Juha Wilhola

YRITYKSEN TESTAUSJÄRJESTELMÄN KEHITYS

Tietojenkäsittelyn koulutusohjelma
2014

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

YRITYKSEN TESTAUSJÄRJESTELMÄN KEHITYS

Wilhola, Juha
Satakunnan ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Toukokuu 2014
Ohjaaja: Nieminen, Hans
Sivumäärä:32

Asiasanat: sovellussuunnittelu, ohjelmointi, Android

Opinnäytetyöni tarkoituksena oli kehittää sovellus, joka helpottaa tuotteiden testaamista huomattavasti. Valmistuessaan sovellus tulee olemaan tärkeä osa Virelabs Oy:n tuotteiden testauksessa.

Toteutettavan sovelluksen kautta tulisi hoitua tuotteiden testien hallinta ja testien tulosten valvonta, kuten testitulosten vertailu eri testien välillä. Sovelluksen pääasiallinen tarkoitus on automatisoida testien ajo sekä testituloksien jatkokäsittely.

Opinnäytetyössäni käyn läpi sovelluksen kehittämisessä käytettäviä menetelmiä sekä kuvaan toteuttamani sovelluksen.

Opinnäytetyön toteutuksessa ja suunnittelussa käytettiin Web2Py:tä, MySql tietokantasovellusta sekä Eclipseä Android ohjelmointiin.

DEVELOPING COMPANY'S TESTING SYSTEM

Wilhola, Juha
Satakunta University of Applied Sciences
Degree Programme in Business Information Systems
May 2014
Supervisor: Nieminen, Hans
Number of pages:32

Keywords: software development process, programming, Android

The purpose of my thesis was to develop an application that significantly facilitates testing of the products. Once completed, the application will be important part of testing of the Vire Labs Ltd products.

Management and control of the product tests, such as comparison of test results between different tests shall be handled through the developed application. A main purpose of the application is to automate test runs and further processing of test results.

In this thesis I will go through the methods used in the application development and get familiar with the implemented application.

Implementation and design of thesis was made using Web2PY, MySql database application and Eclipse for Android programming.

Sisällysluettelo

1 JOHDANTO....6
2 TIETOA YRITYKSESTÄ....6
2.1 Yritys....6
2.1.1 Yrityksen tuotteet....6
2.2 Lähtötilanne....9
3 PROJEKTI....10
3.1 Projektin tavoitteet....10
3.2 Projektin suunnittelu....10
3.2.1 Käyttöliittymän suunnittelu....10
3.2.2 Tietokannan suunnittelu....13
4 KÄYTETYT TYÖKALUT....14
4.1 Web2py....14
4.2 MySQL....15
4.3 Eclipse....16
4.4 MS Visio....17
5 TOTEUTUS....17
5.1 Tietokanta....17
5.2 Android sovellus....21
5.3 Sovellus testitulosten seurantaan....25
5.3.1 Sovelluksen pääsivu....25
5.3.2 Puhelimien tiedot....26
5.3.3 Testien tarkastelu....27
5.3.4 Kehityksen seuranta....29
6 JATKOSUUNNITELMIA....30
6.1 Testijärjestelmän integrointi osaksi versionhallintaa....30
6.2 Testijärjestelmän selainpohjaisen hallintatyökalun kehitys....30
6.3 Testijärjestelmän jatkokehitys....31
7 YHTEENVETO....31

TERMILUETTELO

CPU	CPU:lla tarkoitetaan elektronisesta laitteesta löytyvää suoritinta eli prosessoria.
GPU	GPU:lla tarkoitetaan grafiikan tuotosta vastaavaa suoritinta.
FPS	FPS tulee sanoista frame per second. Tämä kertoo, kuinka nopeasti näytöllä näkyvä kuva päivittyy sekunnin aikana.
ImageDiffer	ImageDiffer on tietokoneessa oleva ohjelma, joka vertaa kahta kuvaa palauttaen tuloksen. ImageDiffer kertoo, onko kyseessä identtiset kuvat.
Diff kuva	ImageDiffer palauttaa usein myös Diff kuvan, jos kuvat eroavat toisistaan. Kuvasta pystyy helposti toteamaan kuvapikseleiden erot.
Referenssi kuva	Referenssi kuva on kuva, johon jotain tiettyä ku vaa verrataan ImageDifferin avulla.
Fingerprint	Identifioi puhelimen siihen asennettujen ajureiden yms. perusteella.
Käyttöliittymä	Tuotteen osa, jonka kautta käyttäjä hallitsee, sekä silmäilee tuotetta.
Primary key	Pääavain identifioi tietokantataulussa olevat rivit.

1 JOHDANTO

Lopputyöni sain toteuttaa samassa yrityksessä, jossa suoritin työharjoittelun. Idea lopputyön aiheesta tuli yrityksen muilta työntekijöiltä, jotka olivat jo pitkään puhuneet testiympäristön puutteista ja sen parantamisesta.

Testiympäristön puutteet eivät olleet vielä merkittäviä yrityksen alkuvaiheessa, sillä silloin ei vielä käytetty yrityksen resursseja tuotteiden testaukseen, vaan suurin osa resursseista käytettiin tuotteiden suunnitteluun, sekä toteutukseen

Tässä lopputyössä tarkoituksenani on suunnitella ja toteuttaa yrityksen testijärjestelmään parannuksia.

1.1 TIETOA YRITYKSESTÄ

1.1.1 Yritys

Tein opinnäytetyöni yritykselle nimeltä Virelabs Ltd. Virelabs Ltd aloitti yritystoiminnan vuoden 2012 alussa. Yritys keskittyy pääasiassa Android laitteisiin suunnattujen käyttöliittymien suunnitteluun, sekä kehittämiseen. Opinnäytetyötä tehdessäni yrityksessä työskenteli kuusi oman alansa huippuosaajaa. Yrityksen työntekijöillä on omat roolinsa yrityksen sisällä.

Työskentelyn osa-alueet on järjestetty siten, että yrityksessä yksi työntekijä vastaa markkinoinnista, toinen grafiikasta, kolmas ohjelmoinnista ja niin edelleen. Virelabs:lta löytyy toimipiste Porin lisäksi myös pääkaupunkiseudulta. Toinen toimipiste avattiin vuoden 2012 lopulla.

1.1.2 Yrityksen tuotteet

Yritys keskittyy kehittämään Android älypuhelimille optimoituja käyttöliittymiä.

Käyttöliittymät perustuu yrityksen itse kehitettyyn VireLifeEngine 3D grafiikkamoottoriin. Tuotteita on tarkoitus myydä kuluttajille suoraan Google Play kaupan välityksellä. Yritys on myös solminut sopimuksen käyttöliittymän toteuttamisesta kiinalaisen puhelinvalmistajan UMEOX:in älypuhelimeen.

Yrityksellä on useita eri käyttöliittymä tuotteita kehitteillä. Jotkin käyttöliittymät tulevat olemaan alkuperäisen Android käyttöliittymän kaltaisia, mutta pääasiassa yrityksen suunnittelemat sekä valmistamat käyttöliittymät ovat aivan uudenlaisia verrattuna niihin johon olemme tottuneet.

Yrityksen tuotteet tulevat uudistamaan Android puhelinten käyttöliittymät uusilla innovatiivisilla ratkaisuilla. Käyttöliittymien tarkoituksena on tuoda huomattava parannus älypuhelimien nopeuteen, silti kuluttamalla vähemmän virtaa, verrattuna laitteen alkuperäiseen käyttöliittymään.

Käyttöliittymä tuo myös paljon visuaalisia parannuksia ulkonäköön sekä paljon enemmän muokattavuuksia ulkoasuun. Tuotteissa on myös pyritty ottamaan eri käyttäjäryhmien tarpeet sekä makutottumukset huomioon suunnittelemalla erilaisia tuotteita.

Kuvissa 1-3 esiintyy muutamia eri vaihtoehtoja yrityksen tuottamista käyttöliittymistä. Huomaa ensimmäisessä kuvassa näppärä kello, jota vetämällä esiin tulee ylimääräinen tila, johon voidaan sijoittaa sovelluksia tai pienoisohjelmia. Toisessa kuvassa taas näkyy yksi kotinäytön siirtymätila, jonka nimi on crystal.

Kuva 1: Vire Launcher.

Kuva 2: Vire Launcher.

Kuva 3: Vire Launcher.

1.2 Lähtötilanne

Yrityksellä oli valmiina suppea testausympäristö, joka toimi yrityksen palvelinkoneella. Testausympäristön tehtävänä on hallita testien suorittamista. Lyhyesti sanottuna testausympäristö ajaa erilaisia testejä puhelimille ja muille kohdelaitteille.

Näillä testeillä valvotaan, ettei tuotteiden lähdekoodissa ole syntaksivirheitä, jotka estävät tuotteen toiminnan/asentumisen kohdelaitteeseen. Testit kertovat samalla myös ruudunpäivitysnopeuden.

Ruudunpäivitysnopeus (FPS eli frame per second) kertoo, kuinka monta kuvaa ruudulle piirretään yhden sekunnin aikana. Mitä useampi kuva piirretään ruudulle, sitä sulavammalta kuva näyttää. Yleissääntönä voidaan pitää, että alle 24 kuvaa sekunnissa tuntuu ihmisen silmään nykivältä, kun taas suuremmalla kuvamäärällä kuva toistuu sujuvasti. Tämän takia on tärkeää seurata ruudunpäivitysnopeutta.

Testituloksia ei tallennettu tässä vaiheessa vielä lainkaan tietokantaan, vaan palvelin lähetti testitulokset etukäteen määriteltyihin sähköpostiosoitteisiin.

2 PROJEKTI

2.1 Projektin tavoitteet

Projektin tavoitteena on kehittää testijärjestelmää vastaamaan paremmin nykyistä tarkoitustaan, sillä testijärjestelmä ei kyennyt nykyisellään täyttämään siltä vaadittuja kriteereitä. Testijärjestelmä oli jäänyt selvästi jälkeen, johtuen Virelabs Ltd:n tuotteiden nopeasta kehittymisestä ja tuotevalikoiman laajentumisesta. Nykyinen testijärjestelmä vaati myös paljon tietämystä testeistä ja perehtymistä itse testijärjestelmään, tästä johtuen järjestelmän käyttö ei ollut mielekästä.

Projektissa testijärjestelmää tullaan kehittämään suuntaan, jossa siitä saa helposti ja vaivattomasti tärkeimmät informaatiot suorituskyvystä ja laadusta ennen asiakastoimitusta, kuten tuotteiden nopeudesta sekä virrankulutuksesta.

Testiympäristö tulee olemaan valmistuessaan yksi yrityksen tuotteista.
Tuotetta on myös tarkoitus myydä räätälöidysti muille sovelluskehittäjille.

2.2 Projektin suunnittelu

Tässä kappaleessa tulen käymään projektin eri suunnitteluvaiheita läpi kuvitellun projektin avulla. En käytä toteuttamaani projektia tässä esimerkkinä, koska kirjallinen suunnitteluvaihe jäi itseltä todella vähäiseksi. Tästä johtuen siitä ole juuri mitään dokumentteja.

2.2.1 Käyttöliittymän suunnittelu

Käyttöliittymän suunnittelusta puhuttaessa olen huomannut monilla ihmisillä olevan vääriä käsitteitä siitä, mitä se oikeastaan pitää sisällään.

Monet ihmiset luulevat suunnittelun sisältävän vain käyttöliittymän ulkoasun suunnittelua. He unohtavat kokonaan käytettävyyden suunnittelun, joka on todella tärkeää, kun lähdetään toteuttamaan mitä tahansa sovellusta.

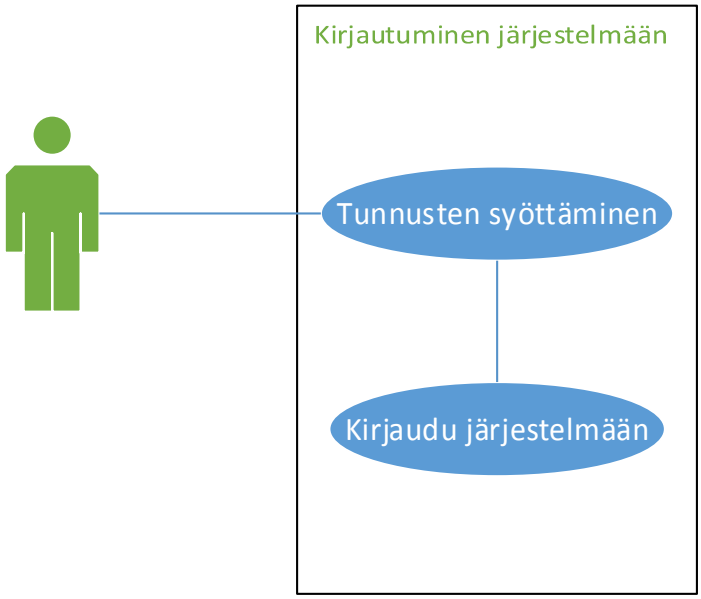
Käytettävyyttä kannattaa lähteä miettimään siltä pohjalta, mihin tarkoitukseen sovellus on tulossa, eli mitä sovelluksessa on tarkoitus tehdä. Kun tämä tiedetään, on aika alkaa hahmottelemaan sovelluksen käyttötapauksia. Käyttötapauksella tarkoitetaan niitä toimenpiteitä, mitä sovelluksen käyttäjä tulee tekemään, kun hän käyttää sovellusta.

Otetaan esimerkiksi koulun SoleOps väline, jonka käyttötarkoitus on mahdollistaa Samk:in oppilaille selainpohjaisen opetuksen suunnittelun, kuten kursseille ilmoittautumisen sekä kurssien arvosanojen tarkastelun. Alla on listattuna muutamia esimerkkejä oppilaiden käyttötapauksista, joita he kohtaavat käyttäessään palvelua:

- Kirjautuminen palveluun
- Opintojaksolle ilmoittautuminen
- Uusintatenttiin ilmoittautuminen
- Kurssi palautteen antaminen
- Opintohistorian tarkastelu
- Opintoselosteiden tarkastelu

Kun kaikki käyttötapaukset on selvitetty, jatkokehitetään niistä jokaisesta käyttötapauskaavio (kuva 4). Käyttötapauskaavio kuvaa käyttäjältä vaadittuja tekemisiä halutun käyttötapauksen suorittamiseksi. Tässä esimerkissä käytän käyttötapauksena kirjautumista palveluun.

Käyttötapauskaavioiden jälkeen on hyvä vielä käydä tarkemmin kaikki käyttötapaukset läpi ja tehdä niistä tarkennetut käyttötapauskuvaukset. Tarkennetussa käyttötapauskuvauksessa kuvaillaan yksityiskohtaisesti opastustyyppisesti, miten käyttäjä totetuttaa käyttötapauksen. Tämä helpoittaa myöhemmässä vaiheessa sovelluksen/sivuston suunnittelua.



Kuva 4: Kirjautuminen palveluun

Tässä kuviteltu esimerkki opintojaksolle ilmoittautumisen tarkennetusta käyttötapauksesta:

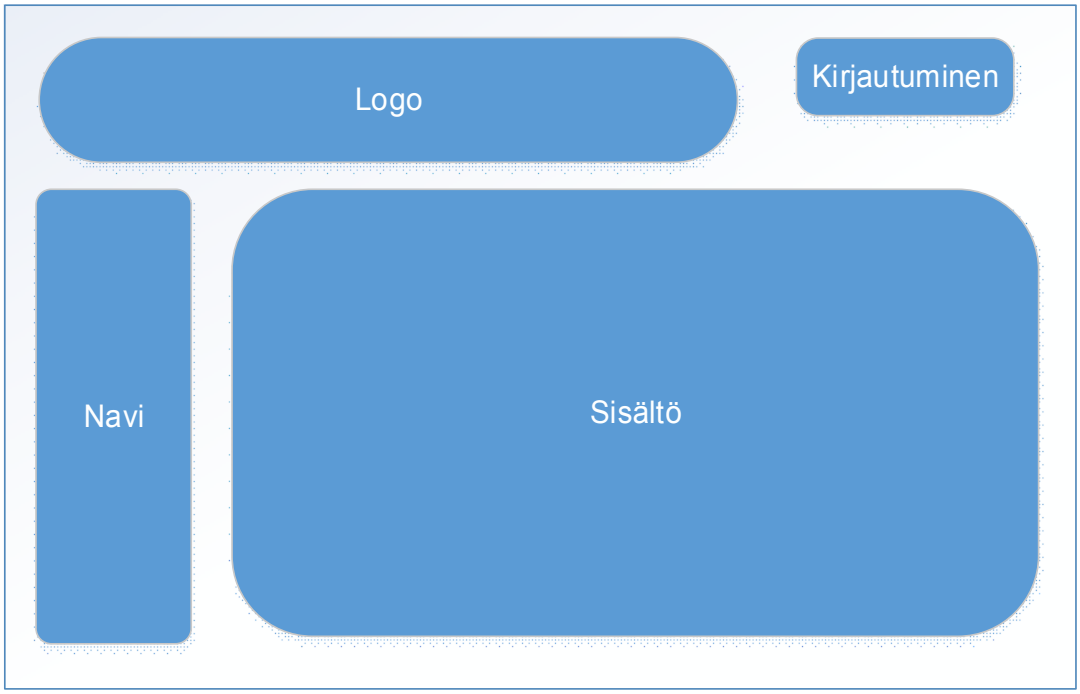
Tarkennettu käyttötapaus: Opintojaksolle ilmoittautuminen

Kirjauduttuaan palveluun, opiskelija klikkaa opiskelijan linkeistä ”Ilmoittautuminen opintojaksoille”-linkkiä.Tällöin näkyviin tulee ensisijaisesti opiskelijan koulutusohjelmaan kuuluvat opintojaksot. Opiskelija voi myös hakea opintojaksoja ”Tarkennettu haku”-toiminnolla. Opintojakson vieressä on valintaruutu. Opiskelija aktivoi kaikkien opintojaksojen valintaruudut, joille hän haluaa ilmoittautua ja painaa lopuksi ”Ilmoittaudu opintojaksoille”-nappia. Opintojakson vieressä näkyy myös arvosana, suoritus ja onko opiskelija hyväksytty meneillään olevalle opintojaksolle.

Lopuksi suunnitellaan käyttöliittymämalli. Käyttöliittymämallista selviää sivujen yleisilme.

Ohjelman käyttöliittymämallin (kuva 5) suunnittelu kannattaa toteuttaa huolellisesti, sillä ulkoasu on aina se, jonka käyttäjät tulevat näkemään ensimmäiseksi. Hyvin suunniteltu ja toteutettu ulkoasu luo ensimielipiteen ohjelmasta ja täten toivottavasti antaa positiivisen ensivaikutelman.

Ulkoasun suunnittelulla on tarkoitus jäsentää sivuston/ohjelman sisältö sekä kiinnittää kävijän huomio tiettyihin asioihin sivulla, kuten tärkeisiin linkkeihin tai muihin keskeisiin asioihin.



Kuva 5: Kuviteltu käyttöliittymämalli

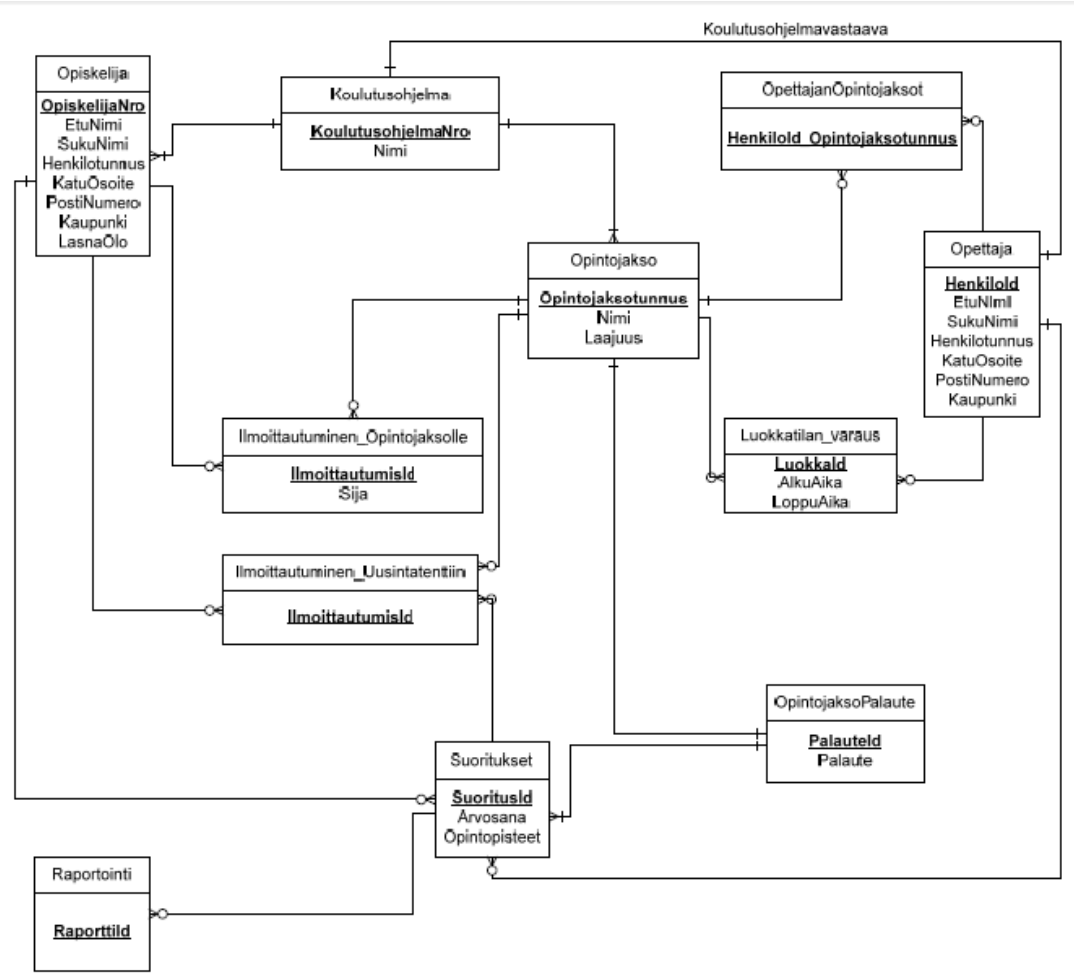
2.2.2 Tietokannan suunnittelu

Tietokantoja suunniteltaessa ensimmäinen vaihe on tehdä käsiteanalyysi. Käsiteanalyysissä kuvataan reaalimaailmasta rajattua osaa eli kohdealuetta, jota on tarkoitus kuvata tietokannassa, jotta kyetään toteuttamaan hyvin toimiva sekä tarpeita vastaava tietokanta. (Hovi 2005, 32.)

Kuvitellaan esimerkiksi tilanne, jossa SoleOps järjestelmää varten toteutetaan käsiteanalyysi.

Käsiteanalyysissä on tarkoitus selvittää tietokannan kannalta tarvittavat käsitteet. Käsitteiksi voidaan luokitella kaikki reaalimaailmassa esiintyvät objektit. SoleOps järjestelmässä käsitteitä voisi olla mm. opiskelija, koulutusohjelma, suoritukset, opintojakso, ilmoittautuminen opintojaksolle, ilmoittautuminen uusintatenttiin, opintojaksopalaute yms.

Käsiteanalyysin perusteella pystytään toteuttamaan käsitemalli (kuva 6), jossa kuvataan tuleva tietokanta graafisesti ja annetaan näin pohja tietokannan suunnittelulle.



Kuva 6: SoleOps käsitemalli

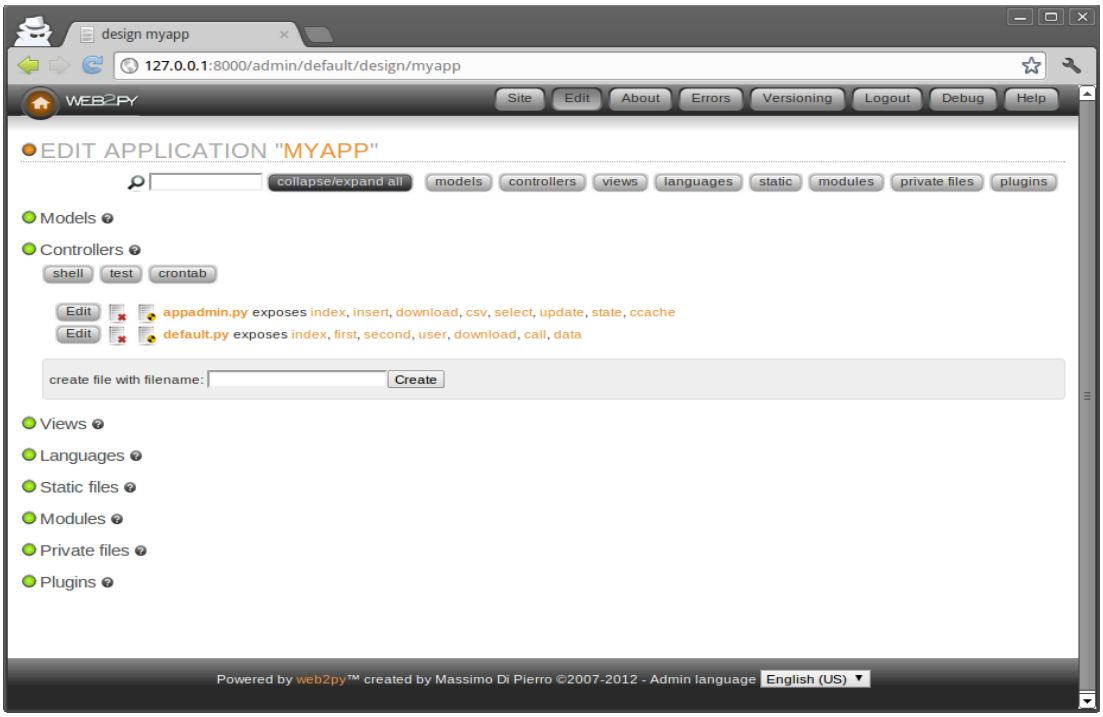
3 KÄYTETYT TYÖKALUT

3.1 Web2py

Web2py on avoimen lähdekoodin ohjelmointiympäristö joka tarjoaa välineet mm. dynaamisten sivustojen rakentamiseen. Web2py on saanut historiansa aikana monia palkintoja. Vuonna 2012 web2py voitti technology of the year-

palkinnon. Web2py:n vahvuuksia on sen monipuolinen työkalutarjonta ja sen helppo käyttöönotto.

Kuvassa 7 näkyy Web2py:n hallintasivu joka tulee näkyviin heti, kun työkalu avataan.

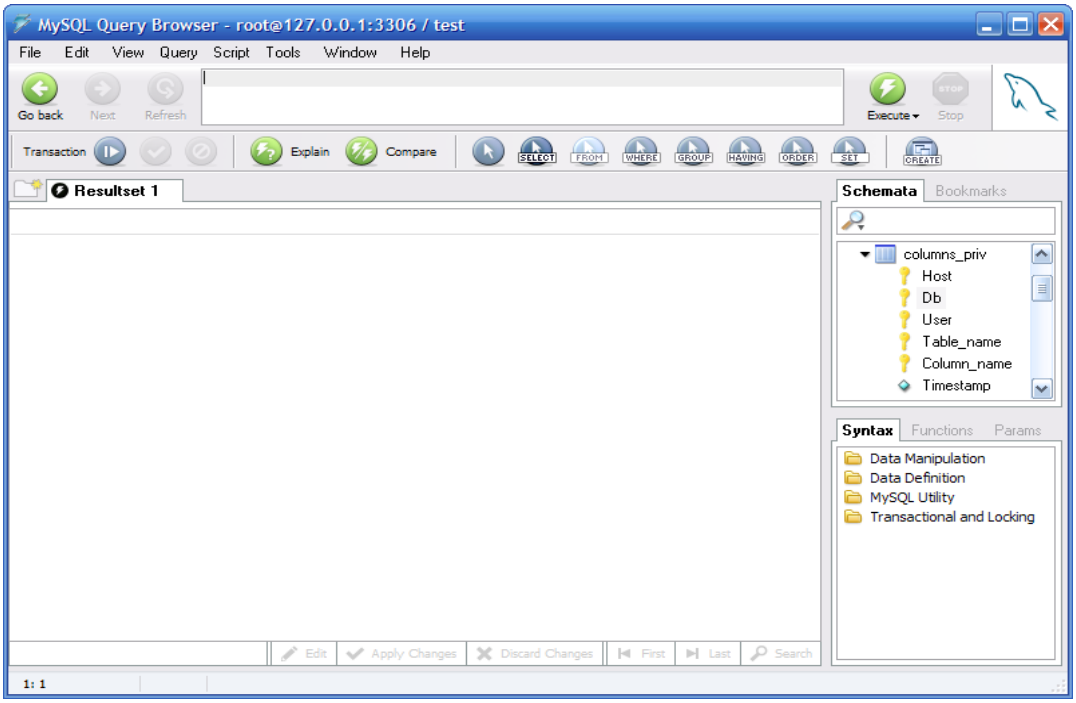


Kuva 7: Web2py hallintasivu

3.2 MySQL

MySQL sovellus on todella suosittu tietokantasovellus, joka tarjoaa kattavat välineet tietokantojen toteuttamista sekä hallitsemista varten. MySQL sovelluksen on kehittänyt ruotsalainen yritys MySQL AB. Kehittämisestä vastasi kaksi ruotsalaista ja yksi suomalainen. Ensimmäinen versio tietokantasovelluksesta julkaistiin 23 toukokuuta 1995.

MySQL query browser ohjelma (kuva 8). Ohjelma tarjoaa graafisen työkalun MySQL tietokantojen käsittelyä varten.

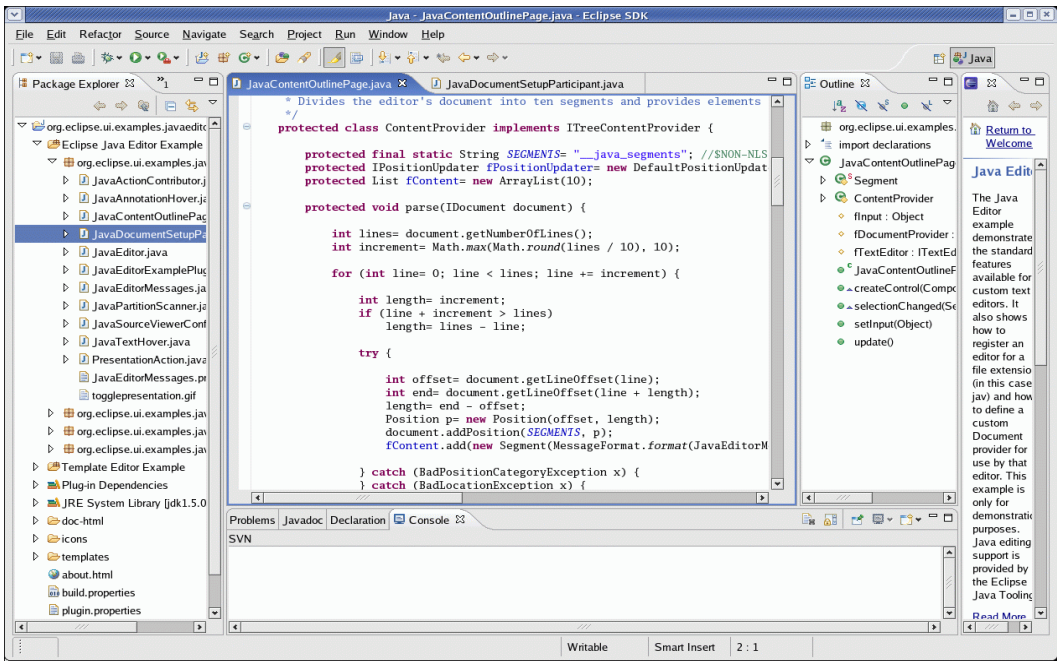


Kuva 8: MySQL query browser

3.3 Eclipse

Eclipse on avoimen lähdekoodin ohjelmointiympäristö, jonka IBM julkaisi marraskuussa vuonna 2001. Eclipse yhdistetään useasti Javalla ohjelmointiin, mutta ohjelma tukee myös monia muita kieliä kuten: C, C++ ja PHP.

Kuvassa 9 näkyy Eclipse sovelluksen Java perspektiivi, jossa näkyy lähdekoodieditorin lisäksi mm. Package Explorer näkymä ja konsoli. Ohjelmointi tapahtuu Java perspektiiviä käyttäen.(Harju 2013, 26.).



Kuva 9: Eclipse sovellus

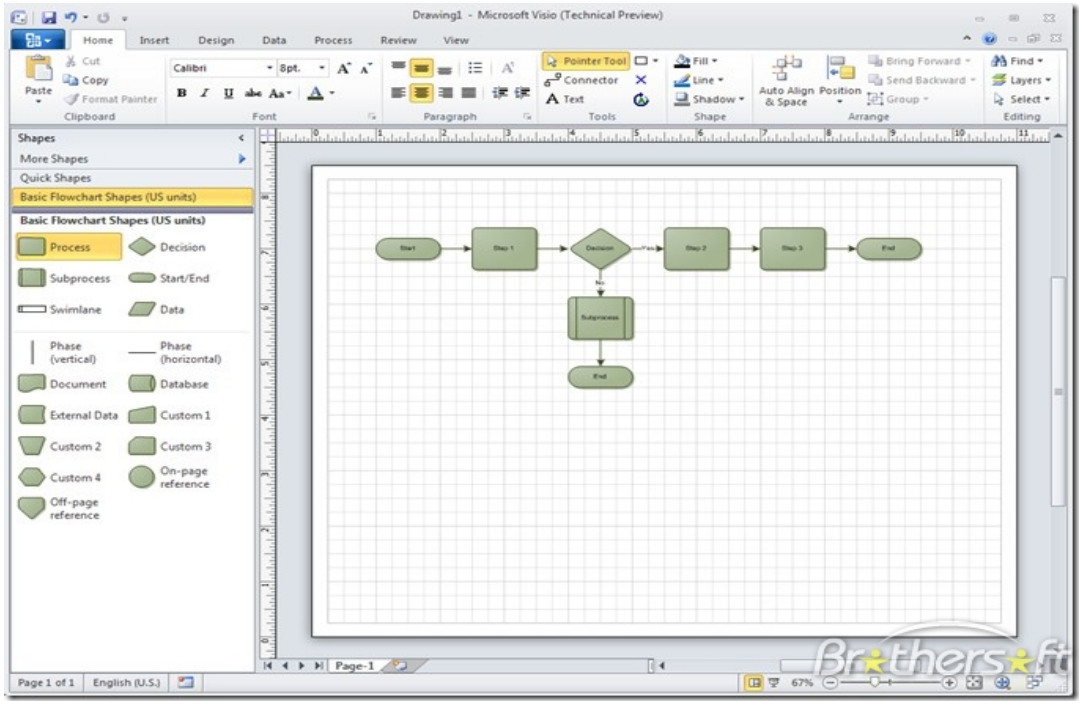
Eclipse foundation perustettiin tammikuussa vuonna 2004. Eclipse foundation on säätiö, joka vastaa Eclipsen kehityksestä.

3.4 MS Visio

MS Visio on osa Microsoft Officen tuoteperhettä. MS Visio on 2D kaaviointiohjelmisto, jolla on tarkoitus suunnitella erilaisia ammattitason kaavioita, kuten Er-kaavioita.

Työkalu helpottaa suuresti ohjelmistojen määrittelyä sekä niiden havainnollistamista. Kuvassa 10 näkyy esimerkki MS Visiolla toteutetusta Er-kaaviosta.

Er-kaaviota käytetään tietokannan määrittelyssä. Sen tarkoitus on identtinen kuvassa 6 näkyvän käsitemallin kanssa, joka on toteutettu UML-luokkakaaviolla.



Kuva 10: MS Visio

4 TOTEUTUS

4.1 Tietokanta

Järjestelmän toteutuksessa ensimmäisenä valmistin tietokannan, jonka toteutin MySql tietokantasovelluksella. Tietokannan toteutus sujui ilman suurempia ongelmia, sillä minulla oli jo aikaisemmin aika selvä käsitys siitä, mitä tietoja tietokantaan tulisi tallentaa.

Tietokanta koostuu seitsemästä taulusta (kuva 11), joita käyn seuraavaksi pääpiirteittäin läpi. Lisäksi kerron taulujen tärkeimpien sarakkeiden tarkoituksista.

Ensimmäisen taulun nimi on *Author*, joka sisältää tiedot järjestelmän kaikista käyttäjistä. Taulu luotiin mahdollistamaan järjestelmän sisällä hallitun ja selkeän kommentoinnin.

Taulun *Author* perusavain on sarake id, joka saa arvonsa automaattisesti. Pääavaimen tarkoituksena on identifioida taulun rivit toisista riveistä. Pääavain ei koskaan saa olla NULL eikä toista samaa arvoa saa esiintyä saman taulun muiden rivien pääavaimissa. Muita sarakkeita taulussa on first_name, last_name sekä email.

Taulu *Test* sisältää tiedot käytettävissä olevista testeistä. Taulun sarakkeita ovat pääavaimena toimiva id ja test_type. Sarakkeeseen test_type tallentuu testityypin virallinen nimi.

Taulussa *Phone* on tiedot jokaisesta testijärjestelmään asennetuista puhelimista. Taulun sarakkeita ovat mm. andr_ver, screen_size, memory, cpu_ghz, manufacturer, model ym.

Phone taulun sarakkeista selviää siis: puhelimessa käytettävä Android versio, näytön resoluutio, muistin määrä, suorittimen nopeus, valmistaja ja puhelimen mallityyppi.

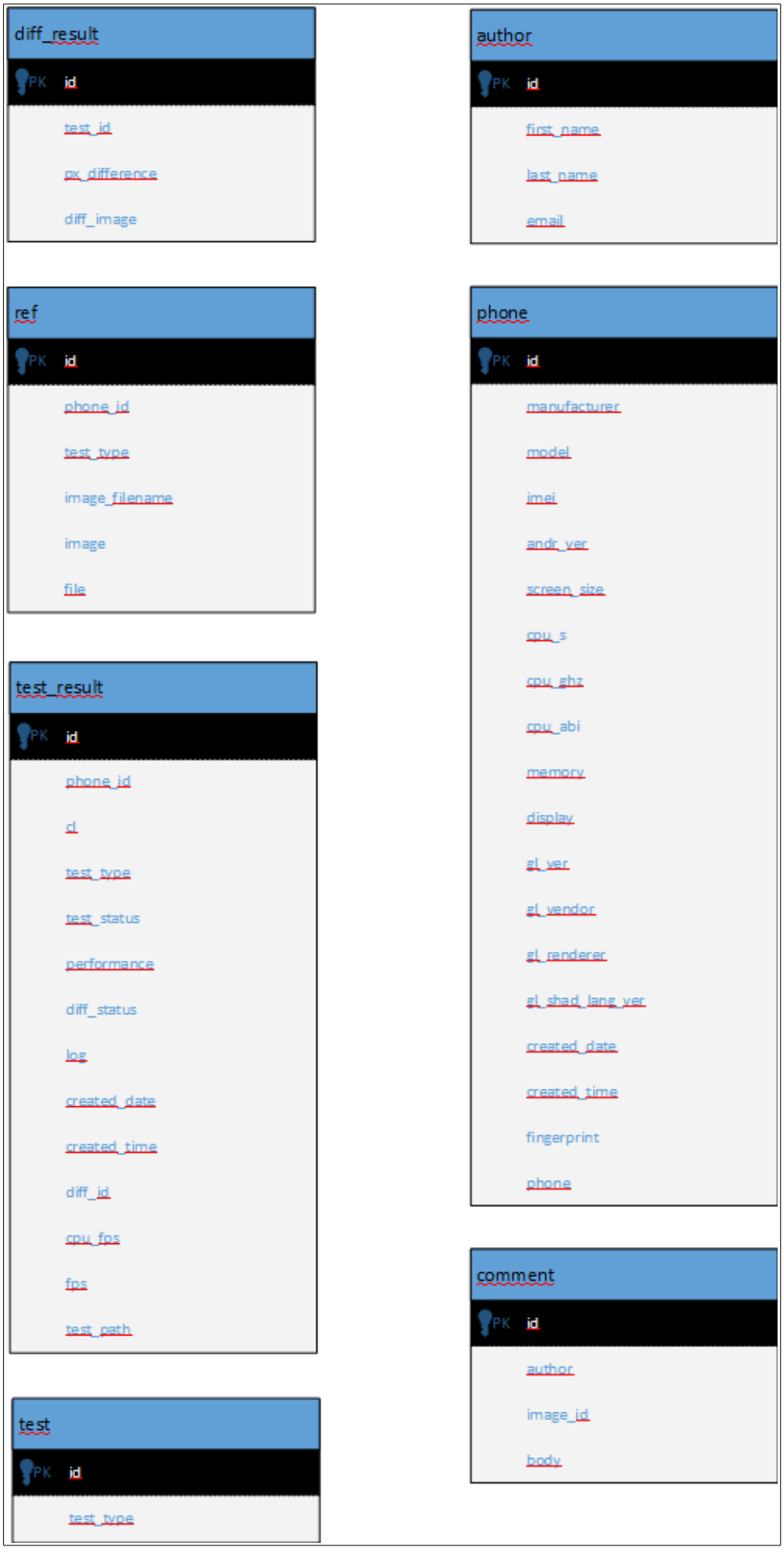
Taulussa *Test_result* on kaikkien suoritettujen testien tulokset. Taulun sarakkeet:

- id (pääavain)
- phone_id joka viittaa phone taulun pääavaimeen

- cl (changelist) kertoo puhelimeen asennetun käyttöliittymän version
- test_type viittaa test taulun pääavaimeen
- test_status kertoo testin tuloksen joka on fail tai pass
- diff_id viittaa diff_result taulun pääavaimeen

Talulussa *Diff_result* näkee testikohtaisesti vertailukuvien tulokset. Taulun sarakkeita on id, test_id, px_difference ja diff_image. Sarakkeessa px_difference näkee kuvien pikselierojen määrän, kun taas sarakkeessa diff_image sijaitsee testistä saadun diff kuvan osoite.

Taulussa *Ref* on jokaisen puhelimen testikohtaiset vertailukuvat, joita käytetään testeistä saatujen kuvien väliseen vertailuun. Viimeisen taulun nimi on *Comment* ja siellä säilytetään käyttäjien kommentteja mm. testeistä.


diff_result
PK id
test_id
px_difference
diff_image
author
PK id
first_name
last_name
email
ref
PK id
phone_id
test_type
image_filename
image
file
phone
PK id
manufacturer
model
imei
andr_ver
screen_size
cpu_s
cpu_ghz
cpu_abi
memory
display
gl_ver
gl_vendor
gl_renderer
gl_shad_lang_ver
created_date
created_time
fingerprint
phone
test_result
PK id
phone_id
cl
test_type
test_status
performance
diff_status
log
created_date
created_time
diff_id
cpu_fps
fps
test_path
comment
PK id
author
image_id
body
test
PK id
test_type

4.2 *Android sovellus*

Tietokannan valmistuttua heräsi kysymys siitä, miten puhelimien tiedot saadaan haettua tietokantaan tallennettavaksi.

Puhelimia koskevien tietojen lisääminen tietokantaan manuaalisesti ei tullut kysymykseen, koska se olisi vienyt liian paljon aikaa ja näin olisi kuluttanut yrityksen resursseja turhaan. Tulin siihen tulokseen, että kyseinen prosessi tulisi suoriutua automaattisesti, kun puhelimet kytketään testijärjestelmään kiinni. Toteutin kyseisen prosessin suunnittelemalla sekä toteuttamalla Android sovelluksen, joka näyttää tärkeimmät puhelimien tiedot.

Ohjelma nimeltään *Device info* asentuu automaattisesti puhelimiin, kun puhelimet kytketään ensimmäistä kertaa testijärjestelmään. *Device info* sovellus käynnistyy sovelluksen asentamisen jälkeen automaattisesti. Koodi 1 asentaa *Device info* sovelluksen puhelimeen, jonka jälkeen se suoritetaan automaattisesti.

```
 def device_info_logcat_callback(str, args):
        if(str.find("DeviceInfo Done") >= 0):
            self.device_info_done = True
            print str+"\n"
self.install(settings.device_info_root + "/DeviceInfo.apk", "com.juvi.Vire")
self.device_info_process=self.start_activity("com.juvi.Vire","Vire",
[("exit", "true")], device_info_logcat_callback)

      print str(self.device_info_done)+"\n"
      while(not self.device_info_done):
         print str(self.device_info_done)+"\n"
         time.sleep(1)
      print str(self.device_info_done)+"\n"
      self.device_info_process.done()
```

Koodi 1

Sovelluksen toiminta suunniteltiin tallentamaan puhelimen tiedot XML tiedostoon puhelimen omaan muistiin. Testijärjestelmä hakee puhelimesta kyseiset tiedot (koodi 2) ja tallentaa tämän jälkeen puhelimen tiedot tietokantaan.

```
self.pull(["/sdcard/myxml.xml"], settings.device_info_root + "/" + str(device) + ".xml")
tree = xml.parse("/home/juha/tech/regressor/device_info/"+str(device)+".xml")
        for node in tree.getiterator():
            if node.tag == "AndroriVersion":
                androidVer = node.text
                print node.tag,  node.text
            elif node.tag == "GlVersion":
                glVersion = node.text
                print node.tag,  node.text
            elif node.tag == "GpuVendor":
                gpuVendor = node.text
                print node.tag,  node.text
            elif node.tag == "GlRenderer":
                glRenderer = node.text
                print node.tag,  node.text
            elif node.tag == "GlShadinLanguageVerion":
                glShadLangVer = node.text
                print node.tag,  node.text
            elif node.tag == "Imei":
                imei = node.text
                print node.tag,  node.text
            elif node.tag == "Manufacturer":
                manufacturer = node.text
                print node.tag,  node.text
            elif node.tag == "Model":
                model = node.text
                print node.tag,  node.text
            elif node.tag == "cpuamount":
                cpuAmount = node.text
                print node.tag,  node.text
            elif node.tag == "Clock_Rate":
                clockRate = node.text
                print node.tag,  node.text
            elif node.tag == "MemTotal":
                memTotal = node.text
                print node.tag,  node.text
            elif node.tag == "Display":
                display = node.text
                print node.tag,  node.text
            elif node.tag == "Cpu_Abi":
                cpuAbi = node.text
                print node.tag,  node.text
            elif node.tag == "Screen_Size":
                screenSize = node.text
                print node.tag,  node.text
            elif node.tag == "Fingerprint":
                fingerPrint = node.text
                self.fingerPrint=fingerPrint
                print node.tag,  node.text
```

Koodi 2: erittellään tiedot XML tiedostosta

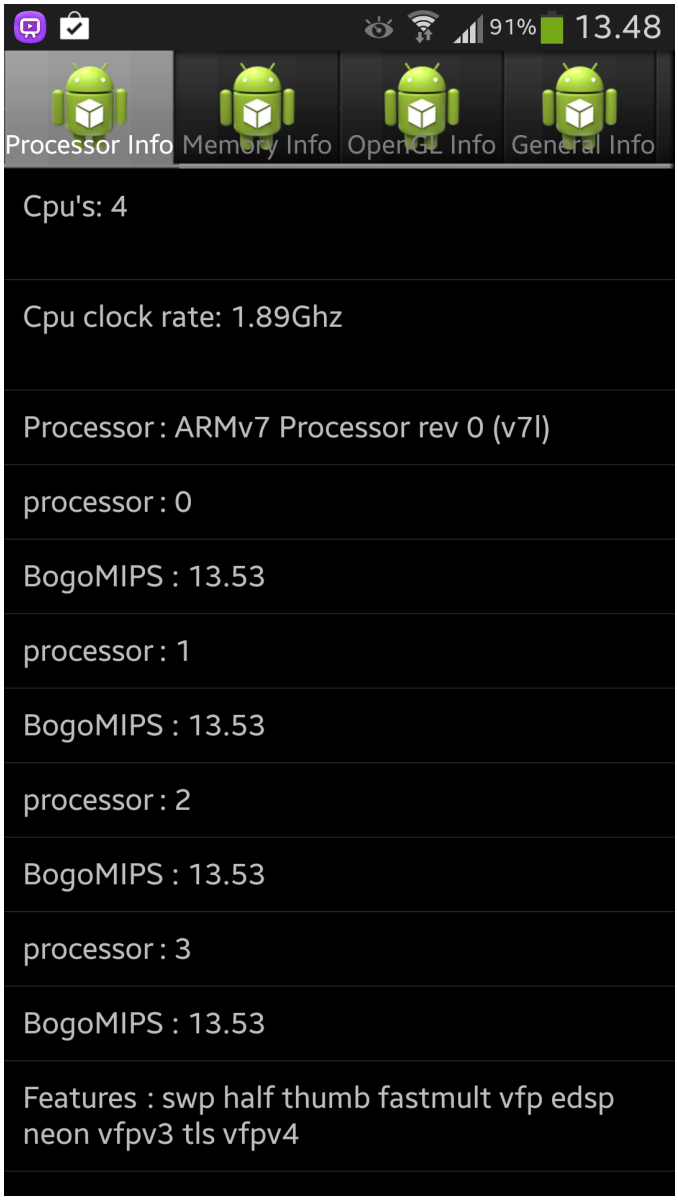
Puhelimien tietoja pidetään myös sen elinkaaren aikana ajan tasalla, sillä jotkut sen tiedoista saattaa muuttua Android versioiden päivitysten yhteydessä. Tämän ratkaisin siten, että ohjelma suoritetaan joka kerta, kun puhelin kytketään uudelleen testijärjestelmään. Jos puhelimeen tallennetusta XML tiedostosta huomataan puhelimen fingerprint arvon muuttuneen kantaan tallenne-

tusta arvosta, niin puhelimen kaikki tiedot korvataan uusilla tiedoilla. Täten toimimalla tietokannassa on aina uusimmat tiedot puhelimista (koodi 3).

```
try:
   con1 = mdb.connect('localhost', 'root', 'xxxx', 'xxxxx');
   cur1 = con1.cursor()
   cur1.execute("SELECT count(*) FROM phone where phone ='"+str(device)+"'")
   amount = cur1.fetchone()[0]

   if amount == 0:
     #puhelinta ei löytynyt asetetaan kantaan
      cur1.execute("INSERT INTO phone (phone, manufacturer, model, imei, andr_ver,
      screen_size, cpu_s, cpu_ghz, cpu_abi, memory, display, gl_ver, gl_vendor, gl_renderer,
      gl_shad_lang_ver, fingerprint, created_date, created_time) VALUES ('"+str(device)
      +"','"+str(manufacturer)+"','"+str(model)+"','"+str(imei)+"','"+str(androidVer)
      +"','"+str(screenSize)+"','"+str(cpuAmount)+"','"+str(clockRate)+"','"+str(cpuAbi)
      +"','"+str(memTotal)+"','"+str(display)+"','"+str(glVersion)+"','"+str(gpuVendor)
      +"','"+str(glRenderer)+"','"+str(glShadLangVer)+"','"+str(fingerPrint)
      +"',CURDATE(),CURTIME())")

   else:
      cur1.execute("SELECT fingerprint FROM phone where phone ='"+str(device)+"'")
      old = cur1.fetchone()[0]

      # tarkistetaan onko fingerprint muuttunut jos on niin päivitetään tiedot
      if old != fingerPrint:
         cur1.execute("INSERT INTO phone (phone, manufacturer, model, imei, andr_ver,
         screen_size, cpu_s, cpu_ghz, cpu_abi, memory, display, gl_ver, gl_vendor,
         gl_renderer, gl_shad_lang_ver, fingerprint, created_date, created_time)
         VALUES ('"+str(device)+"','"+str(manufacturer)+"','"+str(model)+"','"+str(imei)
         +"','"+str(androidVer)+"','"+str(screenSize)+"','"+str(cpuAmount)+"','"+str(clockRate)
         +"','"+str(cpuAbi)+"','"+str(memTotal)+"','"+str(display)+"','"+str(glVersion)
         +"','"+str(gpuVendor)+"','"+str(glRenderer)+"','"+str(glShadLangVer)+"','"+str(fingerPrint)
         +"',CURDATE(),CURTIME())")

   con1.commit()
   cur1.close()

#lopuksi suljetaan yhteys jos yhteys on auki
finally:
        if con1:
            con1.close()
self.runner_thread.start()
```

Koodi 3: tallennetaan tiedot kantaan.

Device info sovellus olisi toiminut myös ilman käyttöliittymää. Halusin kuitenkin tehdä sovellukseen käyttöliittymän (kuva 12), jotta puhelimien tiedot pystyisi tarkistamaan helposti myös manuaalisesti.



Kuva 12: Device info sovellus

Device info sovelluksessa on viisi eri välilehteä. Ensimmäisessä välilehdessä kerrotaan tärkeimmät tiedot puhelimen CPU:sta, kuten ytimien määrän ja CPU:n version. Kuvassa näkyvä BogoMips on Linux kernelin ilmoittama suorittimen nopeus.

Toisella välilehdeltä löytyy tiedot puhelimen muisteista, joista yritystä eniten kiinnosti puhelimen käytettävissä oleva RAM muistin määrä.

Kolmannelta välilehdellä sijaitsee OpenGL tiedot. Tämä sivu kertoo kaikki tärkeimmät GPU:n tiedot, kuten valmistajan ja mallin.

General info välilehdellä kerrotaan puhelimen kaikki yleinen tieto, kuten puhelimen Android versio yms. Sovelluksesta löytyy vielä viides välilehti, jossa on kaikkien edellisten välilehtien tiedot koottu yhteen.

4.3 Sovellus testitulosten seurantaan

Tietokannan ja Android sovelluksen jälkeen täytyi suunnitella itse pääohjelma, josta pystyy tutkimaan ajettujen testien tuloksia, puhelimien tietoja ja hyväksymään/vaihtamaan referenssi kuvia.

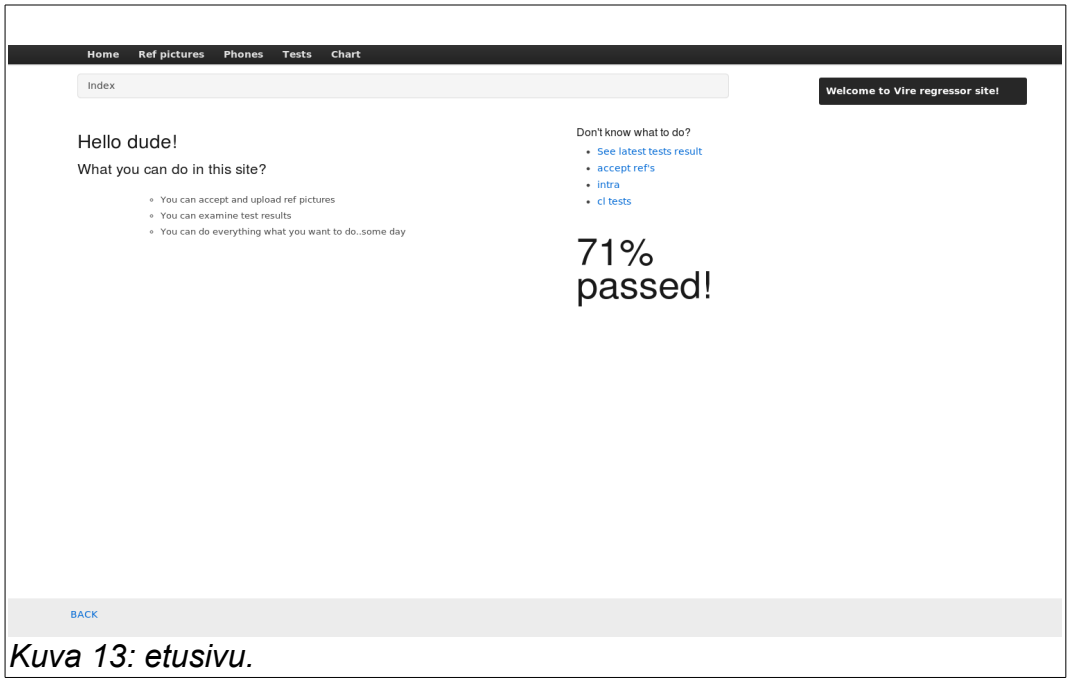
Pääohjelma päätettiin toteuttaa selainpohjaisena, jotta järjestelmän käyttö olisi mahdollisimman helppoa myös työpaikan ulkopuolelta. Sivusto päätettiin toteuttaa Web2py ohjelman avulla. Web2py:n valintaan vaikutti sen viime aikoina saanut positiivinen huomio ja sen voittamat useat palkinnot.

Sivuston toteutus oli projektin selkeästi vaikein ja aikaa vievin vaihe, joka johtui minulle uudesta ohjelmointi välineestä. Myös Web2py:n lyhyt historia ei auttanut tilannetta. Ongelmia aiheutti myös puutteellinen pääohjelman määrittely, joka käytännössä muuttui viikoittain. Toteutettavan ohjelman tarpeet muuttuivat useasti sen takia, kun yrityksen tuotteet kehittyivät nopeasti ja näin ollen vaatimukset ohjelman toiminnoista muuttuivat ja laajenivat jatkuvasti.

4.3.1 Sovelluksen pääsivu

Sovelluksen ”etusivulta” (kuva 13) näkee, kuinka sivujen yläreunassa sijaitsee navigointi valikot, joista pystyy navigoimaan muille sovelluksen sivuille.

Etusivun vasemmalla puolella sijaitsevista linkeistä pääsee suoraan tarkastelemaan viimeisimpien testien tuloksia. Etusivulta on myös nähtävissä kaikkien testien läpimenoprosentti.

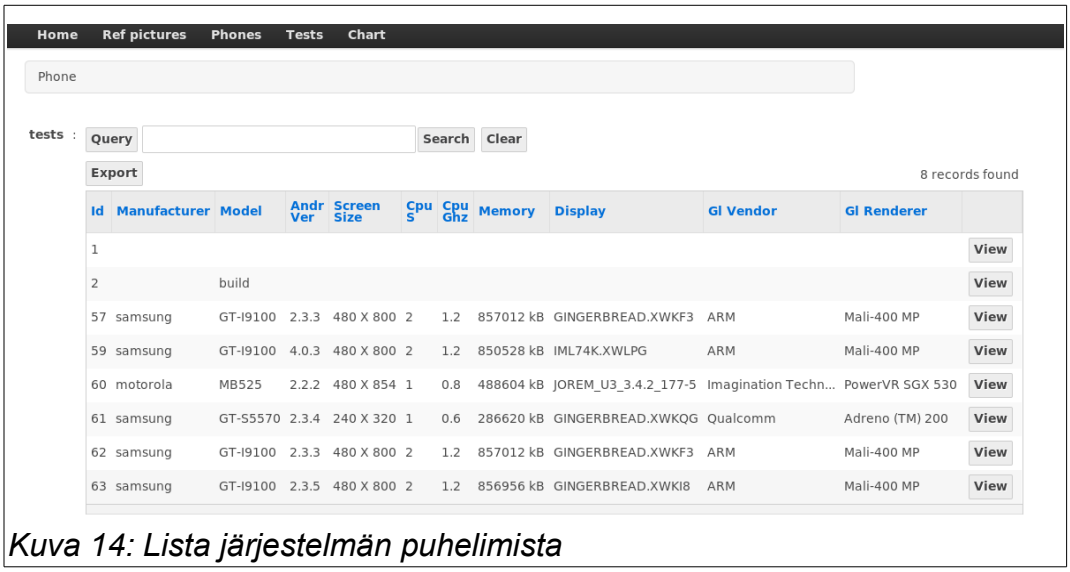


Kuva 13: etusivu.

4.3.2 Puhelimien tiedot

Kuvassa 14 näkyvän sivun kautta pystyy tarkkailemaan kätevästi kantaan tallennettuja puhelimia ja niiden ominaisuuksia. Sivulla on myös kätevä Query toiminto, jonka avulla pystytään tekemään tarkennettuja hakuja puhelimista. Esim. näytä kaikki puhelimet joissa on mali-400 MP:n *GPU*.

Jokaisen rivin perässä näkyvästä *View* painikkeesta painamalla, pääsee tarkastelemaan tarkemmin kyseisen puhelimen tietoja (kuva 15).



Kuva 14: Lista järjestelmän puhelimista

View painikkeen painamisen jälkeen avautuneessa ikkunassa on hyvin näkyvissä puhelimen kaikki oleellinen tieto. Tärkeimpiä tietoja ovat puhelimen valmistaja, mallityyppi, suorittimien lukumäärä ja grafiikkasuorittimen tiedot.

Kuva 15: Näkymä View painikkeen painamisen jälkeen

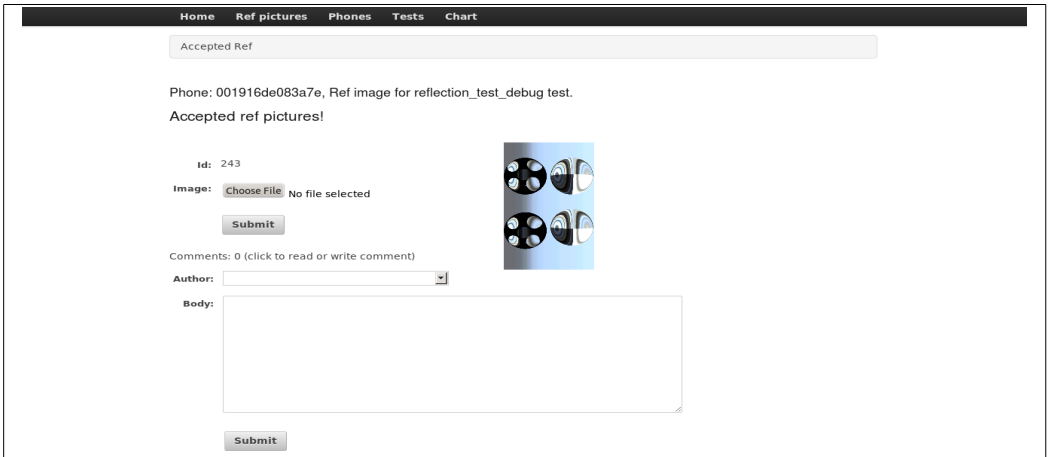
4.3.3 Testien tarkastelu

Useimpien testien tarkoitus on vertailla testien tuottamia kuvia ennalta määriteltyihin kuviin (referenssi kuva), jotta pystytään varmistamaan, ettei tuote ole hajonnut koodimuutosten johdosta.

Aluksi ajatuksena oli, että puhelimilla, joilla on sama resoluutio on myös sama referenssi kuva. Tämä ei kuitenkaan toiminut, sillä puhelimien erilaiset GPU:t aiheuttivat pikselitasolla toisistaan poikkeavia kuvia, jolloin silmämääräisesti ehjä/samanlainen kuva vertailukuvan kanssa nostikin virhe ilmoituksen. Myöskin puhelimet, joissa oli sama GPU, mutta eri fingerprint arvo, tuottivat eri kuvia.

Tästä syystä jokaiselle puhelimelle piti käytännössä tehdä omat vertailukuvat. Ensimmäiset vertailukuvat hyväksytään manuaalisesti ensimmäisellä testauskerralla vertailukuvan hyväksymis sivulla (kuva 16). Jos testin tuottama kuva on jostakin syystä epäkelpo, on tällöin käyttäjän myös mahdollista valita joku valmis kuva omalta tietokoneelta.

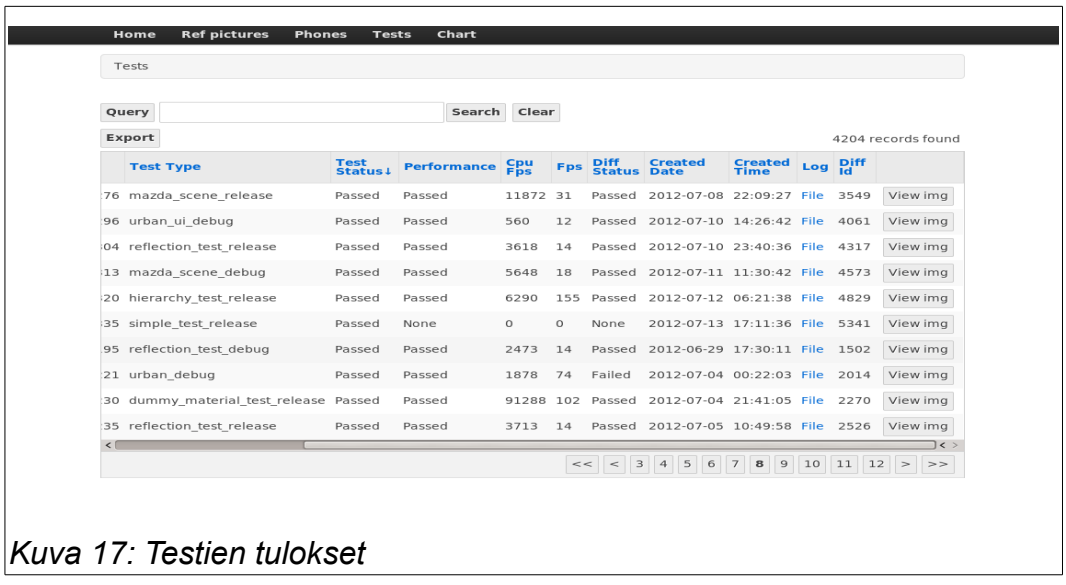
Myös tilanteeseen, jossa kuvia ei pystytä heti hyväksymään, varauduin tekemällä kuvaan kommentointi mahdollisuuden. Tämän kautta järjestelmän käyttäjä voi ilmoittaa esim. kuvanvirheistä tai muista ongelmatapauksista. Vertailukuvan vaihto on myös mahdollista toteuttaa myöhemmässä vaiheessa kyseisen sivuston kautta.



Kuva 16: Vertailukuvan hyväksymis sivu

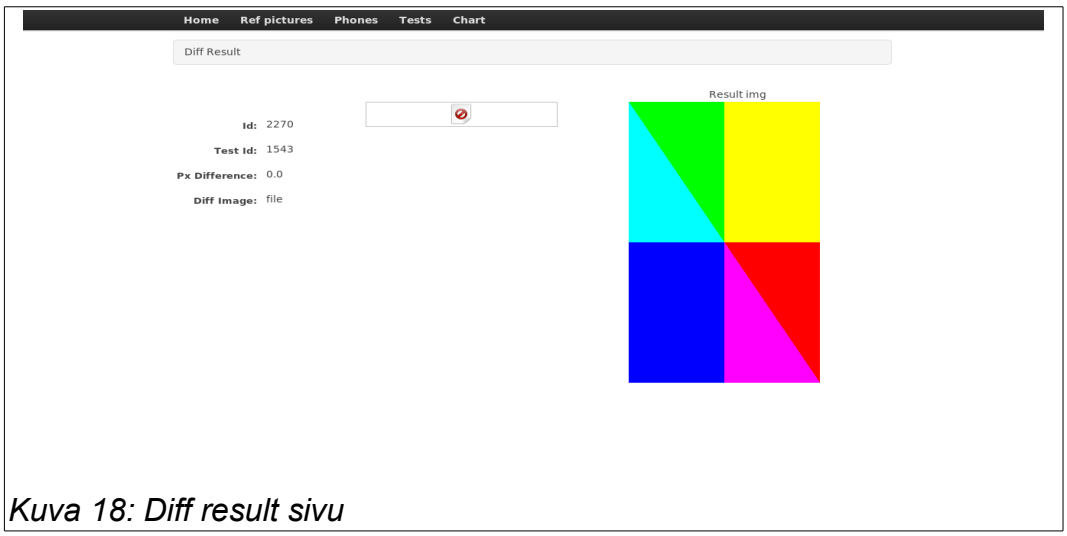
Yksi sivuston tärkeimmistä tehtävistä on kyetä näyttämään testien tulokset. Kuvan 17 sivulta pystyy helposti tarkkailemaan testien tuloksia. Myös tällä sivulla on käytössä Query mahdollisuus taulun järjestämiseen. Taulua pystytään järjestämään myös kentän nimeä painamalla mm. laskevaan tai nousevaan järjestykseen.

Sivussa näkyvästä *View img* painikkeesta painalla pystyy tarkastelemaan paremmin testin tuottaman kuvan vertailua vertailukuvan kanssa.



Kuva 17: Testien tulokset

Kuvassa 18 nähdään esimerkkinäkymä *View img* painikkeen painamisen jälkeen. Sivulla näkyy kaksi kuvapaikkaa, oikealla näkyy testin tuottama kuva, joka vaihtuu referenssi kuvaan sitä klikkaamalla, keskellä taas näkyisi diff kuva, joka näyttäisi kuvien virheet, jos kuvat eroavat toisistaan. Vasemmassa reunassa on nähtävissä mm. pikselierot vertailtavien kuvien välillä, joita tässä tapauksessa ei ole yhtään.



Kuva 18: Diff result sivu

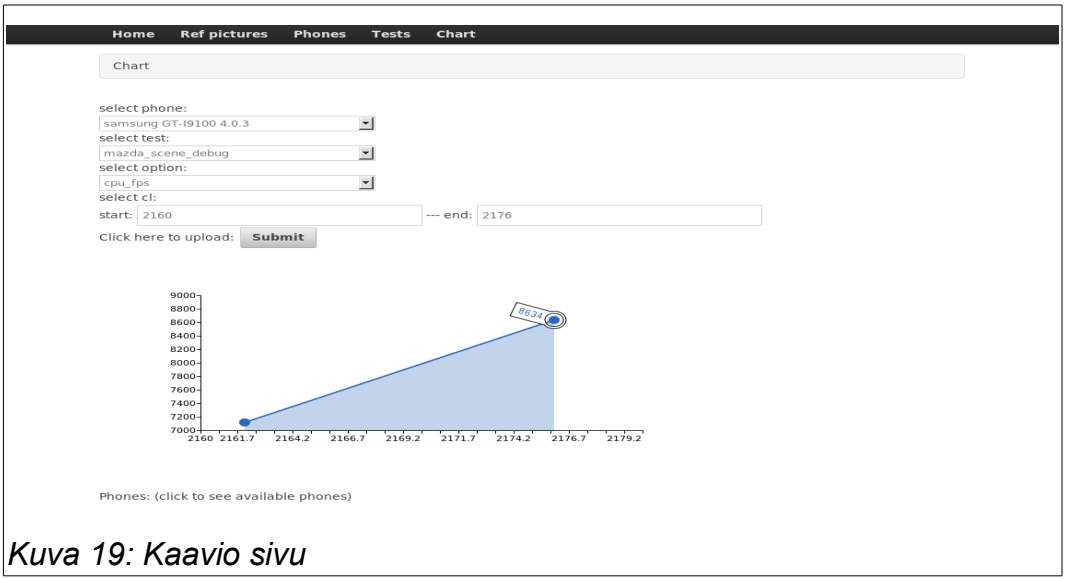
4.3.4 Kehityksen seuranta

Järjestelmän demoaminen herätti yrityksen työntekijöille toiveen sivusta, jolla pystyisi näkemään visuaalisesti, kuinka suorituskyky kehittyy eri tuoteversioiden välillä (kuva 19).

Sivulla on alasvetovalikot, joista voi valita halutun puhelimen sekä käytetyn testin, jonka jälkeen sivulle piirtyy graafinen viivakaavio. Täten viivakaaviosta pystyy tarkkailemaan helposti tuoteversioiden suorituskyvyn kehitystä.

Jatkossa olisi tarkoituksena lisätä sivulle mahdollisuus tarkastella usean puhelimen nopeuksia samalta kaaviolta, jotta pystyttäisiin helposti tarkkailemaan eri puhelinten raudasta johtuvia nopeuseroja.

Kyseinen muutos on tutkimukseni mukaan helppo toteuttaa Google Chartilla, joka on Googlen tarjoama taulukkotyökalu.



Kuva 19: Kaavio sivu

5 JATKOSUUNNITELMIA

5.1 Testijärjestelmän integrointi osaksi versionhallintaa

Toteutin ja suunnittelin yritykselle mm. selainpohjaisen testaustyökalun, jolla pystytään seuraamaan kätevästi testien tuloksia sekä muita testeihin liittyviä tietoja.

Järjestelmä toimii tällä hetkellä vielä ns. koeympäristössä, sillä testijärjestelmä vaatii vielä muutamia muutoksia, ennen kuin tämä selainpohjainen työkalu voidaan huoletta integroida testijärjestelmään.

5.2 Testijärjestelmän selainpohjaisen hallintatyökalun kehitys

Selainpohjaisen työkalun kehittäminen aloitettiin noin vuosi sitten. Olin tuolloin suorittamassa Virelabsilla koulutukseni vaatimaa työharjoittelua.

Kun aikaa on kulunut ja yrityksen tuotteet kehittyneet, on selainpohjaista työkalua jatkokehitettävä täyttämään nykyiset tarpeet. Suurin muutos koskisi järjestelmän päivittämistä ymmärtämään nykyisten testien toimintaa. Ennen testijärjestelmä tulosti yhden kuvan/testi, jota verrattiin referenssi kuvaan.

Nykyään testijärjestelmän täytyisi tulostaa pääsääntöisesti useamman kuvan/testi. Kuvien vertailujen seuraaminen on erittäin tärkeää, jotta voidaan varmistua koko testin oikeellisuudesta.

5.3 Testijärjestelmän jatkokehitys

Järjestelmän yksi tärkeä ominaisuus on kyetä mittaamaan puhelimen virrankulutusta. Tämä mahdollistaisi käyttöliittymämme virrankulutuksen vertailun muiden Android laitteiden käyttöliittymien välillä.

Hyödyllisen virrankulutuksen testaus vaatii toimiakseen myös sovelluksen, jolla pystytään automaattisesti liikkumaan sekä suorittamaan erilaisia toimintoja puhelimen käyttöliittymässä, kuten drag, swipe, press, release, yms. Tämä sovellus tarvitaan, jotta pystytään mittamaan virrankulutusta puhelinta käytettäessä, eikä pelkästään lepokulutusta.

6 YHTEENVETO

Opinnäytetyöni oli omasta mielestäni todella mielenkiintoinen ja haastava, sillä työssäni tarvitsi tehdä ohjelmointia monessa eri ohjelmointiympäristöissä.

Projektin haastavuudesta huolimatta, projekti onnistui mielestäni kohtalaisen hyvin. Työskentely olisi ollut paljon helpompaa, jos oltaisiin kyetty selkeästi esittämään sovelluksen tavoitteet ennen suunnittelun ja ohjelmoinnin aloittamista. Tästä johtuen sovellus jäi hieman keskeneräiseksi. Sovellus tarvitsee vielä hieman lisää toimintoja ja lisää testaamista, jotta sen voi ottaa osaksi virallista testiympäristöä.

Sovelluksen kehittäminen viimeiseen vaiheeseen on tarkoitus saattaa valmiiksi heti, kun yrityksen resurssit sen sallii. Todella tärkeää kuitenkin on, että sovellus pyörii tällä hetkellä testiympäristössä ja antaa tärkeää tietoa kehittäjille. Jälkeenpäin on jäänyt harmittamaan dokumentaation vähäinen määrä, joka olisi kieltämättä voinut olla paljon kattavampi.

LÄHTEET

Hovi 2005: Tietokantojen suunnittelu ja indeksointi, Docendo Finland Oy, Jyväskylä