaksi saamiseen, että laiminlöimmen presentaation näkökulmasta UI:n toteutuksen. Vaikka UI:n toteutus olikin alhaisin prioriteettimme ennen muista varsinaisia jatkokehitysominaisuuksia, näkisimme, että meidän olisi kannattanut panostaa applikaation näyttävyyteen ja panostaa hieman vähemmän code behindin ominaisuuksiin.

# Ongelmat

Ensimmäiset ja oikeastaan isoimmat ongelmat olivat heti projektin alussa, kun havaitsimme ettei androidissa ole sisäänrakennettuja toimintoja ulkoisille tietokantayhteyksille. Tämä tuli tarkoittamaan sitä, että meidän olisi käytettävä jo olemassa olevaa valmista tietokantarajapintaa tai luotava oma. Tutkimme hieman mahdollisuuksiamme ja päätimme tehdän oman toteutuksen, sillä kävi ilmi, että ASP.NETillä voidaan luoda applikaatiotamme palveleva restful api toteutus azuren pilveen.

Itse restful apin toteutus tuotti myös ongelmia heti sen alkuvaiheessa. Ihan ensimmäiseksi pystytimme LAMP-palvelimen ulkoverkkoon digital oceanin pilvipalveluun. Linuxpohjaisena, kävi ilmi että tarvitsemme ASPin ajoa varten Apache2 palvelimelle mod\_mono nimisen lisäosan, joka on osa Mono-projektia. Mono-projekti mahdollistaa ASP.NET toiminnallisuuksien ajamisen linuxympäristössä. Vaikka modin asennus onnistui, applikaation ajaminen sille oli erittäin huonosti dokumentoita, emmekä lopulta saaneet sitä toimimaan. Jouduimme tutustumaan vaihtoehtoihin ja lopulta päädyimme Azuren pilveen, jolla oli parhaat integraatiot suoraan käytössä olevaan IDEen.

Tämän lisäksi itse koodausvaiheessa rest apissa ongelmat muodostuivat reititysongelmiksi. Reitittäminen tässä yhteydessä tarkoittaa URLin polun pohjalta apin ohjaaminen oikeisiin metodeihin. Syntaksi oli hyvin erilaista kuin tähän asti oppimamme, joka dokumentaatio hieman korkealentoista. Tästä huolimatta onnistuimme toteuttamaan täyden toiminnallisuuden.

Androidin puolella ongelmia oli alussa asynkronisten toimintojen debuggaamisen ja synkronoinnin kanssa. AsyncTask oli hyvin epäluotettava erilaisten tulostuskonstien knassa, eikä sitä pystynyt suoraan debuggaamaan. Se skippaa täysin ongelmitta kaikki break pointit eikä suostu näyttämään mitään tietoa siinä threadissa pyörivistä toiminnoista.

AsyncTaskeja seuratessa oli myös ongelmallista seurata threadin suorituksen valmistumista. Kun yksi threadi odottaa kahta muuta threadia ennen kuin se voi jatkaa, eikä ole hyvä käytäntö lukita UI-threadia, tuli koodin liitännöistä interfacejen kautta melko rönsyileviä. Koodia katselemalla ei ole heti selvää mitkä metodit esim MainActivityssä ajetaan asynkronisesti main threadin rinnalla.

Ei niinkään pysäyttäväksi vaan paremminkin hidastavaksi ongelmaksi muodostui gradle-buildin ajo joka ikisellä testiajokerralla. Gradle build tyypillisesti kestää minuutin pari, joten sen pyöräyttäminen jokaisen ajon yhteydessä hidastaa huomattavasti kehitystä. Ongelmaa yritettiin korjata useasti, mutta sille ei löytynyt suoranaista ratkaisua. Ongelma tosin hävisi itsestään projektin loppuvaiheilla.

# Suunnittelu, työnjako ja aikataulutus

Työnjakoa ei suunniteltu missään projektin vaiheessa. Töitä jaettiin as-needed periaatteella. Olisi ollut mahdotonta suunnitella etukäteen ohjelmistorakenne jonka rakentamisen olisi voinut tällätavoin synkronoida. Pääsyyllisinä tähän on nollakokemus androidista ja ASP.NETistä, puhumattakaan web apista. Jotta voi suunnitella hyvin, on tiedättävä ohjelmistojen ja rakenteiden mahdollisuudet ja rajoitukset. Nämä puolestaan selviävät vasta toteutuksen aikana. Näillä perusteluilla skippasimme arkkitehtuurisuunnittelun ja sen pohjalta tehtävän työnjaon.

Suunnitteluvaiheessa teimme kuitenkin määritykset mitä tarvitsemme ulkoiselta tietokannalta, millaiset näkymät haluaisimme ns. appstore-versioon ja millaisia featureja haluamme implementoida applikaatioon. Tuloksena siis käyttötapauskaaviot, UI-mockupit ja tietokantakaavio.

Alkupäerinen aikataulutus suunniteltiin kokonaisuus, feature ja dokumenttitasolla samasta syystä koska arkkitehtuuri ei ollut tarpeeksi tuttua, että sen olisi voinut tehdä luokka ja metoditasolla. Alla on suunnitelmasta mukautettu toeutunut aikataulu.

|  |  |  |
| --- | --- | --- |
| Viikko / Tekijä | Atro | Aleksi |
| 36 | Mockupit | Tietokantasuunnitelma |
| 37 | Mockupit | Tietokantasuunnitelma |
| 38 | Käyttötapaukset | User storyt |
| 39 | LAMPin pystytys | LAMPIn pystytys |
| 40 | mod\_mono pystytys / tietokannan luonti | restful api |
| 41 | test restin julkaisu / androidin paikannusdatan aloitus | test restin julkaisu / androidin httprequest |
| 42 | Lepiä | Lepiä |
| 43 | gps pohjainen etäisyyden tarkistus mock datalla | vaihto azureen LAMPista / AsyncTaskien luonti |
| 44 | Ensimmäinen toimiva rest | Ensimmäinen toimiva rest |
| 45 | Paikallisen tietokannan käyttöönotto | JSONin parsiminen ja kohteiden oliointi |
| 46 | Paikallisen tietokannan käyttöönotto | JSONin parsiminen ja käyttäjien oliointi |
| 47 | Restin refaktorointia ja optimointia, getAll metodien korjaukset kohteille ja käyttäjille | Applikaation refaktorointia ja optimointia |
| 48 | Loppudokumentaatio, SQLite korjailua, Insertti CRUD restiin käyttäjän kohteen vaihtamista varten | Loppudokumentaatio, gesture pohjainen navigointi, fragmentit |
| 49 | Käyttäjän lisääminen kohteeseen POST-metodilla, PUT-metodi pois käytöstä, loppudokumentaation viimeistely | RecyclerView ja korttinäkymät demoa varten, loppudokumentaation viimeistely |

# User storyt

Silja, 28, valmistunut tradenomiopiskelija, joka asuu Jyväskylän keskustassa. Hän käynnistää Dionys appin nähdäkseen, onko Hemingwayssa hänen tuttujaan, sillä hän haluaisi nauttia muutaman oluen kotinsa ulkopuolella.

Erkki 25, insinööriopiskelija Jamkissa rekisteröityy palveluun, jotta pääsee kirjautumaan sisään. Näin hän pääsee selaamaan lähialueen venueita ja niiden tietoja, sekä niissä olevia käyttäjiä. Erkki toivoo löytävänsä venuen käyttäjälistasta kivan oloisen tytön, jonka kanssa hän voisi keskustella ohjelman tarjoaman IM-palvelun avulla.

Vesa 33 on pitkän kaavan automaatiotekniikan opiskelija ja hän tykkää hieman liikaa alkoholista. Vesa on yksinäinen sielu ja epävarma sielu ja haluaisi pariutua jonkun kivan tytön kanssa. Vesalla on kuitenkin paha tapa rohkaista itseään hieman liikaa viinalla ennen lähestymisyritystä. Tästä johtuen Vesa antaa itsestään huonon örveltäjän kuvan kaikille lähestymilleen tyttösille. Vesa toivoo että olisi keino lähestyä digitaalisesti paikan päällä olevia tyttösiä ilman, että tietää heidän numeroaan. Vesa löytää Dionys appin Googlen Play-Storesta, lataa sen ja rekisteröityy appin aloitussivulta. Vesa pystyy välittömästi valitsemaan haluamansa baarin listalta ja nähdä ketä käyttäjiä siellä on paikalla. Vesa pystyy painamaan henkilöä ja lähettämään tälle pikaviestejä.

Marjut 22 on ensimmäisen vuoden hoitoalan opiskelija Jamkissa. Marjut ei vielä tunne kaupunkia kovin hyvin, eikä tiedä kaupungissa olevia menomestoja ja ajanvietto kohteita. Marjut suunnittelee illanistujaisia uusien kavereidensa kanssa, mutta Marjutille on kysymysmerkki mikä voisi olla sopiva paikka. Marjut löytää Dionys appiin kirjautumalla kaikki rekisteröidyt kohteet 10 kilometrin säteellä listattuna ja merkittynä karttaan. Tästä selaamalla Marjut klikkailee kohteita kunnes löytää mieleisensä kohteen kuvauksen, kuvien ja sijainnin perusteella.

# Käyttötapaukset

C:\Users\Aleksi\AppData\Local\Microsoft\Windows\INetCacheContent.Word\AndroidUseCase.png

Kuva 1: Käyttötapaukset end userin näkökulmasta

Käyttötapaukset ovat vain aplikaation käyttäjän perspektiivistä. Rajasimme pois mahdollisen kohteen omistajan interaktiot, koska ne eivät liity suoraan appiin vaan erilliseen html sivuun internetissä, jonka data kulkeutuu appiin rest apin kautta.

# Käyttöpolku

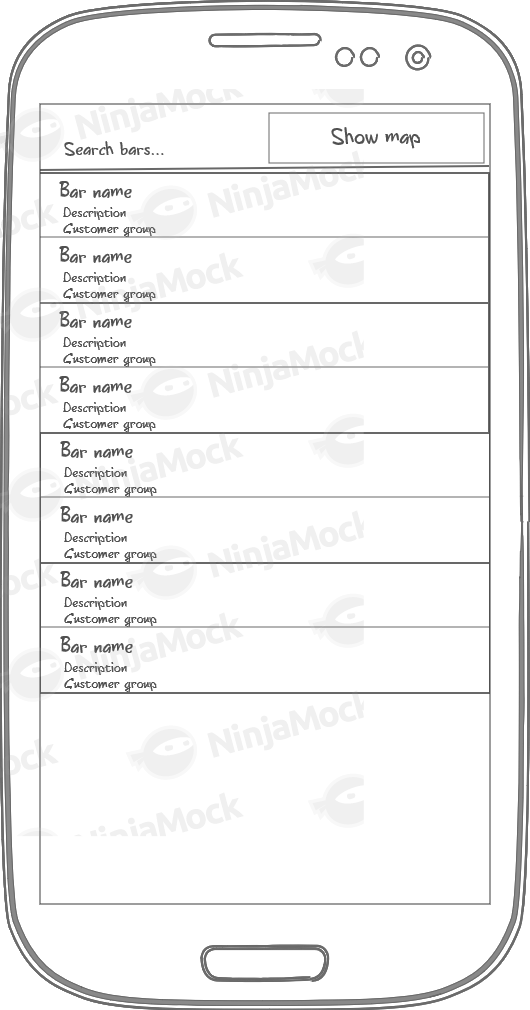
UI:n ollessa hyvin keskeneräinen varsinainen käyttöpolku ei kuvasta kovinkaan hyvin suunniteltua. Tällä hetkellä aloitusnäkymä on loginfragmentti, johon viimeistellään login ja register näkymät. Vasemmalle swipeämällä tulee debug näkymä jossa tulostetaan debug dataa ohjelman suorituksesta. Oikealle swipeämällä näkyviin tulee alla olevan diagrammin mukainen Avenue list näkymä, eli kaupungissa olevat kohteet card view muodossa listana. Tästä listasta avenueita voi klikkailla yksitellen, jolloin kohteen takaa aukeaa uusi fragmentti, joka näyttää kohteessa olevat käyttäjät.

C:\Users\Aleksi\AppData\Local\Microsoft\Windows\INetCacheContent.Word\App Flow.png

Kuva 2: Suunniteltu ohjelman näkymien kulku

# Käyttöliittymämockit



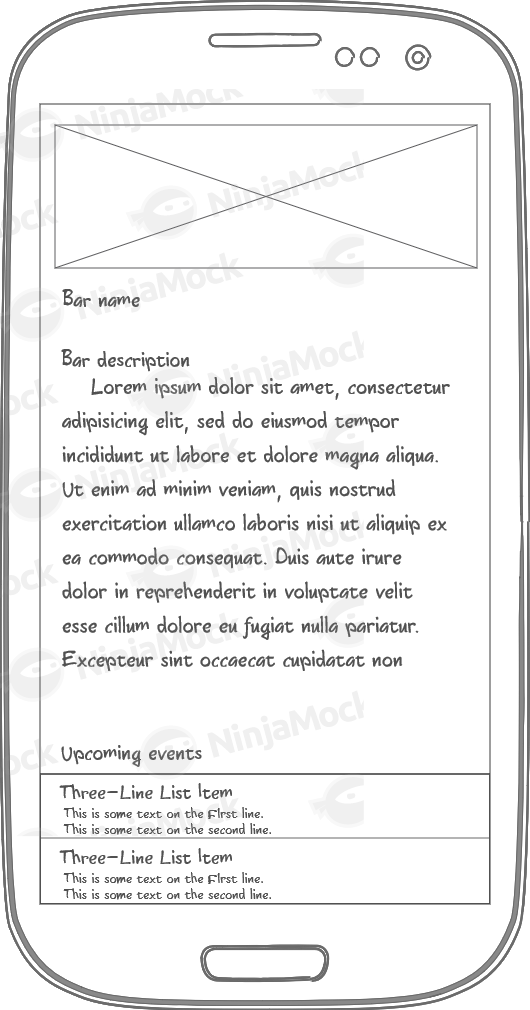
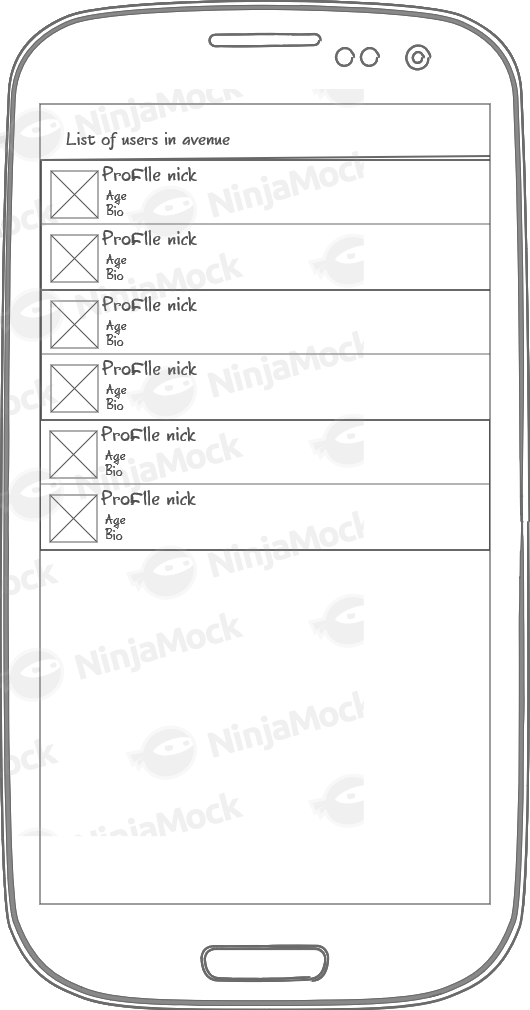


Kuva 3: Login ruutu, appin aloituspaikka

Kuva 4:Kohteet listana card viewissä

Kuva 3 on perinteinen loginruutu, nappulat sisäänkirjautumiselle ja rekisteröinnille, mietittiin mahdollisuutta logata sisään facebook tai google+ tunnuksilla.

Kuvassa 4 perusnäkymä kaikista kohteista listattuna tietyllä etäisyydellä. Suunniteltiin noin 10 kilometrin etäisyyttä.

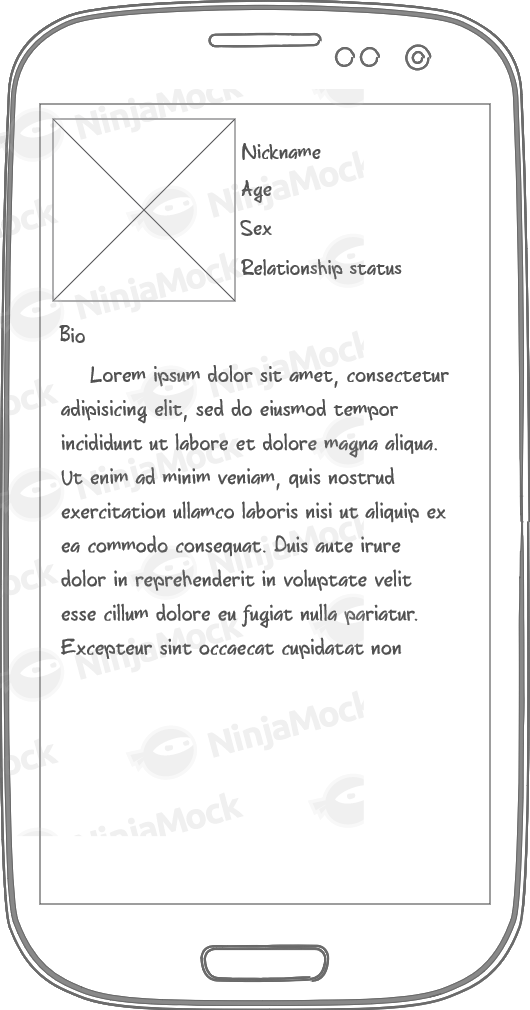


Kuva 5: Käyttäjät venuessa

Kuva 6: Venuen profiilisivu

Kuva kuusi on kohteen oma profiilisivu. Suunniteltiin sen sisältävän kuvauksen ja mahdollisuuden lisätä tapahtumia. Kaavailtiin jatkokehitysmahdollisuutta jossa kohteen omistajat voisivat käyttää nettisivua kuvauksien lisäämiseen, jotka sitten parsittaisiin rest apilla ja toimitettaisiin JSONina applikaatiolle puhelimeen.

Kuva seitsemän näyttää käyttäjät listattuna tietyssä kohteessa. Henkilöt klikattavia ja avaavat käyttäjän profiilinäkymän, joka on kuvattu kuvassa 7.



Kuva 7: pikaviestitys käyttäjien välillä

Kuva 8: Käyttäjän profiilinäkymä

Kuva 7 kuvaa käyttäjän profiilinäkymää. Kuva kahdeksan on XMPP pohjainen pikaviestitysnäkymä jolla käyttäjät voivat viestitellä toisilleen.

# OOP

Olioinnin puolella Dionys appissa on vain kaksi keskeistä olioitavaa luokkaa Venues ja Users. Molemmille luokille on omat fragmentit, joihin toimitetaan niihin kuuluvat oliolistat bundleilla. Molemmat luokat myös implementoivat Parcelablen.

Molemmilla luokilla on niille oleelliset sisäiset propertyt, että niille getterit ja setterit. Näiden lisäksi niillä on oletuskonstruktorit, overloadattavat konstruktorit ja overloadatut parcelable luokan metodit describeContents, writeToParce ja Creator.

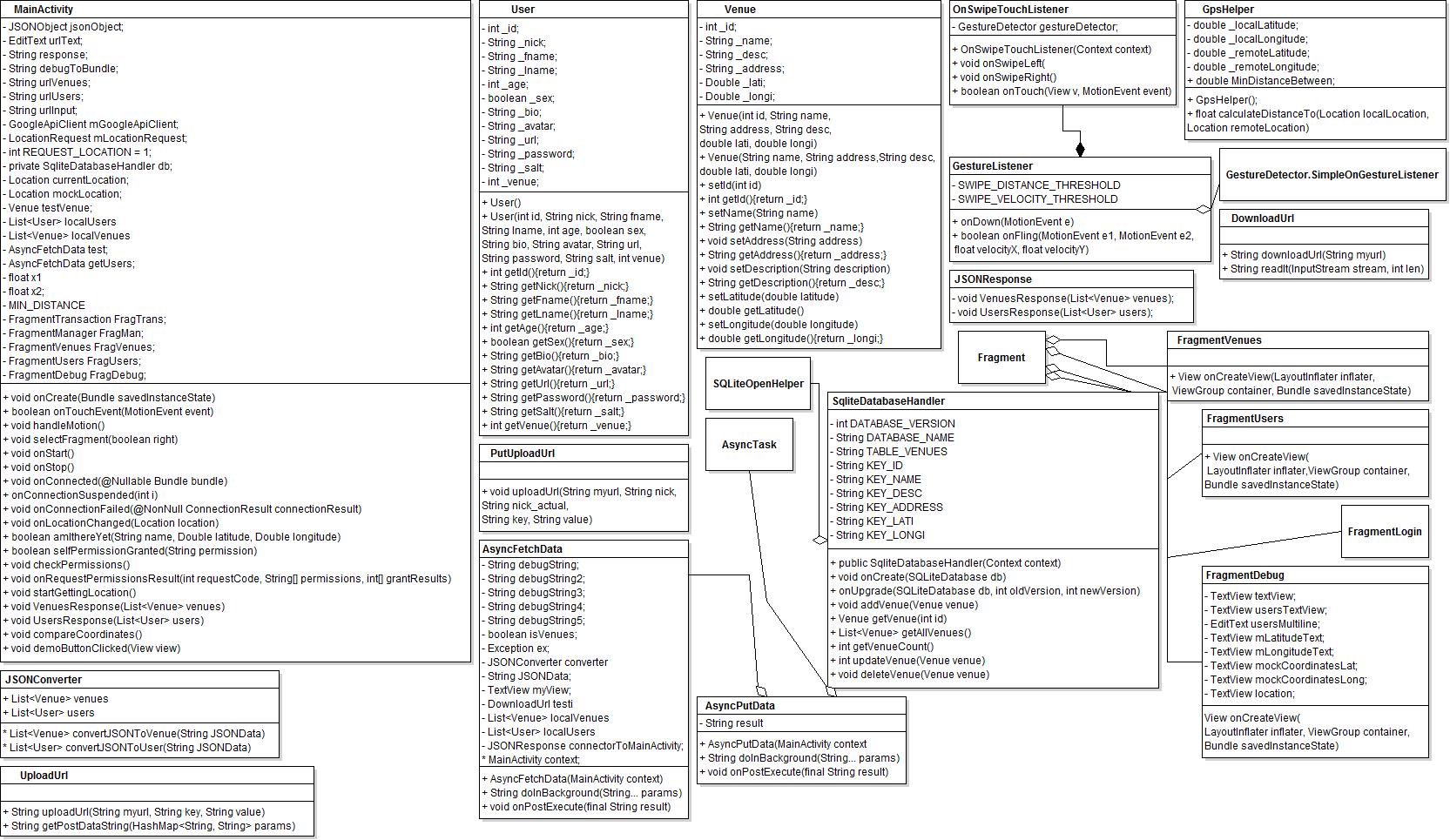
# UML Luokkarakenne

Luokkarakenne on toteutettu niin, että kaikki mikä ei ole raskasta tai hidasta pyörittää, toimii MainActivityssä jos se hiukankin voi. Dokumentaation ja keskustelun pohjalta päätettiin, että ei ole hedelmällistä käyttää useita activityjä ja pallotella dataa niiden välillä. Dokumentaatio sanoo, että siirtymät aktivityjen välillä ovat raskaita ja datansiirto niiden välillä hankalaa. Tästä syystä toteutettiin fragment pohjainen käyttöliittymä. Kaikki näkymät ovat siis rakennettu fragmenteillä MainActivityn päälle.

Datan haut ja datan pistot ulkopuoliseen MySQL tietokantaan tapahtuu asynkronisesti AsyncFetchData ja AsyncPutData luokkien avulla. Datan syöttö asynkronisista luokista MainActivityyn on toteutettu interfaceilla JSONResponse ja UsersInterface.

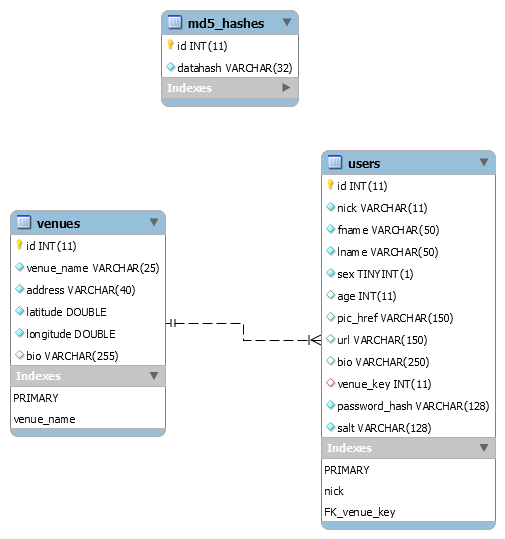
Olioluokkia on kaksi, Venue ja User. Toinen on kohde, baari, puisto tai muu vastaava. User on kohteessa oleva käyttäjä. Näiden lisäksi Fragmenteille on omat luokkansa. Paikalliselle SQLite tietokannalle ja paikannusdatalle on omat avustajaluokkansa.

Kosketuseleille on omat luokkansa OnSwipeTouchListener ja GestureDetector.SimpleGesture.Listener



Kuva 9: Luokkarakenne

# Tietokantaratkaisu



Kuva 10: Ulkopuolisen tietokannan tietokantaratkaisu

Ohjelmiston kanta pyörii MySql-relaatiokantamoottorilla Azuren pilvipalvelussa. Päädyimme käyttämään MySql kantaa, koska alunperin RESTin oli tarkoitus pyöriä Linuxilla Apache 2 palvelimella .NET Core moduulilla. Tämä kumminkin osoittautui työlääksi vaihtoehdoksi, joten päädyimme Azuren pilvipalveluun. Onneksemme Azuren opiskelijalisenssiin sisältyi yksi ASP sovelma ja jaettu MySql kanta sille.

Tietokantaan ei käytännössä tarvittu muutakuin kohteet venues ja käyttäjät users. Käyttäjä voi olla vain yhdessä kohteessa, mutta yhdessä kotheessa voi olla useita käyttäjiä samanaikaisesti. Kohteella on nimi, osoite, koordinaatit ja kuvaus.

Henkilöllä on nikki, etunimi, sukunimi, sukupuoli, ikä, kuvan url, valinnainen url, kuvaus, kohde jossa hän on, salasana hashattuna, ja hashin suola. Hash taulu implementoitiin mutta jätettiin odottamaan jatkokehitystä.

# Loppusanoma

Presentaatio meni omasta mielestä hyvin ja sujuvasti, mutta UI-puolen esittelyn osalta laihanlaisesti. Priorisointi ongelma, koska halusimme tehdä mahdollisimman monipuolisen ja joustavan toteutuksen itse toiminnalliselle logiikalle, ennen kuin lähdemme suunnittelemaan ja toteuttamaan miten käsitelty data olisi nätisti esitettävissä käyttäjälle. UI-puolen jutut on suunniteltu vasta kehityskaaren loppupuolelle jolloin niihin on oleellista laitettavaa, mutta sikäli jos UI on painoarvoltaan yhtä oleellinen kuin toiminnallinen logiikka, ammuimme itseämme jalkaan tämän projektin skaalan kanssa. Tällöin olisi pitänyt joko toteuttaa kovakoodattua dataa päälleliimattuihin UI elementteihin ja unohtaa dynaamisuus, tai skaalata vähemmän kunnianhimoinen projekti jonka kehityskaaressa olisi järkevämmin voinut tehdä dynaamista dataa näyttävän UI:n opintojakson kuluessa.

Kuitenkin projekti toteutettiin suunnitellussa aikataulussa ja kaikki ne toiminnallisuudet joita alussa rajattiin, että on demoon mennessä tehtynä, saatiin tehtyä, sekä niiden päälle vielä ekstraominaisuuksia kuten gesturejen hallinta ja fragmenttipohjaiset näkymät.