Verslag Multimedia: modelleren en programmeren

Het ontwikkelen van een mobiele applicatie in Android en iOS

Joris Schelfaut

G0Q55A - Multimedia: modelleren en programmeren
Prof. ir. E. Duval
2012-2013

Inhoudsopgave

Hoofdstuk 1

Introductie

Dit is het eindverslag voor de cursus 'Multimedia: modelleren en pogrammeren'. Het doel van deze cursus is om een mobiele applicatie te ontwerpen en implementeren in verschillende technologiën (iOS, Android, HTML5, ...) om uiteindelijk inzicht te verwerven in de sterktes en zwaktes van deze technologiën.

De applicatie is een Quantified Self App. Dit soort toepassing vraagt de gebruiker om invoer, en/of verzamelt deze automatisch afhankelijk van het type data, gedurende een zekere periode. Deze data kan dan gevisualiseerd worden om mogelijk gedragspatronen te onthullen[9].

De technologieën die worden besproken in dit verslag zijn iOS en Android. Alle code voor dit project is open source en kan gevonden en gedownloaded worden van github[1]. Naast dit verslag kan informatie over het project gevonden worden op de wordpress blog[2] en youtube vlog[3].

We bespreken eerst het idee achter de applicatie. Vervolgens kijken we naar een scenario en daarna naar het storyboard van de applicatie. Dan wordt de architectuur en het software ontwerp van de applicatie voor de verschillende technologieën bekeken. Hierna maken we een vergelijking tussen de verschillende technologieën. We eindigen met een besluit.

[1] `https://github.com/JorisSchelfaut/mumedev`
[2] `http://mumedev.wordpress.com/`
[3] `http://www.youtube.com/user/mumedev`

Hoofdstuk 2

Idee

De applicatie die we zullen ontwerpen en implementeren is een Quantified Self applicatie getiteld 'Moggle', een woordspeling op 'mood' en 'Toggl'. Het basisidee was om de werkuren die gepresteerd werden door de gebruiker manueel te laten ingeven, of indien mogelijk automatisch laten importeren via Toggle, en vervolgens zijn prestaties te laten evalueren volgens drie indicatoren: slecht, gemiddeld en goed. Ten slotte geeft de gebruiker zijn gemoedstoestand in. Het resultaat is dus een datamodel waarin de werkkwantiteit en -kwaliteit aan de gemoedstoestand gekoppeld worden.

Uitbreidingen hierop zouden kunnen zijn om ook het tijdstipaspect in rekening te brengen (e.g. dag van de week), werklocatie, achtergrondgeluiden tijdens het werk, de specifieke activiteit (studeren, programmeren, trainen, ...) en dergelijke meer. Het automatisch 'tracken' van zulke parameters is waarschijnlijk niet eenvoudig, aangezien het toestel niet noodzakelijk aanstaat tijdens het presteren van de uren zelf.

Hoofdstuk 3

Scenario

We beschrijven een kort scenario van een persoon die de applicatie gebruikt om na te gaan of er een correlatie zou kunnen bestaan tussen zijn geleverde werk en zijn gemoedstoestand.

Het is maandag. Een student heeft net een vijftal uren aan een verslag gewerkt. Hij houdt het voor bekeken voor die dag en vult zijn gepresteerde uren in op de Moggle applicatie, en ook de rating van zijn werk en zijn gemoedstoestand. Hij is tevreden over zijn werk en geeft zijn gemoedstoestand een rating van 9 op 10.

Woensdag, donderdag en zaterdag speelt zich eenzelfde scenario af met respectievelijk 2, 5 en 7 uren gewerkt, 'slecht', 'goed' en 'matig' als werkkwaliteit en 3, 7 en 7 als ratings voor zijn gemoedstoestand.

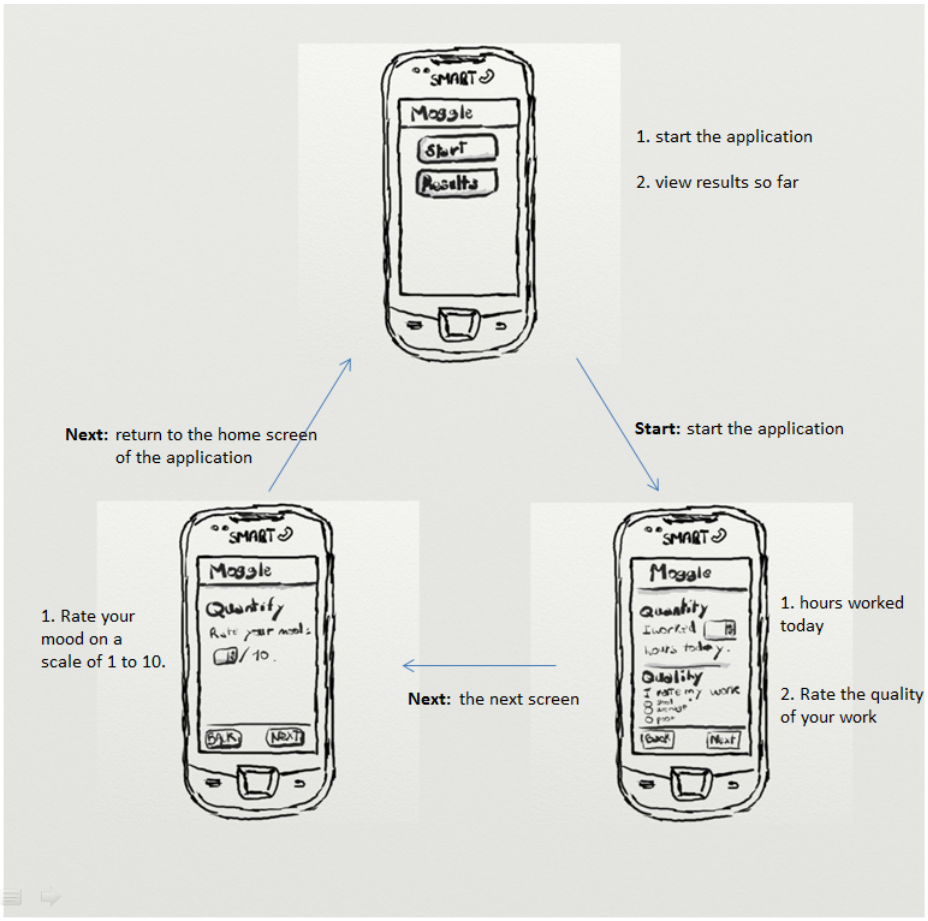
Wanneer hij zijn resultaten bekijkt, merkt hij dat slecht en weinig werk leveren doorgaans correleert met een slechte gemoedstoestand, terwijl goed werk leveren vooral overeenstemt met een goede gemoedstoestand. Veel en goed werk leveren lijkt echter niet altijd te correleren met een goede gemoedstoestand.

Hoofdstuk 4

Storyboard

Het volledige storyboard is te zien op figuur 4.1. Het storyboard bestaat uit drie fases:

1. In de eerste stap heeft de gebruiker de keuze de resultaten te bekijken of om een nieuwe data in te geven.
2. De optie 'start' brengt de gebruiker naar het eerste invoerscherm. De gebruiker kan nu het aantal uren ingeven dat hij/zij heeft gepresteerd en een rating geven betreffende de kwaliteit van deze prestatie.
3. Via de optie 'next' komt gebruiker in het laatste scherm terecht. Hier geeft de gebruiker een kwantificatie van zijn gemoedstoestand in op een schaal van 1 tot 10.



Figuur 4.1: Moggle storyboard

Hoofdstuk 5

Architectuur

5.1 Verzamelen van data

Om relatief snel een applicatie te kunnen implementeren, werd in de applicatie eerst manuele datainvoer voorzien.

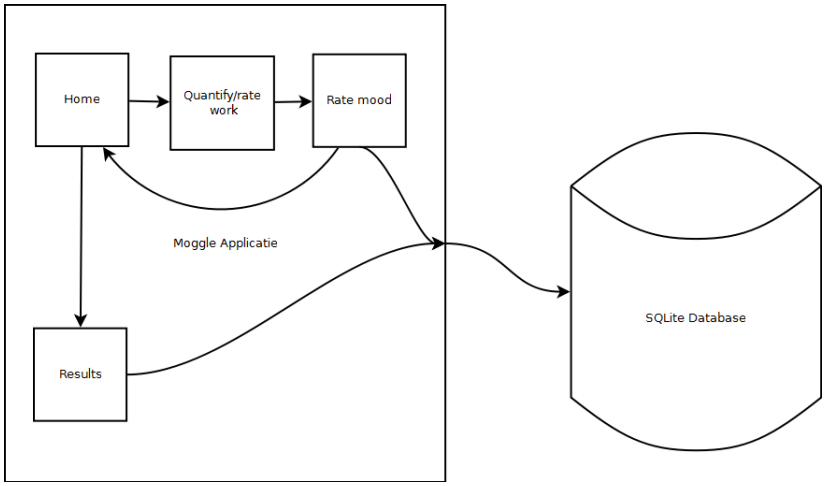
Het aanvankelijke idee om gebruik te maken van de Toggle API[1] is uiteindelijk niet meer geïmplementeerd. Indien de gebruik zou gemaakt worden van de Toggl API zou er natuurlijk een internetconnectie moeten opgezet worden. De gebruiker zou zich moeten authenticeren, waarna logingegevens in lokaal zouden kunnen bijgehouden worden en dergelijke.

5.2 Opslag van data

De applicatie maakt gebruik van lokale opslag om gegevens te persisteren. De records bevatten relatief beperkte, numerieke informatie. De huidige smartphones hebben gemakkelijk meer dan een gigabyte aan intern geheugen[15], dus de gebruiker zou al duizenden records moeten opslaan vooraleer een significant deel van de opslagcapaciteit zou gebruikt worden. Daarnaast heeft lokale opslag het voordeel dat het een hogere beschikbaarheid aanbiedt dan opslag op een externe server. Ten slotte is externe, online opslag geen vereiste om resultaten te verkrijgen die significant zijn. Uiteraard kan het vergelijken met andere gebruikers, gemiddeldes en uitschieters een interessante uitbreiding zijn bij de data-analyse.

De interne architectuur van de applicatie is afgebeeld op figuur 5.1.

[1] `https://www.toggl.com/public/api`



Figuur 5.1: Interne architectuur van de Moggle applicatie

Hoofdstuk 6

Softwareontwerp

6.1 Datamodel

Beide applicaties maken gebruik van hetzelfde datamodel. Het is een eenvoudige tabel in een SQLite database zoals te zien in tabel 6.1. SQLite is relatief eenvoudig te integreren in beide technologieën[1]. De SQL-code om een database en tabellen aan te maken en te verwijderen is uiteraard hetzelfde.

Tabel 6.1: QDat SQLite Table

QDat	
id (pk)	INTEGER
date	TIMESTAMP
work_quantity	REAL[0, 24]
work_quality	INTEGER[0..2]
mood_quantity	REAL[0, 10]

Een andere mogelijkheid, specifiek voor iOS, is het gebruik van Core Data. Met dit framework kan dataopslag gebeuren aan de hand van een object-georiënteerde abstractielaag[20]. Voor deze relatief kleine applicatie volstaat een SQLite database echter.

6.2 Structuur

De projectstructuur voor de twee projecten verschilt weinig in grote lijnen. Het aantal schermen is hetzelfde, net als het aantal controllers. Het model voor de Android-applicatie is iets uitgebreider dan in de iOS-versie, maar dit is eerder een gevolg van de manier waarop de database werd geïmplementeerd, dan de constraints van de technologie zelf.

[1] SQLite tutorial voor iOS: [19], SQLite tutorial voor Android: [2]

6.2.1 Android

De Android-app is opgebouwd uit vier klassen die overerven van de *Activity*-klasse, met bijbehorende XML-files voor de views zoals te zien in tabel 6.2. *HomeActivity* is het startpunt van de applicatie. Het scherm toont twee knoppen: één om te navigeren naar de *QWorkActivity* pagina en één om de resultatenpagina, *DataVisActivity*, te laden. De view van *QWorkActivity* bevat enkele labels, een invoerveld, een radioboxgroep en een knop om naar *QMoodActivity* te navigeren. In de view van *QMoodActivity* staat er een label, een slider en een knop om naar *HomeActivity* terug te keren. Hierbij wordt eveneens de data gepersisteerd in de SQLite databank.

Tabel 6.2: Android Activity klassen met bijbehorende XML-files

Activity	**XML**
HomeActivity.java	*activity_home*.xml
QWorkActivity.java	*activity_qwork*.xml
QMoodActivity.java	*activity_qmood*.xml
DataVisActivity.java	*activity_data_vis*.xml

Om de connectie met de database te maken, wordt er gebruik gemaakt van een implementatie van de SQLiteOpenHelper-klasse; de klasse *DatabaseHandler* in dit project. De klasse *CRUD<E>* is een templateklasse voor het beheren van een database tabel. Deze heeft de typische methodes *createTable*, *dropTable*, *insert*, *delete*, *update* en *select*.

De modelklasse *QDat* implementeert de *Parcelable* interface. Dit zorgt ervoor dat we op een efficiënte manier objecten tussen *Activity*-klassen kunnen doorgeven.

Het volledige klassediagram voor de Android-applicatie is te zien op figuur 6.1.

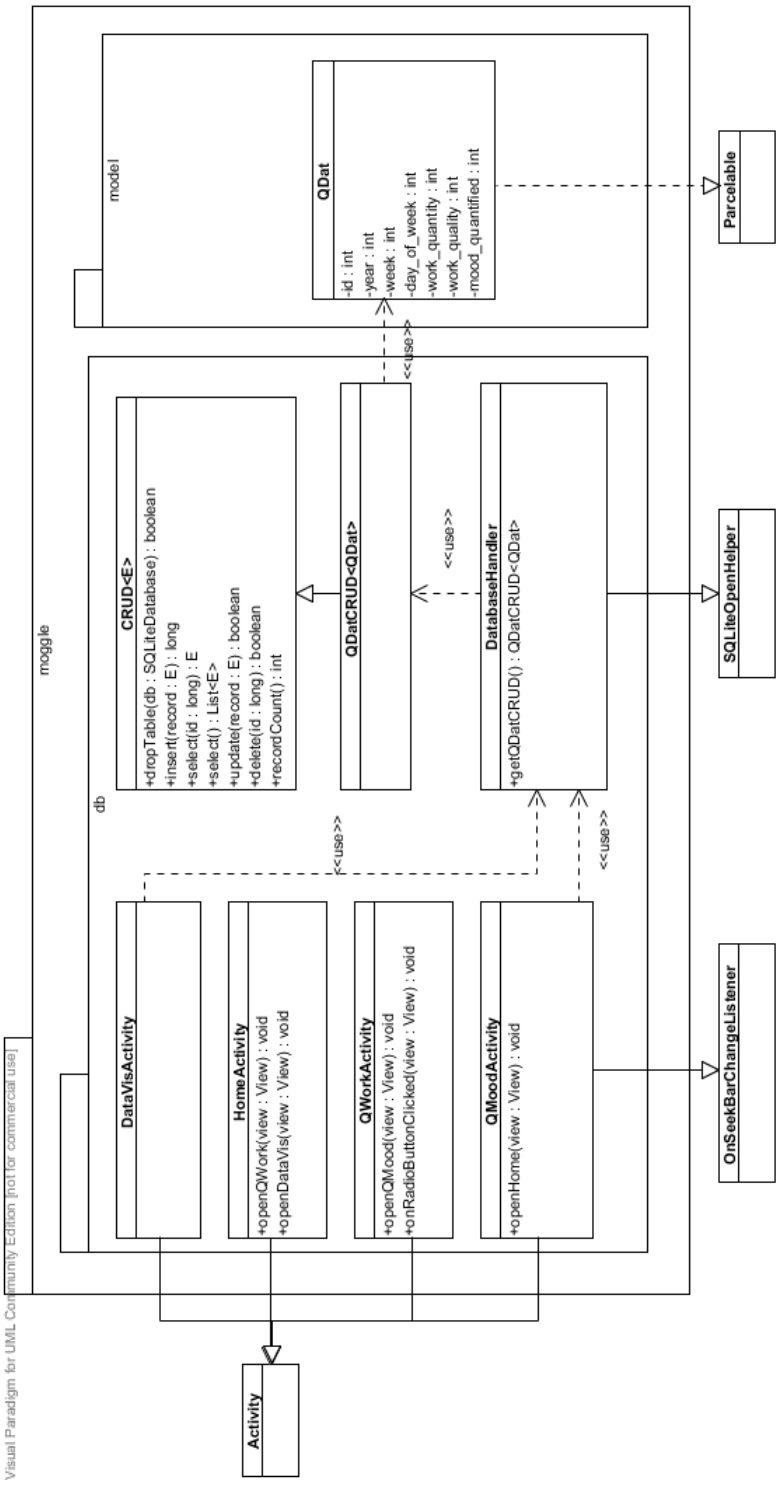
6.2.2 iOS

Het ontwerpen van de iOS-app is grotendeels gebaseerd op het storyboard. Er zijn vier views met bijbehorende *UIViewController*-implementaties: *ViewController*, *QWorkController*, *QMoodController* en *ResultsController*.

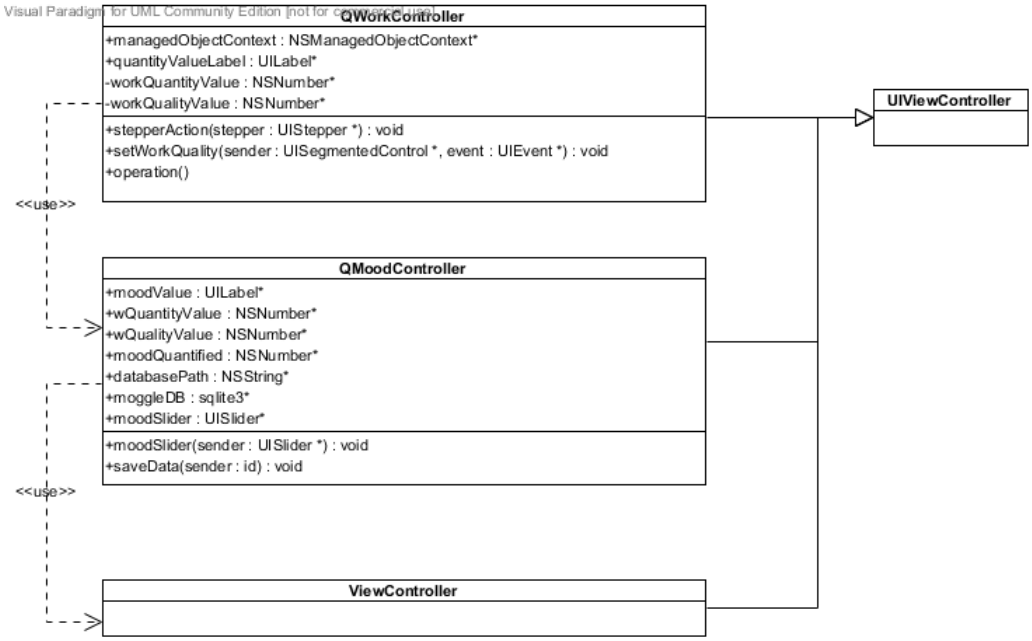
De UI-elementen komen grotendeels overeen met die van de Android-versie. Behalve in de view van de *QWorkController*: de stepper biedt een mooie voorgedefinieerde functionaliteit om op een eenvoudige manier een natuurlijk getal in te geven door de huidige waarde met stappen van één te verhogen of te verlagen.

Een interessante eigenschap van iOS is het doorgeven van argumenten tussen controllers van opeenvolgende views. Het volstaat hierbij in de methode *prepareForSegue* de pointer naar het object expliciet te initialiseren in de targetcontroller.

Het klassediagram voor de iOS-applicatie is te zien op figuur 6.2.



Figuur 6.1: Android Class Diagram van de Moggle applicatie



Figuur 6.2: iOS Class Diagram van de Moggle applicatie

6.3 Uitbreidingen

Mogelijke uitbreidingen aan de huidige applicaties zouden bijvoorbeeld de volgende kunnen zijn:

- dataopslag gecentraliseerd op een server;
- integratie met de Toggl API;
- implementatie van verschillende data visualisaties;
- een notificatiesysteem uitbouwen op basis van gebruikersvoorkeuren;
- combinatie met Google Calendar: afhankelijk van de weekplanning notificaties sturen;
- ...

Al bij al is het nog maar een zeer klein project, dus echt grote ontwerpbeslissingen hoefden nog niet genomen te worden. Door de applicatie enkel voor iPhone te ontwikkelen en niet voor bijvoorbeeld de iPad, worden bovendien een deel van de mogelijke problemen uitgesloten, gerelateerd aan de layout van de schermen en visualisaties.

Hoofdstuk 7

Vergelijking technologieën

7.1 Properties

7.1.1 Ondersteuning

Development Environment en simulator

In termen van voorwaarden om van start te geraken in iOS en Android verschillen beide technologieën. Development in iOS is relatief gelimiteerd, Apple's IDE XCode is immers enkel beschikbaar op Mac OS platformen[8] en applicaties zoals Macincloud[1] waarmee een Mac services vanuit de browser kunnen worden gebruikt[12], lijken nog niet echt op punt te staan, afgaande op eigen ervaring.

Daarentegen is Android development relatief toegankelijk. Dankzij de Java virtuele machine kan Eclipse op zowat elk platform geïnstalleerd worden samen met de Android Eclipse plugin[2].

De Android simulator van Eclipse komt zeer log over, al zijn er kennelijk wel mogelijkheden om de performance te verbeteren[18]. De simulator voor iOS is wel zeer performant. Beide hebben het nadeel dat ze beperkt zijn voor het testen van bepaalde userinput; denk aan touch events, gebruik van acceleratiemeter en dergelijke[8].

Leerproces

Voor zowel Android als iOS ligt een object-georiënteerde taal aan de basis, respectievelijk Java en Objective-C. Beide systemen beschikken over een uitgebreide documentatie.

In [7] beweert men dat je zowat alles omtrent iOS development kunt terugvinden op Stackoverflow[3], in tegenstelling tot voor Android. Op basis van het aantal documenten getagged met 'iOS' en 'Android' werden respectievelijk $117,684$ en $276,882$ resultaten teruggevonden[17],[16]. Dit lijkt voorgaand statement enigszins tegen te spreken, al zegt kwantiteit niet alles uiteraard.

Voor beide technologieën bestaan er echter vele websites en blogs met goede informatie. Een opmerking die hierbij gemaakt moet worden, is dat hieronder ook veel verouderde informatie is.

[1] `http://www.macincloud.com/`
[2] `http://developer.android.com/sdk/installing/installing-adt.html`
[3] `http://stackoverflow.com/`

Afhankelijk van de achtergrond van de programmeur zal programmeren in een van beide technologieën efficiënter verlopen. Al bij al is ondersteuning van het leerproces vrij gelijklopend[8]. Hoewel er veel parallellen zijn tussen iOS en Android, vergt het wel degelijk wat tijd om bijvoorbeeld iOS aan te leren, zelfs al ben je vertrouwd met Android.

Distributie

Een van de verschillen tussen beide technologieën is de manier waarop ze omgaan met het publiceren en distribueren van applicaties. Apple is vrij strikt in het al dan niet toelaten van applicaties op de App Store. Dit verhindert echter niet dat er soms malafide applicaties in de app store terechtkomen[13]. Het inschrijven in een developerprogramma kost bovendien een redelijke som, tenzij je je via een universiteitsprogramma kan inschrijven[4].

Het deployen van een applicatie op een echt toestel, is in vergelijking met het deployen van applicaties op Android ook niet evident, ook al vergt dit slechts een eenmalige configuratie per applicatie[11].

7.1.2 Implementatie

Model-View-Controller

Zoals vele moderne systemen hanteren zowel iOS als Android de Model-View-Controller (MVC) filosofie. MVC tracht de applicatielogica, de schermrepresentatie en de manier waarop het model op gebruikersinput reageert te ontkoppelen[10]. Een interessant punt van vergelijking is hoe dit wordt vertaald in praktijk.

In Android kan het model gezien worden als een verzameling Java-klasses. De views komen overeen met XML-files en staan los van het model. De controllers worden geïmplementeerd als Android Activity en Service subklassen. Deze voorzien de achterliggende functionaliteit van de controls in de userinterface en spreken het model aan. Het blijkt echter dat de loose coupling bekomen tussen controller en view in Activity-klassen niet altijd evident is[14].

De MVC structuur in iOS is gelijkaardig. Storyboard- en/of XIB-files definiëren de view, ViewController-subklassen vormen de controllers in combinatie met onder meer delegates[8]. De communicatie vindt plaats door middel van outlets. De controller plaatst een 'target' op zichzelf en de view stuurt 'actions' naar de targetcontroller[21]. Het model kan geïmplementeerd worden als Objective-C-klassen. Het iOS Core Data framework biedt een goede ondersteuning van model-gerelateerde code[5].

In beide gevallen kent de view de controller, en in beide gevallen wordt er ook een referentie naar de view in de controller bijgehouden[6],[1]. De 'controllers' zoals ze in deze tekst zijn geclassificeerd, ondersteunen dus geen 'pure' loose coupling. In het ideale geval zou dezelfde controller meerdere views zou kunnen ondersteunen.

User interface design

Het design van de layout van een user-interface in Android gebeurt aan de hand van XML files[3]. In Eclipse kunnen elementen zowel via een drag-and-drop-systeem als via een teksteditor worden toegevoegd[8].

Een van de features van XCode zijn de storyboards. Deze vormen een goed visueel overzicht van de applicatie in termen van de sequenties en layouts van schermen. Hierin kunnen elementen aan elk scherm worden toegevoegd via een drag-and-drop-systeem, maar in tegenstelling tot de Android Eclipse plugin is het editeren van de achterliggende XML-structuur minder transparant.

7.2 Overzicht

Tabel 7.1 geeft een overzicht van de besproken properties van iOS en Android in sectie 7.1.

Tabel 7.1: Vergelijking tussen iOS en Android development

		Android	**iOS**
Ondersteuning	IDE	*Compatibel met de meeste platformen.*	*Enkel voor Mac OS, snelle simulator.*
	Leerproces	*Veel online bronnen en fora.*	*Idem.*
	Distributie	*Goedkoop, relatief eenvoudig.*	*Relatief duur, vergt wat configuratie.*
Implementatie	MVC	*Ondersteund, al is de loose coupling tussen controller en view niet altijd even transparant.*	*Ondersteund.*
	GUI design	*D&D, XML-editor*	*D&D, minder transparant, uniformiteit wordt meer afgedwongen*

Hoofdstuk 8

Besluit

Het doel van de cursus 'Multimedia: modelleren en programmeren' bestaat erin een mobiele applicatie te ontwerpen en implementeren in verschillende technologieën (iOS, Android, HTML5, ...) om uiteindelijk inzicht te verwerven in de sterktes en zwaktes van deze technologieën. In dit verslag werden achtereenvolgens een scenario, het storyboard, de applicatiearchitectuur het het softwareontwerp voor zowel de Android- als de iOS-versie van de Moggle applicatie besproken. Ten slotte werden properties van beide technologieën overlopen en vergeleken.

8.1 Vergelijking technologieën

Hoewel de ondersteuning voor zowel Android als iOS vrij gelijklopend is, is het programmeren in iOS minder toegankelijk. Zowel op vlak van kosten, op vlak van distributie en op vlak van platformen scoort iOS minder goed. Men beoogt meer uniformiteit in applicaties en veiligere software in Apple App Store.

Het model-view-controller patroon wordt ondersteund in beide technologieën. Kennelijk is er kritiek op de implementatie door Android, maar al bij al volgen zowel iOS als Android een vrij gelijkaardige strategie. Geen van beide ondersteunt pure loose coupling. Een voordeel van iOS is het Core Data framework, dat een goede architectuur vormt voor het model.

Het maken van layouts voor views in Android is iets transparanter dan voor iOS - al vindt ik persoonlijk het Wysiwyg drag-and-drop systeem met storyboards beter dan de uitwerking in Android.

Het aanleren van Android was voor mij redelijk intuïtief, aangezien ik reeds vertrouwd was met Java en XML. Dit hielp uiteraard ook bij het verwerken van tutorials en documentatie. iOS vond ik minder transparant. Het feit dat ik pas vrij laat met de meer gestructureerde aanpak van de Stanford Slides[1] ben begonnen, heeft vooral voor verwarring gezorgd bij de aanvang van de cursus. Mede door het feit dat er veel verouderde informatie online staat, waarin nog geen gebruik gemaakt wordt van storyboards.

Op basis van de aangehaalde argumenten zou kunnen geconcludeerd worden dat op vlak van toegankelijkheid en ondersteuning Android beter scoort dan iOS. Op vlak van architectuur lijkt iOS de bovenhand te hebben over Android.

[1] `http://www.stanford.edu/class/cs193p/cgi-bin/drupal/downloads-2011-fall`

8.2 Cursus

De cursus is zeker relevant voor een opleiding in softwareontwikkeling en multimedia. De aangeboden topics zijn interessant en ook leuk om mee te werken - ondanks de vele frustraties die gepaard gaan met software development en het leren van een nieuwe programmeertaal.

De cursus is nog relatief nieuw en staat nog niet helemaal op punt, heb ik de indruk. Naar mijn mening mist de cursus wat structuur. Enkele bedenkingen:

- doelstellingen mogen nog concreter geformuleerd worden bij aanvang van de cursus: zowel globale doelstellingen als concrete doelstellingen betreffende implementatie;
- een beter overzicht geven van websites, video's, boeken en cursussen die goed bleken te zijn;
- best practices en richtlijnen voor het schrijven van blogposts: sommige posts missen structuur en lijken het globale verhaal van de blog te missen;
- anders kaderen van presentaties door externen in het geheel van de lessen: omwille van het estaffettesysteem was de timing van sommige presentaties misschien wat ongelukkig. Er leek ook niet echt een logica te zitten in de opeenvolging van de presentaties;

Bibliografie

[1] Android. Activity | android developers. URL: http://developer.android.com/reference/android/app/Activity.html, 2013. [Online; accessed 16-January-2013].

[2] Android. android.database.sqlite | android developers. URL: http://developer.android.com/reference/android/database/sqlite/package-summary.html, 2013. [Online; accessed 15-January-2013].

[3] Android. Ui overview | android developers. URL: http://developer.android.com/guide/topics/ui/overview.html, 2013. [Online; accessed 15-January-2013].

[4] Apple. Choosing an ios developer program - apple developer. URL: https://developer.apple.com/programs/start/ios/, 2013. [Online; accessed 15-January-2013].

[5] Apple. Core data programming guide: Technology overview. URL: http://developer.apple.com/library/mac/#documentation/cocoa/Conceptual/CoreData/Articles/cdTechnologyOverview.html#//apple_ref/doc/uid/TP40009296-SW1, 2013. [Online; accessed 15-January-2013].

[6] Apple. Uiviewcontroller class reference. URL: http://developer.apple.com/library/ios/#documentation/uikit/reference/UIViewController_Class/Reference/Reference.html, 2013. [Online; accessed 16-January-2013].

[7] M. Frederick. Android vs ios development: A developer's perspective. URL: http://viamike.com/blog/post/android-vs-ios-development-a-developer-s-perspective, 2012. [Online; accessed 15-January-2013].

[8] M. H. Goadrich and M. P. Rogers. Smart smartphone development: ios versus android. *SIGCSE '11 Proceedings of the 42nd ACM technical symposium on Computer science education*, pages 607–612, 2011.

[9] S. Govaerts. Quantified self in the multimedia course. URL: http://www.slideshare.net/stengovaerts/quantified-self-in-the-multimedia-course, 2012. [Online; accessed 16-January-2013].

[10] R. Johnson, E. Gamma, J. Vlissides, and R. Helm. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Pearson Education Corporate Sales Division, 201 W. 103rd Street, Indianapolis, 1995.

[11] W.-M. Lee. Deploying iphone apps to real devices | mobiforge. URL: `http://mobiforge.com/developing/story/deploying-iphone-apps-real-devices`, 2009. [Online; accessed 15-January-2013].

[12] Macincloud. Macincloud - rent a mac in the cloud! - mac in cloud. URL: `http://www.macincloud.com/`, 2013. [Online; accessed 15-January-2013].

[13] R. Meeus. Smartphones krijgen de ziekte. *Knack*, 2012.

[14] J. Musselwhite. Android architecture: Part 1, intro - android developer. URL: `http://www.therealjoshua.com/2011/11/android-architecture-part-1-intro/`, 2011. [Online; accessed 15-January-2013].

[15] T. Schiesser. Guide to smartphone hardware (3/7): Memory and storage. URL: `http://www.neowin.net/news/guide-to-smartphone-hardware-37-memory-and-storage`, 2012. [Online; accessed 14-January-2013].

[16] Stackoverflow. Newest 'android' questions - stack overflow. URL: `http://stackoverflow.com/questions/tagged/android`, 2013. [Online; accessed 15-January-2013].

[17] Stackoverflow. Newest 'ios' questions - stack overflow. URL: `http://stackoverflow.com/questions/tagged/ios`, 2013. [Online; accessed 15-January-2013].

[18] Stackoverflow. performance - slow android emulator - stack overflow. URL: `http://stackoverflow.com/questions/1554099/slow-android-emulator`, 2013. [Online; accessed 15-January-2013].

[19] Techotopia. An example sqlite based ios 6 iphone application - techotopia. URL: `http://www.techotopia.com/index.php/An_Example_SQLite_based_iOS_6_iPhone_Application`, 2012. [Online; accessed 15-January-2013].

[20] Techotopia. An ios 6 iphone core data tutorial - techotopia. URL: `http://www.techotopia.com/index.php/An_iOS_6_iPhone_Core_Data_Tutorial`, 2012. [Online; accessed 15-January-2013].

[21] S. University. Lecture 1: Mvc and intro to objective-c. URL: `http://www.stanford.edu/class/cs193p/cgi-bin/drupal/system/files/lectures/Lecture%201_1.pdf`, 2011. [Online; accessed 15-January-2013].

Lijst van figuren

Lijst van tabellen

Bijlage A

Tijdsbesteding

Tabel A.1: Tijdsbestding voor het vak 'Multimedia : modelleren en programmeren'

Categorie	**Tijdsbesteding** (in uren)
Brainstorm, storyboard, scenario	5
Face-to-face meetings	5
Listening (lectures)	33
Programming - Android	50
Programming - HTML5	1
Programming - iOS	42
Reading	12
Reporting (includes blogging, reports, presentations)	23
Social media activity (twitter, commenting on other blogs)	10
Totaal	180

Tabel A.2: Gebruik van sociale media voor het vak 'Multimedia : modelleren en programmeren'

Categorie	**Aantal**
Blogpost (followapp.wordpress.com)	13
Blogpost (mumedev.wordpress.com)	9
Comments (wordpress.com)	8
Video (youtube.com/mumedev)	14
Tweets (twitter.com/JorisSchelfaut)	37