

iPhone Toepassing

Brecht Van der vekens, 2de Master Computerwetenschappen
Pieter Van Geel, 2de Master Computerwetenschappen
Steven Vercammen, 2de Master Computerwetenschappen

November 23, 2009

Abstract

Dit is het verslag voor opgave 3 van multimedia, van groep 2. Wij hebben een iPhone-toepassing uitgewerkt voor het zoeken van liedjes en artiest-informatie op basis van enkele woorden uit de tekst. De belangrijkste uitdagingen waren het werken in ajax en php, en het werkend krijgen van de coverflow. Uit dit project leren wij vooral dat we bij het ontwikkelen van internet toepassingen niet direct naar 1 specifiek platform mogen kijken; zo hebben we voor deze applicatie een eigen server ontwikkeld die de ontwikkeling van toekomstige applicaties eenvoudiger maakt en welke de vorige applicaties ook had kunnen vereenvoudigen.

1 Idee

Grote verschillen met het idee van het vorige project zijn er niet. Voor deze applicatie werken we met de talen HTML, CSS en PHP en hiermee is het gemakkelijker mooie resultaten te krijgen dan met java. Daarnaast bevat iPhone een YouTube applicatie waarmee eenvoudig videos afgespeeld kunnen worden en deze applicatie kan veel eenvoudiger aangesproken worden dan op het Android platform. Naast het beter gebruik van YouTube, maken we ook meer gebruik van de oriëntatie van het toestel, dit wordt verder uitgelegd bij implementatie.

Op het Android platform was het niet altijd duidelijk of de hardware de oriëntatie van het scherm zou kunnen detecteren, op het iPhone platform is dit daarentegen standaard waardoor het eenvoudiger is om hiermee rekening te houden. Daarnaast heeft een van onze groepsleden een iPhone en was het testen van de oriëntatie dan ook makkelijker.

Omdat we de layout van onze storyboard al hadden moeten aanpassen voor het mobiele platform van de Android gsms, zagen we dan ook geen dringende reden om deze nog extra aan te passen of om extra functionaliteit toe te voegen. Het omvormen van een native-applicatie voor Android naar een web-applicatie was reeds een uitdaging.

We maken maar weinig gebruik van het mobiele aspect van de iPhone, dit omdat een mobiel aspect niet past binnen het storyboard, mobiel naar lyrics en muziek zoeken is hetzelfde als op een desktop pc. De kans dat je een liedje zoekt waarvan je enkele woorden tekst hebt gehoord is daarentegen veel groter op een moment dat je geen computer bij de hand hebt. Bij het ontwikkelen

van het storyboard hadden we hier meer rekening moeten houden.

Als alternatieven hadden we nog enkele ideeën om de applicatie beter te integreren met het iPhone platform. Zo wilden we bij het zoeken naar een liedje ook binnen de iTunes bibliotheek op de iPhone maar omdat we een webpagina ontwikkelden ipv een native applicatie hadden we hier geen toegang toe. Een ander alternatief was om per gevonden nummer een link naar de iTunes store in te voegen om het liedje aan te kopen. Maar dit bleek ook niet mogelijk binnen de tijdslimiet.

Als laatste alternatief, omdat een website maar weinig integratiemogelijkheden bood, hebben we ook een poging ondernomen om een native iPhone applicatie te ontwikkelen. Dit was veel moeilijker dan gedacht; We kwamen enkele problemen tegen: De compiler van c++ staat in leopard op een andere plaats dan in snow leopard dus moesten we die handmatig kopiëren. Eenmaal dat gebeurd was bleek het een verkeerde versie van de compiler te zijn. Ook een verkeerde versie van Xcode vertraagde het proces. Nadien bleek dat we ook een developer certificaat nodig hadden, waarvoor betaald moet worden. Het enige alternatief was om de iPhone te jailbreaken en daarna zelf een certificaat te maken enz.. Dit hebben we nog geprobeerd maar uiteindelijk opgegeven.

Het sterkste punt van ons idee is dat we nu gebruik maken van een eigen server, dit maakt geen verschil op vlak van het storyboard maar maakt de ontwikkeling van volgende toepassingen aanzienlijk eenvoudiger. De server doet alle oproepen naar de verschillende diensten (chartlyrics, last.fm) die we gebruiken en groepeerde de invoer naar een eigen xml structuur. Zo wordt de applicatie zelf veel eenvoudiger en moet deze zelf maar 1 oproep doen per zoekopdracht.

Het zwakke punt van ons idee is dat het zoeken naar lyrics maar weinig gebruik maakt van het mobiele aspect van de iPhone, hier hadden we meer over moeten brainstormen.

2 Storyboard

Het storyboard wijkt bijna niet af van het Android storyboard, het originele idee was om meer mogelijkheden van het iPhone platform te gebruiken maar dit bleek niet altijd mogelijk.

De acties zijn nog grotendeels hetzelfde gebleven. De gebruiker zoekt naar een woord of een zin uit een liedje, hierna geeft de applicatie alle gevonden nummers weer in de bekende coverflow-layout. Wanneer de gebruiker dan op een cover klikt (of in het geval van iPhone “tapt”), geeft de applicatie meer informatie weer over het liedje en een knop om meer informatie over de artiest op te vragen en het liedje te beluisteren op YouTube. Voor de overwogen alternatieven zie deel 1 - Idee.

3 Software-ontwerp

Het software ontwerp voor de applicatie is deze keer in 2 delen gebeurd: een deel was de ontwikkeling van de iPhone webapplicatie en het andere deel was de aparte server. Op deze manier scheiden we het opvragen en verwerken van de informatie naar een correct formaat en het tonen van deze informatie in een bepaalde visualisatie. Wanneer de applicatie wil zoeken naar een deel van een liedje, stuurt deze dat deel door naar de server. Daarna doet de server al het harde werk. Het opvragen van alle lyricids, teksten, artiestinfo, YouTubevideoid, albumhoezen,... gebeurt allemaal



Figure 1: Storyboard

op de server.

Het ontwerp van de server is zelf ook niet complex, om te beginnen doet de applicatie een

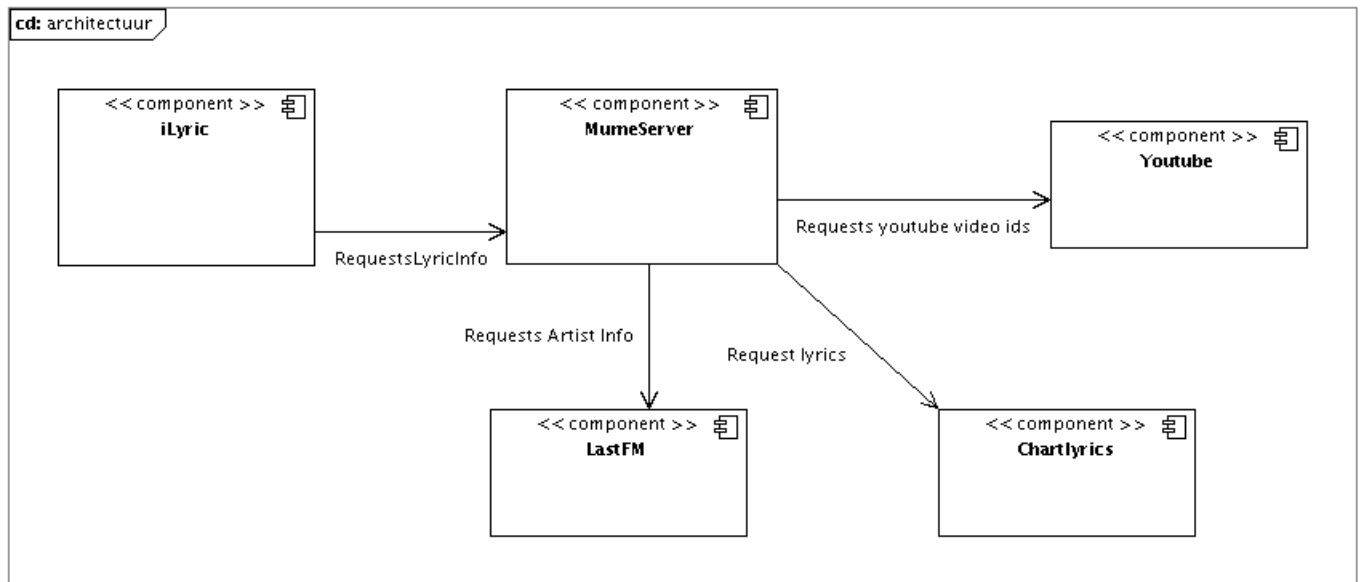


Figure 2: Architecture Diagram

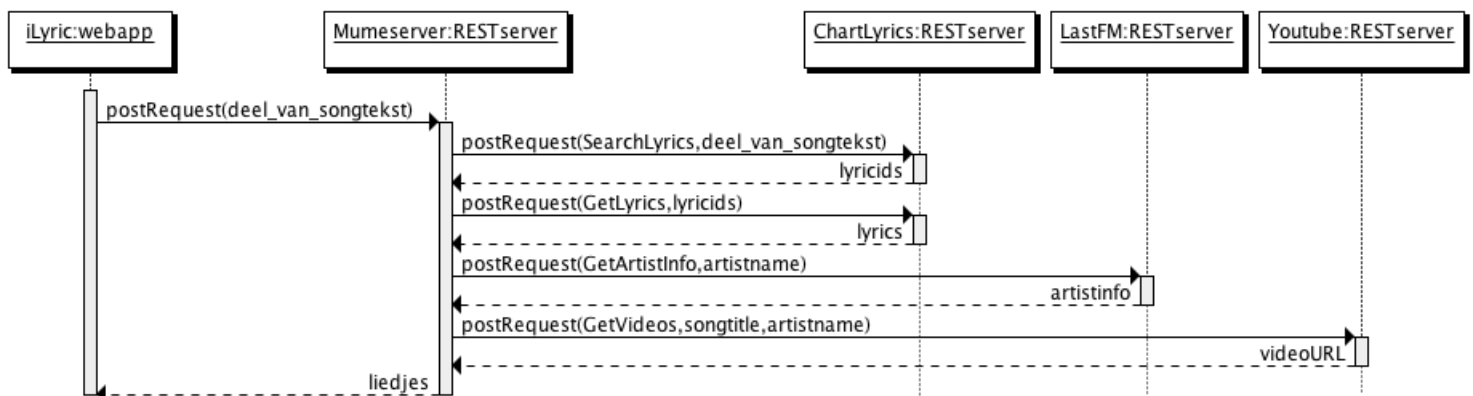


Figure 3: Sequence Diagram

POST-request naar de server met het deel van de te zoeken tekst. Hierna wordt door de server een POST request gestuurd naar chartlyrics en krijgt de server lyricID's terug, met deze ID's kan de server dan de volledige lyrics opvragen. Met de artiestinformatie in deze lyrics vraagt de server dan de artiestgegevens op van lastfm. Met al deze informatie maakt de server dan een eenvoudigere xml bestand aan waarin enkel de nodige informatie zit.

In het ontwerp van de iPhone webapplicatie hebben we 4 pagina's voorzien (Zie figuur 1). Op de eerste pagina is het mogelijk de zoekquery in te voeren. Bij het zoeken wordt deze doorverwezen naar de volgende pagina waar de resultaten zichtbaar zijn. Deze bevat verwijzingen naar de derde pagina waar informatie over de artiest van elk gevonden liedje zichtbaar is. De resultatenpagina bevat ook verwijzingen naar een 4de pagina, dat is de YouTube pagina. De iPhone opent wanneer op de link getapped wordt de ingebouwde YouTube applicatie, deze hebben we dus niet zelf moeten voorzien. Het enige minpunt aan deze aanpak is wel dat eenmaal deze YouTube applicatie geopend is, je blijkbaar terug safari moet openen om terug te keren naar onze applicatie.

Bij het opvragen van pagina 2 wordt de aparte server aangesproken, deze geeft alle informatie direct terug in de juiste vorm.

4 Implementatie

De programmeertalen in deze iteratie zijn PHP, AJAX, HTML en CSS. Na wat zoeken hebben we een coverflow component gevonden die gebaseerd is op HTML en CSS. Deze hebben we dan ook opgenomen in onze implementatie. (Zie: <http://www.satine.org/archives/2008/11/06/coverflow-for-safari-on-iPhone/>).

De oriëntatie van het toestel wordt standaard doorgegeven naar safari die dan kantelt. Wij vangen dit event op met javascript waardoor we bepaalde delen van onze applicatie kunnen verplaatsen of herschalen zodat alles er hetzelfde uitziet.

Alle informatie die we opvragen aan de externe APIs krijgen we in XML formaat. Deze XML bestanden zijn echter veel te uitgebreid en zo ook niet altijd direct voor de hand liggend om te parsen en er de nodige informatie uit te halen. Omdat dit in PHP echter veel gemakkelijker (zowel programmeren als debuggen) is dan in Javascript, hebben we de applicatie dan ook ontworpen om gebruik te maken van een aparte server (geschreven in php). Zo omzeilen we ook direct het feit dan in Javascript geen cross-domain requests mogen gebeuren. We lieten een PHP server de nodige informatie van de externe sites halen en we lieten deze server dan ook de XML vereenvoudigen tot enkel de informatie die we echt nodig hadden.

De programmeeromgeving was deze keer het programma Dashcode. Alhoewel we heel tevreden zijn van het grafische programmeren in dashcode, i.e. dat we componenten gemakkelijk op het scherm kunnen slepen en verslepen, zijn we op veel problemen gestuit om hiermee te werken. Je moet voorzichtig zijn als je met een extern programma bronbestanden aanpast die dashcode gebruikt. In dashcode worden deze wijzigingen pas doorgevoerd als je de bestanden vernieuwt. Ook in de andere richting zijn er problemen. Dashcode houdt zelf de aanpassingen in de bestanden bij en die worden pas doorgegeven aan het filesystem wanneer je het project deployt. Zo is de synchronisatie met het onderliggende bestandensysteem niet constant en dit heeft tot vele frustraties geleid voor we dit doorhadden. De interne simulator van dashcode draait ook geen PHP code wat ook een probleempuntje was. We zijn dus begonnen met Dashcode voor de gemakkelijke manier van werken maar zijn geëindigd met de windows variant waarin je zelf de html code moet schrijven (iWebKit, een variant op wat we van de assistenten kregen, IUI).

Wat we veel te laat gemerkt hebben is dat er toch enkele bugs zitten in de coverflow component die we gebruikt hebben. Deze coverflow is qua uiterlijk de beste die we tot nu toe gebruikt hebben (die van flex leek ook grafisch op de coverflow die we wouden realiseren maar hierbij hadden we geen voordeel van het 'vegen' aspect van de vinger op het touchscreen). Maar achteraf zagen we echter dat de array van figuren die we meegaven aan de component niet in de juiste volgorde getoond wordt. We zagen hier ook geen deterministisch gedrag in. Bij grote aantallen bleek ook dat niet alle resultaten getoond werden. Eerst dachten we dat het probleem lag bij de lange tijd voor het laden van de informatie aangezien deze nu door PHP in 1 keer wordt opgehaald. Nochtans geeft het PHP script alle resultaten correct door dus ligt dit probleem toch aan de coverflow component. Deze 2 bugs hebben we echter te laat opgemerkt om deze nog proberen op te lossen. Ze zijn echter niet cruciaal en laten onze applicatie nog steeds toe te werken en te tonen wat het kan.

Een alternatieve implementatie was om een echte iPhone applicatie te maken. We hebben dit even geprobeerd maar toen we zelfs het Hello-World project niet aan de praat kregen zijn we hier vanaf gestapt. We vinden het opvallend en teleurstellend dat ontwikkelen voor zo'n gebruiksvriendelijk platform als de iPhone OS, niet gebruiksvriendelijk blijkt te zijn. (Certificaten, Xcode)

5 Resultaat

Het eindelijke resultaat vinden we zelf niet helemaal geslaagd. De website werkt wel voor het grootste deel, maar de coverflowcomponent die we gebruiken bevat toch opvallende bugs die we niet opgelost krijgen. Het andere minpunt is de performantie: bij zoekopdrachten die veel resultaten teruggeven is de applicatie zeer traag.

6 Over iPhone

Zoals in puntje 3 al aangekaart werd vinden we dat de programmeeromgeving Dashcode niet de ideale omgeving is om niet voorgeprogrammeerde grafische effecten toe te voegen. De omgeving op zich heeft wel sterke punten want apple heeft ervoor gezorgd dat je heel gemakkelijk interfaces kunt maken die de look & feel hebben van iPhone apps.

Zoals al is aangehaald, is het ons ook opgevallen dat native applicaties ontwikkelen voor de iPhone veel moeilijker is dan gedacht. Het gebruiksgemak wat we als Mac en iPhone gebruikers gewend zijn, is bijna niet terug te vinden wanneer we willen ontwikkelen voor deze platformen. Zo werd het ons ook duidelijk waarom er een hele community ontstaan is om iPhones te unlocken en jailbreken: meer vrijheid voor de ontwikkelaar.

Een zwak punt van deze aanpak is dan weer de mindere ondersteuning van de mogelijkheden van de iPhone (zoals bvb opzoeken in de iTunes bibliotheek), iets waar native applicaties dan weer wel gebruik kunnen van maken.

De iPhone is naar onze mening het meeste geschikt voor toepassingen om technologie op afstand te bedienen en voor communicatie via sms, gsm, email,.. Een totale browserervaring kan de iPhone niet aanbieden; het scherm is te klein om informatie van de volledige website weer te geven. Maar

voor een snelle opzoeking (bvb wanneer vertrekt mijn trein) is de safaribrowser wel geschikt, vooral als de website een mobiele variant heeft.

7 Extra

Zie <http://eriksangels.wordpress.com> voor een video-opname die het gebruik van onze toepassing toelicht.

8 Besluit

We zijn niet helemaal tevreden over de aanpak van dit project. Ons doel was de Android-applicatie omvormen naar de iPhone. De layout was best in orde, alsook de functionaliteit. De coverflow zelf is niet zo eenvoudig te implementeren. We werden behoorlijk enthousiast toen we op het web een versie vonden die speciaal voor de iPhone aangepast werd. Deze zag er erg mooi uit met de voorbeeldprentjes en ook de 'flick'-gebaren maakten het geheel compleet. Het heeft ons echter zoveel werk gekost om deze component in onze app in te bouwen, en dat heeft ervoor gezorgd dat er geen tijd over was om na te denken over nieuwe functionaliteit.

Ook onze tijdsindeling was deze keer niet zo optimaal. Steven had de meeste uren aan Flex gespendeerd, en Pieter aan Android. Dit was onderling afgesproken in overeenstemming met alle groepsleden. Brecht nam graag deze iPhone applicatie voor zich, omdat hij ook de meeste PHP en Javascript ervaring had. We hadden echter meer als een team samen moeten werken om tot een beter resultaat te komen.

Het beste resultaat dat uit de ontwikkeling van deze applicatie is gekomen, is daarom ook de aparte server die de verschillende andere diensten aanspreekt en alle informatie verzameld. Het enige minpuntje aan deze server, is dat we er niet vroeger gebruik hebben van gemaakt: Android en flex hadden op deze manier maar 1 server moeten aanspreken en flex had geen crossdomain verzoeken moeten uitvoeren. Daarnaast had de ontwikkeling van beide projecten korter en eenvoudiger geweest en hadden we ons meer kunnen concentreren op extra functionaliteit. Hopelijk kunnen we bij het ontwikkelen van de multitouchapplicatie van deze server terug gebruik maken en zo meer tijd spenderen aan het gebruiken van de multitouch aspecten.

A Appendix

A.1 Tijdsbesteding

	Brecht Van der vekens	Pieter Van Geel	Steven Vercammen	Totaal
Storyboard	1	1	2	4
Ontwerp	5	2	1	8
Implementatie	31	1	2	34
Verslag	5	5	5	15
Sessies	15	15	15	45
Totaal	57	24	25	106