Opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

Multimediatuotannon suuntautumisvaihtoehto

2009

Niko Englund

TULOSKORTTI –OHJELMA MOBIILIALUSTALLE (iPHONE)

– CASE: SUPERGOLF OY

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietojenkäsittely | Multimediatuotanto

Syksy 2009 | 28

Ohjaaja | Ari Hietala

Niko Englund

Tuloskortti–ohjelma mobiilialustallle (iPhone) case:Supergolf Oy

Opinnäytetyöni tarkoituksena on tehdä sähköinen tuloskortti-ohjelma. Ohjelman käyttöliittymä tulee olemaan helppokäyttöinen ja ulkoasu pelkistetyn tyylikäs. Ohjelma tuo lisäarvoa Supergolf Oy:n palveluihin ja tarjoaa nopean ja helpon ratkaisun tasoituskierroksen päivittämiseen. Eli golfissa jokaisella pelaajalla on tasoitus, joka kirjataan valtakunnalliseen tietokantaan. Jos pelaaja haluaa nostaa tai laskea tasoitustaan pitää hänen lähettää tuloskorttinsa jäsenseuransa toimistoon joko maililla tai viemällä kortti suoraan toimistolle. Tämä ohjelma yksinkertaistaa kyseisestä prosessia. Enää ei tarvita tietokonetta ja internet-yhteyttä kortin lähettämistä varten.

Alustaksi valitsin iPhonen, sen kasvavan asiakasmäärän ansiosta. Ohjelma on myös helppo julkaista iPhonelle ilman julkaisijoita ja isoa rahallista panostusta.

Raportissa kerron, miten ohjelman suunnittelin ja miten päädyin toteuttamaan suunnittelemani asiat. Käyn läpi tuotantovaiheen ongelmat ja kerron projektin johtopäätökset.

Selvitän myös, miten älypuhelimien käytön lisääntymisen myötä myös ohjelmien tekeminen on lisääntynyt ja mitkä ovat niiden tulevaisuuden näkymät.

ASIASANAT: golf, puhelinlaitteet, ohjlemointi, ohjelmistotuotanto.

BACHELOR´S THESIS | ABSTRACT

UNIVERSITY OF APPLIED SCIENCES

Bachelor of business admistration | Multimedia production

Fall 2009 | 28

Instructor: Ari Hietala

Niko Englund

Scorecard program for mobile device (iPhone) case:Supergolf Oy

The purpose of this thesis is to make a scorecard program for iPhone. Programs interface will be easy to use and appearance should be classy. Program will bring value to Supergolf Oy`s services and will offer fast and easy solution for updating players handicap. In golf each player has a handicap that is recorded to national database. If player would like to raise or lower his handicap he has to send scorecard to his membership club via email or deliver it by himself. This program will make this process easy and simple. You don´t need Internet connection and computer to send your scorecard anymore.

I chose for the platform iPhone simply because of it`s growing usage. Program is also easy to publish whitout a large financial investment.

In this raport i will tell you how i designed this program and how i made it happen. I will also go trough problems that i had in developing this program.

Also i will tell about history and future of mobile phones and applications.

ASIASANAT: golf, mobile, programming.

SISÄLTÖ

KUVAT

KUVIOT

LIITE 1

Ohjelman avausikkuna.

LIITE 2

Kentän valinta.

LIITE 3

Kentän tuloskortti.

LIITE 4

Pelatun tuloksen täyttöikkuna.

LIITE 5

Tuloksen lähetysikkuna.

1 Johdanto

Mistä sain idean tehdä mobiili-ohjelma? Jo sana “mobiili” kertoo, että se on liikuteltavissa ja kulkee mukana usein. Ohjelma, joka liittyy golfiin sopii loistavasti juuri kyseiselle alustalle, koska puhelin lukee kätevästi mukana myös golfkentällä. Ohjelmaani on tarkoitus käyttää joko golfkierroksen aikana tai jälkeen, jolloin tulos on vielä pelaajan tuoreessa muistissa. Tähän tarkoitukseen mobiili-ohjelma on paras ja käytettävyydeltään helpoin ratkaisu.

Selvitän myös mitä ovat älypuhelimet ja miksi ne ovat yleistyneet. Käyn läpi älypuhelimien käyttömahdollisuudet ja rajoitukset. Kerron miten puhelimien lisääntymisen myötä myös ohjelmien tekeminen on lisääntynyt ja mitkä ovat niiden tulevaisuuden näkymät. Käyn läpi myös puhelinten markkinaosuudet ja kovan kilpailun hyvät ja huonot puolet.

Miksi tein ohjelman iPhonelle? Tein ohjelman iPhonelle yksinkertaisesti siksi, koska se kiinnostaa minua eniten tällä hetkellä. Kyseiselle alustalle on myös helpoin saada ohjelma levitykseen Applen app storen takia. Kerron myös tämän vaikutuksista mobiilialan markkinoihin.

iPhonelle on myös saatavilla parhaat kehitystyökalut, joilla alkuun pääseminen on suhteellisen helppoa. Tietenkin monimutkaisten ja paljon sisältöä vaativien ohjelmien tekeminen on haastavaa. Kerron, miten suunnittelin kyseisen ohjelman ja mitä asioita otin huomioon. Toteutuksessa yritän selvittää, miksi päädyin tekemiini ratkaisuihin ja miten toteutin ne.

2 Tausta

Golfissa jokaisella pelaajalla on oma henkilökohtainen tasoitus eli handicap (HCP). Pelaajan tasoitus muodostuu hänen pelaamiensa kierrosten lyöntimäärien mukaan. Tasoitus siis elää pelaajan taitojen mukaan nousten tai laskien.

Tasoituksen avulla lasketaan pelattavalle kentälle pelitasoitus. Pelitasoitus määrää sen, kuinka monta lyöntiä pelaaja saa hyödykseen pelattavalla kentällä. Alussa, kun pelaaja aloittaa golfin, hänen henkilökohtainen tasoituksensa on 54. Tasoitus voi

laskea nollaan ja jopa sen alle. Kun pelaajan tasoitus on yli 36, puhutaan klubitasoituksesta ja sen alla olevat ovat pelitasoituksia. Pelaajan tasoituksen ollessa 9,4 tai alle puhutaan single pelaajista. Golfin ammattilaisilla ei ole tasoitusta.

Suomessa pelaajan tasoitus määritellään hänen pelaamansa pistebogey-tuloksen mukaan. Pistebogey on pelimuoto, jossa pelaaja saa jokaiselta pelaamaltaan reiältä pisteitä sen mukaan, kuinka hyvin hän kyseisen reiän pelasi omiin taitoihinsa nähden.

Tämä ohjelma tulee littymään vahvasti golfissa käytettävään tasoitusjärjestelmään. Se on järjestelmä, jonka avulla voidaan laskea, kuinka hyvin pelaaja on pelannut kierroksen omiin taitoihinsa ja omaan tasoitukseensa nähden. Tasoitusjärjestelmän ansiosta tulokset ovat keskenään vertailukelpoisia pelaajien tasoeroista ja kentän vaikeustasosta riippumatta.

Kierroksen jälkeen oman tasoituksensa voi laskea itse. Kuitenkin vain oman jäsenseuran caddiemaster voi syöttää tasoituksen tietokantaan. Caddiemaster on siis henkilö, joka työskentelee golfkentällä ja ottaa vastaan ajanvarauksia ja huolehtii asiakkaista. Heillä on siis tunnukset tietojen syöttämistä varten, Joten aina, kun käy vieraalla kentällä pelaamassa ja haluaa tuloksen vaikuttavan tasoitukseen, pitää joko lähettää täytetty tuloskortti postilla tai viedä se henkilökohtaisesti paikalle tai lähetttää sähköpostia. Tässä kohtaa tekemäni ohjelman edut tulevat esille. Eli tarvitsee vain näppäillä tulos ohjelmaan ja valita kenttä listalta. Ohjelma lähettää tuloskorttisi automaattisesti halutulle jäsenseuralle ja päivittää näin tasoituksen ajantasalle. (Wikipedia Tasoitusjärjestelmä.)

2.1 iPhone

iPhone on kosketusnäytöllinen Applen älypuhelin iPod-musiikkisoittimen mediaominaisuuksilla. Applen ensimmäisen puhelinmallin lanseerasi Applen toimitusjohtaja Steve Jobs Macworld-messuilla 9. tammikuuta 2007. iPhone julkaistiin Yhdysvalloissa 29. kesäkuuta 2007 ja se saapui Eurooppaan loppuvuodesta 2007. Suomessa ja monessa muussa uudessa myyntimaassa iPhonen uudistetun 3G-version jakelu alkoi samanaikaisesti 11. heinäkuuta 2008. (Wikipedia iPhone.)

Ensimmäisessä iPhonessa oli tuki GSM- ja EDGE-matkapuhelinverkkoteknologioille ja langattomat Bluetooth 2.0- ja WLAN-yhteydet. Vuonna 2008 markkinoille tulleessa

toisessa versiossa iPhone sai tuen 3G- ja HSDPA-matkapuhelinverkkoteknologioille sekä GPS -navigoinnille, jolloin sillä voidaan käyttää nopeita 3G-mobiililaajakaistayhteyksiä. Puhelimen käyttöjärjestelmänä on iPhone OS, jossa on mukana Safari-nettiselain, iTunes, Google Maps, Youtube-sovellus ja tuki asennettaville sovelluksilla, joita voi ladata App Store -verkkokaupasta. iPhonessa ei ole tukea Flash- tai Java-tekniikalle. Google Maps navigointi sovellukseen tuli version 2.2 myötä useita parannuksia mutta ääni-opasteet puuttuvat.
iPhonessa on myös kahden megapikselin kamera. Kamerassa ei ole salamaa, ja videokuvaus edellyyttää erillisen ohjelmiston asentamista. Puhelin on varustettu 3,5 tuuman kosketusnäytöllä, jonka resoluutio on 320x480 pikseliä. iPhonen käyttöliittymässä ei ole perinteisiä fyysisiä painikkeita kahta lukuun ottamatta, vaan puhelinta käytetään pääasiassa kosketusnäytön avulla. Puhelin on vajaan kahdentoista millimetrin paksuinen ja painoa sillä on 135 grammaa. Puheajan luvataan olevan kahdeksan tuntia, musiikkitoistossa 24 tuntia. Puhelimessa on sisäänrakennettu akku joka ei ole käyttäjän vaihdettavissa. Akun kesto kuormitettuna on saanut kritiikkiä osakseen. (Wikipedia iPhone.)

Suosio perustui osaltaan hienoon ja isoon kosketusnäyttöön, jota ohjattiin erillisen kynän sijasta sormella. Myöskin uniikki käyttöliittymä ja ulkoasu olivat myyntivaltteja. Käyttöliittymä esitetty kuvassa 1.

Päivitetty 3G-malli tuli myyntiin 11. heinäkuuta 2008 yhtäaikaan 22 maassa. Näiden mukana oli myös Suomi. 3G-tuen lisäksi uudessa mallissa oli sisäänrakennettu GPS-vastaanotin sekä uusittu 2.0-ohjelmistoversio. Vuoden päästä Apple päivitti jälleen iPhone mallistoa sarjan kolmannella versiolla, joka kantoi nimeä 3GS. Siinä oli päivitetty kamera, nopeampi suoritin, tehokkaampi näytönohjain ja kompassi. (Wikipedia iPhone.)

Kuva 1.iPhonen kosketusnäytöllinen käyttöliittymä.

2.2 Mobiiliohjelmien historia

Ensimmäinen älypuhelimeksi luokiteltava laite oli vuonna 1993 julkaistu IBM Simon. Yhteistyössä BellSouth -operaattorin kanssa kehitettyä tuotetta kutsuttiin nimellä 'personal communicator'. Siinä oli kosketusnäyttö ja tavanomaisten puheominaisuuksien lisäksi laitteen ominaisuuksia olivat hakulaite, sähköposti, kalenteri, osoitekirja, laskin ja kynällä käytettävä luonnoslehtiö. Laite kuitenkin kärsi mittavista teknisistä ongelmista vielä julkaisunsakin jälkeen, ja lopulta IBM päätti vetää sen pois markkinoilta vuonna 1994. Vasta 90-luvun keskivaiheilla älypuhelimet tulivat kokokansan tietoisuuteen, kun Nokia julkaisi vuonna 1996 ensimmäisen kommunikaattorinsa. Se nousi nopeasti yrityspuhelimien kärkeen. (Wikipedia Älypuhelin.)

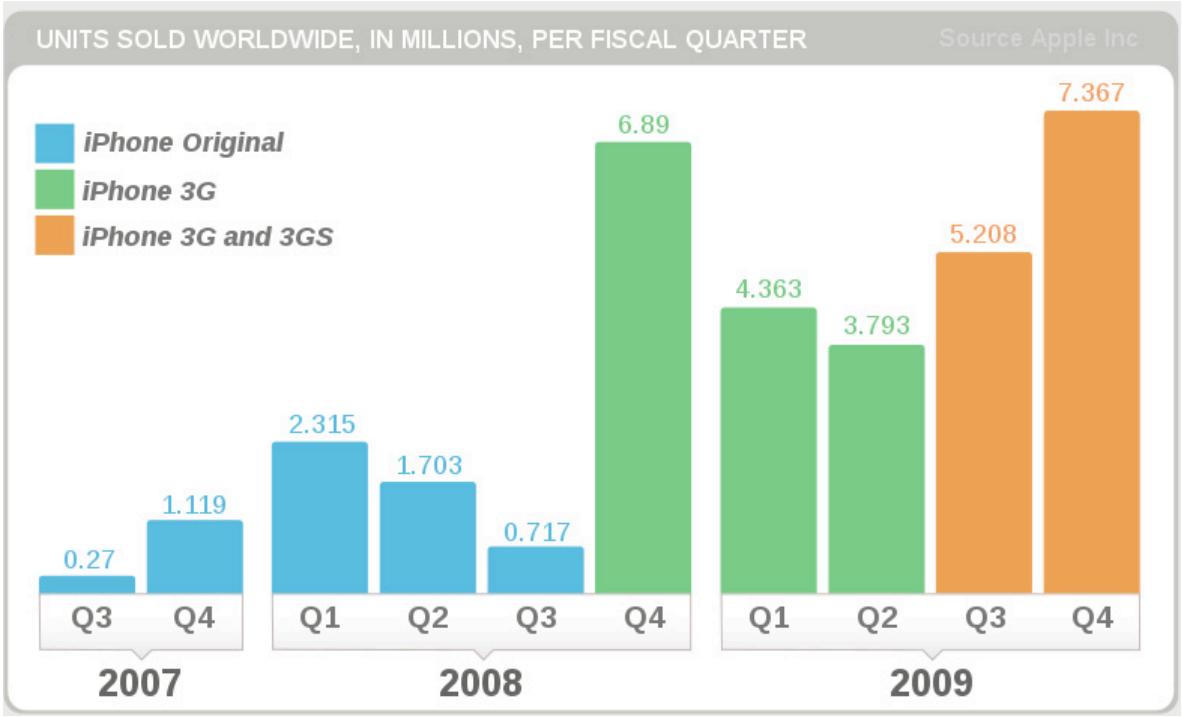
Kuitenkin vasta 2000-luvulla älypuhelimet ovat yleistyneet niinsanottujen ”normaalikäyttäjien” keskuudessa. Tiennäyttäjänä on toiminut Nokia, joka toi

markkinoille paljon kohtuuhintaisia älypuhelimia. Nämä älypuhelimet käyttivät symbian-ohjelmistoalustaa (Wikipedia Alypuhelin).

Ohjelmistokehittäjien oli helppo päättää, mille alustalle tehdä ohjelmia, sillä Symbian hallitsi markkinoita vuoteen 2007 asti seitsemänkymmentäkahden prosentin markkinaosuudella. iPhone kuitenkin mullisti tämän. (It Facts. 2007.)

2.3 Mobiiliohjelmien tulevaisuus

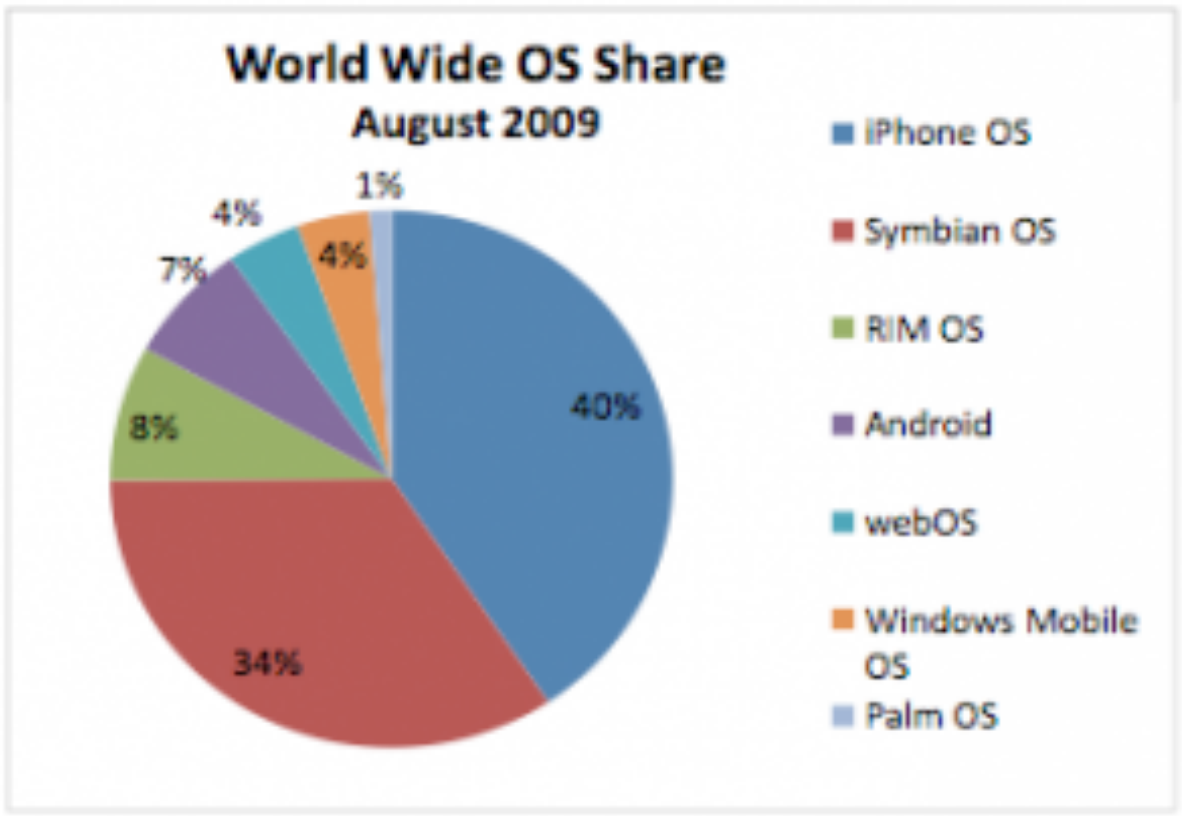
Kun ensimmäinen iPhone julkaistiin kesäkuussa vuonna 2007, merkitsi se uusien älypuhelimien aikakauden alkua. Kun iPhonea myytiin ensimmäisen puolentoista vuoden aikana yli 10 miljoonaa yksikköä, heräsivät muutkin laitevalmistajat mukaan kilpailuun. Nokia ei ainakaan virallisesti kertonut huolestuneensa suurista myyntiluvuista, sillä ensimmäistä iPhonea myytiin lähinnä Yhdysvalloissa, missä Nokia oli panostanut vain vähän älypuhelimien myyntiin. Eniten myynnissä kärsi yhtiö nimeltä RIM, joka valmistaa kuuluisia BlackBerry puhelimia. Hiljalleen markkinoille alkoi valua kosketusnäytöllisiä puhelinmalleja monilta eri valmistajilta, kuten Nokialta ja Samsungilta. (Wikipedia iPhone eng.)



Kuvio1. iPhonen myynnin osuus vuoteen 2009 mennessä. (Wikipedia iPhone eng.)

iPhonen suurimpana valttina pidetään sen ohjelmistoalustaa, jolle periaatteessa kuka vaan voi tehdä ohjelmia. Myös ohjelmien saaminen jakeluun on kohtalaisen helppoa App Store-ohjelman avulla. Tämä ohjelmien jakamistapa on ollut iPhonen suurin valttikortti kilpailussa Nokiaa ja muita älypuhelimien valmistajia vastaan. Peli- ja ohjelmakehittäjien on ollut helppo lähestyä iPhonen ohjelmistoalustaa ilmaiseksi jaettavissa olevan sovelluskehitys-ohjelmiston ansiosta. Nokia ei ole tässä onnistunut, vaikkakin se juuri julkaisi iTunes Storen niin sanotun kilpailijan, Ovi-palvelun. Ovi-palvelusta voi ostaa ohjelmia, pelejä ja musiikkia. Samoihin aikoihin Nokia julkaisi myös sovelluskehitys-ohjelmistopaketin, jota voi pitää selkeänä heräämisenä tähän mobiiliohjelmien kilpailuun. (Ben-Aaron, D. 2009.)

Tämä seikka varmisti, miksi minunkin kannatti valita mobiilialustaksi juuri iPhone.



Kuvio 2. Älypuhelimien käyttöjärjestelmien markkinaosuudet elokuussa vuonna 2009. (Elmer-DeWitt, P 2009.)

2.4 iTunes Store

iTunes Store on Applen 28. huhtikuuta 2003 julkaisema verkkomusiikkikauppa, jossa myydään nykyään myös elokuvia ja TV-sarjoja. Applen toimitusjohtaja Steve Jobs onnistui saamaan viisi suurta levy-yhtiötä mukaan palvelun julkistamiseen. iTunes oli ensimmäinen tuottoisa ja suosittu verkkomusiikkikauppa. iTunes Storesta ostettavissa olevat musiikitiedostot ovat Applen omassa DRM-kopiosuojatussa AAC-formaatissa, ja niiden bittivirta on 256 kbps. Toukokuussa 2007 EMI -levy-yhtiön kappaleita alettiin jakaa ilman DRM-suojausta ja paremmalla laadulla, mutta hieman korkeammalla hinnalla. DRM-vapaiden kappaleiden valikoima on tämän jälkeen laajentunut muidenkin levy-yhtiöiden valikoimiin. iTunes Storesta ostettuja DRM-suojattuja musiikkikappaleita voidaan soittaa ainoastaan iTunes-soitto-ohjelmalla sekä Applen omilla iPod-musiikkisoittimilla. Ostettuja musiikkikappaleita voidaan polttaa rajoittamattomalle määrälle CD-levyjä, kuunnella rajoittamattomalla määrällä iPodeja ja soittaa viidessä eri tietokoneessa. Suojaamattomia kappaleita voidaan soittaa missä tahansa soittimessa, joka tukee AAC-formaattia. Tunes Storesta tuli maailman suurin musiikin jälleenmyyjä vuonna 2008. USA:n iTunes Storessa on nykyään myynnissä myös yli 20 000 TV-sarjaa ja vuokrattavana on yli 1 000 elokuvaa. iTunes Store vuokraa elokuvia yli 50 000 kappaletta päivässä, tehden siitä myös maailman suosituimman Internetissä toimivan elokuvakaupan. (Wikipedia iTunes Store.)

2.4.1 App Store

App Store avattiin heinäkuussa 2008. Se on iTunes Storen lisäosa, joka mahdollistaa käyttäjien selailla ja ostaa ohjelmia, jotka on tehty iPod Touch ja iPhone mobiililaitteille. Storessa myydään ohjelmia, jotka ohjelmistokehittäjät ovat tehneet käyttäen iPhone SDK-kehitysohjelmistoa. Apple seuloo läpi jokaisen ohjelman, joka halutaan julkaista ja ottaa myynnistä itselleen kolmekymmentä prosenttia. Silti julkaisun helppous on ollut se lähtökohta, minkä ansiosta ohjelmistojen määrä oon kasvussa. Suuri osa iPhonen myynnin tuloksesta perustuu juuri sille tarjottavien ohjelmien ja pelien määrään. Uusia ohjelmistokehittäjiä tulee jatkuvasti lisää markkinoille ja ohjelmien määrä vain kasvaa entisestään. (Wikipedia App Store) Tämän takia minunkin oli helppo valita, mille alustalle ohjelmani teen.

Tällä hetkellä App Storessa on ladattavissa satatuhatta ohjelmaa tai peliä ja luku kasvaa kokoajan. Lokakuussa vuonna 2009 oli kaupasta ladattu yhteensä jo yli kaksi miljardia ohjelmaa. (Wikipedia App Store.)

2.5 Toimeksiantaja

Toimeksiantajana toimii Supergolf Oy. Supergolf Oy on vuoden 2004 alussa perustettu yritys. Se on loistava konsepti, joka tarjoaa alennettuja hintoja hieman arvokkaana pidettyyn lajiin.

Supergolfin päiväpelioikeuksilla voi pelata golfia useilla kentillä edullisesti Suomessa ja Virossa. Supergolfin päiväpelioikeudet varataan käyttöön aina internet-palvelusta. Kaudella 2010 Supergolfin valikoimassa on lähes 50 eri kenttää ja kenttäkohtaisten päiväpelioikeuksien määrä vain kasvaa.

Supergolfin päiväpelioikeuksia voi käyttää liittymällä Supergolfin jäseneksi. Tasoitus 54 riittää Supergolfin päiväpelioikeuksilla pelattaessa. Pelaaja tarvitsee vain voimassaolevan tasoituskortin, eli tulee olla jäsenenä jossain golfseurassa. Päiväpelioikeus on käytössä koko päivän.Useimmilla kentistä voi pelata useita kierroksia päivän aikana. (Supergolf. 2009.)

Teen ohjelman, joka tuo lisäarvoa Supergolfin asiakkaille. Ohjelma lisää myös yrityksen näkyvyyttä ja antaa pienen lisäedun kilpailijoihin verrattuna.

3 Suunnittelu

Ohjelmistosuunnittelu koostuu kahdesta vaiheesta. Näistä ensimmäinen on toiminnallinen määrittely. Siinä kuvataan kaikki järjestelmän toteuttamat toiminnot ja liitännät järjestelmän ulkopuolelle. Vaiheen tuotoksena syntyvä *määrittelydokumentti* kuvaa siis mitä kaikkea järjestelmällä voi tehdä sekä miten *käyttäjä* voi ne tehdä. Järjestelmä ei kuvaa sitä, miten toiminnot tulee *toteuttaa*. Näin määrittelydokumentti kuvaa esimerkiksi tekstinkäsittelyohjelman eri näytöt, valikot ja niissä olevat asetukset ja kaikki käyttöliittymäkontrollit. Mikäli järjestelmä tuottaa tulosteita, niin

määrittelydokumentin tulisi sisältää sekä visuaaliset että riittävät tekstuaaliset kuvaukset kaikista järjestelmän eri näytöistä.

Ideaalinen määrittelydokumentti on niin kattava, että teknisessä suunnittelussa tai sitä seuraavissa vaiheissa ei ole enää missään vaiheessa epäselvää, että miten ohjelman tulee toimia kussakin tilanteessa, miten tulosteet ja syöte on muotoiltu jne. Vaikka tähän on käytännössä mahdotonta päästä, niin on kuitenkin havaittu, että siihen tulee pyrkiä.

Tyypillinen ohjelmistoalan ongelma on se, että määrittelydokumentteja ei jakseta kirjoittaa. Jonkinlainen vaatimusanalyysi saatetaan tehdä, mutta sen jälkeen aletaankin heti koodata ohjelmaa. Ajatellaan, että määrittelydokumentin kirjoittamiseen menevä aika ei ole verrannollinen siitä saatavaan hyötyyn. Kuitenkin asia on juuri päinvastoin, koska asioita on paljon helpompi muuttaa luonnollisella kielellä kirjoitetusta dokumentista kuin ohjelmakoodista, ja suoraan koodia kirjoittavat huomaavat hyvin pian tekevänsä määrittelyä samaan aikaan kuin kirjoittavat ohjelmaa. Näin ohjelma muodostuu ad hoc -ratkaisujen varaan, ja isoista ohjelmista muodostuu rakenteellisesti hyvin vaikeaselkoisia ja sitä kautta erittäin virhealttiita rakennelmia.

Toiminnalliselle määrittelylle vastakohta on tekninen määrittely. Tekninen määrittely seuraa toiminnallisen määrittelyn jälkeen, ja siinä ei enää tehdä valintoja tai päätöksiä siitä, millaisia ominaisuuksia laitteessa on – sen tarkoituksena on kuvata ohjelmiston tai ohjelman tekninen arkkitehtuuri tarkasti, joskaan ei vielä ohjelmakoodina. Syötteenä sille toimii toiminnallinen määrittely, ja se kuvaa ne ohjelmistokomponentit, jotka toteuttavat määrittelyn vaatimat toiminnot.

Tekniseen määrittelyyn sisältyy esimerkiksi käytetty ohjelmointikieli/kielet, ohjelmistokomponentit kuten kirjastot, ohjelmistokomponenttien rakenne ja keskinäinen hierarkia, käytetyt tietorakenteet ja niiden väliset sovellusrajapinnat jne.

Huomattavaa on, että ohjelmiston koosta ja monimutkaisuudesta riippuen dokumentti koostuu usean eri kerroksen kuvauksista. Esimerkiksi kompleksissa ohjelmistossa ensimmäisellä tasolla voi näkyä eri moduulit, mistä järjestelmä koostuu: palvelinprosessi, asiakasohjelmaclient-prosessit, tietovarastot sekä niiden liitynnät ympäristöön. Toisella tasolla kukin näistä komponenteista on pilkottu pienempiin osiin kuten luokkiin, ja niistä näkyy tarkemmin, mitä ulkoiset liitännät ovat. Kolmannella tasolla on kuvattu luokkien metodit ja luokkamuuttujat, käytettyjen tietorakenteiden tyypit. (Mikkonen, T.2004. Mobiiliohjelmointi.)

3.1 iPhone SDK-Sovellustyökalu

Tärkein työkalu suunniteltaessa ohjelmistoa iPhonelle on ehdottomasti SDK (software development kit). Se julkaistiin maaliskuussa vuonna 2008. SDK mahdollisti kaikkien kehittäjien suunnitella ja toteuttaa ohjelmia iPhone- ja iPod Touch-alustoille. Kyseessä oli siis täydellinen paketti joka sisälsi seuraavat ohjelmat, Xcode, Interface Builder, Instruments, Dashcode ja Simulator. Kyseinen ohjelmisto oli ilmainen, mutta vaati rekisteröitymisen. Simulator-ohjelmalla pysyttiin testaamaan tehtyjä ohjelmia. Jos kuitenkin halusi julkaista ohjelmansa App storessa täytyi maksaa niin sanottu julkaisulisenssi. Lisenssin hinta vaihtelee jos kyseessä on yksityinen julkaisija tai yritys. Ohjelmantekijä saa myynnistä seitsemänkymmentä prosenttia, loput kolmekymmentä menevät Applelle. Tekijä toki säästää myös julkaisuun liittyvät kustannukset, kun perinteisesti on pitänyt valita julkaisija ja maksaa tälle tietty provisio tai julkaisemismaksu. Tämän sovellustyökalun ansiosta kehittäjillä oli entistä matalampi kynnys aloittaa iPhone -ohjelmien suunnittelu ja toteutus. (Wikipedia SDK.)

3.1.1 Xcode

Xcode tarjoaa erilaisia tapoja käsitellä projektia käyttäjänsä mieltymysten mukaan. Löhdekooditiedostoja voidaan tarkkailla ja muokata projektin pääikkunassa, tai ne voidaan avata omiin ikkunoihinsa. Myös sisennystapaa ja koodin syntaksin värjäystä voi muokata Xcoden asetuksissa. Niitä kannattaa vilkaista pikaisesti, mutta mikäli vaihtoehdot eivät vaikuta helposti tunnistettavilta, ne voi jättää myös oletusarvoihinsa.

Projektin pääikkuna jakautuu kolmeen pääalueseen. Ylhäällä on ikkunan työkalupalkki. Työkalupalkissa kannattaa pitää ainakin komennot projektin kääntämistä ja ajamista varten, sekä painike tekstieditorin näyttämistä varten.

Vasemmalla on näkymä projektin ryhmä- ja tiedostohierarkiaan. Ryhmät ovat loogisia käsitteitä, jotka eivät välttämättä vastaa projektin fyysistä tiedostohierarkiaa. Keskeisimmät ryhmät tässä vaiheessa ovat Classes, Other Sources ja Resources, jotka löäytyvät varsinaisen projektin alta. Targets-ryhmä kuvaa projektin kohteita, esimerkiksi luotavan ohjelman määrityksiä. Executables puolestaan sisältää lopulliset käännetyt

tuotokset.

Ikkunan pääosan vie tiedostolista tai editori. Näiden näkyvyyttä voi muuttaa paneelissa näkyvällä pienellä pyöreällä säätimellä, tai vaihtoehtoisesti työkalupalkin Editor-painikkeella.Varsinaisen editorin yläosassa on navigointipalkki, jossa on työkalut eri tiedostojen välillä liikkumiseen sekä tiettyyn metodiin siirtymiseen nykyisessä tiedostossa.

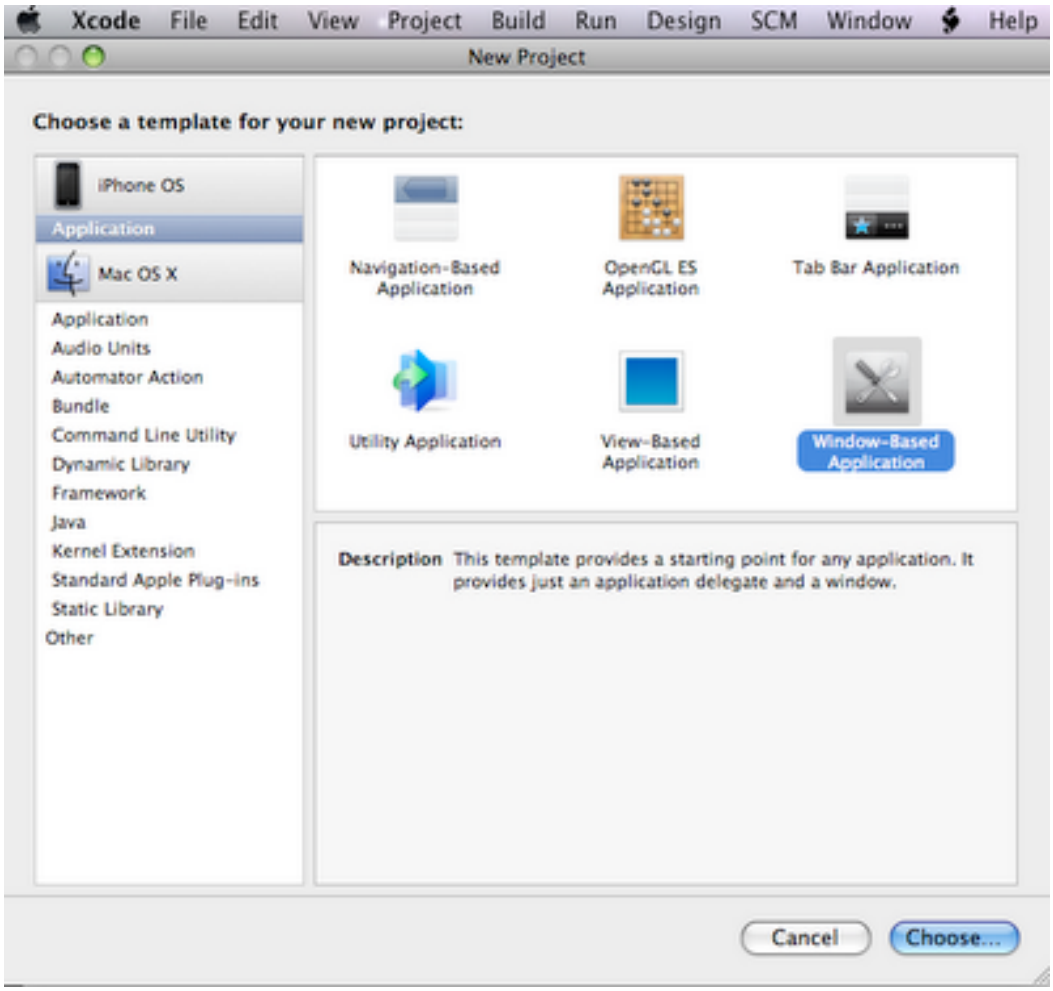
Projekti käännetään valitsemalla Build-valikosta toiminto Build. Vaihtoehtoisesti voidaan käyttää työkalupalkissa olevaa vasarapainiketta. Kun projekti käännetään, voidaan kääntämisen tuloksia tarkastella Build Results -ikkunassa, joka saadaan näkyviin Build-valikosta. Se näytetään myös automaattisesti, mikäli kääntäjä löytää ohjelmasta virheitä. Virheitä osoittamalla ne saadaan näkyviin editorissa.

Käännetty ohjelma voidaan käynnistää suoraan Xcodesta, ja sen ajoa seurata Run Log -ikkunassa, joka puolestaan avautuu Debug-valikon kautta. Tähän ikkunaan ohjautuvat esimerkiksi ohjelmaan mahdollisesti sijoitetut konsoliviestit.

Joskus voi tulla tarpeeseen tyhjentää projektin senhetkiset tuotokset ja aloittaa kääntäminen puhtaalta pöydältä. Tällöin voidaan käyttää Clean tai Clean All -toimintoja. Ne hävittävät projektin käännetyt tuotokset sekä kääntämisen yhteydessä luodut väliaikaistiedostot. Yleensä projekti täytyy tyhjentää, jos esimerkiksi sen olemassaolevaan luokkarakenteeseen halutaan tehdä muutoksia.

Jos haluaa muokata ohjelmaa graafisessa ymäristössä, klikkaa vain auki tiedoston, mitä haluaa muokata ja ohjelma avaa tiedoston Interface builder-nimiseen ohjelmaan.

Keskellä ylhäällä on ehkä ohjelman testaamisen kannalta tärkein nappi. “Build and Run” napista ohjelma avautuu “iPhone simulator” nimiseen ohjelmaan, joka on käytännössä tietokoneen sisällä toimiva iPhone puhelin. Simulaattori auttaa käsittämään ohjelman käyttöliittymää ja helpottaa testausta suunnattomasti. (Goldstein, N. 2009.iPhone Application Development for Dummies)



Kuva 2. Xcoden projektin valitsemis-ikkuna.

3.1.2 Objective- C

Object-C on saanut alkunsa vuonna 1980 ja ensimmäinen kirja (Object-Oriented Programming, An Evolutionary Approach) siihen liittyen, on julkaistu vuonna 1986. Kieltä käytetään nykyään lähinnä Mac OS X:n ja iPhone OS:n kehittämiseen. Se on primäärinen kieli, jota Apple käyttää Cocoa API:ssaan. Kieli on vahvasti kytköksissä Mac OS:in eri versioihin ja versio numeroinnin sijasta usein ilmoitetaankin, että kieli on tukenut tiettyä ominaisuutta tietyn Mac OS:in versiosta lähtien. Eräs suurin asia, joka on otetut huomioon kehitettäessä sitä, on ollut suurten koodimassojen hallinta. Se kopioinut viestinvälitys tapansa SmallTalkista. Objective-C:n kaksi kirjasta kattaa pääosin Applen Cocoa API:n. API:ssa on rajapinta myös Javaan. Sitä kutstutaan Java-sillaksi, mutta se on määritelty poistuvaksi ominaisuudeksi Mac OS 10.4:stä lähtien; tarkoittaen sitä, että ominaisuudet, jotka lisätään Cocoaan sen jälkeen, eivät tule lisätyksi Java- siltaan myös C++:n kirjastot toimivat Object-C:ssä.

Kieli toimii periaatteessa, kuten perinteinen C-kieli, mutta siinä on mukana oliot, joille on oma syntaksi ja semantiikka. Object-C:n kääntäjällä voi kääntää C-kielellä kirjoitettuja ohjelmia, ja se onkin oikeastaan vain kerros C-kielen päällä. Osissa, joissa käytetään kielen olio-ominaisuuksia, id on perustietotyypi. Tästä voidaan päätellä, että kieli on dynaamisesti tyypitetty. Metodin kutsuminen ei aina Object-C:ssä tarkoita sitä, että olio implementoi kyseisen viestin. Yhtälailla se voi välittää sen jollekkin toiselle olioille, joka tietää, mitä tehdä viestille. Toimintaa sanotaan joustavaksi viestinvälitykseksi. Nil tarkoittaa Object-C:ssä samaa, kun null ja se palautuu, jos viestiä ei voida käsitellä. Kaikki kielen perustyypit ovat määriteltynä otsikkotiedostossa.

(Wikipedia Objective-C.)

Objective-C -kielessä erotetaan yleensä luokan rajapinta ja toteutus erillisiin tiedostoihin. Tämä tapa periytyy kielen C-historiasta. C:ssä usein kuvataan ohjelmakomponentin funktiot erillisessä otsikkotiedostossa, ja niiden toteutus varsinaisessa lähdekooditiedostossa. Objective-C -kielessä käytetään otsikkotiedoston tunnisteena tiedostopäätettä .h ja lähdekooditiedoston tunnisteena päätettä .m . Varsinainen tiedostonimi on yleensä sama kuin kuvattavan luokan nimi.

Objective-C -kielessä erotetaan yleensä luokan rajapinta ja toteutus erillisiin tiedostoihin. Tämä tapa periytyy kielen C-historiasta. C:ssä usein kuvataan ohjelmakomponentin funktiot erillisessä otsikkotiedostossa, ja niiden toteutus varsinaisessa lähdekooditiedostossa. Objective-C -kielessä käytetään otsikkotiedoston tunnisteena tiedostopäätettä .h ja lähdekooditiedoston tunnisteena päätettä .m . Varsinainen tiedostonimi on yleensä sama kuin kuvattavan luokan nimi. (Objective-C Beginner’s Guide.)

Otsikkotiedosto voisi olla esimerkiksi seuraavanlainen:

```
#import <Cocoa/Cocoa.h>

@interface OmaLuokka : NSObject
{
      NSString *nimi;
}

-(NSString) nimi;
```

```
-(void) asetaNimi: NSString *uusiNimi;

@end
```

Attribuutit esitellään samaan tapaan kuin Javassa, mutta eräs ero on huomioitava. Samoin kuin Javassa, Objective-C:ssä tehdään ero primitiivityyppien ja oliomuuttujien välille. Objective C:n käyttämät primitiivityypit ovat samat kuin C-kielessä. Perustyypit ovat int, float, double ja char. Näistä on olemassa eri variaatioita, kuten etumerkittömät kokonaisluvut unsigned int, ja pitkät ja lyhyet numerot, kuten long double ja short int. Primitiivityyppejä käytetään samaan tapaan kuin Javassa, esimerkiksi määre int numero; kertoisi kyseessä olevan kokonaisluvun, jota kutsutaan nimellä "numero".

Oliomuuttujiin viitataan Objective-C:ssä osoittimien kautta. Näitä merkitään tähdellä. Näin yllä olevan esimerkin NSString *nimi tarkoittaa osoitinta (viittausta) NSString-tyyppiseen olioon, jota kutsutaan nimellä "nimi". Osoitin voi tässä tapauksessa viitata joko NSString-tyyppiseen olioon tai jonkin NSString-luokan aliluokan olioon. Minkä tahansa tyyppiseen olioon voidaan viitata määreellä id.

3.1.3 Interface Builder

Interface Builder auttaa suunnittelemaan käyttöliittymiä ohjelmiisi. Tällä ohjelmalla voi luoda käyttöliittymän valitsemalla säätimiä määriteltävissä olevien elementtien kirjastosta ja järjestelemällä ne asetteluohjainten avulla. Ohjelmiin voi lisätä vaikuttavia Core Animation -pohjaisia toimintoja, siirrostehosteita käyttöliittymätilojen välillä tai kolmiulotteisia varjoja säätimiin muutamalla osoituksella. Lisäksi Interface Builderilla on helppo testata käyttöliittymiä, koska jokaisen säätimen sijoittaminen ja yhdistäminen toimintaan lähdekoodissa tehdään tyylikkään ja tehokkaan graafisen käyttöliittymän kautta. (Developer Connection Interface Builder.)

Interface Builder koostuu useista pienistä ikkunoista ja erilaisista paneeleista. Ohjelman varsinainen pääikkuna kuvaa käsiteltävän NIB-tiedoston sisällön ja muun projektiin liittyvät resurssit. Ikkuna jakautuu viiteen paneeliin: Instances, Classes, Images, Sounds ja Nib. Instances-paneeli esittää NIB-tiedostoon tallennetut oliot ikoneina. Yleensä nämä ovat erilaisia ikkunoita, valikkoja ja ohjaustason luokkien edustajia. Näiden lisäksi

kuvataan kaksi konseptuaalista elementtiä, jotka jokaisella NIB-tiedostolla on.

File's Owner kuvastaa oliota, joka "omistaa" ohjelmaan ladatun NIB-tiedoston. Se voi olla esimerkiksi ohjelman pääluokka, tai jokin ohjaustason luokka, joka on ladannut kyseisen NIB-tiedoston ohjelmaan. First Responder puolestaan kuvaa abstraktisti ensimmäistä oliota, joka käsittelee jonkin viestin, jolle ei ole määritelty nimenomaista vastaanottajaa. Tällaisia viestejä ovat esimerkiksi tiedon kopiointi ja sijoittaminen, joiden tulee kohdistua yleensä kulloinkin aktiiviselle käyttöliittymäelementille jonkin tietyn elementin sijaan.

Classes-paneeli esittää projektissa käytettäviä luokkia hierarkiana, joka toimii Finderin palstanäkymän tavoin. Sen avulla voidaan määrittää projektiin uusia luokkia, yleensä ohjaustasolle, sekä luoda niille lähdekoodirungot suoraan Xcodeen.

Images ja Sounds -paneelit kuvaavat projektissa käytettäviä kuva- ja ääniresursseja, ja Nib-paneeli esittää NIB-tiedoston itsensä asetukset. Keskeisimmät apuikkunat ovat käyttöliittymäpaletti ja inspektori. Ensimmäinen sisältää prototyypit tavallisille käyttöliittymäelementeille, ja toinen kuvaa yksittäisen käyttöliittymäelementin, luokan, tai muun kohteen ominaisuuksia. Molemmissa ikkunoissa on useita alinäkymiä.

Keskeinen huomion kohde on luonnollisesti muokattava ikkuna tai muu vastaava kohde, kuten valikkorivi. Käyttöliittymäelementit raahataan paletista ikkunaan, ja niiden ominaisuuksia voi muokata Inspektori-ikkunassa. Muokattavaa käyttöliittymää voidaan kokeilla suoraan Interface Builderissä File-valikon alta löytyvällä Test Interface -komennolla. Interface Builder esittää tällöin rakennettavan käyttöliittymän sellaisena kuin se näkyisi ajettavassa ohjelmassa. Ikkunan kokoa voi esimerkiksi muokata ja tarkkailla, kuinka käyttöliittymäelementit käyttäytyvät. Kokeilutilasta pääsee takaisin Interface Builderin perustilaan lopettamalla kokeilu-ohjelma Komento-Q -näppäinoikotiellä, tai osoittamalla valikkorivissä näkyvää kytkinkuvaketta. Myös käyttöliittymäelementtien liitynnät varsinaiseen ohjauskoodiin tehdään Interface Builderin avulla, samoin kuin ohjausluokkien liityntöjen ja toimintojen alustava määrittely.

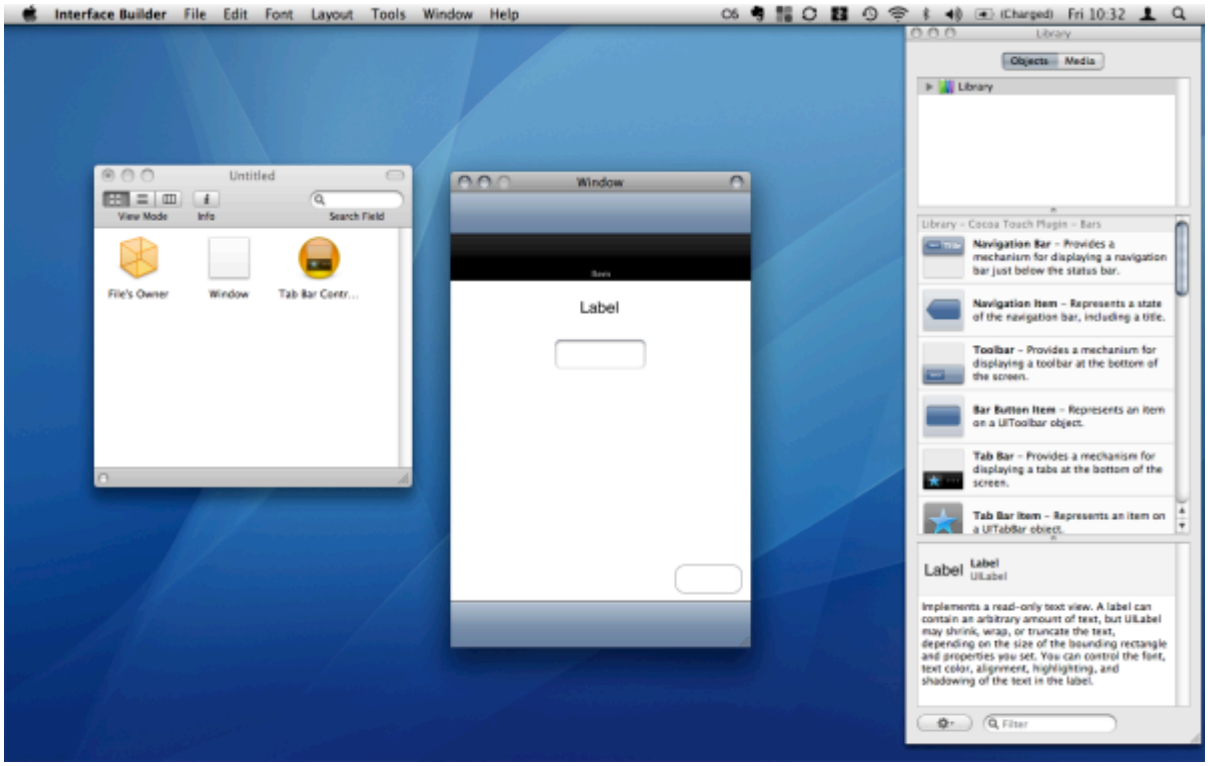
Uusia luokkia voidaan luodaan Interface Builderissä sen pääikkunan Classes-paneelissa. Haluttu yläluokka valitaan hierarkiasta, ja sille luodaan aliluokka Classes-valikon Subclass... -toiminnon avulla. Uudelle luokalle määritellään nimi, sekä liitynnät ja toiminnot Inspektori-ikkunan avulla käyttämällä Add-painiketta Attributes-paneelissa.

Nämä määrittelyt ovat toistaiseksi vielä pelkkiä rajapintamäärityksiä, niitä ei ole varsinaisesti sidottu mihinkään. Varsinainen sitominen tapahtuu instantioimalla luokasta olio ja raahaamalla halutut yhteydet olion ja käyttöliittymäelementtien välille.

Olio voidaan myös instantioida suoraan InterFace Builderissä. Instantioitu olio tallennetaan käyttöliittymäelementtien ohella NIB-tiedostoon, josta se voidaan ladata ohjelmaan sen ajon aikana. Varsinkin ohjaustason luokat instantioidaan usein tällä tavalla. Instantiointi tapahtuu valitsemalla haluttu luokka Classes-paneelista, ja valitsemalla Instantiate... -toiminto Classes-valikosta. Uusi ilmentymä ilmestyy näkyviin kuvakkeena Instances-paneeliin. Tämän jälkeen olion yhteydet käyttöliittymäelementteihin määritellään raahaamalla viiva olion kuvakkeesta haluttuun käyttöliittymäelementtiin control-näppäin painettuna. Inspektori-ikkuna ehdottaa Connections-paneelissaan sopivaa liityntää, joka voidaan hyväksyä Connect-painikkeella. Toiminnot puolestaan vedetään samalla tavalla käyttöliittymäelementeistä olion kuvakkeeseen. Jälleen Inspektori-ikkunassa valitaan halutun toiminnon nimi ja vahvistetaan se.

Olemassaolevia olion liityntöjä käyttöliittymäelementteihin voidaan muokata irrottamalla haluttu kytkentä Disconnect-painikkeella ja sen jälkeen raahaamalla uusi yhteys samalla tavalla kuin uutta yhteyttä luotaessa. Mikäli luokkaan lisätään uusia toimintoja tai liityntöjä Xcodessa, pitää muutoksista kertoa Interface Builderille ennen kuin uusia yhteyksiä niihin voidaan luoda. Tämä tapahtuu käyttämällä IBOutlet ja IBAction -määritteitä lähdekoodissa uusien liityntöjen ja toimintojen yhteydessä, ja sen jälkeen raahaamalla luokan otsikkotiedosto Interface Builderin pääikkunaan.

Lopullinen käyttöliittymä tallennetaan NIB-tiedostoon, joka kääntämisen yhteydessä kopioidaan varsinaiseen ohjelmapakettiin. On jopa mahdollista avata käännettyyn ohjelmapakettiin tallennettu NIB-tiedosto ja muokata sitä ilman, että ohjelmaa tarvitsee kääntää uudestaan.



Kuva 3. Interface Builderin käyttöliittymä.

3.2 Tilaajan rooli

Tilaaja antoi vapaat kädet itse ohjelman suunnitteluun. Ulkoasuun heillä oli muutamia ideoita, joita otin vastaan. Tilaajan kanssa sovimme sisällöstä sen verran, että sen tulisi kattaa kaikki Supergolf Oy:n yhteistyökentät, jotta sillä olisi suora yhteys tilaajan muihin palveluihin. Sovimme myös, että ohjelma toteutetaan englannin kielellä, koska Supergolfilla on yhteystyökumppaneita myös Suomen ulkopuolella ja myös heidän tulisi pystyä käyttämään ohjelmaa.

Tilaajan kanssa sovimme myös, että luovutan kaikki tekijänoikeudet heille ja, että he voivat halutessaan muokata ohjelmaa. Näin molemmat osapuolet ovat varmasti tulokseen tyytyväisiä.

4 Toteutus

Ennen varsinaista toteutusvaihetta perehdyin iPhonen ohjelmointikirjallisuuteen. Minun lähtökohtani oli se, että en aikaisemmin ole ollut hirveästi kiinnostunut ohjelmoinnista. En myöskään ollut toteuttanut monia ohjelmointi-projekteja. Kirjojen kautta oppiminen tapahtui lähinnä esimerkkiohjelmia tehden. Se ei kuitenkaan avannut minulle kovin hyvin kyseisen ohjelman tekemiseen tarvittavia tietoja. Lähdin kuitenkin rohkeasti toteuttamaan näkemystäni toivoen, että matkan varrella oppisin lisää ja onnistuisin ohjelmassa suunnitellulla tavalla.

4.1 Ulkoasu

Ulkoasun suunnittelu oli ainoa asia, minkä suhteen olin yhteydessä tilaajaan. Hänen toivomuksensa oli, että ohjelman ulkoinen olemus olisi linjassa heidän web-sivujensa tyylin kanssa. Tästä syystä sain luvan käyttää heidän web-sivuillaan olevaa kuvaa ja muodostin alkuikkunan tyyliin sen mukaisesti. Eli käytin otsikossa samantyylistä kirjoitusasua. Alaosan kuva taas on sama kuin heidän web-sivujensa yläosan kuva. Näin web-sivuilla vieraillut ohjelman käyttäjä tunnistaa heti alkuikkunasta lähtien, että kyseessä on saman firman tuote. Päätimme myös käyttää ohjelman kielenä englantia, sillä Supergolf Oy:llä on yhteistyökenttiä muunmuassa Virossa. Muuten ulkoasu noudattaa hillittyä linjaa, sillä halusin ohjelmasta hyvin helppokäyttöisen ja sujuvan.

4.2 Käyttöliittymä

Käyttöliittymää tehdessäni halusin sisällyttää ohjelmaan vain sen käytön kannalta tärkeitä ominaisuuksia. Tästä syystä ohjelmassa on vain tärkeimmät toiminnot ja hillitty ulkoasu, eikä se näin ollen vaadi käyttäjältään insinöörin tutkintoa.

Kun ohjelma avataan aukeaa aluksi otsikkoikkuna, missä käytin yrityksen logoa ja kuvaa, joissa on yhtenäisiä piirteitä heidän web-sivujensa kanssa. Kyseinen toiminto on toteutettu käyttäen seuraavaa koodia.

```
-        (void)applicationDidFinishLaunching:(UIApplication
*)application {
    sleep(5);
```

```
// Override point for customization after app launch
    [window addSubview:[navigationController view]];
    [window makeKeyAndVisible];
```

Siinä on määritetty ohjelman avausikkunan toiminnot. Eli ilman tätä toimintoa ohjelma avautuisi suoraan kenttävalinta ikkunaan. Sleep toiminto määrittelee ajan kuinka kauan haluaa niin sanotun splash screenin esiintyvän ennen siirtymistä kenttävalintaikkunaan. (Goldstein, N. 2009.iPhone Application Development for Dummies)

Toisessa ikkunassa valitaan kenttä, jossa pelaaja on pelannut tai minkä tuloskortin hän haluaa täyttää. Kun pelaaja valitsee kentän tulee näytölle kyseisen kentän tuloskortin tiedot. Tässä ikkunassa voi tarkastella kentän tietoja. Painamalla ”Scorecard”-nappia siirrytään valikkoon, missä itse tuloskortti täytetään. Tuloskorttia voi täyttää kierroksen aikana tai vasta pelatun kierroksen jälkeen. Kun tuloskorttiin laittaa yhden väylän tuloksen, voi sen jälkeen siirtyä takaisin ”back”-painikkeella ja katsoa seuraavan väylän tiedot. Lisätyt tulokset jäävät muistiin ja näin ollen tuloksia voi lisätä haluamallaan tavalla. Tulosten täyttöikkuna sisältää myös valikon, josta valitaan minkä värisiltä tiiltä pelaaja kierroksensa pelaa. Tämä tieto on tärkeä, koska sen avulla caddiemaster voi laskea kierroksen kokonaistuloksen vaikutuksen tasoitukseen. Alhaalla on myös kohta kokonaiskierroksen lisäystä varten, mikä on myös tarkeä tieto tasoituksen laskemisen kannalta. Viimeinen lisättävä kohta on ”Hcp” eli pelaajan nykyinen tasoitus, jota tarvitaan tuloksen kirjaamiseen. Nämä kaikki edellä mainitut ovat siis vähimmäistiedot, joita caddiemaster tarvitsee tasoituksenkierroksen laskemista varten.

Kentän valinta on toteuttettu käyttämällä table list-toimintoa. Kyseinen toiminto mahdollistaa yksinkertauksen listaustavan, joka on kuitenkin hyvin muunneltavissa.

```
@interface RootViewController : UITableViewController {
    NSMutableArray *tableList;
```

Tässä on määritetty mitä näkymää controlloidaan. Alempana on kerrottu minkä tyyppistä listaa näkymässä käytetään.

```
    tableList = [[NSMutableArray alloc] init];
    [tableList addObject:@"Alastaro Golf"];
    [tableList addObject:@"Archipelagia Golf Club"];
       …….
          [self setTitle:@"Select Course"];
```

Määritellään listan objektit ja kyseisen listan otsikko.

```
(void)tableView:(UITableView                    *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath {

    if      ([[tableList      objectAtIndex:indexPath.row]
isEqual:@"Alastaro Golf"])
    {
         SecondViewController          *second          =
[[SecondViewController                              alloc]
initWithNibName:@"SecondViewController" bundle:nil];

[second setTitle:@"Alastaro Golf"];
[self.navigationControllerpushViewController:second
animated:YES];
```

Tässä on määritetty listan toiminto, eli se miten siirrytään eri kenttien objekteihin ja mitä ne pitävät sisällään. Toiminto on määritetty IF-lauseella, eli jos kenttä on määritetty oikein siirrytään kyseisen kentän tuloskorttinäyttötilaan, jota ohjataan uudella näkymäohjaimella. (Goldstein, N. 2009.iPhone Application Development for Dummies)

Painamalla ”Send Card”-painiketta ohjelma avaa sähköpostin lähetystä varten tehdyn sovelluksen. Tämä sovellus kuuluu kehityspaketin mukana tuleviin lisäosiin, mutta on täysin muokattavissa omia tarpeita varten. Se käyttää iPhonen oman

sähköpostisovelluksen ulkoasua. Painamalla + -painiketta ohjelma avaa listan, joka sisältää kaikkien Suomen golfseurojen nimet. Tästä listasta valitaan seura, missä tuloksen pelannut pelaaja on jäsenenä. Jos käyttäjä on jäsenenä ulkomaalaisella kentällä, voi hän täyttää osoitteen manuaalisesti. ”Subject” -kohtaan laitetaan otsikko. Otsikon alapuolelle tulostuu täytetyn tuloskortin tiedot. Tässä kohdassa ne kannattaa myös tarkistaa. Tietojen alle voi vielä lisätä tekstiä esimerkiksi oman nimen. Sitten ei enää tarvitse kuin painaa ”Send”-painiketta ja ohjelma lähettää tuloskortin valitun seuran sähköpostiin. Sieltä jäsenseuran caddiemaster lisää tiedot tietokantaan ja tasoitus päivittyy.

```
@class SecondView;

@interface SecondViewController : UIViewController
<MFMailComposeViewControllerDelegate,UINavigationControlle
rDelegate>{

    IBOutlet SecondView *secondView;

}
- (IBAction)sendMail;
-(IBAction)switchViews;
```

Aluksi loin uuden näkymän, jolle annoin nimeksi SecondView. Tämä toteutetaan IBOutlet käskyllä. Seuraavaksi tein muutaman Action toiminnon, jotka sanansa mukaan ovat toiminta käskyjä. Käyttämällä sendMail toimintoa iPhone avaa sisäänrakennutun mail-toiminnon, joka on helpoin tapa toteuttaa tämä osa ohjelmasta. (Goldstein, N. 2009.iPhone Application Development for Dummies)

4.3 Testaus

Testaus tapahtui iPhone simulator-ohjelmalla. Ohjelma on käyttöliittymältään täysin identtinen oikean puhelimen kanssa. Jotta ohjelma saadaan käynnistettyä iPhone simulatorilla, täytyy se tarkistaa Xcodilla. Xcodissa ohjelma testataan eli katsotaan sisältääkö koodi lauserakenteellisia virheitä tai onko ohjelma muuten käyttökelpoinen

esimerkiksi muistin käytön suhteen. Eli myös ohjelman rakenne täytyy olla kunnossa. Jos ohjelma on kunnossa se käynnistyy suoraan simulator ohjelmaan. Jos ohjelmassa on virheitä niin Xcode kertoo, missä kohdassa virheet sijaitsevat ja mitä ne ovat.

Testausvaiheessa ongelmia tuotti erityisesti pienet kirjoitusvirheet itse koodin sisällä. Ongelmat olivat kuitenkin helposti korjattavissa. Suurin ongelma tuli, kun aloin testaamaan sähköpostiominaisuutta. iPhone simulator ei sisällä sähköpostiominaisuutta, joten testaaminen piti suorittaa itse iPhone laitteella. Ohjelman voi siis syöttää iPhone-laitteeseen, kunhan se on ensin konfiguroitu sovellukehityspaketin mukana tulevilla ohjeilla. Itse en myöskään tässä vaiheessa enää omistanut laitetta, joten minun piti lainata se kaverilta. Sain ohjelman kuitenkin hyvin ladattua puhelimeen ja testattua sähköpostiominaisuuden toimivuutta. Testausprosessi kesti noin viikon kaikkine korjauksineen.

5. Johtopäätökset

Vaikka alussa kirjoitin, miten oikeastaan kuka vaan voi nykyään tehdä ohjelmia iPhonelle, en ehkä enää ole aivan samaa mieltä. Soveluksen tai pelin tekoprosessi on aikaa ja hermoja raastava projekti. Luulin myös, että sovellustyökalun avulla ohjelma muotoituisi melkein itsestään. Tämä ei pitänyt paikkaansa alkuunkaan. Koodiin tutustuminen oli tärkein osa prosessia ja kehitystyökalut sananmukaisesti vain työkaluja ohjelman tekemiseen.

Opin tämän prosessin aikana paljon asioita ohjelmatuotannosta ja uskon, että tulen myös jatkossa kehittelemään iPhonelle sovelluksia. Tämä projekti antoi myös hyvän perustan kaikkiin sovelluskehittämiseen liittyviin asioihin, joten jatkossa samanlaisten projektien tekeminen on varmasti helpompaa.

Omasta mielestäni sain aikaiseksi sen, mitä lähdin tekemään. Ohjelma sisältää kaikki ne ominaisuudet, jotka sille alussa määrittelin. Uskon, että ohjelmani tuo hyvää lisäarvoa Supergolf oy:n palveluihin, mikä oli myös yksi tavoitteistani tässä projektissa. En usko, että ohjelma tulee mullistamaan maailmaa, mutta ei se ollut tarkoituskaan. Itse ainakin käyttäisin kyseistä ohjelmaa, joten miksi ei siis muutkin? Mielestäni tuo kysymys on hyvä lähtökohta tulevia projektejani ajatellen. Aion jatkossa tehdä pelejä/ohjelmia, joita itsekin pelaisin, tällöin epäonnistumisen riski on pienempi.

Yhteistyö Supergolfin kanssa oli sujuvaa ja antaa myös uskoa siihen, että jatkossa voimme olla yhteydessä uusien projektien osalta.

5.1 Ongelmat

Ongelmia oli eniten koodausvaiheessa. Kielen rakenteiden ja loogisuuden oppiminen vei aikaa. Onneksi netistä löytyi suuri osa tutoriaalivideoita, jotka auttoivat ohjelman koodauksessa. Kuitenkin suurin osa tuloksista eteni erehdyksien kautta eteenpäin. Eli ensin testasin ja kun ohjelma ei toiminut halutulla tavalla, tutkin mistä se johtui. Tästä syystä testaus vei hyvin paljon aikaa ja ohjelman valmistuminen oli vaakalaudalla.

5.2 Yhteenveto

Lähdin tekemään ohjelmaa, joka olisi helposti lähestyttävä ja antaisi selkeää lisäarvoa tilaajan palvelukokonaisuuteen. Ohjelman tavoitteena oli päivittää helposti omaa golftasoitusta. Nykyinen järjestelmä on sen verran rajoittunut, että pelaajan tarvitsi aina joko lähettää tasoituskortti maililla tai kirjeitse tai viedä tuloskortti suoraan oman seuransa caddiemasterille. Tekemäni ohjelma tekee tästä kaikesta helpompaa. Tavoitteena oli myös saada Supergolffin asiakkaille tästä selvä lisäarvo, mitä muilla ei ole. Mielestäni onnistuin tässä tavoitteessani. Esimerkiksi pelaajalle, joka reissaa paljon ja pelaa usein eri kentillä on tämä sovellus kullan arvoinen,. Varsinkin jos pelaaja haluaa pitää tasoituksensa ajan tasalla. Uskon myös, että jos Supergolf julkaisee ohjelman tuo se heille lisää näkyvyyttä. Voisi siis sanoa, että tavoitteet tuli täytettyä. Ohjelman tekoprosessi oli hyvin antoisa ja vaativa. Asetin tavoitteeksi tehdä hyvin pelkistetyn version, sillä aikaisempaa kokomusta kyseisestä alustasta ei ollut. Tämä osoittautui silti hyvin haastavaksi tehtäväksi juuri tämän kokemattomuuden takia. Toki sain hyvät valmiudet tehdä ohjelmia jatkossa, mikä oli myös yksi tavoitteistani. Opin siis valtavasti uutta ja tästä on hyvä lähteä tekemään muita projekteja. Näistä syistä johtuen tämä oli mielestäni hyvin onnistunut opinnäytetyön aihe. Onneksi satuin olemaan valmiiksi kiinnostunut kyseisestä alustasta, joten tekeminen ei missään vaiheessa ollut pakkopullaa vaikka hyvin haastavaa. Omasta mielestäni ohjelma onnistui hyvin ja edustaa hyvin osaamistani.

Kaikki ongelmat liittyivät itse koodin kirjoittamiseen tai testaukseen. Aikataulutus ei ollut mikään paras mahdollinen, sillä ajattelin pystyväni koodaamaan ajateltua nopeammin. Kuitenkin pikkuvirheet ja kokemattomuus nostivat päätään ja sain tosissani painia ohjelman valmistumisen kanssa. Olen kuitenkin tyytyväinen, että sain sen loppujen lopuksi valmiiksi.

LÄHTEET

Mikkonen, T. 2004. Talentum. Mobiiliohjelmointi.

Mark, D.; LaMarche, J. 2009. Apress.Beginning iPhone Development: Exploring the iPhone SDK.

Goldstein, N. 2009. Wiley Publishing Inc. iPhone Application Development for Dummies.

Elmer-DeWitt,P.30.9.2009. AdMob: iPhone`s share of the smartphone market hits a record 40%. Viitattu 25.10.2009. http://brainstormtech.blogs.fortune.cnn.com/2009/09/30/admob-the-iphones-share-of-the-smartphone-market-hits-a-record-40/

Wikipedia Tasoitusjärjestelmä. Viitattu 3.10.2009. http://fi.wikipedia.org/wiki/Tasoitusjärjestelmä_(golf).

Wikipedia iPhone. Viitattu 15.10.2009. http://fi.wikipedia.org/wiki/Iphone.

Wikipedia Älypuhelin. Viitattu 25.10.2009. http://fi.wikipedia.org/wiki/Älypuhelin.

IT Facts. 2007. Symbian maintains 72% mobile OS market share. Viitattu 25.10.2009. http://www.itfacts.biz/symbian-maintains-72-mobile-os-market-share/8932.

Wikipedia iPhone eng. Viitattu 15.10.2009. http://en.wikipedia.org/wiki/Iphone

Ben-Aaron, D. 2009. Nokia failure to beat iPhone software puts market share at risk. Viitattu 15.10.2009. http://www.bloomberg.com/apps/news?pid=20601087&sid=a.VGNGiOPNCU

Wikipedia iTunes Store. Viitattu 15.10.2009. http://fi.wikipedia.org/wiki/Itunes_store.

Wikipedia App Store. Viitattu 16.10.2009. http://en.wikipedia.org/wiki/App_store.

Wikipedia SDK. Viitattu 15.10.2009. http://en.wikipedia.org/wiki/IPhone_SDK#iPhone_SDK.

Wikipedia Objective-C. Viitattu 25.10.2009. http://en.wikipedia.org/wiki/Objective_c.

Developer Connection Xcode. Viitattu 28.10.2009. http://developer.apple.com/tools/Xcode/.

Objective-C Beginner's Guide. Viitattu 12.12.2009 http://www.otierney.net/objective-c.html/

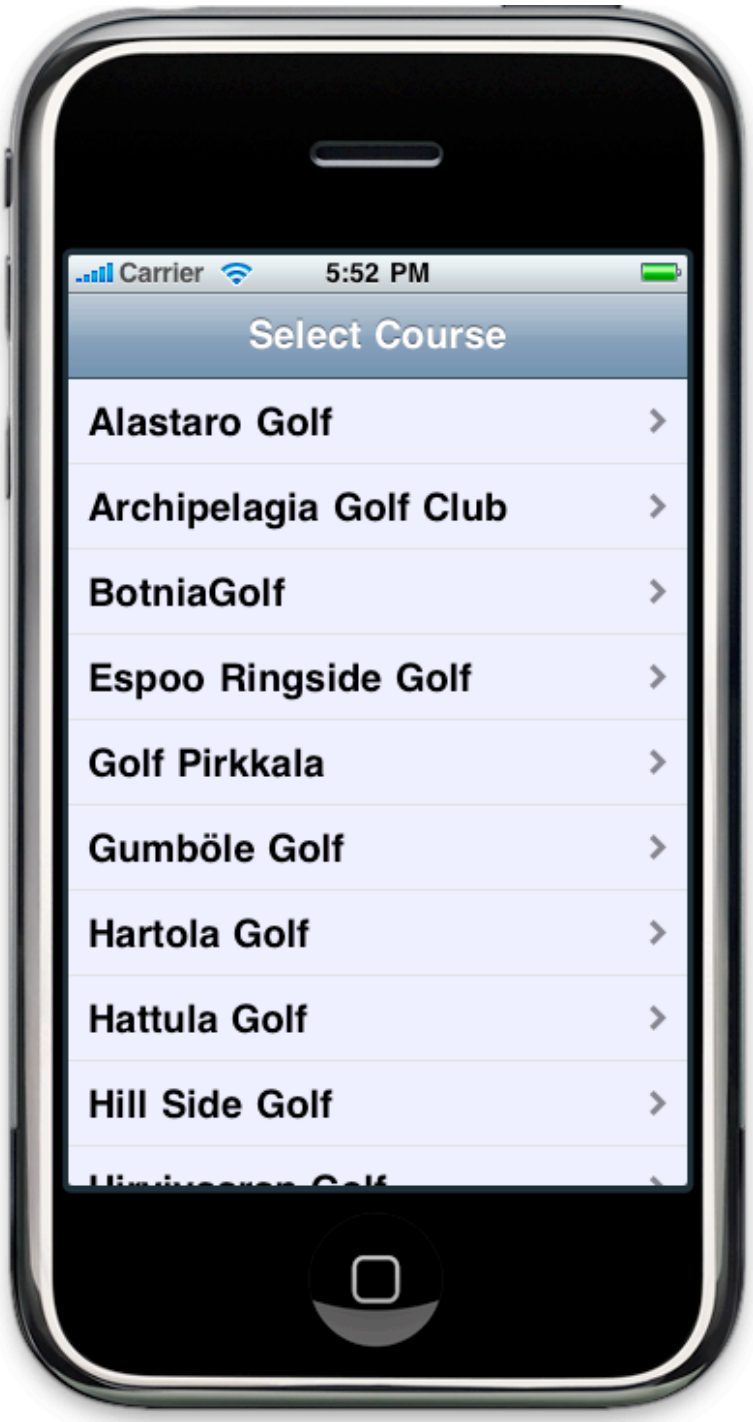
Developer Connection Interface Builder. Viitattu 27.10.2009. http://developer.apple.com/tools/interfacebuilder.html.

Liite 1

Ohjelman avausikkuna.

Liite 2

Kentän valinta.



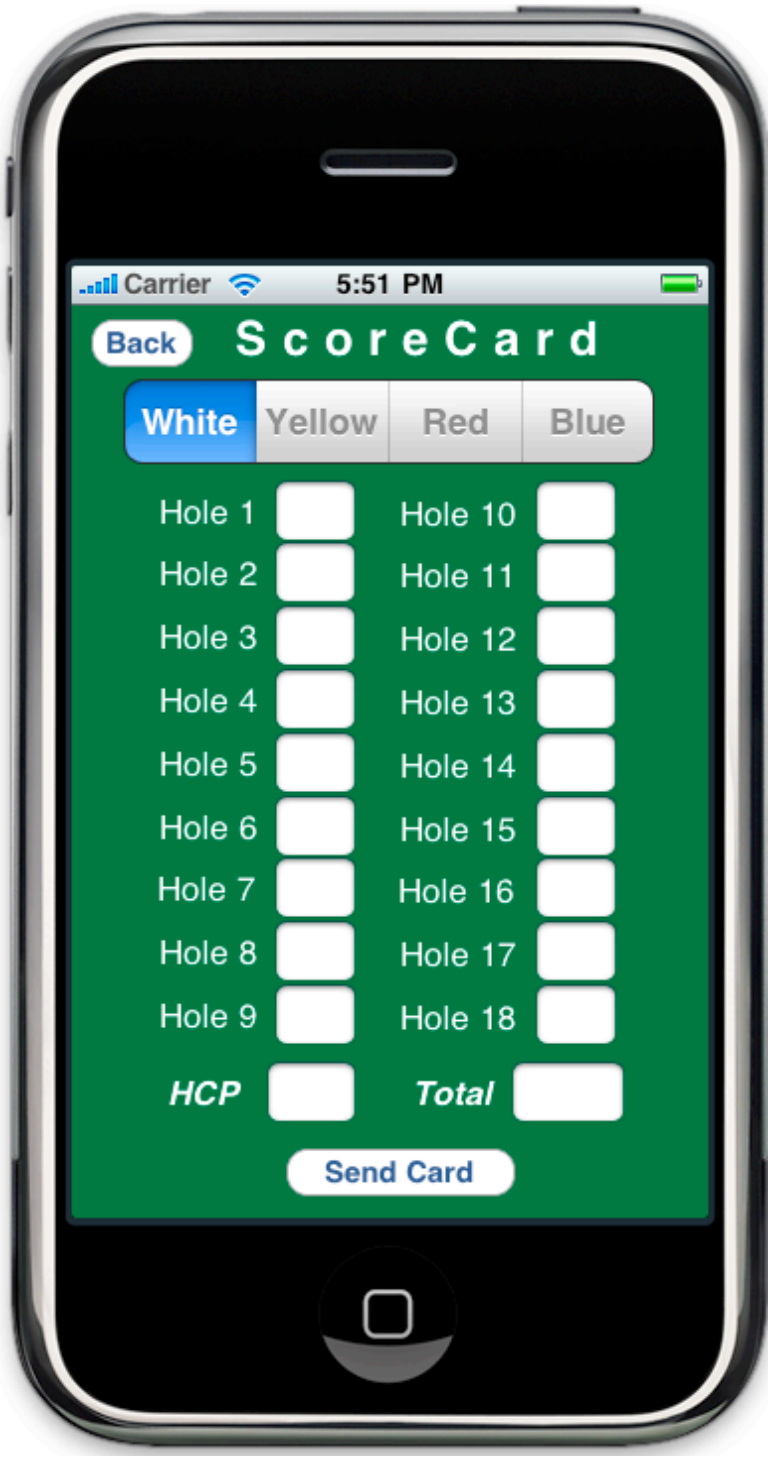
Liite 3

Kentän tuloskortti.

Hole	White	Yellow	Blue	Red	Par	Hcp
1	462	448	431	404	5	8
2	367	352	320	285	4	4
3	142	131	109	97	3	18
4	326	313	300	288	4	12
5	367	353	339	325	4	10
6	406	380	372	335	4	2
7	486	442	432	335	5	6
8	152	135	126	113	3	16
9	340	309	292	270	4	14
10	380	375	351	300	4	1
11	338	324	310	284	4	9
12	372	352	341	310	4	5
13	105	100	92	77	3	17
14	324	311	303	294	4	11
15	332	318	308	276	4	13
16	189	171	153	132	3	15
17	466	448	424	380	5	3
18	377	363	326	320	4	7
Out	3048	2863	2721	2515	36	
In	2883	2762	2608	2373	35	
Total	5931	5625	5329	4888	71	

Liite 4

Pelatun tuloksen täyttöikkuna.



Liite 5

Tuloksen lähetysikkuna.