OULUN SEUDUN
AMMATTIKORKEAKOULU

Teemu Kemppainen

TV-COMPASS

Monikanavainen median toisto

TV-COMPASS

Monikanavainen median toisto

Teemu Kemppainen

Opinnäytetyö

Syksy 2011

Tietotekniikan koulutusohjelma

Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikka, ohjelmistojen kehitys

Tekijä(t): Teemu Kemppainen
Opinnäytetyön nimi: TV-Compass, Monikanavainen median toisto
Työn ohjaaja(t): Pekka Alaluukas
Työn valmistumislukukausi ja -vuosi: Kevät 2011 Sivumäärä: 48

Opinnäytetyössä tutkittiin, kuinka hyvin PhoneGap-sovelluskehys mahdollistaa uusien ja näyttävien mobiilisovellusten luomisen markkinointitarkoituksiin mobiililaitteille. PhoneGap-sovelluskehys mahdollistaa perinteisten web-ohjelmointikielien käytön mobiilisovellusten luonnissa. Tarkoituksena oli saada luotua nopealla aikataululla näyttävä ja toimiva sovelluskokonaisuus, jota voidaan käyttää hyödyksi markkinoinnissa. Opinnäytetyössä keskityttiin kehittämään sovellus iPad-tablet-laitteelle noudattaen Ixonos Oy:n omia sisäisiä projektintoteutusrakenteita.

Työssä on käytetty pääasiassa hyödyksi PhoneGap-sovelluskehyksen kehittäjien omia dokumentaatioita sekä myös HTML5:n ja CSS3:n yleisiä dokumentaatioita. Tutkimusmenetelmät, joita on käytetty sovelluksen kehittämisen aikana ovat noudattaneet loogista järjestystä riippuen siitä, millainen ongelma on missäkin vaiheessa noussut vastaan. Demon jokainen funktionaalinen osa on saanut alkunsa konsepti-ideasta, jonka pohjalta on tutkittu PhoneGap-sovelluskehyksen tarjoamia mahdollisuuksia toteuttaa kyseinen idea käytännössä.

Työn lopullisena tuloksena syntyi TV-Compass-demo iPad-tablet-laitteelle. Demo kokonaisuutena esittelee näyttävästi konseptitasolla luotuja ideoita konkreettisessa ja toimivassa muodossa. TV-Compass on lähtökohtaisesti luotu markkinointitarkoituksiin, jolloin osa demon toiminnallisuuksista on karsittu pois ja jätetty odottamaan jatkokehittelyä. Demo kuitenkin osoittaa hyvin, kuinka monipuoliset ja helpot puitteet PhoneGap-sovelluskehys tarjoaa tällä hetkellä mobiilisovellusten nopeaan ja vaivattomaan kehittämiseen käyttäen hyväksi perinteisiä web-ohjelmointikieliä. Jatkokehityksenä sovelluksen lähdekoodeja on tarkoitus hyödyntää muilla mobiilialustoilla ja selvittää, kuinka hyvin PhoneGap-sovelluskehys loppujen lopulta mahdollistaa lähdekoodien hyödyntämisen usealla eri alustalla.

Asiasanat: PhoneGap, HTML5, CSS3, JavaScript, mobiilisovellukset, mobiililaitteet

ABSTRACT

Oulu University of Applied Sciences
Information technology, Software development

Author(s): Teemu Kemppainen
Title of thesis: TV-COMPASS, Multichannel Media Delivery
Supervisor(s): Pekka Alaluukas
Term and year when the thesis was submitted: Spring 2011 Number of pages: 48

The thesis examines what kind of opportunities does the PhoneGap mobile framework offer for the developers to create new and visually impressive mobile applications or demos for marketing purposes. The PhoneGap mobile framework allows traditional web programming languages to be used in creation of native mobile applications. The main idea in the project was to create a mobile application demo under a tight schedule, and see what kind actual opportunities would the PhoneGap application framework offer for developers to create visually impressive and functional demos for marketing purposes. This thesis focuses on developing the application for the iPad tablet device with the PhoneGap application framework. During the project Ixonos Plc's own internal project structures where used for the applications development.

During the application development the PhoneGap mobile frameworks developers own documentation was the main resource for learning about the framework and figuring out some of the problems that were encountered during the process. Also the HTML5 and CSS3 general documentations were used for acquiring information about the new features that both HTML5 and CSS3 offer. Research methods used during the application development followed a logical sequence. Depending on what kind of problem was encountered.

The final result of the whole project was the TV-Compass demo for the iPad tablet. The demo presents the concept ideas that were the starting point for the whole project in their functional and actual form on a mobile device. In principle the whole TV-Compass demo has been created mainly for marketing purposes. Mainly because of this some of the features where left out of the first demo version for later development. However the demo really shows what kind of new opportunities the PhoneGap mobile framework currently offers for mobile application development. For developers the framework offers new faster and easier ways to develop mobile applications by just using traditional web programming languages. For further development the main idea is to use the source codes made for the iPad, and see how well do they actually work on the other platforms that the PhoneGap mobile framework supports.

Keywords: PhoneGap, HTML5, CSS3, JavaScript, mobile applications, mobile devices

SISÄLLYS

TERMIT

CSS	Cascading Style Sheets, porrastetut tyyliarkit
HTML	Hypertext Markup Language, kuvauskieli
iScroll 4	Mobiileja web-sovelluksia varten kehitetty JavaScript-kirjasto
JavaScript	Web-ympäristössä käytettävä komentosarjakieli
jQuery	Selain riippumaton JavaScript-kirjasto
Laitteistokiihdytys	Käytetään laitteen erillistä ydintä tietyn toiminnon nopeampaan suorittamiseen.
Natiiviohjelmointikieli	Kehitysympäristön alustakohtainen ohjelmointikieli

1 JOHDANTO

Nykyaikana mobiililaitteet ovat yhä enemmän osa jokapäiväistä elämäämme. Ihmiset ovat oppineet kritisoimaan käyttämiänsä laitteita sekä sovelluksia ja tämän johdosta jyvät alkavat nopeasti erottua akanoista. Loppukäyttäjät kiinnittävät enemmän huomiota laitteiden ja sovellusten varmatoimisuuteen sekä laatuun. Kehitysaikataulujen nopeutumisen ja laatuvaatimusten kovenemisen myötä on noussut yhä tärkeämmäksi kyky toteuttaa nopealla aikataululla laadukasta jälkeä. Yhdeksi suureksi ongelmaksi tässä vaiheessa kuitenkin nousee eri mobiililaitealustojen hajanaisuus ja eroavaisuus toisistaan. Tämä luonnollisesti hidastaa kehitysprosesseja, koska yhdestä tuotteesta joudutaan toteuttamaan monta erillistä versiota. Tässä vaiheessa ratkaisuksi on kehitelty erilaisia sovelluskehyksiä, jotka mahdollistavat sovellusten nopean toteuttamisen usealle eri mobiililaitealustalle samanaikaisesti.

Sovelluskehykset ovat nousseet viimeisimpien vuosien aikana mobiilisovelluskehittäjien tietoisuuteen, juuri kehittämisen helppouden ja nopeuden ansiosta. Sovelluskehyksien avulla luodut sovellukset ovat eräänlaisia hybridejä, jotka vaikuttavat natiivisovelluksilta, mutta pinnan alla ovat jotain aivan muuta. PhoneGap-sovelluskehys valittiin osaksi opinnäytetyötä pitkällisten keskusteluiden ja tutkimusten pohjalta. PhoneGap teoreettisella tasolla vaikuttaa muihin sovelluskehyksiin verrattuna huomattavasti kehittyneemmältä, ja sen tarjoamat mahdollisuudet hyödyntää myös mobiililaitealustan omaa natiiviohjelmointikieltä raskaampien toimintojen suorittamiseen tekevät siitä kaikista mielenkiintoisimman vaihtoehdon nopean mobiilisovellusten kehittämisen kannalta. Luonnollisesti tässä vaiheessa voi herätä kysymys, eroaako tällainen käytäntö loppupeleissä normaalista mobiilisovellusten kehittämisestä laisinkaan, koska jokainen alusta vaatii omat natiiviohjelmointikieleen pohjautuvat laajennuksensa.

Lyhykäisyydessään sovelluskehyksen hyödyntäminen mobiilisovelluksen kehittämisessä nopeuttaa prosessia, koska tällä tavoin kaikki ulkoasulliset toiminnot ja graafiset hienoudet pystytään toteuttamaan yhdellä koodilla ja jokaisella mobiilialustalle tarvitsee kehittää vain omat natiivilaajennuksensa. Periaatteessa sovelluksen ulkokuori pystytään käyttämään sellaisenaan jokaisella alustalla ja vain sovelluksen taustalla pyörivät raskaammat natiivitoiminnot kehitetään uudelleen. Luonnollisesti jokainen mobiililaite on oma kokonaisuutensa ja vaatii omat käyttöliittymä viilailunsa, mutta tässä tilanteessa korostuu myös hyvä käyttöliittymäsuunnittelu.

Opinnäytetyö käsittelee TV-Compass-sovelluksen luontia pelkästä konsepti-ideasta valmiiseen käyttöliittymädemoon saakka. Samalla kuitenkin pyrin myös mahdollisimman laajasti käsittelemään ja tutkimaan sovelluskehysten luomia sovellusten kehitysmahdollisuuksia ottaen huomioon, kuinka pirstaloitunut nykyaikainen mobiilisovellusten kehitys on ja kuinka paljon se vaati resursseja kehityksen kannalta. Tällä hetkellä sovelluskehykset tarjoavat varteenotettavan kilpailijan nopeasti kehitettävien ja kevyiden mobiilisovellusten luonnissa. Opinnäytetyön lopputuloksena valmistunut demo on näyte siitä, kuinka nopeasti ja tehokkaasti pelkästä konseptitason ideasta saadaan aikaiseksi näyttävä ja toimiva sovelluskokonaisuus markkinoinnin tueksi.

2 PHONEGAP-SOVELLUSKEHYKSEN RAKENNE

Lähtökohtaisesti koko projekti aloitettiin lähes tyhjästä. Pohjalla oli yksi hyvä idea, joka oli tarkoitus saada laajennettua toimivaksi kokonaisuudeksi. Alkuperäinen markkinoinnin kehittämä idea koko projektille oli yksinkertainen kysymys, kuinka nykyaikaisilla mobiililaitteilla pystyttäisiin laajentamaan ihmisten normaalia nykyaikaista television katselu kokemusta. Yksi hyvä idea poiki nopeasti uusia ideoita, ja tätä kautta projekti eteni nopeasti perusteellisen tutkimus- ja suunnitteluvaiheeseen. Samanaikaisesti käyttöliittymäsuunnittelija loi ensimmäisiä pohjapiirroksia tulevan TV-Compass-demon käyttöliittymästä. Itse syvennyin sopivien sovelluskehitystyökalujen etsimiseen. Tärkeimmät kriteerit sovelluskehitystyökalujen ja -metodien tutkimisen suhteen olivat nopeus ja varmatoimisuus. PhoneGap-sovelluskehys löytyi tutkinnan alkuvaiheessa ja ensimmäiset kokeilut sovelluskehyksen kanssa antoivat lupaavia tuloksia kehittämisen nopeudesta ja helppoudesta. Tutkimus- ja suunnitteluvaiheen lopussa kaikki konseptitason ideat käytiin vielä kertaalleen lävitse ja tärkeimmät ja toteuttavissa olevat ideat nostettiin muiden ideoiden yläpuolelle.

2.1 PhoneGap ja sovelluskehykset

Koko opinnäytetyön alkuperäinen idea oli toteuttaa demo Android-alustalle natiiviohjelmointikielellä. Omista taustoistani johtuen tutkin kuitenkin muita mahdollisia toteutusratkaisuja. Nopealla etsinnällä törmäsin PhoneGap-sovelluskehykseen, joka mahdollisti uusien HTML5-, CSS3- ja JavaScript-ohjelmointikielien hyödyntämisen natiivisovellusten luonnissa. Itselleni web-ohjelmointi on ollut pitkään koulun ohella eräänlaisena harrastuksena, joten sen ansiosta PhoneGapin tarjoamat mahdollisuudet hyödyntää tehokkaasti jo vakaalla pohjalla olevia ohjelmointitaitoja web-ohjelmoinnin saralta antoivat itselleni sen lopullisen sysäyksen syventyä sovelluskehyksen maailman syvällisemmin.

Ensimmäiset kunnolliset testi-demot PhoneGapilla sain luotua muutamassa tunnissa, ja tästä rohkaistuneena aloitin näiden demojen pohjalta esittelemään löytöäni eteenpäin. Yhden palaverin jälkeen sain hyväksynnän sille, että voisin käyttää PhoneGap-sovelluskehystä demon toteuttamisessa, koska PhoneGap mahdollistaa iPadilla luotujen koodien tehokkaan hyödyntämisen muillakin alustoilla. PhoneGap itsessään on pitkän linjan tuote, joka on ollut kehitteillä jo useamman vuoden ja sen tarkoituksena on ollut saada mobiilisovellusten kehitys mahdollisimman helpoksi jokaiselle joka on joskus päässyt näkemään tai ohjelmoimaan web-sivuja. Tässä

vaiheessa en itse joutunut juurikaan käyttämään aikaa syntaksin opiskeluun, vaan pystyin välittömästi aloittamaan työskentelyni, ja minulle esitettyjen konsepti-ideoiden toteuttamisen ensimmäiseen valmiiseen muotoonsa, jonka avulla käyttöliittymäsuunnittelija pystyisi toteamaan oliko hänen alkuperäinen ideansa sittenkään toimiva.

2.2 PhoneGapin toiminnallisuus ja ohjelmointikielet

PhoneGap tarjoaa web-ohjelmoijalle mahdollisuuden hyödyntää perinteisiä web-ohjelmointikieliä ja -ohjelmointimenetelmiä sovelluksensa ulkoasun ja toimivuuden rakentamiseen. Periaatteellisella tasolla PhoneGap-sovelluskehyksen avulla luotu mobiilisovellus on nettisivu, joka pystytään sovelluskehyksen avulla esittämään natiivisovelluksena käyttäjälle. On jo pitkään ollut mahdollista kehittää mobiililaitteille web-tekniikoilla alustakohtaisia web-sivuja, mutta PhoneGap tarjoaa tärkeimmät natiivitoiminnallisuudet ohjelmoijalle hyödynnettäväksi ja käytettäväksi oman web-koodinsa seassa. Tämän vuoksi PhoneGapin tarjoamat mahdollisuudet mobiilisovellusten kehittämisessä ovat niin lupaavat.

2.2.1 HTML5

HTML5 on viimeisin versio HTML-standardista, joka mahdollistaa web-sivujen kehittämisen ja luomisen. HTML5-standardi on ollut kehitteillä jo vuodesta 2004 lähtien, mutta vasta parin viimeisen vuoden aikana se on tullut voimakkaammin esille, koska HTML5-standardin lopullisen version myötä nykyisin laajasti käytössä oleva FLASH-multimedia-alusta tulee jäämään tarpeettomaksi web-ohjelmoinnissa. HTML5 mahdollistaa laajemmat ja kattavammat mahdollisuudet luoda selkeää ja helposti ymmärrettävää koodia, kuten esimerkistä 1 on helppo nähdä. Samalla myös mahdollistuu tehokkaampi multimedian hyödyntäminen web-sivuilla. Samaa tahtia multimedian monimuotoisemman käyttämisen kanssa ovat myös kehittyneet web-sivujen ulkoasut. Pitkään on luonnollisesti ollut käytössä erinäisiä mahdollisuuksia luoda näyttäviä web-sivuja, mutta HTML5-standardin myötä näyttävyyden luominen ei ole enää niinkään ohjelmoijasta kiinni, vaan jokaisen omasta mielikuvituksesta. (HTML5. 2011b.)

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello HTML</title>
```

```
  </head>
  <body>
    <p>Hello World!</p>
    <p>Here's a video for you.</p>
    <video poster="movie.jpg" controls>
      <source src='movie.webm' type='video/webm; codecs="vp8.0, vorbis"'/>
      <source src='movie.ogv' type='video/ogg; codecs="theora, vorbis"'/>
      <source src='movie.mp4' type='video/mp4; codecs="avc1.4D401E, mp4a.40.2"'/>
      <p>This is fallback content</p>
    </video>
  </body>
</html>
```

ESIMERKKI 1. HTML-koodin perusrakenne

Nykyaikana graafiseen ulkoasuun panostaminen on luonnollisesti täysin ymmärrettävää. Näyttävyys on laadun ja toimivuuden mittari. HTML5:n myötä web-ohjelmoijat pystyvät yhä paremmin vastaamaan nykypäivän vaatimuksiin ja tarpeisiin alati kehittyvässä web-ohjelmoinnin maailmassa. Standardien kehittyessä ja päivittyessä syntyy uusia ongelmia, joiden kanssa web-kehittäjät joutuvat kamppailemaan päivittäin. Ongelmaksi ovat nousseet Internet Explorer -selaimen vanhemmat versiot, jotka edelleen ovat suurimmalla osalla tietokoneiden käyttäjistä oletusselaimena. Internet Explorer 9 ja sitä uudemmat versiot kuitenkin jo tukevat uutta HTML5-standardia. Tämä luonnollisesti aiheuttaa suuret määrät ongelmia, koska vanhat selainversiot eivät tue tai ymmärrä uusien standardien kehittäjille tarjoamia elementtejä. Oman opinnäytetyöprojektissani voin onneksi huokaista helpotuksesta, koska iPad-tablet-laite tarjoaa parhaan mahdollisen tuen ja kehitysalustan HTML5 tekniikoiden hyödyntämiseen. Vaikka HTML5-standardi on suuressa roolissa koko sovelluksen kehittämisen aikana, niin silti loppujen lopuksi HTML5 tarjoaa pelkästään pinnallisen kuoren, jota tullaan koristelemaan ja käsittelemään käyttämällä hyväksi CSS3- ja JavaScript-ohjelmointikieliä. (HTML5. 2011a.)

2.2.2 CSS3

CSS3-tyylikieli on kehitteillä oleva uuden sukupolven tyylikieli joka mahdollistaa aikaisempaa monipuolisemman ja selkeämmän web-sivujen ulkoasun kuvailun ja määrittämisen. Tyylikielen avulla määritellään sivustolle, miltä sivuston tulisi näyttää ja samalla myös erotellaan itse sisältö ulkoasusta selkeäksi kokonaisuudeksi. Tyylikielellä määritellyt ehdot eivät ole täysin ehdottomia, ja tämä aiheuttaa sen, että osa selaimista

saattaa tulkita tyylitiedostoja hieman poikkeavalla tavalla. Alkuperäinen ensimmäinen versio CSS-tyylikielestä kehitettiin jo vuonna 1996, ja tästä eteenpäin tyylikieli on kehittynyt nopealla tahdilla versiosta toiseen. Nopeasti kehityksestä huolimatta esimerkissä 2 esitelty CSS-tyylikielen perussyntaksi on pysynyt samanlaisena. Tällä hetkellä kehitteillä on versiot CSS2.1 ja CSS3. CSS2.1 on CSS2-tyylikielen korjailtu ja päivitetty versio joka on toiminut jo pitkään pohja standardina web-sivujen ulkoasujen määrittelemiselle. CSS3 taas on vielä radikaalisti kehitteillä oleva tyylikieli, mutta sen tarjoamat uudet määrittely mahdollisuudet ovat nopealla tahdilla ajaneet web-sivustojen kehittäjät ja myös selainkehittäjät siirtymään CSS3 tukemiseen. (CSS3. 2011b.)

```
Valitsin {
  Ominaisuus: Arvo;
}

/* Leipätekstin fontin määritys ja tasaus */

p {
 font: 11px 'Trebuchet MS', Verdana, Arial, Sans-serif;
 color: #9e9e9e;
 text-align: justify;
}
```

ESIMERKKI 2. CSS-koodin perusrakenne

Opinnäytetyön kannalta CSS3 tyylikieli mahdollistaa demon graafisen ulkoasun toteutuksen sekä myös sen laitteistokiihdytteisen 3D-animaation. CSS3 itsessään ei ole mitään muuta kuin keino helposti saada luotua kaikki tarvittavat ulkoasuelementit näkyviin. Jokainen ulkoasuelementti on sidottu HTML5:llä kirjoitettuun runkoon. Runkoon kiinnitetyille elementeille myös tarpeen vaatiessa suoritetaan animaatioita käyttäen hyväksi WebKitin tarjoamia mahdollisuuksia. CSS3:n suurin etu verrattuna aikaisempiin versioihin on mahdollisuus saada luotua graafiset animaatiot laitteistokiihdytystä hyväksi käyttäen ja tästä johtuen sivuista saadaan sulavampia ja näyttävämpiä, koska animaatioita ei tarvitse luoda pelkästään käyttämällä JavaScript-ohjelmointikieltä. Sovelluskehyksen kanssa yhdistettynä CSS3 antaa kehittäjille todella kattavat työkalut siihen, kuinka sovelluksista saadaan jo tässä vaiheessa näyttävämpiä ja ulkoasullisesti omaperäisempiä kuin natiivisovellukset. (CSS3. 2011a.)

WebKit on Apple Inc:n Kehittämä KHTML-kirjastoon pohjautuva selainmoottori joka toimii Safari- ja Chrome -selainten ytimenä. Sama selainmoottori löytyy iPad-tablet-laitteesta. Selainmoottorin tärkein ja ainut tehtävä on noutaa haluttu data verkosta ja tulkita noudettua yleensä HTML- ja CSS-kielistä data näkyväksi web-sivuksi. WebKit-selainmoottori pohjautuu avoimeen lähdekoodin ja tästä johtuen kyseinen selainmoottori on näyttänyt muutamien viime vuosien aikana vahvuutensa CSS3-tyylikielen kanssa. WebKit on myös alati kehittyvä ja kasvava selainmoottori, joka tarjoaa web-kehittäjille uusia ulottuvuuksia web-sivujensa ulkoasujen luontia varten. (WebKit. 2011a; WebKit. 2011b.)

2.2.3 JavaScript

JavaScript on oliopohjainen komentosarjakieli, joka mahdollistaa dynaamisen sisällön luonnin ja hallinnan web-sivuilla. JavaScriptin perus syntaksi pohjautuu löyhästi C-ohjelmointikieleen. JavaScriptin kehitys on aloitettu jo vuonna 1995 ja siitä eteenpäin JavaScript on kehittynyt ja laajentunut huimaa tahtia. JavaScriptin suurin hienous onkin sen helppo laajennettavuus. Nopean kehittymisensä ja laajenemisensa ansiosta tänä päivänä JavaScript on yksi tärkeimmistä web-ohjelmointi kielistä. (JavaScript. 2011.)

Opinnäytetyössä JavaScriptilla hallitaan ja kontrolloidaan käyttäjälle näytettäviä sisältöjä. Ilman JavaScriptilla luotua kontrollia olisi koko demo pelkästään HTML-runko, johon olisi CSS:llä sijoiteltu graafisia elementtejä ilman minkäänlaisia toiminnallisuuksia. Opinnäytetyössä on käytetty perinteisen JavaScriptin lisäksi myös jQuery- sekä iScroll 4 -kirjastoja, jotka laajentavat ja helpottavat JavaScriptin käyttömahdollisuuksia demon luonnissa. (JavaScript. 2011.)

jQuery

jQuery-kirjasto on selainriippumaton JavaScriptin perusohjelmointisyntaksia laajentava ja selkeyttävä avoimen lähdekoodin kirjasto. jQueryn suurimmat hyödyt ohjelmoijalle ovat elementtien käsittelyn helppous ja selkeys. Kirjaston käyttö selkeyttää JavaScript-koodin luettavuutta sekä mahdollistaa helpomman HTML-elementtien käsittelyn.

```
//div_social-elementin leveyden saaminen el-muuttujaan jQuerylla
var el = $('#div_social').width();
//div_social-elementin leveyden saaminen el-muuttujaan JavaScriptilla
var el = document.getElementById('div_social').offsetWidth;
```

ESIMERKKI 3. jQueryn ja JavaScriptin perusohjelmointisyntaksin vertailu

Opinnäytetyössä jQuery-kirjastoa on käytetty koodin selkeyttämisen. Valitettavasti osa funktioista ja elementtien käsittelyistä piti suorittaa ilman jQueryn tarjoamaa apua ja hyötyä, koska tietyissä tilanteissa jQuery-kirjasto aiheutti huomattavan viiveen toimintojen suorittamisessa. Näissä tilanteissa tuotti perinteisen JavaScriptin käyttö halutun lopputuloksen. (jQuery. 2011a; jQuery. 2011b.)

iScroll 4

iScroll 4 -kirjasto on kehitetty web-pohjaisten mobiilisovellusten sisällön natiivinomaista kontrollointia varten. WebKit-pohjaiset mobiilisovellukset eivät alkuperäismuodossansa tarjoa tapaa, jolla loppukäyttäjä pystyisi selaamaan suuria sisältömääriä pienellä tarkkaan rajatulla alueella käyttäen hyväksi iPad-tablet-laitteen kosketustapahtumia. Yhdistämällä iScroll 4 -kirjasto PhoneGap-sovelluskehyksen kanssa mahdollistetaan suurien sisältölistojen laitteistokiihdytteinen 3D-animaatio ja natiivien kosketustapahtuminen hyödyntäminen. iScroll 4 -kirjasto on vielä kehityksen alla, mutta tarjoaa tällä hetkellä jo todella kattavat mahdollisuudet luoda näyttäviä ja monimuotoisia kokonaisuuksia web-pohjaisten mobiilisovellusten sisällön käsittelyyn.

Opinnäytetyön kannalta iScroll 4 -kirjasto mahdollistaa natiivin kaltaisen sisällönkontrolloinnin ja käyttötuntuman. Kokonaisuuden kannalta tämä on hyvin kriittistä, koska ilman oikeanlaista käyttötuntumaa olisi lopputuloksena syntynyt demo jäänyt puutteelliseksi, ja olisi ollut ilmiselvää ettei luotu demosovellus ole natiivi. (iScroll 4. 2011.)

2.3 TV-Compass-demon lähtökohdat

TV-Compass-demon ensimmäinen ja tärkein kriteeri oli pystyä esittelemään markkinointitilanteessa selkeästi ja näyttävästi luodut konseptit konkreettisessa muodossa mahdolliselle asiakkaalle. Demo esittelee loppukäyttäjän kannalta yleisimpiä käyttöskenaarioita ja näkymiä. Alkuperäisistä konsepteista luonnollisesti jouduttiin

karsimaan joitain osia pois, koska ne todettiin turhiksi. Tärkeää oli myös saada demoon mukaan natiivisovelluksen kaltainen käytettävyys ja tuntuma. Luonnollisesti demoon haluttiin myös mukaan HTML5:n tarjoamia uusia multimediaelementtejä ja näitä myös demossa hyödynnetään konseptien selkeyttämiseksi. Tärkeimmät konseptit, jotka selviytyivät lopullisen karsinnan lävitse, ovat juuri niitä piirteitä jotka palvelevat selkeästi ja näyttävästi markkinoinnin tarpeita.

2.3.1 Demon ulkoasun rakentaminen

Kokonaisuutena demon ulkoasut tuli saada luotua pikselitarkasti niiden grafiikkapohjien mukaisesti, jotka graafikko minulle toimitti demon kehityksen alkuvaiheessa. Demon lopullinen ulkoasu on toteutettu kokonaisuudessaan hyödyntäen CSS3:lla luotua vektorigrafiikka ja graafikon minulle toimittamia kuvia. Hyödyntämällä vektorigrafiikkaa normaalin grafiikan seassa saatiin demo-sovelluksesta huomattavasti kevyempi välimuistin käytön kannalta ja nopeampi toiminen. Sovelluksen ulkoasun peruslogiikka pohjautuu näkymien kerrostamiseen, eli jokainen demon elementeistä ladataan demon käynnistyksen yhteydessä valmiiksi omaan sitovaan näkymäänsä ja tämän jälkeen sijoitetaan oikeaan tasoon muihin näkymiin nähden. Elementtien ulkoasumääritykset sisälsivät myös elementtien animaatioiden alkuarvot.

2.3.2 Demon sisällönhallinta

Vaikka demo olisi kuinka visuaalisesti vakuuttava ja toimiva se ei juuri hyödytä mitään, ellei visuaalista sisältöä pystytä kontrolloimaan sujuvasti. Demon kerrostettua kokonaisuutta hallitaan JavaScript-funktioiden avulla. Yksinkertaisimmat funktiot määritelevät elementtien animaatioiden alkuarvoja uudelleen koska alkuarvon muuttaminen haluttuun arvoon laukaisee välittömästi CSS3:lla elementille määritellyn animaation. Jokainen skenaario, joka demoon on luotu käyttäjän nähtäväksi ja kokeiltavaksi, käsitellään aina JavaScript-funktion kautta. Sisällönhallinta on periaatteellisella tasolla hoidettu mahdollisimman yksinkertaisesti ja tätä myöten myös mahdollisimman resurssitehokkaasti, koska opinnäytetyön kehitysalustana toimi kuitenkin rajallisilla resursseilla varustettu mobiililaite.

3 DEMON KEHITYS- JA TOIMINTAYMPÄRISTÖ

Alkuperäinen idea oli toteuttaa TV-Compass-demo kokonaisuudessaan natiivi ohjelmointikielellä Android-alustalle, mutta ajankäyttösyistä demo päädyttiin toteuttamaan iOS-alustalle PhoneGap-sovelluskehystä hyödyntäen. Alustan vaihto tuotti tiettyjä muutoksia ja lisävaatimuksia sovelluksen kehittämisen kannalta, mutta kaikesta huolimatta iOS-alustan ja iPad-tablet-laitteen tarjoamat puitteet demo-sovelluksen toteuttamisen kannalta olivat parhaat mahdolliset.

3.1 iOS

iOS on Applen kehittämä mobiililaitekäyttöjärjestelmä, joka alun perin oli käytössä vain iPhone-puhelimessa, mutta ajan saatossa sitä on laajennettu tukemaan myös muita Applen kehittämiä laitteita, kuten esimerkiksi iPod-musiikkisoittimet, iPad-tablet-laitteet sekä myös Apple TV. iOS-käyttöjärjestelmä on suljettu ainoastaan Applen omien tuotteiden käyttöön. Käyttöjärjestelmän käyttö perustuu suoraan kosketukseen, eli käyttöjärjestelmää ja sovelluksia käytetään suoraan koskettamalla laitteen näyttöä. Tällä tavoin itse laitteesta saadaan käyttöjärjestelmän ansiosta mahdollisimman helppo ja nopea käyttää, koska laite ja käyttöjärjestelmä reagoivat välittömästi käyttäjän tekemiin painalluksiin. Käyttöjärjestelmään on myös sisällytetty tiettyjä erikoiseleitä, joilla pystytään innovatiivisemmin kontrolloimaan sovelluksia.

iOS pohjautuu voimakkaasti Applen kehittämään Mac OS X -pöytäkonekäyttöjärjestelmäänsä. Kumpikin käyttöjärjestelmä pohjautuu Darwin-ytimeen, ja tämän ansiosta käyttöliittymät ovat pohjimmiltaan Unix-pohjaisia. Kumpikin käyttöjärjestelmä pohjautuu samaan ytimeen, mutta eri prosessorialustan takia eivät sovellukset ole suoraan yhteensopiva alustakohtaisesti. Tämän johdosta Mac OS X:lle kehitetty sovellus tulee vähintään kääntää uudelleen iOS SDK:n kautta, jotta sovellus toimisi iOS-laitteissa.

Opinnäytetyön kannalta iOS-käyttöjärjestelmä tarjoaa vakaan kehitysympäristön, joten sovelluksen kehittämisen aikana ei tarvitse tuhlata aikaa käyttöjärjestelmän ja kehitysympäristön aiheuttamien ristiriitojen ratkomiseen. Luonnollisesti joitain pieniä ongelmia ilmeni sovelluksen kehityksen aikana, mutta suurin osa näistäkin ongelmista johtui enimmäkseen PhoneGap-sovelluskehyksen aiheuttamista pienistä ristiriidoista ja sovelluskehyksen keskeneräisyydestä. Opinnäytetyössä käytettiin testaus- ja kehitysvaiheessa iOS:n 4.2 versiota. (iOS. 2011.)

3.2 iPad

iPad on Applen kehittämä taulutietokonemallisto, joka on tällä hetkellä edennyt jo toiseen versioonsa. Ensimmäinen versio laitteesta julkaistiin vuoden 2010 alussa, ja laitteen pääasiallinen tarkoitus on toimia multimediatoistimena erinäisille multimediaformaateille. Fyysisiltä ominaisuuksiltaan laite sijoittuu älypuhelinten ja kannettavien tietokoneiden välimaastoon. Laitteen käyttöjärjestelmänä toimii aikaisemmassa kappaleessa kuvailtu iOS-käyttöjärjestelmä, joka löytyy jokaisesta Applen kehittämistä mobiililaitteesta. Ainoa ero iPadin käyttöjärjestelmä versiossa muihin Applen mobiililaiteisiin verrattuna on se, että käyttöjärjestelmä on muokattu tukemaan laitteen suurempaa näyttöä ja resoluutiota. Tästä johtuen iPhone- ja iPad-sovellukset toimivat sellaisinaan eri laitteiden välillä, mutta luonnollisesti sovellusten skaalaus tulee ottaa huomioon jos sovelluksen halutaan olevan mahdollisimman helppo käyttää kummassakin laitteessa. (iPad. 2011.)

iPad luottaa vahvasti iOS-käyttöjärjestelmän kosketuspohjaisuuteen ja ottaa laitteen käyttötavasta kaiken mahdollisen hyödyn irti. Laite on suunniteltu käytettäväksi ilmaan mitään erillistä kosketuslaitetta tai muuta vastaavaa, joten laitteesta itsessään löytyy virtuaalinäppäimistöt ja muut tarvittavat toiminnot, jotta käyttäjä pystyy helposti käyttämään laitettansa. Demon kehitysvaiheessa käytettiin iPadin ensimmäistä versiota testaus- ja kehitysalustana, mutta myöhemmässä vaiheessa demon esitys ja testaus päätettiin siirtää iPad 2 -laitteeseen. Tämä siitä syystä että iPad 2:n tarjoamat resurssit ja tehot mahdollistavat vielä näyttävämmän ja sulavamman TV-Compass-demon esittelyn. (iPad. 2011; iPad 2. 2011.)

Laitteiden välillä ei ole suuriakaan eroavaisuuksia. Näytön koko ja resoluutio ovat täysin identtiset. Näyttönä toimii kummassakin laitteessa 9,7 tuuman monikosketusnäyttö, joka toimii 1024 x 768 pikselin resoluutiolla. Laitteiden fyysisissä mitoissa ei myöskään ole suuria eroja. iPad 2 on edeltäjäänsä verrattuna fyysisiltä mitoiltansa hieman kapeampi ja ohuempi. Suurimmat eroavaisuudet laitteiden välillä löytyvät prosessori- ja muistipuolelta. Ensimmäinen iPad-versio on varustettu yksiytimisellä 1 GHz:n A4 -prosessorilla, kun taas iPad 2 on varustettu kaksiytimisellä 1 GHz:n A5 prosessorilla, joka mahdollistaa huomattavasti tehokkaamman ja nopeamman sovellusten ajamisen. Alkuperäinen iPad oli myös varustettu ainoastaan 256 MB DDR RAM muistia, kun taas iPad 2:ssa muistin määrä on tuplattu ja se on muutettu DDR2 RAM muotoon, jolloin muistiväylä on saatu nostettua 1066 MHz:n. Muilta osin laitteet ovat lähes täysin identtiset, eli sisäisen tallennustilan määrä on

sama kummassakin laitteessa, kuten myös tekniikat joilla laiteet yhdistyvät internetiin. Tärkeimmät ja merkittävimmät eroavaisuudet löytyvät näiden kahden laite version välillä tehopuolelta. (iPad. 2011; iPad 2. 2011.)

Opinnäytetyön kannalta oli hyvä aloittaa kehitys käyttämällä pelkästään iPadin ensimmäistä versiota, koska tämä pakotti miettimään ja kehittämään koodia mahdollisimman resurssiystävälliseksi. Luonnollisesti tämä aiheutti omat ongelmansa, koska laitteen resurssit eivät alkuvaiheessa tuntuneet millään riittävän demon ajamiseen, mutta yllättävistä paikoista löytyi pieniä keinoja nipistää hieman lisää tehoa, ja suorittaa demoa huomattavasti järkevämmin itse laitteessa. Lopulta demoa päätettiin myös testata iPad 2:lla, ja testin tulokset olivat todella positiiviset, koska aikaisemmin toteutetut animaatiot, jotka oli optimoitu ensimmäiselle iPadille toimivat silmiä hivelevän sulavasti iPad 2:lla.

3.3 Xcode

Xcode on Applen kehittämä ohjelmointityökalu Mac OS X- ja iOS-käyttöjärjestelmissä suoritettaville sovelluksille. Ohjelmointityökaluna Xcode tarjoaa OS X- tai iOS-sovelluskehittäjälle kaikki tarvittavat työkalut, niin projektimanagerin, debuggerin, iOS -emulaattorin kuin myös mahdollisuuden kääntää omaa koodiansa eri koodikielille sekä Applen hyväksymille ja tukemille alustoille. Xcode on kehittynyt jo elinkaarensa ensimmäiseen maksulliseen versioonsa 4.0, mutta sovelluksesta on myös saatavissa vielä vanhempi versio ilmaiseksi, ja tämän ilmaisversion viimeisin versio Xcode 3.2.6 oli käytössä koko demon kehityksen ajan.

Xcode kokonaisuutena sisältää sovelluskehitys ympäristön, Applen sovelluskehitys dokumentaatiot ja sekä myös graafisen käyttöliittymärakentajan. Xcode sisältää muokatun version GNU-kääntäjäkokoelmasta GCC, apple-darwin9-gcc-4.2.1 sekä myös apple-darwin9-gcc-4.0.1, joista jälkimmäinen on toiminut oletuksena. Xcode tukee tämän ansiosta seuraavia koodikieliä C, C++, Objective-C, Objective-C++, Java, AppleScript, Python, Ruby, Cocoa, Carbon ja Java. Kolmannen osapuolen kehittäjät ovat luoneet Xcodelle tuet myös seuraaville ohjelmointikielille: GNU Pascal, Free Pascal, Ada, C#, Perl, Haskell, ja D. Xcoden suurin hienous löytyy kuitenkin sovelluksen mahdollisuudesta kääntää lähdekoodeja useille eri prosessoriarkkitehtuureille. Xcodella kehittäjä pystyy kääntämään sovelluksesta prosessorikantakohtaisen version, kuten myös joko 32-bittisen tai 64-bittisen version riippuen siitä, mitä kehittäjä haluaa. Periaatteessa yhden koodin pohjalta pystytään

luomaan uusi sovellus jokaiselle Applen tarjoamalle laitealustalle pelkästään kääntämällä koodi uudelleen ja valitsemalla haluttu prosessorikanta ja kehitysympäristö. (Xcode. 2011.)

Opinnäytetyössä Xcode toimi loistavana sovelluksen kehitystyökaluna, tarjoamalla ensiluokkaiset koodinkääntö työkalut, sekä mahdollisuuden testata luotua koodia nopeasti emulaattorissa. Itselleni Xcoden tarjoama käyttöliittymä ei koodin kirjoittamisen kannalta ollut se paras mahdollinen, joten suoritin koodin kirjoittamisen TextWrangler-tekstinkäsittelyohjelmalle. Tästä johtuen käytin Xcodea enimmäkseen pelkästään sovelluksen testaamiseen ja mahdollisten vikojen etsimiseen.

3.4 Muut käytetyt ohjelmisto

Luonnollisesti koska kyseessä on visuaalisuuteen panostava markkinointidemo, jouduin useaan otteeseen käyttämään kuvanmuokkausohjelmistoja, jotta sain pilkottua valmiiksi luoduista pohjista haluamani näytettävät elementit oikeaan kokoonsa ja muotoonsa. Mainitsin myös aikaisemmin, että käytin koodin kirjoittamiseen erillistä ohjelma, ja tämä vain sen takia, että tarvitsin omien tottumuksieni puolesta selkeämmän ja nopeammin käytettävissä olevan kevyen tekstinkäsittelyohjelman. Myös JavaScriptin tarkempaa tulkintaa varten tarvitsin erillisen selaimessa ajettavan liitännäisen, koska web-pohjaista koodia oli huomattavasti helpompi tulkita ja muokata suoraan selaimen kautta saatavan palautteen pohjalta.

3.4.1 Adobe Photoshop ja Adobe Illustrator

Adobe Photoshop ja Illustrator ovat molemmat ammattitason kuvankäsittely- ja -muokkausohjelmistoja, ja kumpikin näytti opinnäytetyön aikana parhaat puolensa kuvaelementtien muokkauksessa sekä luonnissa. Kumpikin sovelluksista on saanut alkunsa jo 80-luvun lopulla ja pitkä kehityskaari näkyy jo tässä vaiheessa selkeästi. Kummallakin ohjelmistolla on ilo työskennellä ja vain mielikuvitus on rajana siinä vaiheessa, kun grafiikkaa saa luoda vapaasti ilman rajoitteita. Illustrator oli demon kehityksen alkuvaiheessa pääasiallisena grafiikanmuokkaustyökaluna, koska projektin ensimmäisen graafikon minulle lähettämät grafiikkapohjat olivat sillä luotuja ja grafiikoita oli huomattavasti helpompi ja nopeampi käsitellä Illustratorilla verrattuna siihen, että olisin yrittänyt käsitellä niitä Photoshopilla. Luonnollisesti vektorigrafiikan tarjoamat skaalausmahdollisuudet olivat hyödyksi kehityksen alkuvaiheessa ja käyttöliittymän selkeys nopeutti sovelluksen pohjan rakentamista. Demon kehityksen

loppuvaiheessa osa grafiikoista korvattiin CSS3:lla luoduilla vektorigrafiikoilla, jotta sovelluksen keskusmuistin käyttöä pystyttäisiin paremmin kontrolloimaan.

Myöhemmässä vaiheessa projektia graafikko vaihtui ja tässä vaiheessa jouduin siirtymään Phohoshopin maailmaan, joka itselleni ei sekään ollut vieras, mutta Illustratorin selkeyden ja helppouden puolesta Photoshopiin siirtyminen hieman aluksi hidasti kuvaelementtien luontia ja muokkaamista, mutta lopulta pystyin aivan yhtä nopeaan työskentelyyn kuin aikaisemmassa vaiheessa. Koko projektin onnistumisen kannalta laadukkaiden kuvanmuokkausohjelmistojen käyttö oli edellytys, koska grafiikka joutui todella intensiivisellä tahdilla muokkaamaan ja luomaan uutta tarpeen vaatiessa. (Adobe Illustrator. 2011; Adobe Photoshop. 2011.)

3.4.2 TextWrangler

TextWrangler on Mac-alustan vastike suositulle NotePad++ -tekstinmuokkaus ja -editointiohjelmistolle. TextWrangler ei anna käyttäjällensä mahdollisuutta muokata tekstin ulkoasua, mutta lähdekoodien kirjoittamisen kannalta TextWrangler tarjoaa laajat tuet erinäisille lähdekoodi syntakseille. TextWrangler on Bare Bones Softwaren valmistama kevyt versio yrityksen päätuotteestansa BBEdit. TextWranglerin suurin hyöty löytyy sen yksinkertaisesta ja käytännöllisestä käyttöliittymästä, joka mahdollistaa nopean ja helpon lähdekoodi tiedostojen käsittelyn ja hallinnoinnin. Ohjelmistoa on myös mahdollista laajentaa erinäisillä lisäosilla ja näiden avulla monia ohjelmointikieliä pystytään ajamaan suoraan TextWranglerista, eikä erillistä kääntäjää näin ollen tarvita. (TextWrangler. 2011a; TextWrangler. 2011b.)

Opinnäytetyön kannalta TextWranglerin tärkeimmät hyödyt olivat sovelluksen keveys ja nopeasti navigoitava käyttöliittymä. Usean laajan lähdekooditiedoston muokkaaminen Xcodella vaati useiden erillisten ikkunoiden välillä liikkumista, kun taas TextWranglerilla kaikki lähdekooditiedostot löytyivät välilehtien takaa, ja tämä takasi huomattavasti nopeamman ja tehokkaamman työskentelyn. Myös TextWranglerin tarjoamat hyödyt web-pohjaisten ohjelmointikielien syntaksien kanssa helpotti tietyissä tilanteissa virheiden paikantamista, mutta lopulta suurimmat toiminnalliset virheet joutui aina etsimään koodin suorittamisen aikana.

3.4.3 Safari-selaimen kehitystyökalu Web Inspector

Web Inspector on WebKit-kirjaston yhteydessä luotu web-kehitystyökalu WebKit-selainmoottoria käyttäviä selaimia varten. Web Inspectorin avulla web-sivustojen

kehittäjät pystyvät vaivattomasti tutkimaan web-sivujen elementtejä ja elementteihin liittyviä graafisia määrityksiä. Web Inspectorin tärkein ominaisuus on kyky paikantaa virheitä JavaScript-koodista. Tämä antaa kehittäjälle mahdollisuuden reaaliaikaisesti testata käyttämiänsä JavaScript-funktioita. Web Inspector löytyy Googlen Chrome-selaimesta sisäänrakennettuna, mutta Safarille kehitystyökalu pitää ottaa käyttöön manuaalisesti. (Hatcher. 2006.)

Web Inspector mahdollistaa myös web-sivuston koodin suoran muokkaamisen ja mahdollisten muutosten välittömän päivittymisen näkymään. Tämä on HTML-elementtien CSS-määrityksien asettamisessa hyödyllinen ja aikaa säästävä toiminto. Hyödyllinen toiminto joka osoittautui tarpeelliseksi koodin resurssien käyttöä tutkittaessa, oli Web Inspectorin aikajanatyökalu, josta kehittäjä pystyy näkemään millisekunnin tarkkuudella kuinka kauan selaimelta menee ladata jokin sivuston tietty osa. Tämä millisekuntien tarkkailu ja pohtiminen alkoi demon kehityksen loppuvaiheessa nousta tärkeäksi elementiksi, koska iPadin rajalliset resurssit aiheuttivat omia ongelmiansa, ja tämä johti optimointitoimenpiteisiin. (Safari Features. 2011.)

Opinnäytetyön kannalta Web Inspector nousi tärkeimmäksi työkaluksi koko demon kehityksen aikana, koska kaikki toiminnallisuus ja näkyvyys on toteutettu web-ohjelmointikielillä. Tästä johtuen demoa oli alkuvaiheessa helppo kehittää luottaen pelkästään siihen mitä selain näyttää. Myöhemmässä vaiheessa jouduin luonnollisesti luopumaan kokonaan selaimen hyödyntämisestä, koska PhoneGapin laitealustakohtaiset komennot eivät suostuneet yhteistyöhön selaimessa, joten jouduin hyödyntämään huomattavasti enemmän Xcodea ja sen emulaattoria. Loppujen lopulta ilman Web Inspectorin kaltaista työkalua olisi työskentelyni ja virheiden löytäminen ollut huomattavasti hitaampaa ja tuskallisempaa.

4 TV-COMPASS-DEMON TOTEUTUS

4.1 HTML5 – demon runko

TV-Compass-demon runko koostuu pohjimmiltaan HTML5-koodisyntaksin kahdesta erillisestä pääosasta. <head>-osio sisältää kaikki sivustoon liittyvien muiden tiedostojen määrittelyt, ja tässä tapauksessa myös osan PhoneGap-sovelluskehyksen vaatimista perustiedoista, jotta sovellus varmasti tulisi toimimaan oikein mobiililaitteessa. <body>-osio taas sisältää sovelluksen runkorakenteen, joka toimii pelkkänä pohjana CSS3:lla ja JavaScriptilla suoritettavaa käsittelyä varten. Sovelluksen runkorakenne on hyvin yksinkertainen ja helposti ymmärrettävissä. Runkorakenne on jaoteltu sovelluksen erillisten näkymin mukaan ja jokaisen näkymän sisälle on luotu omat tarvittavat elementtinsä. Koko sovelluksen runko pohjautuu eri tasojen käsittelyyn ja niiden näkymien ja erillisten toiminnallisuuksien rakentamiseen käyttäen hyödyksi web-ohjelmointi tekniikoita. HTML5:llä luotu pohjarunko on vain yksinkertainen runkorakenne jossa kerrotaan jokaisen elementin järjestys sovellusrungon sisällä.

4.1.1 Aloitusnäkymä

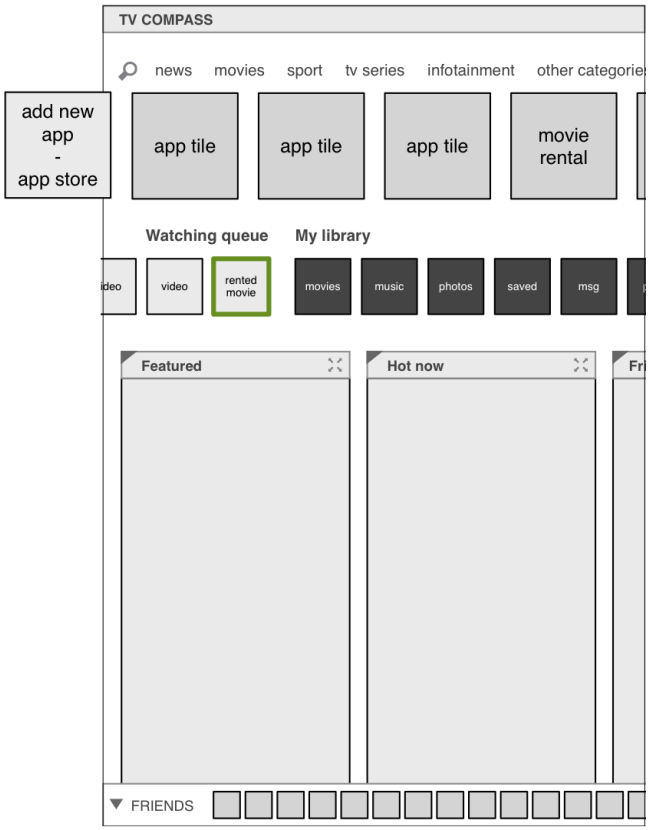
Sovelluksen runkorakenne alkaa sovelluksen sisäänkirjautumisnäkymästä, joka on otettu erilleen muusta kokonaisuudesta, koska ensimmäisen alkunäkymän jälkeen näkymää ei enää tarvita ja näkymä poistetaan käytöstä. Sisäänkirjautumistaso pitää sisällänsä latausruudun pohjagrafiikat, joiden päälle on luotu yksinkertainen sisäänkirjautumisnäkymä, ja tässä vaiheessa käyttäjältä pyydetään sisäänkirjautumistiedot. Sovelluksen demoluonteesta johtuen näkymä ei vaadi mitään muuta kuin napin painalluksen, jonka jälkeen sisäänkirjautuminen on näennäisesti suoritettu ja itse sovelluksen ensimmäinen taso tuodaan näkyville. Alun alkaen näkymää ei pitänyt tulla mukaan demoon, mutta loppuvaiheessa sille ilmeni tarve, ja loin näkymän käyttöön nopeasti. Näkymän luonnin aikana ilmeni tiettyjä pieniä ongelmia, jotka aiheutuivat sovelluskehyksen tarjoamasta toimintaympäristöstä.

Ongelmat olivat päällisin puolin graafisia ja johtuivat sovelluksen selainpohjaisuudesta, mutta ongelmat oli mahdollista kiertää estämällä käyttäjää koskemasta muuhun kuin pelkkään sisäänkirjautumispainikkeeseen. Tekstikentät merkittiin vain luku -formaattiin, jolloin kyseiseen tekstikenttään ei voi kirjoittaa tai tekstikenttä ei anna käyttäjälle fokusta, joka taas laukaisisi tämän aikaisemmin mainitun ongelman. Sisäänkirjautumispainike taas sidottiin iOS-alustan kosketustapahtumaan ontouchstart,

joka mahdollistaa laitteen nopeamman reagoinnin käyttäjän tekemään painallukseen. Painalluksista tapahtuva välitön reagointi on käyttökokemuksen kannalta kriittinen osa toiminnallisuuden vakuuttavuutta ja näyttävyyttä.

4.1.2 Päänäkymä

Sisäänkirjautumisnäkymän jälkeen käyttäjälle tuodaan näkyville sovelluksen varsinainen päänäkymä (kuva 1), joka sovelluksen runkorakenteen kannalta sisältää kaikki mahdolliset näkymät erillisinä osioina. Kaikki näkymätasot on sidottu yhteen yhdellä -keskittäjäelementillä, joka pitää huolen siitä, että kaikki näkymät pysyvät tiukasti näytön sallimalla alueella ja mahdollistaa myös helpomman sisällön käsittelyn sekä sijoittelun. Keskittäjäelementin käyttö on yleistä web-ohjelmoinnissa, koska tällä tavoin pystytään helposti pitämään huoli siitä, että kaikki näkyville tuotava materiaali on juuri siellä, missä sen halutaan olevan. Keskittäjäelementti pitää sisällänsä kaikki muut runkoon liittyvät elementit, joista jokainen on oma sisällön keskittäjänsä ja paikallaan pitäjänsä.



KUVA 1. Päänäkymän pohjapiirros

Päänäkymä (kuva 1) on jaettu neljään alueeseen, joita käyttäjä pystyy kontrolloimaan ja käsittelemään. Ensimmäisenä käyttäjän on mahdollista valita niin sanotusta hakuosioista haluamansa kategoria ja tämän valinnan pohjalta avataan hakunäkymä, joka antaa käyttäjälle mahdollisuuden selata haun perusteella valittuja tietoja. Demoluonteen johdosta toiminto avaa vain siis uuden näkymän, johon luotu sisältö valmiiksi. Jokainen elementti hakuosiossa on sidottu osaksi ontouchstart-tapahtumaa, ja painallusten pohjalta JavaScript-puoli hoitaa graafisten muutosten luomisen runkorakenteeseen hakunosion kannalta.

Hakuosiosta seuraavana on näkyvillä niin kutsuttu sovellustaso, joka näyttää käyttäjän tilaamat erilliset sovellukset jotka on yhdistetty hänen henkilökohtaiseen TV-Compass-käyttäjätunnukseensa. Sovellustasolla on kolme ensimmäistä elementtiä kiinnitetty ontouchstart -tapahtumaan, ja kun jos käyttäjä painaa jotain näistä kolmesta ensimmäisestä elementistä avataan käyttäjälle näkyville hänen valitsemansa uusi taso. Sovellustason elementit on sidottu iScroll-kirjaston vaatimalla tavalla, jotta sovelluselementtejä pystyttäisiin selaamaan natiivikäyttäytymistä vastaavalla tavalla. Sovelluselementit on sidottu keskenään yhteen ensinnä -keskittäjäelementillä ja tämän jälkeen toisella -elementillä, jolla mahdollistetaan elementtien laitteistokiihdytteinen 3D-animaatio käyttäen hyväksi CSS3:n tarjoamia animaation luonti mahdollisuuksia. Loppujen lopulta jokainen sovelluselementti on aivan yksinkertainen -listaelementti.

Sovellusosion alapuolelta käyttäjä löytää omakirjasto- ja ohjelmistojonoalueen, joka on luotu samalla tavalla rakenteellisesti kuin aikaisempi sovellusosio. Tässä osiossa ainoastaan yksi elementti on sidottu osaksi ontouchstart-tapahtumaa, ja tämän tapahtuman kautta aukaistaan sovelluksen käyttäjän oma elokuvakirjasto, josta käyttäjän voi selailla omistamiansa ja katsomiansa elokuvia. Omakirjasto ja ohjelmistojonojen muodostama yhteinen elementti on ainut, joka on jouduttu käsittelemään JavaScriptin ja CSS3:n avulla toimimaan käänteisesti, koska elementin animointi toimiikin muihin selattaviin elementteihin verrattuna väärinpäin.

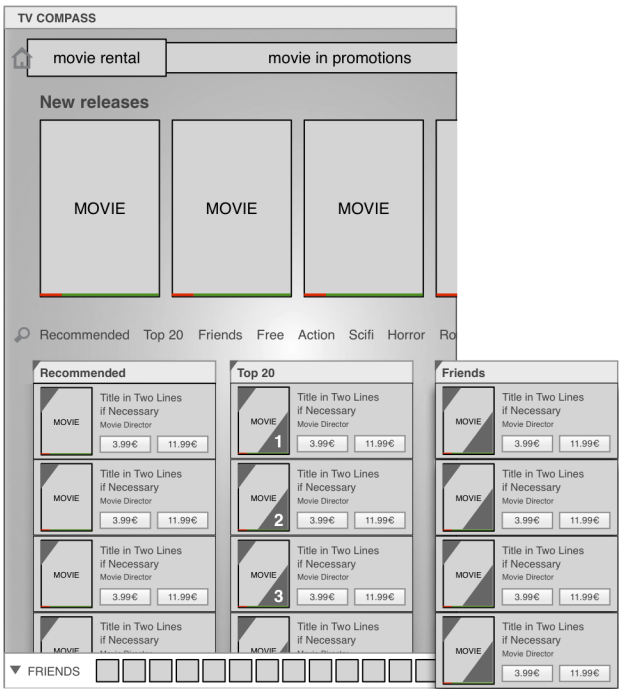
Päänäkymän rungon viimeisenä osana löytyy käyttäjän omiin valintoihin ja sovelluksiin pohjautuva eräänlainen mainosalue, josta käyttäjä voi nopeasti selata hänelle suositellut ohjelmat ja ohjelmat, joita hänen ystävänsä ovat katsoneet tai kommentoineet viimeksi. Rungon kannalta viimeinen osio koostuu viidestä eri pääelementistä, joista ensimmäinen on kaiken sisäänsä ottava - sekä -elementti yhdistelmä, joilla saadaan luotua muille

animaatioelementeille yksi yhteinen alusta, jolla mahdollistetaan muiden vertikaalisesti käsiteltävien elementtien käsittely horisontaalisessa tasossa. Runkorakenteen kannalta tämäkin osio on hyvin yksinkertainen, vaikka iScroll-kirjaston hyödyntämisen kannalta rungon rakennetta joutui hieman muuttamaan, mutta loppujen lopulta suurimman työn joutui tekemään CSS3:lla, jolla elementtien määrittelyt pitää tehdä toistensa sisään.

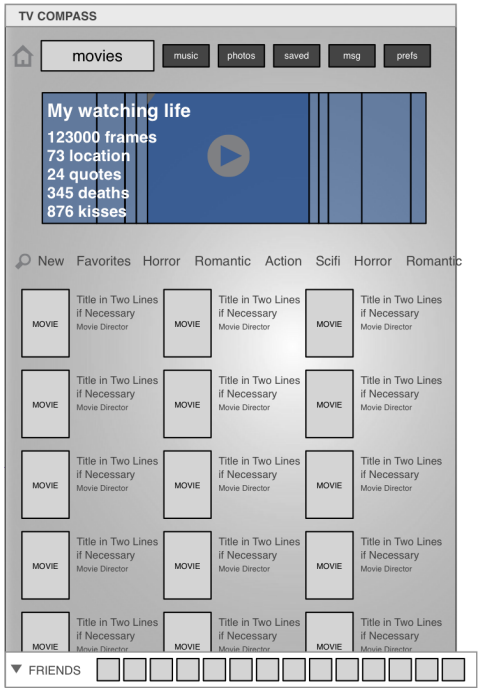
Viimeisenä päänäkymän yhteydessä näkyville tulee sosiaalisentaso, joka on koko ajan käyttäjän näkyvissä, vaikka käyttäjä vaihtaisikin päänäkymästä johonkin toiseen näkymään. Taso toimii käyttäjän kommunikointivälineenä muiden TV-Compass-sovelluksen käyttäjien kanssa ja sisältää mahdolliset sisäiänkirjautuneet ystävät sekä heidän kanssaan auki olevat keskustelut. Sosiaalinen taso koostuu rungoltaan yhdestä kaiken sisäänsä sulkevasta -elementistä, jonka sisälle on luotu kaksi erillistä - ja -elementtikokonaisuutta, joista toinen näyttää käyttäjälle hänen ystävänsä, jotka ovat kirjautuneet sillä hetkellä sisään. Toinen kokonaisuuksista sisältää käyttäjän auki olevat keskustelut. Keskusteluelementteihin on sidottu mukaan ontouchstart-tapahtuma, jonka myötä sovelluksen käyttäjälle kerrotaan milloin mikäkin keskustelu hänellä on aktiivisena ja mihin keskusteluun hän on kirjoittamassa viimeisintä viestiänsä. Sovelluksen demoluonteesta johtuen keskustelu on niin sanotusti väärennetty, koska markkinointitarkoitusten kannalta keskustelun täydellinen toimivuus ei ollut välttämätöntä.

4.1.3 Muut näkymät

Päänäkymän ulkopuoliset näkymät koostuvat suurilta osin samantyylisistä elementeistä kuin päänäkymäkin. Näkymät pitävät sisällänsä samankaltaisia selattavia listoja, joista myös päänäkymä suurimmaksi osaksi koostuu. Omat elokuvat -näkymä (kuva 3) ja elokuvien vuokraus -näkymä (kuva 2) ovat melkein identtiset sisältäen kolme erillistä vertikaalisesti selattavaa listaa, joista käyttäjä pystyy joka selaamaan vuokrattavina olevia elokuvia, tai jo katsomiansa elokuvia. Vuokrattavat elokuvatosiossa kolmen selattavan listan lisäksi on myös yksi suurempi horisontaalinen lista, josta käyttäjä voi selata vasta vuokralle tulleita elokuvia.



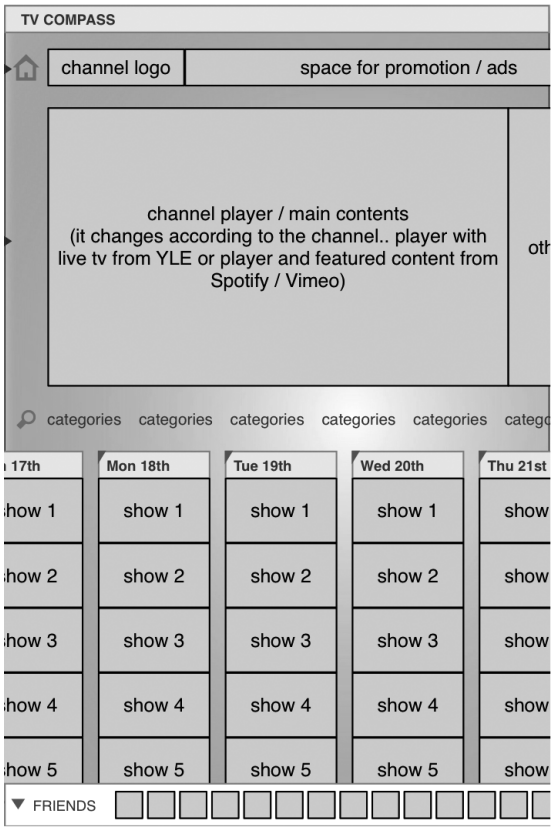
KUVA 2. Elokuvien vuokraus -näkymän pohjapiirros



KUVA 3. Omat elokuvat -näkymän pohjapiirros

Tv-kanava näkymä (kuva 4) eroaa siinä määrin muista näkymistä, että siihen on tehty poikkeuksellisesti karusellityyppinen selausrunko tärkeimmille elementeille ja näiden elementtien alle on sijoitettu erillinen videontoistonäkymä, joka tulee näkyville kun

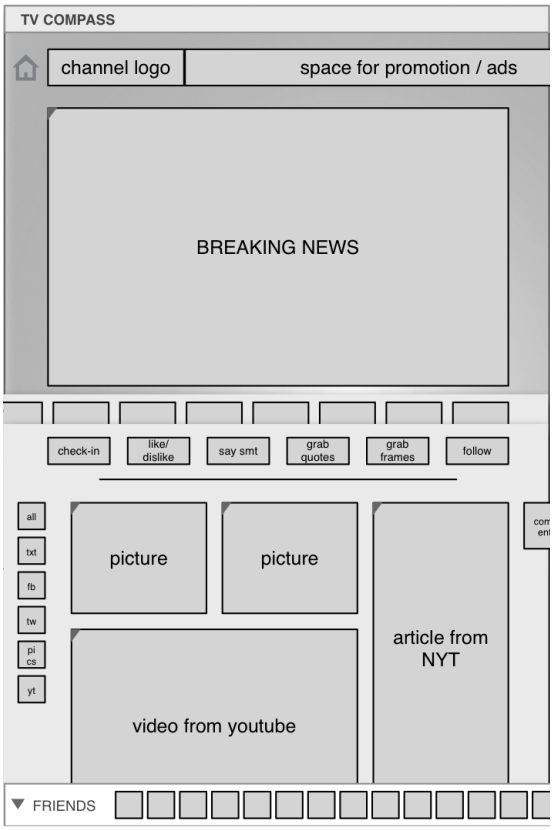
karusellin viimeisen elementin toistonappia painetaan. Videon toisto -näkymä luottaa täysin HTML5:n tarjoamiin uusiin mahdollisuuksiin lisätä helposti videoelementtejä mukaan sivustokokonaisuuteen. Videoelementin rakenteellinen osuus on hyvin yksinkertainen ja selkeä ja noudattaa normaalia HTML5:n dokumentaatioista otettua mallia. Videonäkymän runko koostuu itse videoelementistä sekä videoelementin sulkevasta suljinelementistä, joka on erikseen luotu videoelementin rinnalle, jotta käyttäjän olisi helppo palata alkuperäiseen videokarusellinäkymään.



KUVA 4. Tv-kanava näkymän pohjapiirros

Viimeisenä kokonaisuudesta poikkeavana näkymänä on uutiskanavanäkymä (kuva 5), joka toimii suurimmaksi osaksi samalla runko rakenteella kuin muutkin näkymät, mutta tässä näkymässä on huomattavasti suurempia animaatioitavia elementtejä. Uutisnäkymän pohja koostuu kahdesta erillisestä osasta, eli näkymän alaosa on erillinen verrattuna muuhun näkymään, koska osaa tulee pystyä liikuttamaan käyttäjän toimintojen mukaan mahdollisimman helposti ja nopeasti. Tästä syystä näkymän keskeltä löytyvä piilotettu aikajanaosio tulee pystyä näyttämään käyttäjälle helposti ja nopeasti ja tämä vaati sen, että näkymän alaosa on oma erillinen tasonsa. Uutisnäkymästä löytyy myös mahdollisuus niin sanottuun koko näytön toistoon

uutiskanavan tarjoamiin uutisartikkeleihin. Tämän ansiosta, koska artikkelit haluttiin lisätä mukaan kokonaisuuteen, on mahdollista avata yksi artikkeleista, ja artikkeli tulee animaatioiden mukana näkyviin ja osaksi kokonaisuutta. Jos uutisnäkymän artikkeli on näkyvissä, se toimii osana kokonaisuutta ja animoituu muun kokonaisuuden mukana käyttäjän reaktioiden mukaan. Kuten aikaisemminkin, uutisnäkymä koostuu yhdestä kaiken ympäröivästä -elementistä, jonka sisälle kaikki muu sisältö on rakennettu erinäisten painikkeiden ja näkymien kokonaisuudeksi.



KUVA 5. Uutiskanavanäkymän pohjapiirros

Lyhykäisyydessään jokainen demoon toteutettu näkymä koostuu sisällön ympäröivästä -elementistä, ja kaikki sen sisälle sijoitettu sisältö on suurimmaksi vain uusia -elementtejä jotka on sidottu joko tunnuksen tai luokan mukaan CSS3-määrityksiin. Ilman elementtien sitomista tunnuksen tai luokan kautta olisi lähes mahdotonta määritellä CSS3:n avulla elementeille mitään erillisiä tietoja tai käsitellä elementtejä JavaScriptillä. (Esimerkki 4.)

```
<html>
<head>
```

```
</head>
<body>
  <div id=”keskitin”>
  <!--keskitin elementti-->
    <div id=”näkymä1”>
    <!--näkymä elementti-->
      <div id=”wrapper”>
        <div id=”scroller”>
          <ul> <!--iScroll kirjaston vaatimuksien mukaan rakennettu elementti kokonaisuus-->
            <li><a href=”#” class=”elementti1” ontouchstart=”funktio1();”></a></li>
            <li><a href=”#” class=”elementti2” ontouchstart=”funktio2();”></a></li>
          </ul>
        </div>
      </div>
    </div>
  </div>
</body>
</html>
```

ESIMERKKI 4. Perusrakenne HTML5-syntaksin mukaiselle sivustorungolle

4.2 CSS3 ja demon ulkoasun rakentaminen

TV-Compass-demo luottaa markkinointidemona vahvasti visuaaliseen ulkoasuunsa, ja sen tarjoamaan näyttävyyteen, joten oli kriittisen tärkeä, että jokainen elementti, joka demossa on nähtävillä, olisi juuri siinä paikassa missä sen kuuluukin olla. Siksi jouduin luottamaan elementtien absoluuttiseen määrittelyyn, eli jokaiselle elementille on määritelty oma tarkka sijaintinsa sovelluksessa. Tarkalla sijainnilla tarkoitetaan sijaintia xyz-koordinaatistossa, koska jokaiselle elementille on CSS3:n avulla määritelty sen sijainti näkymissä sekä tasoissa joissa näkymät esitetään käyttäjälle. Samalla on myös määritelty jokaiselle elementille sen tarkka pikselikohtainen koko. Pikselin tarkka määrittely mahdollistaa elementtien mahdollisimman tarkan ja täsmällisen hallitsemisen CSS3 avulla.

4.2.1 Aloitusnäkymä

Aloitusnäkymä on sijoitettu ulkoasu määrittelyissä heti kärkeen, jotta se saataisiin käyttäjälle mahdollisimman nopeasti näkyville, siinä vaiheessa kun PhoneGap-sovelluskehys on ilmoittanut olevansa valmis. Tuodaan sisäänkirjautumisnäkymä

käyttäjän eteen vasta siinä vaiheessa, kun itse laite on ladannut kaiken mahdollisen sovelluskehykseen liittyvän valmiiksi. Aloitusnäkymä luottaa yksinkertaiseen grafiikkaan, jotta demo käynnistyisi mahdollisimman nopeasti ja sulavasti, jos aloitusnäkymä sisältäisi paljon graafisia elementtejä. Loppukäyttäjälle ilmenisi sovelluksen käynnistymisen yhteydessä elementtien vilkkumista.

Näkymä on sidottu yhdellä -elementillä, joka sitoo aloitusnäkymän sisäänsä ja sijoittelee sen oikein osaksi kokonaisuutta. Elementtiä piti siirtää kaksikymmentä pikseliä ylöspäin, koska iOS-alustan ylälaidan otsakepalkkia ei pystytty poistamaan näkyvistä sovelluksen ajon ajaksi. Elementille on myös määritelty valmiiksi WebKit-pohjainen laitteistokiihdytteinen 3D-animaatio, joka on sidottu elementin näkyvyyteen sekä z-akselin indeksiin.

Esimerkistä 5 näkee selkeästi ja nopeasti näkymäelementtien perusrakenteen. Jokainen näkymäelementti koostuu sille määritellystä näkymän vaihtoanimaatiosta, absoluuttisesta sijainnista xyz-koordinaatistossa, absoluuttisesta koon määrittelystä sekä tarvittavista taustakuva tiedoista. -elementille on määritelty erikseen myös taustaväri, koska jos jostain syystä sovelluskehys ei ehdi lataamaan ensimmäistä näkymää tarpeeksi nopeasti tällä tavoin pystytään edes osittain välttämään epämiellyttävää vilkkumista sovelluksen alkuvaiheessa.

```
#startUp{
    opacity: 1;
    -webkit-transition-property: opacity z-index;
    -webkit-trasition-duration: 1000ms;
    top: 20px;
    width: 768px;
    height: 1024px;
    margin: 0 auto;
    text-align: left;
    overflow-y: hidden;
    overflow-x: hidden;
    background-image: url('default.jpg');
    background-color: #090909;
    position: absolute;
    z-index: 1000;
}
```

ESIMERKKI 5. StartUp-elementin CSS3 määrittelyt

Taustaelementin lisäksi aloitusnäkymässä on myös sisäänkirjautumisosa, joka animoidaan näkyville vastaa siinä vaiheessa kun taustaelementti on ladattu valmiiksi. Rakenteeltaan kyseinen elementti on peruslogiikaltaan täysin vastaavanlainen kuin aikaisemmin esitelty -elementti. Ainoat erot löytyvät luonnollisesti elementin sijoittelusta ja koosta. Sisäänkirjautumiselementti sisältää myös tekstikentät ja painonapin, jolla sisäänkirjautuminen suoritetaan. Jokainen näistä elementeistä muotoiltiin uusiksi käyttäen hyväksi CSS3:n tarjoamia mahdollisuuksia muokata <input> -elementtejä huomattavasti monipuolisemmin mitä aikaisemmin. CSS3:n tarjoamien muotoilu mahdollisuuksien myötä pystytään kyseiset elementit nykypäivänä muotoilemaan täysin ilman suurempia rajoitteita. Aikaisemmin <input>-elementtien muotoilussa joutui käyttämään päällekkäisiä tasoja ja elementtien läpinäkyvyyttä hyödyksi ja tämä aiheutti tietynlaista täysin turhaa työtä.

Esimerkki 6 myös näyttää selkeästi sen kuinka helposti tunnuksella nimettyihin elementteihin pystytään sitomaan CSS3:lla määriteltyjä ulkoasu muutoksia. Yhdellä määrittelyllä pystytään helposti muokkaamaan ja määrittelemään useille samankaltaisille elementeille tarvittavat pohjainformaatiot.

```
input#userId, input#userPassword{
```

```
    -webkit-border-radius: 5px;
    font-family: ”Lucida Grande”, sans-serif;
    font-size: 12px;
}
```

ESIMERKKI 6. Input-elementin muotoilu

4.2.2 Päänäkymä

Päänäkymä sisältää muihin näkymiin verrattuna eniten yksittäisiä elementtejä. Tämä johtuu siitä, että päänäkymästä käyttäjän tulisi saada nopeasti kaikki haluamansa informaatio siitä mitä on tapahtunut hänen ollessaan kirjautuneena ulos TV-Compass-sovelluksesta. Kuitenkin suurin osa päänäkymän elementeistä pohjautuu jo aikaisemmin mainittuun iScroll-kirjaston vaatimaan elementtirakenteeseen. Päänäkymänsisällön sitova elementti käyttää hyödykseen luokka muuttujaa elementin tunnistamiseen, koska alun perin sovelluksessa piti olla mukana myös vaakatason näkymä, ja oli kaikista yksinkertaisin suorittaa elementtien ominaisuuksien vaihto vaihtamalla elementille vain uusi luokka. CSS3:n perussyntaksin mukaan luokkia voi käyttää useammissa elementeissä useita kertoja, mutta tunnusta voi käyttää vain yhdessä tietyssä elementissä. Tämän vuoksi oli helpompi käyttää luokkaa elementin tunnistamiseen, koska tunnuksia käyttämällä olisin joutunut luomaan kaksi erillistä päänäkymää. Luokkia hyödyntämällä pystyin käyttämään pelkästään yhtä pohjanäkymää, jonka sisältöä käsiteltiin JavaScriptin avulla.

Päänäkymän tärkeimmät elementit käyttäjän kannalta rakentuvat iScroll-kirjaston vaatimista elementtirakenteista. iScroll-kirjasto mahdollistaa kyllä monipuolisten erilaisten selailtavien elementtikokonaisuuksien luonnin, mutta päänäkymän kannalta niin kutsuttu normaali rakenne on tärkein ja eniten demossa hyödynnetty.

Animoidun listan esimerkki 7:n koodi kertoo selkeästi ja hyvin yksiselitteisesti kuinka CSS3:n avulla elementin tärkeimmät ominaisuudet saadaan määriteltyä helposti ja nopeasti. Lista on otettu suoraan sovelluksen runkorakenteen yläpäästä ja se on niin kutsutun sovellukset listan ulkoasumäärittely. Listaelementti sijoitetaan -elementin määritelmän avulla näkymän yläosioon. Sen jälkeen -elementin sisään tulevalla -elementillä määritellään listan fyysinen koko päänäkymän yläosassa. Kahden edellä mainitun elementin sisään tuleva lista määritellään toimivaksi ilman listamuotoa ja jokaiselle -listaelementille

määritellään tarkasti sen pikselintarkka koko joka on 140 pikseliä kertaa 140 pikseliä. Listaelementin avulla myös määritellään listan niin sanottu puskurialue, johon lista tulee pysähtymään aina siinä vaiheessa, kun lista rullataan alkuun, ja kaikilla listoilla jotka demossa toimivat horisontaalisessa tasossa puskuri on 20 pikseliä näkymän vasemmasta reunasta. Jokaisen listaelementin sisään on sijoitettu linkkielementti, johon on sidottu luokka tunnuksen avulla linkkielementin taustakuva määritelmä.

```
#wrapper1{
    position: absolute;
    z-index: 1;
    top: 108px; bottom: 0px; left: 0px; right: 0px;
    width: 100%;
    overflow: auto;
}
#warpper1 #scroller{
    width: 1620px;
    height: 140px;
    float: left;
    padding: 0;
}
#wrapper1 #scroller ul{
    list-style: none;
    display: block;
    float: left;
    width: 100%;
    padding: 0;
    margin: 0;
    text-align:left;
}
#wrapper #scroller li{
    display: block;
    vertical-align: middle;
    float:left;
    margin-top: 0px; margin-bottom: 0px, margin-left: 20px; margin-right: 0px;
    width: 140px;
    height: 140px;
    font-size: 14px;
}
```

ESIMERKKI 7. iScroll-listaelementin CSS3 määritys

Esimerkissä 8 määritellään listaelementin sisäisen linkkielementin luokka. Luokkamääritelmiä pystytään tarpeen vaatiessa käyttämään HTML5 rungossa useilla samankaltaisilla elementeillä.

```
a.apps1{
    -webkit-box-shadow: 3px 6px 10px #010101;
    width: 140px;
    height: 140px;
    text-decoration: none;
    display: block;
    background-image: url(apps_1.jpg);
    background-repeat: no-repeat
}
```

ESIMERKKI 8. Linkkielementin CSS3 luokkamäärittely

Tässä tapauksessa luokka on kuitenkin sidottu käytettäväksi ainoastaan linkkielementtien kanssa. Luokkamääritelmässä määritellään kyseisellä linkkielementille varjostus käyttäen hyväksi CSS3:n uusia ominaisuuksia. Tarpeen vaatiessa kaikki varjostukset pystyttäisiin luomaan dynaamisesti sekoittaen CSS3:n ja JavaScriptin tarjoamia mahdollisuuksia, mutta tässä tilanteessa kyseinen toimenpide ei ollut tarpeellinen. Koska luokka määritellään käytettäväksi pelkästään linkkielementeissä, on määrittelyssä myös merkintä jolla poistetaan käytöstä kaikki linkkielementin normaalit toiminnot. Linkkielementti ei tämän johdosta reagoi painalluksiin ilman, että painallusta käsiteltäisiin JavaScriptin avulla. Lopuksi elementille määritellään sen käyttämä taustakuva, ja tällä tavoin HTML5 rungosta löytyvä elementti saa itsellensä kaikki tarvittavat määrittelyt, jotta elementti varmasti on sijoitettu oikeaan paikkaan ja näyttää siltä miltä sen kuuluukin.

Päänäkymän alalaidan suurin elementti lähes puolet koko näkymästä vievä viimeisimmät tapahtumat osio on rakenteeltaan vastaavanlainen kuin mikä tahansa muistakin demossa käytetyistä iScroll-kirjaston avulla käsitellyistä elementeistä, mutta kyseisessä osiossa on sovellettu iScroll-kirjaston ja CSS3:n monipuolisia elementtien merkintätapoja. Alaosa koostuu neljästä erillisestä selattavasta listasta, joista jokaista voi selata erikseen vertikaalisesti, mutta nämä neljä selattavaa listaa on sidottu yhteen horisontaalisesti selattavaksi listaksi. Tämä aiheutti koodin luontivaiheessa

mielenkiintoisia ongelmia, mutta pienen selvittelyn ja CSS3:n hieman pidemmälle viemisen jälkeen kokonaisuus ryhtyi toimimaan halutulla tavalla.

Alkuperäisessä CSS3:n määrittelyssä loin jokaisen listaelementti kokonaisuuden samalla tavoin kuin olin jo aikaisemmin luonut horisontaalisesti selattavia listoja. Luonnollisesti muutin listan rakennetta hieman, jotta lista olisi vertikaalisesti selattava. Vertikaaliset listat sisäänsä sulkeva horisontaalisesti selattava lista luotti samaan rakenteeseen kuin aikaisemminkin luodut horisontaaliset listat, mutta HTML:n puolella horisontaalisen listan listaelementtien sisällöt oli nyt korvattu pelkkien kuvien sijasta kokonaisilla omilla listoillansa. Ongelmaksi tässä vaiheessa muodostui se, että sisemmät listat kadottivat CSS3:ssa määrittelynsä sovelluksen käynnistymisen yhteydessä. Tästä johtuen listat eivät osoittaneet minkäänlaista toiminnallisuutta. Lopulta useamman yrityksen ja erehdyksen jälkeen selvisi, että sisemmät vertikaaliset listat tulisi CSS3:ssa määritellä vain tarkemmin. Tämän tiedon pohjalta muutin vertikaalisten listojen CSS määritykset esimerkki 7:n esittämään muotoon, joka ratkaisi ongelmani.

```
#contentWrapper #contentScroller #wrapper4{
    position: absolute;
    z-index: 3;
    top: 64px;
    left: 22px;
    width: 350px;
    height: 460px;
    overflow: auto;
}
#contentWrapper #contentScroller #wrapper4 #scroller{
    position: absolute;
    z-index: 3;
    width: 100%;
    padding: 0;
}
#contentWrapper #contentScroller #wrapper4 #scroller ul{
    list-style: none;
    padding: 0;
    margin: 0;
    width: 340px;
    text-align: left;
}
```

```
#contentWrapper #contentScroller #wrapper4 #scroller li{
    margin-top: 0px:
    margin-left: 0px;
    margin-bottom: 20px;
    height: 140px;
    width: 340px;
    font-size: 14px;
}
```

ESIMERKKI 9. Kuinka määritellä elementin sisäinen elementti

Esimerkistä 9 näkee selkeästi kuinka CSS3:n ohjelmointisyntaksi mahdollistaa elementtien määrittelyn muiden elementtien sisällä. Esimerkissä määritellään päänäkymän alaosion ensimmäisen vertikaalisesti selattava lista horisontaalisesti selattavan suuremman listan sisällä. Käytännössä siis suuremman listaelementin sisältö on korvattu kokonaisuudessaan täysin uusilla elementeillä poiketen iScroll-kirjaston alun perin esittelemästä listojen perusrakenteesta.

4.2.3 Muut näkymät

Muut demon näkymistä koostuvat pääosin samankaltaisista elementeistä. jotka jo aikaisemmissa luvuissa on esitelty. Kuitenkin jokaisen näkymän sitova elementti poikkeaa hieman päänäkymän sitovasta elementistä. Muiden näkymien sitovat elementit on asetettu elementin läpinäkyvyys- ja tasoarvon puolesta nollaan, kuten esimerkissä 8 voidaan havaita. Läpinäkyvyys- ja tasoarvoon on kiinnitetty CSS3:lla laitteistokiihdytteinen 3D-animaatio määritys, joka suoritetaan joka kerta kun kumpaa tahansa edellä mainituista arvoista muutetaan JavaScriptin avulla. Tällä varmistetaan näkymästä toiseen siirtymisen näyttävyys ja sulavuus.

Esimerkki 10 näyttää selkeästi sen kuinka helposti elementeille pystytään määrittelemään kaikki tarvittavat alkuarvot, joiden pohjalta JavaScriptillä pystytään helposti hallinnoimaan elementtien animaatioita. Opacity eli läpinäkyvyysarvo on yksi niistä arvoista jonka laitteistokiihdytteistä animaatiota WebKit-selainmoottori tukee. Laitteistokiihdytyksen avulla demon animaatioista saadaan todella sulavia ja miellyttävän näköisiä ilman minkäänlaisia nykimisiä. WebKit-selainmoottori mahdollistaa myös elementtien liikuttelun xyz-koordinaatistossa laitteistokiihdytteisenä 3D-animaationa. Tätä ominaisuutta hyödynnetään kaikissa liikuteltavissa elementeissä näkymien sisällä. Ilman tätä mahdollisuutta, ei demoa olisi pystytty saamaan tarpeeksi

sulavaksi. Ilman laitteistokiihdytteistä animaatiota osien animaatiot olivat nykiviä ja miellyttävän käyttökokemuksen kannalta se ei ole hyväksyttävää.

```
#divNews{
    opacity: 0; /*läpinäkyvyysarvo*/
    -webkit-transition-property: opacity z-index; /*animoitavat arvot*/
    -webkit-transition-duration: 750ms; /*animaation kesto millisekuntteina*/
    background-image: url('newsBackground.jpg');
    height: 100%;
    width: 100%;
    overflow-y: hidden;
    overflow-x: hidden;
    position: absolute;
    z-index. 0; /*tasoarvo*/
}
```

ESIMERKKI 10. Uutisnäkymän sitovan elementin määrittely

Chat-näkymä on koko ajan käyttäjän käytettävissä ja muiden elementtien yläpuolella taso indeksissä. Chat-näkymän animaatiot toteutetaan laitteistokiihdytteisenä 3D-animaationa ja tämä mahdollistuu -webkit-transform: translate3d(x, y, z); ominaisuuden xyz-koordinaatiston arvoja muuttamalla. Elementin alkuarvot määrittelevät elementin sijainnin käynnistymistilanteessa. Arvoja on yksinkertaista muuttaa JavaScriptin avulla elementin yksilöllisen id-tunnuksen ansiosta. Demossa chat-näkymää animoidaan ainoastaan y-akselistaan, koska näkymä tulee saada vieritettyä käyttäjän näkyville sovelluksen alalaidasta silloin kun käyttäjä niin haluaa. Samankaltaista animaatio logiikkaa käytetään myös monessa muussakin demon elementissä. Luonnollisesti kyseisiä animaatioita olisi helppo luoda käyttäen hyväksi esimerkiksi jQueryn tarjoamia animaatio funktioita, mutta jQuerya hyödyntämällä animaatiot eivät tapahtuisi laitteistokiihdytteisesti. Siksi jQueryn hyödyntäminen demossa on jätetty melko vähälle, ja animaatioiden määrittelyt hoidetaan puhtaalla JavaScriptillä.

```
#div_social{
    -webkit-transform: translate3d(0, 919px, 0);
    -webkit-transition: -webkit-transform;
    -webkit-transition-duration: 500ms;
    -webkit-transition-timing-function: ease-out;
    width: 768px;
    height: 770px;
```

```
    background-image: url('friendsBg.jpg');
    position: absolute;
    display: block;
    z-index: 20;
}
```

ESIMERKKI 11. Chat-näkymän sitovan elementin määrittely

Peruslogiikaltaan jokainen näkymä noudattaa samaa peruskaavaa. Tällä tavoin on ollut helppo mahdollistaa demon peruslogiikan yksinkertaisuus ja toimivuus. CSS3:n ja WebKitin tarjoamat laitteistokiihdytteiset 3D-animaatiot olivat demon näyttävyyden ja toimivuuden kannalta täydellinen siunaus sille, että demosta oli mahdollista saada niin näyttävä kuin siitä vain haluttiin. Animaatiot olisi luonnollisesti ollut mahdollista toteuttaa käyttäen hyväksi pelkkää puhdasta JavaScriptiä, mutta CSS3:n tarjoamat mahdollisuudet animaatioiden kannalta nopeuttivat ja helpottivat huomattavasti demon kehittämistä ja työstämistä valmiiseen muotoonsa.

4.3 TV-Compass-demon sisällönhallinta JavaScriptillä

Ilman JavaScriptillä luotua sisällönhallintaa ei TV-Compass-demossa tapahtuisi mitään muuta kuin alkunäkymän taustakuvan näkyviin tuleminen. PhoneGap-sovelluskehys luottaa vahvasti sisällönhallinnassa JavaScriptiin, koska kaikki sisältö ja näkyvät elementit tulee pitää yhdessä index.html-tiedostossa. Sovelluksen käynnistymiseen on liitetty PhoneGap-sovelluskehyksen oma kuuntelija, joka ilmoittaa sovellukselle missä vaiheessa laite on valmis ja tämän jälkeen tullaan suorittamaan haluttuja toimintoja. TV-Compass-demon alkuvaiheessa HTML5 rungon <body>-elementin onload-tapahtumaan on sidottu PhoneGap-sovelluskehyksen ohjeiden mukaisesti onLoad()-funktio, jossa alustetaan osa iScroll-kirjaston olioista ja samalla myös odotetaan laitteelta ilmoitusta siitä, että sovelluksen alkulataus on valmis. Siinä vaiheessa kun alkulatauksen kuuntelija saa itsellensä true-arvon kutsutaan välittömästi onDeviceReady()-funktiota, joka tuo sovelluksen käyttäjälle näkyviin ensimmäisen näkymän. Sisäänkirjautumisnäkymän tekstikentät sekä painike-elementin, jonka kautta käyttäjä pystyy niin sanotusti kirjautumaan sovellukseen sisälle.

```
function onLoad(){
    myScroll2.scrollToElement('li:nth-child(1)' 0);
    myScrollnewsTimeline1.scrollToElement('li:nth-child(1)' 0);
    myScrollnewsTimeline2.scrollToElement('li:nth-child(1)' 0);
```

```
    document.addEventListener(”deviready”, onDeviceReady, false);
}
function onDeviceReady(){
    document.getElementById('login').setAttribute(”style”,”opacity: 1; z-index: 1000;”);
    delete onDeviceReady();
}
```

ESIMERKKI 12. Funktiot onLoad() ja onDeviceReady()

Esimerkistä 12 näkee selkeästi kuinka helposti ymmärrettävä JavaScriptin perussyntaksi on. OnDeviceReady()-funktion ensimmäisellä rivillä on nähtävissä peruslogiikka jolla elementtien animaatioita hoidetaan sovelluksessa näkymien vaihdossa ja elementtien siirtelyssä paikasta toiseen. Käytännössä siis JavaScriptillä määritellään elementin id-tunnuksen perusteella kyseiselle elementille uudet arvot halutuille arvoille, ja tässä tapauksessa halutaan tuoda -elementti näkyviin, jolloin elementin läpinäkyvyys muutetaan aloitusarvosta uuteen haluttuun arvoon. Samalla myös tuodaan elementti z-akselilla oikealla korkeudelle. Uusien arvojen määrittelyn myötä hoitavat CSS3 ja WebKit muutoksen alkutilanteesta uuteen haluttuun tilanteeseen.

Päänäkymään tullessa näkyviin on käyttäjällä edessään selattavat listat, joista jokainen on iScroll-kirjaston avulla luotu uusi listaselain olio. iScroll-kirjaston avulla on yksinkertaista ja helppoa luoda uusia listaselain olioita, joissa on natiivisovelluksen omainen käyttäytyminen ja käyttötuntuma. iScroll on alun alkaen suunniteltu käytettäväksi juuri iOS-alustalle kehitettävien web-sovellusten yhteydessä, mutta koska PhoneGap-sovelluskehys pohjautuu web-ohjelmointikieliin, oli iScroll-kirjastoa helppo hyödyntää täydellä teholla demon luontivaiheessa. Kaikki listaselain oliot on luotu jo sovelluksen käynnistysvaiheessa, ainoastaan listojen sisällöt ladataan näkyviin myöhemmässä vaiheessa.

iScroll -luokan olio sidotaan elementtiin sen id-tunnuksen avulla. Samalla myös määritellään oliolle informaatioita siitä, kuinka sen tulisi muotoilla ja näyttää sille määrätyn elementin osia. Esimerkissä 13 asetetaan esimerkiksi listojen vierityspalkit false-tilaan jolloin vierityspalkit piilotetaan pois näkyvistä.

```
myScroll1 = new iScroll('wrapper1', { hScrollbar: false, vScrollbar: false });
myScroll2 = new iScroll('wrapper2', { snap: true, hScrollbar: false, vScrollbar: false });
```

```
myScrollShowCarousel = new iScroll( 'wrapperShowCarousel', { snap: 'li', momentum: false;
hScollbar: false });
```

ESIMERKKI 13. iScroll-olioiden alustuksia

Demossa jokaiselta elementiltä vierityspalkit on jätetty pois, koska niitä ei ollut sisällytetty konsepteihin tai grafiikoihin. Esimerkissä 13 ensimmäinen luoduista iScroll-olioista myScroll1 on päänäkymän ensimmäinen listaelementti, joka ei tarvitse muita määrittelyjä kuin noiden vierityspalkkien piilottamisen näkyvistä. Seuraava olio on taas liitetty päänäkymän keskimmäiseen listaelementtiin. Listaelementti eroaa aikaisemmasta siinä mielessä, että elementin nollakohdaksi on määritetty CSS:llä näytön oikea laita vasemman laidan sijasta ja tämän johdosta listaolion snap-arvo on asetettu todeksi eli true-arvoon. Tämän johdosta listaa selattaessa lista lukittuu listaelementteihin ja listaolio pitää lukua siitä, mihin elementtiin on lukittauduttu. Listaa on mahdollista vierittää olion päivitysfunktion avulla mihin tahansa haluttuun listaelementtiin joka on listan sisällä.

Kolmas olioista poikkeaa muista olioista siinä mielessä, että kyseessä on niin kutsuttu kuvakaruselli. Tämä listaolio pyörittää valittuja listaelementtejä karusellissa siten, että kerrallaan listan sisäisistä elementeistä vain yksi on näkyvissä kerrallaan. Karuselli pysähtyy aina seuraavan listan sisäisen elementin kohdalle. Tällä tavoin karusellin käyttö pysyy selkeänä ja yksinkertaisena. Oliolta on poistettu myös käytöstä liikemäärä efekti, joka aiheuttaisi elementtien liikuttelun päätteeksi pienen kuminauhaefektin. iScroll-kirjaston helppokäyttöisyys ja olioiden helppo muokattavuus mahdollisti demon saamisen siihen pisteeseen, että sovelluksen testaajien oli vaikea kertoa ilman taustatietoja onko kyseessä natiivisovellus vai sovelluskehyksen avulla luotu hybridisovellus.

4.4 Näkymien vaihdot ja niiden hallinnointi

Demon näkymien vaihtelu perustuu hyvin yksinkertaiseen ja toimivaan logiikkaan. Jokaista näkymää varten on oma näkyvyys muuttujansa jolla on käytössään vain kaksi erillistä arvoa. Näkymä on joko näkyvissä, tai sitten näkymä ei ole näkyvissä. Näiden muuttujien pohjalta tehdään näkymien vaihtofunktioiden sisällä tarvittava vertailuja, joiden pohjalta käyttäjälle näytetään käyttäjän haluama näkymä. HTML5 rungossa on sisällytetty haluttuihin elementteihin ontouchstart()-tapahtuma, jonka avulla kutsutaan kyseiseen elementtiin sidottua funktiota. Funktio kutsun tapahtuessa otetaan huomioon

kaikki tarvittavat muuttujat, joiden pohjalta tullaan joko näyttämään uusi näkymä, tai suorittamaan jokin muu toiminto, jonka käyttäjä on halunnut suorittaa.

```
var socialVisible = false;
function showHideSocial(){
    if (videoPlaybackOpen == true){
        if (socialVisible == false){
            var el2 = document.getElementById('div_social');
            var topmain = 530;
            el2.setAttribute(”style”,”-webkit-trasform: translate3d(0,” + topmain + ”px, 0);”);
            socialVisible = true;
        }
        else{
            var el2 = document.getElementById('div_social');
            var topmain = 919;
            el2.setAttribute(”style”,”-webkit-trasform: translate3d(0,” + topmain + ”px, 0);”);
            socialVisible = false;
        }
    }
    else{
        if (socialVisible == false){
            var el2 = document.getElementById('div_social');
            var topmain = 234;
            el2.setAttribute(”style”,”-webkit-trasform: translate3d(0,” + topmain + ”px, 0);”);
            socialVisible = true;
        }
        else{
            var el2 = document.getElementById('div_social');
            var topmain = 919;
            el2.setAttribute(”style”,”-webkit-trasform: translate3d(0,” + topmain + ”px, 0);”);
            socialVisible = false;
        }
    }
}
```

ESIMERKKI 14. Keskusteluosion animointi ja näkyvyyden vertailu

showHideSocial-funktio näyttää hyvän peruskaavan siitä kuinka yksinkertaisesti sovelluksen näkymiä pystytään hallinnoimaan. Keskusteluosion eli chat-näkymän animaatioon ei liity läpinäkyvyyden animaatiota ollenkaan, koska kyseinen näkymä on

koko ajan kaikkien muiden näkymien yläpuolella z-akselilla. Funktion logiikka on yksinkertainen, koska ainoa asia mitä funktion tulee tarkistaa ennen mahdollista suorittamista, on itse keskusteluosion ja videoelementti näkyvyys. Tämä siitä syystä, että HTML5 videoelementti aiheutti ongelmia iOS-alustalla. Videoelementissä on vielä tässä vaiheessa iOS-alustalla jonkinlainen virhe, jonka takia videoelementti sijoittuu z-akselilla aina kaikkien yläpuolelle. Tästä syystä keskusteluosion animaatioon suorittamiseen piti sijoittaa tarkistus sille onko videoelementti näkyvissä ja suorittaa animaatio tämän pohjalta keskusteluosio hieman matalammalle, jolloin keskusteluosio ei pääse peittämään videoelementtiä. Ilman tätä toimenpidettä keskusteluosioita ei olisi pystytty sulkemaan, video elementin ollessa päällä koska keskusteluosion vaihtopainike on sijoitettu osion ylälaitaan ja keskusteluosio olisi sijoittunut videoelementin päälle. Tässä tilanteessa vaihtopainikkeen tapahtumat olisi menetetty, koska alue johon tapahtumat on sidottu, olisi jäänyt videoelementin alapuolelle.

Demon muutkin sisällönhallintaan liittyvät funktiot noudattavat pitkälle lähes täysin samaa logiikkaa mikä esimerkissä on nähtävissä. Luonnollisesti jokainen funktio tutkii siihen liitettyjen omien muuttujiensa tilan ja niiden pohjalta suorittaa tarpeelliset toiminnot. Tarkoituksena on ollut pitää sovelluksen sisällönhallinta mahdollisimman yksinkertaisena sen takia, jotta koodia on pystynyt helposti optimoimaan ja miettimään mahdollisia rajoituksia joita iOS-alusta ja PhoneGap-sovelluskehys toivat demon luontivaiheessa. Alkuperäinen iPad 1 -laite, joka toimi demon luontivaiheessa testialustana, kärsi laskentatehon puutteesta, ja tämän ansiosta varsinkin sisällönhallintaa joutui miettimään moneen kertaa uusiksi ja kehittelemään mitä erinäisimpiä kokeiluja sen suhteen, kuinka demon näkymienvaihdot saisi toimimaan tarpeeksi sulavasti. Sisällönhallinnan mietintä ja kehittely kannatti, koska viimeistely vaiheessa kun sovellusta testattiin iPad 2 -laitteessa, minkäänlaisia nykimisiä ei enää ilmentynyt ja sovellus toimi uskottavan natiivimaisesti.

5 TESTAUS

TV-Compass-demoa testattiin tiheällä tahdilla, koska luontivaiheessa ei ollut vielä täyttä varmuutta siitä, kuinka PhoneGap-sovelluskehys tulisi lopulta toimimaan. Alkuvaiheessa ensimmäisiä versioita ei vielä pystytty testaamaan itse laitteessa, mutta heti testilaitteiston saapumisen jälkeen testauksia suoritettiin päivittäin ja tiheällä tahdilla, jotta jokainen sovellukseen luotu elementti olisi sellainen kuin sen pitäisikin olla, niin ulkoasultaan kuin toiminnallisuuksiltaan.

Testaamisten myötä ilmeni myös sovelluskehyksen tiettyjä rajoituksia demon kehittämisen alkuvaiheessa, ja nämä liittyivät aikaisemmin tekstissä mainittuun ontouchstart-tapahtumaan, jota demon kehityksen alkuvaiheessa ei hyödynnetty ollenkaan. Sovelluskehyksessä ilmeni huomattavia viiveitä käytettäessä web-ohjelmoinnin hiiritapahtumia elementtien painalluksien tunnistamisessa ja tämän johdosta kaikki tapahtumat tuli muuntaa ontouchstart-tapahtuman alaisiksi. Tässä vaiheessa demon testaaminen selaimessa muuttui todella vaikeaksi, koska selaimet eivät tunnistaneet ontouchstart-tapahtumaa, joten jokainen tapahtuma joka haluttiin testata selaimessa mahdollisten virheiden varalta, tuli suorittaa erikseen Web Inspectorin komentoriviä hyväksi käyttäen. Luonnollisesti Xcoden simulaattori tarjosi hyvän testialustan tässä vaiheessa, mutta JavaScript-koodin tutkiminen sen avulla oli lähes mahdotonta. Päällisin puolin kuitenkin pyrittiin kehitysvaiheen testaaminen suorittamaan pääsääntöisesti koko ajan itse testilaitteessa. Tällä tavoin pystyttiin helposti löytämään jatkokehittelyä vaativia elementtejä ja toiminnallisuuksia.

Demon viimeistely testaus oli hyvin yksinkertaista ja nopeaa toimintaa. Demo asennettiin kolmeen eri iPad 2 -laitteeseen ja tämän jälkeen luovutettiin markkinointihenkilöille, jotka tulisivat käyttämään demoa markkinoinnissa. Tällä tavoin jo yhden päivän aikana pystyttiin löytämään nopeasti puuttuvia toiminnallisuuksia ja havaitsemaan muita puutteita jotka kaipasivat välitöntä korjausta. Demon lopullinen viimeistelyvaihe kesti noin puolitoista viikkoa ja sinä aikana elementtien animaatioita muokattiin järkevimmiksi ja yhdenmukaisemmiksi. Tässä vaiheessa sovellukseen myös lisäiltiin viimeisiä toimintoja joita markkinointihenkilöt olettivat tarvitsevansa pystyäkseen mahdollisimman tehokkaasti hyödyntämään demoa markkinointiesityksiensä tukena.

6 JATKOKEHITYSMAHDOLLISUUDET

TV-Compass-demon tärkeimmät jatkokehittelymahdollisuudet alkuvaiheessa pohjattiin sovelluskehyksen tarjoamaan mahdollisuuteen siirtää demosovellus alustalta toiselle ilman suurempaa koodin muokkaamista. Demolle on suunniteltu Android-version luomista, sekä mahdollisesti myös demon toiminnallisuuksien laajentamista vielä laajemmaksi kokonaisuudeksi. Alkuperäisen suunnitelman mukaan demon olisi pitänyt koostua kolmesta osasta, joista itse tablet-laitteeseen toteutettu käyttöliittymädemo oli tärkein ja sen lisäksi olisi vielä toteutettu yksinkertainen palvelinpuolen osanen, jonka avulla olisi pystytty luomaan tablet-laitteelle tv-vastakappale. Tässä vaiheessa syystä tai toisesta osat jätettiin toteuttamatta ja jätettiin odottamaan jatkokehittelyä varten.

Demon laajennettavuus ja jatkokehittely on ollut koko kehityskaaren ajan esillä, ja mahdollisuuksia on monia. Luonnollisesti kaikista mielenkiintoisin henkilökohtaisella tasolla olisi päästä toteuttamaan ja viemään sovelluskehyksellä luotu demo valmiiksi sovellukseksi asti käyttäen hyödyksi sovelluskehyksen tarjoamia mahdollisuuksia. Totta kai tässä vaiheessa sovelluksen saattaminen pelkästä käyttöliittymädemosta valmiiksi sovellukseksi vaatisi huomattavasti enemmän kehitystyötä ja aikaa, mutta silti olisi mielenkiintoinen nähdä millaiseen muotoon tässä vaiheessa luodut toimivat konseptit ja ideat loppujen lopulta päätyisivät.

7 POHDINTA

Opinnäytetyöprojektin keskeisin ja tärkein saavutus oli saada luotua näyttävä ja toimivia demosovellus, jota pystyttäisiin hyödyntämään markkinoinnissa etsittäessä mahdollisia uusia asiakkaita. Lähtökohtaisesti koko projekti tähtäsi tähän tilanteeseen, jossa markkinointipuolen henkilöillä olisi käsissään näyttävä ja toimiva demo kokonaisuus, joka on toteutettu nopeasti ja tehokkaasti. Nykyaikaisessa sovelluskehityksessä nopeus ja tehokkuus ovat suuria myyntivaltteja, koska ei kukaan halua maksaa turhasta kehitystyöstä ylimääräistä. Luonnollisesti jos demo olisi toteutettu natiiviohjelmointikielellä, olisi lopputulos voinut olla hieman erilainen, ja luonnollisesti projekti olisi vaatinut enemmän aikaa. Tässä tilanteessa kuitenkin PhoneGap-sovelluskehyksen mahdollisuudet demon nopean luomiseen olivat kaikista kriittisimmät ja tärkeimmät kriteerit, miksi juuri PhoneGap-sovelluskehys alun perin hyväksyttiin mukaan projektiin.

TV-Compass-demo on näyttävä esimerkki siitä, kuinka enää ei ole pakko luottaa perinteisiin mobiilisovellusten kehitysmetodeihin, vaan kehittäjillä on tarjolla uusia mahdollisuuksia, joiden ansiosta mobiilisovellusten kehittämistä pystytään nopeuttamaan ja helpottamaan. Valmistunut demo on tässä vaiheessa vielä yhdistetty tiukasti iOS-alustan ja iPad-tablet-laitteen natiivitoimintoihin, mutta demo olisi silti helppo siirtää mille tahansa PhoneGap-sovelluskehyksen tukemalle laitealustalle. Kuten aikaisemmin jo mainitsin, demolle on suunniteltu jatkokehittelymahdollisuuksia, joissa demo olisi tarkoitus siirtää esimerkiksi Android-alustalle. Demon siirtäminen iOS-alustalta Android-alustalle ei olisi mikään ongelma, koska kummassakin laitealustassa PhoneGap-sovelluskehys luottaa samaan WebKit-selainmoottoriin.

Suurimmat ongelmat demon siirtämisessä alustalta toiselle ovat korkeintaan itse fyysiseen laitteen hieman eroava näytön resoluutio, mutta tämän pienen ongelman korjaaminen ei vaatisi muutama tuntia enempää työtä. Periaatteellisella tasolla yhdestä koodista siis pystytään luomaan sovelluskehyksen avulla suoraan kaikille sen tukemille alustoille omat sovelluksensa. Luonnollisesti sovelluksen alustakohtaiset natiiviohjelmointikielellä luodut taustalla pyörivät raskaammat toiminnot tulisi toteuttaa uudelleen, mutta tässäkin vaiheessa sovelluksen kehittäminen olisi silti huomattavasti nopeampaa kuin perinteisillä toteutustavoilla, koska siinä vaiheessa, kun tiedetään tarkasti tarvittavat taustakomponentit, on helppo samanaikaisesti kehittää ne jokaisella

alustalle erikseen. Totta kai tällä hetkellä PhoneGap-sovelluskehys on parhaimmillaan juuri kevyiden ja yksinkertaisten pienten sovellusten kehittämisessä, mutta tällä hetkellä pystyy jo löytämään esimerkiksi pelejä ja laajempia sovelluksia. jotka on luotu pelkästään hyödyntämällä PhoneGap-sovelluskehystä. Ajan saatossa sovelluskehykset tulevat oikeasti tarjoamaan todellakin varteenotettavan vaihtoehdon mobiilisovellusten kehittämiselle. Web-ohjelmointia on ehkä kautta aikain ollut jossain mielessä hieman aliarvostettu ohjelmoinnin ala, mutta tällä hetkellä HTML5:n, CSS3:n ja PhoneGapin kaltaisten sovelluskehysten myötä pystytään web-ohjelmointikieliä ja -tekniikoita hyödyntämään huomattavasti monipuolisemmin.

Itselleni koko opinnäytetyöprojekti oli mielenkiintoinen kokemus, koska pystyin hyödyntämään jo entuudestaan tuttuja ohjelmointikieliä ja viemään jo valmiiksi oppimani asiat huomattavasti pidemmälle. Koko sovelluksen kehittämisen ajan työ oli hektistä, mutta silti mielenkiintoista ja inspiroivaa, koska jokainen pieni muutos ja kokeilu vei sovellusta eteenpäin ja parempaan suuntaan. Luonnollisesti kehitysvaiheessa tuli vastaan tilanteita joissa jouduttiin myöntämään oma erehdys ja yrittämään tietyn toiminnon suorittamista aivan uudesta näkökulmasta. Tässä rajoituksia loivat kehitysalusta sekä sovelluskehys, mutta näiden pienten ongelmien ratkominen oli todella antoisaa ja palkitsevaa. Haastavinta oli pyrkiä löytämään juuri se sovelluskehyksen kannalta toimivin ratkaisu ongelmatilanteissa. HTML5 ja CSS3 tarjoavat web-ohjelmoinnissa niin kattavat mahdollisuudet luoda näyttäviä kokonaisuuksia, että tällä hetkellä pelkkä mielikuvitus alkaa olla rajana.

LÄHDELUETTELO

Adobe Illustrator. 2011. Saatavissa: http://en.wikipedia.org/wiki/Adobe_Illustrator. Hakupäivä 30.5.2011

Adobe Photoshop. 2011. Saatavissa: http://en.wikipedia.org/wiki/Adobe_Photoshop. Hakupäivä 30.5.2011

CSS3. 2011a. Saatavissa: http://www.css3.info/. Hakupäivä 30.5.2011

CSS3. 2011b. Saatavissa: http://en.wikipedia.org/wiki/Cascading_Style_Sheets. Hakupäivä 30.05.2011

Hatcher, Timothy 2006. Introducing the Web Inspector. Saatavissa: http://www.webkit.org/blog/41/introducing-the-web-inspector/. Hakupäivä 30.5.2011

HTML5. 2011a. Saatavissa: http://en.wikipedia.org/wiki/HTML5. Hakupäivä 30.05.2011

HTML5. 2011b. Saatavissa: http://dev.w3.org/html5/spec/Overview.html. Hakupäivä 30.05.2011

iOS. 2011. Saatavissa: http://en.wikipedia.org/wiki/IOS_(Apple). Hakupäivä 30.05.2011

iPad. 2011. Saatavissa: http://en.wikipedia.org/wiki/IPad. Hakupäivä 30.05.2011

iPad2. 2011. Saatavissa: http://en.wikipedia.org/wiki/IPad_2. Hakupäivä 30.05.2011

iScroll 4. 2011. Saatavissa: http://cubiq.org/iscroll-4. Hakupäivä 30.05.2011

JavaScript. 2011. Saatavissa: http://en.wikipedia.org/wiki/JavaScript. Hakupäivä 30.5.2011

jQuery. 2011a. Saatavissa: http://docs.jquery.com/How_jQuery_Works. Hakupäivä 30.5.2011

jQuery. 2011b. Saatavissa: http://en.wikipedia.org/wiki/JQuery. Hakupäivä 30.5.2011

PhoneGap. 2011. Saatavissa: http://docs.phonegap.com/. Hakupäivä 30.5.2011

Safari Features. 2011. Saatavissa: http://www.apple.com/safari/features.html. Hakupäivä 30.5.2011

TextWrangler. 2011a. Saatavissa: http://www.barebones.com/products/textwrangler/. Hakupäivä 30.5.2011

TextWrangler. 2011b. Saatavissa: http://en.wikipedia.org/wiki/TextWrangler. Hakupäivä: 30.5.2011

WebKit. 2011a. Saatavissa: http://www.webkit.org/. Hakupäivä 30.5.2011

WebKit. 2011b. Saatavissa: http://en.wikipedia.org/wiki/WebKit. Hakupäivä: 30.5.2011

Xcode. 2011. Saatavissa: http://en.wikipedia.org/wiki/Xcode. Hakupäivä: 30.5.2011