Haaga-Helia
ammattikorkeakoulu Oy

Mobiilisovelluskehitys hybriditekniikalla ja open-source työkaluilla

Niki Ahlskog

7.5.2015

Haaga-Helia ammattikorkeakoulu Oy

Tiivistelmä

7.5.2015

Tekijä
Niki Ahlskog

Koulutusohjelma
Tietojenkäsittely

Opinnäytetyön nimi
Mobiilisovelluskehitys hybriditekniikalla ja open-source työkaluilla

Sivu- ja liitesivumäärä
65+1

Opinnäytetyön nimi englanniksi
Mobile app development with hybrid technique and open-source tools

Opinnäytetyön aiheena on mobiilisovelluskehitys selainpohjaisilla ohjelmointikielillä ja niiden kääntäminen mobiililaitteeseen soveltuvaksi, jolloin lopputuloksena on niin sanottu hybridisovellus, joka voidaan kääntää nopeasti mille tahansa mobiililaitteelle. Hybriditekniikan käyttö mahdollistaa sen, että sovelluskehittäjä kirjoittaa sovelluksen kerran ja se voidaan kääntää jatkossa mille tahansa alustalle. Työ on tehty Haaga-Helian mobiilituotekehityskurssin puutteellisten ohjeiden takia.

Kaikki työssä käytettävät työkalut ovat avointa lähdekoodia ja täysin ilmaisia. Pääkäännöstyökaluksi on valittu PhoneGap-sovelluskehys, lisäksi työssä esitellään kahta verkkopohjaista käännöspalvelua. Tavoitteena on käsitellä PhoneGapin toimintaa, ominaisuuksia, etuja ja haittoja sekä kehitysprosessin sisältöä.

Opinnäytetyössä ohjeistetaan mitä kaikkea hybridi-mobiilisovelluskehityksen aloittaminen vaatii Linux-työasemalla. Teoriaosuudessa kerrotaan tarkemmin PhoneGapin toiminnasta ja sen periaatteista. Opinnäytetyön aikana tehtiin toimiva PhoneGap työasema-asennus, Android SDK -työkalut otettiin käyttöön ja luotiin esimerkkinä mobiilipeli, joka hyödyntää jQuery Mobile -kirjastoa. Työssä käydään läpi kaikki vaiheet mobiilisovelluskehityksen aloittamiseksi ja valmiin sovelluksen tuottamiseksi.

Projektin ohjelmistokoodeja ylläpidettiin GitHub-palvelussa, jonka käyttöön annetaan myös ohjeet. Sovelluksen kehitysprosessia ja rakennetta esitellään opinnäytetyössä tarkemmin asennusohjeiden lisäksi. Projektin aikana syntyneet koodit ovat myös yleiskäyttöisiä selainsovellusten kannalta, joten työn tuloksena saatua tutkimustietoa voidaan hyödyntää myös tulevissa selainsovellusprojekteissa.

Kokonaisuutena opinnäytetyöprojekti onnistui hyvin alusta loppuun. Lopputuloksena syntyi ohjeet mobiilikehittämisen aloittamiseen, avoimesta Github-varastosta löytyvät esimerkkisovelluksen lähdekoodit valmiina käännettäväksi mille tahansa mobiililaitteelle, sekä Google Play -kaupasta ladattava Android-peli.

Asiasanat
Android, PhoneGap, mobiilikehitys, sovellus, Linux

HAAGA-HELIA
University of Applied Sciences

Abstract

7 May 2015

Author
Niki Ahlskog

Degree programme
Bachelor of Information Technology

Report/thesis title
Mobile app development with hybrid technique and open-source tools

Number of pages and appendix pages
65+1

This thesis is about mobile application development with web-based programming languages, and their translation to the mobile device application. The end result is a so-called hybrid application, which can be quickly compiled to any mobile device. Hybrid technique allows the developer to write an application once, after which the application can be compiled to any platform. The thesis work is done due to lack of instructions in Haaga-Helia's Mobile development course.

All work tools used are open-Source and completely free of charge. The main tool selected is the PhoneGap application framework. In addition, two web-based translation services are introduced. The aim is to go through the PhoneGap features, its advantages and disadvantages, and look at the development process as a whole.

This thesis provides guidance on what the developer needs in order to start a hybrid mobile application development on a Linux workstation. The theoretical part describes in more detail, how the PhoneGap operates and what its main principles are. During the study working PhoneGap workstation installation was accomplished, the Android SDK tools were introduced as well as a working example of a mobile game that utilizes the jQuery Mobile library was published in Google Play Store. The thesis goes through all the steps of mobile application development from the initiation to the finished application.

Project software codes were maintained at the GitHub service, which also provided instructions. The application development process and the structure of the application are presented in more detail in addition with the installation instructions. In browser application point of view, all project codes are also general-purpose type, so the results of the research work can be utilized in future browser application projects as well.

All in all, the objectives set were achieved. The thesis project resulted in an open GitHub repository, where the source codes can be found, that are ready to be compiled to any mobile device. And as well as the Google Play Store displays a downloadable mobile game for Android.

Keywords
Android, PhoneGap, Mobile development, App, Linux

Sisällys

1 Johdanto

Tämä opinnäytetyö on laadittu auttamaan Haaga-Helian opiskelijoita sekä muita kiinnostuneita aloittamaan hybridimobiilisovelluskehitys. Aihe on valittu, koska Haaga-Helian mobiilisovelluskehityskurssilta puuttuivat selkeät ohjeet valittujen tekniikoiden käyttämisestä. Opinnäytetyössä esitellään kuinka mobiilisovelluskehitys voidaan aloittaa mille tahansa alustalle, ja miten tarvittavat työkalut asennetaan Linux-ympäristöön. Opinnäytetyö tehdään projektityyppisesti, jossa käydään lävitse jokainen työvaihe, kehitysympäristön asennus, sovelluksen tekeminen, versionhallinta, lopulta pakkaus ja allekirjoitus valittuun sovelluskauppaan ladattavaksi. Tämän työn lukija saa valmiudet mobiilisovelluskehityksen aloittamiselle.

Kehitysympäristönä käytetään Linux-käyttöjärjestelmää ja PhoneGap-sovelluskehystä. Kehittäjän käyttöjärjestelmäksi on valittu Linux-pohjainen käyttöjärjestelmä Xubuntu, joka on Ubuntun rinnakkaisversio. Linux-käyttöjärjestelmä on vapaata, avointa lähdekoodia ja on valittu käyttöjärjestelmäksi tehokkaan komentotulkin, sekä kätevän pakettienhallinnan vuoksi.

Itse sovellus kehitetään hybriditekniikalla Android alustalle. Lopuksi sovellus ladataan Google Play -kauppaan. Aihe on kiinnostava, koska PhoneGap soveltuu käytettäväksi kehitysalustana mille tahansa mobiililaitteelle, Androidille, Applen iOS ja Windows-puhelimille.

Opinnäytetyö on suunnattu kaikille mobiilisovelluskehityksestä kiinnostuneille, sekä web-kehittäjille, jotka haluavat siirtyä ketterästi mobiilisovelluskehitykseen. Työ sopii myös kehittäjälle, jolla on tarve saada nopeasti luotua sovellus kaikille kolmelle eri alustalle.

Työssä vertaillaan myös paikallista PhoneGap-asennusta pilvipalveluihin. Mitä hyötyä ja haittoja on paikallisesta asennuksesta? Miten pilvipalvelut eroavat tästä? Kumpaa kannattaa lopulta käyttää?

1.1 Projektin tausta, tehtävät ja tavoitteet

Projekti on valittu Haaga-Helian mobiilituotekehityskurssin puutteellisten ohjeiden takia. Työn tarkoituksena on auttaa tietojenkäsittelyn opiskelijoita viemään mobiilikehitysprojekti onnistuneesti maaliin asti.

Projektissa syntyvä lähdekoodi on avointa ja julkista. Koodit tullaan julkaisemaan avoimessa GitHub repositoriossa. Projektilla ei ole budjettia.

Projektin tavoitteina on luoda mahdollisimman yksinkertainen ohjeistus PhoneGapin asentamisesta ja käyttämisestä. Esimerkkinä tullaan luomaan mobiilipeli PhoneGapilla. Valmis sovellus tullaan lataamaan ja allekirjoittamaan Google Play -kauppaan. Konkreettiset lopputulokset ovat toimiva PhoneGap-työasema-asennus Linux-ympäristössä, sovelluksen avoimet lähdekoodit, sekä allekirjoitettu ja paketoitu valmis sovellus Google Play -kaupassa, joka on kaikkien vapaasti ladattavissa.

Projektin tehtävänä on toteuttaa monialustainen, alustariippumaton mobiilipeli PhoneGap-sovelluskehystä hyödyntäen. PhoneGap mahdollistaa sovelluskehityksen alustariippumattomasti. Sovelluksia luodaan niin sanotulla hybriditekniikalla, eli JavaScript, HTML5 ja CSS3 -web-tekniikoilla. Tavoitteena on tuottaa toimiva ja julkaisukelpoinen sovellus.

Asetetut tavoitteet saavutetaan käytännön projektityössä.

1.2 Haasteet ja aiheen rajaus

Projektiin liittyy muutamia haasteita. Opinnäytetyöntekijällä on vain vähän JavaScript ja jQuery osaamista. Haaga-Helia ei tarjoa näihin erillisiä kursseja, vaan oppiminen on jätetty opiskelijan omalle vastuulle. Lisäksi PhoneGap sisältää monia laajennuksia (plugin), joista projektin tekijällä ei ole aikaisempaa kokemusta. Projektin tekninen puoli edellyttää siis JavaScript ja jQuery osaamisen hankkimista muualta.

Haasteista selviytyminen vaatii tietämystä ohjelmistokehityksestä, Linux-ympäristöstä, erilaisista web-tekniikoista ja ohjelmoinnista. Valitut tekniikat ja kirjastot tullaan valitsemaan sovelluskehityksen aikana.

Aihe on rajattu vain Linux ja Android ympäristöihin. Linux-ympäristö on valittu, koska terminaalityöskentely on jouhevampaa ja tarvittavat paketit on helppoa asentaa Linuxille. Sovellus testataan ja paketoidaan lopuksi Androidille, koska tarvittavia fyysisiä testilaitteita muille alustoille ei ole saatavilla. Opinnäytetyön lukijoiden oletetaan ymmärtävän ohjelmoinnin, Linuxin ja web-kehityksen perusteet. Jotta tätä ohjetta pystyisi seuraamaan, tulee lukijan lisäksi osata työskennellä komentokehoitteella.

2 Teoriatausta

Tässä luvussa perehdytään PhoneGap-sovelluskehykseen ja sen toimintaan syvemmin. Erotellaan käsitteet, kuten hybriditeknologia, natiivikehitys ja mobiili web-sovellus. Lisäksi selitetään sovelluksessa käytetyt teknologiat ja katsotaan muiden laitealustojen lisenssimaksuja.

2.1 Mobiilisovelluskehitystekniikat

Mobiilisovelluksia on mahdollista kehittää kolmella eri tekniikalla:

1. Natiivisovellus
2. Hybridisovellus
3. Mobiili Web-sovellus

Natiivisovellukseksi sanotaan sovellusta, joka on kehitetty toimimaan tietyllä laitteella, tai alustalla. Koska natiivisovellus on kehitetty tietylle alustalle, voi sovellus hyödyntää käyttöjärjestelmän toimintoja, tai muita ohjelmistoja jotka on tyypillisesti asennettu kyseiselle laitteelle. Natiivisovellukset voivat yleensä hyödyntää viimeisintä teknologiaa laitteessa, kuten esimerkiksi GPS:ää (Global Positionin System), tai vaikkapa kameraa. Natiivisovellus asennetaan suoraan tietylle laitteelle ja kehittäjät tekevät erillisen version ohjelmistosta jokaiselle laitteelle. (Rouse 2013.)

Hybridisovellus on termi, jolla viitataan sovellukseen, joka on kehitetty käyttämällä HTML5:ttä käyttöliittymän toteuttamisessa. Sovellus on riippuvainen natiiveihin laitekohtaisiin toimintoihin pääsyyn, jotka eivät ole saatavilla normaaleissa web-sovelluksissa. Suurin osa tästä natiivista koodista ei ole näkyvää. Tietoa vain yksinkertaisesti ohjataan takaisin HTML5 kerrokseen, jossa se renderöidään, eli muutetaan käyttäjälle näkyväksi. (Traeg 2013.) Sovelluskehitys HTML5:llä on tapa jakaa koodi eri käyttöympäristöille, kirjoittamatta sovellusta alusta loppuun erikseen jokaiselle laitteelle.

Mobiilisovellusta sekä mobiilia web-sivua käytetään kumpaakin älypuhelimella tai tabletilla. Mobiili web-sivu on sama kuin mikä tahansa internet-sivu, jota voidaan käyttää tietokoneella internetissä. Erona mobiilisivussa on se, että sivuston ulkoasu on suunniteltu käytettäväksi myös pienellä mobiililaitteella sekä koskettamalla. Kuten mikä tahansa internet-sivu, voi mobiilisivuilla näyttää tekstiä, kuvia, videoita ja niin edelleen. Sivuilla voi lisäksi olla mobiililaitteille tarkoitettuja toimintoja, kuten klikkaa ja soita tai sijaintiin

perustuvia karttapalveluita. Tavallinen mobiilisovellus eroaa web-sovelluksesta siten, että sovellukset ladataan ja asennetaan mobiililaitteeseen, kun taas web-sovellus renderöidään selaimella aina pyydettäessä. Jos tavoitteena on saada laajin mahdollinen kävijäkunta sivuille, on syytä toteuttaa projekti mobiilina web-sovelluksena. Mobiilisovellus kannattaa tehdä silloin, kun tarvitaan toimintoja, joita ei voida web-sovelluksessa tehdä tai toteuttaa tehokkaasti. Mobiilisovelluksen etuina ovat esimerkiksi laitteen sensoreiden käyttö, kuten kamera, GPS, tai muut vastaavat. (Summerfield.)

2.2 Android SDK ja emulaattori

Android SDK on kehityspaketti, joka mahdollistaa sovellusten tekemisen Android-alustalle. Android SDK sisältää esimerkkiprojektin koodeineen, kehitystyökalut, emulaattorin ja tarvittavat kirjastot Android-sovelluksen rakentamiseen (Webopedia). SDK tulee sanoista Software Development Kit.

Android SDK sisältää virtuaalipuhelimen, jota ajetaan kehittäjän tietokoneella. Emulaattori mahdollista prototyyppien rakentamisen, testaamisen ja kehittämisen ilman fyysistä Android laitetta. Android emulaattori matkii kaikkia laitteisto- ja ohjelmistotoimintoja, joita oikeassa mobiililaitteessa olisi. Poikkeuksena, että sillä ei voi soittaa oikeita puheluita. Myöskään sensoreiden toimintaa ei voida oikeasti kokeilla, vaan niitä voidaan matkia. Esimerkiksi GPS:n tai kiihtyvyysanturin käyttämistä voidaan ”huijata”. Emulaattorilla on hyvä kokeilla sovelluksen toimintaa erilaisilla laitteistoasetuksilla ennen julkaisua. (developer.android.com.)

2.3 Node.js ja NPM

Node.js on avoimen lähdekoodin järjestelmäriippumaton ajoympäristö palvelinpuolelle ja verkkosovelluksille. Node.js on kirjoitettu JavaScriptillä ja sitä voidaan suorittaa esimerkiksi OS X, Windows ja Linux ympäristöissä. Node.js on alusta, jolla on tarkoitus rakentaa helposti ja nopeasti skaalautuvia verkkosovelluksia. (NodeJS.org).

NPM on pakettienhallintasysteemi JavaScriptille (NPMJS.com). NPM antaa JavaScript-kehittäjille mahdollisuuden jakaa ja käyttää koodia helposti uudelleen. NPM on online repositorio, jossa voidaan julkaista avointa lähdekoodia. NPM mahdollistaa tarvittavien lisäosien ja moduulien asentamisen kätevästi yhdellä komennolla. (Reed, N 2011).

2.4 PhoneGap

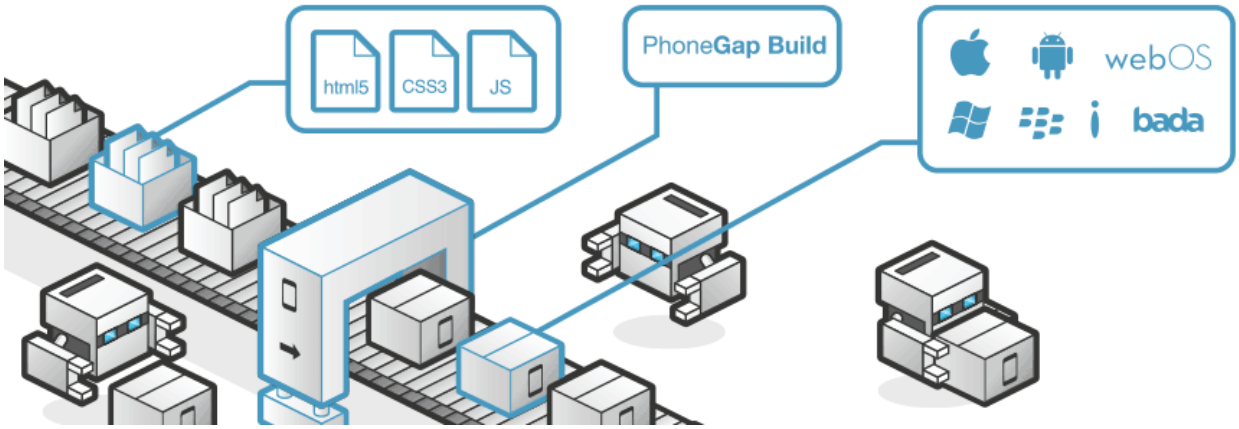
PhoneGap on ilmainen ja avoimeen lähdekoodiin perustuva kehys, jonka avulla voidaan luoda mobiilisovelluksia käyttäen standardisoituja web-tekniikoita halutuille alustoille.

(phonegap.com). PhoneGapilla voidaan nopeasti tuottaa järjestelmäriippumattomia mobiilisovelluksia hyödyntäen HTML5, JavaScript ja CSS web-tekniikoita.

PhoneGap on yksi Cordovan jakeluista. Asian voi ajatella niin, että Cordova on PhoneGapin moottori. Samoin kuin WebKit on Chromen tai Safarin. Tällä hetkellä ainoa ero Cordovalla ja PhoneGapilla on ladattavan paketin nimi ja se tulee pysymään tällaisena toistaiseksi. (Brian 2012.)

Mobiilisovelluksen kehitys erikseen jokaiselle laitteelle, iPhonelle, Androidille, Windows Mobilelle, vaatii monia eri kehitysympäristöjä ja ohjelmointikieliä. PhoneGap ratkaisee tämän kaiken käyttämällä standardeja web-teknologioita, eli se siis yhdistää webin ja mobiililaitteet yhden katon alle.

PhoneGappia on ladattu yli miljoona kertaa ja sitä käyttää yli 400 000 kehittäjää aktiivisesti. Tuhansia mobiilisovelluksia on ladattavissa eri sovelluskaupoissa, jotka on kehitetty PhoneGapilla. PhoneGap tulee aina olemaan ilmainen ja vapaaseen lähdekoodiin perustuva, Apache lisenssin 2.0 alla. (Phonegap. Tietoja.)



Kuva 1. Kuvakaappaus PhoneGapin toimintaperiaatteesta. Lähde: phonegap.com

2.4.1 PhoneGapin toimintaperiaate

PhoneGap on sovelluskehys, joka mahdollistaa natiivisti asennettavan sovelluksen kehityksen käyttäen HTML5, CSS ja JavaScript tekniikoita (Kuva 1). PhoneGapin ”core-engine”, eli ydinmoottori, on lisäksi sataprosenttisesti avointa lähdekoodia. (Cordova.apache.org. Cordovan dokumentit, yleiskatsaus.)

Käyttäjälle näytettävä ”User Interface”, eli käyttöliittymä luodaan samaan tapaan, kuten nettisivustoilla. PhoneGapin käyttöliittymäkerros on web-selain, joka vie 100 % laitteen

leveys- ja korkeussuunasta (Kuva 2). Tämä voidaan ajatella esimerkiksi web-selaimena, ilman Chromea tai Firefoxia. Laitteen sisäinen web-selain renderöi HTML-sisällön ilman normaalista web-selaimesta tuttuja painikkeita, kuten osoitekenttää tai sivuhistoriaa.

Sovellus kehitetään hyödyntäen käytettävissä olevaa tilaa. Navigaatio ja sisältöelementit luodaan kuten käyttöliittymän halutaan toimivan. Käytettävissä ei ole selaimesta tuttuja eteen-, tai takaisinpäin nappuloita, vaan sovellus toimii juuri niin kuin sen halutaan toimivan.

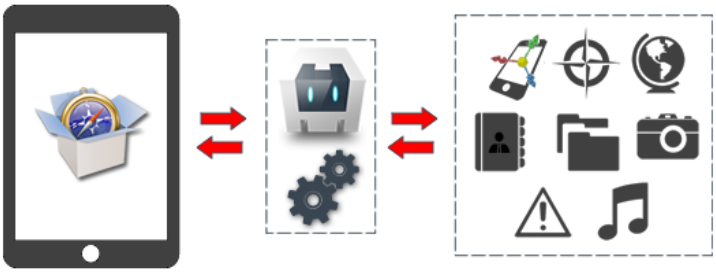
PhoneGapin web-näkymä, on sama kuin laitteen natiivisti käyttämä käyttöjärjestelmä. iOS-laitteella tämä näkymä on ”Objective-C UIWeb View class” Androidilla ”android.webkit.WebView” (Atrice 2012.) Koska laitteiden web-näkymän renderöinnissä on eroja, tulee sovelluksen käyttöliittymää kehittäessä ottaa tämä huomioon.

Kuva 2. Esimerkki PhoneGapin toiminnasta mobiililaitteella.

2.4.2 PhoneGap API

Ohjelmointirajapinta on ohjelmistokoodia tietyillä rutiineilla, jonka mukaan ohjelmat voivat tehdä pyyntöjä keskenään ja vaihtaa tietoa. Ohjelmointirajapinta liitetään usein osaksi SDK-paketteja. Ohjelmointirajapinta on hieman sama asia, kun vertaisi sovelluksen käyttöliittymää ihmisten ja tietokoneiden rajapinnaksi. (Wikipedia. Application Programming Interface.)

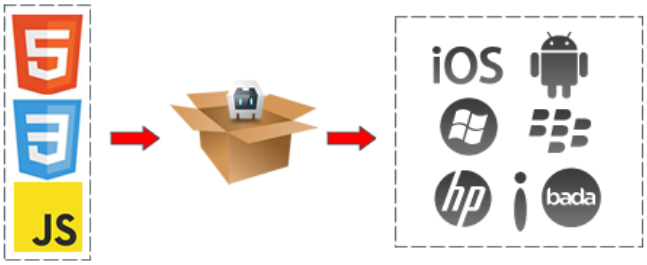
PhoneGap API (Application Programming Interface), eli ohjelmointirajapinta mahdollistaa laitteen natiivien toimintojen käyttämisen JavaScriptiä hyödyntäen. Sovelluksen logiikka ohjelmoidaan JavaScriptillä ja PhoneGapin ohjelmointirajapinta käsittelee kommunikaation natiivin käyttöjärjestelmän välillä (Kuva 3). (Stackoverflow keskustelufoorumi. Meaning of API.)



Kuva 3. Kuvakaappaus PhoneGapin API:n toiminnasta. Lähde: Phonegap.com

2.4.3 PhoneGap sovelluksen paketointi ja jakelu

PhoneGap sovelluskehitys toteutetaan HTML, CSS ja JavaScript -tekniikoilla. Kuitenkin lopullinen PhoneGap-tuote on binäärisovellusarkisto, joka voidaan jakaa tai levittää standardien ekosysteemien kautta (Kuva 4). Esimerkiksi Google Play store, Windows Phone Marketplace tai iTunes Store. Esimerkiksi iOS sovelluksen lopputulema on IPA-tiedosto (iOS Application Archive), Androidilla sovelluksen lopputulema on APK-tiedosto, (Android Package) ja Windows puhelimella XAP-tiedosto. Nämä ovat samoja paketointiformaatteja, kuin mitä natiivisovellukset käyttävät.



Kuva 4. Kuvakaappaus PhoneGapilla paketoinnista eri alustoille. Lähde: Phonegap.com

2.4.4 PhoneGapin korkean tason sovellusarkkitehtuuri

Sovellusarkkitehtuurit eroavat toisistaan tapauskohtaisesti. Useimmat tietokantoihin perustuvat sovellukset noudattavat kuitenkin ”perusarkkitehtuuria”. PhoneGap-sovellus toimii asiakassovelluksena, jolla käyttäjä voi olla vuorovaikutuksessa sovelluksen palvelimen kanssa. PhoneGap-sovellus keskustelee sovelluksen palvelimen kanssa ja hakee dataa. Sovelluksen palvelin hoitaa bisneslogiikan ja käsittelee tietokantaa.

Sovelluksen palvelin on normaalisti web-palvelin, esimerkiksi Apache, IIS, tai muu vastaava. Palvelin tukee jotakin serveripuolen ohjelmointikieltä, esimerkiksi Javaa, .NET:tiä, tai vaikka PHP:tä. PhoneGap-sovellus ei normaalisti keskustele suoraan tietokannan kanssa. Kommunikaatio ohjataan sovelluspalvelimen kautta. Asiakasohjelma keskustelee sovelluspalvelimen kanssa normaaleilla HTTP-pyynnöillä. HTML-sisältöä voidaan lähettää esimerkiksi REST-rajapinnan kautta, JSONilla, tai XML-tiedostoina.

Nämä ovat samoja teknologioita, joita käytettäisiin normaaleissa AJAX-työasemasovelluksissa.

2.4.5 Tuetut toiminnot ja PhoneGapin suorituskyky

	iPhone / iPhone 3G	iPhone 3GS and newer	Android	Blackberry OS 6.0+	Blackberry 10	Windows Phone 8	Ubuntu	Firefox OS
Accelerometer	✓	✓	✓	✓	✓	✓	✓	✓
Camera	✓	✓	✓	✓	✓	✓	✓	✓
Compass	x	✓	✓	x	✓	✓	✓	✓
Contacts	✓	✓	✓	✓	✓	✓	✓	✓
File	✓	✓	✓	✓	✓	✓	✓	x
Geolocation	✓	✓	✓	✓	✓	✓	✓	✓
Media	✓	✓	✓	x	✓	✓	✓	x
Network	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Alert)	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Sound)	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Vibration)	✓	✓	✓	✓	✓	✓	✓	✓
Storage	✓	✓	✓	✓	✓	✓	✓	✓

Kuva 5. Kuvakaappaus PhoneGapin tukemista ominaisuuksista eri mobiilikäyttöjärjestelmissä. Lähde: Phonegap.com

PhoneGap-sovelluskehyksellä toteutetut sovellukset ovat teknisen toteutustavan vuoksi suorituskyvyltään heikompia kuin vastaavat natiivit sovellukset. Moderneilla laitteilla sovelluksen suorituskyky voi kuitenkin olla lähellä natiivia. Jos tarkoituksena on kehittää mobiilipeli, tai mikä tahansa animaatioita runsaasti hyödyntävä sovellus, kannattaa suosiolla kääntyä natiivisovelluksen puoleen. (Rust-Smith.) Ongelmat ilmenevät tyypillisesti monimutkaisissa CSS-elementeissä tai animaatioissa. Sovelluksen huono suorituskyky saattaa lisäksi alentaa sen käyttökokemusta ja vaikuttaa sovelluksesta annettuihin arvioihin. Huonot arviot taas vaikuttavat sovelluksen menestymiseen.

Koska PhoneGap toimii webview näkymän päällä, on kehyksellä myös käytännössä mahdotonta tehdä widgettejä Android-alustalle (Kuva 5). Widgetit ovat Androidin natiivikoodia, jotka extendoidaan näkymän päälle. Jos olisi mahdollista tehdä widgetti natiivikoodilla, joka sisältäisi webview näkymän, voisi PhoneGapilla teoriassa tehdä myös widgettejä. Näin ei kuitenkaan ole tehty, joten widgetin rakentaminen on toistaiseksi natiivikoodin varassa. (Cordova.apache.org/docs.)

Jos sovelluskehitystä tehtäisiin iOS ja Android -laitteille, täytyisi hallita kahta eri koodikirjastoa. Tämä tarkoittaa sitä, että jokainen bugi, eli softavirhe, jokainen lisätty toiminto ja joka ikinen muutos tulisi tehdä kahdesti. PhoneGapilla on vain yksi koodikirjasto, tämä tekee sovelluksen hallinnasta helpompaa. PhoneGapilla

sovelluskehitystä tehdään JavaScriptillä, joka on nopeampaa ja helpompaa kuin esimerkiksi Java, tai vaikkapa Objective-C. PhoneGapilla ulkoasun luominen on myös helppoa. Tarvitsee vain etsiä sopiva tyylitiedosto ja ulkoasu on lähes valmis. Valmiita mobiilikirjastoja onkin saatavilla useampia, esimerkiksi jQuery Mobile. Lisäksi JavaScript ohjelmoijia on helpompi löytää, kuin esimerkiksi Objective-C osaajia. (Rust-Smith.)

2.4.6 PhoneGapin laajennukset

PhoneGapin laajennukset tai lisäosat eli pluginit tekevät sovelluskehyksen käytöstä mielenkiintoista. Pluginit ovat lisättyä koodia, jolla Cordovan webview-näkymä voi kommunikoida laitteen natiivien funktioiden kanssa. Pluginit mahdollistavat toimintojen käyttämisen, mitkä eivät normaalisti web-näkymässä ole mahdollisia. Kaikki Cordovan API-toiminnot ovat plugineja. Lisäksi on saatavilla monia muita plugineja, jotka mahdollistavat esimerkiksi viivakoodien skannaamisen, tai NFC lähilukemisen. Osoitteessa http://plugins.cordova.io/ on rekisteri saatavilla olevista plugineista. Nämä lisäosat käytännössä piilottavat natiivit koodit JavaScript kehyksen taakse. (Cordova.apache.org/docs.)

Pluginien käyttö tapahtuu kätevästi komennoilla:

$ cordova plugin add org.apache.cordova.media
$ cordova plugin ls
['org.apache.cordova.media']
$ cordova plugin rm org.apache.cordova.media

Joista ensimmäisessä komennossa lisätään plugin nimeltä media, joka mahdollistaa äänitiedostojen toistamisen ja nauhoittamisen laitteella. ls-komento listaa kaikki lisätyt pluginit ja rm taas poistaa halutun lisäosan. (Cordova.apache.org/docs.)

2.5 Git ja GitHub

Git on hajautettu versionhallintajärjestelmä. Gitissä on painotettu nopeutta, sekä tietojen eheyttä. Gitin on alun perin kehittänyt Linus Torvalds, Linuxin kernelin kehittäjä. Git on nykyään yksi käytetyimmistä työkaluista ohjelmistokehityksessä. Git mahdollistaa useiden ihmisten työskentelyn samanaikaisesti samoilla tiedostoilla käyttöjärjestelmästä riippumatta. (Finley 2012.) Tässä projektissa Git-työkalua käytetään sovelluksen versionhallintaan.

GitHub on koodin jakamiseen ja julkaisuun tarkoitettu palvelu. GitHub on Git repositorion hosting-palvelu. GitHubilla on lisäksi monia omia työkaluja. Esimerkiksi kun Git-työkalua käytetään ainoastaan komentokehoitteesta, on GitHubilla saatavilla graafinen työkalu. Lisäksi GitHub mahdollistaa koodien selaamisen webin kautta. Palveluun voi perustaa esimerkiksi omat wikisivut. (Finley 2012.)

2.6 Laitealustat iOS, Android ja Windows Phone

iOS on Applen ja Windows Phone Microsoftin kehittämä käyttöjärjestelmä. Kehitysprosessi kaikille alustoille on PhoneGapilla sama, mutta jokainen käyttöjärjestelmä vaatii oman SDK:n, jotta sovellus voitaisiin paketoida ja julkaista oikeassa sovelluskaupassa. Lisäksi sovelluskauppojen toiminta eroaa toisistaan ja sovellusten hyväksymiskriteerit ovat erilaiset. Esimerkiksi Applen App Storessa julkaistava sovellus käy läpi tarkan prosessin, jossa sovelluksen julkaisukelpoisuus tarkistetaan. Sovelluksen julkaiseminen kaupassa voi siis kestää useamman päivän. Tieto perustuu kirjoittajan omiin kokemuksiin.

Googlen kaupassa sovellus tulee puolestaan automaattisesti esille muutamissa tunneissa ja sovellus tarkistetaan jälkikäteen. Sovelluskauppojen rekisteröintimaksu on myös erilainen. Esimerkiksi iOS sovelluskehittäjä joutuu maksamaan 99 dollaria vuodessa pystyäkseen julkaisemaan sovelluksia. Apple tarjoaa myös ”Enterprise” kehittäjävaihtoehtoa 299 dollarilla. (Developer.apple.com. iOS.) Android kehittäjän lompakko kevenee 25 dollaria, jolla saa ikuisen lisenssin. Microsoftin kehittäjälisenssi maksaa 19 dollaria, joka voi vaihdella maakohtaisesti. Yrityslisenssi maksaa Microsoftilla 99 dollaria. (msdn.microsoft.com.)

2.7 DevOps mobiilisovelluskehityksessä

DevOps on puunhakkaajan moottorisaha. DevOpsin tavoitteena on automatisoida tuotteiden paketointi-, laadunvarmistus- ja julkaisuprosesseja. DevOps lainaa konsepteja ketteristä menetelmistä ja yrittää nitoa yhteen kehittäjät, ylläpitäjät ja asiakkaat. DevOps nojaa automaatioon, virtuaalisatioon ja fiksuihin työkaluvalintoihin. Tylsästä työstä tulee robotin hommaa ja ihminen saa tehtäväkseen ohjelmistotuotannon parhaan palan eli luovan ja asiakaskeskeisen työn (Eficode, DevOps-opas).

Kuvitellaan vaikkapa suuren yrityksen mobiilisovellusta, joka päivittyisi jatkuvasti. Tällöin DevOps voisi toimia esimerkiksi niin, että jokaisen muutetun koodirivin jälkeen ohjelma ajettaisiin automaattisesti usealle eri emulaattorille, jotka olisi asetettu toimimaan erilaisilla laitteistoasetuksilla. Vaikkapa kaikille kolmelle alustalle, Windows, Iphone ja Android

virtuaalipuhelimille, joita olisi vielä useampia erilaisilla asetuksilla ja resoluutioilla. Seuraavaksi etukäteen suunnitellut testit ajettaisiin jokaiselle emulaattorille, jolloin nähtäisiin suoraan ohjelmistovirheet. DevOps voi auttaa automatisoimaan esimerkiksi laadunvarmistuksen. DevOpsia voi hyödyntää mobiilisovelluskehityksessä myös käyttämällä pilvipalveluita, jolloin säästytään yhdeltä työvaiheelta, kun SDK pakettien hallinta on pilvessä. Työskentely on työasemariippumatonta ja kehitystyö voidaan aloittaa käytännössä heti.

2.8 Kirjastot ja projektin keskeiset tekniikat

Kirjastot ovat resursseja, joita voidaan hyödyntää ohjelmistokehityksessä. Kirjastot ovat valmista koodia, jotka nopeuttavat sovelluskehittäjän työtä. Tyypillisesti ohjelmistokehittäjä lisää kirjaston sovellukseen saadakseen lisää toimintoja tai automatisoidakseen toistuvia prosesseja. Esimerkiksi kehitettäessä laskinsovellusta, voidaan käyttää valmiita matematiikka-kirjastoja, jolloin kehittäjän ei itse tarvitse miettiä, kuinka laskut lasketaan. Valmiit kirjastot, tai niin sanotut kehykset, voivat helpottaa sovelluksen käyttämistä ja nopeuttaa kehittäjän työtä. (Janssen.) Käydään lävitse tämän työn esimerkkisovelluksen käyttämiä kirjastoja. Osa tiedoista perustuu työn aikana saatuihin kokemuksiin.

jQuery on nopea, pieni ja täynnä toimintoja oleva JavaScript-kirjasto. jQueryllä voidaan manipuloida HTML-elementtejä ja sen tarkoitus on tehdä JavaScriptin käytöstä helpompaa. jQuery tekee paljon toimintoja muutamalla komennolla, jotka normaalisti vaatisivat usean rivin koodaamisen. (jQuery.com.) On olemassa paljon erilaisia JavaScript kirjastoja, mutta jQuery on yksi suosituimmista ja myös parhaiten laajennettavissa. jQuery on ydinkirjasto jQuery-perheessä. (W3Schools. jQuery intro.) PhoneGapilla työskennellään pääasiassa JavaScriptillä, siksi jQuery on valittu tähän projektiin.

jQuery Mobile on HTML5-kieleen perustuva käyttöliittymäkokonaisuus, jolla voidaan luoda responsiivisia verkkosivuja ja sovelluksia, jotka toimivat kaikilla laitteilla: tableteilla, matkapuhelimilla ja tietokoneilla. jQuery Mobile perustuu jQueryn ydinkirjastoon ja jQuery UI kirjastoihin. Esimerkkisovelluksen käyttöliittymä pohjautuu jQuery Mobileen. (Stackoverflow keskustelufoorumi. jQuery, jQuery Mobile, jQuery UI.)

Swipe.js kirjastossa luodaan wrapper div-elementin sisälle uusia div-elementtejä, joiden sisältönä on esimerkiksi kuva, tai tekstiä. Swipe.js piilottaa muut elementit ja näyttää niitä järjestyksessä riippuen liu'utetaanko kuvaa vasemmalle tai oikealle. Swipe.js kirjasto automatisoi kuvakarusellin, jota käyttäjä voi ohjata halunsa mukaan.

Shake.css lisää elementteihin CSS-animaation, joka näyttää tärisyttävän elementtiä. Animaatio on toteutettu ”keyframeilla” eli ns. muistipisteillä. Kuvan jokainen liike on kirjoitettu koodiin, jota toistetaan tietyllä nopeudella tietyssä järjestyksessä. Animaatiot tulivat CSS3-versiossa (Wikikirjasto. CSS3).

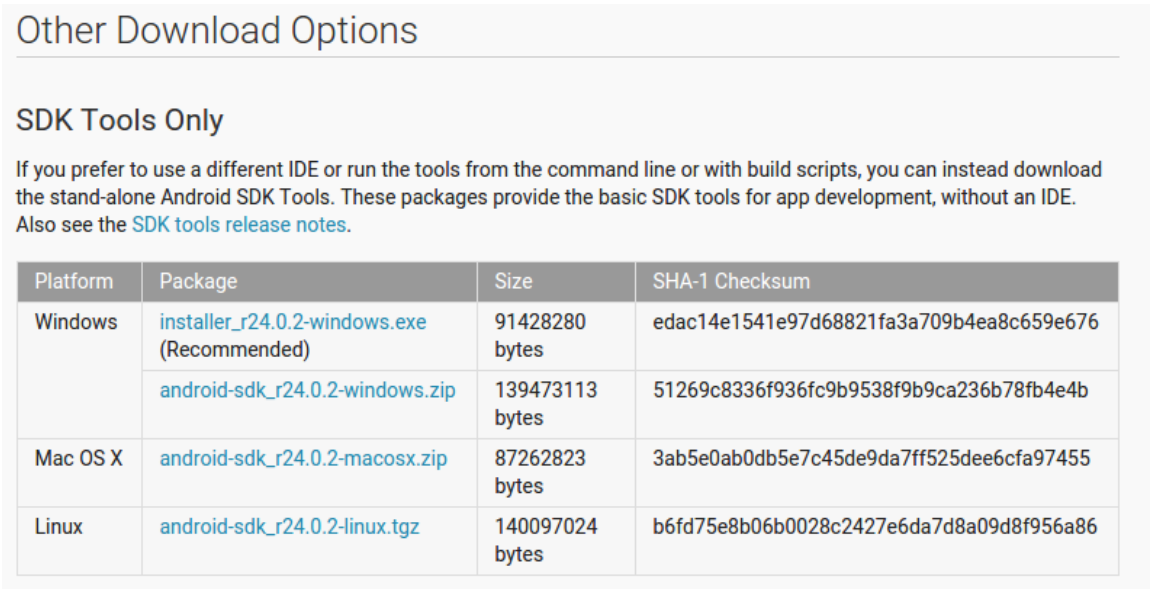
3 Asennusohje Android ympäristölle

Tässä kappaleessa asennetaan tarvittavat työkalut Linux-käyttöjärjestelmälle ja asetetaan projekti GitHubiin versionhallintaa varten. Gitin käyttöön tutustutaan tarkemmin alaotsikossa 3.4. Ensiksi asennetaan Android SDK, sekä virtuaalipuhelin, jonka jälkeen vasta itse PhoneGap. Kun PhoneGapilla on luotu HelloWorld -sovellus, voidaan se laittaa GitHubiin, jonka jälkeen päästään aloittamaan varsinainen projekti. Huomioi, että Android emulaattori ei tue 32-bittistä käyttöjärjestelmää, vaan on käytettävä aina 64-bittistä ongelmien välttämiseksi (Askubuntu.com 2013. Ubuntun keskustelufoorumi, 32 bittinen Android emulaattori). Tässä ohjeessa käytetyn käyttöjärjestelmän versio on Xubuntu 14.04.2 LTS 64-bit. PhoneGap versio 4.2.0-0.24.2 ja Android SDK Manager versio 24.1.2.

3.1 Android SDK ja virtuaalipuhelin

Lataa uusin versio Android SDK:sta osoitteesta:
https://developer.android.com/sdk/index.html#Other

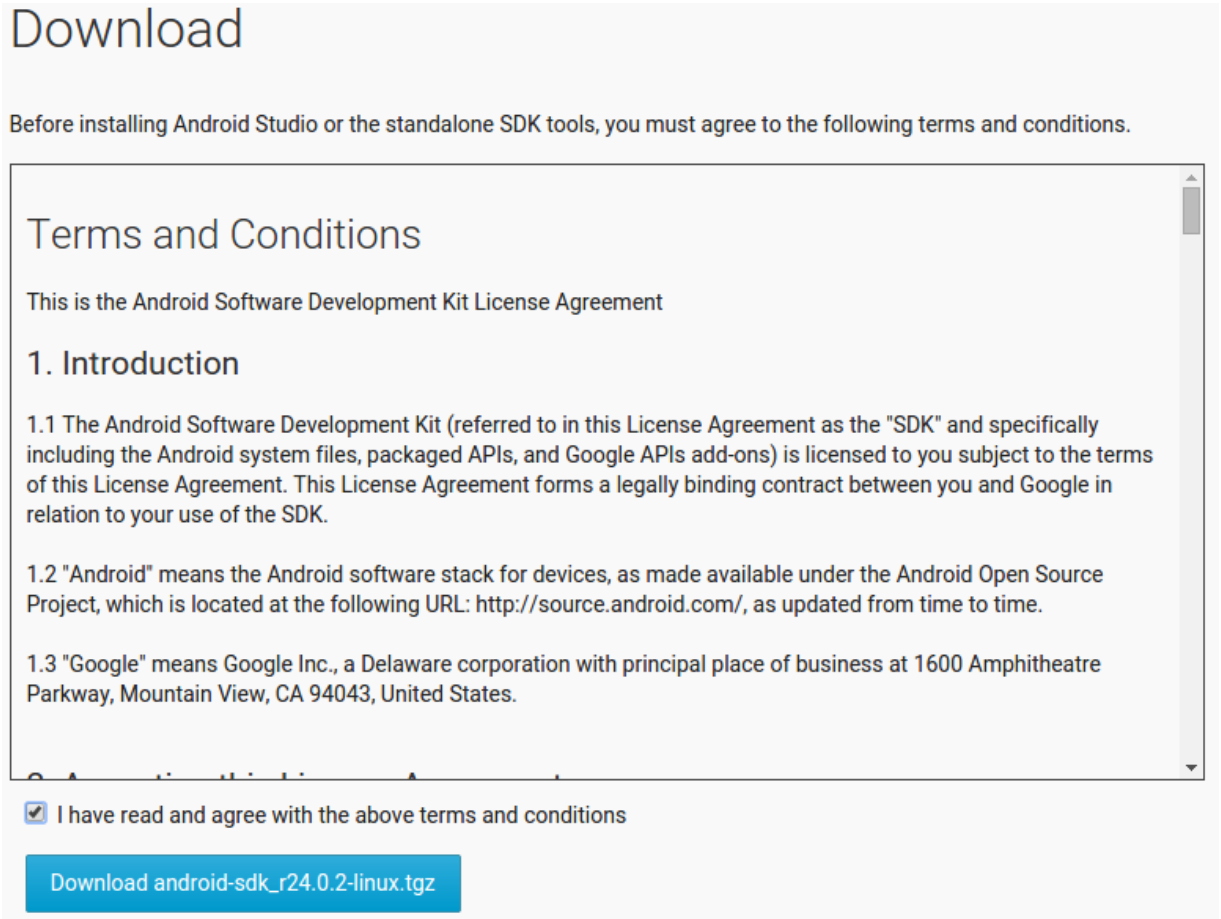
Valitse Linux paketti kohdasta “SDK Tools Only” (Kuva 6).
android-sdk_r24.0.2-linux.tgz



Kuva 6. Kuvakaappaus developer.android.com

Vaihtoehtona pelkälle SDK:lle on valita Android studio, joka sisältää mm. IntelliJ Idean IDE:seen perustuvan community edition editorin. Ladataan pelkkä SDK. (SDK = Software development kit) (Askubuntu.com 2013. Ubuntun keskustelufoorumi, Android SDK asennus.)

Laita ruksi valintaruutuun, jossa pyydetään hyväksymään käyttöehdot (Kuva 7).



Kuva 7. Kuvakaappaus developer.android.com

Android SDK:n vaatimukset ovat Linuxilla:

- GNOME tai KDE työpöytäympäristö
- GNU C kirjasto (glibc) 2.15 tai myöhempi
- 2 GB RAM minimi, 4 GB RAM suositeltu
- 400 MB kovalevytilaa
- Vähintään 1 GB Android SDK, emulaattorille, järjestelmäkuvalle ja välimuistille
- 1280 x 800 resoluutio, tai suurempi
- Oracle® Java kehittäjän paketti (JDK) 7

Tiedosto latautuu valittuun kansioon, oletuksena Downloads. Pura tiedosto käyttämällä komentoa:

$tar xvf android-sdk_r24.0.2-linux.tgz

Vaihtoehtoisesti tiedoston voi purkaa käyttämällä graafista ohjelmaa.

Tiedosto ladattuna työasemalle (Kuva 8):

Kuva 8. Android SDK ladattuna työasemalle.

Siirrä ladattu tiedosto sijaintiin /opt/ komennolla:

$sudo mv android-sdk-linux /opt/

OPT kansio on Linuxin tiedostorakenteessa tarkoitettu ohjelmistoille ja lisäosille, jotka eivät kuulu perusasennukseen. Kolmannen osapuolen ohjelmat kannattaa normaalisti asentaa /opt hakemistoon (Kuva 9).

```
shnigi@shnigi-ThinkPad-X1-Carbon:~/Downloads$ cd /opt
shnigi@shnigi-ThinkPad-X1-Carbon:/opt$ ls
android-sdk-linux  atom  google  vagrant
shnigi@shnigi-ThinkPad-X1-Carbon:/opt$
```

Kuva 9. SDK siirrettynä /opt hakemistoon.

Seuraavaksi navigoi /opt kansioon jonne Android SDK juuri siirrettiin. Mene tools hakemistoon ja aja komento: **$./android** (Kuva 10). (Ahlskog 2014.)

Tämä suorittaa Android SDK Managerin, jolla ladataan tarvittavat kehitystyökalut. Huomioi, että tässä vaiheessa tulee javan olla asennettuna koneelle. Jos et ole vielä asentanut javaa, saat sen suorittamalla komennon:

$sudo apt-get install openjdk-7-jre

```
shnigi@shnigi-ThinkPad-X1-Carbon:/opt$ cd android-sdk-linux/
shnigi@shnigi-ThinkPad-X1-Carbon:/opt/android-sdk-linux$ cd tools
shnigi@shnigi-ThinkPad-X1-Carbon:/opt/android-sdk-linux/tools$ ./android
```

Kuva 10. Android Managerin ajaminen.

Jotta Android kehitys voidaan aloittaa, tarvitsee minimissään ladata seuraavat:

- Android SDK Tools
- Android SDK Platform-tools

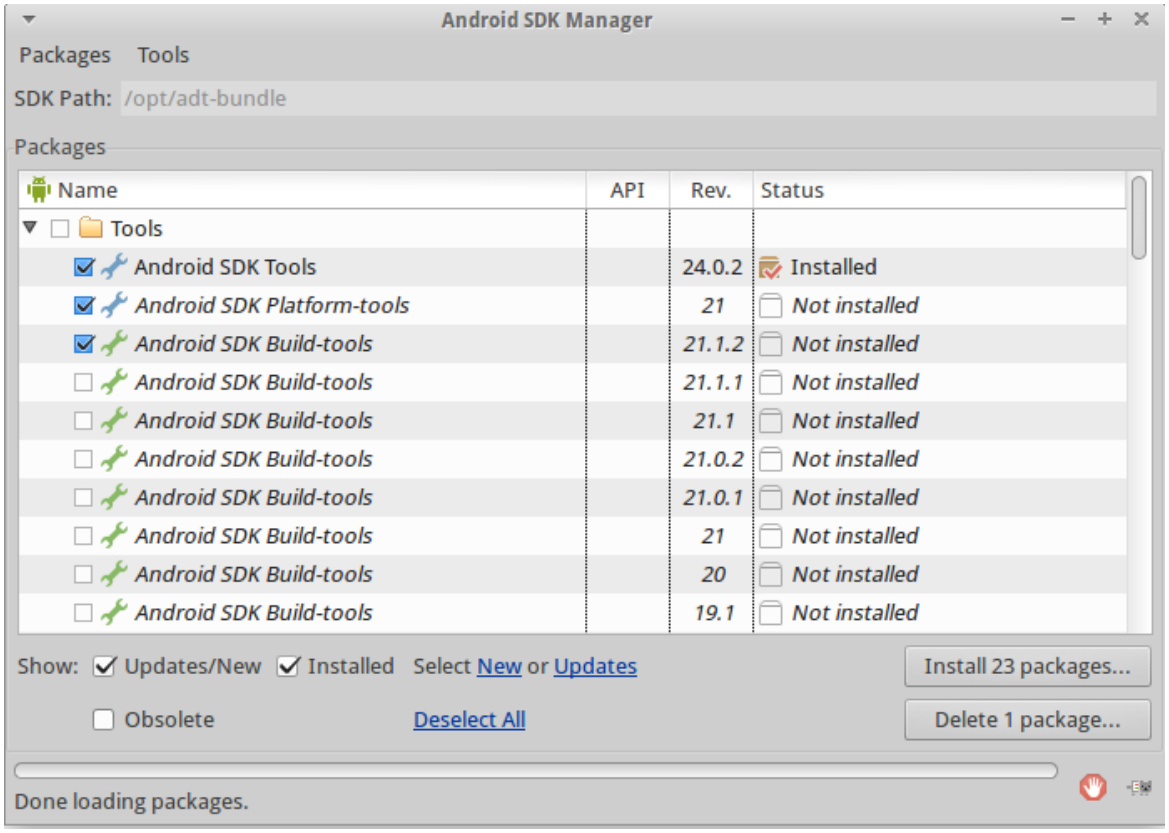
- Android SDK Build-tools (viimeisin versio)
- Android 5.0.1 (API 21):n kokonaisuudessaan

(Developer.android.com. SDK pakettien lisääminen.)

Tätä ohjetta tehtäessä API versio 19 toimi parhaiten PhoneGapin kanssa.

Lisäksi voidaan haluttaessa ladata esimerkiksi:

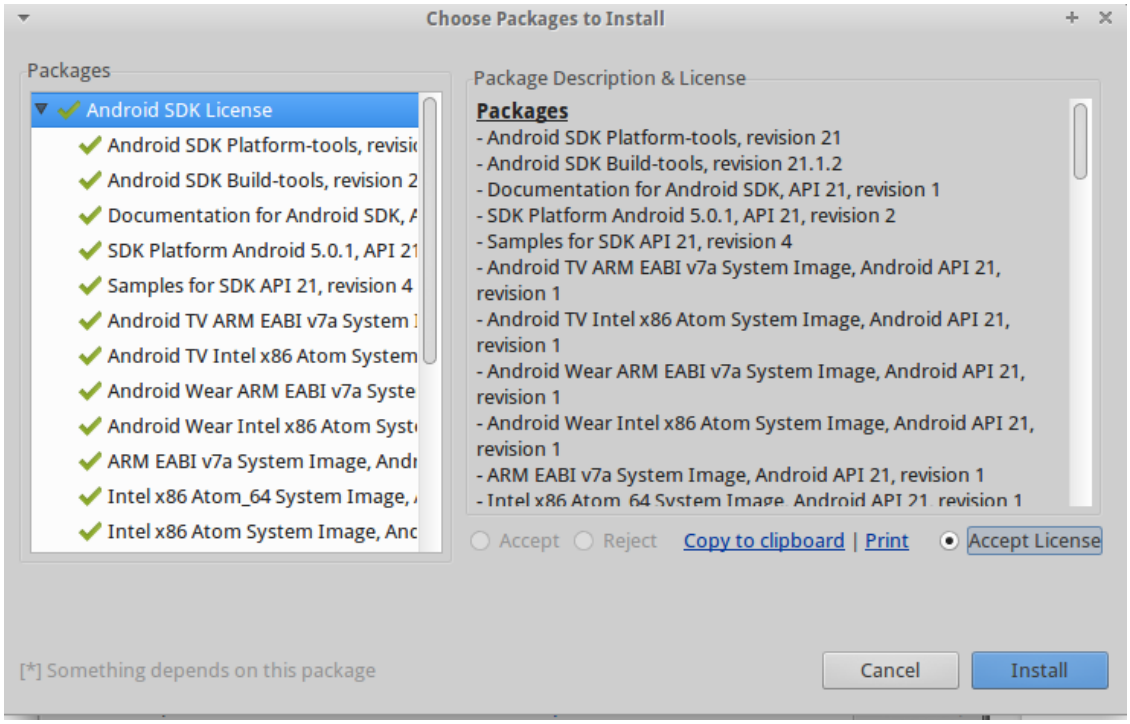
- Tools-kansio
- Extras-kansiosta: Android Support Repository, Google Play Services, Google Repository



Kuva 11. Android SDK Manager.

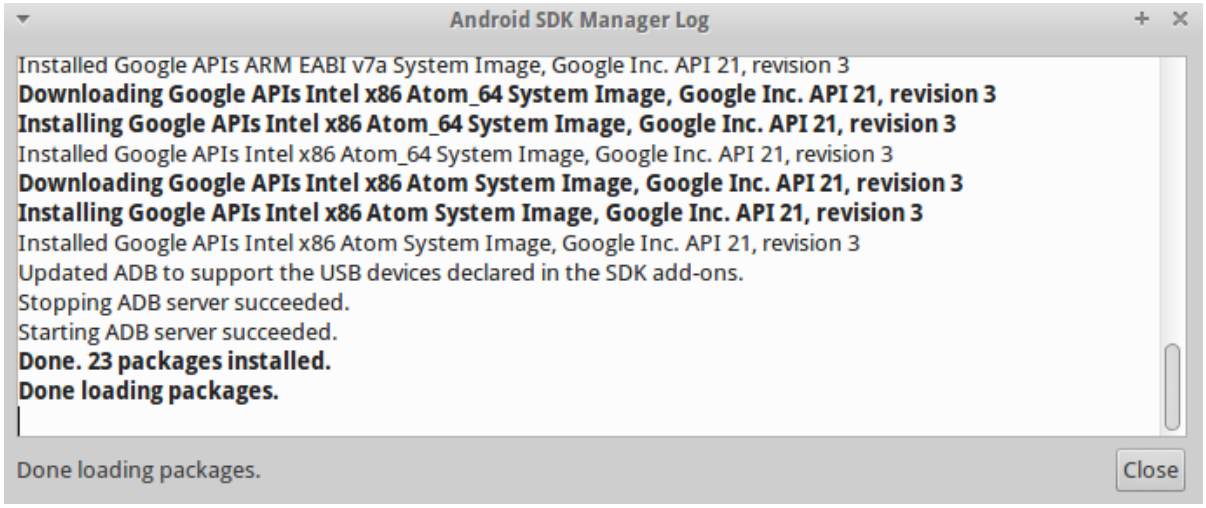
Valitse haluamasi työkalut ja klikkaa “Install” (Kuva 11).

Hyväksy käyttöehdot (Kuva 12).



Kuva 12. Pakettien asennuslisenssien hyväksyminen.

Odota latauksen valmistumista (Kuva 13).



Kuva 13. Pakettien lataaminen.

Seuraavaksi editoidaan tiedostoa ".bashrc". Bashrc on tiedosto, joka sisältää komentotulkin komentoja. Tiedostoa käytetään tyypillisesti muokkaamaan ympäristömuuttujia ja määrittämään komentotulkin toimintoja. Bashrc-tiedosto voi sijaita esimerkiksi käyttäjän kotihakemistossa piilotettuna, josta se ajetaan, kun käyttäjä avaa komentotulkin. (SuperUser keskustelufoorumi. Bashrc tiedosto.)

Avaa tiedosto bashrc kotihakemistossasi komennolla:

$nano ~/.bashrc

Lisää seuraavat komennot joko tiedoston alkuun tai loppuun (Kuva 14). Tällä ei ole väliä.

export PATH=${PATH}:~/android-sdk-linux/tools export PATH=${PATH}:~/android-sdk-linux/platform-tools

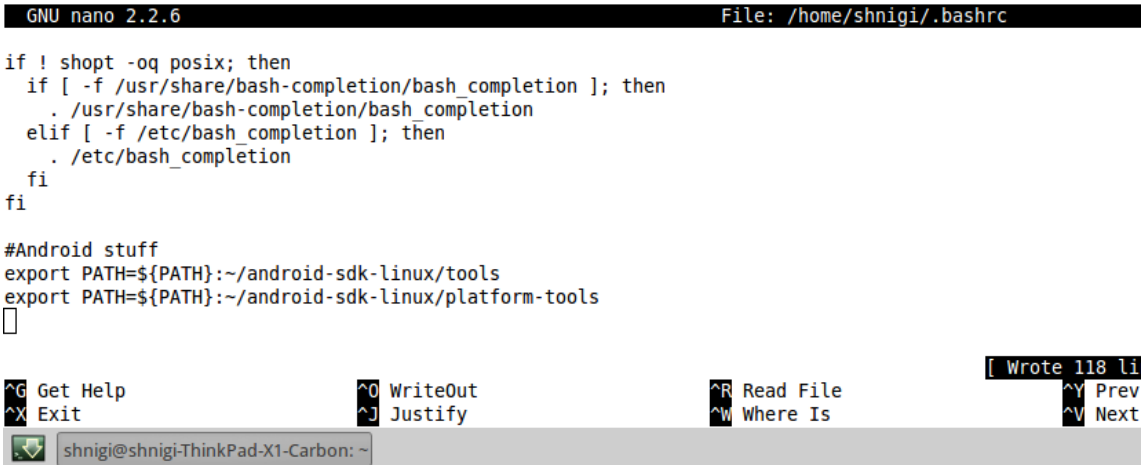
Vaihtoehtoinen tapa merkitä polku on käyttää muuttujaa ANDROID_HOME:

export ANDROID_HOME=/opt/ android-sdk-linux/

export PATH=$ANDROID_HOME/tools:$PATH

export PATH=$ANDROID_HOME/platform-tools:$PATH

Voit lisäksi laittaa kommentin ”#Android stuff” muistuttamaan mitä varten komennot on lisätty tiedostoon. # merkki tarkoittaa kommenttia, joka jätetään välistä tiedostoa ajettaessa. (Ahlskog 2010.)



Kuva 14. Bashrc-tiedoston käsittely.

Paina CTRL + O tallentaaksesi tiedoston, mutta älä sulje sitä vielä. Seuraavaksi muutetaan tilde-merkin kohtaan ”~” -polku, jonne Android SDK on siirretty, eli /opt (Kuva 15).

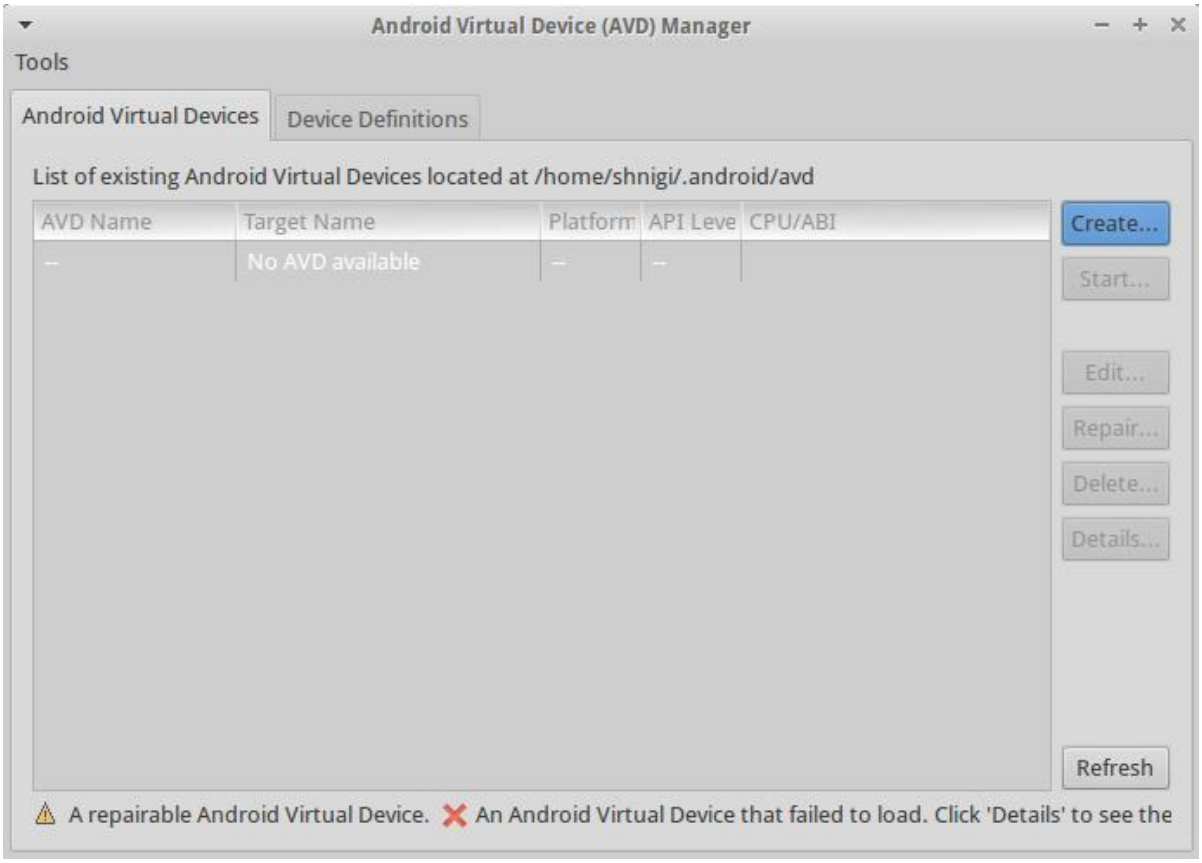
```
#Android stuff
export PATH=${PATH}:/opt/android-sdk-linux/tools
export PATH=${PATH}:/opt/android-sdk-linux/platform-tools
```

Kuva 15. Android SDK polun asettaminen.

Tallenna tiedosto uudestaan painamalla CTRL + O ja sulje tiedosto painamalla CTRL + X Seuraavaksi kirjaudu ulos järjestelmästä ja takaisin sisään, jotta ympäristömuuttujat aktivoituvat.

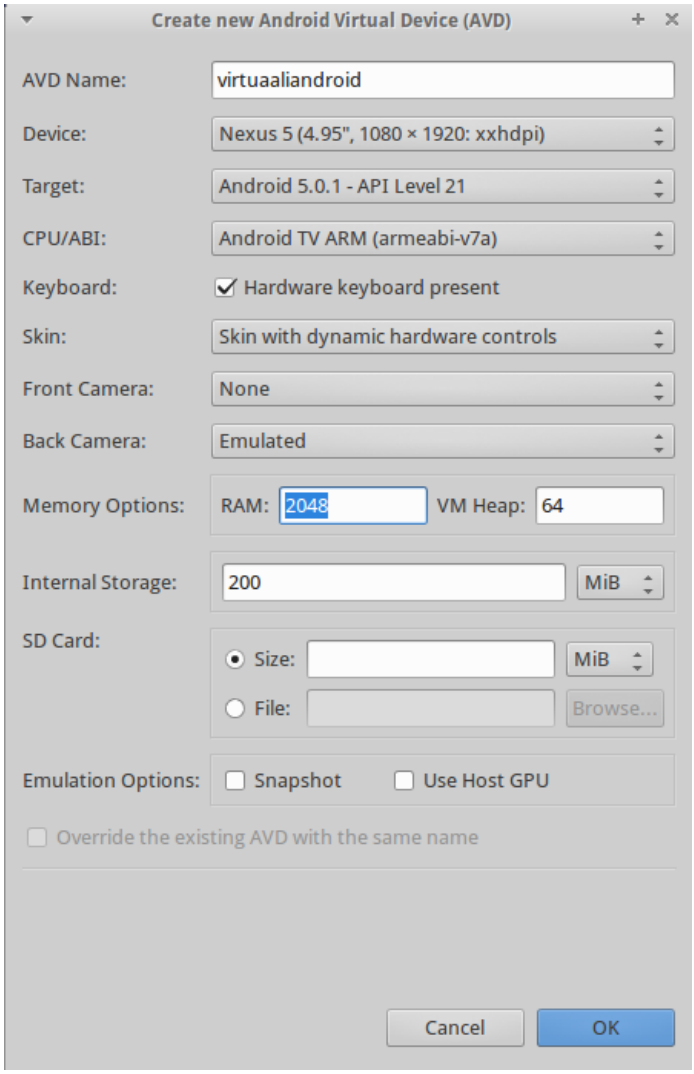
Jotta koneesi pystyy ajamaan virtuaalista Android puhelinta, tulee seuraavat paketit olla asennettuna, vaikka kyseiset paketit ovatkin 32-bittiselle järjestelmälle tarkoitettuja. Asenna paketit komennolla:
sudo apt-get install libstdc++6:i386 libgcc1:i386 zlib1g:i386 libncurses5:i386
(Ajatuksella.com 2014.)

Nyt .bashrc komentosi tulisi toimia kun ajat komentotulkin. Anna komento:
$android avd
Jos onnistuit tekemään kaiken oikein tähän asti, pitäisi Android-virtuaalilaiteikkunan avautua. Paina “Create” luodaksesi uuden virtuaalisen Android laitteen (Kuva 16).



Kuva 16. Android Virtual Device Manager.

Nimeä uusi virtuaalinen Android laitteesi ja aseta sille sopivat laiteominaisuudet, esimerkiksi seuraavan kuvan mukaisesti (Kuva 17). Huomioi kuitenkin RAM-muistin määrä, kuinka paljon pystyt omasta koneestasi jakamaan virtuaalilaitteelle. Jos käytettävissäsi on esimerkiksi 8gb muistia, voit hyvin antaa virtuaalipuhelimelle 2gb. Jos käytössäsi on vain 4gb, kannattaa laittaa esimerkiksi 512, tai 1024mb. (Code.tutsplus.com 2010. Android virtuaalipuhelimen asetukset.)



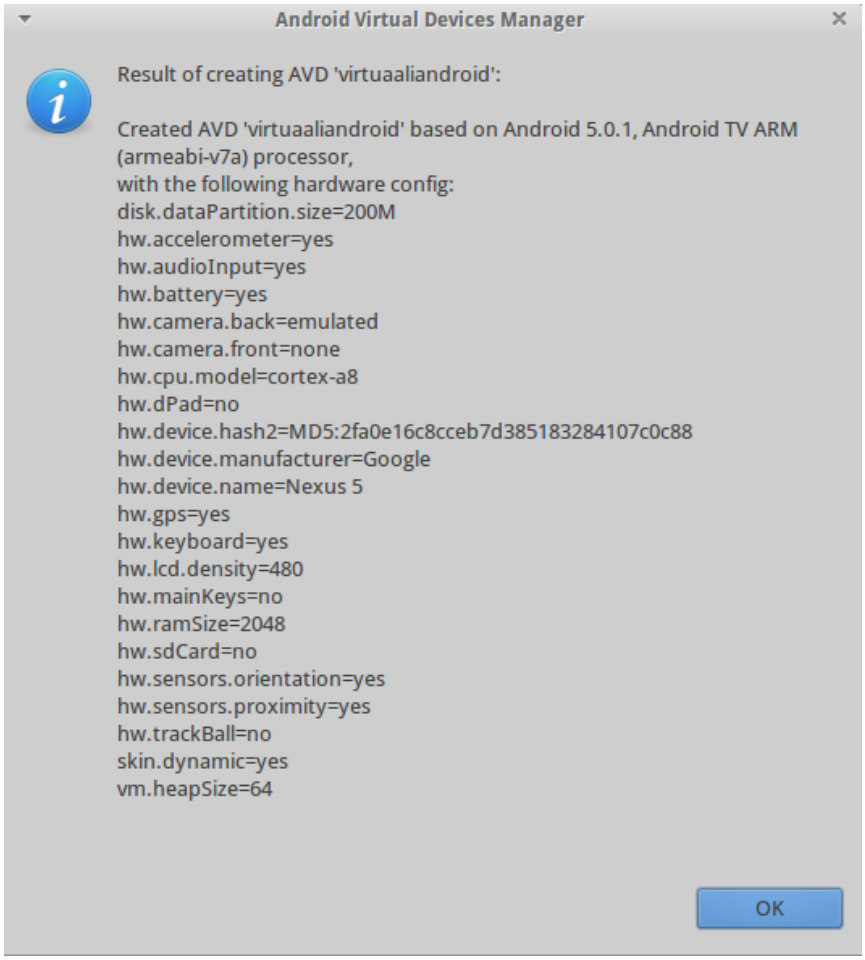
Kuva 17. AVD:n asetukset.

Virtuaalipuhelimen asetuksissa Skin tarkoittaa emulaattorin ikkunan kokoa. Skiniä kannattaa säätää käytettäessä esimerkiksi tietokonetta, jossa ei ole full-hd näyttöä. Ikkunan koko ei ole tekemisissä laitteen resoluution kanssa. (Stackoverflow keskustelufoorumi. Emulaattori samoilla asetuksilla, kuin fyysinen puhelin.) Muita säädettäviä ominaisuuksia virtuaalipuhelimelle ovat esimerkiksi etu- ja takakamera, laitteen sisäinen muisti, muistikortin koko ja prosessoriarkkitehtuuri. Lisäksi voidaan valita halutaanko käyttää fyysisiä näppäimiä. (Developer.android.com. Emulaattorin käyttäminen.) Prosessoriarkkitehtuuria muuttamalla voi aktivoida laitteistokiihdytyksen. Jotta laitteistokiihdytyksen saa käyttöön, tulee asentaa ensin tietokoneelle tarvittavat paketit Android managerista.

Kuva 18. Kuvakaappaus: monta erilaista virtuaalipuhelinta samalla työasemalla. Lähde: Code.tutplus.com

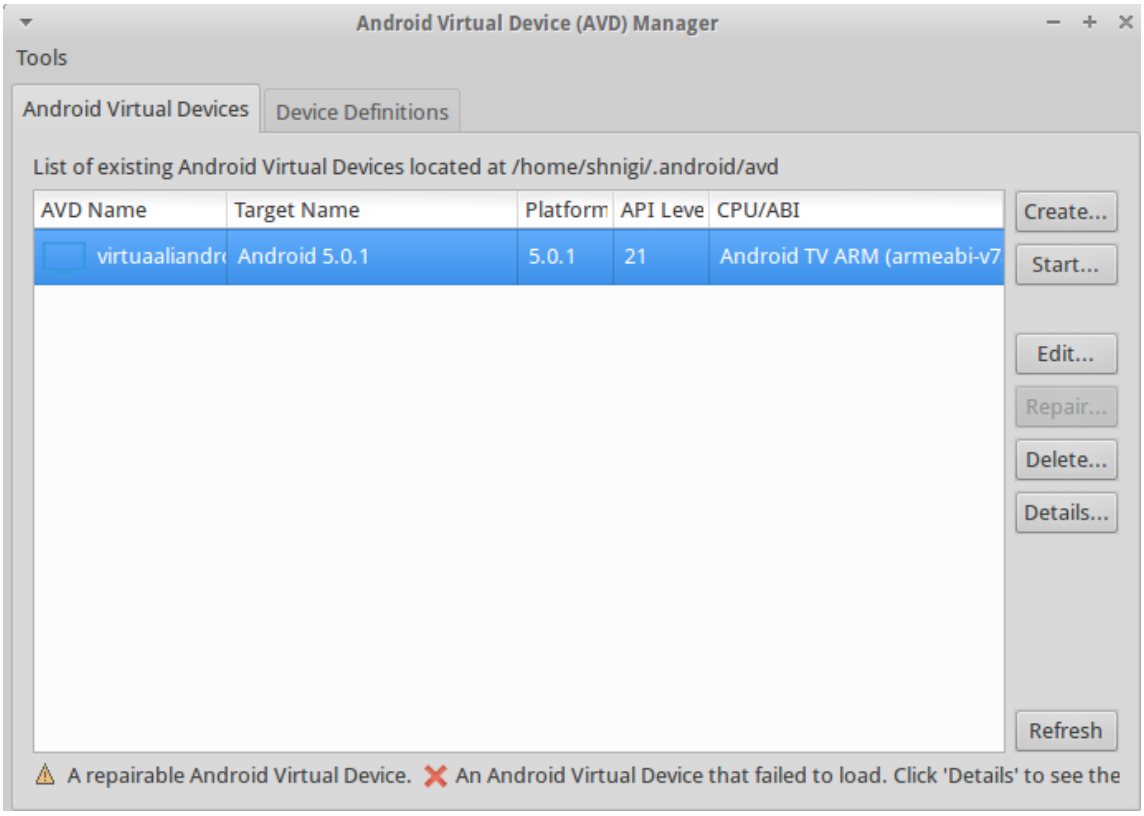
Virtuaalipuhelimia voi luoda useita erilaisia erilaisilla laitteistoasetuksilla (Kuva 18). Sovellusta on helppoa testata virtuaalipuhelimilla ja niitä voikin luoda halutessaan valmiiksi muutaman erilaisen. (Developer.android.com. Laitteet.)

Paina OK ja seuraavaan ilmoitukseen myös OK. Virtuaalipuhelimesi on valmis käytettäväksi (Kuva 19).



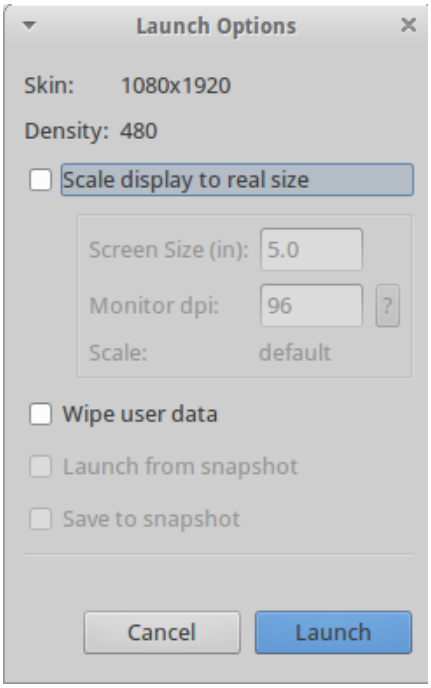
Kuva 19. Virtuaalipuhelin valmiina käytettäväksi.

Asetusten jälkeen paina ”Start” (Kuva 20).



Kuva 20. Virtuaalipuhelimet managerin listassa.

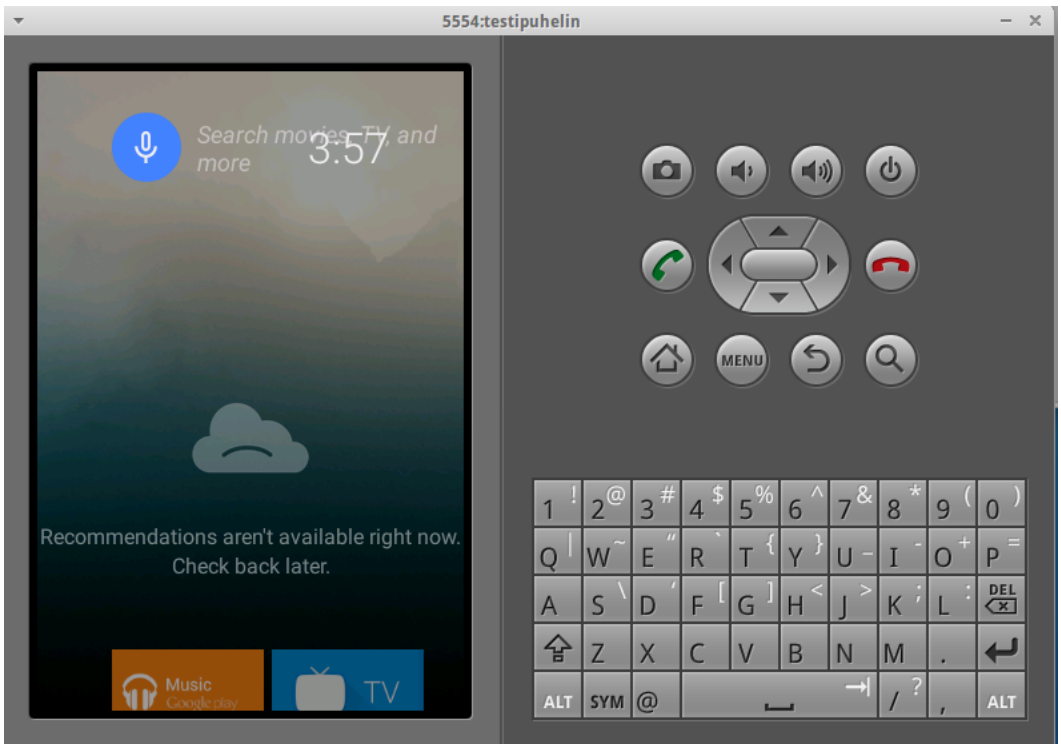
Paina Launch (Kuva 21).



Kuva 21. Virtuaalipuhelimen käynnistys.

Virtuaalipuhelin käynnistyy (Kuva 22). Huomioi että virtuaalikoneen ajaminen on melkoisen hidasta ja ensimmäinen käynnistys voi kestää jopa kymmenen minuuttia. Suositeltavaa onkin käyttää fyysistä puhelinta sovelluksen kehityksessä ja

testauksessa. On myös suositeltavaa kokeilla omaa sovellusta mahdollisimman monella erilaisella virtuaalipuhelimella, jotta sovelluksen toimivuudesta erilaisilla laitteilla voitaisiin olla varmoja. Kun virtuaalipuhelimen on kerran saanut käyntiin, sitä ei tarvitse sulkea kesken sovelluskehityksen, vaan uusi sovellus voidaan aina ajaa jo käynnissä olevaan virtuaalipuhelimeen.

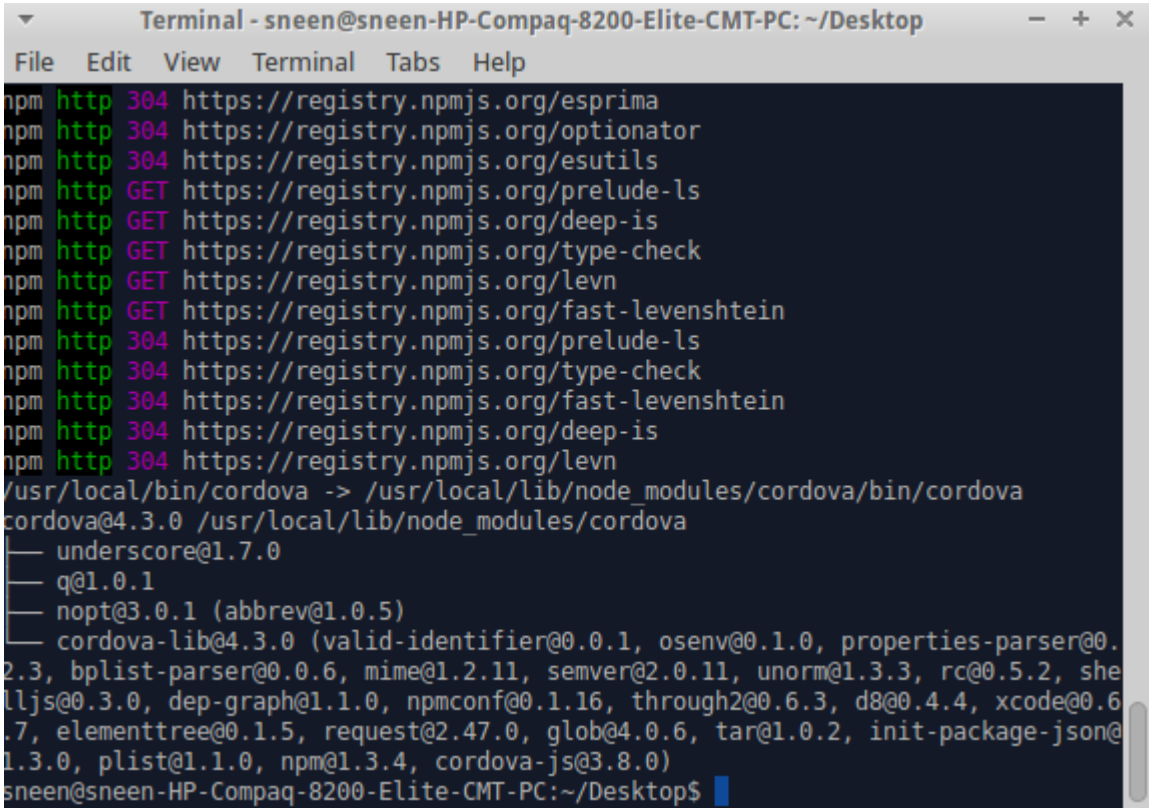


Kuva 22. Android virtuaalipuhelin API 22.

3.2 PhoneGap ja Cordova asennus

Asenna seuraavaksi koneellesi NPM ja NodeJS komennolla:
$sudo apt-get install npm ant nodejs nodejs-legacy
NPM on oletuksena Node.js versiossa 0.6.3 ja eteenpäin.
(Wikipedia)

Itse PhoneGapin ja Cordovan asennus käy kätevästi komennoilla:
$sudo npm install -g phonegap
$sudo npm install -g cordova



Kuva 23. PhoneGapin asennus.

Nyt sinulla pitäisi olla toimiva PhoneGap asennus Linux-työasemallasi, jos komentokehoitteeseen ei tulostunut virheilmoituksia (Kuva 23).

3.3 Projektin aloittaminen PhoneGapissa

Kun PhoneGap on saatu onnistuneesti asennettua, voidaan projektin kehittäminen aloittaa. Luo projektillesi uusi kansio komentokehoitteella ja siirry kyseiseen kansioon. PhoneGapilla on helppoa luoda testiprojekti komennolla:

$ **phonegap create helloworld**

Voit kokeilla toimiiko projektisi menemällä luotuun kansioon ”helloworld” ja ajamalla komennon

$ phonegap run android

Nyt koneen pitäisi automaattisesti käynnistää virtuaalipuhelin ja avata ”helloworld” -sovellus (Kuva 24).

Tarvittaessa alustan voi ensin generoida käyttämällä komentoa:

$phonegap build android, tai jos törmäät virheilmoitukseen käytä komentoa:

$cordova build android (Docs.phonegap.com. Komentorivin käyttöliittymä.)

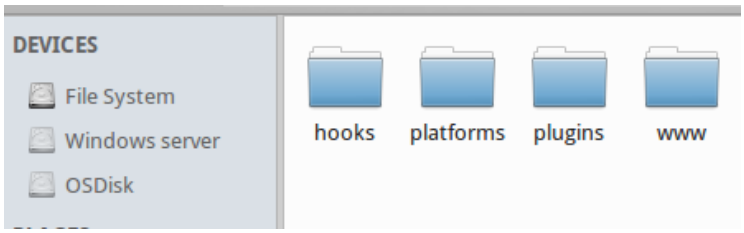
Kuva 24. PhoneGap HelloWorld ohjelma virtuaalipuhelimessa.

Oman projektin voit luoda käyttämällä seuraavaa komentoa:
$ phonegap create hello com.example.hello HelloWorld

(Createn jälkeen tuleva sana ”hello” tarkoittaa projektin kansion nimeä. Tätä kansiota ei saa olla ennestään olemassa. ”com.example.hello” on pakettisi nimi. Tämä tieto ei ole pakollinen, mutta on suositeltavaa täyttää se. Paketin nimeä voi muokata myöhemmin config.xml tiedostosta. ”HelloWorld” on sovelluksesi nimi.)

Tarkastellaan seuraavaksi PhoneGapin luomaa kansiorakennetta.

Varsinaisen projektikansion alta löytyy neljä kansiota (Kuva 25):



Kuva 25. PhoneGapin kansiorakenne.

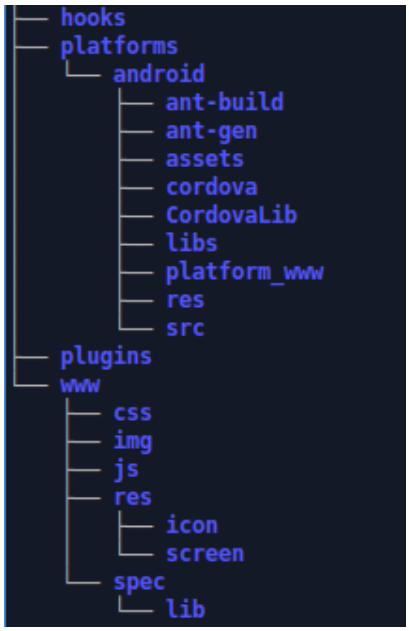
Hooks-kansio voi sisältää skriptejä, jotka muokkaavat cordovan komentoja. Tämä on kätevä, jos halutaan integroida omia build-työkaluja tai käyttää versionhallintaa. (Mmocny. GitHub.)

Platforms-kansiossa sijaitsevat eri alustoille tarvittavat työkalut projektin kääntämiseksi. Sen alla voi olla esimerkiksi Android / iOS / Windows kansiot. (Mmocny. GitHub.)

Plugins-kansioon asetetaan sovelluksen käyttämät lisäosat. (Mmocny. GitHub.)

WWW-kansiossa sijaitsevat sovelluksen resurssit, kuten HTML, CSS, JavaScript ja kuvatiedostot. Tässä kansiossa tullaan viettämään eniten aikaa sovelluskehityksen aikana. WWW-kansiossa sijaitsevassa config.xml tiedostossa on tarvittava metadata sovelluksen kääntämistä ja jakelua varten. Tiedostossa on mm. sovelluksen nimi, lyhyt kuvaus sovelluksesta, tekijän tiedot ja sovelluksen tarvitsemat luvat (esimerkiksi kamera, kiihtyvyysanturi jne.) (Gulati, 2013.)

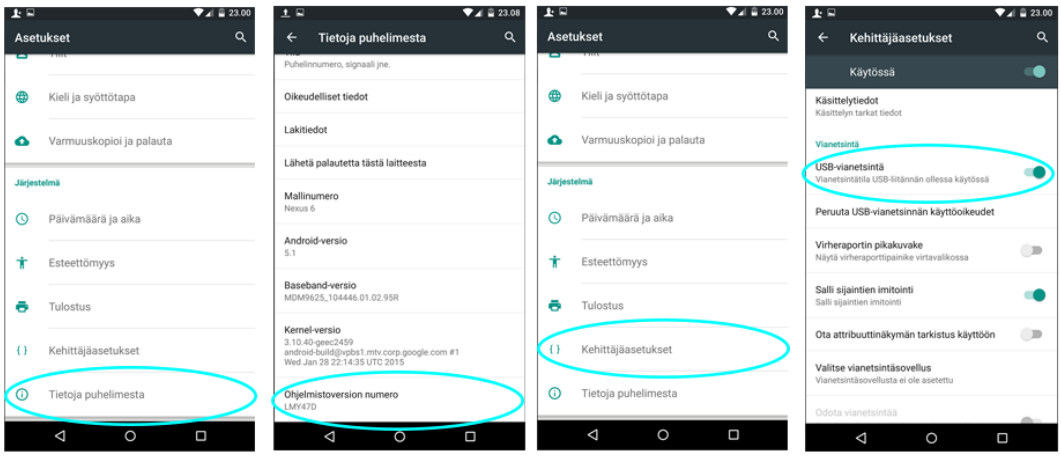
Puurakenteena tarkasteltuna PhoneGap-projekti näyttää seuraavalta (Kuva 26):



Kuva 26. PhoneGap puurakenteena Linuxin Tree -työkalulla tarkasteltuna.

Seuraavaksi otetaan käyttöön fyysinen Android-puhelin. Sovellusta on helpompi testata fyysisellä laitteella, kuin emulaattorilla. Saadaksesi PhoneGapin asentamaan sovellus fyysiselle puhelimelle, tulee luoda tiedosto 51-android.rules Linuxin asetushakemistoon. Luo tiedosto komennolla:
$sudoedit /etc/udev/rules.d/51-android.rules
Lisää tiedostoon seuraava rivi:
SUBSYSTEM=="usb", ATTR{idVendor}=="0bb4", MODE="0666", GROUP="plugdev"
Kohtaan idVendor lisätään oman puhelimesi valmistajan käyttämä nelinumeroinen koodi (0bb4), jonka voi katsoa osoitteesta: http://developer.android.com/tools/device.html
Kun olet lisännyt laitteesi valmistajan koodin, tallenna tiedosto ja anna komento:
$chmod a+r /etc/udev/rules.d/51-android.rules (Ajatuksella.com 2014.)

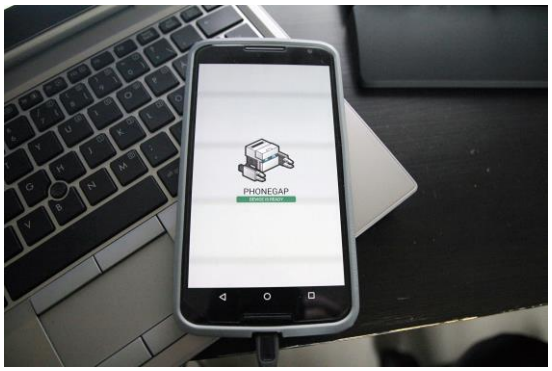
Nyt tietokoneella olevat asetukset ovat kunnossa. Seuraavaksi tulee puhelimesta kytkeä kehittäjätoiminnot päälle ja sallia USB-virheenkorjaus (Kuva 27). Menemällä puhelimen asetuksiin ja napauttamalla 7 kertaa ”versionumero” -kohtaa saadaan käyttöön kehittäjävaihtoehdot. Versionumero löytyy ”tietoja puhelimesta” -kohdan alta. (Kingoapp. USB Debugging toiminto.)



Kuva 27. USB-virheenkorjauksen ottaminen käyttöön fyysisessä puhelimessa.

Nyt puhelimen voi kytkeä tietokoneeseen USB-kaapelilla ja avata lukituksen. Ajamalla komennon **$phonegap run android** pitäisi sovelluksen asentua oletuksena puhelimeen (Kuva 29), kun puhelimesta on vielä hyväksytty yhteys tietokoneeseen (Kuva 28). Komennolla **$phonegap emulate android** voidaan sovellus ajaa virtuaalipuhelimeen, vaikka fyysinen puhelin olisi kytkettynä koneeseen.

Kuva 28. USB-virheenkorjauksen salliminen fyysisessä puhelimessa.



Kuva 29. PhoneGap HelloWorld asennettuna oikealle Nexus 6 puhelimelle.

3.4 GitHub

Seuraavaksi otetaan projektille käyttöön GitHub versionhallinta. Mene osoitteeseen https://github.com/ ja rekisteröidy palvelun käyttäjäksi. Klikkaa oikeassa yläreunassa käyttäjänimesi vieressä näkyvää plus-painiketta ja valitse ”create new repository” (Kuva 30).

Kuva 30. Kuvakaappaus GitHubin käyttäjävalikosta.

Seuraavaksi täytä kentät ”Repository name”, eli projektisi tai repositoriosi nimi. ”Description” kohtaan voit vapaamuotoisesti kuvailla projektiasi. Tätä ei ole pakko täyttää, mutta jos pidät repositoriosi julkisena, se tarjoaa muille käyttäjille tietoa mistä koodissa on kyse. Alempana määritellään onko repositorio julkinen, eli kaikille näkyvä, vai yksityinen, jolloin vain sinä ja valitsemasi henkilöt pääsevät käsiksi repositorioon. Tässä projektissa repositorio on julkinen, sillä yksityisestä repositoriosta GitHub perii maksun. Vaihtoehtoinen palvelu GitHubille on Bitbucket, joka tarjoaa päinvastaisen palvelun. Yksityiset repositoriot ovat ilmaisia ja julkiset maksullisia. Kummassakin palvelussa on puolensa ja niitä kannattaa vertailla projektista riippuen. (Freeman 2013.) Lopuksi klikkaa ”Create repository”.

Jotta GitHubiin voidaan työntää tavaraa, tarvitsee käyttäjän lisätä oma henkilökohtainen SSH-avain palveluun. SSH eli secure shell, on UNIX:iin perustuva protokolla, jolla voidaan turvallisesti ottaa yhteys etätietokoneeseen. SSH:ta käytetään laajalti palvelimien etähallintaan. Liikenteen molemmat päät ovat salattuja. Asiakkaan ja palvelimen yhteys on varmennettu digitaalisella sertifikaatilla ja salasanat ovat lisäksi salattuja. (Rouse 2005.)

Tarkista onko koneellasi jo luotuna SSH-avainta (Kuva 31) komennolla:

$ ls –al ~/.ssh

```
shnigi@shnigi-VirtualBox:~$ ls -al ~/.ssh
ls: cannot access /home/shnigi/.ssh: No such file or directory
```

Kuva 31. Olemassa olevan SSH-avaimen tarkistaminen.

Jos komennolla löytyy seuraavat tiedostot sinulla on SSH-avain valmiina ja voit siirtyä avaimen luomisvaiheen ylitse.

- id_dsa.pub
- id_ecdsa.pub
- id_ed25519.pub
- id_rsa.pub

SSH-avain generoidaan Linuxilla seuraavalla komennolla (Kuva 32):

$ ssh-keygen –t rsa –C ”omanimesi@sähköpostiosoitteesi.com”

```
shnigi@shnigi-VirtualBox:~$ ssh-keygen -t rsa -C "niki.ahlskog@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/shnigi/.ssh/id_rsa):
```

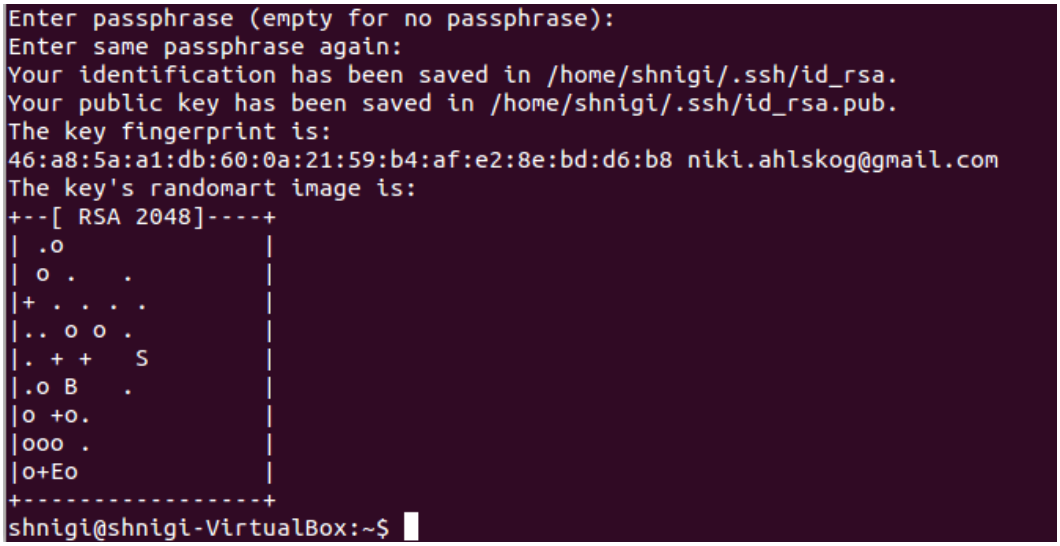
Kuva 32. SSH-avaimen generoiminen.

Paina “Enter” jatkaaksesi. SSH-avain tallentuu oletusarvoilla (Kuva 32). Seuraavaksi SSH haluaa avaimelle salasanan, jotta julkista ja yksityistä avainta voidaan vertailla keskenään (Kuva 33). Salasana tulee antaa kahteen kertaan.

```
Enter file in which to save the key (/home/shnigi/.ssh/id_rsa):
Created directory '/home/shnigi/.ssh'.
Enter passphrase (empty for no passphrase):
```

Kuva 33. SSH-avaimen salasanan asettaminen.

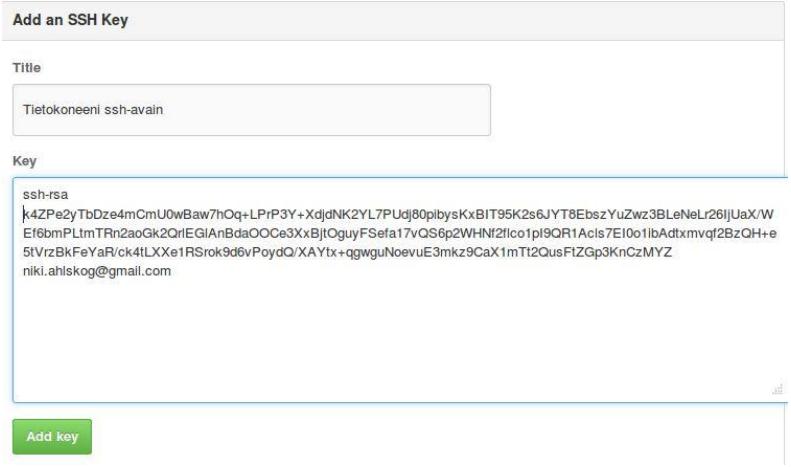
Tiedosto generoitui käyttäjän kotihakemistoon piilotettuun kansioon “.ssh”, jossa piste tarkoittaa piilotettua kansiota. Tiedoston olisi toki voinut myös luoda muualle.



Kuva 34. SSH-avain generoituna. Kuvio ilmoittaa avaimen olevan valmis.

Lopulta SSH-generoi käyttäjälle oman kuvion (Kuva 34). Tämä tarkoittaa, että avaimesi on luotu. Kuvion ainoa tarkoitus on saada avaimesta tunnistettava muoto ihmiselle. Kuviota voi käyttää esimerkiksi avainten validointiin (superuser.com keskustelufoorumi. SSH avaimen kuvio).

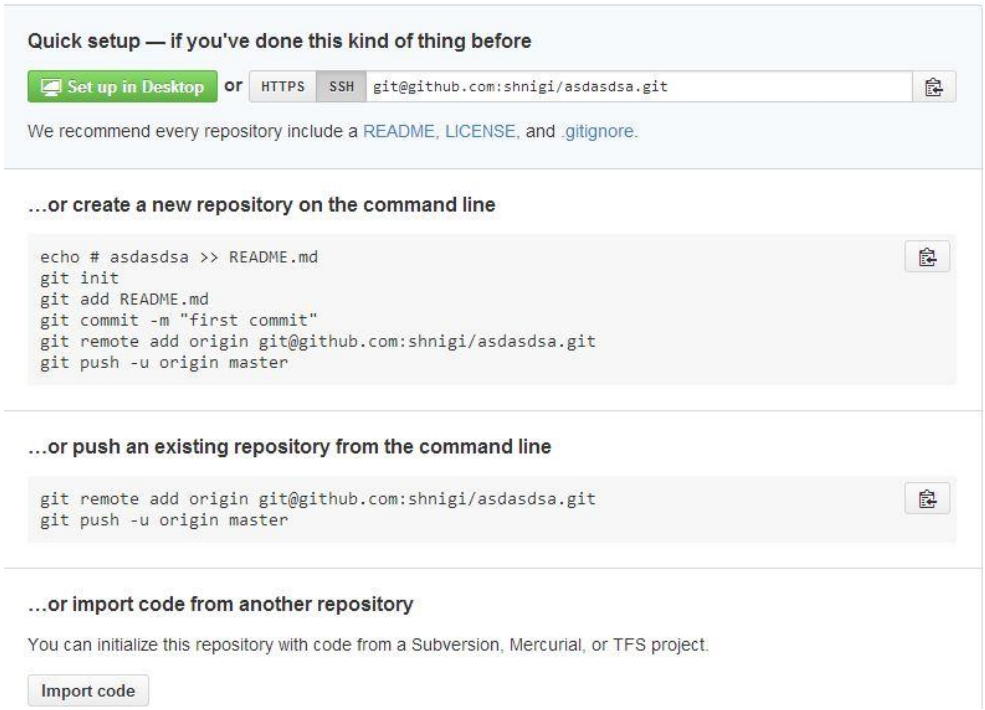
SSH avaimen kopiointia varten kannattaa asentaa työkalu nimeltä ”xclip” komennolla:
$sudo apt-get install xclip
xclip työkalulla on helppoa kopioida luotu SSH-avain ja lisätä se Githubiin. Komennolla:
$ xclip -sel clip < ~/.ssh/id_rsa.pub
saadaan kopioitua juuri luotu SSH-avain. SSH-avaimen voi myös kopioida millä tahansa tekstieditorilla. SSH-avain löytyy piiloitetusta ssh-kansiosta. Mene GitHubin asetuksiin kohtaan ”SSH keys” ja klikkaa ”Add SSH key” (Kuva 35). Liitä kopioimasi avain ja klikkaa ”Add key”.



Kuva 35. Kuvakaappaus GitHubista. SSH-avaimen lisääminen palveluun.

Saat GitHubilta sähköpostiisi ilmoituksen lisätystä SSH-avaimesta.

Kun olet lisännyt SSH-avaimen ja luonut uuden repositorion GitHubiin, voit navigoida aikaisemmin luodun PhoneGap projektin juureen, taikka www-kansioon, riippuen mitä kaikkea haluat sovelluksestasi laittaa versionhallintaan. Tässä projektissa versionhallinta otetaan käyttöön koko projektikansiolle ja platforms kansio asetetaan ohitustiedostoon. Ohitustiedostoa on käsitelty kappaleessa 3.5. Jos projektissa ei ole käytössä liitännäisiä, eikä hookkeja, riittää että versionhallinnassa on pelkkä www-kansio. Kun GitHubiin on luonut uuden repositorion, saat seuraavat ohjeet (Kuva 36):



Kuva 36. Kuvakaappaus uuden repositorion luomisesta GitHubissa. Lähde: GitHub.com

Mene projektissasi haluamaasi kansioon ja ajat komennot GitHubin ohjeiden mukaisesti, hieman muokaten:

$echo ”readme” >> README.md luodaan sovellukselle Readme-tiedosto. Hipsuissa oleva teksti tulee readme-tiedoston sisällöksi. Readme-tiedoston tekstit näkyvät repositorion infoteksteinä.

$git init initialisoidaan paikallinen Git-repositorio

$git add. lisätään kaikki tiedostot gitin seurantaan, ns. stage-vaihe. Tämä täytyy tehdä ennen kommitointia, jotta haluamasi tiedostot voidaan siirtää. Vaihtoehtoisesti saman ajaa komento **$git add -A**

$git commit –m ”ensimmäinen commit” tehdään ensimmäinen kommitointi Gitille paikalliseen repositorioon. Tiedostot eivät vielä siirry GitHubiin (Kuva 37). Tiedostot ovat

nyt siis paikallisella koneella Gitin hallinnassa, joka tarkoittaa sitä, että versionhallintaa voi tehdä myös yhdellä tietokoneella. Etä-repositorion käyttö mahdollistaa useamman käyttäjän työskentelyn samanaikaisesti ja projektin ylläpito helpottuu.

$git remote add origin git@github.com:käyttäjänimesi/repositorionnimi.git jaetaan lokaali repositorio GitHubin kanssa. (Git Reference. Git remote.)

$git push –u origin master Siirretään lokaalissa repositoriossa olevat tiedostot GitHubiin (Kuva 38). GitHub pyytää käyttäjänimeäsi palveluun sekä salasanaa käyttäjän varmentamiseen.

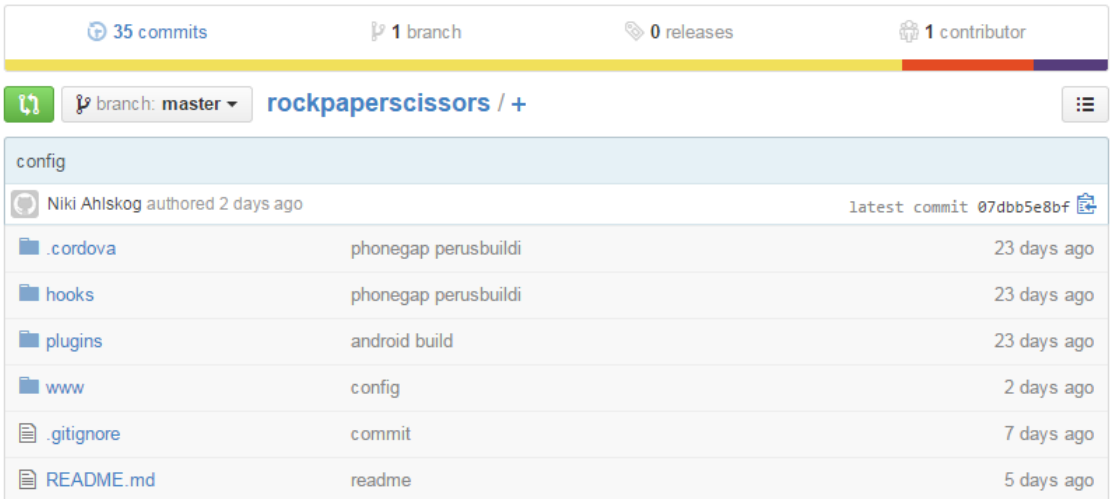
```
shnigi@shnigi-HP-EliteBook-2570p:~/oppari/ont/www$ git add .
shnigi@shnigi-HP-EliteBook-2570p:~/oppari/ont/www$ git commit -m "kaikki mukaan"
[master 305016a] kaikki mukaan
 57 files changed, 3697 insertions(+)
 create mode 100644 config.xml
 create mode 100644 css/index.css
 create mode 100644 icon.png
 create mode 100644 img/logo.png
```

Kuva 37. Gitin käyttö.

```
shnigi@shnigi-HP-EliteBook-2570p:~/oppari/ont/www$ git push -u origin master
Counting objects: 82, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (67/67), done.
Writing objects: 100% (81/81), 1.17 MiB | 0 bytes/s, done.
Total 81 (delta 1), reused 0 (delta 0)
To git@github.com:shnigi/mobiilisovellus.git
   54f728c..305016a  master -> master
Branch master set up to track remote branch master from origin.
shnigi@shnigi-HP-EliteBook-2570p:~/oppari/ont/www$
```

Kuva 38. Tiedostot siirretty GitHubiin.

Nyt projektin pitäisi löytyä GitHubista julkisena, ja varsinainen kehitystyö voidaan aloittaa (Kuva 39).



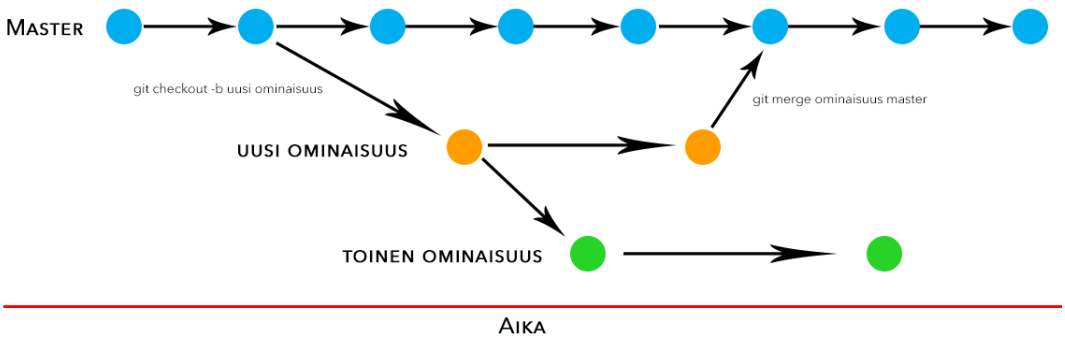
Kuva 39. Kuvakaappaus projektin versiohallinnasta GitHubissa. Lähde: GitHub.com

Gitillä työskennellään jatkossa seuraavilla komennoilla:
$git add . (lisätään kaikki tiedostot)
$git commit –m ”viesti” (tehdään lokaali commit)
$git push origin master (työnnetään tiedostot GitHubiin)

Pelkkä tiedoston paikallinen poistaminen ei poista jo kerran Gittiin siirrettyä tiedostoa. Jos Gitistä haluaa poistaa tiedoston, tulee käyttää komentoa **$git rm tiedosto.html,** jonka jälkeen tehdään normaalisti commit ja push.

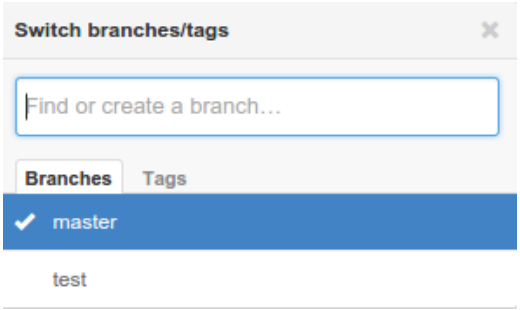
3.5 Gitin haarat ja tiedostojen ohitus

Jos sovellusta olisi kehittämässä useampi kehittäjä, voisi sovellukselle luoda omia kehityshaaroja, jotka lopulta ”mergetään”, eli yhdistetään yhdeksi kokonaisuudeksi (Kuva 40). Tämä tarkoittaa sitä, että jokainen kehittäjä tekee sovellukseen omaa toimintoaan omassa haarassaan, ”branchissa” (Kuva 41). Kun toiminto on valmis, siitä voidaan luoda ”pull request”, pull pyyntö GitHubiin, joka tarkoittaa sitä, että jonkun toisen kehittäjän on tarkistettava koodi ja hyväksyttävä sen yhdistäminen lopulliseen sovellukseen.



Kuva 40. Gitin työhaarat.

Sovellukselle voi luoda uuden haaran komennolla:
$ git checkout -b [oman_branchisi_nimi]
Branchi työnnetään GitHubiin komennolla:
$ git push origin [oman_branchisi_nimi]



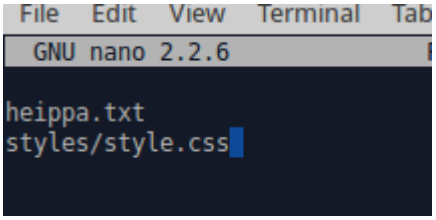
Kuva 41. Kuvakaappaus Test branchista sovellukselle. Lähde: GitHub.com

Komennolla: **$git branch** nähdään kaikki branchit, eli haarat. Kun halutut haarat halutaan lopulta yhdistää, valitaan ensin haara, johon halutaan yhdistää. Esimerkiksi **$git checkout master,** jonka jälkeen yhdistetään luotu testihaara **$git merge master test** Huomioi kuitenkin, että test-haaraan on pitänyt ensin tehdä joitakin muutoksia (Kuva 42).

```
shnigi@shnigi-HP-EliteBook-2570p:~/public_html/phpverkkokurssi$ git merge master
 test
Updating 38860a1..91b2c40
Fast-forward
 test.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 test.txt
```

Kuva 42. Esimerkki siitä, miten haarat master ja test on yhdistetty samaksi.

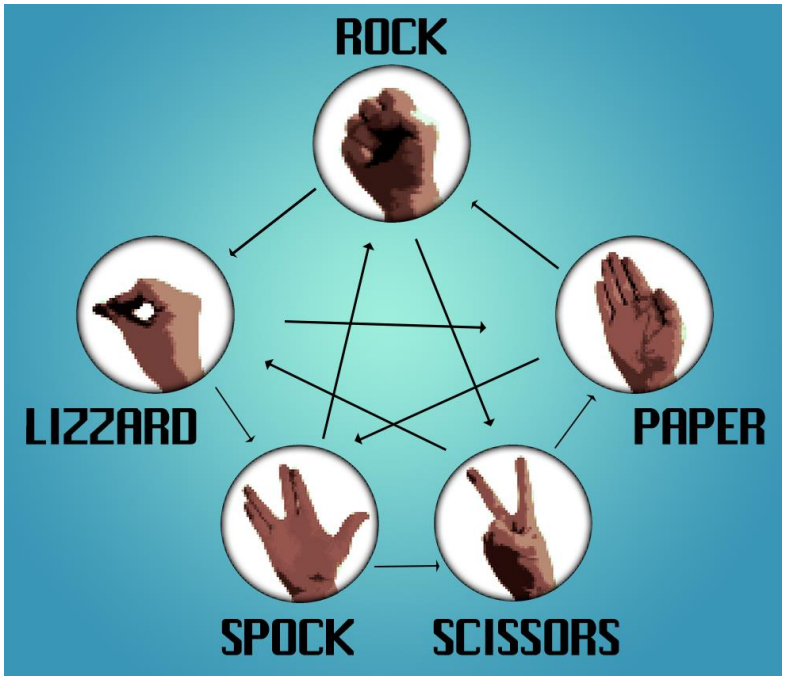
”.gitignore”, on tiedosto, jossa luetellaan kaikki tiedostot, joita Gitin ei tule seurata. Git ohittaa kaikki tiedostossa määritetyt tiedostot. Tähän tiedostoon on kätevää merkitä esimerkiksi asetustiedostoja, jotka ovat jokaisella kehittäjällä omat. Tiedosto on pienellä yhteen kirjoitettuna gitignore ja sen alussa on piste. Gitignore ei ota huomioon tiedostoja, jotka ovat jo seurannassa. Tiedosto tulee siis luoda ennen kuin tehdään commit. (Help.github.com. Ignoring files.) Tiedostoon merkitään ei seurattavat tiedostot allekkain ja polkuineen. Jos esimerkiksi projektin juuressa on tekstitiedosto ”heippa.txt”, jota ei haluta Gitin seurantaan, kirjoitetaan kyseinen tiedostonimi ohitustiedostoon. Jos projektin alakansiossa ”styles/style.css” oleva tiedosto halutaan ohitettavan versionhallinnassa, kirjoitetaan heippa.txt tiedoston alle kyseinen tiedosto, kuten se on tässäkin kirjoitettu (Kuva 43).



Kuva 43. Esimerkki .gitignore tiedostosta.

4 Sovelluksen kehittäminen

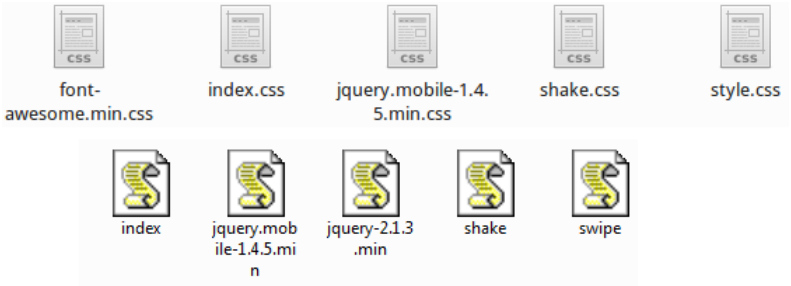
Esimerkkisovelluksena luodaan kivi, paperi, sakset, lisko ja spock -mobiilipeli. Peli toimii samalla logiikalla kuin alkuperäinen peli, mutta nyt pelaaminen tapahtuu tietokonetta vastaan ja normaalin kolmen aseen lisäksi peliin on tuotu kaksi lisäasetta. Pelin mekaniikka on selitetty kuvassa 44.



Kuva 44. Pelin logiikka

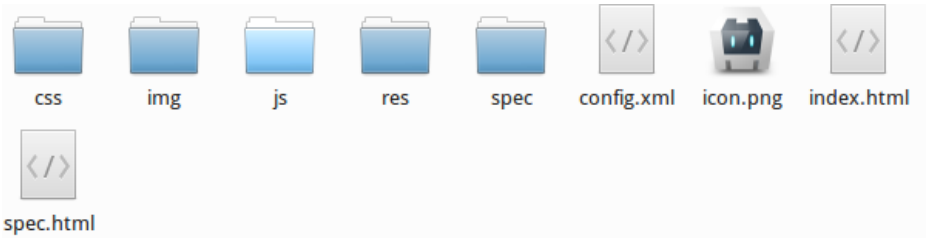
Pelin kehittäminen on aloitettu seuraavasti:
Luodaan PhoneGapilla tyhjä sovellus.
Alustetaan Git ja luodaan GitHubiin repositorio.
Buildataan, eli rakennetaan projekti Androidille.
Pushataan kaikki tiedostot GitHubiin talteen.

Sovelluskehitys tapahtuu pääosin www-kansion alla. Aloitetaan projekti tuomalla kaikki tarvittavat kirjastot käyttöön (Kuva 45). Sovelluksessa käytettäviksi kirjastoiksi valikoituivat jQuery-mobile, jQuery, Swipe.js, shake.js ja shake-animation. Lisäksi style.css, joka sisältää sovelluksen muut muotoilut.



Kuva 45. Käytetyt kirjastot.

Tiedostot on asetettu niille kuuluviin alakansioihin noudattaen PhoneGapin luomaa kansiorakennetta. WWW-kansio näyttää seuraavalta (Kuva 46):



Kuva 46. PhoneGap projekti www-kansiossa.

Aloitetaan pelin luominen index.html tiedostosta. PhoneGapin luoma ”HelloWorld” pohja näyttää seuraavalta (Kuva 47):

```
<html>
    <head>
        <meta charset="utf-8" />
        <meta name="format-detection" content="telephone=no" />
        <meta name="msapplication-tap-highlight" content="no" />
        <!-- WARNING: for iOS 7, remove the width=device-width and
height=device-height attributes. See https://issues.apache.org/jira/
browse/CB-4323 -->
        <meta name="viewport" content="user-scalable=no, initial-
scale=1, maximum-scale=1, minimum-scale=1, width=device-width,
height=device-height, target-densitydpi=device-dpi" />
        <link rel="stylesheet" type="text/css" href="css/index.css" />
        <title>Hello World</title>
    </head>
    <body>
        <div class="app">
            <h1>PhoneGap</h1>
            <div id="deviceready" class="blink">
                <p class="event listening">Connecting to Device</p>
                <p class="event received">Device is Ready</p>
            </div>
        </div>
        <script type="text/javascript" src="cordova.js"></script>
        <script type="text/javascript" src="js/index.js"></script>
        <script type="text/javascript">
            app.initialize();
        </script>
    </body>
</html>
```

Kuva 47. PhoneGap HelloWorld-ohjelma, joka vilkuttaa vihreällä pohjalla ”Device is Ready”

Tuodaan ensiksi valitut kirjastot käyttöön sovelluksen head-tageissa (Kuva 48).

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <meta name="format-detection" content="telephone=no" />
  <meta name="msapplication-tap-highlight" content="no" />
  <meta name="viewport" content="user-scalable=no, initial-scale=1, maximum-scale=1, minimum-scale=1,
  width=device-width, height=device-height, target-densitydpi=device-dpi" />
  <!--JavaScript-->
  <script src="js/jquery-2.1.3.min.js"></script>
  <script src="js/jquery.mobile-1.4.5.min.js"></script>
  <script src="js/swipe.js"></script>
  <script src="js/shake.js"></script>
  <!--Style-->
  <link rel="stylesheet" type="text/css" href="css/style.css">
  <link rel="stylesheet" type="text/css" href="css/shake.css">
  <link rel="stylesheet" href="css/jquery.mobile-1.4.5.min.css">
  <title>Rock-paper-scissors</title>
</head>
<body>
```

Kuva 48. Kirjastot otettu sovelluksen käyttöön.

Pelin logiikka on toteutettu seuraavasti: Shake.js kirjasto tarkkailee laitteen kiihtyvyysanturia ja laukaisee shakeEventDidOccur () funktion, jos mobiililaitetta ravistaa. Sen jälkeen ajetaan funktio, joka tarkistaa minkä aseen pelaaja on valinnut. Pelaajan valinta tarkistetaan swipe.js tiedostosta funktiolla mySwipe.getPos();. Funktio tarkistaa swipe-karusellissä olevan kuvan position. Lukema alkaa tyypillisesti nollasta ja kasvaa niin paljon kuin karusellissä on kuvia. Tietokoneen valitsema ase on generoitu yksinkertaisesti numerolla (Kuva 49). Javascriptin funktio Math.floor(Math.random() * 5) + 1, tuottaa tasaluvun yhden ja viiden väliltä. Jos numero on yksi, tietokone valitsee kiven. Jos lukema on kaksi, tietokone valitsee paperin. Jos luku on jokin muu, tietokone valitsee numeroa vastaavan aseen.

Kun molemmat ehdot on tarkistettu, eli pelaajan ja tietokoneen valinta, ohjataan pelaaja jQuery mobilen komennolla $.mobile.changePage sivun toiseen osaan, joka on merkattu risuaidalla, esimerkiksi #pageone. Näkymiä on yhtä monta kuin on pelitilanteitakin. Vaihtoehdot on kuvattu kuvassa 44. Kun käyttäjä liikkuu näiden näkymien välillä, ajetaan uutta näkymää vastaavat JavaScript-funktiot tapahtumakuuntelijoiden avulla. Suoritettavat funktiot päivittävät sovelluksen näkymiä, kuvia ja tekstejä ohjaamalla käyttäjä haluttuun sivun osaan. Tämä perustuu sovelluksessa käytettävään jQuery Mobile-kirjastoon ja navigointimalliin, joka pohjautuu tapahtumakuuntelijoihin. Jokaiselle pelin otteluvaiheelle on siis luotu oma näkymänsä (Kuva 50), yhteensä yhdeksän näkymää. Lisäksi pelin päänäkymä (Kuva 52) ja dialogi (Kuva 53), jossa on kerrottu pelin ohjeet.

```
//Logiikka paperille
else if (pos == "1" && computerWeapon == 1){
  //alert("Valitsit paperin, tietokone valitsi kiven");
  score ++;
  setTimeout(function(){  document.getElementById("scoreboard").innerHTML = "Your score: " + score;}, 2000);
  //document.getElementById("scoreboard").innerHTML = "Your score: " + score;
  $.mobile.changePage( "#pagesix", { transition: "slide", changeHash: false });
}
```

Kuva 49. Pelin logiikka toteutettuna JavaScriptillä. Näkymien välillä liikkuminen tapahtuu jQuery Mobilen omalla changePage -funktiolla.

Pelissä on lisäksi pistelaskuri, joka laskee voitot ja häviöt. Häviöstä pelaaja saa miinuspisteen ja voitosta pisteen. Tasapeleistä ei tule pisteitä.

```
<!-- sakset vs kivi -->
<div data-role="page" id="pagenine">
  <div data-role="header">
    <h1>Battle!</h1>
  </div>
  <div data-role="main" class="ui-content">
    <div class="ui-grid-a">
      <div class="ui-block-a fade-in one textcenter"><img src="img/saksi.png" alt="sakset" class=
      "responsive-image shake"><h3>You selected scissors</h3></div>
      <div class="ui-block-b fade-in two textcenter"><img src="img/kivi.png" alt="kivi" class=
      "responsive-image shake"><h3>Opponent selected rock</h3></div>
    </div><!-- /grid-a -->
    <h1 class="textcenter fade-in three">You lose, rock wins!</h1>
    <a href="#pageone" data-transition="slide" data-direction="reverse"><button>Play again!</button></a>
  </div>
</div>
```

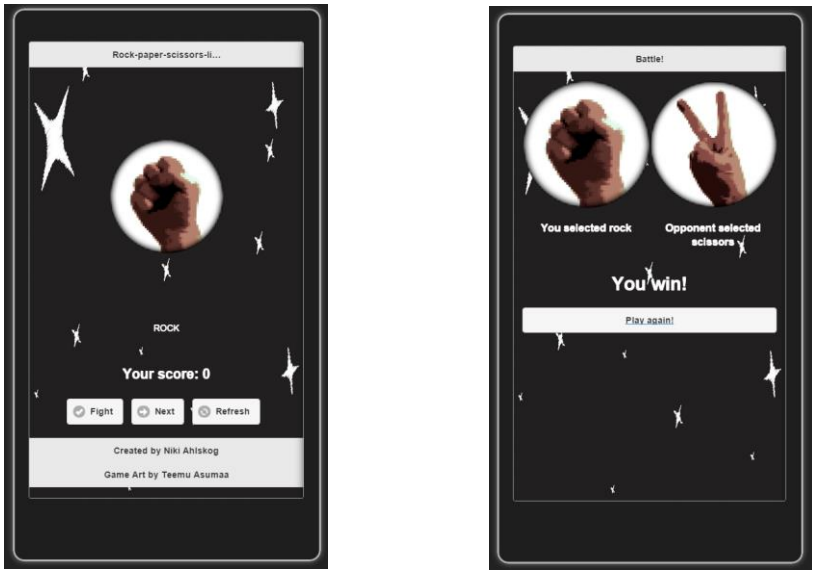
Kuva 50. Yksi pelin näkymistä, tai tilanteista. Sakset vastaan kivi -näkymä.

Sovelluksen style.css -tiedostoon on lisätty CSS3-animaatioita, joilla pelistä saadaan elävämmän näköinen (Kuva 51). Vaikka ottelun lopputuloksen tietokone päättää jo siinä vaiheessa, kun puhelinta ravistetaan, näytetään käyttäjälle sulavalinjaiset animaatiot. Tämä saa pelin tuntumaan ja näyttämään paremmalta.

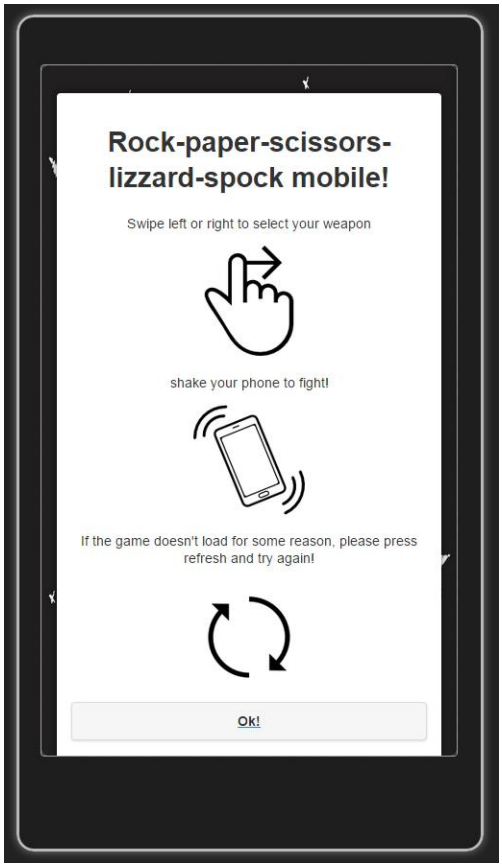
```
/*Fadein animation*/
@-webkit-keyframes fadeIn { from { opacity:0; } to { opacity:1; } }
@-moz-keyframes fadeIn { from { opacity:0; } to { opacity:1; } }
@keyframes fadeIn { from { opacity:0; } to { opacity:1; } }
```

Kuva 51. Sovelluksen animaatiot.

Valmiissa sovelluksessa on noin 700 riviä käsin kirjoitettua koodia, jotka voidaan jakaa JavaScriptiin ja jQueryyn, joilla sovelluksen ohjelmointi on toteutettu, sekä HTML ja CSS koodiin, joilla sovelluksen ulkoasu on luotu. Sovelluksessa käytetään lisäksi .PNG muotoisia kuvia grafiikkana. Kuvat ovat taustaltaan läpinäkyviä.



Kuva 52. Pelin päänäkymä ja oikealla näkymä yhdestä pelitilanteesta.



Kuva 53. Pelin alussa käyttäjää ohjeistetaan pelaamaan peliä.

4.1 Sovelluksen paketointi

Jotta sovelluksen voisi julkaista sovelluskaupassa, tulee lähdekoodit kääntää puhelimen tukemaan muotoon.

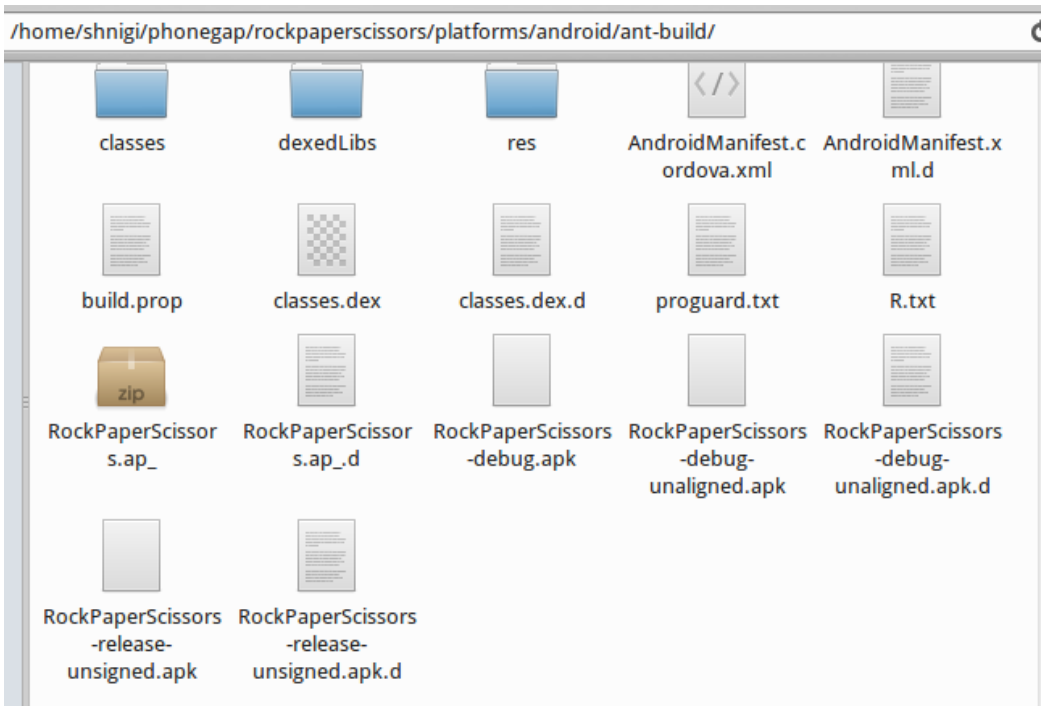
PhoneGapilla sovelluksen lähdekoodeista kääntäminen esimerkiksi Android laitteen tukemaan .APK tiedostomuotoon onnistuu seuraavalla komennolla:

$cordova build android –release (Stackoverflow keskustelufoorumi. Build Android release APK.) (Cordova.Apache.Org. Docs – Windows Phone 8 platform guide.)

Build-sanan jälkeen voidaan vaihtaa haluttu laitealusta, esimerkiksi iOS, Android, WP8 jne. (Stackoverflow keskustelufoorumi. Build iOS PhoneGap). Cordova kääntää sovelluksen haluttuun tiedostomuotoon ja tekee esimerkiksi Android alustalle seuraavat tiedostot:

- Sovellus-debug.apk
- Sovellus-debug-unaligned.apk
- Sovellus-release-unsigned.apk

Lisäksi muita tiedostoja, jotka eivät ole julkaisun kannalta merkitseviä tiedostoja. Tiedostot generoituvat sovelluksen juurikansiosta lähdettäessä platforms – android – ant-build kansioon (Kuva 54).

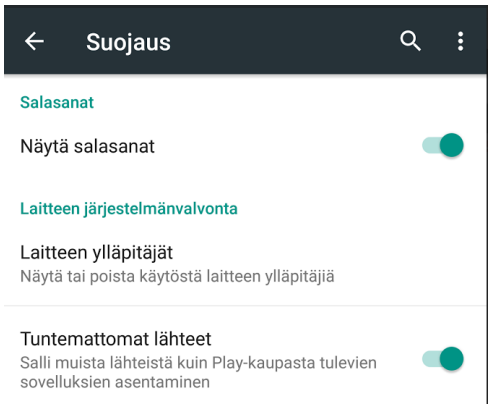


Kuva 54. Julkaisutiedostot käännettynä ja generoituna Android-alustalle.

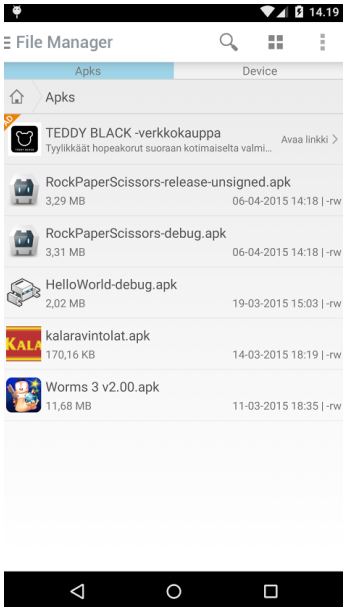
Kuvassa 56, on käännetty sovellus Android-alustalle. RockPaperScissors-debug.apk on sovelluksen virheenkorjaustiedosto. Sovelluksen testauksessa aiheuttama virhe voidaan paikallistaa ja korjata tämän tiedoston avulla. Sovellus on asennettavissa suoraan omaan puhelimeen (Kuva 57), jos puhelimesta sallitaan sovellusten asentaminen tuntemattomista lähteistä (Kuva 55). Debug tiedostossa voidaan kirjoittaa lokiin tietoja, sekä tarvittaessa sovelluksen näytölle voidaan tulostaa virheilmoituksia. Omaa sovellusta kannattaa testata aluksi debug-tiedostolla. Sovellus-debug-unaligned.apk tiedosto on tarkoitettu käytännössä samaan tarkoitukseen. Sovellus-release-unsigned.apk on tiedosto, joka viedään lopulta sovelluskauppaan. Tiedosto ei kuitenkaan kelpaa sellaisenaan, vaan se vaatii ensin zipalignin ajamisen ja sähköisen allekirjoituksen. Tätä tiedostoa ei pysty asentamaan omaan puhelimeen ilman allekirjoitusta (Kuva 59).

APK-tiedostot voi siirtää omaan puhelimeen esimerkiksi USB-kaapelilla, jonka jälkeen ne voi etsiä haluamallaan FileManagerilla, eli tiedostoselaimella, joita on saatavilla useita erilaisia Google Play kaupasta. Sen jälkeen asennus tehdään normaalisti (Kuva 58).

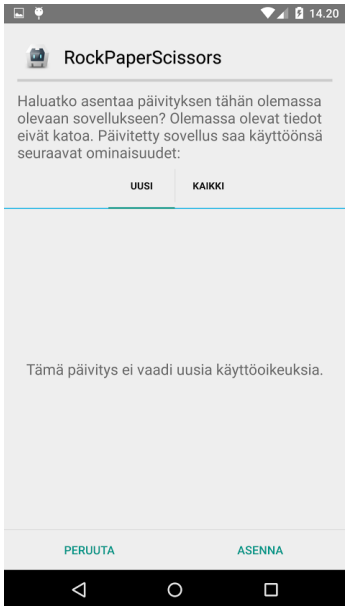
Tuntemattomista lähteistä asentamisen täytyy olla hyväksyttynä (Kuva 55). Asetus löytyy normaalisti Android puhelinten asetuksista kohdasta turvallisuus ja tuntemattomat lähteet:



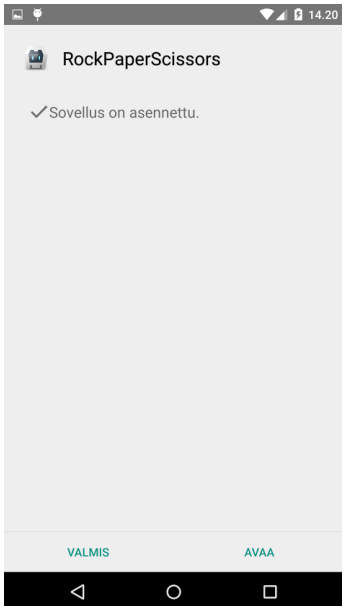
Kuva 55. Tuntemattomista lähteistä asentaminen sallittuna Android-puhelimella.



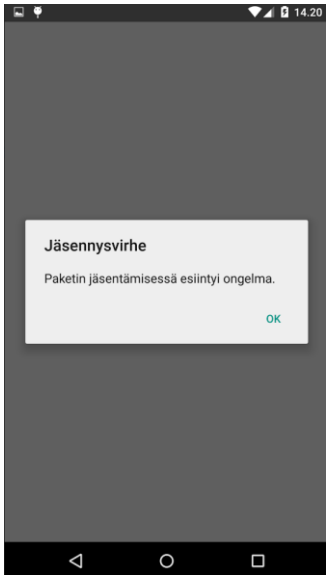
Kuva 56. APK tiedostot siirrettynä Nexus 6 puhelimeen USB-kaapelia käyttäen. Tiedostot näkyvät File Managerissa, joka on ladattu Google Play kaupasta.



Kuva 57. Debug sovellus voidaan asentaa kuin mikä tahansa muukin sovellus, jos tuntemattomista lähteistä asentaminen on sallittu.



Kuva 58. Sovellus asentui puhelimeen oikein ja on testattavissa.



Kuva 59. Sovelluksen release versio ei asennu puhelimeen ennen kuin se on sähköisesti allekirjoitettu ja zipalignattu.

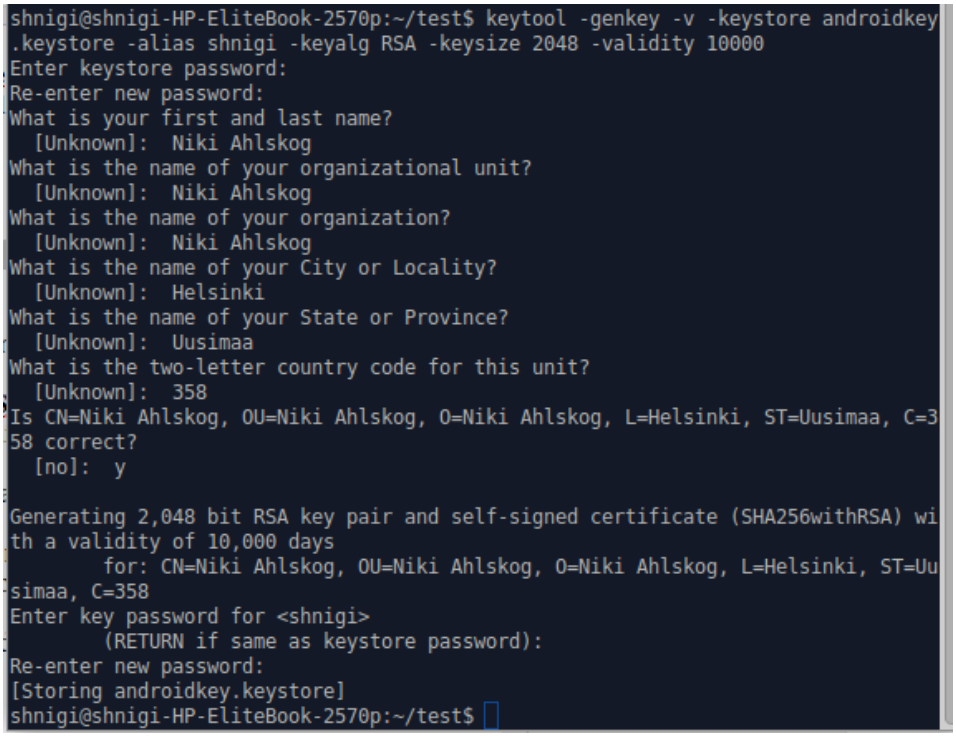
Zipalign on tiedostojen tarkistukseen käytettävä työkalu Androidilla, joka optimoi sovelluksen APK-tiedoston. Sen tarkoitus on varmistaa, että kaikki pakkaamaton data saadaan optimoitua ja näin sovelluksen kokoa saadaan mahdollisesti pienennettyä sekä varmistutaan, että sovellus toimii parhaalla mahdollisella tavalla. Tämä on pakollinen toimenpide sovellukselle, jotta se voidaan hyväksyä Google Play kauppaan. (Developer.android.com. Zipalign.)

Sähköinen allekirjoitus tarkoittaa sitä, että sovelluksen julkaisija luo ensin oman sähköisen avaimen, jolla sovellus allekirjoitetaan ja ladataan kauppaan. Jatkossa omaa sovellusta voi päivittää vain käyttämällä samaa sähköistä avainta, jolla sovellus oli alun perin allekirjoitettu. Tämä lisää turvallisuutta, jotta voidaan olla varmoja sovelluksen oikeasta julkaisijasta. Lisäksi voidaan ajatella, että jos kehittäjän tunnukset jostain syystä varastettaisiin, ei kukaan pysty korvaamaan olemassa olevia sovelluksia ilman alkuperäistä avainta. Sähköinen allekirjoitus on myös pakollinen vaihe ladattaessa sovellusta Google Play kauppaan.

Allekirjoitetaan sovellus manuaalisesti. Sovelluksen allekirjoituksen voi toki tehdä myös esimerkiksi IDE ohjelmasta, vaikkapa Eclipsestä. Luodaan sovellukselle oma allekirjoitusavain komennolla:

$ keytool -genkey -v -keystore my-release-key.keystore
-alias alias_name -keyalg RSA -keysize 2048 -validity 10000

Jossa my-release-key tarkoittaa avaimesi nimeä ja alias_name käyttäjän nimeä. Validity, tarkoittaa avaimen voimassaoloaikaa. Yllä oleva komento siis luo avaimen nimellä my-release-key.keystore, käyttäjän nimi olisi alias_name ja voimassaoloaika 10 000 päivää, eli noin 27 vuotta (Kuva 60).

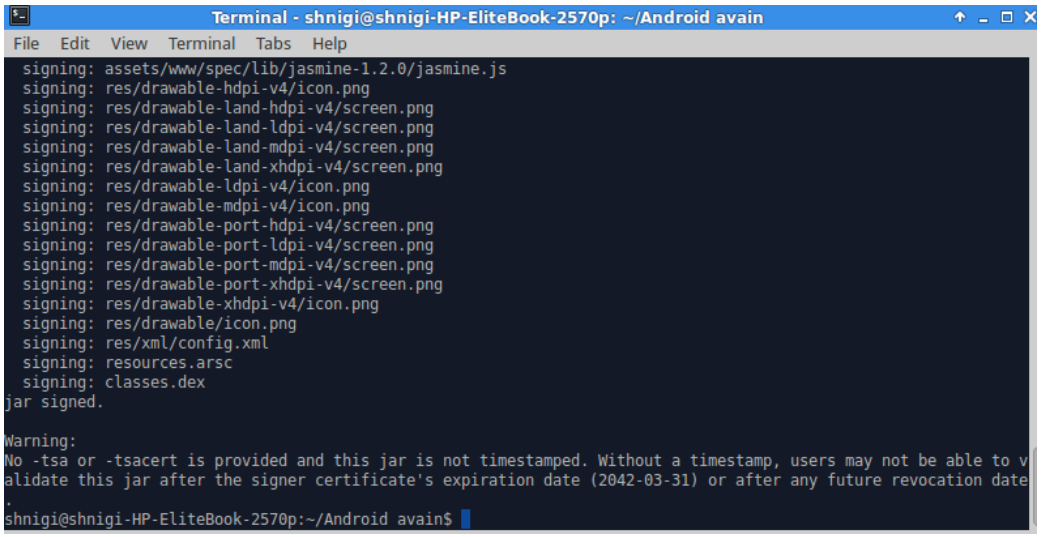


Kuva 60. Allekirjoitusavaimen luomisessa kysytyt kysymykset.

Generoidun avaimen ja sovelluksen release-unsigned.apk tiedostot voi siirtää esimerkiksi samaan kansioon, jolloin allekirjoitus on helppoa. Allekirjoitetaan sovellus komennolla:

$jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1
-keystore my-release-key.keystore sovellus-release-unsigned.apk alias_name

Kannattaa selkeyden vuoksi korvata esimerkissä käytetyt esimerkkinimet omilla tiedoillaan. Helpointa on myös avata yllämainittu komento jollakin tekstieditorilla, laittaa kaikki samalle riville ja tarkistaa, että nimet täsmäävät avaimesi, käyttäjänimen ja sovelluksen kanssa, jonka jälkeen se kopioidaan komentokehoitteeseen. Jos virheilmoituksia ei tullut, sovellus on nyt allekirjoitettu luodulla avaimella (Kuva 61). (Ahlskog 2014.)

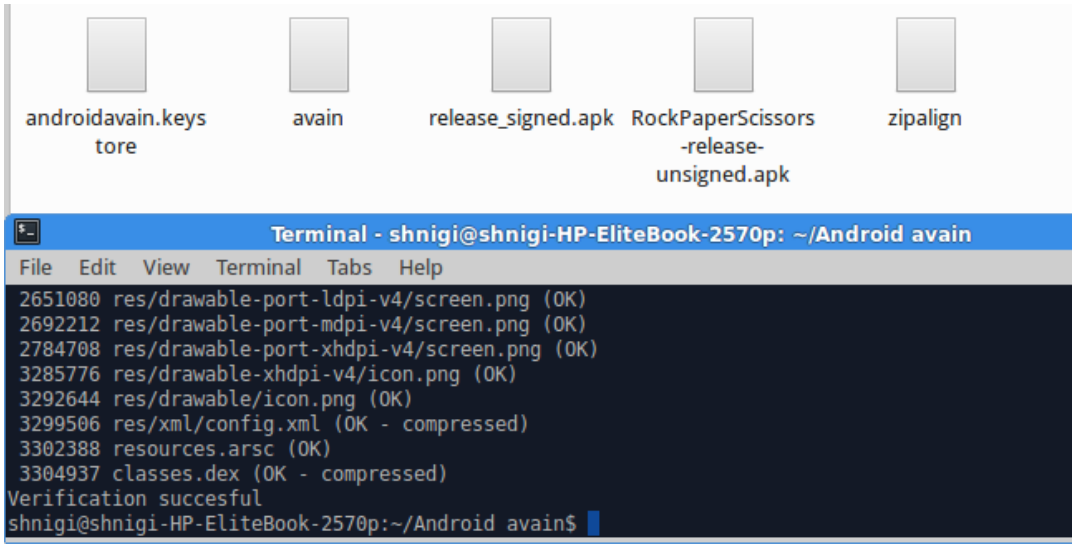


Kuva 61. Sovellus allekirjoitettuna sähköisellä avaimella.

Seuraavaksi suoritetaan sovelluksen tarkastus zipalign työkalulla Android SDK paketista. Mene /opt hakemistoon, jonne Android SDK on asennettu. Kansiosta build-tools löytyy kaikkien SDK-versioiden numerot, valitse uusin versio. Kansion sisältä löytyy tiedosto zipalign. Tämän tiedoston voi kopioida samaan paikkaan, jonne allekirjoitettu sovellus on laitettu (Kuva 62). Ajamalla komennon:

$ ./zipalign -v 4 sovellus-release-unsigned.apk release_signed.apk

Zipalign ajetaan tiedostolle unsigned ja ohjelma luo uuden apk-tiedoston nimeltä release_signed.



Kuva 62. Sovellustiedostot, allekirjoitusavain ja zipalign samassa kansiossa.

Tiedosto release_signed.apk on valmis tiedosto, joka voidaan ladata Google Play kauppaan. Ennen sovelluksen siirtämistä kauppaan, on hyvä testata sen toimintaa omassa puhelimessa vielä kerran. Näin vältytään käännösvirheiltä, jonka ansiosta sovellus ei välttämättä toimikaan oikein. Jos sovellus asentuu puhelimellesi ja toimii oikein, on aika siirtyä lataamaan se Google Play kauppaan.

4.2 Sovelluksen testaus

Työn tekijä on aikaisemmin työskennellyt ohjelmistotestaajana ja suunnitteli kevyen testauksen omien kokemustensa pohjalta. Ohjelmistotestausta suoritetaan yleensä laadun varmistamiseksi. Koska projektissa toimi vain yksi kehittäjä, toteutettiin testausta smoke-menetelmällä. Tämä tarkoittaa sitä, että peliä testattiin satunnaisista kohdista silloin tällöin. Tämä taso kattaa ainakin sen, että ohjelmistoa voidaan käyttää ja perusvirheet on tarkistettu. Testauksen ylin taso oli whitebox-testausta, joka tarkoittaa sitä, että sovelluksen syvempi toiminta ymmärretään. Näin voidaan ennalta määritellä millaista dataa testauksessa tarvitaan. Whitebox testaaja on yleensä tekniikkaan perehtynyt, ymmärtää ohjelmointia ja hänellä on pääsy lähdekoodeihin. Ohjelmistoa testattiin kehittäjän toimesta samalla kun sovellusta luotiin, sekä lisäksi lukuisia kertoja ohjelmointiajan ulkopuolella. Testausta voisi sanoa myös regressiiviseksi, joka tarkoittaa ennalta suunniteltujen testitapausten toistamista aina uuden toiminnon lisäyksen jälkeen. Tässä esimerkissä sovellukselle ei kuitenkaan suunniteltu testitapauksia, mutta smoke-testaus oli toistuvaa, eli regressiivistä.

Destructive testaus, joka tarkoittaa testausta, missä tahallisesti yritetään saada ohjelmisto kaatumaan, tai toimimaan ei toivotulla tavalla, löydettiin ainakin kolme ohjelmistovirhettä. Tämä toi ilmi testauksen tärkeyden sovelluskehityksessä. Sovellusta testattiin lisäksi muutamalla potentiaalisella loppukäyttäjällä ja mobiililaitteella. Havaintoina olivat ravistustoiminnon herkkyys, joka oli säädetty liian voimakkaaksi, sekä swipe.js kirjaston ongelma, jossa kuvat eivät satunnaisesti latautuneet pelaajalle. Tästä syystä peliin tuli luoda nappula, jota painamalla pelin voi resetoida. Peliin on luotu lisäksi nappulat aseen vaihtamiselle ja taistelun aloittamiselle, siltä varalta, että peli ei toimi kaikissa mobiililaitteissa oletetulla tavalla.

Sovellus vaatisi paljon testausta, jotta voitaisiin olla varmoja sen toimimisesta erilaisilla laitteilla ja nappulat voitaisiin poistaa lopullisesta versiosta. Sovelluksen jatkokehitysehdotukset on käyty läpi kappaleessa 7 Pohdinta. Käyttäjiltä saatua palautetta hyödynnettiin sovelluksen kehitysprosessissa, joka on käyttäjälähtöistä suunnittelua ja

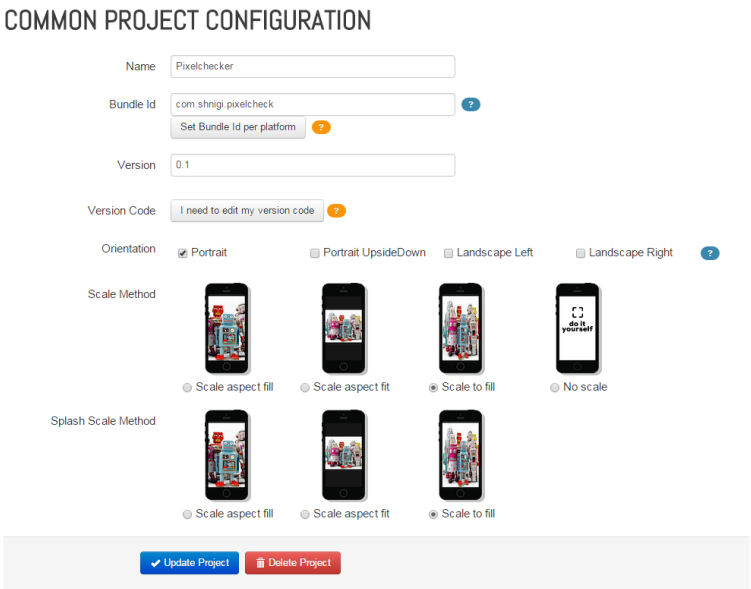
johtaa yleensä parhaisiin lopputuloksiin. Parhaan palautteen saa yleensä sovelluksen mahdollisilta käyttäjiltä (Vainio 2013).

5 Vaihtoehtoiset kehitystavat PhoneGapille – pilvipalvelut

Jos PhoneGap asennus tuntuu liian haastavalta, tai aikaa vievältä, voi kehittäjä luoda sovelluksia kääntämällä ne pilvessä. Pilvessä kääntäminen voi myös nopeuttaa kehitysprosessia. Kaksi kätevää palvelua pilvessä kääntämiseen ovat CocoonJS ja Adobe PhoneGap Cloud.

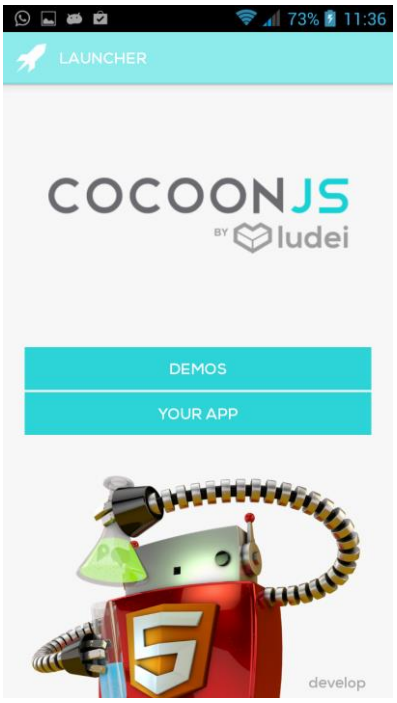
5.1 CocoonJS

Ludein kehittämä CocoonJS https://www.ludei.com/cocoonjs/ on pilvipalvelu, jossa HTML5-sovelluksen voi kääntää suoraan haluamalleen alustalle (Kuva 63). Oma HTML5-sovellus ladataan palveluun zip-tiedostossa, jonka jälkeen palveluun ladataan sovelluksen käyttämä sovelluskuvake ja splash, eli introkuva. CocoonJS lisää sovelluksen alkuun oman logonsa, muutoin palvelu on ilmainen ja mainoksista vapaa. CocoonJS yrittää lisäksi parantaa sovelluksen toimintaa ja hyödyntää laitteistokiihdytystä mahdollisimman tehokkaasti. Palvelu soveltuu erinomaisesti esimerkiksi pelien tekemiseen. CocoonJS käyttää HTML5-standardeja apeja, eli ohjelmistorajapintoja, jonka avulla voitaisiin käyttää esimerkiksi kiihtyvyyssensoria. Palvelun rajoituksena on kuitenkin 30mb zip-tiedosto. Tämä rajoittaa sovelluksen kokoa, joten esimerkiksi suuria ääni-, tai kuvatiedostoja palveluun ei pysty lataamaan. Lisäksi sovelluksen lopullinen koko saattaa kasvaa ladatusta zip-tiedostosta. Huomioi tämä ladatessasi sovellusta Google Play kauppaan, joka ei ota vastaan 50mb suurempaa tiedostoa.



Kuva 63. Kuvakaappaus CocoonJS projektin perusasetuksista. Lähde: https://www.ludei.com/cocoonjs/

CocoonJS tarjoaa lisäksi oman kehittäjille suunnatun sovelluksen, CocoonJS Launcherin (Kuva 64), joka on ladattavissa sovelluskaupoista. Sovelluksella pystyy testaamaan omaa sovellusta ajamalla ohjelma jostakin web-osoitteesta. Oman sovelluksen voi esimerkiksi julkaista omalla serverillä, tai muulla paikallisella palvelinpaketilla ja avata sovelluksen sieltä.

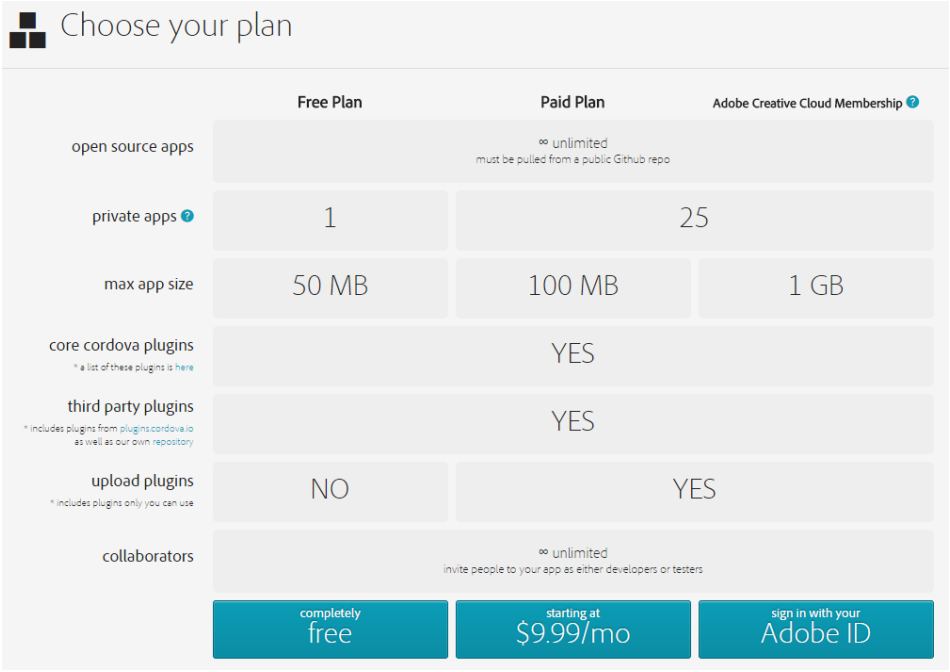


Kuva 64. CocoonJS Launcher, jolla omaa sovellusta voi testata ennen sen kääntämistä.

Yksinkertaisuudessaan CocoonJS on loistava palvelu, joka soveltuu hyvin tietyntyyppiselle kehittämiselle. Jos esimerkiksi sovelluksen alussa näkyvä Ludein logo ei haittaa, on palvelu oiva valinta kehittäjälle, joka ei halua nähdä vaivaa PhoneGapin asennuksessa. CocoonJS lähettää kehittäjälle lopuksi sähköpostiin linkin, josta valmiit sovellukset voidaan ladata. Huomioi kuitenkin, että esimerkiksi Android-alustalla sovellus tulee lopulta vielä allekirjoittaa ja suorittaa zipalign. Tämä täytyy siis tehdä muutoin.

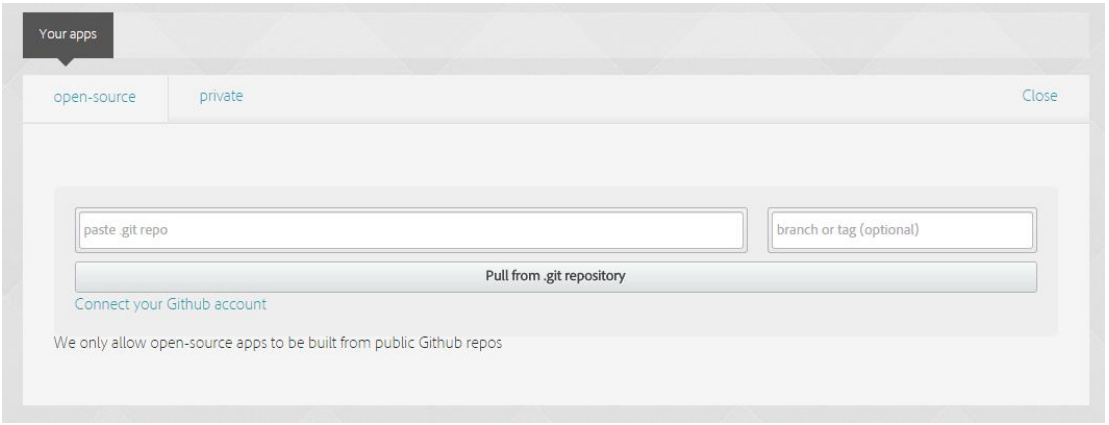
5.2 Adobe PhoneGap Cloud

Adoben pilvipalvelussa PhoneGap projekteja voi kääntää käden käänteessä. Android SDK pyörii Adoben pilvessä, jossa kääntäminen tapahtuu. Kehittäjä voi ladata esimerkiksi PhoneGapin ”HelloWorld” projektin GitHubista, sen jälkeen voidaan kehittää oma sovellus ja ladata se Adoben pilvipalveluun. Palvelu sopii erinomaisesti Adobe Creative Cloud -jäsenille, sillä heillä todennäköisesti on muitakin Adoben palveluitaan käytössä. Ilmaiseksi Adoben pilvipalvelussa saa ainoastaan yhden yksityisen sovelluksen ja sovelluksen maksimikoko on 50mb (Kuva 65).



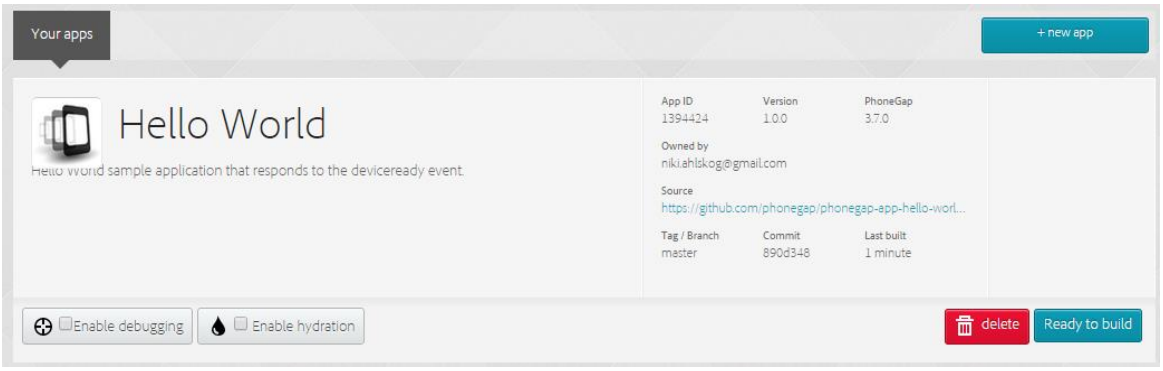
Kuva 65. Kuvakaappaus Adobe PhoneGap Cloudin hinnoittelumallista. Lähde: https://build.phonegap.com/

Kymmenellä dollarilla kuukaudessa saa jo 25 yksityistä sovellusta palveluun. Open-source projekteja palvelussa voi olla loputtomasti. Adoben palveluun ladataan projekteja GitHub palvelusta (Kuva 66).



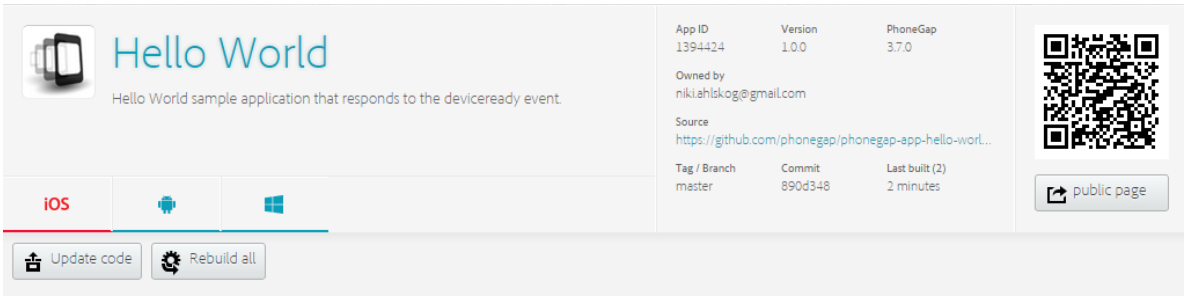
Kuva 66. Kuvakaappaus Adobe PhoneGap Cloud näkymästä. Lähde: https://build.phonegap.com/

Omaan GitHubiin voi tehdä ”forkin” esimerkiksi PhoneGapin ”HelloWorld” sovelluksesta, josta sovellusta aletaan kehittää ja tämä GitHub repositorio pullataan Adoben pilvipalveluun.



Kuva 67. Kuvakaappaus PhoneGapin HelloWorldista ladattuna Adobe PhoneGap Cloud -palveluun. Lähde: https://build.phonegap.com/

Klikkaa Ready to build ja palvelu kääntää sovelluksen iOS, Android ja WP8 alustoille valmiiksi (Kuva 67). Sovelluksen asennus ja testaus tapahtuvat skannaamalla palvelun ilmoittama QR-koodi (Kuva 68). Sovelluksen voi myös ladata omalle koneelleen.



Kuva 68. Kuvakaappaus Sovelluksesta valmiina ladattavaksi Adoben palvelusta. Lähde https://build.phonegap.com/

Tietokoneelle ladattu sovellus tuli debug muodossa. Adoben pilvipalvelun käyttö on yksinkertaista, helppoa ja ennen kaikkea nopeaa.

5.3 Yhteenveto pilvipalveluista

Kummassakaan palvelussa ei voinut vaikuttaa siihen, mihin laitteen ominaisuuksiin käyttäjän tarvitsee antaa sovellukselle käyttölupa. Molemmissa palveluissa sovellus vaatii täydet oikeudet, esimerkiksi kameran käyttö, GPS:n käyttö ja niin edelleen. Tämä oli yksi miinuspuoli pilvipalveluissa. Toinen miinus tulee siitä, että sovellus tulee aina ladata palveluun. Sovellusta ei voida vain yksinkertaisesti ”laukaista” omassa puhelimessa.

Yksinkertaisimmillaan pilvipalvelut mahdollistavat kuitenkin mobiilisovelluskehityksen ilman hankalia SDK-pakettien lataamista ja asennusta. Näin kehittäjä voi nopeasti kääntää sovelluksen mille tahansa puhelimelle, huolehtimatta siitä, onko tarvittavat kehitysympäristöt asennettu ja konfiguroitu oikein omalle tietokoneelle. Kummallakin

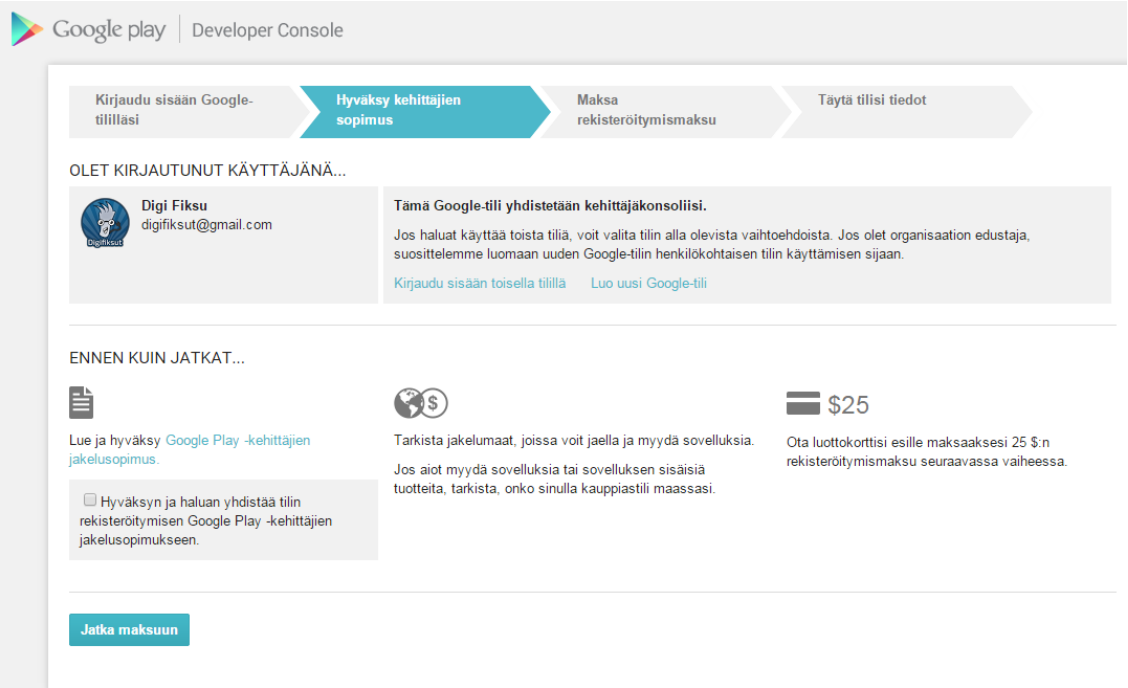
palvelulla on oma tapansa testata sovellusta ennen sen käyttämistä ja palveluiden käyttäminen on tehty yksinkertaiseksi. Yksinkertaisuus voi myös olla miinusta, kun kehittäjä ei itse pääse vaikuttamaan kaikkiin asetuksiin. Jokainen voi valita näistä itselleen parhaan työskentelytavan. Ennen sovelluksen lataamista johonkin sovelluskauppaan, se tulee vielä allekirjoittaa, mikä ei pilvipalveluissa onnistu suoraan. Sovelluksen allekirjoitus täytyy siis tehdä muualla.

6 Sovelluksen julkaisu Google Play kaupassa

Google Play, alun perin Android Market, on Googlen digitaalinen jakelukanava. Google Play toimii sovelluskauppana Android-käyttöjärjestelmällä varustetuille laitteille. Playsta voi etsiä ja ladata sovelluksia, jotka on kehitetty käyttäen Android SDK:ta. Google Play tarjoaa myös muuta digitaalista mediaa, kuten musiikkia, elokuvia ja kirjoja. Google Play sisältää ilmaista ja maksullista sisältöä. Sovelluskaupassa oli yli 1,43 miljoonaa sovellusta ladattavissa tammikuussa 2015. (Wikipedia.)

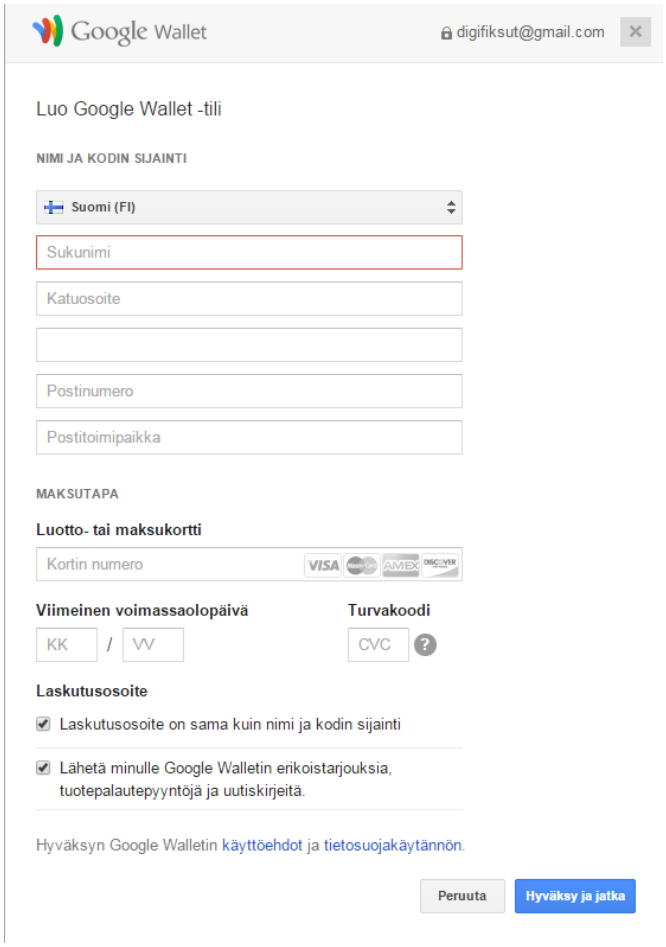
Jotta sovelluksen voi julkaista Google Play kaupassa, tulee käyttäjällä olla Google-tili ja maksaa 25 dollarin hintainen rekisteröitymismaksu.

Mene osoitteeseen https://play.google.com/apps/publish/signup/ ja kirjaudu sisään Google-tililläsi, johon haluat kytkeä kehittäjäkonsolin (Kuva 69).



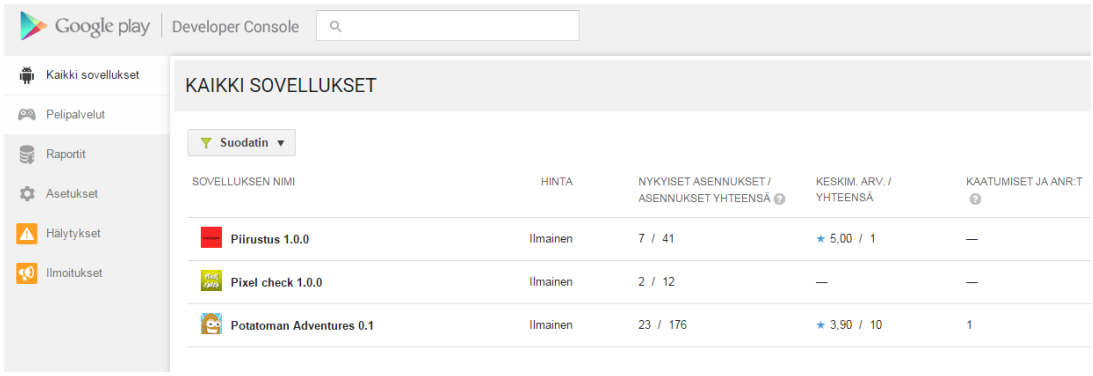
Kuva 69. Kuvakaappaus Google Developer Consolen kehittäjän sopimuksesta. Lähde: Google Developer Console.

Hyväksy ehdot ja maksa rekisteröitymismaksu (Kuva 70).



Kuva 70. Kuvakaappaus Google Wallet tilin luonnista. Lähde: Google Developer Console.

Kun olet kirjautunut sisään, aukeaa eteesi Googlen kehittäjäkonsoli (Developer Console) jossa sovelluksia voidaan hallita (Kuva 71).

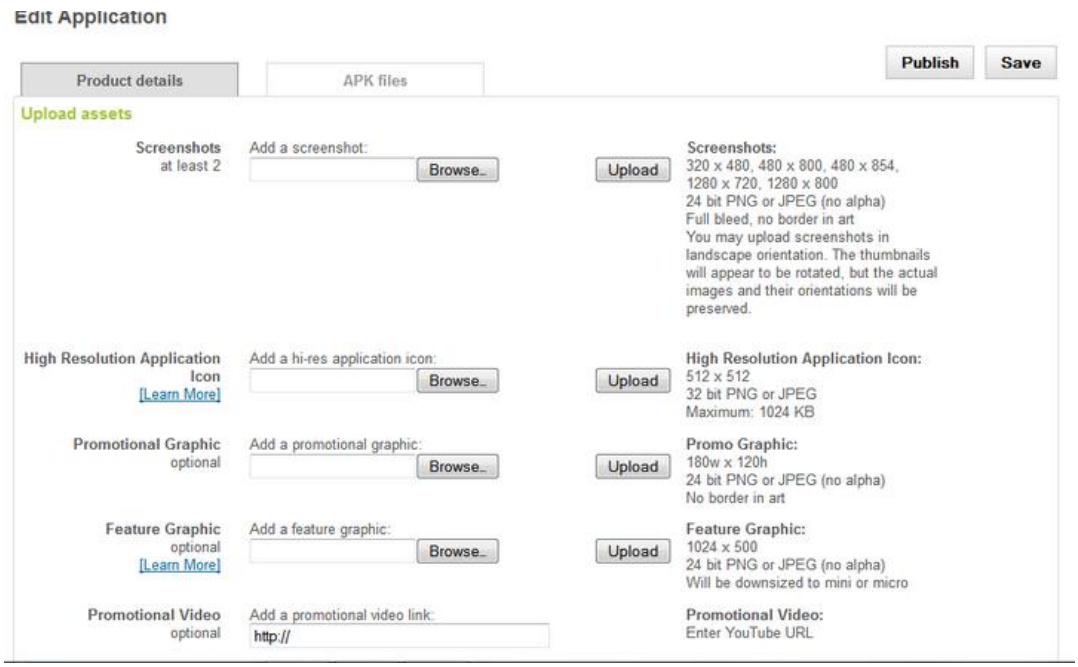


Kuva 71. Kuvakaappaus Google Developer Consolen päänäkymästä. Lähde: Google Developer Console.

Klikkaa sivun oikeassa yläreunassa näkyvää ”Lisää uusi sovellus” -nappulaa. Seuraavaksi pääset täyttämään sovelluksesi vaatimat tiedot.

Ennen sovelluksen julkaisemista, tulee Google Play kauppaan ladata kuvankaappaus sovelluksesta. Kuvankaappaus voi olla mikä tahansa seuraavista resoluutioista, tai

tiedostomuodoista: 320x480,480x800,480x854,1280x720,1280x800 , PNG tai JPEG. Lisäksi sovelluksesta tarvitaan 512x512 sovelluskuva, maksimissaan 1024kb (Kuva 72). Omasta sovelluksesta voi myös ladata mainosvideon. Kaikki edellä mainitut tiedot lisäävät oman sovelluksen kiinnostavuutta ja voivat nopeasti vaikuttaa latausmääriin. Kuvat ovat sovelluksesi mainoksia. Palvelun muita pakollisia täytettäviä tietoja ovat muun muassa nimi, sovelluksen lyhyt kuvaus, pitkä kuvaus, sovelluksen tyyppi: esimerkiksi peli tai sovellus, lisäksi sovelluksen luokka: viihde, pelit ja niin edelleen, sisältörajoitus, eli sisältääkö peli esimerkiksi aseita, huumeita, tai onko sisältö lapsille haitallista. Viimeisenä sovelluksen tekijän yhteystiedot, sivusto ja sähköposti. Sovelluksen tietosuojakäytäntöön voidaan laittaa ruksi, ”En lähetä tietosuojakäytännön URL-osoitetta nyt”.

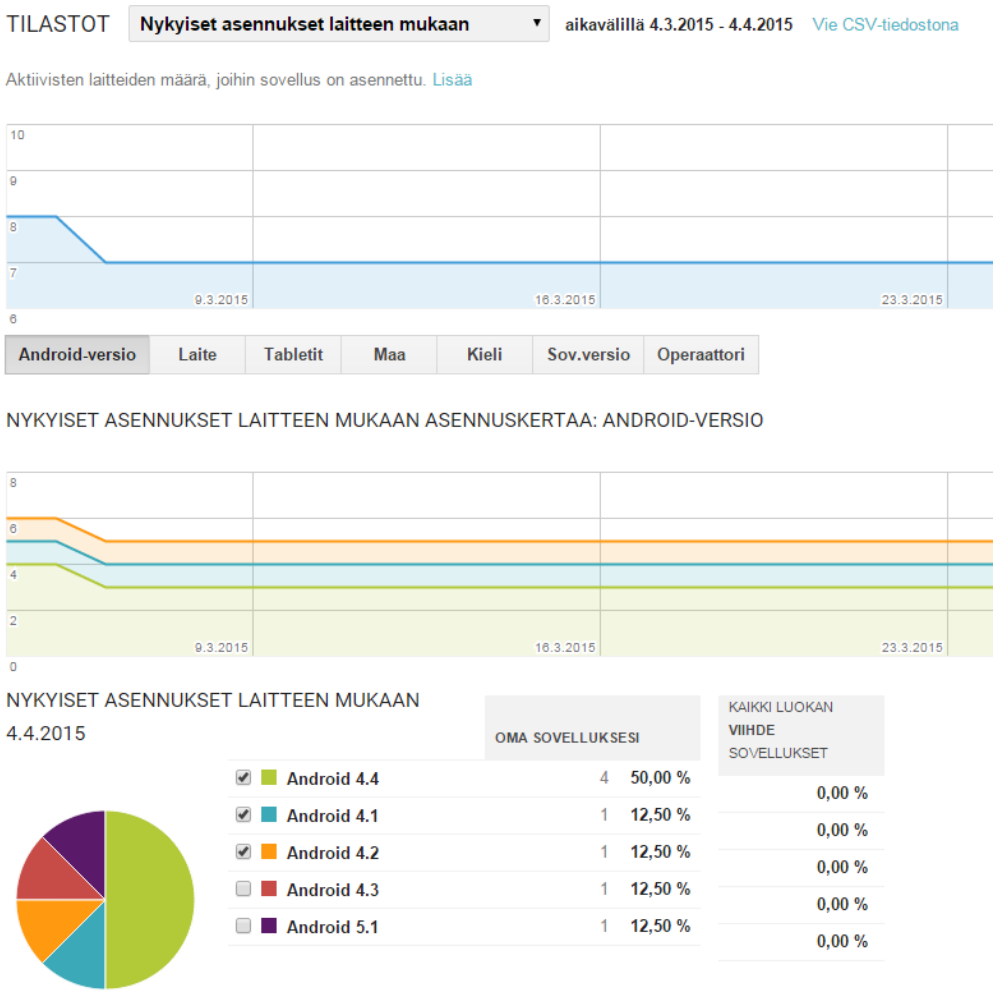


Kuva 72. Kuvakaappaus tarvittavista tiedoista Google Play kaupassa. Lähde: Google Developer Console.

Sovelluskaupassa voidaan tietysti myös myydä sovelluksia. Jotta sovellusta voitaisiin myydä, tulee sovelluksen lataajan luoda Google kauppiastili. Google tarjoaa omasta sovelluksesta myös tilastoja, arvioita ja arvosteluita, optimointivinkkejä, sekä kaatumisilmoituksia. Nämä kaikki tiedot auttavat sovelluksen jatkokehityksessä ja latausmäärien tarkastelussa. Yksi tärkeä seikka Google Play kaupassa on se, että julkaistua sovellusta ei voida koskaan poistaa lopullisesti! Sovelluksen julkaisemisen voi ainoastaan perua, tämä johtuu siitä, että Google pitää kaikki julkaistut apk-tiedostot itsellään. Sovelluksia voidaan filtteröidä, eli suodattaa kehittäjäkonsolissa, joten perutut julkaisut pystyy piilottamaan. Perutun julkaisun voi jättää perutuksi, tai sen voi korvata jollakin toisella sovelluksella. Näin ei periaatteessa ole edes tarvetta saada sovellusta poistettua kaupasta. Sovelluksen julkaisun peruuttaminen piilottaa sovelluksen näkymisen

Play kaupassa. Kuitenkin jos sovellusta ei ole vielä julkaistu, sen pystyy poistamaan kokonaan.

Jos sovellusta halutaan tulevaisuudessa päivittää, tulee versionumeron olla isompi kuin aikaisemmassa versiossa. Muutoin Google ei hyväksy sovellusta kauppaan. Sovelluksen tulee myös olla allekirjoitettu aiemmin luodulla avaimella. Uusi avain ei kelpaa. Viimeinen huomioon otettava seikka on, että Google hyväksyy maksimissaan 50mb kokoisen apk-tiedoston. Jos tiedosto on suurempi kuin 50mb, tulee käyttää julkaisussa ladattavia lisäosia. Kun sovelluksen kaikki tiedot on lopulta täytetty, voidaan sovellus julkaista. Julkaisun jälkeen kestää muutamasta tunnista jopa vuorokauteen, että oma sovellus tulee näkyville Play kaupassa kaikkialla maailmassa. Tilastojen näkyminen kehittäjäkonsolissa kestää vielä kauemmin (Kuva 73).



Kuva 73. Kuvakaappaus tilastojen näkymisestä kehittäjäkonsolissa. Lähde: Google Developer Console.

7 Pohdinta

Opinnäytetyön tuloksena syntyi toimiva työasema-asennus PhoneGapista minimitoiminnoilla, joka tarkoittaa sitä, että työskentely tapahtuu pääsääntöisesti komentokehotteesta. Monet kehittäjät suosivat komentokehoitetta sen tehokkuuden takia. Näin vältyttiin myös asentamasta ylimääräisiä sovelluksia ja paketteja kehittäjän koneelle.

Lisäksi luotiin kohtuullisen haastava mobiilipeli, joka sisältää useita näkymiä: ohjenäkymän, päänäkymän ja taistelunäkymät. Haaga-Helia ei opeta opiskelijoille JavaScriptiä, eikä jQueryä, joka toi projektiin haasteita. Siitä huolimatta sovelluksesta saatiin toimiva ja pelattava versio, jonka lähdekoodit ovat kaikkien saatavilla avoimessa GitHub-repositoriossa. Projektin aikana jQuery Mobile kirjaston käyttö tarkentui entisestään ja aiheutti innostusta jatkaa mobiilikehitystä. Mobiilipeliin saatiin luotua toimiva logiikka ja pistelaskuri.

Tekniikoiden valintaa voidaan pitää onnistuneina. jQuery Mobile toimii PhoneGap kehyksen kanssa mainiosti yhteen. Siirtymäanimaatiot ovat tarpeeksi kevyitä ja käyttöliittymän toteuttaminen on nopeaa. PhoneGap-sovelluskehys soveltuukin erinomaisesti sovelluksille, jotka eivät pidä sisällään paljon raskaita animaatioita. Sovelluksen suorituskyky oli hyvä kaikilla testatuilla laitteilla. Esimerkki suuren mittakaavan sovelluksesta, joka on toteutettu PhoneGapilla ilman raskaita animaatioita, on Wikipedian sovellus (phonegap.com. Wikipedia.), joka vaikuttaa toimivan erinomaisesti. PhoneGap rajoittuu suhteellisen kevyisiin web-pohjaisiin sovelluksiin, joten esimerkiksi 3D pelin toteuttaminen sillä olisi hyvin todennäköisesti mahdotonta.

Projekti tarjosi runsaasti haasteita. Suurimmat haasteet liittyivät uuden oppimiseen. JavaScript, jQuery, jQuery Mobile, kirjastojen käyttö ja kaikki uudet termit sekä tekniikat olivat lähes kaikki olennaisia opinnäytetyön tekemiselle. Projekti opetti todella paljon hyödyllisiä asioita ohjelmoinnista, JavaScriptistä ja jQuery Mobilen käytöstä.

Projektin haasteena oli myös eri työasemien käyttö projektissa. Ongelma johtui PhoneGapin buildista, joten ongelma ratkaistiin laittamalla PhoneGap projektin platforms-kansio gitignoreen, eli ohitustiedostoon. Projektin buildaus, eli rakentaminen suoritettiin aina paikallisella työasemalla ja tämä työskentelytapa todettiin toimivaksi.

Sovelluksen testauksessa (luku 4.2) todettiin muutama havaittu ohjelmistovirhe.

1. Testauksessa ilmi käyneet ohjelmistovirheet tulisi korjata jatkokehityksessä. Esimerkiksi swipe.js kirjaston kuvien lataamatta jättämisen korjaamisen myötä voitaisiin pelistä poistaa tarvittavat avustavat näppäimet. Lisäksi käyttäjät toivoivat ravistustoiminnosta herkempää.

2. Sovelluksen yksi selkä jatkokehitystoimenpide olisi web-palvelimelle tallennettava tulostaulu, jonne sovellus tallentaisi parhaimpien pelaajien voittoputket. Sovellus voisi sen jälkeen hakea tietokannasta parhaimmat pelaajat ja listata voittoputket.

3. Mobiilipeli voitaisiin myös kääntää iOS ja Windows Phone -laitteille. Teoriassa sovelluksen tulisi toimia kaikilla laitealustoilla ilman muutoksia, koska se on toteutettu PhoneGap alustalla ja HTML5 kielellä.

4. Jatkokehityksestä voisi kerätä loppukäyttäjiltä, eli pelin pelaajilta palautetta ja toivomuksia peliin haluttavista ominaisuuksista, ja toteuttaa palautteen perusteella haluttuja ominaisuuksia tärkeysjärjestyksessä.

Kaiken kaikkiaan PhoneGapin asennus oli monimutkainen prosessi, mutta onnistuneen asennuksen jälkeen työskentely oli jouhevaa. Alustariippumattomasta kehityksestä aiheutuvia haasteita on tiettyjen laiteominaisuuksien käyttö, niitä ei vain yksinkertaisesti ole mahdollista käyttää tai toteuttaa. Esimerkkinä widgetti Android-alustalla, se on mahdotonta toteuttaa hybriditekniikalla. Toinen haaste on suorituskyky. Sovellukset pyörivät web-näkymän päällä, joten sovelluksen suorituskyky on rajattu laitteen selaimen tehoihin. Kolmas haaste on sovelluksen käyttöliittymän toteuttaminen jokaiselle laitteelle mahdollisimman vähäisillä muutoksilla.

Alustariippumattomuus aiheuttaa myös haasteita testauksessa. Sovellus voidaan kääntää nopeasti kolmelle eri alustalle ja näiden alustojen alla voi olla useita eri käyttöjärjestelmän versioita. Lisäksi laitteita voi olla eri tehoisia ja näyttöjen resoluutiot voivat vaihdella. Testauksessa ei voida mitenkään olla varmoja sovelluksen toimivuudesta jokaisella laitteella. Periaatteessa sovellusta voidaan testata usealla emulaattorilla ja käyttöjärjestelmällä, mutta ne eivät koskaan vastaa fyysistä laitetta. Pitäisi siis olla useampia kymmeniä puhelimia joilla sovellusta voisi testata, jotta oltaisiin varmoja sovelluksen toimivuudesta kaikilla laitteilla ja alustoilla. Tämä on haaste ja ehkä ongelma hybriditekniikassa. Varsinkin jos sovellus käyttää laitteen sensoreita, on sovellusta hankalaa testata emulaattoreilla.

Alustariippumaton sovelluskehitys on samaan aikaan tehokasta ja haastavaa. Alustariippumaton kehitys on hyvä valita silloin, kun sovellus halutaan tehdä nopeasti jokaiselle laitteelle ja säästää aikaa sekä rahaa. Käytännössä jos halutaan tehdä jokin raskas peli, tai monimutkaisempi sovellus, ei alustariippumatonta sovelluskehitystä voida valita, sen edellä mainittujen haasteiden takia. Alustariippumatton sovelluskehitys ei sovellu todennäköisesti 3D-pelien tekemiseen teknisen toteutustapansa vuoksi. Sovelluksen suorituskyky, käyttöliittymä, liitännäiset ja alusta tulee ottaa huomioon ennen tekniikan valintaa. Ennen kehitystyön aloittamista kannattaa myös tarkistaa tuettuina olevat komponentit, jotta voidaan olla varmoja sovelluksen kääntämisestä kaikille laitteille.

Pilvipalvelut tarjosivat varteenotettavan ratkaisun mobiilikehitykselle ilman PhoneGappia, mutta sovelluksen allekirjoitus ja Zipalign täytyy tehdä silloin muualla. PhoneGapin asennuksesta ei löytynyt varsinaisia haittoja kehittäjän koneelle. Pilvipalveluita kannattaa käyttää esimerkiksi silloin, jos halutaan nopeasti protota jotakin sovellusta. Varsinaisen sovelluksen julkaisu voidaan aina myöhemmin toteuttaa PhoneGapilla ja lisäksi kehittäjä pääsee heti töihin. Kaikesta huolimatta PhoneGap kehitystyötä voi suositella sen monipuolisuuden takia. Projektin aikana suurimmat ongelmat keskittyivät juurikin PhoneGapin asennukseen, mutta kun sen on kerran asentanut käy kehitys jatkossa vaivatta.

Lähteet

Ahlskog, N 2014. Android sovelluskehityksen aloittaminen. Luettavissa: http://www.nikinlinux.blogspot.fi/2014/10/starting-android-developement-with.html. Luettu: 3.3.2015

Ajatuksella.com 2014. Phonegap, Eclipse ja Android-sovelluskehys. Luettavissa: http://ajatuksella.com/?l=a&id=1409295976. Luettu: 3.3.2015

Askubuntu.com 2013. Ubuntun keskustelufoorumi, 32 bittinen Android emulaattori. Luettavissa: http://askubuntu.com/questions/534044/error-32-bit-linux-android-emulator-binaries-are-deprecated-when-attemping-to-r. Luettu: 3.3.2015

Askubuntu.com 2013. Ubuntun keskustelufoorumi, Android SDK asennus. Luettavissa: http://askubuntu.com/questions/318246/complete-installation-guide-for-android-sdk-adt-bundle-on-ubuntu. Luettu: 3.3.2015

Atrice. Phonegap 2012. Phonegap selitetty visuaalisesti. Luettavissa: http://phonegap.com/2012/05/02/phonegap-explained-visually/. Luettu: 2.3.2015

Brian. Phonegap 2012. Phonegap, Cordova, erot? Luettavissa: http://phonegap.com/2012/03/19/phonegap-cordova-and-what%E2%80%99s-in-a-name/. Luettu: 2.3.2015

Code.tutsplus.com 2010. Android virtuaalipuhelimen asetukset. Luettavissa: http://code.tutsplus.com/tutorials/common-android-virtual-device-configurations--mobile-1574. Luettu: 14.3.2015

Cordova.apache.org. Cordovan dokumentit, yleiskatsaus. Luettavissa: http://cordova.apache.org/docs/en/4.0.0/guide_overview_index.md.html#Overview. Luettu: 14.4.2015

Cordova.Apache.Org. Docs – Windows Phone 8 platform guide. Luettavissa: http://cordova.apache.org/docs/en/4.0.0/guide_platforms_wp8_index.md.html Luettu: 6.4.2015

Developer.android.com. Emulaattorin käyttäminen. Luettavissa: http://developer.android.com/tools/devices/emulator.html. Luettu: 3.3.2015

Developer.android.com. Laitteet. Luettavissa:
https://developer.android.com/tools/devices/index.html. Luettu: 14.3.2015

Developer.android.com. SDK pakettien lisääminen. Luettavissa:
https://developer.android.com/sdk/installing/adding-packages.html. Luettu: 3.3.2015

Developer.android.com. Zipalign. Luettavissa:
http://developer.android.com/tools/help/zipalign.html Luettu: 6.4.2015

Developer.apple.com. iOS. Luettavissa:
https://developer.apple.com/programs/start/ios/. Luettu: 22.3.2015

Docs.phonegap.com. Komentorivin käyttöliittymä. Luettavissa:
http://docs.phonegap.com/en/4.0.0/guide_cli_index.md.html. Luettu: 8.3.2015

Finley, K. 14.7.2012. Mikä GitHub oikeastaan on? –Techcrunch.com. Luettavissa:
http://techcrunch.com/2012/07/14/what-exactly-is-github-anyway/. Luettu: 8.3.2015

Freeman, J 2013. Bitbucket vai GitHub? –Infoworld.com Luettavissa:
http://www.infoworld.com/article/2611771/application-development/bitbucket-vs--github--which-project-host-has-the-most-.html. Luettu: 8.3.2015

Git Reference. Git remote. Luettavissa:
http://gitref.org/remotes/#remote. Luettu: 8.3.2015

Gulati, S. 7.11.2013. PhoneGap sovelluskehitys for dummies. –OpenShift blogi. Luettavissa:
https://blog.openshift.com/day-10-phonegap-mobile-development-for-the-dummies/. Luettu: 8.3.2015

Help.github.com. Ignoring files.
Luettavissa: https://help.github.com/articles/ignoring-files/. Luettu: 6.4.2015

Janssen, G. Sovelluskirjasto. Luettavissa:
http://www.techopedia.com/definition/3828/software-library. Luettu: 23.3.2015

jQuery. Luettavissa:
https://jquery.com/. Luettu: 23.3.2015

Kingoapp. USB Debugging toiminto. Luettavissa:
http://www.kingoapp.com/root-tutorials/how-to-enable-usb-debugging-mode-on-android-5-lollipop.htm. Luettu: 23.3.2015

Mmocny. GitHub. Projektikomennot ja kansiorakenne. Luettavissa:
https://github.com/apache/cordova-cli#project_commands. Luettu: 8.3.2015

msdn.microsoft.com. Käyttäjätyypit Windows Phone sovelluskehittäjille. Luettavissa:
https://msdn.microsoft.com/en-us/library/windows/apps/jj863494.aspx. Luettu: 22.3.2015

NODEjs. Luettavissa:
https://nodejs.org/. Luettu: 14.3.2015

NPMJS.com. Mikä on NPM? Luettavissa:
https://docs.npmjs.com/getting-started/what-is-npm. Luettu: 14.3.2015

Phonegap. Tietoja. Luettavissa:
http://phonegap.com/about/. Luettu: 2.3.2015

phonegap.com. Wikipedia. Luettavissa:
http://phonegap.com/app/wikipedia/. Luettu: 13.4.2015

Reed, N. 26.8.2011. Mikä on NPM? Luettavissa:
https://docs.nodejitsu.com/articles/getting-started/npm/what-is-npm. Luettu: 14.3.2015

Rouse, R. 2.2013. Natiivisovellus. –searchsoftwarequality. Luettavissa:
http://searchsoftwarequality.techtarget.com/definition/native-application-native-app.
Luettu: 2.3.2015

Rouse, R. 7.2005. Secure Shell. –searchsecurity.com. Luettavissa:
http://searchsecurity.techtarget.com/definition/Secure-Shell. Luettu: 8.3.2015

Rust-Smith, D. 9.3. Pitäisikö käyttää PhoneGappia vai natiivia. –Davidrs.com blogi. Luettavissa:
http://davidrs.com/wp/should-you-build-phonegap-or-native/. Luettu: 2.3.2015

Stackoverflow keskustelufoorumi. Build Android release APK. Luettavissa:
http://stackoverflow.com/questions/20402818/build-android-release-apk-on-phonegap-3-x-cli
Luettu: 6.4.2015

Stackoverflow keskustelufoorumi. Build iOS PhoneGap. Luettavissa: http://stackoverflow.com/questions/19628957/how-to-make-release-build-for-ios-using-phonegap-command-line-tools Luettu: 6.4.2015

Stackoverflow keskustelufoorumi. Emulaattori samoilla asetuksilla, kuin fyysinen puhelin. Luettavissa:

http://stackoverflow.com/questions/22596527/setting-up-avd-to-same-specs-as-physical-device. Luettu: 14.3.2015

Stackoverflow keskustelufoorumi. jQuery, jQuery Mobile, jQuery UI. Luettavissa: http://stackoverflow.com/questions/6636388/jquery-vs-jquery-mobile-vs-jquery-ui. Luettu: 23.3.2015

Stackoverflow keskustelufoorumi. Meaning of API. Luettavissa: http://stackoverflow.com/questions/7440379/what-exactly-is-the-meaning-of-an-api Luettu: 2.4.2015

Summerfield, S. Mobiili web-sivu vastaan sovellus. - hswsolutions.com. Luettavissa: http://www.hswsolutions.com/services/mobile-web-development/mobile-website-vs-apps/. Luettu: 2.3.2015

SuperUser keskustelufoorumi. Bashrc tiedosto. Luettavissa: http://superuser.com/questions/49289/what-is-the-bashrc-file. Luettu: 3.3.2015

SuperUser keskustelufoorumi. SSH avaimen kuvio. Luettavissa: http://superuser.com/questions/22535/what-is-randomart-produced-by-ssh-keygen. Luettu: 8.3.2015

Traeg, P. 17.10.2013. HTML 5 ja natiivikoodin sekoittaminen, parhaat puolet molemmista. – Smashing Magazine. Luettavissa: http://www.smashingmagazine.com/2013/10/17/best-of-both-worlds-mixing-html5-native-code/. Luettu: 2.3.2015

W3Schools. jQuery intro. Luettavissa: http://www.w3schools.com/jquery/jquery_intro.asp. Luettu: 23.3.2015

Vainio, V. 2013. Käyttäjäkeskeisen suunnittelun hyödyntäminen PhoneGapmobiilisovelluksen kehitysprosessissa. Luettavissa: https://www.theseus.fi/bitstream/handle/10024/66977/Vainio_Ville.pdf?sequence=1. Luettu: 2.3.2015

Webobedia. Android SDK –twiittaus. Luettavissa: http://www.webopedia.com/TERM/A/Android_SDK.html. Luettu: 3.3.2015

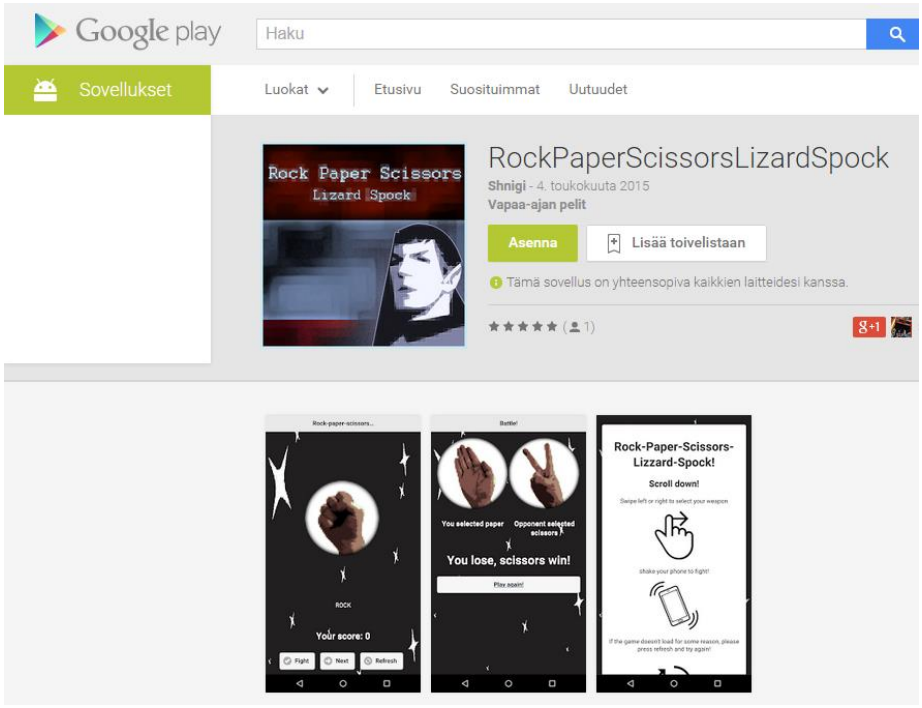
Wikikirjasto. CSS3. Luettavissa: http://fi.wikibooks.org/wiki/CSS3. Luettu: 23.3.2015

Wikipedia. Application Programming Interface. Luettavissa: http://en.wikipedia.org/wiki/Application_programming_interface Luettu: 2.4.2015

Liitteet

GitHub-repositorio: https://github.com/shnigi/rockpaperscissors

Google Play: https://play.google.com/store/apps/details?id=com.kps.shnigi



Kuva 74. Valmis sovellus Google Play kaupassa.