Haaga-Helia
ammattikorkeakoulu Oy

Sovelluksen toteutus Androidille

Tommi Leskinen

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2017

Tiivistelmä

Haaga-Helia
ammattikorkeakoulu Oy

Tekijä(t)
Tommi Leskinen

Koulutusohjelma
Tietojenkäsittely

Opinnäytetyön otsikko	**Sivu- ja liitesivumäärä**
Sovelluksen toteutus Androidille	22

Opinnäytetyön aiheena on Android sovellus. Opinnäytetyössä tavoitteena on toteuttaa elokuva-aiheinen katselulista sovellus. Sovelluksella on mahdollista luoda katselulistoja elokuvista ja lisäksi myös sarjoista. Elokuvien tietoja voidaan hakea ulkopuolisesta tietokannasta ja lisätä sovelluksen käyttäjän omiin katselulistoihin. Itse elokuvaa ei sovelluksella voi ladata tai katsoa. Sovelluksessa käsitellään vain elokuvien tekstipohjaista tietoa.

Työ on rajattu siihen, miten se on mahdollista toteuttaa Android alustalla, eikä muilla alustoilla, kuten esimerkiksi iOS. Tavoitteena on saada sovelluksesta melko yksinkertainen kokonaisuus valmiiksi. Tarkoituksena on, että sovelluksen pääominaisuudet toimivat, vaikka esimerkiksi käyttöliittymä jäisi vielä keskeneräiseksi.

Työn alussa käydään läpi opinnäytetyön tietoperusta, kuten ohjelmointiympäristöt Androidilla ja millaisia eri teknologioita Androidille löytyy sovelluksen toteuttamiseksi. Tietoperustan jälkeen käydään läpi itse sovelluksen toteutus ja kuvataan sen toimintaa. Lisäksi tässä kuvataan toteutuksessa ilmenneet ongelmat ja muut haasteet. Työn lopussa on pohdinta, jossa kerrotaan sovelluksen jatkokehityksestä, miten työ onnistui ja mitä työstä tuli opittua.

Asiasanat
Sovellus, Android, elokuva, rajapinta, mobiili

Sisällys

1 Johdanto

Opinnäytetyön aiheena on sovelluksen toteutus Androidille. Erilaisia mahdollisia sovelluksia ja aiheita on useita, mutta tähän työhön valitsin elokuva-aiheisen sovelluksen tekemisen. Työ sisältää miten sovellus voidaan toteuttaa Androidille ja millä tekniikoilla se voisi onnistua parhaiten.

Työssä tehtävän sovelluksen päätarkoituksena on mahdollistaa erilaisten katselulistojen luominen ja elokuvien tietojen haku. Käyttäjä voisi luoda sovelluksessa oman elokuvalistan tai useamman ja nimetä ne esimerkiksi katsottavien elokuvien teeman mukaan. Elokuvalistoihin voisi lisätä elokuvia hakemalla elokuva ulkoisesta elokuvatietokannasta sovelluksen avulla. Sovelluksen tarkoitus ei ole kuitenkaan mahdollistaa itse elokuvan latausta tai katsomista, vaan tarkoituksena on pitää kirjaa katsottavista elokuvista.

Sovellus on tarkoitus toteuttaa mahdollisimman yksinkertaisesti ja siten, että se olisi käyttäjän kannalta helppokäyttöinen. Lisäksi sovellus on hyvä pitää yksinkertaisena, jottei sen kehitysvaiheesta tulisi liian pitkä. Sovellus tullaan julkaisemaan Googlen Play -kaupassa kun se on julkaisuvalmis. Ennen virallista julkaisua, sovellus mahdollisesti julkaistaan rajatussa alfa tai beeta -muodossa Play -kaupassa, sekä rajatulle yleisölle.

Opinnäytetyössä tutkitaan ja vertaillaan ensin mahdollisia työvälineitä ja teknologioita työn toteuttamiseksi. Näitä vaihtoehtoja on monia ja siksi työn alussa on mielestäni hyvä vertailla niitä. Tämän jälkeen esitellään itse sovelluksen toteutus, rakenne, ongelmat ja haasteet. Lopuksi opinnäytetyössä pohditaan mahdollisia jatkokehitys mahdollisuuksia, työn toteutusta sekä tuloksia.

1.1 Lyhenteet ja sanasto

ADT	Android Development Tools. Kehitystyökalu Eclipse -nimiselle kehitysympäristölle.
Ajax	Asynchronous JavaScript and XML. Yhdistelmä eri web-sovelluskehitystekniikoista.
API	Application Programming Interface. Tarkoittaa ohjelmointirajapintaa, jonka kautta eri ohjelmat voivat esimerkiksi vaihtaa tietoa.
DOM	Document Object Model. DOM on esimerkiksi XML dokumentin ohjelmointirajapinta.
GPS	Global Positioning System. Paikantamisjärjestelmä.
JSON	JavaScript Object Notation. JSON on tiedostomuoto tiedonvälitykseen.
OMDB	The Open Movie Database.
SDK	Software Development Kit. Kokoelma ohjelmistokehitystyökaluja.
SQL	Structured Query Language. Standardoitu tietokanta kyselykieli.
TMDb	The Movie Database
XML	Extensible Markup Language. XML on merkintäkieli, jonka avulla voidaan säilyä tai siirtää tietoa.
XSLT	Extensible Stylesheet Language Transformations. XSLT on merkintäkieli XML-tiedostojen muunnoksiin.

2 Tietoperusta

2.1 Miksi kehittää Androidille?

Mobiilisovellus on mahdollista toteuttaa jollekin tietylle alustalle, esimerkiksi Androidille, iOS:lle tai selainpohjaisesti. Selainpohjaisena sovellus voisi teoriassa toimia millä alustalla tahansa.

Android käyttää ohjelmointikielenä Javaa, joka on yleisesti käytetty ohjelmointikieli (Lifewire, 2011). IOS taas käyttää Objective-C kieltä, joka muistuttaa C ja C++ kieliä (Lifewire, 2011). IOS:lle on myös uudempi ohjelmointikieli, Swift (Apple, 2017). Lifewire sivuston mukaan monialustaiset ratkaisut ovat suosittuja, mutta ne eivät välttämättä ole optimaalisia alkuperäisen tiedon näyttämiseen (Lifewire, 2011). Tämänlaisia ovat esimerkiksi käyttöjärjestelmän painikkeet ja muu käyttöliittymän osat, joilla tietoa voidaan näyttää.

Android alustalle kehittämisessä on se hyöty, että siinä on todella laaja käyttäjäkunta. Gartner sivuston julkaisemassa raportissa on kuvattu eri mobiilikäyttöjärjestelmien markkinaosuudet. Raportin mukaan Android ja iOS ovat selvästi suosituimpia kaikista mobiilikäyttöjärjestelmistä. Androidilla on noin 81,7% markkinaosuus ja iOS:llä noin 17,9% markkinaosuus. Muiden mobiilikäyttöjärjestelmien markkinaosuus on alle prosentti. (Gartner, 2017).

Android on myös avoin kehitysalusta, toisin kuin Applen iOS. Android alustalla on saatavilla enemmän kolmannen osapuolen kehitystyökaluja, joka helpottaa erilaisten ominaisuuksien lisäämisen sovellukseen. Apple taas pitää tarkemmin kiinni omista kehittäjien ohjeistuksestaan iOS alustalla. (Lifewire, 2011).

Androidin suosiosta huolimatta usein kuulee, että sovellusten tekeminen iOS:lle on helppoa. Tätä näkemystä tukee erään henkilön vastaus Quora sivustolla esitettyyn kysymykseen, että miksi useimmat kehittäjät sanovat, että iOS:lle sovellusten kehittäminen on helpompaa kuin Androidille. Sivuston käyttäjän Trausti Thor Johannssonin mukaan iOS:lle kehitykseen pääsee mukaan niin helposti kuin omistamalla Mac-koneen ja lataamalla Xcode sovelluksen (Trausti Thor Johannsson, 2016). Lisäksi hän mainitsee, että 90-95 prosenttia käyttäjistä siirtyy käyttämään aina uusinta versiota Mac käyttöjärjestelmästä muutamassa viikossa (Trausti Thor Johannsson, 2016). Androidilla tämä tilanne on hieman eri. Suurin osa käyttäjistä ei käytä uusinta versiota käyttöjärjestelmästä, vaan suosituin on jo useamman vuoden vanha (Google, 2017, 10). Johannsson myös mainitsee

myöhemmin vastauksessaan, että Android kehitys on kuitenkin myös helppoa kokeneille Java-kehittäjille (Trausti Thor Johannsson, 2016).

2.2 Ohjelmointiympäristöt Androidille

Ohjelmointiympäristö eli integrated development environment (IDE) on sovelluskokonaisuus, joka on tarkoitettu sovelluskehittäjille sovelluksien kehittämistä sekä testaamista varten. Se voi olla joko yksittäinen sovellus tai lisäosa muuhun sovellukseen. Ohjelmointiympäristöt sisältävät yleensä editorin, kääntäjän, virheenkorjausohjelman sekä tulkin. (Margaret Rouse, 2016). Sovelluksen ohjelmointiympäristöksi Android alustalla on muutamia vaihtoehtoja. Joitakin suosittuja vaihtoehtoja ovat Android Studio, IntelliJ IDEA sekä Android Developer Tools (Slant, 2017).

Android Studio on virallinen ohjelmointiympäristö Android alustalle. Se perustuu IntelliJ IDEA ohjelmointiympäristöön. Android studio tarjoaa muutamia varteenotettavia ominaisuuksia kuten InstantRun, monipuolinen emulaattori sekä älykäs editori. (Google, 2017, 1).

InstantRun ominaisuuden avulla muutokset voidaan suoraan siirtää käynnissä olevaan sovellukseen. Tämä tarkoittaa, että sovellusta ei tarvitse rakentaa alusta asti kokonaan aina kun haluaa nähdä muutokset. Sovellusta ei välttämättä tarvitse edes käynnistää uudelleen nähdäkseen koodin tuomat muutokset. (Google, 2017, 1).

Emulaattorin avulla kehitettävä sovellus voidaan asentaa virtuaaliselle laitteelle, kuten puhelimelle, tabletille tai vaikka televisiolle. Emulaattorilla voidaan myös simuloida erilaisia laitteen ominaisuuksia kuten sensorit sekä GPS. Android Studion editori auttaa koodin kirjoittamisessa ehdottamalla sopivia koodipätkiä samalla kun kirjoitat koodia (Google, 2017, 1). Android Studio on saatavilla Windowsille, Macille sekä Linuxille (Google, 2017, 2).

Eräs toinen vaihtoehto Android kehitykseen on ollut Android Developer Tools(ADT). Android Developer Tools on lisäosa Eclipse -kehitysympäristöön. 26.6.2015 lähtien ADT ei ole enää virallisesti tuettu eikä sen kehitys jatku. (Google, 2017, 3).

IntelliJ IDEA on hyvin samankaltainen ulkonäyllisesti kuin Android Studio. Se sisältää lisäksi lähes samat ominaisuudet. Siinä on monia sisäänrakennettuja työkaluja ja se tukee useita suosittuja sovellusrunkoja kuten Spring, AngularJS, Node.js, SQL, Java sekä

Android. IntelliJ on ilmainen Java ja Android kehitykseen, mutta ohjelman täysi versio on maksullinen. (JetBrains, 2017).

2.3 Tietokannat ja tallennus

Sovelluksessa on tarkoitus tallentaa paikallisesti laitteelle tietoa elokuvista, jotta niitä voidaan lukea myöhemmin uudelleen. Tähän tarkoitukseen on olemassa monenlaisia ilmaisia tietokantoja sekä muita tapoja tallentaa tietoa laitteen omaan muistiin.

SQLite on tietokantakirjasto joka ei tarvitse erillistä palvelinprosessia ja on itsenäinen. Lisäksi se ei tarvitse mitään asetuksien määrittelyä vaan toimii suoraan asennuksen jälkeen. SQLite käyttää transaktio -pohjaista SQL tietokantamoottoria. SQLiten koodi on julkista ja sitä voidaan käyttää minkälaiseen tarkoitukseen tahansa. (Hwaci, 2017).

SQLite eroaa muista yleisimmistä tietokannoista siten, että sillä ei ole erillistä palvelinprosessia. SQLite toimii siten, että se lukee ja kirjoittaa suoraan tavallisille levytiedostoille. Tietokanta sisältyy yhteen tiedostoon tauluineen ja sen tietomuoto on monialustainen. Tämä tarkoittaa, että tietokantatiedostoa voidaan käyttää täysin eri alustoilla, kuten Android laitteella tai tietokoneella. SQLite on pieni kirjasto ja sen koko saattaa olla vain 500 kilotavua. Tämä tekee siitä erittäin suositun mobiililaitteille, joissa muistin määrä on yleensä hyvin pieni. (Hwaci, 2017).

BerkeleyDB on yleiskäyttöinen tietokantamoottori, joka on suunniteltu suorituskykyä vaativiin sovelluksiin. Se voi hallita pieniä tai todella suuria datamääriä, mikä tekee siitä erittäin skaalautuvan. SQLiten tavoin, BerkeleyDB toimii sovelluksen prosessissa, eikä tarvitse erillistä palvelinprosessia. Tietokantaa käytetään ohjelmointirajapintaa käyttäen. Rajapinta mahdollistaa tietojen lukemisen, kirjoittamisen sekä muun tietokannan hallinnan. Java API rajapintoja on kaksi. Ensimmäinen niistä on korkeamman tason rajapinta ja se tukee pysyviä Java -luokkia, jotka voidaan tallentaa tietokantaan. Toinen rajapinta on alemman tason rajapinta ja se antaa enemmän joustavuutta tietokannanhallinnassa. (Oracle, 2017).

UnQLite on myös edellä mainittujen tietokantasovellusten tavoin sovelluksen prosessissa toimiva tietokantakirjasto. Siinä on lähes samat ominaisuudet kuin SQLitessä, mutta UnQLite käyttää NoSQL tietokantamoottoria. UnQLite tukee lisäksi dokumenttipohjaista tallennusta sekä perinteistä avain/arvo -tallennusta. Sillä voidaan esimerkiksi tallentaa JSON -pohjaista tietoa suoraan tietokantaan. (Symisc Systems, 2017).

Android Studiolla on mahdollista tallentaa tietoa laitteen sisäiseen tai ulkoiseen muistiin. Tietoa voidaan lukea muistista. Tieto voidaan tallentaa esimerkiksi avain/arvo muodossa tietokantojen tavoin. (Google, 2017, 4).

2.4 Ulkopuoliset rajapinnat

Koska sovellus tarvitsee jostain tietolähteen elokuvista, niin olisi hyvä, että tietolähde olisi mahdollisimman kattava ja sopiva sovelluksen käyttötarkoitusta varten. Tietokannan täyttäminen itse sadoilla tuhansilla eri elokuvilla veisi ikuisuuden. Tämän takia lähes helpoin vaihtoehto on käyttää jotain valmista elokuvatietokantaa. Ja lisäksi on sovellus järkevää toteuttaa niin, että sen käyttämä tieto tulee pääosin ulkopuolisesta lähteestä, eli esimerkiksi yleisesti käytettyjen ohjelmointirajapintojen eli API:n kautta. Tarkoituksena on löytää sopiva avoin elokuvatietokantarajapinta, joka tarjoaa riittävät ominaisuudet sovelluksen tarkoitukseen.

OMDB on elokuva-aiheinen tietokantasivusto, josta käyttäjä voi hakea elokuvia sekä sarjoja sekä niiden sisältöön liittyvää tietoa. Rajapinta on ilmainen ja sitä voidaan käyttää eikaupalliseen tarkoitukseen. Rajapinnan käyttöä varten kehittäjän ei tarvitse hankkia avainta, joita usein rajapinnoissa käytetään käyttöoikeuksien hallintaan. Rajapintaan lähetettävien Http-pyyntöjen määrä ei ole myöskään rajoitettu. (Brian Fritz, 2017).

TMDb (The Movie DB) on toinen elokuvatietokanta sivusto, joka tarjoaa avoimen rajapinnan. Sivustolta voi OMDB:n tavoin etsiä elokuvia ja tietoa elokuvista. Rajapinta on pääosin avoin kaikille, mutta kehittäjien tulee hankkia avain rajapinnan käyttämistä varten. Rajapinta palauttaa dataa JSON muodossa. TMDb:n API:n ominaisuuksiin kuuluu elokuvien haun lisäksi rekisteröityneiden käyttäjien ominaisuudet, joita varten käyttäjän tulee olla rekisteröitynyt TMDb:n palveluun. Näitä ovat esimerkiksi kirjautuminen, elokuvien arvostelu sekä katselulistojen luominen. (The Movie DB, 2017).

2.5 Tiedon hakeminen ulkopuolisesta rajapinnasta

Sovelluksessa on tarkoitus hakea tietoa ulkopuolisesta lähteestä ja haku tapahtuu internetin välityksellä. Tätä varten sovelluksessa tulisi olla käytössä teknologia, jonka avulla dataa voitaisiin noutaa netistä taustalla ilman, että käyttäjän sovelluksen käyttö keskeytyy. Tätä varten on olemassa monia erilaisia tekniikoita ja teknologioita, joiden avulla tiedon hakeminen onnistuu esimerkiksi ulkopuolisen ohjelmointirajapinnan kautta.

Eräs usein käytetty apuväline tähän tarkoitukseen on Ajax. Ajax on yhdistelmä teknologioita, jotka mahdollistavat dynaamisen DOM sisällön näyttämisen sekä käytön, tiedon välittämisen ja muuntamisen XML sekä XSLT syntakseja käyttäen, asynkronisien tiedon hakujen käyttämällä XMLHttpRequestia. Ajax on käytettävissä JavaScriptiä käyttäen. (Jesse James Garrett, 2005).

JQueryllä on myös mahdollista käyttää Ajax kutsuja. JQueryn Ajax toimii lähes samalla tavalla kuin JavaScript Ajax. Erona on enimmäkseen jQueryn oma syntaksi sekä joitakin lisätoimintoja. (The jQuery Foundation, 2017).

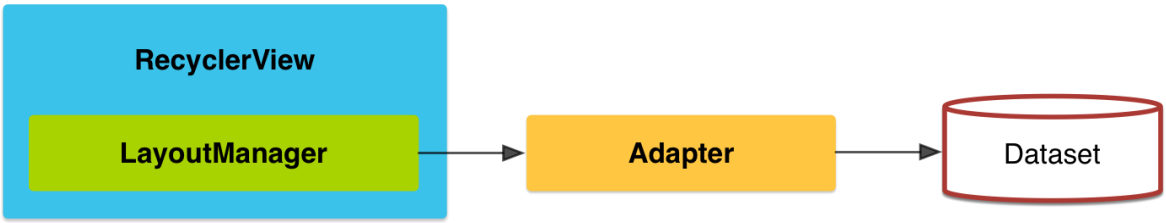
Android Studiossa on mahdollista käyttää Android Studion omaa AsyncTask rajapintaa. Se toimii hieman samalla periaatteella kuin Ajax. Se mahdollistaa toimintojen suorittamisen sovelluksen taustalla. Se on tarkoitettu pääasiassa lyhyiden toimintojen suorittamiseen. Vaativimpien toimintojen suorittamiseen on Executor, ThreadPoolExecutor sekä FutureTask rajapinnat. (Google, 2017, 5).

2.6 Tiedon näyttäminen käyttöliittymässä

Sovelluksessa tulisi näyttää tietoa elokuvista käyttäjälle mahdollisimman selkeästi, joten tätä varten tulisi valita siihen soveltuva tekniikka ja tapa näyttää tietoa. Android tarjoaa monenlaisia vaihtoehtoja erityyppisen tiedon näyttämiseen.

ListView soveltuu saman tyyppisten kuvien ja tekstien näyttämiseen. Se on optimoitu lukemista varten. Androidin dokumentaatiossa suositellaan, että ListViewiä tulisi käyttää maksimissaan 3 rivin kokoisen tietokokonaisuuden näyttämiseen. Mikäli halutaan näyttää suurempi määrä tietoa, tulisi käyttää CardViewiä. (Google, 2017, 8).

RecyclerView on käyttöliittymäelementti, joka tekee suuren tietomäärän näyttämisestä yksinkertaisempaa. Sitä suositellaan käytettäväksi suuriin tietomääriin tai jos tieto muuttuu sovelluksen ajon aikana. ReyclerViewin käyttöä varten tulee määrittää LayoutManager tiedon esittämistä varten sekä Adapter tiedon hallintaa varten (kuva 1). (Google, 2017, 6).



Kuva 1. RecyclerView, Google, 7.

RecyclerView:in LayoutManager osa voi olla kolmea erilaista tyyppiä. Se voi olla Linear-LayoutManager, GridLayoutManager tai StaggeredGrid-LayoutManager tyyppinen. LinearLayoutManager näyttää tuotteet vertikaalisena tai horisontaalisena listana. Grid-LayoutManager näyttää tuotteet ruudukkona. StaggeredLayoutManager näyttää tuotteet nimensä mukaan porrastettuna ruudukkona. (Google, 2017, 6).

RecyclerViewissä tarvittava Adapter muodostaa tietokokonaisuudet käytettävästä tiedosta. Se luo näkymät ”tuotteille” sekä muuttaa näkymien sisältöä tiedon muuttuessa. Esimerkiksi jos jokin listan tieto ei ole enää saatavilla, Adapter voi poistaa sen listasta. (Google, 2017, 6). RecyclerViewerin avulla tietoa voi näyttää esimerkiksi päivittyvän listan tapaan.

CardView näyttää tietoa korttien muodossa. CardViewille voi asettaa erilaisia ominaisuuksia, kuten varjostus ja reunojen pyöristäminen. CardView on hyödyllinen sellaisten elementtien näyttämiseen, joiden koko ja toiminnot voivat vaihdella. Tämänlaisia voisivat olla esimerkiksi erikokoiset kuvat. (Google, 2017, 6).

Androidin taulukot soveltuvat parhaiten suuren tietomäärän näyttämiseen. Niissä voi näyttää tietoa useilla riveillä sekä sarakkeilla. Taulukkojen tietoa on mahdollista muut-taa sekä järjestää uudelleen. Taulukoita on myös mahdollista asettaa CardViewin sisään. (Google, 2017, 9).

3 Elokuvalista sovelluksen toteutus

Sovelluksen kehityksen alkuvaiheena oli tarkoitus tutkia mahdollisia vaihtoehtoja kehitysmenetelmille, sekä eri työkaluille ja teknologioille. Pyrin löytämään omaa näkemystäni parhaiten vastaavan tavan toteuttaa sovellus. Tämän lisäksi halusin, että toteutus ei olisi liian monimutkainen tai hankala rajoitetun aikataulun takia. Mielestäni parempi oli aluksi luoda vakaa pohja ja rakentaa vain sovelluksen toimintaan vaadittavat osat sovelluksen käyttötarkoitusta varten. Myöhemmin tätä olisi helpompi jatkokehittää ja laajentaa sovelluksen toimintoja tarpeen mukaan. Esimerkiksi käyttöliittymä on alussa todella yksinkertainen ja siinä on mielestäni paljon jatkokehitettävää.

Valitsin sovelluksen toteutettavaksi Android käyttöjärjestelmälle ja Android Studion kehitysalustaksi. Valitsin Androidin, koska se on selvästi suosituin mobiilikäyttöjärjestelmä.

Opinnäytetyön suurimpana lähteenä käytin Googlen Android Studio dokumentointia. Dokumentointi on todella laaja ja sieltä löytyy paljon erilaisia esimerkkejä sekä ratkaisuja ongelmiin. Tekniikoiden ja teknologioiden sekä niiden ominaisuuksien vertailuun opinnäytetyössä käytin useita muita lähteitä, jotka myös koostuivat lähinnä niiden teknillisestä dokumentaatiosta.

Sovelluksen toteutus tehtiin Android Studiolla. Android Studion valitsin, koska se on todella helppokäyttöinen, hyvin tuettu, ja siinä on useita erilaisia tapoja ja valmiita kirjastoja toteuttaa jokin asia. Android Studio valikoitui työvälineeksi lisäksi sen takia, koska se on Androidin virallinen kehitystyökalu. IntelliJ olisi myös ollut hyvä vaihtoehto, mutta koska Android Studio perustuu IntelliJ:hin ja sisältää lähes samat ominaisuudet, mielestäni Android Studio oli parempi vaihtoehto. ADT (Android Development Tools) ei ollut mielestäni varteenotettava vaihtoehto, koska se ei ole enää virallisesti tuettu ja sen kehitys on lopetettu.

Valitsin sovelluksessa käytettäväksi ulkopuoliseksi elokuvatietokannaksi OMDb:n rajapinnan. Mielestäni OMDb:n rajapinta oli todella selkeä ja yksinkertaisesti toteutettu. Rajapinnan palauttama JSON muotoinen data on hyvin rakennettu ja sitä oli helppo työstää sovelluksen koodi puolella. TMDB olisi ollut muuten hyvä vaihtoehto, mutta rajapinnan avaimen saaminen kuulosti hieman hankalalta. TMDB:ssä on todennäköisesti enemmän elokuvia sekä ominaisuuksia kuin OMDb:ssä, mutta en kuitenkaan päätynyt sen kannalle. Tämä johtui lähinnä siitä, että TMDB:ssä joidenkin ominaisuuksien käyttöönotto olisi vaatinut käyttäjän rekisteröitymisen TMDB:n omaan sovellukseen. TMDB:n ominaisuuksiin kuuluu esimerkiksi käyttäjän omien elokuvalistojen luominen, joka on myös oman sovellukseni

päätarkoitus. Tämän ominaisuuden käyttäminen vaatisi kuitenkin käyttäjän rekisteröitymisen TMDB:n sovellukseen ja lisäksi joutuisin rakentamaan sovellukseeni toimintalogiikan vähintään käyttäjän kirjautumista varten. Tämä saattaisi osoittautua hankalaksi ja viivästyttää kehitystä, joten en halunnut ottaa riskiä.

Tietokannan valinnassa päädyin SQLite tietokantaan. Valitsin SQLiten, koska se oli erittäin helppo ottaa käyttöön. Siinä on paljon hyödyllisiä ominaisuuksia, kuten valmiita menetelmiä tiedon tallentamiseen tietokantaan sekä vähäinen muistin käyttö. Halusin, että tietokanta olisi mahdollisimman kevyt ja helppo ylläpitää. Muut tietokanta vaihtoehdot olisivat myös olleet hyviä vaihtoehtoja niiden kuvausten perusteella, mutta koska SQLite löytyy valmiina Android Studiosta ilman erillisiä asennuksia, se oli mielestäni paras vaihtoehto.

3.1 Käyttöliittymä

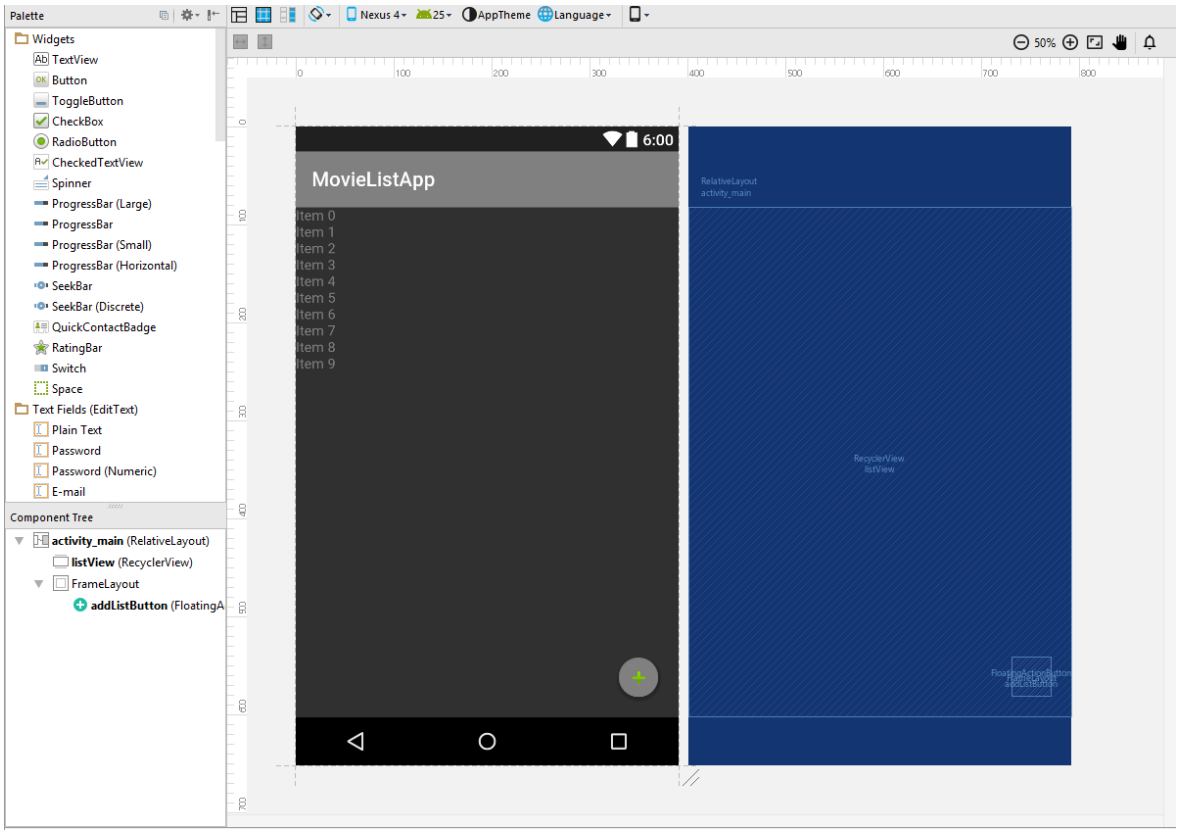
Tavoitteena oli luoda yksinkertainen käyttöliittymä elokuvalista -sovellukselle Android alustalle. Kehityksen alkuvaiheessa harkitsin käyttöliittymän toteutusta yksittäisen näkymän tekniikalla, eli sovelluksessa olisi vain yksi näkymä tai sivu, jonka käyttöliittymä ja sisältö muuttuisi toimintojen mukaan. Päädyin kuitenkin useampaan näkymään, koska se oli mielestäni helpompaa toteuttaa. Lisäksi mahdollinen jatkokehitys ja sovelluksen toimintojen laajennus olisi mielestäni helpompaa useamman näkymän toteutuksessa. Käyttöliittymän kieleksi valitsin englannin kielen. Valitsin englannin kielen suomen kielen sijaan, koska mahdollisesti julkaisen sovelluksen maailmanlaajuisesti, enkä vain Suomessa. Kielen valintaan vaikutti myös se, että sovelluksen käyttämä elokuvatietokanta sisältää vain englanninkielistä materiaalia.

Koska käyttöliittymän tekemisessä pyrin pääosin mahdollisimman yksinkertaiseen toteutukseen. Halusin toteuttaa käyttöliittymän sovelluksen kehitysvaiheen aluksi, siten että vain lähes minimi määrä tietoa näytetään käyttäjälle. Näytettävän tiedon määrää voisi myöhemmin lisätä ja tehdä käyttöliittymään tarvittavia muutoksia uusien tietojen näyttämistä varten.

Sovelluksessa näytettävien elokuvalistojen ja muiden listojen esittämiseksi valitsin RecyclerViewerin. Päädyin tähän, koska RecyclerViewer on monipuolinen, tukee isoa tietomäärää, sekä on laajennettavissa. Lisäksi siinä on hyvät mahdollisuudet muokata käyttöliittymässä esitettävää tietoa.

Koodipuolella käyttöliittymä koostuu käytännössä XML tiedostoista, joihin on määritelty kaikki käyttöliittymän elementit. Jokaista käyttöliittymä näkymää vastaa yksi XML tiedosto.

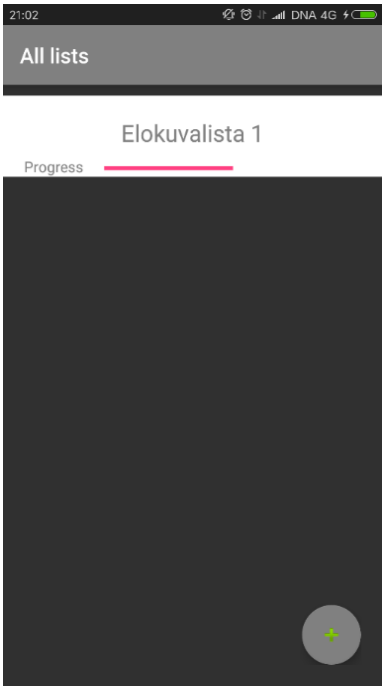
Käyttöliittymän suunnittelu ja luonti olivat yllättävän helppoa. XML tiedostoihin voi lisätä käyttöliittymäosia joko Design näkymän kautta tai kirjoittamalla ne XML koodin muodossa. Design -näkymän kautta uusia käyttöliittymän osia on todella helppo lisäillä ja osat voi asetella todella tarkasti. Lisättävä osa valitaan listasta ja pudotetaan Design -näkymään, jossa on oletuksena puhelimen käyttöliittymää muistuttava kuva (kuva 2). Käyttöliittymän osia voi uudelleen sijoittaa ja niiden kokoa voi myös muuttaa. Suunnittelunäkymä muistuttaa melko paljon Applen Xcode kehitysympäristön näkymää, josta minulla on aikaisempaa kokemusta. Aikaisemman kokemuksen ansiosta vastaavalla työkalulla, käyttöliittymän suunnittelu onnistuu Android Studion Design näkymällä hyvin vaivattomasti. Lisäksi sillä on helppo kokeilla jotain uutta käyttöliittymäosaa ja muutoksen näkee heti.



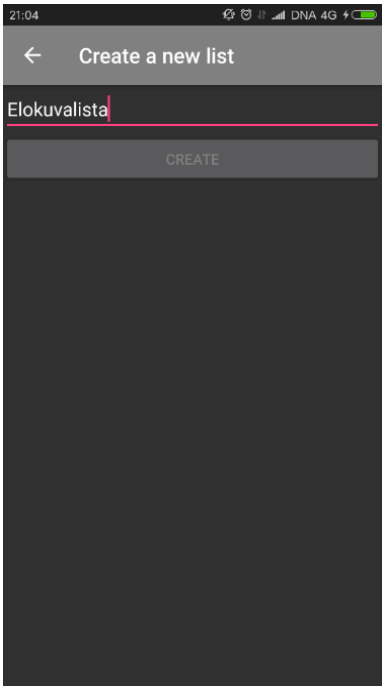
Kuva 2. Android Studion Design näkymä

Kun käyttäjä käynnistää sovelluksen, ensimmäisenä sovelluksessa näkyy kaikki käyttäjän kaikki elokuvalistat (kuva 3). Listoissa näkyvät listojen nimet sekä listojen edistymispalkit katsottujen elokuvien mukaan. Tämä aloitusnäkymä on erittäin yksinkertainen ja tulee vielä todennäköisesti muuttumaan myöhemmin.

Ensimmäistä kertaa käynnistettäessä, näkymässä ei näy yhtään listaa. Käyttäjä voi itse luoda uusia elokuvalistoja. Listoja voi olla yksi tai enemmän. Käyttäjä voi valita näkymästä + -painikkeesta, jolloin käyttäjä siirtyy uuden listan luonti -näkymään (kuva 4). Tässä näkymässä käyttäjä voi antaa uudelle listalle haluamansa nimen. Painamalla luonti -painiketta, sovellus lisää uuden listan tietokantaan.

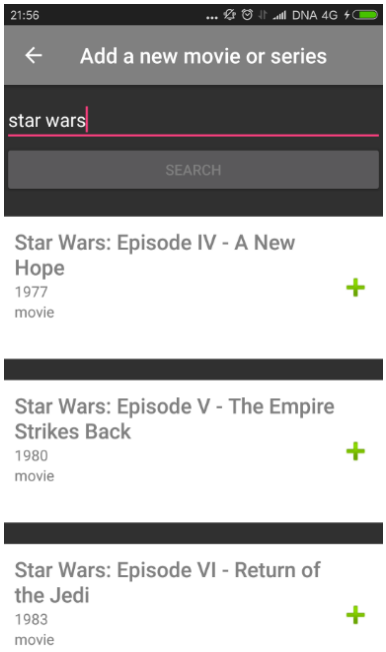


Kuva 3. Elokuvalista esimerkki

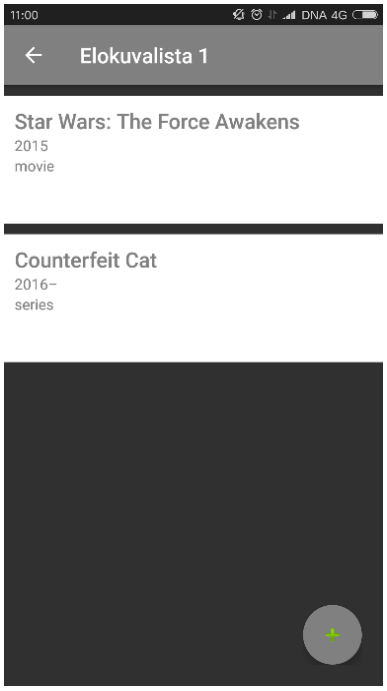


Kuva 4. Elokuvalistan luonti

Listan luonnin jälkeen käyttäjä siirtyy automaattisesti uuden listan näkymään, jossa näkyy kaikki listalla olevat elokuvat. Elokuvia käyttäjä voi lisätä listan näkymässä suuresta + -painikkeesta samalla tavalla kuin uuden listan luonnin yhteydessä. Kun käyttäjä valitsee + -painikkeen lisätäkseen uuden elokuvan, käyttäjä siirtyy elokuvien etsintä -näkymään.



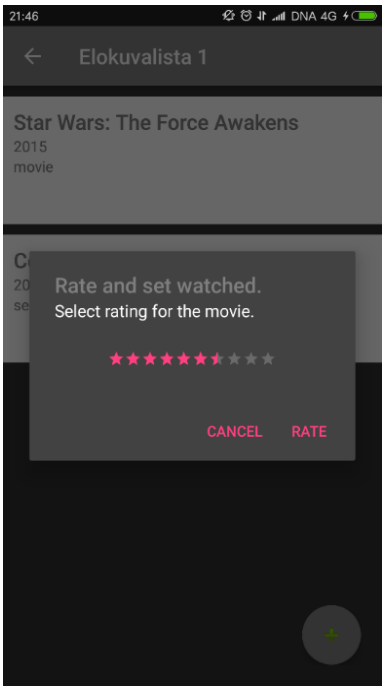
Kuva 5. Elokuvien haku -näkymä



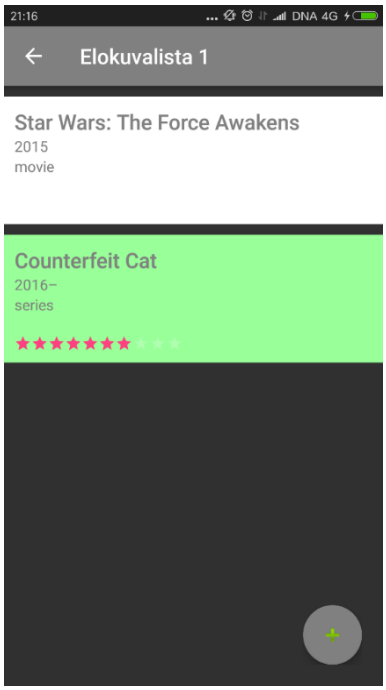
Kuva 6. Lista elokuvista

Sovelluksen elokuvien etsintä näkymässä (kuva 5), käyttäjä voi etsiä elokuvaa tai sarjaa nimen perusteella. Kun käyttäjä syöttää etsittävän elokuvan nimen ja painaa etsi -nappia, sovellus hakee OMDb:n tietokannasta elokuvia nimen perusteella. Löydetyt hakutulokset listataan näkymään allekkain. Hakutuloksia voi olla useita, mutta niitä on kuitenkin rajattu määrä. Käyttäjä voi etsiä elokuvia täyttämällä hakukentän etsittävällä elokuvan nimellä ja valitsemalla etsi -painikkeesta. Myöhemmin tähän on mahdollista lisätä muita hakuvaihtoehtoja.

Elokuvan lisääminen tapahtuu käyttäjän valitsemalla hakutuloksen kohdalta + -painikkeesta. Elokuva lisätään listaan, josta näkymään on siirrytty. Elokuvaa ei voi lisätä, mikäli se jo löytyy listalta. Sovellus tarkistaa, löytyykö elokuva jo listalta ja ilmoittaa, mikäli se on jo listalla. Lisäyksessä elokuva tallennetaan laitteen omaan tietokantaan. Tämän jälkeen käyttäjä siirtyy takaisin lista -näkymään, johon on lisättynä käyttäjän valitsema elokuva (kuva 6). Tällä hetkellä listassa ei ole rajoitusta, montako elokuvaa siihen voidaan lisätä. Tähän olisi kuitenkin hyvä lisätä jokin rajoitus. Listassa ei myöskään näy, milloin elokuva on lisätty listaan, mikä voisi olla hyödyllinen tieto. Listaan voisi myös lisätä oman hakuominaisuuden, jotta elokuvia olisi mahdollista löytää nopeammin, mikäli niitä on listalla todella paljon. Lisäksi, listaan voisi lisäksi tehdä ominaisuuden, että sitä olisi mahdollista järjestää uudelleen jälkeenpäin.



Kuva 8. Elokuvalista ja arvostelut

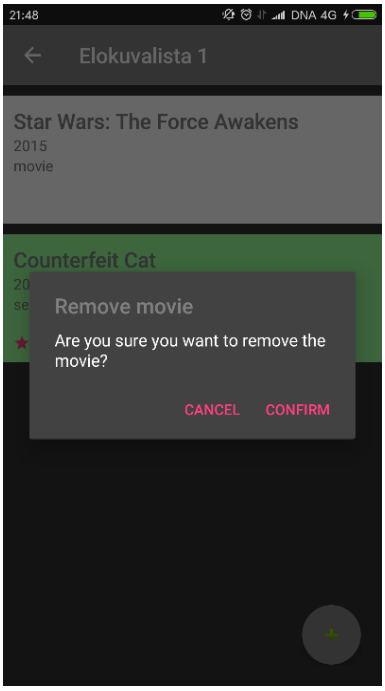


Kuva 7. Elokuvan arvostelu

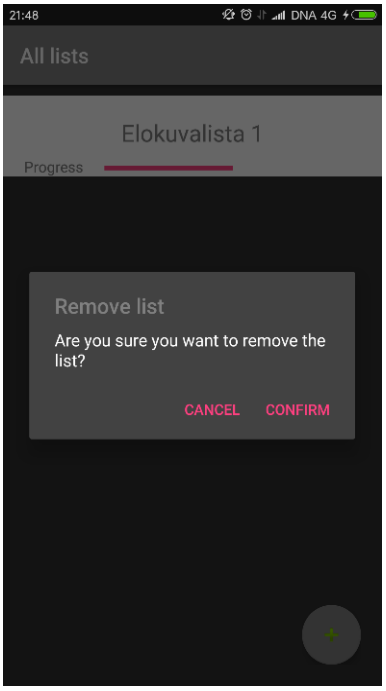
Napsauttamalla elokuvaa listalta, käyttäjä voi merkitä elokuvan katsotuksi sekä arvostella sen (kuva 7). Arvostelu tapahtuu valitsemalla 0-10 tähteä. Kun elokuva arvostellaan ja merkitään katsotuksi, se muuttuu listassa taustaväriltään vihreäksi ja elokuvan kohdalle ilmestyy käyttäjän antama arvosana (kuva 8). Arvosana näkyy alimpana tähtinä elokuvan tiedoissa. Arvostelu on vielä tässä vaiheessa hieman epäselvä, sillä kaikkia tähtiä ei näy listan taustaa vasten muuten kuin tarkasti katsomalla. Arvosteluasteikko saattaa myös muuttua vielä. Vihreä väri ei välttämättä ole paras vaihtoehto merkitsemään, että elokuva on katsottu. Tähän minulla on muutamia erilaisia vaihtoehtoja, joita aion kokeilla myöhemmin.

Käyttäjä voi halutessaan poistaa elokuvan painamalla hieman pidempään elokuvan kohdalta listalla (kuva 9). Sovellus näyttää vielä varmistus -ikkunan poistamisen vahvistukseksi. Poistaminen tapahtuu vasta kun käyttäjä on valinnut poiston vahvistuksen. Elokuva poistetaan elokuvalistasta ja laitteen omasta tietokannasta.

Käyttäjä voi halutessaan poistaa elokuvalistan samaan tapaan kuin yksittäisen elokuvan listalta (kuva 10). Sovellus näyttää poiston yhteydessä vielä varmistus -ikkunan poistamisen vahvistukseksi. Elokuvalista poistetaan myös laitteen omasta tietokannasta ja lisäksi kaikki elokuvalistaan liittyvät elokuvat poistetaan.



Kuva 9. Elokuvan poisto

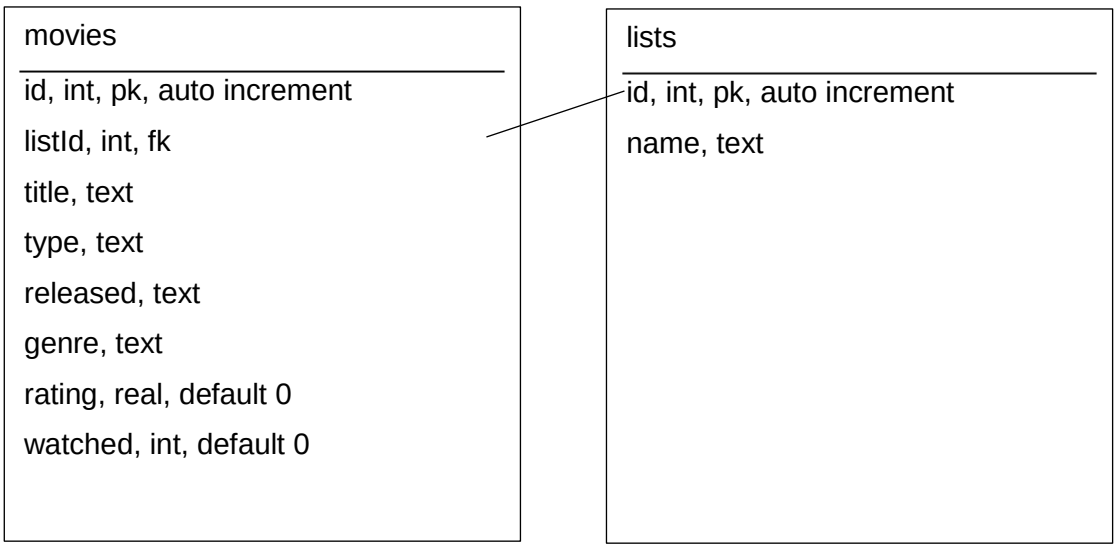


Kuva 10. Listan poisto

3.2 Tietokanta

Sovelluksen tallennusmenetelmäksi valikoitui SQLite. SQLite soveltui mielestäni tähän parhaiten sen helppokäyttöisyyden, sekä hyvien ominaisuuksien takia. SQLitestä on myös saatavilla paljon tietoa ja dokumentaatiota ja se on valmiiksi käytettävänä Android Studion kirjastoissa. Tämä helpottaa tietokannan käyttöönottoa huomattavasti.

Sovelluksen tietokanta koostuu vain kahdesta taulusta kuten kuvassa 11 on esitetty. Taulut ovat ”movies” sekä ”lists”. Elokuva taulussa on kaikki listoihin tallennetut elokuvat, elokuviin liittyvää tietoa sekä viittaus listaan, johon elokuva on tallennettu. Lista tauluun on tallennettu kaikki käyttäjän luomat listat. Lista taulu koostuu id sekä listan nimi sarakkeista. Elokuvat taulussa on id, listan id, nimi, tyyppi, julkaisupäivä, laji, arvosana sekä katsottu -tieto. Huomioitavaa arvosanan tallennuksessa tietokantaan, on että arvosanan arvo saadaan käyttöliittymältä desimaalimuodossa. Arvosana olisi tietenkin mahdollista tallentaa vain kokonaislukuna, jos ei halua käyttää puolikkaita tähtiä arvostelussa. Tauluissa on melko vähäinen määrä tietoa listoista ja elokuvista, mutta sarakkeita tai tauluja on mahdollista lisätä myöhemmin tarpeen mukaan. Sovelluksen tämän hetkisessä kehitysvaiheessa nykyinen määrä tietokantaan tallennettavista tiedoista on kuitenkin riittävä.



Kuva 11. Tietokannan rakenne

Koodipuolella sovelluksessa on DBHelper niminen tietokantaluokka, jossa on määritetty kaikki tietokantaan liittyvät asetukset, tietokannan luontilauseet sekä muut metodit. Tietokantaluokan metodeihin on määritetty esimerkiksi listojen ja elokuvien hakuun, lisäämiseen sekä poistamiseen liittyvät SQL lauseet. DBHelper luokka on jatkettu SQLiteOpenHelper -luokasta, joka on Android Studion SQLite laajennoksen valmis avustajaluokka.

3.3 Elokuvatietokantarajapinta

Elokuvatietokantarajapinnaksi valitsin OMDB elokuvatietokannan. Se on todella monipuolinen ja helppokäyttöinen. OMDB:n kautta on saatavana suuri määrä erilaista tietoa elokuvista. Rajapinnan käyttörajoitus on tällä hetkellä rajaton, mutta se saattaa toki muuttua tulevaisuudessa. Projektin tavoitteena ei ole saavuttaa suuria käyttäjämääriä, joten mahdollisten pyyntöjen rajattu määrä ei ole kovin kriittisessä asemassa. Mikäli käyttöraja tulisi vastaan, ja rajapinta-avain olisi mahdollista päivittää maksua vastaan, olisi varteenotettavaa harkita maksua, jolloin käyttöraja todennäköisesti suurenisi huomattavasti.

3.4 Ongelmat ja haasteet

Opinnäytetyön yhtenä ongelmana oli löytää sopiva avoin elokuvatietokantarajapinta. Vaihtoehtoja oli pari ja niiden väliltä päättäminen ei ollut kovin helppoa. Ominaisuuksien vertailu ja pohdinta vievät paljon aikaa. Omasta mielestä parhaaksi soveltui OMDB:n API, joka on ilmainen ei -kaupalliseen käyttöön. OMDB:n tietokannassa on huomattava määrä elokuvia ja sarjoja, mikä on tärkeää, jotta käyttäjän etsimä elokuva tai sarja todennäköisemmin löytyisi sovelluksen kautta. Rajapinta sisältää todella paljon monipuolista tietoa elokuvista. Tietoa oli oikeastaan enemmän kuin tarpeeksi. Tällä hetkellä käytän kaikesta mahdollisesta tiedosta vain pienen määrän. Lisäksi rajapinta vaikutti todella helpolta käyttää, mikä helpottaa kehitystä huomattavasti.

Eräs toinen ongelma oli androidin API taso. Projektia luodessa Android Studiolla, sovellus kysyy minimi SDK versiota. SDK määrittää projektissa käytettävät kirjastot sekä API tason. Käytännössä tämä tarkoittaa sitä, millä Android versiolla sovellus tulee toimimaan. SDK:n valinta riippuu suurelta osin siitä, mitä ominaisuuksia haluaa käyttää sovelluksessa. Vanhemmat Android versiot eivät välttämättä tue esimerkiksi joitain uusia käyttöliittymäominaisuuksia. Projektia luodessani olin valinnut lähes uusimman kohde Android version. Myöhemmin aloin miettiä asiaa, kun sattumalta huomasin projektin asetuksista ”min SDK level” sekä ”target SDK level” asetukset (kuva 12). Näissä asetuksissa projektissani minulla oli minimissään taso 23 taso ja kohde tasona 25.

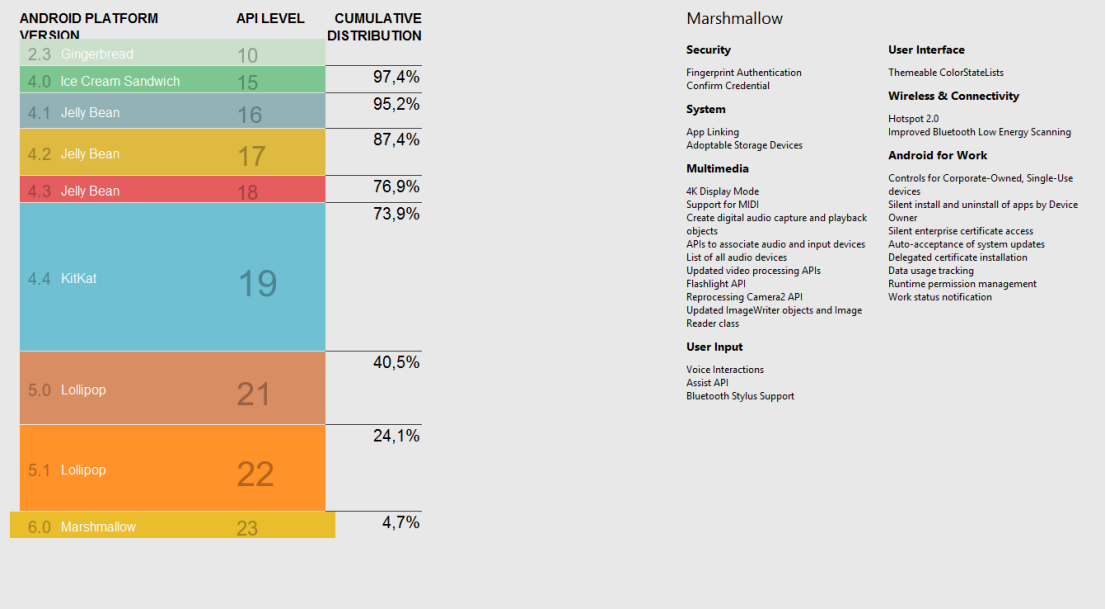
```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 25
    buildToolsVersion "25.0.2"
    defaultConfig {
        applicationId "com.tads.movielistapp"
        minSdkVersion 16
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
    productFlavors {
    }
}

dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:25.1.1'
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:recyclerview-v7:25.1.1'
    compile 'com.android.support:design:25.1.1'
    compile project(':sqlite-android-3170000')
}
```

Kuva 12. Projektin asetuksia ja Android SDK versio

Kuten alla olevasta Android Studion kuvakaappauksesta (kuva 13) näkyy, API, eli rajapintatasolla 23 on vain alle 5% Google Play -kauppaan rekisteröidyistä laitteista. Tätä korkeammista rajapintatasoista ei ole tietoa kuvassa näkyvillä. Jos pitäisin nykyisen API tason projektissani, vain alle 5% mahdollisista käyttäjistä voisi edes asentaa sovelluksen. Päätin tämän tiedon perusteella vaihtaa minimi API tason 23:sta tasoon 16. Valitsin tason 16, koska se oli matalin mahdollinen taso, minkä projektille pystyi asettamaan. Tämä johtui käyttämästäni SQLite tietokannasta, jonka uusimman version minimi vaatima taso on 16. Taso 16 on lisäksi riittävä, sillä se kattaa jo 95,2% prosenttia laitteista. Kohde rajapintatason pidin uusimmassa, eli 25 (Android Nougat) tasossa.



Kuva 13. Android versioiden API tasot ja käyttömäärät, Android Studio

4 Pohdinta

4.1 Jatkokehitys

Olen suunnitellut jatkokehittäväni sovellusta vielä opinnäytetyöprojektin jälkeen melko paljon. Jouduin karsimaan paljon pieniä ja suuriakin ominaisuuksia sovelluksesta, ettei työstä tulisi liian massiivinen. Mielestäni oli hyvä pitää sovellus mahdollisimman yksinkertaisena, jotta sovelluksen voisi julkaista yksinkertaisena mutta toimivana kokonaisuutena.

Kuten aikaisemmin käyttöliittymäkuvauksessa mainitsin, sovelluksen käyttöliittymä on vielä melko yksinkertainen. Tavoitteena olisi tehdä useita parannuksia myöhemmin. Parannuksia olisi käyttöliittymän ulkoasun muuttaminen tyylikkäämmäksi, sillä tällä hetkellä se on melko karu. Käyttöliittymän ulkoasua voisi parantaa jollain teemalla ja väreillä. Lisäksi Käyttöliittymä ei tällä hetkellä välttämättä näytä käyttäjälle kaikkea tietoa elokuvista ja sarjoista mitä käyttäjä haluaisi nähdä. Käyttöliittymä ei myöskään välttämättä viestitä kovin hyvin, kuinka se toimii. Tähän lisäksi olisi ehkä järkevää lisätä ohje-sivu, missä selitetään kaikki sovelluksen toiminnot.

Sovelluksesta puuttuu kaiken edellisen lisäksi vielä eräs melko pieni, mutta tärkeä osa. Siitä puuttuu informaatiosivu, jossa kerrotaan yleistä tietoa sovelluksesta. Sivulla olisi hyvä mainita, mitä tietoa sovellus mahdollisesti kerää käyttäjistä, käyttöoikeuksista,

OMDB:n tarjoamasta palvelusta ja muusta. Mielestäni erityisen tärkeää on kertoa sivulla, että sovellukseni ei kerää tietoja käyttäjistä tai laitteesta.

Uutena suurempana jatkokehityskohteena haluaisin lisätä sovellukseen ominaisuuden, että katselulistoihin voisi lisätä pelkkien sarjojen lisäksi myös sarjojen kausia sekä yksittäisiä jaksoja. Tämä helpottaisi sarjojen seuraamista todella paljon. Ominaisuuden lisäys on kuitenkin melko haastava ja työläs ja sen takia en ole vielä ottanut sitä kehityksen alle.

Uusien ominaisuuksien ja käyttöliittymän ulkoasun lisäksi, on mielestäni tärkeää, että sovellus pysyy toimivana. Pyrin ylläpitämään sovelluksen toimivuutta priorisoimalla bugien korjausta yli uusien ominaisuuksien lisäämisen. Tällä hetkellä sovelluksessa ei ole tullut esiin suuria sovelluksen lamaannuttavia bugeja, mutta pieniä käyttöliittymään liittyviä bugeja löytyy.

4.2 Opinnäytetyöprosessi

Opinnäytetyöprosessi eteni alkuun lähes suunnitellusti. Mielestäni oli hyvä tutkia teoriaa ennen työssä esitetyn sovelluksen toteutusta. Tämä helpotti opinnäytetyön tekemistä huomattavasti. Hieman ennen seminaaria työn kanssa tuli vähän kiire. Seminaarin jälkeen aloitin tekemään korjauksia työhön palautteen perusteella. En kuitenkaan ehtinyt tehdä korjauksia ennen kesälomaa, joten työn palautus jäi syksylle. Muiden kiireiden takia työn palautus viivästyi vielä myöhemmälle.

4.3 Oma oppiminen

Olen ollut kiinnostunut Android ohjelmoinnista jo pitkään. Minulla ei ollut aikaisempaa kokemusta Androidilla ohjelmoinnista. Olen kuitenkin ohjelmoinut vähän iOS:lle aikaisemmin, mutta koin sen melko haasteelliseksi siinä käytettävän ohjelmointikielen takia. Olin yllättynyt kuinka helposti sovelluksen tekeminen onnistui Androidille. Android ohjelmointi tuntui minusta helpolta, mikä johtui varmasti aikaisemman Java kokemukseni ansiosta. Odotin, että sovelluksen tekeminen olisi ollut paljon haasteellisempaa. Androidilla kehittämistä helpotti huomattavasti hyvä Googlen tarjoama dokumentaatio sekä kehittynyt kehitysympäristö.

Opin lopuksi opinnäytetyöstäni mielestäni todella paljon. Opin muun muassa, miten sovellus voidaan toteuttaa Android alustalle ja miten erilaisia käyttöliittymäominaisuuksia voidaan hyödyntää erilaisissa sovelluksissa. Lisäksi opin useita minulle kokonaan uusia teknologioita ja työkaluja, kuten Android Studio, SQLite ja ulkopuolisten rajapintojen käyttö.

Lähteet

Android Studio Features, Google, 2017, 1. Luettavissa: https://developer.android.com/studio/features.html
Luettu: 12.3.2017.

Download Android Studio and SDK Tools, Google, 2017, 2. Luettavissa: https://developer.android.com/studio/index.html#downloads Luettu: 12.3.2017.

ADT Plugin, Google, 2017, 3. Luettavissa: https://developer.android.com/studio/tools/sdk/eclipse-adt.html
Luettu: 12.3.2017.

About SQLite, Hwaci, 2017. Luettavissa: https://www.sqlite.org/about.html Luettu: 12.3.2017.

Chapter 1. Introduction to Berkeley DB, Oracle, 2017. Luettavissa: http://docs.oracle.com/cd/E17076_05/html/gsg/JAVA/introduction.html Luettu: 12.3.2017.

UnQLite, Symisc Systems, 2017. Luettavissa: https://unqlite.org/index.html Luettu: 12.3.2017.

Saving Key-Value Sets | Android Developers, Google, 2017, 4. Luettavissa: https://developer.android.com/training/basics/data-storage/shared-preferences.html Luettu: 12.3.2017

OMDB, Brian Fritz, 2017. Luettavissa: http://www.omdbapi.com/ Luettu: 13.3.2017

TMDb, The Movie DB, 2017 Luettavissa: https://www.themoviedb.org/documentation/api
Luettu: 12.3.2017

Ajax: An New Approach to Web Applications, Jesse James Garrett, 2005. Luettavissa: https://pdfs.semanticscholar.org/c440/ae765ff19ddd3deda24a92ac39cef9570f1e.pdf Luettu: 12.3.2017.

jQuery.ajax() | jQuery API Documentation, The jQuery Foundation, 2017. Luettavissa: http://api.jquery.com/jquery.ajax/ Luettu: 12.3.2017.

AsyncTask | Android Developers, Google, 2017, 5. Luettavissa: https://developer.android.com/reference/android/os/AsyncTask.html Luettu: 12.3.2017.

Creating Lists and Cards | Android Developers, Google, 6, 2017. Luettavissa: https://developer.android.com/training/material/lists-cards.html Luettu: 13.3.2017.

RecyclerView, Google, 2017, 7. Luettavissa: https://developer.android.com/training/material/images/RecyclerView.png Luettu: 13.3.2017.

Lists – Components – Material Design Guidelines. Google, 2017, 8. Luettavissa: https://material.io/guidelines/components/lists.html#lists-usage Luettu:13.3.2017.

Data tables – Components – Material Design Guidelines, Google, 9, 2017. Luettavissa: https://material.io/guidelines/components/data-tables.html# Luettu: 13.3.2017.

Integrated development environment (IDE), Margaret Rouse, 2016. Luettavissa: http://searchsoftwarequality.techtarget.com/definition/integrated-development-environment Luettu: 13.3.2017.

What are the best IDEs for Android development, Slant, 2017. Luettavissa: https://www.slant.co/topics/1321/~ides-for-android-development Luettu: 13.3.2017.

IntelliJ iDEA, JetBrains, 2017. Luettavissa: https://www.jetbrains.com/idea/?fromMenu Luettu: 13.3.2017.

Android.database.sqlite, Google, 2017, 10. Luettavissa: https://developer.android.com/training/basics/data-storage/databases.html Luettu: 13.3.2017.

Gartner Says Worldwide Sales of Smartphones Grew 7 Percent in the Fourth Quarter of 2016, Gartner. Luettavissa: https://www.gartner.com/newsroom/id/3609817 Luettu: 12.11.2017.

Dashboards | Android Developers, Google, 11. Luettavissa: https://developer.android.com/about/dashboards/index.html Luettu: 12.11.2017.

Why do most developers say it's easier to develop apps for iOS than Android?, Trausti Thor Johannsson, 2016. Luettavissa: https://www.quora.com/Why-do-most-developers-say-its-easier-to-develop-apps-for-iOS-than-Android Luettu: 12.11.2017.

Android OS Vs. Apple iOS – Which Is Better for Developers? Lifewire, 2011. Luettavissa: https://www.lifewire.com/android-os-vs-apple-ios-for-developers-2373400 Luettu: 12.11.2017.

Swift, Apple, 2017. Luettavissa: https://developer.apple.com/swift/ Luettu: 12.11.2017.