Titta Riihimäki

Android-sovelluksena toteutettu Anatomian tietopeli

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Insinöörityö

15.3.2015

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Titta Riihimäki Android-sovelluksena toteutettu Anatomian tietopeli 41 sivua 15.3.2015
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja(t)	Tutkintopäällikkö Markku Karhu

Insinöörityön tavoitteena oli laatia opetusmateriaaliksi soveltuva Android-tietopelisovellus, joka pohjautuu ihmisanatomian sanastolle. Sovelluksen toteuttamisessa pyrittiin huomioimaan terveydenhoitoalan opiskelijoiden ja oppilaitosympäristön vaatimukset sekä maksimoimaan sovelluksen oppimista tukevat ominaisuudet, kuten toisto ja motivointi. Sovelluksen laadinta perustui käytännössä havaittuun uusien opetusmenetelmien monipuolistamistarpeeseen.

Sovellus, nimeltään Anatomian tietopeli, esittää käyttäjälleen ihmisanatomian käsitteitä yksi kerrallaan. Käyttäjän on löydettävä käsitteelle oikea latinankielinen käännös. Käyttäjälle tarjotaan hänen valintansa mukaan joko kolme tai kuusi latinankielistä käännösvaihtoehtoa. Käyttäjän on myös mahdollista valita, mitä kehonosia koskevalle käsitteistölle tietopelin kysymykset perustuvat. Peli päättyy kannustavaan tulosyhteenvetoon. Tietopelisovellus saatiin toteutettua suunnitellussa muodossa. Sen todettiin soveltuvan sekä hyöty- että viihdekäyttöön. Laaditun sovelluksen pohjalta on mahdollista kehittää uudenlaisia Android-tietopelejä opiskelukäyttöön.

Avainsanat	Android, oppiminen, peli, anatomia

Author(s) Title Number of Pages Date	Titta Riihimäki Anatomy quiz for Android 41 pages 15 March 2015
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Programming
Instructor(s)	Markku Karhu, Dean

The objective of the project described in this thesis was to develop a quiz of human anatomy that can be used as a teaching aid. The application was implemented for Android. The target user group for the application is health care students. One of the design goals was to maximize learning support features. The driving force behind the application design was the need for finding new teaching methods.

The application presents the user Finnish human anatomy concepts one at the time. The user's task is to find the correct Latin translation. The game shows the user the choice of either three or six Latin translation options. The user can also select the parts of the body on which the questions are based. The game ends with an encouraging result summary. The game application was implemented as designed. It is useful for learning and also entertaining. On the basis of the application it is possible to develop a variety of Android trivia games.

Keywords	Android, learning, game, anatomy

Sisällys

1 Johdanto

Yhteiskuntamme on yhä vahvemmin tukeutumassa teknisiin ratkaisuihin. Älypuhelimet ovat yleistyneet kiihtyvällä tahdilla. Niistä on tullut liki jokaisen kansalaisen arjen perusvälineitä. Opetuksessa älypuhelimia ei juurikaan vielä osata hyödyntää, vaikka opetusteknologisia ratkaisuja onkin kehitelty jo vuosikymmeniä ja teknologiahankinnat ovat vuosien varrella saaneet kasvavan osuuden opetusvälinehankintabudjetista. Tietokoneita, pöytäkoneina, kannettavassa ja tablettimuodossa, on hankittu lisääntyvissä määrin oppilaitosten opiskelijoiden käyttöön. Älypuhelin on hieman erilainen tekninen opetuksen väline, sillä useimmilla opiskelijoista, jopa alakoululaisista, on oma älypuhelin käytössään. Se myös kulkee heidän mukanaan, joten sen käyttäminen on luonnollista erilaisissa tilanteissa.

Tämän insinöörityön tavoitteena on tutkia Android-laitteiston soveltuvuutta opetuskäyttöön sekä kartuttaa työn tekijän valmiuksia opetusteknologian ymmärtämisessä ja kehittämisessä. Työssä toteutetaan ihmisanatomian sanastolle perustuva Android-tietopeli. Ensimmäiseksi työssä pohjustetaan Android-sovelluksen toteuttamista esittelemällä mobiilikäyttöjärjestelmien maailmaa keskittyen Android-käyttöjärjestelmän taustoihin ja toimintaan sekä sovellusten laadintaan. Oma lukunsa on suotu myös teknisien ratkaisujen opetus- ja oppimiskäytön taustojen ja mahdollisuuksien käsittelemiselle. Insinöörityön viimeisessä luvussa, ennen pohdintaa, esitellään Anatomy Game -nimisen Android-tietopelisovelluksen kehittämisen vaiheet.

2 Mobiilikäyttöjärjestelmät

Tietokone tarvitsee toimiakseen käyttöjärjestelmän. Tämä tietokoneen keskeisin ohjelma ladataan tietokoneeseen aina sen käynnistämisen yhteydessä. Käyttöjärjestelmän tarkoituksena on hallita, valvoa ja palvella tietokoneessa käytettäviä ohjelmia eli mahdollistaa niiden käyttäminen lataamalla kyseinen ohjelma esimerkiksi kiintolevyltä muistiin sen suorittamista varten ja tarjoamalla ohjelman käyttöön erilaisia käyttöjärjestelmäpalveluita ohjelman sisältämien toimintojen toteuttamiseksi. Käyttöjärjestelmä myös hallinnoi tietokoneen resursseja, kuten muistia jakaen sitä tarvittavalla tavalla tietokoneen eri osien käyttöön. Käyttöjärjestelmä määrittelee, mitkä ominaisuudet ja toiminnot kyseisessä tietokoneessa ovat käytettävissä (Käyttöjärjestelmä 2014; Mobile Operating Systems (Mobile OS) Explained 2014.)

Mobiilikäyttöjärjestelmä on erityisesti mobiililaitteissa, kuten älypuhelimissa ja tablettitietokoneissa käytettäväksi suunniteltu käyttöjärjestelmä. Mobiilikäyttöjärjestelmä määrittelee käyttöjärjestelmän yleisten tehtävien lisäksi sen, minkä toimittajan sovelluksia kyseiseen mobiililaitteeseen on mahdollista ladata ja mitä sovelluksia käyttää (Mobile Operating System (OS) 2014; Mobile Operating Systems (Mobile OS) Explained 2014.)

Mobiilikäyttöjärjestelmien markkinaosuudet maailmalla ja Suomessa eroavat hieman toisistaan johtuen Windows-käyttöjärjestelmän Suomi-taustasta. Maailmanlaajuisesti mobiililaitemarkkinat kasvoivat vuoden 2014 toisella neljänneksellä yli 25 prosenttia. Markkinoille toimitettiin International Data Corporationin (IDC) mukaan yli 301 miljoonaa uutta mobiililaitetta, mikä on enemmän kuin koskaan aiemmin vuosineljänneksen aikana eikä tahdin ennusteta laantuvan loppuvuonna. Androidin osuus mobiilikäyttöjärjestelmätoimituksista on kasvanut jatkuvasti vuodesta 2011 lähtien. Kesällä 2014 julkaistussa uusimmassa maailmanlaajuisessa markkinatilastossa Androidin osuus on 85 prosenttia siinä missä iOS saavuttaa 12 prosentin ja Windows Phone 2,5 prosentin osuuden. Muiden mobiilikäyttöjärjestelmätoimittajien osuudeksi jää vain sadasosa markkinoista. (Smartphone OS Market Share 2014.) Suomenkin markkinoilla Android on selvästi suurin käyttöjärjestelmätoimittaja (41 %), mutta iOS- (36 %) ja etenkin Windows Phone (22 %) -käyttöjärjestelmien osuudet ovat selvästi suurempia kuin maailmanlaajuisesti tilastoitaessa (Mobiilia liiketoimintaa ja teknologiaa 2014). Mobiilikäyttöjärjestelmämarkkinat ovat viime vuosina keskittyneet voimakkaasti.

2.1 Android

Android on Googlen luoma, Linux-pohjainen, avoimeen ja ilmaiseen lähdekoodiin perustuva käyttöjärjestelmä (Mobile Operating Systems (Mobile OS) Explained 2014). Linuxin valinta Android-käyttöjärjestelmän kehittämisen pohjaksi perustui moniin syihin, joista tärkeimpinä mainitaan Linuxin tarjoama joustavuus. Linux-alustan siirtäminen erilaisiin laitteisiin on yksinkertaista, jolloin sovelluskehitys vapautuu merkittävissä määrin laiteriippuvuudesta. Toiseksi Linux on vuosikymmenien laajojen testausten tuloksena varsin turvallinen järjestelmä, mihin Android pohjaa toimintansa. Kolmantena valintaa puoltavana päätekijänä mainitaan Linuxin monipuoliset ominaisuudet, joille Android-sovelluskehitystä on mielenkiintoista ja hedelmällistä rakentaa. On kuitenkin muistettava erottaa Android Linuxin ilmentymistä, kuten Ubuntusta. Kaikki Linuxin ominaisuudet eivät ole Android-ympäristössä käytettävissä. (Gargenta & Nakamura 2014, 31–33.)

Androidista on laajan yhteistyöverkostonsa (yli 300 laite- ja ohjelmistotoimittajaa) vuoksi tullut nopeasti maailman nopeimmin kasvava mobiilikäyttöjärjestelmä. Androidin suurimpia etuja on laaja ja tiheästi päivittyvä sovellusvalikoima. Useimmat sovelluskehittäjät julkaisevat ohjelmansa ensimmäisenä juuri Android- ja Applen iOS-alustoille. Toinen Androidin etu on käyttäjien selkeäksi kokema käyttöliittymärakenne. (Android, the world´s most popular mobile platform 2014.) Android-käyttöjärjestelmään perehdytään tarkemmin työn seuraavissa luvuissa suurimpien kilpailevien käyttöjärjestelmien esittelemisen jälkeen.

2.2 iOS

Applen valmistamien mobiililaitteiden, kuten iPhonen ja iPadin, toiminta perustuu juuri niitä varten kehitettyyn käyttöjärjestelmään nimeltä iOS. Käyttöjärjestelmää ei ole saatavilla muihin laitteisiin. (Mobile Operating Systems (Mobile OS) Explained 2014; What is iOS? 2014.) Käyttöjärjestelmän uusin versio on nimeltään iOS 8 julkaistiin syyskuussa 2014 ja seuraavan neljän kuukauden aikana sitä päivitettiin neljä kertaa (iOS 8 2015; The biggest iOS release ever 2014). Myös iOS-alustalle, kuten Androidillekin, on tarjolla laajalti erilaisia sovelluksia. (The world´s most advanced mobile OS 2014.)

2.3 Windows Phone

Windows Phone on Microsoftin kehittämä mobiilikäyttöjärjestelmä. Microsoft myy tuotteensa lisenssejä myös muiden laitevalmistajien käyttöön, noudattaen tosin tiukkoja valintakriteerejä. Ensimmäinen Windows Phone, mallimerkinnältään "7", tuli markkinoille loppuvuonna 2010 ja uusin versio Windows Phone 8 julkaistiin loppuvuonna 2012. Nokia valitsi vuonna 2011 Windows Phonen tulevaisuuden älypuhelintensa käyttöjärjestelmäksi. (Windows Phone OS 2014.) Nykyään Nokian matkapuhelintoiminta on osa Microsoftin liiketoimintaa (Microsoft ostaa Nokian puhelinliiketoiminnan 2013; Microsoft and Nokia complete mobile phone unit deal 2013).

3 Android-käyttöjärjestelmän kuvaus

3.1 Android-käyttöjärjestelmän historia

Android on suunniteltu useille erilaisille laitteille soveltuvaksi, kovanäppäimisistä WVGA-puhelimista kosketusnäytöllisiin QVGA-laitteisiin (Meier 2010, 9). Android-käyttöjärjestelmän pohjalle luotavien "älykkäämpien, käyttäjänsä paremmin huomioivien mobiililaitteiden" kehitystyö käynnistyi Kaliforniassa loppuvuodesta 2003 perustetussa Android Inc -yrityksessä, jonka Google osti vuonna 2005. Työntekijät jatkoivat työtään tavoitteenaan edelleenkehittää Linux-pohjaista käyttöjärjestelmää mobiililaitteille. (Android (operating system) 2014; Android versions comparison 2014; Deitel ym. 2012, 4; Meier 2010, 9.) Joulukuussa vuonna 2006 huhuttiin Googlen tulevan mukaan mobiilimarkkinoille (History of Android: First Applications, Prototypes & Other Events 2014).

Googlen mobiilialustojen johtaja Andy Rubin kuvaili marraskuussa 2007 vielä julkaisemattoman Googlen Androidin olevan enemmän kuin puhelin. Rubinin mukaan Android tulisi olemaan ensimmäinen aidosti avoin ja kattava mobiililaitealusta, jossa on käyttöjärjestelmä, käyttäjäympäristö sekä sovelluksia, jotka tulevat kaikki toimimaan ilman kehitystä hidastavia patentointeja. Googlen tarkoituksena oli luoda uusi avoimen lähdekoodin mobiililaitealustastandardi, jonka pohjalta voidaan kehittää erilaisia laitteita moniin eri tarkoituksiin. (Where´s my Gphone? 2014.)

Ensimmäinen nykyisenkaltainen Android-käyttöjärjestelmäpohjainen älypuhelin (HTC Dream, T-Mobile G1) julkaistiin loppuvuonna 2008. (Android (operating system) 2014; Android versions comparison 2014; Deitel ym. 2012, 4; Gargenta & Nakamura 2014, 3; Meier 2010, 9.) Seuraavan vuoden loppuun mennessä Androidia käyttäviä mobiilipuhelimia oli julkaistu jo yli kaksikymmentä ympäri maailmaa. Nyt erilaisia Android-laitteita on jo useita satoja (vuonna 2012 ylittyi 300). Lisenssittömyyden vuoksi Androidin valitseminen on puhelinvalmistajille edullinen ratkaisu, minkä on arveltu edistäneen Androidin nopeaa leviämistä. (Deitel ym. 2012, 4; Meier 2010, 9.) Android-käyttöjärjestelmää hyödyntävistä laitteista suurin osa on nykyään Samsungin toimittamia. Muita toimittajia ovat Huawei, Lenovo ja LG. (Smartphone OS Market Share 2014.) Nykyään Android-käyttöjärjestelmää kehittää useiden yritysten muodostama, vuonna 2007 perustettu Open Handset Alliance, jonka johdossa on Google. Allianssiin kuuluvat lisäksi muun muassa HTC, Motorola, Intel, Qualcomm, Sprint Nextel, T-Mobile ja NVIDIA. (Gargenta & Nakamura 2014, 3; History of Android: First Applications, Prototypes & Other Events 2014.)

3.2 Androidin versioerot ja niiden osuudet

Android nimeää versionsa aakkosjärjestyksessä käyttäen makeiden ruokien nimiä (ks. taulukko 1). Uusin, lokakuussa 2014 julkaistu versio Android 5.0 on nimeltään Lollipop. (Android 5.0 2014.) Kaikista Android-laitteista noin joka sadannessa (1,6 %) oli Lollipop tammi-helmikuun vaihteessa 2015. Jelly Bean (Android 4.1-4.3) oli edelleen laajimmin käytössä oleva (44,5 %) Android-versio. KitKat (Android 4.4.) oli käyttöjärjestelmänä melkein yhtä monessa Android-laitteessa (39,7 %). KitKatin osuus kasvoi Lollipopin julkaisemisen jälkeen nopeasti (lokakuussa osuus oli 25 %). Androidin vanhempia versioita Ice Cream Sandwich ja Gingerbread oli enää alle joka kymmenennessä Android-laiteessa (osuudet 6,4 % ja 7,4 %). Tätä vanhempia Android-versioita käyttäviä laitteita ei juurikaan enää ole. (Dashboards 2015; Android versions comparison 2014; Top android SDK versions 2014.)

Taulukko 1. Androidin versioiden API-tasot ja julkaisuajankohdat (Android 2015; Android versions comparison 2014).

Versio	API-taso	Julkaistu
1.5 Cupcake	3	04/2009
1.6 Donut	4	09/2009
2.0 Eclair	5-7	01/2010
2.2 Froyo	8	05/2010
2.3 Gingerbread	9-10	12/2010
3.0 Honeycomb	11-13	02/2011
4.0 Ice Cream Sandwitch	14-15	10/2011
4.1-4.3 Jelly Bean	16-18	06/2012
4.4 KitKat	19-20	09/2013
5.0 Lollipop	21	10/2014

Android-käyttöjärjestelmän versionkehityksessä on edetty tasaiseen tahtiin ja pienin parannuksin. Esimerkiksi Ice Cream Sandwich (Android 4.0-4.0.4) uudisti Android-käyttöliittymän. Jelly Bean (Android 4.2) mahdollisti useamman käyttäjätilin samassa laitteessa sekä elekirjoituksen. Sen päivitetty versio Android 4.3 toi käyttäjälle mahdollisuuden luoda rajoitettuja käyttäjätilejä. KitKat (Android 4.4) keskittyi käyttöliittymän keventämiseen ja akunkeston parantamiseen. (Android 2014; Android versions comparison 2014; Gargenta & Nakamura 2014, 5–7.) Uusin tulokas, Lollipop (Android 5.0) yksinkertaistaa näyttönäppäimiä sekä lanseeraa uuden Asetukset-sovelluksen (Android 2014; Android versions comparison 2014).

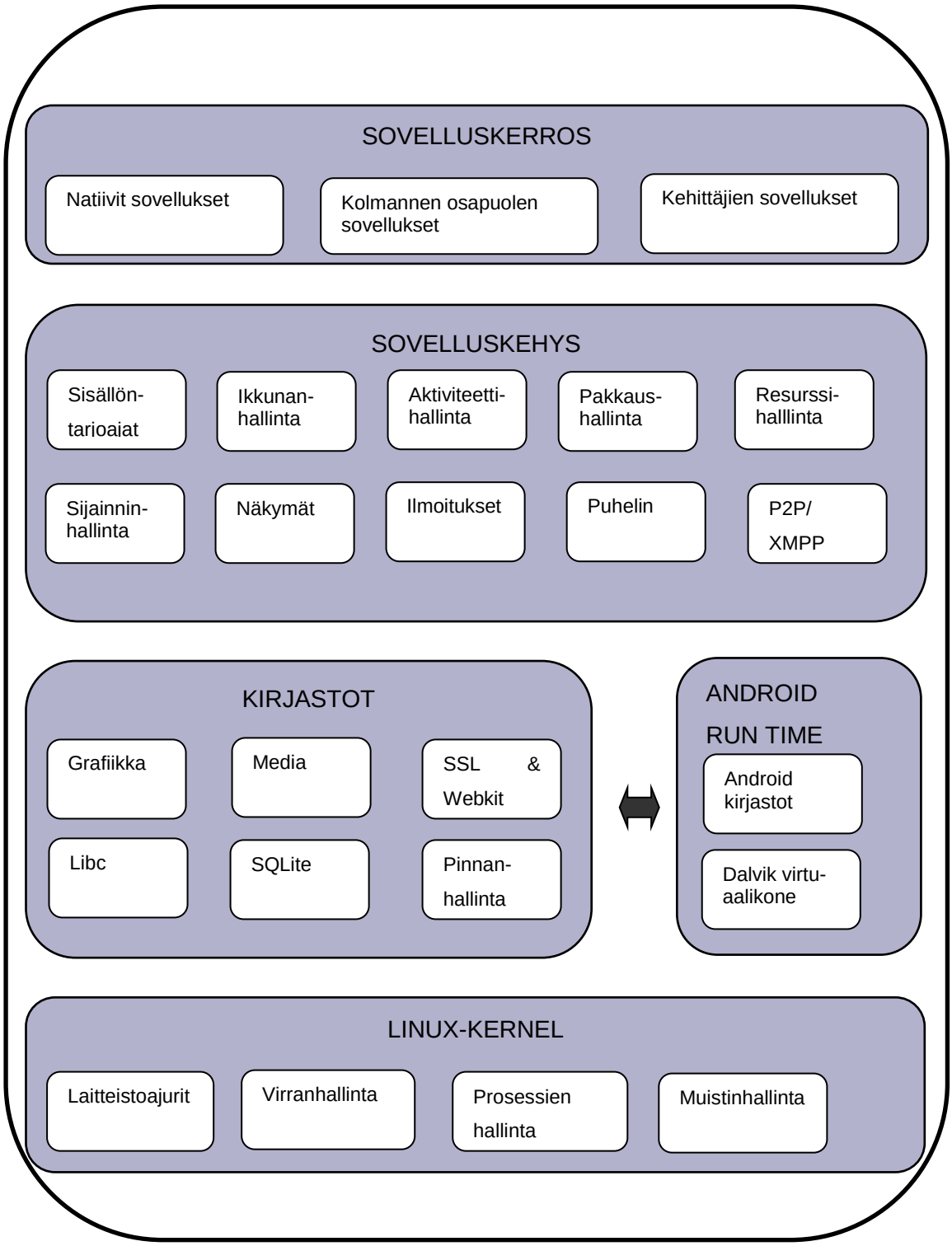
3.3 Androidin arkkitehtuuri

Yksinkertaisesti kuvattuna Android on kolmen komponentin yhdistelmä. Ensinnäkin siinä on ilmainen, avoimeen lähdekoodiin perustuva, mobiililaitteille tarkoitettu käyttöjärjestelmä. Toiseksi se tarjoaa mobiililaitteiden sovelluskehitykselle avoimen lähdekoodin alustan. Kolmantena tekijänä ovat laitteet, etenkin mobiilipuhelimet, jotka on tarkoitettu käyttämään näitä kahta edellä mainittua komponenttia. (Meier 2010, 4.)

Tarkemmin kuvattuna Android pohjautuu Linux-käyttöjärjestelmään, jonka kernel mahdollistaa mobiililaitteille optimoidun muistinhallinnan ja prosessien hallinnan. Linux-kernel sisältää perustavanlaatuiset palvelut, kuten laitteistoajurit sekä turvallisuuden, verkon- ja virranhallinnan. Kernel myös tarjoaa yhteyden (Abstraction Layer) laitteiston ja käskypinon välille. Avoimen lähdekoodin kirjastot tukevat ohjelmistokehitystä tarjoamalla käyttöön muun muassa SQLiten (tietokanta), WebKitin, OpenGL:n (Open Graphics Library, graafinen 3d-rajapinta) ja mediamanagerin. Pieneksi ja tehokkaaksi rakennettu Android Run time aikaansaa Androidille tyypillisen toiminnan. Siellä ajetaan ja isännöidään sovellukset käyttäen Dalvikin virtuaalikonetta sekä pääkirjastoja. (Meier 2010, 4, 13.) Androidin arkkitehtuuri esitetään graafisena kuviossa 1.

Android rakentuu Linux-kernelin päälle. Seuraavana kerroksena ovat kirjastot ja Android Run Time, joiden päälle on rakennettu ohjelmistokehys. Ohjelmistokehys tarjoaa rajapinnan sovelluskehittäjille. Rajapinnan kautta sovelluskehittäjälle mahdollistuu esimerkiksi GPS-paikannustietojen tarkastelu. Tässä kerroksessa sijaitsevat eri toimintoja hallinnoivat managerit. Sovelluskehys (Application Framework) tarjoilee sovellusten toteutuksessa käytettävät luokat. Tämän ohella se mahdollistaa geneerisen yhteyden laitteistoon sekä hallinnoi käyttäjärajapintaa ja sovelluksen resursseja. (Android Architecture 2014; Gargenta & Nakamura 2014, 31–33; Meier 2010, 14.) Android-arkkitehtuurin ylimmäisenä kerroksena on itse sovellus. Sovelluskerroksessa (Application Layer) luodaan kaikki sovellukset, mukaan lukien sekä natiivi- että kolmannen osapuolen sovellukset, käyttäen sovelluskehyksen tarjoamia luokkia ja palveluita. Kaikki sovellukset ajetaan sovelluskerroksen tasolla. (Android ohjelmointi. Mobiiliohjelmointi 2014; Meier 2010, 14.)

Kirjastot tarjoavat erilaisia ohjelmiston osia ja aputoimintoja, kuten tietokannan sovelluskehityksessä hyödynnettäviksi. Java-ydinkirjasto sisältää kaikki Javan tärkeimmät kirjastot. Androidissa on erilaisia C/C++ -peruskirjastoja, kuten libc ja SSL sekä kirjastoja muun muassa median toistoon sekä grafiikoiden ja tietokantojen luomisen mahdollistamiseksi. (Android ohjelmointi. Mobiiliohjelmointi 2014; Meier 2010, 13.)



Kuvio 1. Androidin arkkitehtuuri (Gargenta & Nakamura 2014, 8; Meier 2010, 14, mukaeltu).

Android Run time on osio, jonka ansiosta Android-puhelin on puhelin eikä vain Linuxin kannettava toteutus. Run timessa sijaitsevat sekä ydinkirjastot että Dalvik-virtuaalikone, josta jokaiselle sovellukselle käynnistetään oma instanssi. Tämä muodostaa sovellusten ajamisen perusteet. Androidin ydinkirjastot aikaansaavat suurimman osan Java-kirjastojen ja Androidin omien erityiskirjastojen toiminnallisuuksista. (Android ohjelmointi. Mobiiliohjelmointi 2014; Meier 2010, 13.)

Android-ohjelmat käyttävät räätälöityä Java-virtuaalikonetta nimeltään Dalvik. Vaikka Android onkin ohjelmoitu Java-kielellä, ei Dalvik ole Java-virtuaalikone. Dalvik on rekisteripohjainen virtuaalikone, joka on optimoitu mahdollistamaan laitteelle useampien samanaikaisten instanssien tehokkaan toiminnan. Tämä pohjautuu Linux-kernelin säikeistykseen ja kevytrakenteiseen muistinhallintaan. (Meier 2010, 13.) Virtuaalikone sisältää muun muassa ajuri-, prosessinhallinta-, tietoturva-, käyttäjäoikeus- sekä muistin- ja virranhallintatoimintoja. Virtuaalikoneen käyttäminen helpottaa Android-sovellusten eri laitteille sopivuuden määrittämistä, sillä samaa sovellusta voidaan sen avulla ajaa kehityksen aikana erilaisilla laitteilla. (Android ohjelmointi. Mobiiliohjelmointi 2014.)

Androidin sovellukset voidaan luokitella karkeasti neljään osaan. Foreground-ryhmän sovellukset ovat käyttökelpoisia vain ollessaan etualalla aktiivisena, muulloin niiden suoritus keskeytyy (esim. pelit). Background-ryhmän sovelluksille on tyypillistä rajoitettu vuorovaikutus ja että ne ovat taustalla suurimman osan käynnissäoloajastaan. Kolmantena ryhmänä ovat Intermittent-sovellukset. Ne toimivat vain kutsuttaessa ja tekevät suurimman osan työstään taustalla informoiden käyttäjää tarvittaessa (esim. mediasoitin). Neljäntenä on vielä Widget-ryhmä, jonka sovellukset ovat aloitusnäytön kuvakkeita. (Meier 2010, 29.)

Foreground-ryhmän sovellukset voivat siirtyä etualalta taka-alalle useamman kerran käynnissäoloaikanaan. Sovelluskehityksen yksi tärkeimmistä kohteista onkin tehdä siirtymisistä sujuvia. Androidin sovellukset eivät itse juurikaan voi vaikuttaa omaan käynnissäoloaikaansa (life cycle). Androidin resurssinhallinnalla on mahdollisuus sulkea passiivisen, taka-alalla oleva sovellus milloin vain. Tämän vuoksi Foreground-ryhmän sovellusten luomisessa on kiinnitettävä tarkasti huomiota sovelluksen tilan tallentamiseen aina kun se poistuu etualalta. Tallentaminen mahdollistaa sovelluksen palauttamisen etualalle samassa tilassa. (Meier 2010, 29.)

Background-ryhmän sovellukset toimivat liki ilman käyttäjän komentoja hiljaisina taustalla. Tyypillisesti sovelluksen tehtävänä on rekisteröidä laitteiston, systeemin tai toisten sovellusten viestejä sekä toimintoja. Sovellukset voitaisiin tehdä käyttäjälle täysin näkymättömiksikin eli automaattisesti toimiviksi, mutta sovelluskehityksessä on päädytty säilyttämään kevyt käyttäjäkontakti. Käytännössä on katsottu tarpeelliseksi tarjota käyttäjälle mahdollisuus vahvistaa sovelluksen käynnistyminen, tauottaa sitä tai sulkea sovellus. Tyypillisiä taka-alan sovelluksia ovat palvelut (Services) ja herätteen vastaanottajat (Intent Receivers). (Meier 2010, 29–30, 285–286.)

Intermittend-ryhmän sovelluksilla mahdollistetaan muun muassa sähköpostitoiminnot. Sovellus reagoi käyttäjän syötteeseen, mutta toimii aktiivisesti myös taka-alalla ollessaan. Sen on tiedettävä tilansa käyttäjän sitä kysyessä. Näin esimerkiksi sähköpostiliikenne pidetään ajantasaisena ja käyttäjän avatessa sovelluksen etualalle, ovat uusimmatkin sähköpostiviestit näkyvillä. Kyseessä on oikeastaan näkyvien aktiviteettien ja näkymättömien taka-alan palveluiden yhteensulauma. (Meier 2010, 30.)

Widget-ryhmän tarkoituksena on tarjota käyttäjälle mahdollisuus lisätä aloitusnäytölleen haluamansa kuvakkeet linkeiksi valitsemiinsa sovelluksiin. Dynaamista tietoa sisältävä sovellus, kuten esimerkiksi päiväyksen tai kellonajan näyttö, voi koostua kokonaisuudessaan widgetistä. (Meier 2010, 30.)

3.4 Android-sovelluskehityksessä käytetyt kielet

Android-sovellusten ohjelmoinnissa käytetään yleisimmin Java-ohjelmointikieltä toimintojen ohjelmointiin ja XML-ohjelmointikieltä käyttöliittymän toteuttamiseen (Android ohjelmien tekeminen). Sovellusohjelmoinnin pääkielenä on Java, mutta joiltain osin on mahdollista käyttää myös C-ohjelmointikieltä. Java on luonnollinen valinta pääohjelmointikieleksi, sillä se on maailman eniten käytetty ohjelmointikieli, tehokas, ilmainen ja avoimeen lähdekoodiin perustuva (Deitel ym. 2012, 5). Java-kielen käytön etuna on automaattinen muistinhallinta, jolloin muistivarauksia ei tarvitse tehdä. Toisena etuna on sovelluskehittämisen helppous ja edullisuus, kun saatavilla on useampiakin ilmaisia Java-kehitysympäristöjä, kuten esimerkiksi Eclipse ja Netbeans. (Android ohjelmointi. Mobiiliohjelmointi 2014.)

Androidissa käytetty Java eroaa niin sanotusta tavallisesta Javasta siinä määrin, että sillä on omat kirjastot esimerkiksi grafiikan toteuttamiseen näytöllä sekä tietokantojen ylläpitoon. Java ohjelmointikielenä sisältää runsaasti käyttökelpoisia ominaisuuksia, kuten rinnakkaisuuden hallinnan, runsaat rajapinnat, verkko-ominaisuudet sekä graafisen käyttöliittymäkirjaston. Tämä mahdollistaa yhdessä ympäristössä luodun Java-sovelluksen ajamisen periaatteessa kaikissa muissa Javaa tukevissa ympäristöissä. (Android ohjelmien tekeminen 2014.)

Aktiviteettien käyttäjäympäristö laaditaan yleensä XML-kielisinä (eXtensible Markup Language) tiedostoina (Android application components overview 2014). XML-kieli on tekstimuotoista ja muistuttaa jossain määrin internetsivujen kirjoittamiseen käytettyä HTML-kieltä. XML-ohjelmointikieltä kutsutaan merkintäkieleksi. Sitä käyttäen tiedon merkitys voidaan kuvata tiedon yhteyteen. Sillä kuvataan tiedon rakenne ilman ennalta määrättyjä koodeja. XML-kielellä voi sen sijaan muodostaa koodeja, joilla voidaan puolestaan luoda dokumentteja monenlaisiin tarkoituksiin. XML-kieltä voidaan käyttää järjestelmien väliseen tiedonvälitykseen sekä laajojen tietomassojen jäsentämiseen ja dokumenttien tallentamiseen. Android-sovelluskehityksessä XML-kielellä luodaan sovelluksen käyttöliittymä sekä kuvataan ohjelmistoprojektin rakennetta, ominaisuuksia ja oikeuksia. Viimeksi mainittu tieto tallennetaan AndroidManifest.xml -tiedostoon (Android ohjelmien tekeminen 2014.)

3.5 Android-sovelluskehityksen vaatimukset

Androidin markkinoille tuleminen yksinkertaisti mobiililaitteiden sovelluskehitystä merkittävästi. Mobiililaitteen ohjelmoinnin rajoitukset on kuitenkin edelleen ymmärrettävä ja huomioonotettava sovelluksia luotaessa. Sovelluskehittäjän tärkeimpiä lähtökohtia on huomioida kehitystyössään kohdelaitteen näytön ja muistin pöytätietokonetta pienemmät koot sekä varastointi- ja prosessorikapasiteettien rajoittuneisuus. Pöytätietokoneeseen verrattuna mobiililaitteen ohjelmoinnissa rajoitteeksi nousevat tiedonsiirron hitaus sekä myös kalleus, tiedonsiirtoyhteyksien epävarmuus ja akkukäyttöisyys, jonka seurauksena sovellusten virrankäyttöä on suunniteltava tarkemmin. Toki monia näistä rajoituksista on saatu vähennettyä uusien mobiililaitteiden kehitystyössä vuosien varrella, esimerkiksi näytön tarkkuus on kasvanut mahdollistaen aiemmasta poikkeavien sovellusten toteuttamisen. Sovellukset on silti edelleen parasta tehdä muistaen näytön pienehkö koko sekä käyttäjän laitteen vilkaisemistaipumukset. Sovellusten intuitiivisuutta

ja helppokäyttöisyyttä kannattaa lisätä vähentämällä käyttäjältä vaadittujen toimien määrää sekä keskittämällä tärkeimmät tiedot näytön keskelle ja etuosaan. Ohjelmoimalla näyttöjen sormella kosketettavat valikko-osat riittävänkokoisiksi voidaan edelleen lisätä käyttäjäkokemuksen miellyttävyyttä. (Meier 2010, 30–38.)

4 Android-sovelluskehityksen pääpiirteet

Android-sovelluskehityksen aloittaminen onnistuu liki kustannuksitta, sillä Android pohjautuu vapaan lähdekoodin aineistolle ja välineistölle. Sovelluskehitykseen tarvittavan ohjelmiston voi ladata internetin kautta useammastakin eri lähteestä ilmaiseksi. Myös Android-sovelluskehityksen tutoriaaleja löytyy kiitettävässä määrin internetistä useammilla eri kielillä ja eritasoisille sovelluskehittäjille.

4.1 Android-kehitysympäristön perustamisprosessi

Android-sovellusten yleisin kehittämisympäristö on avoimen lähdekoodin Eclipse-ohjelma, johon on asennettava Androidin Software Development Kit (SDK). SDK on ohjelmallinen paketti, joka sisältää sovelluskehittäjän tarvitsemat API-kirjastot ja muut työkalut sovelluksen laatimiseen, testaamiseen sekä virheiden paikantamiseen. Kyseessä on tavallaan sovelluskehittäjän kokonaisvaltainen työkalupakki, jota voi täydentää monenlaisin lisäosin. Esimerkiksi Eclipse Android Development Tools (ADT) -paketin asentamalla käyttäjä saa kaiken tarvitsemansa Android-sovelluskehittämisen aloittamiseksi. (Android Studio 2014; Deitel ym. 2012, 18–19; SDK - software development kit 2014.) Sovelluskehitysympäristö voidaan asentaa vaihtoehtoisesti Windows-, Mac- tai Linux-alustalle.

Android SDK sisältää mobiililaitteiden emulaattorin, joka on tietokoneessa toimiva virtuaalinen mobiililaite. Emulaattoria käytetään hiirellä ja sen toiminta vastaa fyysistä Android-mobiililaitetta. Laite-emulaattorin näytön koon, laitetyypin sekä API-tason voi valita emulaattorin luomisen yhteydessä Eclipsessä. Tämä mahdollistaa kehitettävän sovelluksen testaamisen useammilla eri laitteilla kuin olisi mahdollista käytettäessä konkreettisia laitteita. Emulaattori helpottaa sovelluksen testaamista ollen oivallinen mobiilisovellusten kehittämisen tuki. (Android Emulator 2014; Android Studio 2014; Deitel ym. 2012, 18–20; Gargenta & Nakamura 2014, 59; Näin pääset Android kehityksessä alkuun 2014.)

4.2 Android-sovelluksen perusrakenne

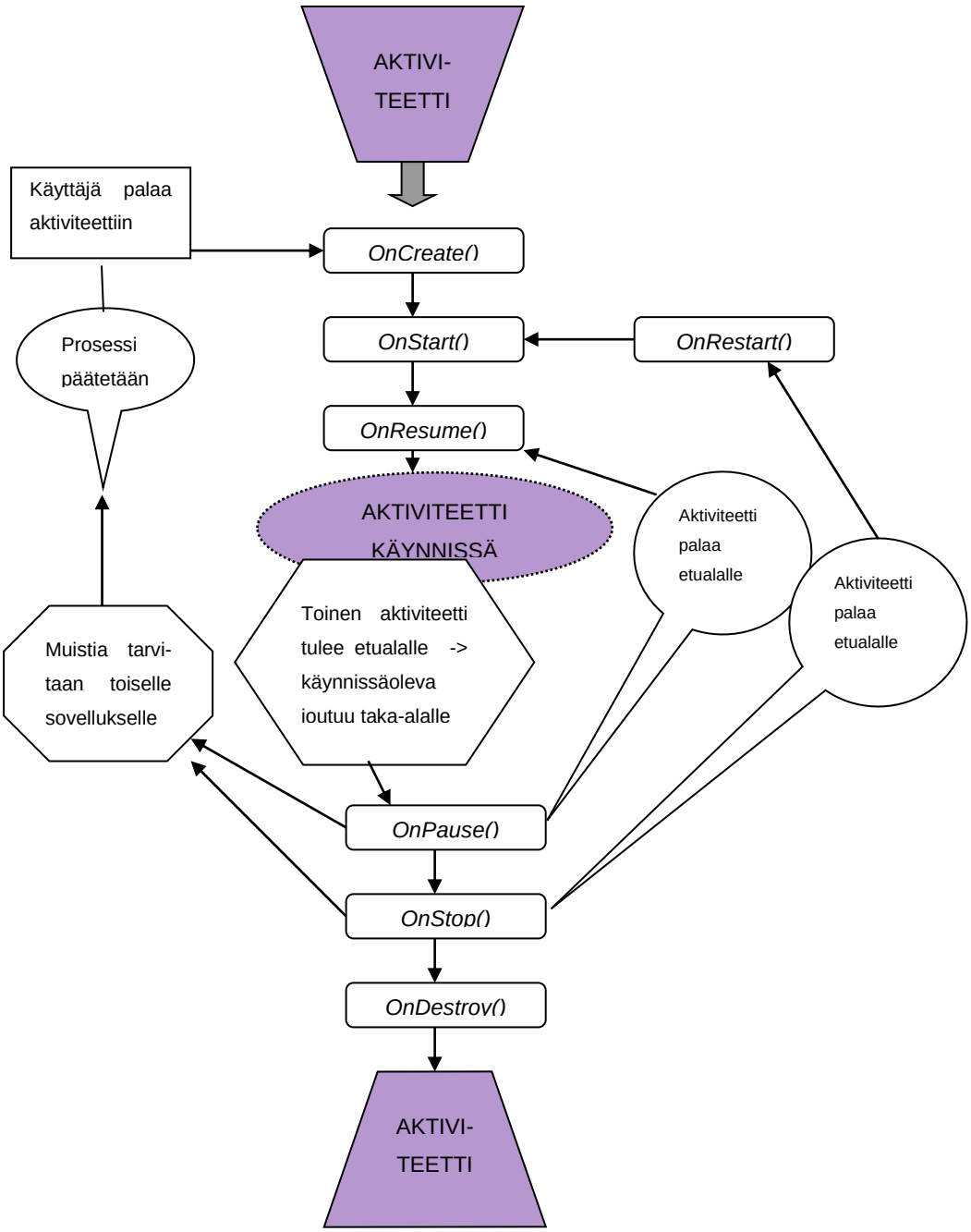
Android-sovellus koostuu Android-ohjelmistokomponenteista ja resurssitiedostoista (ks. taulukko 2). Android-sovelluksella on sovellusluokka, joka asennetaan välittömästi sovelluksen käynnistyessä ja se on viimeinen komponentti, joka lopetetaan ohjelman sulkemisen yhteydessä. Android-sovellus on yksittäisenä asennettava yksikkö, joka voidaan käynnistää ja sitä voidaan käyttää itsenäisesti toisista Android-ohjelmasovelluksista riippumatta. (Android application components overview 2014.) Lähtökohtaisesti jokainen sovellus toimii omassa prosessissaan, omassa paikassaan Dalvik-virtuaalikoneessa. Muistin- ja prosessinhallinta hoidetaan Run time -tasolla. Android-sovelluksella on perinteisestä ympäristöistä poiketen rajattu mahdollisuus vaikuttaa omaan elinkaareensa eli käynnissäoloaikaansa (life cycle). Android hallinnoi rajusti resurssejaan tehden kaiken tarvittavan laitteen toimivuuden takaamiseksi. Tämän vuoksi prosessit sovelluksineen voidaan päättää varoittamatta, jos niiltä on vapautettava resursseja korkeamman prioriteetin sovellusten käyttöön. Yleisimmin sovellukset, joille resursseja vapautetaan, ovat laitteen käyttäjällä juuri aktiivisessa käytössä. (Meier 2010, 57–59.)

Taulukko 2. Android-sovelluksen perusrakenteeseen kuuluvat projektikansiot (Android ohjelmointi. Mobiiliohjelmointi 2014; App Structure 2014).

Projektikansio	Tehtävä
src	Lähdekoodi
gen	Automaattisesti generoidut javatiedostot (R.java)
res	Resurssikansio, alakansioina mm. testit sisältävä string.xml
Manifest	Sovelluksen perusmääritykset

Android-sovelluksen rakenne riippuu sen sisältämistä toiminnoista ja sisällöstä. Yhden aktiviteetin sovelluksessa kaikki toiminnot tapahtuvat samalla näytöllä. Kamera- ja laskinsovellukset ovat tästä hyviä esimerkkejä. Tyypillisesti sovellukset sisältävät ja käyttävät kuitenkin useampia näyttöjä. Useimmiten Android-sovelluksessa on kahdentasoisia näyttöjä: ylätason ja yksityiskohta/muokkausnäytöt. Ylätason näytöt esittelevät laitteessa käytössä olevat toiminnot. Tämän näytön luomiseen on syytä paneutua huolelli-

sesti, sillä se antaa käyttäjälle ensivaikutelman koko sovelluksesta, sen visuaalisesta ilmeestä, helppokäyttöisyydestä ja tarkoitukseen sopivuudesta. Yksityiskohtaisemmalla, alemman tason näytöllä, käyttäjä voi tarkastella ja muokata sen sisältämää tietoa. (App Structure 2014.)



Kuvio 2. Aktiviteetin elinkaari (Android Studio 2014, mukaeltu).

Näytöillä on ominaisuuksia, joita voidaan käyttää niiden olemassaolemisen tai toiminnan varmentamiseksi. Näyttörypäs (ViewGroup) vastaa toisten näyttöjen järjestämisestä, minkä vuoksi sitä kutsutaan myös näytönjärjestelijäksi (Layout Manager). Näytönjärjestelijät kirjoitetaan yleensä android.view.ViewGroup-nimisiksi luokiksi ja ne määrätään perimään näyttöjen perusominaisuudet määrittelevä android.view.View-luokka. Näytönjärjestelijät voidaan sulauttaa monimutkaisten näyttöjen asemointien luomiseksi. Ikoneita (Widgets) käytetään pääasiassa Android-sovellusten aloitusnäytöillä. Ne ovat interaktiivisia osia, joita useimmiten käytetään linkkinä tiettyyn toimintoon. Ikoni voi näyttää kohteen perustietoja (esim. lyhyen yhteenvedon uusista sähköposteista) tai vain siirtää käyttäjän haluttuun toimintoon. (Android application components overview 2014.)

Sovelluskomponentteja on kuusi erilaista: aktiviteetti (Activity), palvelu (Service), sisällöntoimittaja (Content Provider), heräte (Intent), tapahtumakuuntelija/ -vastaanottaja (Broadcast Receivers), ikoni (Widget) ja huomautus (Notification). Jokaisella komponentilla on oma tarkoituksena ja oma elinkaarensa. Aktiviteetin elinkaari on esitetty kuviossa 2 edellisellä sivulla. Aktiviteettia voisi kutsua sovelluksen esityskerrokseksi. Se on visuaalinen Android-sovelluksen esitysmuoto. Sovelluksella voi olla useampia aktiviteetteja. Ne käyttävät näkymiä (views) ja fragmentteja luodakseen käyttäjäympäristön, jossa ne toimivat vuorovaikutteisesti käyttäjän kanssa. (Activities 2014; Android application components 2014; Android application components overview 2014; Android ohjelmointi. Mobiiliohjelmointi 2014; Meier 2010, 50–51.)

Palvelut toimivat taka-alalla, käyttäjälle näkymättöminä työläisinä päivittäen tietosisältöjä ja aktiviteetteja sekä luoden huomautuksia. Niitä käytetään ylläpitämään säännöllisiä prosesseja, joiden on jatkuttava sovelluksen aktiviteettien ollessa passiivisina tai taka-alalla. Palvelu suorittaa tehtäviä riippumatta käyttäjäympäristöstä. Palvelut voivat kommunikoida toisten Android-komponenttien kanssa. (Android application components overview 2014; Meier 2010, 50, 286.) Android sovellusarkkitehtuurissa sovelluspalvelut jakautuvat viiteen pääryhmään, jotka esitellään taulukossa 3. (Meier 2010, 15.)

Taulukko 3. Sovelluspalveluiden viisi pääryhmää (Meier 2010, 15).

Palvelu	Tehtävä
Activity Manager	Kontrolloi sovellusten elinkaarta
Views	Sovelluksen käyttäjäympäristön luomisen työkalu
Notification Manager	Tarjoaa tasalaatuisen käyttäjän tavoittamismenetelmän
Content Provides	Mahdollistaa datan jakamisen
Resource Manager	Tukee koodaamattoman materiaalin julkaisemista (string, grafiikka)

Sisällöntoimittajat määrittelevät sovelluksen tiedolle järjestelmällisen rakenteen. Ne toimivat jaettavina tietovarastoina hallinnoiden ja jaotellen sovelluksen tietokantaa. Tärkeimmät sisällöntoimittajat ovat valmiiksi ohjelmoituina Android-laitteissa, mutta sovelluskehittäjä voi luoda myös omia sisällöntoimittajia. Räätälöidyillä sisällöntoimittajilla voidaan mahdollistaa tietokannan jakaminen myös sovellusten välillä. Android-järjestelmä sisältää tiedon tallentamisen mahdollistavan SQLite-tietokantarakenteen, jota usein käytetään sisällöntoimittajien yhteydessä. (Android application components overview 2014; Meier 2010, 50.)

Herätteet mahdollistavat sovellusten välisen viestinvaihdon ja tehtävien toteuttamisen. Näin Android-sovelluksen komponentit voivat olla yhteydessä toisten Android-sovellusten komponenttien kanssa. Herätettä käyttämällä sovelluskehittäjän on mahdollista lähettää määrätyn toiminnon suorittamiseen johtava viesti joko koko systeemille tai vain valikoidulle aktiviteetille tai palvelulle. Vastaanottajat (Broadcast Receivers) voidaan rekisteröidä kuuntelemaan systeemin viestejä sekä herätteitä mahdollistaen tarvittaessa niihin reagoinnin. Tällä menetelmällä voidaan välittää tieto esimerkiksi saapuvasta puhelusta tai laitteen käynnistyksen päättymisestä valmiustilaan. (Android application components overview 2014; Meier 2010, 50–51, 57–59.)

Ilmoitukset muodostavat käyttäjäviestinnän kehyksen. Ilmoituksella sovellus voi viestiä käyttäjälle häiritsemättä tämän toimintaa ja keskeyttämättä meneillään olevaa aktiviteettia. Ilmoitus voi tulla esimerkiksi saapuneesta uudesta viestistä ja käyttäjä voi tarkastaa sen haluamallaan aikataululla. (Meier 2010, 51, 286.) Fragmentit ovat aktiviteetin kontekstissa toimivia komponentteja. Fragmentti kapseloi sovelluksen koodin, jotta sen uudelleenkäyttö helpottuu ja sovelluksen käyttö erikokoisissa laitteissa mahdollistuu. (Android application components overview 2014.)

4.3 Android Manifest -tiedosto

Android Manifest on XML-tiedosto, joka syntyy automaattisesti sovelluksen asennuksen yhteydessä. Android-järjestelmä käy läpi tämän tiedoston määritelläkseen sovelluksen toiminnot. Manifestissa määritellään sovelluksen komponentit ja asetukset sekä metadatatieto, kuten sovelluksen versionumero. (Android application components overview 2014; Deitel ym. 2012, 45–46.) Kuviossa 3 on esimerkki manifest-tiedoston rakenteesta. Kaikki sovelluksen aktiviteetit, palvelut ja sisältöä tuottavat komponentit on määriteltävä pysyvästi (staattisesti) manifest-tiedostossa. Sen sijaan tapahtumakuuntelijat voidaan määritellä vaihtoehtoisesti staattisena manifest-tiedostossa tai dynaamisesti sovelluksen käynnissäoloaikana. (Android application components overview 2014.)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.titta.anatomy"
   android:versionCode="1"
   android:versionName="1.0" >

   <uses-sdk
      android:minSdkVersion="8"
      android:targetSdkVersion="21" />

   <application
      android:allowBackup="true"
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >
      <activity
         android:name=".AnatomyGame"
         android:label="@string/app_name" >
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
   </application>

</manifest>
```

Kuvio 3. Anatomy Game -sovelluksen AndroidManifest.xml -tiedoston rakenne.

Sovelluksen sanallinen, käyttäjälle näkyvillä oleva nimi määritellään manifest-tiedostossa otsikolla ”android:versionName”. Nimen tarkennukseen ja yksilöintiin käytetään numeroita (integer) otsikkolla ”android:versionCode”. Version numerinen arvo aloitetaan tyypillisesti numerosta 1 ja sitä kasvatetaan yhdellä suurempaan numeroon, mikäli sovellus päivitetään. Manifest-tiedoston kohtaan <application> määritellään sovelluksen metadata sekä valinnaisena tietona täsmällinen sovellusluokka. Tähän kohtaan voidaan myös koota Androidin komponenttien selitykset. Kohdassa <activity> määritellään luonnollisesti aktiviteettikomponenetit. Nimiatribuutti ilmaisee luokan, joka on suhteessa pakkausatribuutissa määriteltyyn pakkaukseen. Samalla kaavalla määritellään palvelut, vastaanottajat sekä toimittajat. (Android application components overview 2014.) Esimerkiksi palvelun perusrunko lisätään manifest-tiedostoon rivillä: <service android:enabled="true” android:name= ”.MyService"/>, jossa viimeisten lainausmerkkien väliin tulee sovellukseen lisättävän palvelun nimi. (Meier 2010, 289.)

Manifest-tiedostossa otsikolla @string/app_name viitataan resurssitiedostoihin, joista tällä nimellä löytyy kyseiseen kohtaan tarkoitettu teksti (tässä tapauksessa sovelluksen nimi). Resurssitiedostojen käyttö mahdollistaa erilaisten resurssien, kuten värien, ikonien ja tekstien joustavan käytön sekä käyttäjän että tietokoneen kannalta. (Android application components overview 2014.) Tyyli- ja teemamääritysten kautta ohjelmoijalla on mahdollisuus toteuttaa johdonmukaisennäköinen sovellus, jossa jokainen näyttö ilmaisee kuuluvansa kokonaisuuteen. Yleisin tyylien ja teemojen käyttötapa on määritellä sovelluksen eri osissa käytettävät värit ja tekstityylit manifest-tiedostossa, jolloin koko sovelluksen ulkonäköä voi muuttaa kätevästi ja nopeasti vaihtamalla näiden asetuksia vain tässä yhdessä tiedostossa ja kohdassa. (Meier 2010, 62–63.)

Sovelluksen manifest-tiedostoon kirjataan <uses-feature> -merkinnällä sovelluksen laite- ja ohjelmistovaatimukset (Deitel ym. 2012, 38–40). Projektia perustettaessa on valittava sovellukselle kohde-SDK, johon se suunnitellaan. Tämä on korkein versio, jossa sovellus toimii ja tätä versiota testataan sovelluskehityksen aikana. Osio nimeltä <uses-sdk> tarjoaa mahdollisuuden määritellä kyseiselle sovellukselle minimi- ja kohdeversiot (minSdkVersion, targetSdkVersion). Minimiversiolla tarkoitetaan aikaisinta Androidin versiota (API-taso), jossa kyseinen sovellus toimii. Mitä alemmaksi API-tason vaatimuksen asettaa, sitä useammissa laitteissa sovellus toimii. Toisaalta käytettävissä olevien sovelluksen ominaisuuksien määrä kutistuu. Valitsemalla API 8 -tason, saavut-

taa arviolta 95 prosenttia markkinoilla olevista laitteista. Sovellukset on suositeltavaa kehittää uusimmalla saatavissa olevalla versiolla, jotta kaikki mahdolliset ominaisuudet saadaan hyödynnettyä. (Android application components overview 2014.) Anatomy Game -sovelluksessa minimiversioksi valittiin juuri API 8 -taso ja kohdeversioksi API 21 eli uusin saatavilla oleva Android-versio, kuten kuviosta 3 näkyy.

4.4 Resurssitiedostot

Android-sovelluksissa resurssitiedostot, kuten XML-muotoiset konfiguraatiotiedostot ja kuvat tallennetaan erilleen kooditiedostoista. Resurssitiedostojen käyttäminen helpottaa myös sovelluksen eri versioiden muokkaamista. Nämä tiedostot sijoitetaan /res -hakemistossa valmiiksi luotuihin alikansioihin. Erityyppisille resurssitiedostoille luodaan omat alakansionsa. Taulukossa 4 esitellään tyypillisimmät Androidin tukemat resurssit: values, drawables, layout, raw ja menu. Kullakin on oma tehtävänsä ja mahdollisesti lukuisiakin eri xml-tiedostoja. Esimerkiksi values-kansiossa määritellään merkkijonot, värit, asemointi sekä merkkijono- ja numerotaulukot omissa xml-tiedostoissaan. (Android application components overview 2014; Meier 2010, 60–61.)

Asemoinnin määrittämisellä tarkoitetaan näytön eri komponenttien sijoittamista suhteessa näytön reunoihin sekä toisiin komponentteihin. Asemointi voidaan ilmaista kuudella eri mitta-asteikolla: näytön pikseleinä, näytön tarkkuudesta tai koosta riippumattomina pikseleinä tai vaihtoehtoisesti fyysisinä pisteinä, tuumina tai millimetreinä. Asemoinnin määrittämistapaa valitessa ohjelmoijan on valittava, minkäasteisesti mukautuvan sovelluksen hän haluaa toteuttaa. (Meier 2010, 60–62.)

Taulukko 4. Resurssitiedostojen jakautuminen (Android application components overview 2014).

Kansio	Määriteltävät sovelluksen ominaisuudet
/res/values	Teemojen ja tyylien käyttö (ulkoasu), mittasuhteet,
/res/values/colors.xml	värien käyttö,
/res/values/strings.xml	tekstikenttien sisältö,
/res/values/arrays.xml	stabiilien teksti- ja numerotaulukkojen (string, integer) sisältö
etc.	
/res/drawables	Kuvat ja kuviot
/res/layout	Aktiviteetien käyttäjäpinnan muotoilu, fragmentit
/res/raw	Käyttäjäsyötteen sisältö raakamuodossa
/res/menu	Valikkojen asetukset

Sovelluksen tyylit voidaan määritellä res/values -hakemistossa <resources> -juuressa, missä kirjoitetaan erikseen <style> -merkintä, jonka sisään voidaan määritellä tekstin koko ja väri sekä asemointitietoja. Yhdessä sovelluksessa voi olla käytössä useampia tyylejä ja ne määritellään tässä resurssitiedostossa erillisinä peräkkäin. (Android Styles and Themes 2015.)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="title_color">#00314f</color> <!-- sininen -->
    <color name="question_color">#2b0806</color> <!-- tummanpunainen -->
    <color name="text_color">#000000</color> <!-- musta -->
    <color name="background_color">#b79aa5</color> <!-- vaaleanroosa -->
    <color name="correct_answer">#004419</color> <!-- vihrea -->
    <color name="incorrect_answer">#FF0000</color> <!-- punainen -->
</resources>
```

Kuvio 4. Anatomy Game -sovelluksen colors.xml-tiedosto.

Sovelluksessa kussakin kohdassa käytettävät värit määritellään resurssitiedostossa <color> -merkinnällä, jossa itse värisävyn määrittämiseen käytetään RGB-värijärjestelmää (Meier 2010, 60–62). Kyseinen järjestelmä perustuu kaikkien värisävy-

jen muodostamiseen punaisen (Red), vihreän (Green) ja sinisen (Blue) erilaisilla sekoitussuhteilla, jotka määritellään käyttäen heksadesimaalinumerointia (Color-Hex 2015). Esimerkiksi kuviossa 4 lausekkeella <color name="title_color">#00314f</color> määritellään otsikkokohtien tekstin väriksi sininen. Anatomy Game -sovelluksessa määritettiin sovelluksen käyttäjäliittymän eri kohdissa esitettävät tekstit erivärisiksi; oikea vastaus ilmaistaan vihreällä ja väärän vastauksen ilmetessä kehotetaan käyttäjää yrittämään uudelleen vastaamista kirkkaanpunaisella tekstillä. Näin eri tilanteiden viestit erottuvat selvemmin toisistaan ja sovelluksen ilmeikkyys lisääntyy. Käyttäjäliittymän tekstien kokoa säädellään samassa res/values -hakemistossa sijaitsevassa dimens.xml-tiedostossa. Anatomy Game -sovelluksessa määriteltiin otsikolle, kysymykselle, vastaukselle sekä muulle tekstille omat fonttikokonsa.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">Anatomy Game</string>
    <string name="quiz_title">Anatomian tietopeli</string>
    <string name="choices">Valitse vastausvaihtoehtojen määrä</string>
    <string name="parts">Valitse kehonosat</string>
    <string name="question">Kysymys</string>
    <string name="incorrect">Yritä uudelleen!</string>
    <string name="choose_name">Valitse oikea käännös</string>
    <string name="your_result">Tuloksesi: </string>
    <string name="guesses">vastausta, joista </string>
    <string name="correct">oikein!</string>
    <string name="reset_quiz">Pelaa uudelleen!</string>

    <string-array name="partsList">
        <item>sisäelimet</item>
        <item>luut</item>
        <item>lihakset</item>
    </string-array>
    <string-array name="guessesList">
        <item>3</item>
        <item>6</item>
    </string-array>

</resources>
```

Kuvio 5. Anatomy Game -sovelluksen strings.xml-tiedosto.

Resurssikansion values-alakansiossa sijaitsevassa strings.xml-tiedostossa määritellään sovelluksen käyttöliittymässä erikohdissa näytettävät tekstit merkkijonoina. Merkkijonojen määritteleminen varsinaisen koodin ulkopuolella, omassa resurssitiedostossa auttaa pitämään nimeämiskäytännöt johdonmukaisessa linjassa. (Meier 2010, 60–62.) Anatomy Game -sovelluksen strings.xml-tiedoston (kuvio 5) ensimmäisenä toimenpiteenä annetaan merkkijono AndroidManifest.xml -tiedostossa määritellylle, strings.xml-tiedostosta haettavalle kohteelle “app_name” eli sovelluksen nimi. Käsillä olevan sovelluksen nimeksi tuli siis Anatomy Game. Toisena merkkijonomäärittelynä asetetaan sovelluksen aikaansaama käyttäjäliittymässä näkyvä kysely toimimaan otsikolla Anatomian tietopeli (kohta <string name=”quiz_title”>). Tämän määrityksen seurauksena sovellus tuottaa käyttöliittymään näkymään ”Anatomian tietopeli” -tekstin joka kerta, kun sovelluksen koodissa mainitaan ”quiz_title”. Jos siis myöhemmin halutaan muuttaa sovelluksen nimeä, joka mahdollisesti esiintyy useammassa kohdassa käyttöliittymän näytöillä, onnistuu muuttaminen kätevästi vaihtamalla nimi vain strings.xml-tiedoston yhteen kohtaan. Tämä varmistaa, että nimi esiintyy yhtenäisenä eri kohdissa. Kuviossa 5 määritellään tekstit varsinaisen tietopelinäytön lisäksi sovelluksessa käytettäviin valikoihin (kohdat <string-array name=”partsList”> ja <string-array name=”quessesList”>), joilla sovelluksen käyttäjä voi vaikuttaa tietopelin asetuksiin.

```
<TextView
  android:id="@+id/title"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:layout_marginBottom="10dp"
  android:gravity="center"
  android:text="@string/quiz_title"
  android:textSize="@dimen/title_size"
  android:textColor="@color/title_color"
  android:typeface="serif" />
```

Kuvio 6. Tekstikentän määritteleminen, koodiesimerkki Anatomy Game -sovelluksesta.

Sovelluksen käyttöliittymään tarvittavat tekstikentät edellä mainittujen merkkijonojen esiin saattamiseksi määritellään res/layout-hakemistossa sijaitsevassa xml-tiedostossa,

jossa määritellään kentät myös muun muassa esiin haluttaville nappuloille. Kuviossa 6 esitellään esimerkki tekstikentän määrittämiseen vaadittavasta koodista. Kyseinen tekstikenttäkoodi sisältyy Anatomy Game -sovelluksen Activity_anatomy_game.xml-tiedostoon.

4.5 Sovelluksen versionhallinta

Versionhallinnan yläkäsite on konfiguraationhallinta, jolla tarkoitetaan kehittyvien järjestelmien organisoinnista ja hallinnasta huolehtimista. Ohjelmistoprojektin tiedostoista muodostetaan konfiguraatio-objekteja ja sitä kautta konfiguraatioita. Konfiguraatio-objektilla tarkoitetaan esimerkiksi asiakirjaa, ohjelmistokomponenttia tai näiden yhdistelmää. Konfiguraationhallinnalla määritellään menettelytapa ohjelmatiedostojen ja -dokumenttien sekä niiden yhdistelmien tunnistamiseksi, muutosten hallitsemiseksi sekä tuotteen yhtenäisyyden varmistamiseksi. Ohjelmistojen konfiguraationhallinta on laajasti automatisoitavissa. (Ohjelmiston versionhallinta 2014.)

Versionhallinta on tiedostojen tai tiedostojoukkojen muutoksia tallentava järjestelmä. Ohjelmiston versionhallinnalla tarkoitetaan ohjelmistoprojektin tuotosten hallintaa, mikä mahdollistaa ohjelmiston kehityksen seuraamisen sekä ohjelmiston hallitun kehittämisen. Sen tärkeimpiä tehtäviä on mahdollistaa ristiriidaton, yhtäaikainen ja turvallinen ohjelmistokomponenttien kehittäminen. Se mahdollistaa myös ohjelmiston kehittämisen eri suuntiin, esimerkiksi eri alustoille. Versionhallintajärjestelmää käyttämällä sovelluskehittäjän on mahdollista taltioida kaikki kehityksen vaiheet ja palata niihin tarvittaessa helposti. Versionhallinta mahdollistaa vaikka koko sovelluksen palauttamisen johonkin aikaisempaan muotoon. Myös ongelmien ratkominen helpottuu, kun kehittäjä voi tarkastella, mitä on muutettu sen jälkeen kun sovellus viimeksi toimi. Versionhallinta estää tilanteen, jossa käyttäjä vahingossa tallentaa tietoja toistensa päälle tai vääriin tiedostoihin tuhoten jotain tärkeää. Versionhallinnan ominaisuudet ovat parhaiten käytössä isoissa projekteissa, joissa on monia toteuttajia ja säilö sijaitsee keskitetysti jollain palvelimella. Yksittäisenkin ohjelmoijan tosin kannattaa hyödyntää versionhallinnan tarjoamat ominaisuudet työskennellessään. (A Visual Guide To Version Control 2014; GIT 2014; Ohjelmiston versionhallinta 2014.)

Versionhallinta toimenpiteenä voidaan jakaa neljään osaan: versiointi, versioiden merkitseminen, versioiden välisten erojen tunnistaminen ja versioiden tallentaminen. Versi-

ointi edelleen voidaan jakaa historialliseen, yhteistoiminnalliseen ja loogiseen versiointiin. Historiallisessa versioinnissa uudempi versio syrjäyttää vanhemman, mitä kutsutaan revisioksi. Loogisen versioinnin ideana on kehityshaarojen luominen. Eri haarat kuvastavat ohjelmiston kehityksen vaihtoehtoisia kulkuja. Haarat etenevät itsenäisesti muista haaroista välittämättä omaan suuntaansa. Loogisen versioinnin tuloksena ohjelmistoa voidaan kehittää rinnakkain. Yhteistoiminnallisen versioinnin tarkoituksena on luoda hetkellisiä versioita komponenteista. Eri komponenttien versiot tullaan myöhemmässä vaiheessa integroimaan toisiinsa. Tätä voisi kutsua hetkelliseksi poikkeamaksi, joka yleensä syntyy kehitysprosessin tarpeesta ja jonka tiedetään jo luotaessa olevan väliaikainen. (Ohjelmiston versionhallinta 2014.)

Versioiden merkitsemisellä tarkoitetaan jokaisen konfiguraatio-objektin yksikäsitteistä nimeämistä. Versiot järjestetään yleensä juoksevalla kasvavalla numeroinnilla. Numerointi on usein kaksitasoinen. Numeroinnin ensimmäinen numero kuvaa julkaisunumeroa (versionumero) ja toinen tasonumeroa (revisio). Julkaisunumeron muutos kertoo suuresta muutoksesta ja pienemmät päivitykset merkitään tasonumeroilla. Lisäksi versioinnissa voidaan käyttää kolmatta numeroa merkitsemään ohjelmaan tehtyä paikkaa. Tämä merkitseminen auttaa ohjelmistovirheiden korjaamisprosessissa. (Ohjelmiston versionhallinta 2014.)

Tallennetuista versioista voidaan ohjelmistokehityksen aikana valita tarpeen mukaan millä tahansa hetkellä tallennettu ohjelmiston versio tarkasteltavaksi. Versiot ovat yhteenvetoja tai kuvauksia ohjelmiston senhetkisestä tilasta eli ohjelmiston muunnelmia. Peräkkäiset versiot ovat yleensä hyvin samanlaisia. Eroavaisuuksien tarkan määrittelemisen seurauksena huomio voidaan keskittää muutokseen ja säästää näin resursseissa, kuten tallennukseen vaaditussa tilassa. Ohjelmistotiedostojen välinen merkkivertailu on yleisin erojen tunnistamistapa. Menetelmän tavoitteena on löytää mahdollisimman pieni määrä eroja ja tallentaa vain nämä eroavat kohdat version A muuttamiseksi versioksi B. Menetelmä ei toimi binääritiedostoissa, joissa yhden merkin eroavaisuudella voi olla laajakin vaikutus tiedoston koko sisällön kannalta. (Ohjelmiston versionhallinta 2014.)

Ensimmäiset versionhallintaratkaisut olivat paikallisia. Tietokoneelle luodaan tällöin yksinkertainen tietokanta tiedostojen muutosten tallentamista varten. Mikäli sovelluskehittäjä haluaa työstää sovellustaan kollegojensa kanssa, on keskitetty versionhallintajärjestelmä paikallista käyttökelpoisempi ratkaisu. Näissä järjestelmissä yksittäinen

palvelin tallentaa kaikki versioidut tiedostot ja käyttäjät hakevat tiedostot sieltä. Järjestelmän etuna on, että jokaisella käyttäjällä on käsitys siitä, mitä kukin projektissa tekee. Järjestelmänvalvoja voi hallinnoida käyttäjien oikeuksia tiedostoihin. Järjestelmän haittapuolina mainittakoon versiointiin käytettävän palvelimen alasajon aiheuttama useampien käyttäjien työskentelyn keskeytyminen sekä palvelimen korruptoitumisesta seuraava suuri tietokato ellei käytössä ole kattavaa varmuuskopiointia. (GIT 2014.)

Hajautetussa versionhallintajärjestelmässä on ratkottu keskitetyn järjestelmän heikkouksia. Hajautetussa mallissa jokaisella käyttäjällä on oma paikallinen säilö (repository). Eri henkilöiden säilöissä oleva materiaali yhdistetään valituin perustein tai valittujen henkilöiden päätöksillä. (Subversion ja versionhallinta 2014.) Tässä järjestelmässä käyttäjät peilaavat palvelimelta tiedoston viimeisimmän tilannekuvan hakemisen sijasta koko tietolähteen. Näin palvelimen poistuessa syystä tai toisesta käytöstä, voidaan sen sisältö korvata käyttäjältä saadulla peilauksella. Jokainen tiedonhaku siis toimii todellisuudessa täytenä varmuuskopiona kaikesta lähteessä olevasta tiedosta. Lisäksi hajautetussa versionhallinnassa on mahdollisuus työskennellä monien etätietolähteiden kanssa samanaikaisesti, mikä ei ole mahdollista keskitetyssä järjestelmässä. Esimerkiksi Git, Mercurial, Bazaar ja Darcs toimivat hajautetun versionhallinnan järjestelmän periaattein. (GIT 2014.)

5 Tietokone oppimisympäristönä

Tieto ja tietämys lisääntyvät kiihtyvällä vauhdilla nykymaailmassamme. On entistä tärkeämpää oppia erottamaan olennainen ja luotettava tieto kaikesta saatavilla olevasta tietomäärästä. On myös olennaista oppia jäsentelemään ja yhdistelemään tietoa tehokkaasti pystyäkseen hyödyntämään sitä. Ihmisen on ymmärrettävä osaamisensa kulloinenkin tila pystyäkseen täydentämään sitä (Bransford ym. 2004, 152).

5.1 Ihmisen oppimisen pääpiirteet, vaatimukset ja vaiheet

Oppimismotivaation syntyminen on edellytys oppimiselle. Opiskelijan oppimismotivaatio määrää, paljonko hän on halukas käyttämään aikaa oppiakseen määrätyn asian. Motivaatio voi olla sisä- tai ulkosyntyinen. Sisäsyntyisen oppimismotivaation omaava opiskelija opiskelee oppiakseen, lisätäkseen pätevyyttään ilman odotuksia ulkoisista palkinnoista tai rangaistuksista. Viimeksi mainittu ulkoisen palautteen odottaminen ku-

vaa ulkosyntyistä motivaatiota. Motivaation säilymisen kannalta on merkityksellistä, kuinka haasteellisena opiskelija pitää oppimistehtävää. Liian suuri haaste lannistaa ja toisaalta liian vähän haastetta sisältävä tehtävä turhauttaa herkästi. Tehtävän ääressä vietetty aika on yksi oppimisen mittari, mutta myös oppimisen tehokkuutta on tarkasteltava. Ymmärtämiseen tähtäävällä oppimisella on selvästi paras kontekstista toiseen siirtymisen mahdollisuus. Opitun siirtymisen laajuuteen on todettu vaikuttavan kontekstien samankaltaisuus sekä oppimistapahtuman eri osatekijät, kuten se, miten paljon opiskelija ymmärtää oppimaansa sen sijaan että vain opettelisi sen ulkoa tai noudattaisi ennalta määrättyjä menettelytapoja. Saman asian opiskelu useammassa eri kontekstissa lisää myös todennäköisyyttä sille, että opiskelija ymmärtää käsitteiden merkitykselliset piirteet ja pystyy yhdistämään ne yhä uusiin tilanteisiin oppimaansa hyödyntäen. (Bransford ym. 2004, 28, 69, 74–75.)

Tiedon jäsentäminen vaatii aiempien tietojen, käsitysten ja osaamisen tiedostamista sekä niiden ottamista uuden oppimisen perustaksi. Opetuksen metakognitiivinen lähestymistapa tukee opiskelijaa hallitsemaan itse omaa oppimistaan määrittelemällä itselleen oppimistavoitteita ja seuraamalla edistymistään niitä kohti. Opiskelijalle pitäisikin tarjota mahdollisuus testata osaamisen ja ymmärtämisen tasoaan useasti ja mielellään kätevällä tavalla oppimisen ohessa. Tietokoneavusteinen oppiminen, etenkin pelimuotoiset materiaalit, ovat omiaan tukemaan opiskelijan omatoimista oppimista. Ne mahdollistavat opiskelijan omat kokeilut ja itselle sopivassa tahdissa etenemisen tai asioihin uudelleen palaamisen. Opiskelijan jäsentyneiden tietojen ja taitojen saavuttamista voidaan tukea tarjoamalla hänelle mielekkäitä ongelmanratkaisutehtäviä ja samanaikaisesti auttamalla ymmärtämään kyseisten tietojen ja taitojen tarpeellisuuden kontekste-ja. Opiskelijaa voidaan opettaa tarkastelemaan ja seuraamaan aktiivisesti oppimisstrategioitaan ja resurssejaan sekä arvioimaan osaamisen tasoaan. (Bransford ym. 2004, 28, 31, 36–37, 66, 71–72, 81, 83, 93–94, 162.) Opettajien aika palautteen antamiseen kullekin opiskelijalle on rajallinen, joten tietokoneavusteisuus on hyvä lisä opettajan resursseihin.

5.2 Oppimisen edellytykset

Opetussuunnitelmissa on jo vuosikymmeniä pyritty lisäämään oppiaineksen integraatiota eli sulauttamista yhtenäisemmäksi kokonaisuudeksi. Tavoite on edelleen ajankohtainen. Oppiaineiden tasolla integraatio on useimmiten toteutettu olemassa olevien

oppiaineiden opetusresursseja yhdistämällä ja aihepiirejä samanaikaistamalla. Opiskelijan tasolla integraation tavoitteena on asiakokonaisuuksien laaja-alaisempi hallinta. Integroidun opetussuunnitelman noudattamisen on todettu lisäävän opiskelijoiden oppimismotivaatiota ja erityisesti käsitteiden syvällisempää ja laaja-alaisempaa hallintaa. (Kari 1994, 95, 97.) Tutkimusten mukaan opetuksessa tapahtuva tietotekniikan hyödyntäminen tukee juuri opiskelijan itsenäistä tiedonrakentelua auttamalla opiskelijan ajatteluprosessien muodostumista näkyviksi. Tietotekniset opetusympäristöt tarjoavat opiskelijalle entistä laajempia mahdollisuuksia tiedon tuottamiseen, etsimiseen, esittämiseen ja edistävät keskustelua ja pohdintaa. (Järvelä ym. 2011.)

Koulujen ja oppilaitosten toiminnan painopiste on viime vuosikymmeninä siirtynyt yhä vahvemmin opettamisesta opiskelijoiden oppimisen tukemiseen. Uudistuvalle toimintatavalle on etsitty toteutusmenetelmiä, jotka mahdollistavat vastuun jakamisen vahvemmin myös opiskelijalle. Opiskelijan valmiudet tiedon omaksumiseen, ymmärtämiseen ja käyttämiseen ovat nousseet opiskelun keskiöön. Tämä asettaa opetusmenetelmien ja -järjestelyiden kehittämiselle uusia haasteita. Opetusmateriaalien ja -välineiden valinnan perusteena tulisi ensisijaisesti olla oppimiselle asetettujen tavoitteiden saavuttamisen mahdollistaminen. (Kari 1994, 103, 116, 174.) Teknologiset ratkaisut voivat tarjota uusia mahdollisuuksia edellä mainittujen seikkojen huomioimiseen (Järvelä ym. 2011).

5.3 Opetusteknologisten ratkaisujen suunnittelu

Ihmisen toiminta alkaa aina omien tarpeiden tiedostamisesta, jota seuraa suunnitelma niiden tyydyttämiseksi. Teknologiaa suunniteltaessa onkin olennaista aloittaa ihmisen tarpeiden kartoittamisesta. Teknologiaa voidaan kehittää joko ihmis- tai teknologiakeskeisesti. Ihmisen ja teknologian menestyksellinen vuorovaikutussuhde sen sijaan on monimuotoinen kokonaisuus. On otettava huomioon kaikki ihmisen oppimisesta sosiaaliseen vuorovaikutukseen ja koneiden kapasiteettitekijöihin asti. Teknologian suunnittelun lähtökohdan tulisi olla ihmistieteisiin pohjaava, jolloin on mahdollista saada kokonaisvaltainen käsitys ihmisen toiminnasta ja tavoitteista. Katsantokannan olisi hyvä olla monitieteinen. (Saariluoma ym. 2010, 22–24, 67.)

Ihmisen raajojen ja aistien yhteistoiminta on olennaista sekä fyysisen ergonomian näkökulmasta että teknologisten ratkaisujen suunnittelun pohjana. Fyysisen ergonomian

rinnalle voidaan tekniikan digitalisoitumisen seurauksena nostaa kognitiivinen ergonomia. Sillä tarkoitetaan ihmisen ja teknologian vuorovaikutuksen tarkastelemista ihmisen kognitiivisten toimintojen pohjalta. Kognitiivisia toimintoja ovat havainnointi, tarkkaavaisuus, muisti, kielelliset prosessit ja ajattelu. Ihmisen rajallisen suorituskapasiteetin, kuten tarkkaavaisuuden, muistin ja oppimiskyvyn rajojen sekä aistien erotuskynnyksen, huomioimatta jättäminen teknologian suunnittelussa johtaa teknologian väärinkäyttämistodennäköisyyden kasvamiseen. (Saariluoma ym. 2010, 25, 63, 66.)

Teknologian vuorovaikutustutkimuksella tarkoitetaan ihmisen ja teknologian vuorovaikutuksen tarkastelemista inhimillisen tietämisen alueella. Aiheesta voi käyttää myös termiä käytettävyys, joka tosin voidaan käsittää huomattavasti suppeammin keskittyen vain käyttäjien tekniikan käyttämisen osaamiseen. Käytettävyys on kustannustekijä. Huonon käytettävyyden seurauksia ovat sovelluksen käytön oppimisen ja käytön hitaus, käyttämismotivaation, työtyytyväisyyden ja sitoutumisen laskeminen, suoritusvirheet, menetetyt asiakkaat, luottamus ja maine. Edellä mainittujen lisäksi kustannuksia nostavat suuremmat ylläpito-, tukipalvelu- ja edelleenkehittämiskustannukset. Koska ihmisen oppiminen perustuu aina aiemmin opitulle ja opitun siirtämiseen uusiin konteksteihin, uusien toimintojen oppiminen on sitä helpompaa, mitä enemmän ne muistuttavat aiemmin opittuja toimintoja. Tässä on käyttöliittymäratkaisujen suunnittelun olennainen aspekti. Järjestelmien välisten yhteisten toimintoketjujen ja muiden osaratkaisujen käyttö kannattaa maksimoida. (Saariluoma ym. 2010, 15–16, 20, 62.)

Teknologian käytöstä ja käyttökokemuksista syntyy käyttäjäkokemuksena kuvattava tunnetila, joka otettiin teknologiansuunnittelun osatekijäksi vastareaktiona mekanistiselle ajattelutavalle käyttäjästä. Mekanistisessa ajattelutavassa käyttäjä käsitetään teknologian toiminnan mahdollistajana eikä nykyiseen tapaan teknologian hyödyntäjänä. Käyttäjäkokemuksen syntymiseen vaikuttaa tuotteen muotoilun lisäksi käyttäjän tavoitteet (mitä hän haluaa tuotteella tehdä), aiempi tieto tuotteesta sekä siihen kohdistuvat odotukset ja tuotteen käyttämisen onnistuminen. Toistaiseksi useimmat kuluttajat vaativat teknologian käytettävyydeltä vielä varsin vähän, mutta odotettavissa on käytettävyyskriittisyyden kasvua ihmisten tietoisuuden ja kokemusten kasvaessa. Tällöin käytettävyydelle on annettava teknologian suunnittelussa nykyistä enemmän painoarvoa. (Saariluoma ym. 2010, 29, 40–42.)

Teknologian käyttämiselle on aina syynsä. Yleensä syynä on ihmisen toimintapäämäärien aiempaa parempi saavuttaminen. Kriteerinä voi olla esimerkiksi aika, edullisuus,

turvallisuus, luotettavuus tai helppous. Ilman näiden saavuttamista ihmisen motivaatio teknologian käyttämiseen on minimaalinen. Teknologian merkitys syntyy yksilön henkilökohtaisten symbolisten arvojen ja sosiaalisten suhteiden kautta. Merkitykset voivat painottua sisäisille tai ulkoisille tavoitteille. Teknologian sisäisesti koettu merkitys tarkoittaa teknologian itseisarvoa eli esimerkiksi pelisovellus sinänsä tuottaa käyttäjälleen tyydytystä ilman minkäänlaista hyödyn hakemista tai saavuttamista. Vastakohtana on luonnollisesti hyötyä tavoitteleva ulkoisesti koettu teknologian merkitys. Suunnittelijoiden on ymmärrettävä, mihin ihmiset tarvitsevat kyseistä palvelua ja millaisia nämä käyttäjät ovat. Laitteiden sopeuttaminen on huomattavasti helpompaa kuin ihmisten sopeuttaminen. (Saariluoma ym. 2010, 29, 31, 40–41, 44–46, 60.)

5.4 Oppimisohjelmien kehitys

Tietokoneen käyttäminen oppimistulosten kohentamiseen sai alkunsa 1960-luvulla (Bransford ym. 2004, 229). Oppimispelejä on ollut olemassa 1950-luvulta lähtien, mutta vasta 1990-luvulla ne tulivat säännöllisemmin käyttöön kouluopetuksessa. Kouluissa on samanaikaisesti vahvistunut pelaamalla oppimisen (learning by playing) näkökulma. (Saarenpää 2009.) Uuden tekniikan avulla on entistä helpompi tukea opiskelijan tekemällä oppimista (learning by doing). Nykytekniikka luo aivan uusia oppimisympäristö- ja havainnollistamismahdollisuuksia, joita voidaan hyödyntää sellaisinaan tai täydentämässä perinteisiä oppimisympäristöjä ja -materiaaleja. Oppimispeleissä voidaan opiskelijan motivaation ylläpitämiseksi käyttää pelillisiä ominaisuuksia, kuten tehtävien haastetason muokkaamista. Viihteellisten ja oppimispelien erot ovat viime vuosina kaventuneet. Pelialan ja pedagogiikan asiantuntijat ovat yhdistämässä voimiaan. (Saarenpää 2009.) Adekvaatisti ja asiallisesti käytettynä tekniikasta voi tutkimusten mukaan olla ratkaisevaa hyötyä opiskelijalle. Muun muassa peleillä voidaan tukea oppimista erittäin tehokkaasti. Pelatessaan opiskelija prosessoi aktiivisesti oppimaansa, mikä kiihdyttää oppimista. Pelissä opittava asia on sen omaksumisen ja ymmärtämisen kannalta merkityksellisessä kontekstissa. Asia, jota itse pääsee kokeilemaan, vaikkakin virtuaalisesti, tulee opittua syvemmin kuin pelkkä teoreettinen tiedoksisaanti. (Saarenpää 2009.) Teknisten ympäristöjen interaktiivisuus on hyvin tärkeää oppimisen kannalta. Opiskelijan on helppo palata tutkimaan joitain osia tarkemmin, testaamaan ymmärtämistään sekä keräämään palautetta. Ympäristöjä voi tutkia myös eri työryhmien kesken ja laajentaa osaamistaan sitä kautta. (Bransford ym. 2004, 229–232, 256.)

Pelit ovat vakiintuneet osaksi eri-ikäisten, etenkin nuorten arkea. Pelaaminen on toimintaa, jonka parissa usein viihdytään pitkiäkin aikoja. Pelit motivoivat tarjoamalla mielekkäitä haasteita kiinnostavassa ympäristössä sekä antamalla palautetta toiminnasta. Pelien positiivisia ominaisuuksia voidaan hyödyntää myös opetuksessa. Varsinkaan nuoret eivät aina ymmärrä opiskeltavien asioiden mielekkyyttä ja tarpeellisuutta omassa elämässään, mikä vaikeuttaa asian oppimiseen motivoitumista. Oppimispeleissä pelaaminen ei ole pelkästään viihdetarkoituksellista vaan sillä tavoitellaan myös välitöntä hyötyä. Oppimispelillä tarkoitetaan peliä, jonka pelaamisen tarkoituksena on oppia jokin uusi taito tai tietoa tai harjaannuttaa jo opittuja taitoja ja tietoja. Oppimispeli voidaan liittää osaksi koulussa tapahtuvaa opetusta, se voi olla opitun kertaamisen väline ja sitä voidaan käyttää myös kotona tukemaan kouluopetusta. Pelien oppimista lisäävä vaikutus vaatii pelin huolellisen laatimisen ja sen käytön tarkoituksenmukaisuuden, optimaalisuuden sekä oikea-aikaisuuden. Parhaimmillaan oppimispelit tarjoavat miellyttävän ja motivoivan tavan oppia erilaisia tietoja ja taitoja saaden opiskelijan viihtymään perinteisiä oppimisympäristöjä pidemmän aikaa ja intensiivisemmin opittavan asian parissa. Pelien tarjoama potentiaali oppimisen tukena on osattava hyödyntää oikein. (Saarenpää 2009.)

Ensimmäiset oppimispelit keskittyivät mahdollistamaan jo opitun taidon tai tiedon harjaannuttamista. Tällöin samaa asiaa toistetaan lukuisia kertoja sen vakiinnuttamiseksi. Harjaannuttamispelisovellus on tyypillisesti varsin pieni ja kapea-alainen. Sen sijaan simulaattoripelit tarjoavat huomattavasti laajemman oppimisympäristövalikon, mikä lisää pelin ja oppimisen kiinnostavuutta merkittävästi. Simulaattori mallintaa todellisia tilanteita tietokoneen avulla, jolloin opiskelijalla on mahdollisuus kokea laajempi kirjo tilanteita (esim. lentosimulaattori) tai toistaa tilannetta (esim. injektionantamista) lukuisia kertoja toisin kuin reaalimaailmassa. Simulaattori mahdollistaa myös asian tarkastelun useammilta eri näkökannoilta sekä hypoteesien luomisen ja kokeilemisen. (Saarenpää 2009.)

5.5 Androidin soveltuvuus oppimisympäristön luomiseen

Mobiilien viestintälaitteiden, kuten Android-puhelimen, käyttäjinä nykynuoret ovat varsin tottuneita. He käyttävät monenlaisia teknologisia työvälineitä ja sovelluksia vapaa-ajallaan. Älypuhelimen monipuoliset, muokattavat ja täydennettävät toiminnot mahdollistavat sen mukaan ottamisen hyvin moniin erilaisiin toimintoihin. Palveluntarjoajat ovat

huomanneet älypuhelinten laajan levinneisyyden ja monikäyttöisyyden luoden ihmisille kasvavan tarpeen omistaa älypuhelin. Nykyihminen alkaakin olla sidoksissa älypuhelimen käyttämiseen selviytyäkseen täysipainoisesti arjessaan (esimerkiksi sähköinen tiedotus, palveluiden saatavuus ja ajanvaraus vaativat mobiilia internetyhteyttä). Älypuhelimen sovellusten käyttämiseen sisältyy sekä hyödyntavoittelua että ajankulua. Kyseisiä välineitä ja niitä kohtaan tunnettua vetovoimaa hyödyntämällä opetusta ja oppimista voidaan tuoda lähemmäksi nuorille nykyisin luontaista toimintatapaa. (Kankaanranta & Vahtivuori-Hänninen 2011.)

Useat yksityiset oppilaitokset sekä laajenevissa määrin kunnatkin ovat luoneet sponsorisopimuksin tai omalla rahoituksellaan opiskelijoille teknologian käyttömahdollisuuden opetuksen osaksi (Huhta ym. 2011). Suomen Kuvalehti (37/2014) selvitti syksyllä 2014 älypuhelinten käyttöä Helsingin ja Rovaniemen peruskouluissa ja lukioissa kyselytutkimuksella. Rehtorit totesivat älypuhelimen pedagogisten hyötyjen jäävän toistaiseksi pieneksi. He mainitsivat rajaavina tekijöinä opettajien valmiudet liittää opetukseensa uusia opetusmuotoja. Älypuhelimista ei juurikaan käytetty opetusohjelmia vaan puhelimen perusominaisuuksia, kuten muistiota ja kameraa.

Älypuhelin, laajimmin levinneenä Android-pohjainen laitteisto, onkin mitä mainioin opetusohjelman alusta. Se on jo valmiiksi suurimmalla osalla potentiaalisista käyttäjistä tai se on suhteellisen edullinen tarjota heille käyttöön opetusohjelman käyttämiseksi. Laite on helpostikäsiteltävänkokoinen, sen perusominaisuuksien käyttö on tuttua suurimmalle osalle kohderyhmästä ja laitteiston ominaisuudet mahdollistavat sen opetuskäytön.

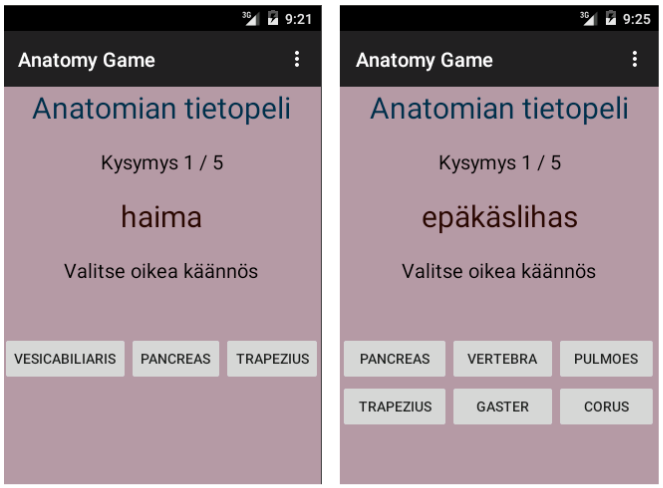
6 Anatomy Game -sovelluksen toteutus

Sovelluksen aihe ja toteutus nousivat tekijän omien kokemusten ja osaamisen pohjalta. Olen aiemmalta koulutukseltaan sekä terveydenhoitaja että opettaja. Aiemmissa työtehtävissä tuleville hoitajille anatomiaa opettaessani huomasin opiskelijoiden kokevan latinankielisten nimien opiskelun vaikeaksi ja osin turhauttavaksikin. Havaitsin myös puhelimen jatkuvasti kilpailevan opettajan huomiosta tuntien aikana. Nyt toteutettavan sovelluksen avulla opiskelijat voisivat hyödyntää oppimisessaan puhelimen käyttämishalukkuuttaan samalla tehden oppimisesta asteen verran miellyttävämpää, ehkä kiehtovampaakin, itselleen.

6.1 Sovelluksen tavoitteet

Tavoitteena on kehittää helppokäyttöinen Android-sovellus, jolla käyttäjä voi testata tasoaan ihmiselimistön osien latinankielisten nimien hallinnassa. Sovelluksessa käyttäjällä on mahdollisuus valita haluaako hän testata osaamistaan sisäelinten, lihasten vai luiden latinankielisten nimien osalta vai ottaako hän haasteekseen testata osaamistaan useamman kehonosan nimistön tai kaikkien mainittujen kehonosien nimistöjen parissa samassa testissä. Käyttäjän on myös mahdollista valita haluaako hän testiin kolme vai kuusi vastausvaihtoehtoa.

Sovelluksen tavoitteena on olla mahdollisimman kevytrakenteinen, jotta se toimisi vaivatta ja käyttäjäkokemus kannustaisi palaamaan sen pariin. Sovelluksen tulisi olla mahdollisimman vähällä laitteen muistikapasiteetilla käytettävissä, mikä kasvattaa sen potentiaalista käyttäjäryhmää. Sovelluksen käytön tavoitteena on olla mahdollisimman yksinkertaista, jotta sen käyttäminen on vaivatonta. Sovelluksen valittiin myös olevan äänetön käyttäjän yksityisyyden lisäämiseksi sekä käyttöympäristöjen maksimoimiseksi.

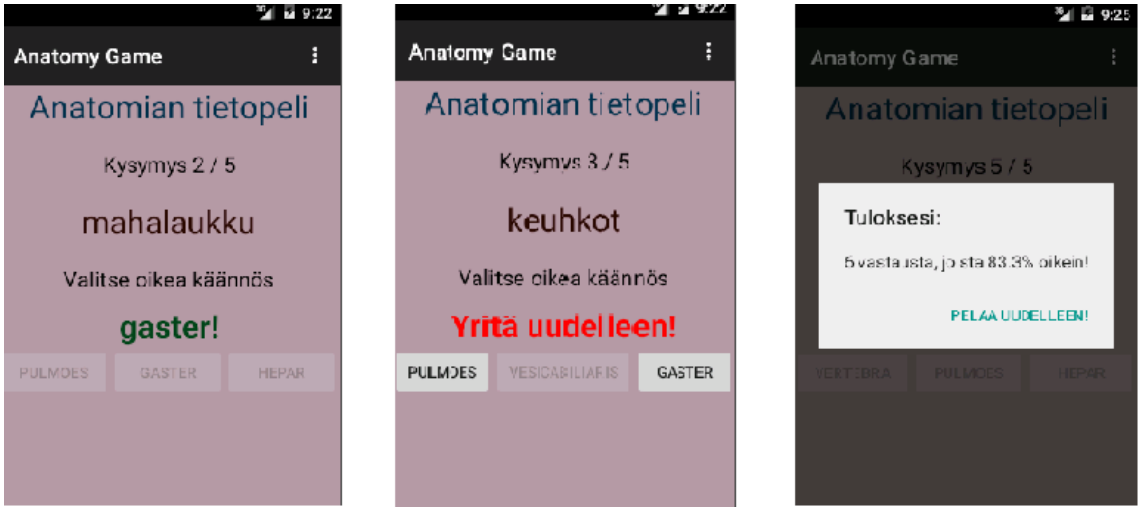


Kuvio 7. Anatomian tietopelin kaksi vaihtoehtoista aloitusnäkymää.

Sovelluksen kohde-SDK:ksi määritettiin uusin Android-versio Android 5 (Lollipop), joka vastaa API 21 -tasoa. Alin API-taso, jolla sovellus toimii on API 8 ja se vastaa Android 2.2 (Froyo) -versiota. Androidin toisiksi viimeisin versio 4.4 eli KitKat toimii sovelluksen testausversiona (compile with). Sovellus siis toimii Androidin versioissa 2.2 - 5, joita on käytössä 2010-luvulla julkaistuissa laitteissa. Toimintaa testattiin Eclipsen tarjoamalla emulaattorilla sekä Samsung S3 mini- ja S5-laitteilla, joissa on Androidin versiot 4.1.2 ja 5.

6.2 Sovelluksen kuvaus

Toteutettu Anatomy Game -sovellus on tietopeli. Käyttäjälle esitetään ihmiselimistön osan, esimerkiksi sisäelimen nimi, suomeksi ja annetaan hänelle elimen vaihtoehtoisia käännöksiä latinaksi. Käyttäjä voi valita tietopeliin sisältyvät kehonosat. Vaihtoehtoina ovat sisäelimet, luut ja lihakset tai jokin näiden kehonosien yhdistelmä. Vastausvaihtoehtoja käyttäjälle on tarjolla kolme tai kuusi hänen tekemiensä valintojen mukaan (ks. kuvio 7). Käyttäjän on valittava käännösvaihtoehtoja, kunnes hän valitsee oikean käännöksen, kuten kuvio 8 ilmentää.



Kuvio 8. Anatomian tietopelin näytön sisältö, kun käyttäjä on valinnut oikean vastausvaihtoehdon (vasemmanpuolimmainen kuva), väärän vastausvaihtoehdon (keskellä) ja pelin päättyessä näytettävä tulosikkuna (oikeanpuolimmainen kuva).

Tietopelin tarjoamat vastausvaihtoehdot ovat nappuloissa, kuten kuvioista 7 ja 8 ilmenee. Mikäli käyttäjä painaa väärän vastausvaihtoehdon sisältävää nappulaa, muuttuu tämä nappula inaktiiviseksi sekä väreiltään himmennetyksi merkiksi siitä, että käyttäjä on jo tarjonnut sitä vastausvaihtoehtoa. Lisäksi näytölle tulostuu ”Yritä uudelleen!” -teksti. Kyselyn on siis tarkoitus reagoida kannustamalla oikeaan vastaukseen. Kysely sisältää viisi satunnaisjärjestyksessä olevaa sanaa. Kyselyn päättymisen jälkeen sovellus ilmoittaa, kuinka monta kertaa käyttäjä valitsi sanan löytääkseen kaikille sanoille oikean latinankielisen käännöksen ja montako prosenttia hänen vastauksistaan oli oikeita (kuvio 8).

Sovellukseen valittiin mahdollisuus valita useamman kerran oikeaa käännöstä, koska näin käyttäjä pysyy saman sanan parissa pidemmän aikaa ja saa kokemuksen oikeasta sana–käännös -parista. Jokaisen sanan kohdalla päädytään myös onnistumisen kokemukseen. Nämä kaikki seikat edistävät oppimista. Sovelluksen valittiin myös laskevan kaikkien käännösyritysten määrää, jotta käyttäjä voi verrata, kuinka monella yrityskerralla milläkin kertaa onnistuu saamaan kaikki käännökset oikein. Sovelluksen tarjoama runsaiden toistojen määrä sekä itsen ja toisten kanssa kisailu edistävät oppimismotivaation ylläpitoa sekä oppimista. Latinankielisten käännösten oppiminen vaatii runsasta toistoa, minkä raskautta keventää pelimuotoinen toiminta. Nykyihmisen, etenkin nuorison, opiskeluun keskittymistä edistävät tutkitusti tekniikkapohjaiset oppimisympäristöt, jotka ovat heille muista yhteyksistä tuttuja.

6.3 Sovelluksen toteutus

Sovellus toteutettiin Linux-pohjalle asennetulla Eclipse-kehitysympäristössä, jota käyttäen olin suorittanut aiemmin Android-ohjelmointi -kurssin. Halusin suorittaa mainitun kurssin esivalmisteluna opinnäytetyön laatimiselle. Kurssitehtävinä toteutetut sovellukset olivat ratkaisevasti erityyppisiä kuin opinnäytteenä toteutettavaksi valitsemani sovellus. Tämä mahdollistaa osaamiseni laajentamisen ja syventämisen. Versionhallintaohjelmistona käytin Apache Subversion -ohjelmaa, johon tutustuin nyt ensimmäistä kertaa. Kyseessä on avoimen lähdekoodin versionhallintajärjestelmä, joten se sopii hyvin käytettäväksi Linux-ympäristössä. (Apache Subversion 2015.)

Sovelluksen laatimisen aloitin luomalla projektin nimeltä Anatomy, johon kaikki sovellukseen kuuluvat tiedostot sijoitetttaisiin. Ensimmäisenä rakensin sisällön Android-

Manifest.xml-tiedostolle, jossa määrittelin sovelluksen perustiedot. Laaditun sovelluksen AndroidManifest.xml-tiedosto on esitelty kuviossa 3, sivulla 17, manifest-tiedoston yleisten rakenneperiaatteiden läpikäymisen yhteydessä.

Päätin toteuttaa sovelluksen toimimaan yhden näytön -periaatteella ja hyödyntää valikkorakennetta sovelluksen muuntelun mahdollistajana. Päätiedostolle annoin nimeksi ”AnatomyGame” ja se tuli sisältämään sovelluksen varsinaisen ohjelmistokoodin. Toisena src-tiedostona sovelluksessa on databaseHelper-tiedosto, jossa sijaitsee sovelluksen tietokannan luomiseen ja ylläpitämiseen vaadittava koodi. Käytin sovelluksessa lisäksi resurssitiedostoja tässä työssä aiemmin esitellyllä tavalla.

7 Pohdinta

Sovelluksen toteuttaminen oli erittäin mielenkiintoista ja opettavaista, sillä aihealue on minulle läheinen ja Android-ohjelmointitaitoni ovat vasta kehittymässä. Sovelluksen toteutuksen aikana tutustuin Androidin tarjoamiin monenlaisiin mahdollisuuksiin ja komponentteihin. Minun on myös todettava tehneeni ajoittain virheellisiä päätelmiä ja komponenttivalintoja. Esimerkiksi valikkoja ei olisi kannattanut toteuttaa käyttäen Alert Dialogy -komponenttia, sillä se ei tarjoa kaikkia ominaisuuksia, joita tämän sovelluksen toteuttamisessa olisi voinut käyttää. Preference Activity -komponentti olisi todennäköisesti ollut toimivampi ratkaisu valikkojen toteutukseen. Sen perusominaisuuksiin kuuluu esimerkiksi tähänkin sovellukseen mahdollisesti hyödyllinen sovelluksen vallitsevan tilan tallentaminen sovelluksen käytön keskeytyessä. Preference Activity -komponentin käyttäminen soveltuu tilanteeseen, jossa sovelluskehittäjän tavoitteena on luoda hieman vanhemmissakin Androidin versioissa toimiva sovellus. Uusimmille käyttöjärjestelmäversioille suunnattujen sovellusten valikkorakenteen voi toteuttaa omalla Activityn perivällä luokalla, jolloin käytettävissä ovat muun muassa monipuoliset preference- ja settings -toiminnot. Sovelluksissa, joihin tämän työn tekijä tutustui, valikkojen toteutukseen on käytetty kaikkia edellä mainittuja komponentteja, joten kaltaiseni aloittelijan on liki mahdoton ilman kokeiluja heti tehdä parasta mahdollista komponenttivalintaa. Insinöörityön tekoprosessi jättikin minulle innostuksen laajentaa alan ymmärtämystäni ja osaamistani.

Työskentelyssä minulla ei ollut mainittavia ongelmakohtia. Sain toteutettua sovelluksen toimimaan ennen aloittamista määrittelemälläni tavalla toimivaksi ja tarkoituksenmukai-

seksi. Laatimani sovellus noudattaa perinteisen tietopelin perusrakennetta: kysymys, vastausvaihtoehdot ja tuloslaskenta. Terveysalan taustastani johtuen sovelluksen sisältö perustuu ihmisanatomiselle terveyssanastolle, mutta sovellus on tietokannan sisältöä lisäämällä tai vaihtamalla helposti muokattavissa laajemmaksi tietopeliksi tai toiselle kielelle tai alalle soveltuvaksi. Ohjelmoin sovellukseen viiden kysymyksen tietopelin demonstraatiomielessä. Kysymysten määrä on yksinkertaisin toimin nopeasti muokattavissa. Sovelluksen toteutuksessa pyrkimyksenäni oli ohjelmoinnin ideaalin mukaisesti tehdä mahdollisimman paljon dynaamisia määrityksiä sekä koota käsitteitä vakioiksi jatkokehityksen helpottamiseksi.

Sovellus on toimiva ja käyttökelpoinen, mutta siihen voisi lisätä monenlaisia lisäominaisuuksia. Niiden lisäämisessä on kuitenkin muistettava käytettävyyden ylläpitäminen. Käyttäjäystävällisyyttä voisi lisätä sovelluksen ominaisuus käyttää uudelleenkäynnistettäessä edellisellä kerralla tehtyjä asetuksia koskien kyselyn aluetta ja vastausvaihtoehtojen määrää. Sovelluksen toteutuksessa valitsemani linjaus toteuttaa kysely oletuksena kaikkien kehonosien mukana oleminen on perusteltu lähtötilanne. Käyttäjälle on tarjolla laajin mahdollinen tietokanta, jota hän voi halutessaan kaventaa. Vastausvaihtoehtojen oletusmääräksi taas asetettiin kolme kuuden sijasta. Näin käyttäjälle on tarjolla keskimääräinen toteutus, johon kyseinen sovellus taipuu. Käyttäjän on sitten mahdollista muokata toteutusta helpommaksi tai vaikeammaksi tarpeidensa mukaan.

Toteutin pelin nyt äänettömänä, mutta käyttäjälle voisi antaa myös mahdollisuuden kytkeä peliin äänet. Ääni- ja värimaailmoja voisi olla tarjolla käyttäjälle valikkoina. Koska peli toteutettiin äänettömänä, on useamman pelaajan mahdollista pelata sitä samassa tilassa, esimerkiksi luokassa, samanaikaisesti toisiaan häiritsemättä. Ympäristö ei myöskään saa tietoa sovelluksen käyttäjän tietotasosta, koska sovellus ei sitä äänin ilmianna. Sovellusta olisikin mahdollista käyttää myös osaamistestitarkoituksessa opetuksessa tai vaikka työhaastattelussa.

En katsonut olevan tarpeellista, että sovellus muistaisi kyselyn tilanteen siinä tilanteessa, kun sovelluksen käyttö keskeytyy kyselyn ollessa kesken. Näin sovelluksesta on saatavilla tulosyhteenveto vain yhdellä kertaa täytetyn kyselyn vastauksista, mikä tuntuu luonnolliselta. Sovelluksen kyselyominaisuuksia voisi laajentaa osana jatkokehitystä. Sovellus toimii vain suomesta latinaan käännössuunnassa ja sen voisi laajentaa toimimaan toiseenkin suuntaan. Käyttäjälle voisi antaa valinnanmahdollisuuden sanojen kääntämissuunnasta tai mahdollisuuden valita tietopeliin sekaisin sekä suomen-

että latinankielisiä sanoja. Sovelluksessa voisi olla useammanlaisia kysymyksiä sekaisin tai käyttäjällä valittavissa haluaako hän esimerkiksi kuvallisia vai sanallisia tehtäviä. Sovellukseen voisi myös lisätä käyttäjälle mahdollisuuden ladata uusia sanoja tai sanastoja kyselyn pohjaksi.

Mikäli sovelluksen tietokantaa laajentaisi ja asettaisi kyselyyn useampia kysymyksiä, toteutettu tietopeli olisi käytettävissä esimerkiksi toisen asteen ammatillisessa koulutuksessa sekä ammattikorkeakoulutasolla. Kymmenen kysymyksen tietopeli olisi todennäköisesti sopivanlaajuinen hoitajaopiskelijoille. Sovellusta voisi käyttää myös hoitoalalla työskentelevien osaamisen ylläpitomateriaalina. Opettaja voisi anatomian kurssin oppitunnin aluksi toteuttaa perinteisen pistokokeen nuorille mieleisellä menetelmällä älypuhelimia hyödyntäen. Opettaja saisi jokaiselta opiskelijalta testituloksen valmiiksi laskettuna, mikä säästäisi aikaa muulle opetuksen toteuttamiselle sekä antaisi opiskelijoille tilannesidonnaisen palautteen, minkä on todettu olevan tehokkaampaa kuin jälkikäteen saatavan palautteen. Mikään ei poissulje myöskään sovelluksen yleissivistävää tai viihdekäyttöä vaikka illanistujaisissa. Yhteenvetona voin todeta työskentelyn opettaneen ja innostaneen minua tuottaen samalla jotain todellisessa elämässä hyödyntämiskelpoista, mikä lisää tyytyväisyyttäni sekä saa miettimään, josko voisin tulevaisuudessa toteuttaa enemmänkin opetusmateriaalia.

Lähteet

A Visual Guide to Version Control. 2014. Verkkodokumentti. <http://betterexplained.com/articles/a-visual-guide-to-version-control/>. Luettu 17.11.2014.

Activities. 2014. Verkkodokumentti. <http://developer.android.com/guide/components/activities.html>. Luettu 28.10.2014.

Android (operating system). 2014. Verkkodokumentti. <http://en.wikipedia.org/wiki/Android_(operating_system)>. Luettu 17.10.2014.

Android 5.0, Lollipop. 2014. Verkkodokumentti. <http://www.android.com/versions/lollipop-5-0/>. Luettu 17.10.2014.

Android application components overview. 2014. Verkkodokumentti. <http://www.vogella.com/tutorials/Android/article.html#components>. Luettu 28.10.2014.

Android application components. 2014. Verkkodokumentti. <http://www.tutorialspoint.com/android/android_application_components.htm>. Luettu 27.10.2014.

Android Architecture. 2014. Verkkodokumentti. <http://www.tutorialspoint.com/android/android_architecture.htm luettu 20.10.2014.

Android Emulator. 2014. Verkkodokumentti. <http://developer.android.com/tools/help/emulator.html>. Luettu 17.11.2014.

Android ohjelmien tekeminen. 2014. Verkkodokumentti. <http://www.mikarantakeisu.fi/internetsivut/content/android-ohjelmien-tekeminen>. Luettu 20.10.2014.

Android ohjelmointi. 2014. Mobiiliohjelmointi. Verkkodokumentti. <http://some.lappia.fi/wiki/images/f/fe/Android_ohjelmointi_01.pdf>. Luettu 20.10.2014.

Android Studio. 2014. Verkkodokumentti. <https://developer.android.com/sdk/index.html>. Luettu 20.10.2014 ja 17.11.2014.

Android Styles and Themes. 2015. Verkkodokumentti. <http://www.tutorialspoint.com/android/android_styles_and_themes.htm>. Luettu 15.1.2015.

Android versions comparison. 2014. Verkkodokumentti. <http://socialcompare.com/en/comparison/android-versions-comparison>. Luettu 17.10.2014.

Android, the world´s most popular mobile platform. 2014. Verkkodokumentti. <http://developer.android.com/about/index.html>. Luettu 17.10.2014.

Android. 2014. Verkkodokumentti. <http://fi.wikipedia.org/wiki/Android>. Luettu 17.10.2014.

Android. 2015. Verkkodokumentti. <www.android.com/history/>. Luettu 16.1.2015.

Apache Subversion. 2015. Verkkodokumentti. <https://subversion.apache.org/>. Luettu 20.1.2015.

App Structure. 2014. Verkkodokumentti. <http://developer.android.com/design/patterns/app-structure.html>. Luettu 20.10.2014.

Bransford, J. D., Brown, A. L., Donovan, M. S., Cocking, R. R. (toim.), Pellegrino, J. W. (toim.) 2004. Miten opimme. Aivot, mieli, kokemus ja koulu. Helsinki: WSOY.

Color-Hex. 2015. Verkkodokumentti. <http://www.color-hex.com/>. Luettu 15.1.2015.

Dashboards. 2015. Verkkodokumentti. <http://developer.android.com/about/dashboards/index.html#2015>. Luettu 4.2.2015.

Deitel, P., Deitel, H., Deitel, A., Morgano, M. 2012. Android for Programmers An App-Driven Approach. Deitel Developer Series. Prentice Hall.

Developing. 2014. Verkkodokumentti. <https://source.android.com/source/developing.html, 3.11.2014.

Gargenta, M., Nakamura, M. 2014. Learning Android. Develop Mobile Apps Using Java And Eclipse. 2. painos. O'Reilly Media.

GIT. 2014. Verkkodokumentti. <http://git-scm.com/book/fi/v1/Alkusanat-Versionhallinnasta>. Luettu 17.11.2014.

History of Android: First Applications, Prototypes & Other Events. 2014. Verkkodokumentti. <http://www.brighthub.com/mobile/google-android/articles/18260.aspx>. Luettu 17.11.2014.

Huhta, E., Väänänen, M., Smeds, R. 2011. Koulujen ja yritysten verkostoyhteistyö - odotukset, edellytykset ja johtaminen (s. 234–250). Teoksessa Opetusteknologia koulun arjessa II. Jyväskylän yliopisto. Koulutuksen tutkimuslaitos. Verkkodokumentti. <http://ktl.jyu.fi/img/portal/21724/Verkkoversio_102.pdf>. Luettu 15.1.2015.

iOS 8. 2015. Verkkodokumentti. <http://mobiili.fi/tag/ios-8/>. Luettu 4.2.2015.

Järvelä, S., Järvenoja, H., Simojoki, K., Kotkaranta, S., Suominen, R. 2011. Miten opettajat ja oppilaat käyttävät tieto- ja viestintäteknologiaa koulun arjessa (s. 42–54). Teoksessa Opetusteknologia koulun arjessa II. Jyväskylän yliopisto. Koulutuksen tutkimuslaitos. Verkkodokumentti. <http://ktl.jyu.fi/img/portal/21724/Verkkoversio_102.pdf>. Luettu 15.1.2015.

Kankaanranta, M., Vahtivuori-Hänninen, S. 2011. Johdanto (s. 9–16). Teoksessa Opetusteknologia koulun arjessa II. Jyväskylän yliopisto. Koulutuksen tutkimuslaitos. Verkkodokumentti. <http://ktl.jyu.fi/img/portal/21724/Verkkoversio_102.pdf>. Luettu 15.1.2015.

Kari, J. (toim.) 1994. Didaktiikka ja opetussuunnittelu. Juva: WSOY.

Käyttöjärjestelmä. 2014. Verkkodokumentti. <http://www.cs.tut.fi/etaopetus/titepk/luku15/kj.html>. Luettu 17.10.2014.

Meier, R. 2010. Professional Android 2 Application Development. Wiley Publishing, Inc.

Microsoft and Nokia complete mobile phone unit deal. 2013. Verkkodokumentti. <http://www.bbc.com/news/business-27163667>. Luettu 17.10.2014.

Microsoft ostaa Nokian puhelinliiketoiminnan. 2013. Verkkodokumentti. <http://yle.fi/uutiset/microsoft_ostaa_nokian_puhelinliiketoiminnan/6811366>. Luettu 17.10.2014.

Mobiilia liiketoimintaa ja teknologiaa. 2014. Verkkodokumentti. <http://ristola.wordpress.com/2014/03/27/markkinakatsaus-q12014-mobiilikayttojarjestelmat/>. Luettu 17.10.2014.

Mobile Operating System (OS). 2014. Verkkodokumentti. <http://www.webopedia.com/TERM/M/mobile_operating_system.html, 17.10.2014.

Mobile Operating Systems (Mobile OS) Explained. 2014. Verkkodokumentti. <http://www.webopedia.com/DidYouKnow/Hardware_Software/mobile-operating-systems-mobile-os-explained.html>. Luettu 17.10.2014.

Näin pääset Android kehityksessä alkuun. 2014. Verkkodokumentti. <http://mobiilikehitys.fi/nain-paaset-android-kehityksessa-alkuun/>. Luettu 17.11.2014.

Ohjelmiston versionhallinta 2014. Verkkodokumentti. <http://fi.wikipedia.org/wiki/Ohjelmiston_versiohallinta>. Luettu 3.11.2014.

Saarenpää, H. 2009. Johdatusta oppimispelien ja pelaamalla oppimisen maailmoihin. Verkkodokumentti. <http://pelitieto.net/oppimispelit-ja-hyotypelaaminen/>. Luettu 24.11.2014.

Saariluoma, P., Kujala, T., Kuuva, S., Kymäläinen, T., Leikas, J., Liikkanen, L. A., Oulasvirta, A. 2010. Ihminen ja teknologia. Hyvän vuorovaikutuksen suunnittelu. Helsinki: Teknologiateollisuus.

SDK - software development kit. 2014. Verkkodokumentti. <http://www.webopedia.com/TERM/S/SDK.html>. Luettu 20.10.2014.

Smartphone OS Market Share, Q3 2014. 2014. Verkkodokumentti. <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>. Luettu 17.10.2014 ja 4.2.2015.

Subversion ja versionhallinta. 2014. Verkkodokumentti. <http://verteksi.net/soks/2008/05/07/subversion-ja-versionhallinta/>. Luettu 17.11.2014.

The biggest iOS release ever. 2014. Verkkodokumentti. <https://www.apple.com/ios/>. Luettu 17.10.2014.

The world´s most advanced mobile OS. 2014. Verkkodokumentti. <https://www.apple.com/ios/what-is/>. Luettu 17.10.2014.

Top android SDK versions. 2014. Verkkodokumentti. <http://www.appbrain.com/stats/top-android-sdk-versions>. Luettu 17.10.2014.

What is iOS? 2014. Verkkodokumentti. <http://whatisios.org/>. Luettu 17.10.2014.

Where´s my Gphone? 2014. Verkkodokumentti. <http://googleblog.blogspot.fi/2007/11/wheres-my-gphone.html>. Luettu 4.11.2014.

Windows Phone OS. 2014. Verkkodokumentti. <http://www.gsmarena.com/glossary.php3?term=windows-phone-os>. Luettu 17.10.2014.

Älypuhelin kuluu koulussakin. 2014. Suomen Kuvalehti 37/2014.