Estat de l'art en el desenvolupament d'aplicacions mòbils multiplataforma

Memòria PFC

Fernando Navajas Jiménez
Enginyeria en Informàtica

Marc Domingo Prieto
Consultor

Decembre de 2011

UOC
Universitat Oberta de Catalunya
www.uoc.edu

A tots els companys de viatge per estar sempre al meu costat…

Resum

La nostra societat està vivint una evolució tecnològica important en la actualitat. La informació i la comunicació tal i com l'enteníem fins ara està canviant totalment. La forma d'interactuar entre nosaltres i de veure el mon que ens envolta no para d'evolucionar i en gran mesura és gracies a les noves tecnologies mòbils.

Fa molts anys que preveiem que en un futur no molt llunyà els nostres ordinadors de sobretaula podrien estar a l'abast de la nostra mà. Avui en dia, això és tota una realitat a la nostra societat. La proliferació dels telèfons mòbils en els últims anys i l'evolució d'aquestos, cap a sistemes cada vegada més potents, ha fet realitat aquella visió de futur.

És indiscutible que la competència ens fa evolucionar i ens els últims anys, sobretot a partir de l'aparició del primer iPhone d'Apple, aquesta és més agressiva que mai. No paren d'aparèixer noves patents cada dia i les denuncies per plagi estan més de moda que mai i això fa que l'avanç vagi a passos agegantats.

En aquest projecte volem esbrinar en quin moment ens trobem i quines tecnologies tenim actualment al nostre abast. És molt importat conèixer el mercat a l'hora de desenvolupar noves aplicacions i veure quines són les possibilitats que ens ofereixen els nous dispositius.

Complementàriament farem un estudi comparatiu entre tecnologies multiplataforma. Aquestes venen amb molta força per lluitar cara a cara amb les tecnologies natives que estaven establertes fins ara. Això no vol dir que siguin dos tipus de tecnologia oposades, però és cert que amb la gran proliferació de noves plataformes que hi ha en la actualitat, és fa patent la necessitat de tecnologies que ens donin l'opció de desenvolupar aplicacions que es puguin executar en el màxim de plataformes possibles per tal d'arribar al major número de públic.

Índex general

Llistat d'acrònims

- API - *Application Programming Interface*
- APP - *Application*
- ARM - *Advanced Risc Machine*
- CDC – *Connected Device Configuration*
- CLCD – *Connected Limited Device Configuration*
- CSS – *Cascading Style Sheets*
- DOM – *Document Object Model*
- GPS – *Global Positioning System*
- HTML - *HyperText Markup Language*
- IDE - *Integrated Development Environment*
- IMP – *Information Module Profile*
- J2ME – *Java 2 Micro Edition*
- MIDP - *Mobile Information Device Profile*
- MSDN - *Microsoft Developer Network*
- MVC – *Model-View-Controller*
- MXML - *Macromedia eXtensible Markup Language*
- OS – *Operating System*
- PC – *Personal Computer*
- PDA - *Personal Digital Assistant*
- RIM - *Research in Motion*
- SDK – *Software Development Kit*
- SMS - *Short Message Service*
- SO – *Sistema Operativo*
- WAP – *Wireless Application Protocol*
- WM – *Windows Mobile*

Agraïments

Vull agrair de tot cor el recolzament per part de la meva parella i dels meus companys. També al meu tutor per comprendre les meves necessitats en tot moment per la realització d'aquest projecte, ha estat un camí llarg i segurament no s'acabarà aquí.

No vull oblidar-me de tots els companys que he conegut durant els meus estudis i tots els consultors que m'han donat el seu ajut en tot moment.

Capítol 1

Introducció

És un fet evident que els dispositius mòbils s'han multiplicat exponencialment en el últims anys. En la actualitat formen part de la nostra vida i sembla gairebé impensable com podíem viure sense ells fins ara.

En aquest projecte volem veure en quin moment tecnològic ens trobem actualment i cap a on ens dirigim. Farem un repàs dels plataformes mòbils i de les tecnologies de desenvolupament que podem trobar al mercat.

En una segona part ens centrarem en un tipus de tecnologia de desenvolupament que ve amb molta força, sobretot gracies a la proliferació de nous dispositius, anomenada multiplataforma.

Es realitzarà un estudi comparatiu entre algunes d'aquestes tecnologies i finalment es compararan amb les tecnologies de desenvolupament natives de cada plataforma.

Tot aquest estudi comparatiu es realitzarà des d'un punt de vista teòric i posteriorment, gracies a alguns exemples, des d'un punt de vista pràctic.

1.1. Objectius del treball

L'objectiu d'aquest projecte és aprofundir en les noves tecnologies que permeten desenvolupar aplicacions multiplataforma per a dispositius mòbils i fer un estudi comparatiu entre algunes d'elles. També es realitza un estudi comparatiu entre les aplicacions generades amb aquestes tecnologies multiplataforma i les generades amb codi natiu del propi sistema.

El projecte es divideix en dos parts, la primera dona una visió general de moltes de les tecnologies disponibles però per tal de centrar els nostres estudis, amb la finalitat de fer un estudi comparatiu, ens fixem en tres tecnologies. Les tecnologies de desenvolupament amb les que treballem són **PhoneGap**, **Titanium**

Appcelerator i **Adobe Air Mobile**. Les dos primeres tenen molta potencia, són gratuïtes i estan basades en el desenvolupament web i la tercera té llicència comercial d'una gran empresa i treballa amb un codi propietari.

Una vegada analitzades les diferents tecnologies, passem a la segona part on farem la instal·lació de tot el entorn de desenvolupament i la posterior realització d'un projecte base i d'una aplicació molt simple. També provarem algunes aplicacions de codi lliure disponibles a Internet.

S'han intentat portar a la pràctica totes aquelles diferencies que van aparèixer al apartat teòric i sempre intentant donar un punt de vista des de totes les plataformes possibles.

1.2. Estudi descriptiu

El primer pas en aquest projecte és realitzar un estudi de l'estat de l'art de les tecnologies mòbils i del desenvolupament d'aplicacions per a elles. Complementàriament farem una repassada a l'evolució fins al moment actual i una visió de futur per veure cap a on ens poden portat.

Repassarem els nous dispositius que tenim al nostre abast i quines són les tendències del mercat. El món de les aplicacions també a evolucionat moltísim i ara la tendència general és distribuir-les mitjançant algun tipus de mercat virtual i sempre en format electrònic sense cap suport físic.

Donarem una pinzellada als principals sistemes operatius i tecnologies de desenvolupament per a cadascun d'ells.

Finalment, ens centrarem en les tecnologies multiplataforma on veurem les ultimes tendències en programació per dispositius mòbil i on ens centrarem en les tres tecnologies que utilitzarem per el nostre estudi comparatiu, que com ja hem dit abans són **PhoneGap**, **Titanium Appcelerator** i **Adobe Air Mobile.**

1.3. Estudi comparatiu

Una vegada realitzat l'estudi descriptiu passarem a realitzar l'estudi comparatiu. Per tal de realitzar les comparacions entre tecnologies s'han seleccionat els 10 punts a valorar que indiquem a continuació:

1. Llenguatges de programació
2. Maquetació i disseny d'aplicacions
3. Entorns de desenvolupament (IDE)
4. Components necessaris per la programació
5. Rendiment de les aplicacions
6. Fiabilitat de les eines
7. Documentació disponible de la tecnologia
8. Seguiment de la comunitat
9. Distribució i venda de aplicacions
10. Preu de desenvolupament i distribució

Amb tota aquesta informació podem donar un resultat numèric a cadascuna de les tecnologies analitzades i per tant realitzar una valoració teòrica i pràctica de cadascuna de elles.

1.4. Desenvolupament

Hi ha diferents requeriments segons la plataforma en la que treballem però principalment necessitem un equip per desenvolupar l'aplicació i un altre per provar-la. Per poder desenvolupar l'aplicació la majoria de les plataformes disposen d'un *IDE* propi o utilitzen un genèric que funciona tant en Windows com en Linux, això si, per poder desenvolupar per iOS és necessari, en la majoria dels casos, disposar d'un equip d'Apple amb Mac OS X o d'una màquina virtual emulant Mac OS X.

Es podrà desenvolupar les aplicacions en les tres tecnologies sota Windows, però la única que és totalment independent és *Adobe Air Mobile*, les altres dos necessiten Mac OS X i XCode per poder empaquetar les aplicacions per iOS.

Nota: No disposo actualment d'un equip d'Apple així que si necessito programar o fer córrer un programa en Mac OS X ho faré des de Windows emulant Mac OS X amb l'eina **VirtualBox**.

A part de l'*IDE* és necessari instal·lar un *SDK* per cadascuna de les plataformes mòbils.

Proves

Encara que actualment hi ha molts sistemes operatius per a dispositius mòbils, ens centrarem en els que actualment tenen més quota de mercat. Entre ells **Android** de Google, **iOS** d'Apple, **Windows Mobile** de Microsoft i **BlackBerry** de RIM.

Un cop desenvolupada l'aplicació tenim dos mètodes per tal de provar-la, amb un emulador o en un dispositiu físic. Gairebé tots els *SDK* porten un emulador per poder provar les nostres aplicacions.

Per provar realment les aplicacions necessitem disposar d'un dispositiu mòbil amb el sistema operatiu corresponent. Android i Windows Mobile treballem amb moltes marques que fabriquen dispositius (HTC, Samsung, LG, ...), iOS nomes amb alguns aparells de Apple com són iPhone, iPad o iPod i finalment Blackberry als seus propis dispositius.

Hi ha també casos en els que s'ha de pagar una llicencia com a desenvolupador per tal de poder empaquetar les aplicacions i poder instal·lar-les a un dispositiu físic.

Nota: Només disposo d'un dispositiu Android amb la versió 2.1, així que la majoria de proves les realitzaré mitjançant els emuladors disponibles i intentaré realitzar les proves físicament amb el telèfon Android.

Programari per cada plataforma

Phone Gap: Programarem en Javascript amb l'ajuda de *HTML5* i *CSS 3*, per això fem us d'Eclipse i de *frameworks* externs com Sencha Touch o JQuery Mobile. En el cas d'iOS necessitarem Xcode i en el cas de Android i BlackBerry necessitarem els seus *SDK*.

Titanium Accelerator: Programarem en Javascript i aquest porta un *IDE* propi basat en Eclipse anomenat *Titanium Studio.*

Adobe Air Mobile: Programarem en ActionScript 3.0 i aquest disposa del seu *IDE* anomenat Flash Builder. és de pagament però utilitzarem una versió *Trial*. També es pot utilitzar el *SDK* de Flex.

1.5 Detall d'objectius

S'han dividit els objectius en petites fites per tal de distribuir-les correctament dintre del calendari establert de presentacions. La idea és anar recopilant informació per després plasmar-ho tot a la memòria que es definirà la setmana abans de cadascuna de les entregues:

1. Recopilar informació sobre sistemes operatius mòbils.
2. Recopilar informació sobre plataformes de desenvolupament per a dispositius mòbils.
3. Anàlisi exhaustiu de les tres tecnologies seleccionades.
4. Comparació teòrica de les tecnologies seleccionades.
5. Instal·lació dels kits per programar en les tres tecnologies seleccionades.
6. Programació de exemple en les diferents plataformes.
7. Programació de aplicació per Android amb tecnologies multiplataforma.
8. Comparació practica de les tres tecnologies.
9. Instal·lació dels kits de programació natius per Android.
10. Programació de aplicació per Android amb codi natiu.

11. Comparació practica de les tecnologies multiplataforma VS tecnologies natives.

A continuació es marcaran tots els objectius dintre d'un calendari remarcant les dades d'entrega:

1.6 Calendari d'objectius

	Setmana	**Activitat**	**Memòria**
1	19-set/25-set		
2	26-set/02-oct	Recopilació informació + Decisió PFC (1,2)	
3	03-oct/09-oct	Lliurament Pla treball (PAC1)	Índex
4	10-oct/16-oct	Anàlisi exhaustiu (3)	
5	17-oct/23-oct	Comparació teòrica (4)	
6	24-oct/30-oct	Preparació de la memòria	
7	31-oct/06-nov	Lliurament PAC2: Anàlisi entorns	Anàlisi Teòric
8	07-nov/13-nov	Instal·lació del programari (5)	
9	14-nov/20-nov	Programació d'exemples (6)	
10	21-nov/27-nov	Programació aplicació Android multi plataforma (7)	
11	28-nov/04-dec	Programació aplicació codi natiu (10)	
12	05-dec/11-dec	Comparació Practica (8,11))	
13	12-dec/18-dec	Preparació de la memòria	
14	19-dec/25-dec	Lliurament PAC3: resultats	Anàlisi Pràctic + Conclusions
15	26-dec/01-gen	Refinament de la documentació	
16	02-gen/08-gen	Preparació de la Presentació	
17	09-gen/15-gen	Lliurament PFC i Presentació	
18	16-gen/22-gen		

Capítol 2

Estudi descriptiu

En aquest capítol volem realitzar un estudi general de les tecnologies mòbils i en particular de les tecnologies mòbils multiplataforma. Per començar veurem una mica de historia fins arribar a veure a l'estat de l'art de les plataformes mòbils i del desenvolupament d'aplicacions per a elles.

2.1 Historia de les plataformes mòbils

La historia del que generalment coneixem com telèfons mòbils es remunta fa gairebé 40 anys. Al 1973, el Sr. Martin Cooper (al que es coneix com el pare de la telefonia mòbil) va realitzar la seva primera comunicació via telèfon mòbil a l'empresa Bell Labs. La seva historia és molt extensa i nomes ens pararem a veure alguns de les fites més importants de tots aquests anys i sobretot aquelles que han fet esdevenir tot el que disposem en la actualitat.

IBM Simon

La miniaturització, el increment de la capacitat de la bateria i la millora de les xarxes mòbils van ser els principals factors a millorar des de la comercialització al 1974 del DynaTAC, l'evolució del telèfon que el Sr. Cooper va utilitzar per realitzar la primera comunicació, fins al 1992 quan apareix el que es pot considerar com el primer *Smartphone,* el IBM Simon, un telèfon amb programes d'assistència personal *(PDA).* Uns anys més tard, al 1996, es comercialitzaria el Nokia Communicator 9000 que constava amb tecnologia Intel 386. Aquest terminal ja disposava de teclat QUERTY, possibilitat de comunicació via e-mail i de visualització de planes web en mode text. Aquest telèfon es basava en el sistema operatiu GEOS v3.0. No va ser fins al 1997 quan Ericsson utilitza el concepte de *Smartphone* amb el seu model GS88.

A finals dels 90 la majoria de telèfons mòbils disposen nomes dels serveis basics de telefonia i molta gent comença a necessitar un dispositiu que estigui

entre la seva PDA i el seu telèfon mòbil. Comencen a llençar al mercat PDAs amb capacitat de comunicació mòbil basades en les primeres versions de Palm OS, BlackBerry OS y Windows CE/Pocket PC. Sistemes que posteriorment esdevindrien sistemes operatius per els nostres Smartphones. Per altre banda els telèfons normals cada vegada incloïen més atractius multimèdia. A començaments del segle XXI es comencen a comercialitzar mòbils amb capacitat de reproducció de música en format MP3, apareixen les pantalles amb colors per poder visualitzar fotografies, s'inclou radio FM, apareixen les primeres pantalles tàctils, s'inclou la càmera de fotos, la possibilitat d'enviar missatges multimèdia, etc...

A partir del llançament al 2000 del Ericsson R380, el primer dispositiu amb el sistema operatiu Symbian OS, els telèfons mòbils cada vegada disposen de més capacitats d'assistència personal. La companyia Nokia impulsa l'ús d'aquest sistema operatiu amb la seva sèrie Communicator i amb la seva sèrie N.

Al 2002 la companyia RIM treu al mercat la seva primera BlackBerry amb funcionalitats mòbils integrades i optimitzat per l'ús de l'enviament de correus electrònics sense fils.

Al voltant del 2005 apareixen els primers dispositius amb tecnologia 3G i esdevé una revolució en la connexió a Internet des de dispositius mòbils.

Al 2007 la companyia Apple llança el seu dispositiu iPhone amb tecnologia tàctil i la revolució total esdevé al 2008 amb la sortida del model 3G i el llançament de la App Store. Es comencen a desenvolupar un munt de programes per iOS, el sistema operatiu del iPhone i a vendre mitjançant la App Store. Aquest model es fa molt popular i molts dels seus conceptes són copiats per la resta de companyies.

iPhone d'Apple

Al 2008 la companyia Google desenvolupa un sistema operatiu per a dispositius mòbils anomenat Android. El primer dispositiu el llença T-Mobile amb el nom de G1. El sistema operatiu és una revolució ja que integra moltes de les aplicacions de Google com el calendari, el correu o els Mapes. A finals del 2008

anuncia el seu **Android Market** per distribuir aplicacions per el seu sistema operatiu.

Comencen a aparèixer competidors de l' iPhone 3G, com el Samsung Instinct o la BlackBerry Storm amb pantalles tàctils i unes interfícies molt semblants. Comencen a aportar novetats com els vídeo en alta definició, millors càmeres o major capacitat d'emmagatzematge.

BlackBerry Storm

Apple no tarda en reaccionar i al 2010 treu el seu iPhone 4, per demostrar de nou que va un pas per endavant. El mateix any Google treu el seu propi dispositiu, el Nexus One, el dispositiu més potent fins aleshores amb el sistema operatiu Android.

HTC al 2010 s' avança a la resta presentant el HTC Evo 4G, el primer telèfon amb tecnologia 4G.

Al 2011 LG surt amb dos novetats, el LG Optimus 3D i el LG Optimus 2X. El primer el primer telèfon en 3D sense necessitat d'ulleres i el segon el primer telèfon amb processador de doble nucli, el Tegra 2.

Aquest és un petit resum de la historia de la telefonia mòbil però no ens podem oblidar de la gran quantitat de dispositius mòbils que tenim avui dia que no són telèfons mòbils. Entre ells cal destacar els netbooks i els tablets.

Els netbooks van popularitzar-se al 2007 quan Asus va treure al mercat el seu Asus Eee PC. L'èxit d'aquest mini portàtils va radicar a la seva portabilitat i al seu baix cost. Al inici es carregaven amb sistemes operatius per PC, com Windows Vista, però el seu consum era massa elevat i per això últimament molts d'ells disposen de sistema operatiu Android.

iPad d'Apple

El terme *Tablet* es va popularitzar al 2001 quan Microsoft va presentar el concepte de *Microsoft tablet PC*. Encara que el gran èxit va esdevenir quan al 2010 Apple va treure al mercat el seu iPad. Un dispositiu tan odiat com

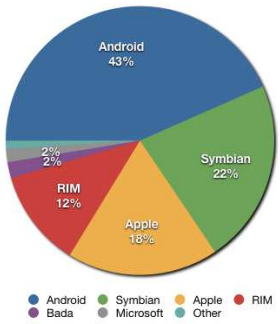
desitjat. Va tenir molts detractor encara que va vendre 2 milions de unitats els primers 60 dies. Aquests es poden considerar com una mena de *smartphones* amb una pantalla amb una mida més grans. Gairebé totes les marques de dispositius mòbils tenen actualment un *tablet* al mercat, la majoria d'ells amb alguna de les distribucions de Android.

2.2 Estat d'art de les plataformes mòbils

Com hem vist fins fa relativament poc, encara que 10 anys tecnològicament parlant és molt, els telèfons mòbils nomes eren aparells amb els que es podien realitzar trucades, enviar missatges *SMS* i poca cosa més. Hi havia un petit mercat per professionals on dominava RIM per la integració al terminal del servei de correu electrònic per l'empresa. En aquells moments gairebé no s'instal·laven aplicacions, nomes algunes amb Java, gairebé tot eren jocs, però l'únic mètode de distribució era mitjançant missatges *SMS*.

El gran canvi sorgeix al 2007 quan apareix el iPhone. La seva interfície d'usuari, els gestos multi tàctils i sobretot la sortida de l'App Store fa que el telèfon passi de ser un dispositiu amb un software fix a estar totalment obert a canvis sense tenir que canviar de dispositiu.

Després d'això han vingut noves plataformes com Android, WebOs, Windows Phone, Meego i altres com Symbian estan perdent quota de mercat. Parlem una mica d'algunes d'elles.



Dispositius venuts mundialment al 2011 per plataforma segons Gartner

Sense cap dubte, la plataforma **Android** és la més estesa actualment, ja que un percentatge molt alt de dispositius integren aquest sistema. Per fer-nos una idea, gairebé la meitat de les vendes al 2011 són seves i segons la consultora IDC s'espera que dintre de 4 anys Android tripliqui la quantitat de dispositius Apple i que Symbian gairebé desaparegui.

També hem de parlar de Microsoft que actualment té al mercat el seu **Windows Phone 7** i que s'espera que ens els pròxims 4 anys s'arribi al 20% de les

vendes, multiplicant per 10 les seves vendes respecte a les actuals. Un dels primers dispositius al mercat és el **Nokia Lumia** llençat a finals del 2011 i s'espera que moltes altres marques, com LG o Motorola, treguin durant el pròxim any dispositius amb aquest *SO* al mercat.

WebOs és la plataforma desenvolupada per Palm i que actualment és propietat de Hewlett-Packard (http://www.hp.com). En la segona meitat del 2011 hi ha hagut moltes noticies sobre aquesta plataforma ja que a l'agost HP va anunciar que no continuaria donant suport a WebOS, des de el punt de vista de hardware. WebOs va costar 1200 milions de dòlars a HP i encara no han sabut treure profit d'aquesta plataforma. No es descarta que un altra companyia compri aquesta plataforma o que la mateixa HP la continuí utilitzant en un altre divisió de la seva empresa com al mercat de les impressores.

Apple ha tret a finals del 2011 el seu **iOS 5**. Era un sistema molt esperat ja que estava previst que la seva sortida vingués de la ma d'un nou dispositiu iPhone. Finalment no va arribar el esperat iPhone 5 però si una nova versió del iPhone 4 integrant el revolucionari *Siri*. Aquest sistema suposa un altre revolució en la interacció amb el telèfon mòbil com ja va fer amb la pantalla multi tàctil al 2007.

Molts analistes acusen a RIM de no estar a l'altura de les circumstancies en quant a hardware. Té durs competidors actualment com l'iPhone 4S, el Samsung Galaxy S II, el HTC Sensation o el Nokia Lumia. Segons Gartner la quota de RIM en sistemes operatius va baixar del 19% en el primer trimestre del 2010 al 13% del mateix període de 2011. Encara que el sistema **BlackBerry** sempre ha estat lligat al món empresarial, en els últims temps va descobrir un nou mercat, el dels adolescents. El seu sistema integrat de missatgeria instantània i el seu teclat QWERTY ha fet que molts joves optin per l'adquisició d'un dispositiu BlackBerry. Aquest augment de vendes s'ha vist frenat amb la sortida de aplicacions com *Whatsapp* o *iMessage* que fan el mateix que feia el BlackBerry Messenger.

El món de les aplicacions també ha evolucionat moltísim i ara la tendència general és distribuir-les mitjançant algun tipus de mercat virtual i sempre en format electrònic sense cap suport físic. Cadascuna de les plataformes té el seu

mercat virtual, **OVI** pels dispositius Nokia, **Android Market** per els dispositius amb Android o **AppStore** per els dispositius amb iOS són alguns dels exemples.

Per conèixer una mica la situació actual direm que les dos grans botigues d'aplicacions són **l'Android Market** i **l'AppStore**. Això té molt a veure amb el nostre projecte així que s'ha d'analitzar molt be. Les empreses desenvolupadores d'aplicacions per mòbils fins ara estaven lligats de mans ja que tenien que decidir cap a on dirigir els seus projectes. El cost de desenvolupament es multiplica quan has de fer una mateixa aplicació per diferents plataformes. Aquelles empreses amb baix pressupost o que volen disminuir costos opten actualment per el desenvolupament d'aplicacions multiplataforma o bé per el desenvolupament d'aplicacions web directament.

L'Android Market té un creixement d'aplicacions impressionant ja que no posa gaires restriccions a l'hora de publicar les nostres aplicacions. Com a punt positiu, disposa de moltes més aplicacions gratuïtes que la Store de Apple. Per altre banda sempre s'ha associat la Store de Apple amb aplicacions de qualitat, potser per l'estudi exhaustiu que realitza Apple abans de permetre la publicació qualsevol aplicació a la seva botiga.

Altres tendes com l'**App World** de BlackBerry o **OVI** de Nokia disposen d'una quantitat molt inferior de aplicacions i de vendes. Tot va lligat a la quantitat de dispositius que hi ha al mercat. Les empreses desenvolupadores d'aplicacions mòbils analitzen com es reparteixen el mercat les diferents plataformes i generalment aposten per les que tenen major part, actualment Android i iOS.

L'empresa Ovum (http://www.ovumkc.com/) va divulgar al setembre de 2011 que el número de descarregues de l'Android Market al 2011 seria superior al número de descarregues de l'AppleStore. El Android Market podria acabar l'any amb un 8,1 mil milions de transferències contra els 6 mil milions de transferències d'Apple. Això demostra el gran creixement de la plataforma de Google ja que al 2010 nomes va aconseguir 1,4 mil milions de descarregues. De totes formes Apple també ha vist incrementat el número de descarregues ja que al 2010 va obtenir 2,7 mil milions de descarregues a la seva botiga oficial.

Segons el analista Nick Dillon, d'Ovum, el nombre de descarregues tendeix a créixer degut a que els consumidor busquen aplicacions que ofereixin noves experiències amb els seus dispositius.

Aquest creixement tant gran per part d'Android s'explica per la gran varietat de dispositius que hi ha disponibles amb sistemes Android. La gran oferta de tablets i smartphones amb les diverses versions d'aquest software fan que Google conquisti el mercat i faci front al domini d'Apple.

Encara que Google aconsegueixi més descarregues, Apple continua liderant les descarregues de pagament i per tant traient més beneficis de la seva botiga d'aplicacions.

S'especula que amb la sortida de nous dispositius amb Windows Phone 7 el seu **Windows Phone Marketplace** es posicioni als primers llocs del ranking de botigues d'aplicacions.

No podem deixar de banda un món paral·lel que està sofrint una revolució gracies a tot l'evolució de les aplicacions mòbils que és el món dels videojocs. En els últims mesos els jocs per smartphones i tablets van superar en beneficis a les principals consoles portàtils. La millora important en capacitats tècniques dels nous dispositius han permès competir i inclús superar als dispositius dissenyats específicament per jugar.

No obstant, s'ha de tenir en compte que no es tracta del mateix tipus de mercat, ja que els jocs per smartphones o tablets solen ser senzills i addictius, dirigits a jugadors *casuals,* mentre que a les consoles portàtils, trobem títols més complexes amb arguments més profunds en els que l'usuari ha d'invertir més temps.

Per posar un exemple, el videojoc més venut a la historia de les plataformes mòbils i que ha suposat una autentica revolució ha estat l'**Angry Birds**, on la seva base argumental és llençar ocells contra porcs apilats per anar acumulant punts i passant nivells.

Fa només dos anys aquest mercat estava dominat totalment per la consola portàtil **Nintendo DS** amb un 70% de la quota. Segons les previsions d'aquest 2011 aquest percentatge ha disminuït fins un preocupant 36%. Els principals motius del seu descens, el augment de jugadors casuals que busquen jocs per poder jugar en petites estones per perdre el temps, per exemple a la parada de l'autobús o a l'aeroport i el baix cost de les aplicacions mòbils, on podem trobar jocs per sota d'un euro en contra dels 40 o 50€ d'un joc de consola portàtil.

2.3 Sistemes operatius mòbils

Definim sistema operatiu mòbil com el sistema operatiu que controla un dispositiu mòbil al igual que els ordinadors utilitzen Windows o Linux. No obstant, els sistemes operatius mòbils són bastant més simples i estan molt orientats a la connectivitat sense fils, els formats multimèdia i actualment les xarxes socials i comunicació entre dispositius.

El sistema operatiu disposa d'un nucli o kernel que proporciona accés als diferents elements de hardware del dispositiu. Aquests són cada vegada més, pantalla tàctil, memòria externa, acceleròmetre o GPS són alguns exemples.

El middleware és el conjunt de mòduls que fan possibles l'existència d'aplicacions mòbils. És totalment transparent per l'usuari i ofereix serveis claus com el motor de missatgeria i comunicacions, codecs multimèdia, intèrprets de pagines web, gestió del dispositiu i seguretat.

La interfície d'usuari facilita la interacció amb l'usuari i el disseny de les presentacions visuals de l'aplicació. Els serveis que inclou són el de components gràfics (botons, llistes, pantalles, etc.) i el del marc d'interacció. Aquest ha sofert una evolució important ja que la majoria de dispositius ara porten pantalla multi tàctil i tot s'ha tingut d'adaptar a l'ús de les mans.

Analitzem com han evolucionat les diferents versions de sistemes operatius mòbils i que detalls fan que ens posicionem per un sistema operatiu o un altre.

Un de les parts més importants, com hem vist, és el nucli o kernel. Android utilitza un kernel Linux amb una barreja especial de Java. El iPhone/iPad està basat en OS X, que a la vegada és una variant d'Unix, un dels sistemes operatius més potents de la historia de la informàtica. RIM utilitza un kernel propi, que al igual que Android, té un motor Java.

L'evolució de la interfície d'usuari també fa que ens posicionem davant un sistema operatiu. De moment Apple és capdavanter en aquest camp ja que va ser dels primers en adaptar el seu sistema operatiu a pantalles capacitatives. Anteriorment es feia necessari l'ús d'un llapis o stylus per poder fer funcionar una pantalla resistiva però gracies a les noves pantalles capacitatives ja podem utilitzar els dits i fins i tot més d'un a la vegada el que fa que puguem utilitzar gestos multi tàctils. Encara que hi ha molta gent partidària del teclat físic, el teclat virtual fa que els dispositius puguin ser més petits i lleugers. També ha permès l'ús de pantalles cada vegada més grans.

Un dels punts que ha creat més controvèrsia ha estat el del Multitasking. Cada vegada els sistemes operatius tenen més funcions i aplicacions i es fa necessari que es puguin processar més d'una aplicació a la vegada. Fins que es va presentar iOS 4, era una batalla que tenia guanyada Android sobre Apple, però a partir de iOS 4 es va afegir un administrador de tasques per tal de poder tancar aplicacions o canviar entre elles.

Sobre el desenvolupament de tercers és important fixar-se en que la plataforma sigui robusta i senzilla d'utilitzar. Ja veurem al següent apartat més profundament els SDK que ofereixen cadascuna de les plataformes i com és de fàcil o complicat distribuir les aplicacions en cadascuna de les plataformes.

Veiem com han evolucionat els principals sistemes operatius mòbils fins a dia d'avui. S'ha de dir que l'evolució dels sistemes operatius origina una sèrie de versions que crea un problema de fragmentació que veurem més endavant.

Podem dir que **Palm OS** va ser el pioner dintre del món dels sistemes operatius mòbils amb la seva versió 1.0 llançada al 1996. Va anar evolucionant fins la versió 5.0 al 2002, anomenada Garnet i al 2004 presenten la seva última versió

6.0, anomenada Palm OS Cobalt. Finalment, PalmSource (*http://www.access-company.com/*) va desenvolupar un nou concepte de SO basat en Linux anomenat Palm WebOS que finalment va ser comprat per HP, donant lloc al seu **HP WebOS**.

Microsoft per la seva part va treure al mercat a l'any 2000 el seu Pocket PC 2000 que tres anys després, al 2003, va anomenar Windows Mobile 2003 en una nova versió. Aquesta versió va ser actualitzada l'any 2005 amb la versió 5.0 on es deixa d'utilitzar el any com a nom de la versió. Aquesta versió utilitza Windows CE 5.0 i .NET Compact Framework 1.0. En aquesta versió s'afegeix el paquet d'ofimàtica Office Mobile i Windows Media Player 10 Mobile entre d'altres. Fins al 2007 no surt la versió 6 amb canvis significatius però no és fins l'any 2009, quan presenten el seu Windows Phone 6.5, quan hi ha salt important respecte els sistemes anteriors, sobretot a la interfície gràfica per l'adaptació a les pantalles tàctils on es fa ús dels dits. També és la primera versió on es fa ús del nom Windows Phone i també és la primera versió on apareix el Windows MarketPlace on es poden comprar aplicacions per el nostre *smartphone*. L'ultima novetat de Microsoft és el **Windows Phone 7** presentat al 2010 amb una interfície adaptada del reproductor multimèdia Zune HD. Part d'aquesta interfície ha inspirat també la interfície metro de Windows 8. En aquests moments les principals marques de telèfons mòbils estan desenvolupant telèfons amb aquest sistema operatiu i s'espera que Microsoft torni a entrar en la lluita dels sistemes operatius mòbils on s'havia quedat endarrerit respecte a Android i iOS.

Per la seva part, **Symbian** és un sistema operatiu que va sorgir a partir de la unió de diferents empreses de telefonia mòbil com Nokia, Sony Ericcson, Psion, Samsung, etc. Està basat en el sistema operatiu EPOC32 de l'empresa Psion, utilitzat principalment en *PDA's*. El objectiu principal d'aquest sistema era competir amb el sistema de Palm i amb el Windows Phone de Microsoft. La primera versió data del 2001, anomenada Symbian OS 6.0. Aquesta va anar evolucionant fins la versió 9.5 (Symbian^3) llançada al 2010 que consta amb diverses actualitzacions al 2011 amb nom de dona com Symbian Anna, Symbian Belle i les que estan per arribar Carla i Donna.

iOS va ser presentat a l'any 2007 com iPhone OS per que originalment nomes anava dirigit cap aquest dispositiu. Posteriorment s'està utilitzant a *l'iPhone*, *l'iPad* i a *l'iPod Touch*. És un sistema operatiu derivat del seu germà gran, el Mac OS X. La versió 3.0 llançada a l'any 2009 va crear molta controvèrsia ja que solucionava dos de les característiques que més critiques havia rebut aquest sistema operatiu:l'opció de copiar i enganxar i l'enviament de MMS. A l'any 2010 es llançat el sistema iOS 4 on la principal novetat és la multi tasca. Actualment disposem de la versió **iOS 5** llançada a l'any 2011 on una de les novetats més rellevants és la sincronització i actualització sense fils i la integració de *iCloud*, on l'usuari pot penjar els seus continguts al núvol i tenir-los disponibles a tots els seus dispositius a l'instant.

El primer dispositiu BlackBerry va ser introduït al 1999 i funcionava com un localitzador. El seu sistema operatiu **BlackBerry OS** és propietari de RIM i és un entorn multi tasca desenvolupat per els dispositius de la pròpia RIM. L'any 2002, el BlackBerry més conegut es va llençar al mercat, juntament amb aquest es va presentar la versió 3.6. Aquest BlackBerry suportava push e-mail, telefonia mòbil, missatgeria de text, fax per Internet, navegació i altres serveis de informació sense fils. En 2010 RIM anuncia el seu sistema 6.0 amb moltes novetats adaptant-se als nous dispositius. Actualment disposem del **BlackBerry OS 7** que va ser llençat junt a una sèrie de nous dispositius. Un dels punts forts d'aquest sistema operatiu des de el primer moment ha estat el seu servei de correu electrònic i de missatgeria instantània.

Finalment veurem el sistema Android ja que és dels més contemporanis. **Android OS** és un sistema operatiu basat en Linux i dissenyat originalment per dispositius mòbils com *smartphones* però posteriorment s'ha expandit per soportar un gran número de dispositius com tablets, reproductors MP3, netbooks o televisors. La primera versió va ser alliberada al 2008. Com a curiositat totes les versions que van aparèixer posteriorment tenen nom de dolços. La més recent és l'anomenada Ice Cream (4.0). La seva evolució és molt constant ja que s'ha d'adaptar a un munt de dispositius diferents, no com iOS o BlackBerry OS que treballen amb els seus propis dispositius. També té molt a veure que treballa amb una comunitat oberta que fa que l'evolució sigui molt més rapida. Això suposa

també un problema conegut com a fragmentació. El sistema evoluciona però no dona temps a que tots els dispositius evolucionin amb ell, per tant, un dispositiu comprat amb la versió 1.6 com a molt el veurem actualitzat a la versió 2.0 i posteriorment ja no suportarà més versions ja que els requisits del sistema seran superiors al del nostre dispositiu.

Hi ha molts altres sistemes operatius com Meego o Bada amb una quota de mercat per sota del 5% i possiblement en els pròxims anys surtin molts més sistemes operatius per tal de desbancar als que es trobin als primers llocs del mercat.

2.4 Tecnologies de desenvolupament d'aplicacions mòbils

Després d'una petita introducció al món dels dispositius mòbils i una revisió als seus sistemes operatius més coneguts anem a veure com es desenvolupa programari per ells.

Tal com hem vist fa uns anys el software sempre venia preinstal.lat als sistemes operatius i no hi havia opció de ampliar-lo o modificar-lo. Tampoc existia l'opció d'actualitzar el propi sistema operatiu. Mica en mica els sistemes operatius van anant afegit compatibilitat amb la màquina virtual de JAVA i es van començar a distribuir petites aplicacions programades en JAVA, sobretot jocs. Conjuntament amb les melodies de mòbil van impulsar la distribució de continguts mitjançant els MMS. L'evolució de la comunicació entre dispositius passant de la connexió IR al *Bluetooth* també va fer evolucionar aquesta distribució de continguts. **Nokia** va donar un gran pas endavant amb el seu dispositiu N-GAGE. Aquest dispositiu era una barreja de telèfon intel·ligent i consola de videojocs. El sistema operatiu que hi havia al seu interior era Symbian de la sèrie 60, exactament el 6.1. Els jocs es distribuïen en targetes externes MMC i es podien fer córrer emuladors d'altres sistemes més antics. Van ser moltes les empreses que van sorgir en aquesta època, adaptant videojocs d'èxit al sistema Symbian.

Com hem vist els entorns d'execució més importants en l'actualitat són Android, iOS, BlackBerry, HP WebOS, Symbian OS i Windows Phone. Normalment les aplicacions per aquests han de ser compilades especialment per arquitectures

ARM menys Windows Phone/Mobile que pot ser compilat en executables x86 per ser depurats al PC sense cap mena d'emulador. La majoria d'ells ofereixen SDKs gratuïts i IDEs per als desenvolupadors. Veiem quin és el sistema de desenvolupament en alguns d'ells.

Desenvolupament amb Symbian

Symbian Ltd. originalment desenvolupa la base del sistema operatiu Symbian que inclou un microkernel, una sèrie de controladors, el *middleware* i una considerable quantitat de protocols de comunicació i interfícies d'usuari molt bàsiques. Els desenvolupadors han d'obtenir una llicencia per tal de treballar amb Symbian i a continuació implementar les seves pròpies interfícies de usuari i el conjunt d'aplicacions necessàries per els seus dispositius. Aquesta forma de treballar va permetre a Symbian que es posiciones amb un sistema molt flexible, que s'adaptava als requisits de la majoria de dispositius fabricats i al mateix temps permetia un alt grau de diferenciació entre fabricants.

Veurem com Symbian defineix una sèrie de interfícies d'usuari (*UI platforms*) que permeten definir famílies de mòbils segons la seva versió del sistema operatiu i les característiques de interfície d'usuari. Per posar un exemple, per la versió 9.1 de Symbian, tenim la plataforma s60 3rd Edition, per telèfons mòbils com el Nokia E70 i per aquesta mateixa versió de Symbian tenim la plataforma UIQ3 per mòbils amb pantalla tàctil com el Sony Ericsson p990. Degut a la gran quantitat de empreses amb les que ha hagut de treballar Symbian es van haver de diferenciar aquestes edicions segons les característiques del dispositiu mòbil objectiu. Actualment diferencia 5 tipus d'edicions. La principal diferencia entre elles no és tant el nucli del sistema operatiu sinó la interfície gràfica utilitzada.

Sèrie 60: És la més popular de totes degut fundamentalment a que Nokia, com sabem un dels fabricants més importants del mon, va fer de Symbian i de la sèrie 60 el nucli de quasi tots els seus models de smartphones. Defineixen una sèrie de dispositius amb pantalla petita i un teclat del tipus 0-9#. També és utilitzat per fabricants com Siemens, Samsung o Panasonic.

Sèrie 80: Aquesta edició, també utilitzada per Nokia, està més orientada a dispositius amb pantalla tàctil i permeten multi tasca.

Sèrie 90: Molt similar a l'edició Sèrie 80 però afegint dispositius amb pantalles més grans i que incorporen sensors tàctils més desenvolupats. Utilitzen teclats virtuals o connectats mitjançant per exemple *Bluetooth*.

UIQ: La interfície d'aquesta edició de Symbian es troba molt influenciada per Palm OS. Implementa una espècie de multi tasca virtual. Normalment aquesta sol tenir un alt cost computacional i influeix negativament en el temps de resposta del dispositiu. És utilitzada en alguns models de Sony Ericcson i Motorola.

MOAP: Aquesta edició utilitzada principalment per al fabricant FOMA es distribuïda únicament al Japó.

Podem trobar informació més detallada sobre les diferents versions de Symbian a la seva web (*http://www.symbian.com/phones/index.html*).

Desenvolupar per Symbian és relativament senzill. No és necessària aprendre cap llenguatge de programació nou ja que permet utilitzar llenguatges habituals com Java o C++ a l'hora de desenvolupar aplicacions. Aquest fet ha permès que hi hagi milers d'aplicacions i utilitats disponibles per Symbian actualment.

En general, cada versió consisteix en un conjunt de *APIs* que donen accés a les funcions de les que disposa el dispositiu mòbil. Hem de fixar-nos en el conjunt de funcions que necessiti la nostra aplicació, això determinarà quina versió de la plataforma Symbian necessitarem. Per posar un exemple, si la nostra aplicació necessita incorpora missatgeria instantània necessitarem la Instant Messaging API, que ens obligarà a utilitzar la versió de la S60 3rd Edition o superior. Una vegada definida la plataforma objectiu, podem descarregar i instal·lar la SDK corresponent. El paquet a descarregar inclou les llibreries necessàries, documentació, exemples i un emulador per poder provar les nostres aplicacions amb diferents configuracions sense la necessitat de disposar d'un dispositiu físic. L'emulador és imprescindible a les primeres fases del desenvolupament però sempre hem de tenir en compte que el seu funcionament no és 100% igual que el

del dispositiu real, així que es fa necessari fer les proves finals amb un dispositiu real.

Finalment seleccionarem el IDE amb el que treballarem. En aquest cas hi ha diferents opcions. Històricament el més utilitzat era **CodeWarrior** però des de fa temps Nokia aposta per **Cabide.C++,** que està basat en Eclipse i té una versió gratuïta.

La distribució de les aplicacions es fa majoritàriament gracies als mercats virtuals de les diferents empreses amb dispositius Symbian. Possiblement la més important actualment és la Nokia Store que distribueix continguts per els seus dispositius Symbian i també per els seus dispositius amb Java, MeeGo o Maemo.

Desenvolupament amb Windows Mobile

Com hem vist Windows Mobile és el sistema operatiu de Microsoft destinat a dispositius mòbils. Fa relativament poc ha estat substituït per Windows Phone 7 però analitzarem el desenvolupament d'aplicacions sobre la versió estable que teníem disponible fins ara.

Bàsicament Windows Mobile es va originar en una versió del kernel de Windows per sistemes embeguts que va anar creixent fins ser un sistema operatiu per ordinadors de butxaca (Pocket PC) i smartphones.

Per desenvolupar aplicacions per Windows Mobile necessitem baixar una de les SDK disponibles: versió Standard per smartphones o Professional per Pockets PC. Per tal d'instal·lar el SDK necessitarem una sèrie de requeriments disponibles a la web de Microsoft (*http://www.microsoft.com/*) dels quals crida la atenció la necessitat de disposar de Microsoft Visual Studio en versió llicenciada. És una mica sorprenent que Microsoft no ofereixi en aquest cas una forma gratuïta de que un aficionat pugui desenvolupar per Windows Mobile.

Una vegada instal·lat el entorn de desenvolupament i el SDK ja tenim tot el necessari per desenvolupar aplicacions per WM. Es poden desenvolupar dos tipus de aplicacions per WM: en codi natiu o en codi administrat (managed code). Anomenem codi natiu al codi C++ que utilitza directament l'API de Windows

Mobile i codi administrat al que utilitza les classes de .NET Compact Framework amb C# o VB.Net. Hi ha diferencies entre els dos, el codi natiu és més rapid i proporciona accés a algunes característiques del hardware que són inaccessibles des de el Compact Framework. Per altra banda, en la majoria dels casos desenvolupar en codi administrat és millor opció ja que el desenvolupament resulta molt més fàcil i còmode.

Com a punt positiu del desenvolupament en Windows Mobile, la qualitat de les eines disponibles (Visual Studio, SDK, documentació, activitat de la comunitat MSDN,...) i com a inconvenient principal la necessitat d'adquirir una llicencia per Visual Studio sense cap tipus de alternativa gratuïta. També hem de dir que sempre existeix algun opció menys o més legal de aconseguir-ho, existeix una eina anomenada VC++ que és gratuïta.

Per desenvolupar amb el nou **Windows Phone 7** és necessari disposar de Visual Studio 2010 i programar en C#.

La principal forma de distribució actualment és mitjançant el Windows *Phone Marketplace*.

Desenvolupament amb J2ME

J2ME és com es coneix el *Java Platform Micro Edition*. És un subconjunt de Java orientat a dispositius de recursos limitats, és a dir, amb unes característiques com memòria o processador inferiors als d'un ordinador convencional.

La principal virtut, així com en el cas del seu germà gran, és l'habilitat d'executar-se en diferents plataformes. Avui en dia gairebé qualsevol mòbil incorpora suport Java. Tot i això no tots els dispositius tenen el mateix suport de J2ME.

J2ME es divideix en configuracions (*Configurations*), perfils (*Profiles*) i **APIs** opcionals. Les configuracions defineixen el tipus de dispositiu en funció de les seves característiques a nivell de hardware. En la actualitat existeixen dos tipus de configuració: **CDC** (**Connected Device Configuration**), utilitzat sobre tot per sistemes de telemetria, automoció o domòtica y **CDLC** (*Connected Limited Device*

Configuration) que és una versió més limitada i que és la ens interessa per que es present a la majoria de telèfons mòbils.

Dintre d'una configuració, un perfil ens defineix una sèrie de característiques concretes, com la interfície d'usuari. Existeixen tres perfils per la configuració CLDC: **MIDP** (*Mobile Information Device Profile*) que és la utilitzada en telèfons mòbils , **IMP** (*Information Module Profile*) que és una versió sense interficie d'usuari i **DoJa** que està destinat a un tipus de mòbil japonès.

Finalment, cada dispositiu mòbil pot incloure suport per a diferents APIs opcionals. Per exemple si el dispositiu té Bluetooth necessita disposar de la Bluetooth API. Com aquestes capacitats varien segons el dispositiu, abans de començar a desenvolupar tenim que disposar de les especificacions del dispositiu al que ens dirigim per veure quines APIs suporta.

En ocasions es dona el cas de que les APIs desenvolupades per cada fabricant no segueixen totalment la implementació de referència que dona **Sun** (*www.oracle.com/es/index.html*) i es troben a la practica diferencies significatives en el comportament que creen la necessitat de desenvolupar més d'una versió de la mateixa aplicació.

A l'hora de començar a desenvolupar en aquesta tecnologia ens trobem amb el dilema de triar per quin dispositiu volem programar. Les marques líders que treballen amb aquesta tecnologia són Nokia, Samsung, Motorola, Sony Ericsson y RIM. La millor opció és trobar un IDE amb el que fàcilment puguem treballar per aquests fabricants amb els seus SDKs i emuladors. Nokia, Motorola i Sony Ericsson faciliten eines basades en Eclipse com Eclipse+EclipseME *(http://eclipseme.org/)* o **Motodev** (*http://developer.motorola.com/)* . Amb aquestes eines i els SDKs corresponents i la documentació amb els exemples ja podem començar a programar.

Una bona opció és començar per Nokia per exemple utilitzant Eclipse amb el plug-in **EclipseMe** y la SDK adequada. Una vegada tenim el codi que funciona la provem a la resta de emuladors, si en algun d'ells falla passarem el codi a l'entorn de desenvolupament que dona conflictes i treballarem des d'allà.

Desenvolupament per a BlackBerry

Com s'ha indicat la plataforma per desenvolupar aplicacions per BlackBerry és Java Me i tot el que s'explica al punt anterior és vàlid per aquesta plataforma. Tot i així existeixen una sèrie de particularitats que anem a veure a continuació.

En primer lloc les BlackBerry originalment són dispositius orientats al negoci, és a dir, estan pensats per ser clients de una xarxa corporativa que inclogui una solució integral amb correu electrònic, serveis web, seguretat integrada,etc. Per aquestes aplicacions empresarials existeix un entorn de desenvolupament especial anomenat BlackBerry MDS Studio. Bàsicament és un editor que permet crear interfícies d'usuari amb la tècnica *drag and drop* y codi Javascript amb l'objectiu de accedir mitjançant serveis web a les dades situades a un servidor empresarial.

En segon lloc dèiem que les aplicacions per BlackBerry es porten a cap mitjançant JAVA Me. Es desenvolupa amb un sistema anomenat *BlackBerry Handheld Software (http://es.blackberry.com/support/kb/KB04538_SPA.html) que* ofereix per un costat les APIs de CLCD 1.1 i de MIDP 2.0 i per altre banda unes APIs JAVA pròpies de BlackBerry. Aquestes APIS ofereixen serveis basics com interfície d'usuari, localització, accés a xarxes i serveis com sincronització o missatgeria. Aquests serveis són controlats ja que per ser instal·lats en un dispositiu BlackBerry, el codi ha d'anar firmat amb una certificació autoritzada per RIM.

Com a conclusió, per desenvolupar una aplicació per BlackBerry tenim dues opcions, programar-la amb JAVA Me amb el que obtindrem un codi que servirà per altres dispositius amb suport CLDC 1.1 i MIDP 2.0 o l'opció de desenvolupar l'aplicació amb les APIs propietàries de BlackBerry, aquesta serà l'opció obligada si volem donar a la nostra aplicació la imatge característica de les aplicacions BlackBerry o si volem utilitzar les funcions més avançades.

BlackBerry proporciona gratuïtament un entorn de desenvolupament anomenat BlackBerry Java Development Environment o **BlackBerry JDE** (*http://us.blackberry.com/developers/javaappdev/javadevenv.jsp*) . Aquest IDE ens dona accés a les dos opcions comentades anterior i incorpora una sèrie de

simuladors de dispositius BlackBerry per tal de fer proves amb les nostres aplicacions.

Com a principal avantatge d'aquesta plataforma veiem que els desenvolupadors de JAVA poden adaptar-se ràpidament a la programació per BlackBerry, en contra, trobem que moltes vegades la mateixa versió de sistema operatiu BlackBerry no sembla comportar-se igual segons el dispositiu en el que s'executa. Per sort com hem dit el seu IDE ve acompanyat de simuladors de la majoria de dispositius i això ajuda a poder fer el màxim de proves abans de llençar l'aplicació al mercat.

La plataforma de distribució principal de BlackBerry en la actualitat és la seva **App World** (*http://es.blackberry.com/services/appworld/*)

Desenvolupar per Android

Android és el somni dels desenvolupadors, és gratuït, de codi obert i es desenvolupat i recolzat per una de les empreses de desenvolupament de software més important dels nostres temps, Google.

Amb la finalitat de crear aplicacions per Android, s'utilitza un dels llenguatges de programació més populars, JAVA, i eines de primera classe que inclouen el plug-in i el emulador per el popular entorn de desenvolupament Eclipse.

L' utilització de tecnologies amb llarga trajectòria i populars eines, dona un gran avantatge a Android sobre la resta de sistemes. Encara que Android utilitza tecnologies conegudes, l'arquitectura i la plataforma són totalment noves i modernes, construïdes de zero i destinades directament als dispositius mòbils. Veiem a continuació una mica més en profunditat com treballa Android interiorment.

Android no é simplement un sistema operatiu com hem descrit anteriorment. Tal i com es descriu dintre de la documentació de Google (*http://code.google.com/android/*) Android és una pila de software per dispositius mòbils que inclou un sistema operatiu, middleware i aplicacions clau.

Aquesta estructura de pila de software respon a un sistema format per capes de software que es col·loquen una damunt de l'altra, de manera que cadascuna pot utilitzar la que téimmediatament a sota. Tenim la següent figura que descriu aquesta separació de capes.

Capes de software del sistema Android

A la figura podem diferenciar la part publica en blau composta per les aplicacions del dispositiu, moltes de elles basades en les aplicacions emblema de Google. També en blau, la part de llibreries que podem utilitzar gracies al SDK que distribueix Android, el **Application Framework**.

En verd podem veure les llibreries de sistema a les que en principi no tenim accés directe des de el nostre codi però que podrem utilitzar mitjançant la capa de Application Framework.

El motor del sistema el tenim en groc, el **Android Runtime**. Cadascuna de les aplicacions que s'executin dintre del sistema tindrà el seu propi procés amb la seva pròpia instancia de la maquina virtual. Les llibreries que trobem al Core inclouen les funcionalitat que trobem en les llibreries bàsiques de JAVA, com les d'entrada/sortida o utilització de cadenes, això no vol dir que la maquina virtual

sigui la màquina virtual de JAVA però si que JAVA és llenguatge utilitzat. Aquesta màquina virtual és un dels punts clau com veurem més endavant.

En la part vermella tenim el nucli del sistema operatiu, el **Linux Kernel**. Està basat en el nucli 2.6 de Linux i é la base de tot el sistema. Com veiem és el que dona accés a tot el hardware del dispositiu amb una sèrie de controladors.

Com hem indicat, un punt clau del sistema és **Dalvik**, la màquina virtual de Android. Dalvik és una màquina virtual molt lleugera, optimitzada per utilitzar molt poca memòria. Això ho aconsegueix delegant en el kernel de Linux el *multithreading* i la gestió de memòria i processos. Per programar per Dalvik s'utilitza un compilador de JAVA però el que es genera no són JAVA bytecodes sinó un tipus diferent que no ofereix compatibilitat amb JAVA SE, encara que ofereix llibreries estàndards de Java.

En resum, Google aprofita els punts forts de la màquina virtual de JAVA sense tenir que donar explicacions a Sun i sense haver de demanar permís per realitzar les modificacions que facin falta. A més, un punt molt positiu, és que tots els programadors acostumats a JAVA no noten gairebé la diferencia a l'hora de programar per Android i la corba d'aprenentatge és molt suau.

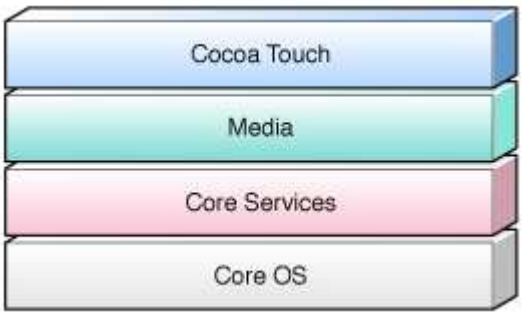
Veiem una mica més en profunditat el SDK que utilitzem per desenvolupar per Android. Les aplicacions d'Android es construeixen al voltant de quatre conceptes: las activitats (Activities) que representen les pantalles de l'aplicació, els receptors objectiu (Intent Receivers) que gestionen els events del sistema, els serveis (Services) que executen el codi no associat a la interfície d'usuari i per últim els proveïdors de continguts (Content Provider) que se encarreguen d'emmagatzemar dades i compartir dades. També s'ha de destacar el format XML per la definició de la interfície d'usuari amb la que Google promet acabar amb els problemes de *layout* tant habituals als dispositius mòbils.

El SDK (*http://developer.android.com/sdk/index.html*) inclou un plug-in per Eclipse que dona la opció de utilitzar varies eines del SDK des d'aquesta IDE. A més a més disposem del típics assistents de creació de codi i compilació.

Les aplicacions majoritàriament es distribueixen mitjançant el conegut **Android Market**. Aquest es diferencia de la resta per distribuir molt bones aplicacions de forma gratuïta.

Desenvolupar per iOS

Com ja hem vist iOS és el sistema implementat per Apple que permet executar aplicacions natives en els seus dispositius mòbils, fins ara al iPhone, iPod Touch i iPad. L'arquitectura d'aquesta plataforma es basa en el nucli del sistema Mac OS X incorporant una nova capa a l'estructura per donar suport a la interfície multi tàctil. Aquí podem veure les capes que formen el sistema.



Capes de l'estructura del sistema iOS

La primera capa, **Cocoa Touch**, és la que fa diferenciar aquesta estructura de l'estructura de Mac OS on trobem la capa Cocoa. Aquesta és una sèrie llibreries que tenen com objectiu controlar els gestos tàctils del usuari. La segona capa anomenada Media s'encarrega dels serveis multimèdia que es poden utilitzar a les aplicacions. La capa Core Services dona accés a tots els serveis que proporciona el sistema que trobem a l'ultima capa. La capa base Core OS é el nucli del sistema operatiu, s'encarrega de la administració de memòria, el sistema de fitxers, les comunicacions, entre d'altres tasques. Aquesta capa interactua directament amb el hardware del dispositiu.

Per desenvolupar per iOS és necessari utilitzar el SDK gratuït que posa Apple al nostre abast (*http://developer.apple.com/devcenter/ios/index.action*). Un dels aspectes que porta més critiques sobre Apple é la necessitat d'un compte de desenvolupador que té un cost de 79€ al any. Les aplicacions per iOS s'escriuen en

Objective-C, un llenguatge familiar per els que hagin desenvolupat anteriorment per Mac OS però per ningú mes. El entorn per desenvolupar aplicacions és el conegut **Apple XCode,** i aquí arriba l'altre punt criticat d'aquest sistema, només s'executa en sistemes Mac OS, el que significa que si no es disposa d'un equip d'Apple no podrem desenvolupar per iOS. Sempre podrem trobar solucions com ja vam veure mitjançant una màquina Virtual i una imatge de Mac OS.

El iOS SDK conté el codi, la informació i les eines necessàries per desenvolupar, provar, executar i ajustar les aplicacions per iOS. Dintre d'aquest kit trobarem tres aplicacions principals per desenvolupar per iOS:

Xcode: Conté un conjunt d'eines per el desenvolupament d'aplicacions. Permet editar, depurar i compilar el codi font.

Interface Builder: Permet la creació de interfícies gràfiques i la seva vinculació amb Xcode. A partir de XCode 4, Interface Builder està incorporat en la mateixa interfície que XCode.

iOS Simulator: Executa les aplicacions desenvolupades en un emulador del dispositiu físic.

Tot i aquesta inversió inicial necessària i la falta d'alternatives en l'elecció de hardware, el sistema operatiu, les eines i la documentació que ofereix Apple són simplement excel·lents. Aquesta és una de les raons principals de l'èxit i el gran número d'aplicacions disponibles per iOS.

Un altre punt que fa que les aplicacions disponibles per iOS siguin de qualitat és l'anàlisi previ que la pròpia Apple fa de les nostres aplicacions abans de donar-nos l'oportunitat de distribuir la nostra aplicació a la seva **App Store**.

Com podem veure hem fet un repàs a les tecnologies de desenvolupament més importants en la actualitat. Està clar que ens deixem moltes tecnologies per veure però aquestes són les que dominen actualment el mercat. A més a més hem volgut descriure aquelles que treballen amb codi natiu de la plataforma, deixant de banda JAVA Me que l'hem explicat perquè ha tingut una gran repercussió a la resta de tecnologies. A la *Wikipedia* podem trobar un gran quadre amb la majoria de tecnologies disponibles actualment amb dades comparatives entre cadascuna d'elles (*http://en.wikipedia.org/wiki/Mobile_application_development*).

També hem introduït en cadascuna d'elles els seus principals mètodes de distribució d'aplicacions. No cal dir que hi ha empreses de tercers que també distribueixen aplicacions mòbils com per exemple **Amazon** (*http://www.amazon.com/mobile-apps/*) i moltes altres.

2.5 Tecnologies de desenvolupament d'aplicacions mòbils multiplataforma

Segons un dels últims informes de la consultora Forrester *(http://www.forrester.com*), els desenvolupadors d'aplicacions mòbils intenten dirigir-se a la majoria de plataformes existents al mercat, encara que tenen com a preferència el iPhone de Apple i els terminals Android de Google.

El desenvolupament d'aplicacions mòbils està obrint el seu camp de visió cap a les multiplataformes mòbils per evitar la complexitat de desenvolupar una mateixa aplicació per a diferents dispositius i més encara ara quan l'evolució és tant constant i tenim dispositius nous gairebé cada dia. Forrester recomana al seu estudi crear des de el inici una estratègia multiplataforma, tant per el desenvolupament com per les proves.

El que va començar com "necessito una aplicació per un dispositiu concret" ha progressat cap al desenvolupament per a múltiples dispositius sobre tecnologies web que són suportades per qualsevol tipus de plataforma. Moltes empreses de desenvolupament d'aplicacions estan replicant les seves aplicacions natives per tal d'adaptar-se a la resta de plataformes.

Mentrestant, tecnologies mòbils Web com *WAP* (*Wireless Application Protocol)*, XHTML-Mobile o inclús JAVA Me comencen a desaparèixer, assegura Forrester al seu estudi " Encara que milers de milions de dispositius són compatibles amb aquestes tecnologies, no són atractives per als desenvolupadors, a més el temps mig de reemplaçament del dispositiu és de dos anys" afirma la consultora. No

obstant això, és interessant plantejar-se l'ús d'eines obertes de desenvolupament web com HTML5 o similars.

El mercat està en continu moviment iOS i Android són els que lluiten actualment al capdavant del mercat. Amb aquesta situació, com seleccionem els dispositius mòbils als que en dirigim i quins descartem?.

Les tecnologies multiplataformes són una solució per tal d'ignorar aquesta difícil decisió. Encara que són tecnologies que encara necessiten evolucionar donen al desenvolupador l'opció de desenvolupar de cop per dos o tres plataformes a la vegada.

A continuació farem un repàs d'algunes de les tecnologies multiplataforma recomanades. Està clar que hi ha moltes més però hem volgut fer una pinzellada sobre algunes d'elles.

Desenvolupament amb RhoMobile

Rhomobile Rhodes és un framework de codi obert basat en Ruby que ens permet crear aplicacions multiplataforma per la majoria de sistemes operatius. Actualment té suport per iOS, Android, RIM, Windows Mobile i Windows Phone 7. Un dels principals avantatges és la possibilitat de utilitzar hardware integrat del dispositiu, com el GPS o la càmera, de forma senzilla. Això és gracies a que les aplicacions són molt semblants a les aplicacions natives, no són aplicacions web mòbils.

A més a més, Rhomobile és l'únic framework que té suport per *MVC* (*Model View Controller*), els programadors que estiguin acostumats a treballar amb aquesta lògica de negoci ho agrairan moltísim.

Rhodes, en la actualitat a la versió 3.0, ofereix **RhoHub**, un entorn de desenvolupament organitzat y **RhoConnect**, un servidor independent que mante les dades de l'aplicació actual dels dispositius mòbils dels usuaris.

Podeu veure tota la informació a *http://rhomobile.com/*

Desenvolupament amb Appcelerator

Una de les plataformes favorites pels usuaris i que estudiarem en profunditat és Appcelerator Titanium. Actualment disposa d'una de les comunitats més actives de desenvolupament multiplataforma amb les següent xifres: 1,5 milions de desenvolupadors i més de 20.000 aplicacions. Gràcies a l'extensa documentació que podem trobar a la xarxa s'ha fet molt popular entre els programadors.

Dintre d'aquesta comunitat hi ha grans companyies com **eBay** o **PayPal** que utilitzen aquesta plataforma per crear les seves pròpies aplicacions.

Aquesta tecnologia ens permet desenvolupar tant aplicacions natives mòbils, com per *Tablets* i per aplicacions d'escriptori amb llenguatges web com JavaScript, PHP, Python, Ruby i HTML. La comunitat dona accés als seus integrants a més de 300 APIs. Actualment disposem de compatibilitat amb iPad, iPhone, Android i pròximament BlackBerry en fase beta.

Dintre dels principals avantatges que ofereix Appcelerator tenim que és gratuït tant per us personal com comercial, és software lliure, per tant tenim accés al codi font i podem verificar que no hi ha codi maliciós a la nostra aplicació. L'ús de llenguatges de programació web ofereix una corba d'aprenentatge molt suau i per finalitzar tenim a les nostres aplicacions un ús i un aspecte gràfic que sembla natiu.

Les aplicacions principalment són desenvolupades en JavaScript, utilitzant l'API que Titanium aporta en comú per les plataformes mòbils. Titanium precompila el JavaScript i l'empaqueta juntament amb un Runtime i les classes natives de iOS o Android.

El *Runtime* executarà el nostre codi línea per línea i cada invocació JavaScript a l'API deriva en una crida al codi natiu. Veiem un gràfic que ho deixa una mica més clar.

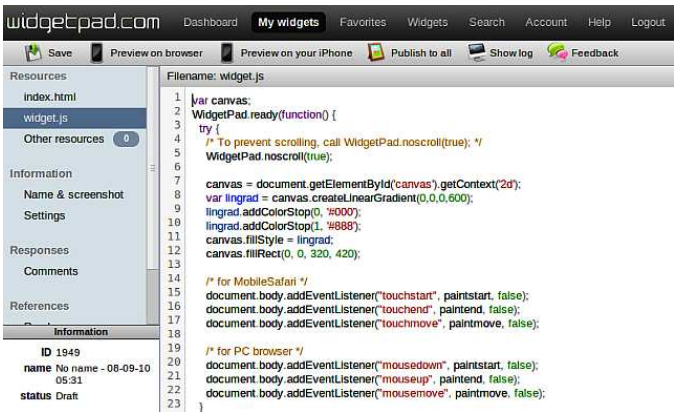
Hem de tenir en compte que el compilador de Titanium precompila el JavaScript per una execució rapida i empaqueta el projecte, però en cap cas transforma el nostre codi en codi natiu, això nomes succeeix durant l'execució de l'aplicació línea a línea i mitjançant el Bridge.

Podeu veure tota la informació a *http://www.appcelerator.com/*

Desenvolupament amb WidgetPad

WidgePad és una tecnologia de desenvolupament de codi obert basat en la web que genera un entorn per a que els desenvolupadors de programes per

smartphones puguin crear noves aplicacions utilitzant tecnologies web estàndards com CSS3, HTML 5 i JavaScript. WidgePad té per objectiu, com els altres frameworks que hem vist, disminuir la complexitat en el desenvolupament d'aplicacions per mòbils evitant aprendre l'API especifica de cada plataforma.

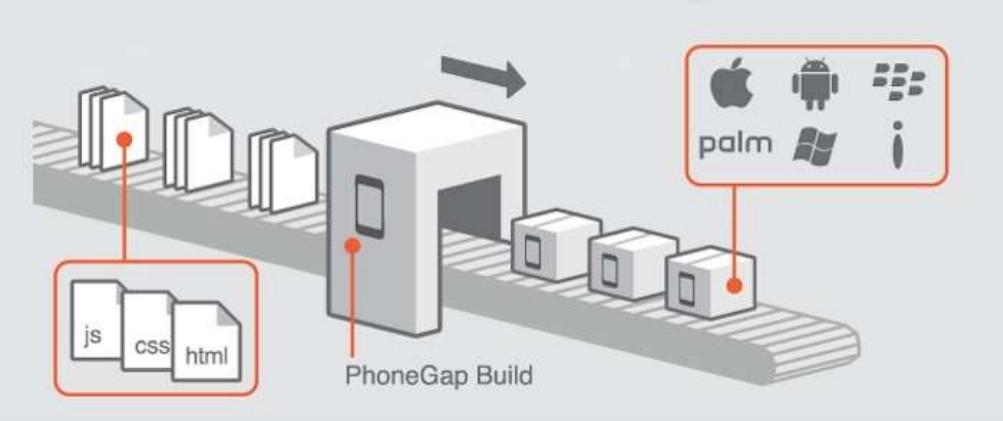


Aquesta versió inclou gestió de projectes, edició de codi font, depuració, la col·laboració de versions i la distribució. És pot utilitzar per crear aplicacions per sistemes operatius iOS, Android i WebOS.

Per més informació a *http://widgetpad.com/*

Desenvolupament amb PhoneGap

PhoneGap és un sistema per crear aplicacions utilitzant exclusivament HTML5, CSS3 i JavaScript, executades dintre d'un component WebKit del mòbil. Disposa d'una sèrie de llibreries JavaScript desenvolupades en el llenguatge específic de cada plataforma (Objective-C per iOS, JAVA per Android, etc.) que ens permeten accedir a les característiques del mòbil com GPS, acceleròmetre, càmera, contactes, bases de dades, filesystem, etc. Al ser una aplicació web, tenim accés al DOM i poder utilitzar flameworks web com jQuery o qualsevol altre.



Per treballar amb cada plataforma és necessari utilitzar un sistema distint. Per exemple, per iPhone/iPad és necessari utilitzar xCode i una plantilla que proporciona PhoneGap, per Android s'ha d'utilitzar Eclipse amb la seva plantilla específica.

Actualment suporta el desenvolupament per el sistema operatiu Apple iOS, Google Android, HP webOS, Microsoft Windows Mobile, Nokia Symbian OS and RIM BlackBerry. En les ultimes versions incorpora BlackBerry 5 i 6 i Windows Phone 7. Pròximament disponibilitat per Bada.

Per més informació consultar en *http://phonegap.com/*

Desenvolupament amb MoSync

MoSync és un altre software lliure multiplataforma basat en estàndards comuns de la programació. El SDK inclou compiladors totalment integrats, llibreries, perfils de dispositius, eines i utilitats, emuladors, etc. A més a més una IDE basada en Eclipse per C/C++ amb una documentació bastant detallada. Soporta JavaScript, Ruby, PHP, Python i altres llenguatges ja estan previstos.

El *framework* és compatible amb una gran quantitat de sistemes operatius, incloent Android, Symbian, Windows Mobile i inclus Moblin, una distribución de Linux mobil. Proximament s'espera el suport a iOS y BlackBerry.

Per més informació *http://www.mosync.com/*

Desenvolupament amb Corona SDK

Corona SDK és una excel·lent opció per tot tipus de desenvolupador d'aplicacions mòbils, des de principiants fins avançats. Corona és una eina per el desenvolupament fàcil i ràpid d'aplicacions i jocs per iPhone, iPad i Android. El més interessant d'aquest SDK és que disposa d'un motor físic molt avançat que ens permet crear jocs escrivint molt poques línees de codi.

Ansca és la companyia que hi ha darrera de Corona, aquest SDK permet als desenvolupadors crear aplicacions ràpides, poderoses i multiplataforma que tenen accés a APIs que ens permetran manipular components dels dispositius com càmeres, GPS o acceleròmetre.

Corona utilitza el llenguatge de programació **Lua** per crear aplicacions. Lua és un llenguatge de *scripting* normalment utilitzar per desenvolupar jocs. Té molta

acceptació per part de la comunitat i un aprenentatge molt senzill, sobretot per la seva similitud amb JavaScript o ActionScript 3. Per conèixer més d'aquest llenguatge podeu visitar *http://www.lua.org/.*

Per conèixer més sobre Corona visiteu *http://www.anscamobile.com/.*

Capítol 3

Estudi comparatiu

En aquest últim capítol, ens centrarem en les tres tecnologies que utilitzarem per el nostre estudi comparatiu, que com ja hem dit abans són **PhoneGap**, **Titanium Appcelerator** i **Adobe Air Mobile.**

Estudiarem en profunditat cadascuna d'aquestes tecnologies explicant com funcionen internament, quins requisits tenen i finalment realitzant un realitzant un cas pràctic. El cas pràctic, ens portarà des de la instal·lació de tots els components necessaris per el desenvolupament, fins a la programació d'una aplicació senzilla.

En tot el procés d'anàlisis anirem veient com s'assemblen o es diferencien al desenvolupament natiu de l'aplicació. En el desenvolupament del cas pràctic sempre ens basarem en Android ja que és una plataforma que tenen en comú les tres tecnologies però aportarem informació addicional per la resta de plataformes.

Una vegada estudiat el comportament de les tres tecnologies multiplataforma, realitzarem un estudi comparatiu entre elles, remarcant totes les virtuts i defectes detectats.

Finalment, compararem les tecnologies multiplataforma amb el desenvolupament natiu per cada plataforma. Veurem tots els punts forts i els punts febles que ens aporten les tecnologies multiplataforma.

3.1 Cas d'estudi 1: PhoneGap

Com ja hem vist en el transcurs d'aquest document PhoneGap és un *framework* per el desenvolupament d'aplicacions mòbils que possibilita als desenvolupadors a que implementin els seus projectes utilitzant tecnologies estàndards web: HTML5, Css3 i JavaScript. PhoneGap les transforma en

aplicacions hibrides, és a dir, aplicacions natives de les diferents plataformes mòbils existents que tenen accés a gran part del API natiu.

Això és possible gracies a que empaqueta l'aplicació web original amb un navegador basat en *webkit* per desplegar-la com si fos una aplicació veritablement nativa.

Actualment aquest framework suporta sis de les principals plataformes mòbils del mercat: iOS, Android, BlackBerry, PalmOS, Windows Mobile i Symbian. Per més informació sobre el estat actual del suport per cadascuna d'aquestes plataformes podeu visitar *http://www.phonegap.com/features*.

A continuació farem la descripció d'instal·lació de PhoneGap, coneixerem el seu funcionament intern a partir de la creació d'un projecte base amb aquest framework i la posterior elaboració de una aplicació senzilla.

Instal·lació de PhoneGap

En aquest cas farem una instal·lació dirigida a compilar l'aplicació per executar-la en una plataforma Android. Bàsicament entre les diferents plataformes és que inicialment hem de disposar del entorn de desenvolupament de la plataforma en qüestió i del seu SDK. Per exemple per Android disposarem d'Eclipse + ADT Plugin i l'Android SDK, per iOS necessitarem l'entorn Xcode d'Apple i per Windows Phone el seu Windows Phone SDK. Per veure els requisits per cadascuna de les plataformes visiteu *http://phonegap.com/start/* on trobareu informació molt detallada.

Descarreguem Eclipse de *http://www.eclipse.org/downloads/*, en aquest cas la versió Eclipse Classic 3.7.1. La versió Classic és una de les recomanades per la seva compatibilitat amb el plugin ADT. Per instal·lar només es necessari descomprimir el paquet en una carpeta. Normalment s'aconsella ubicar-la a l'arrel del sistema.

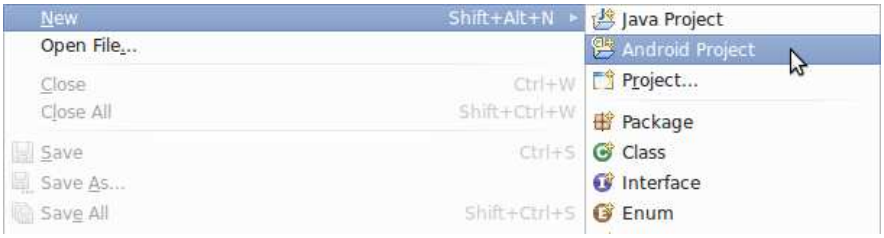
A continuació descarregarem l'Android SDK des de la ubicació *http://developer.android.com/sdk/index.html*. Es pot descarregar un paquet per descomprimir però en el nostre cas hem descarregat una versió instal·lable.

L'últim pas abans d'instal·lar PhoneGap és instal·lar el plugin ADT necessari per poder interactuar des d'Eclipse amb Android. Ho fem seguint les instruccions que ens donen a *http://developer.android.com/sdk/eclipse-adt.html#installing*. És necessari fer-ho des de el propi Eclipse, des de l'accés Help>Install New Software... i afegint el següent repositori *https://dl-ssl.google.com/android/eclipse/*.

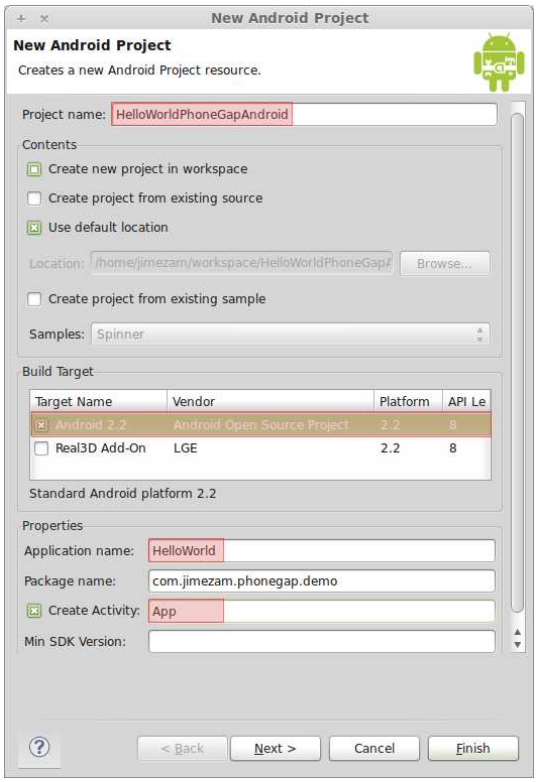
Ens dirigim a la següent ubicació *http://phonegap.com/download/*, des de on descarregarem l'ultima versió disponible del framework PhoneGap. A efectes de documentació, instal·larem la versió actual que en aquest moment és la 1.3.

Projecte base amb Android

Obrirem Eclipse i crearem un nou projecte d'Android des de File >New >Android Project.

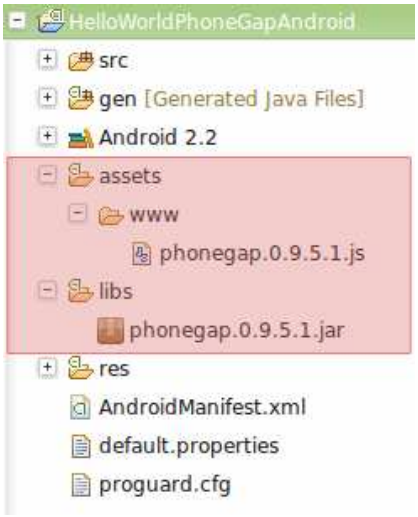


Indicarem a continuació el nom del projecte (*Project Name*), l'API d'Android que volem utilitzar (*Build Target*), el nom de l'aplicació (*Application Name*) i crearem una activitat (*Create Activity*).



Una vegada creat el projecte ens hem de dirigir a l'explorador de paquets (*Package Explorer*) d'Eclipse. Disposem d'una sèrie de carpetes que Eclipse crea per nosaltres. Crearem la carpeta /assets/www i la carpeta /libs si no disposem d'elles sota del nostre projecte.

Dintre de la carpeta /assets/www copiarem el fitxer **phonegap.x.x.x.x.js** i a la carpeta /libs copiarem el fitxer **phonegap.x.x.x.x.jar**. Els dos fitxers els podem trobar sota del directori /Android de la ruta on hem instal·lat el paquet PhoneGap anteriorment.



A continuació tenim que afegir la llibreria que acabem de copiar al nostre projecte. Fem clic dret sobre /libs i seleccionem el menú Build Path >Configure Build Path i a la pestanya Libraries afegim la llibreria /libs/phonegap.x.x.x.x.jar fent clic al botó Add JARs...

Seguidament modificarem el codi font de l'activitat per tal de fer referència a aquesta nova llibreria. Aquest fitxer està situat sota la carpeta /src del nostre projecte.

- Reemplacem la línia 3 (*import android.app.Activity;)* per la següent: *import com.phonegap.*;.*
- En la línia 6 canviar la superclase de App de Activity a DroidGap.
- Reemplaçar la línia 11 (*setContentView(R.layout.main);*) per la següent: *super.loadUrl(file://android_asset/www/index.html);.*

L'últim pas abans de començar a desenvolupar la Nostra aplicació és modificar el fitxer **AndroidManifest.xml** per tal de donar permís d'accés al

hardware del dispositiu. Fem clic dret damunt del fitxer i seleccionem el menú *Open With... >Text Editor*. Fem les següents modificacions:

1. Agreguem el següent text entre l'obertura de l'etiqueta <*manifest*> i l'obertura de l'etiqueta <*application*>.

```
<supports-screens
android:largeScreens="true"
android:normalScreens="true"
android:smallScreens="true"
android:resizeable="true"
android:anyDensity="true"
/>
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.RECEIVE_SMS" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.WRITE_CONTACTS" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

2. Afegim el següent atributo a l'etiqueta <*activity*>:

```
android:configChanges="orientation|keyboardHidden"
```

Finalment crearem el fitxer amb el codi font del nostre exemple. Aquest serà cridat des de l'activitat amb la comanda *loadUrl* de la superclase. Creem el fitxer **index.html** dins de /assets/www/.

```
<!DOCTYPE HTML>
<html>
<head>
<title>PhoneGap demostration with Android</title>
<script type="text/javascript" charset="utf-8" src="phonegap.js"></script>
</head>
<body>
<h1>Hello Android's World with PhoneGap</h1>
</body>
</html>
```

Per poder veure el resultat de la nostra aplicació necessitem executar l'emulador des del menú *Run >Run As >Android Application*.

3.2 Cas d'estudi 2: Titanium Appcelerator

Com ja hem vist Appcelerator Titanium és un Framework JavaScript per desenvolupar tant aplicacions per escriptori com per mòbils.

En la seva versió per a mòbils, que és la que ens interessa en aquests estudi, es pot utilitzar per desenvolupar aplicacions per iPhone, Android i BlackBerry. Es distingeix principalment dels altres frameworks per que genera aplicacions natives en lloc d'aplicacions que s'executen dintre d'un navegador.

A continuació en centrarem en preparar l'entorn de desenvolupament i de la creació d'una aplicació bàsica, com en el cas anterior la desenvoluparem per Android encara que la mateixa aplicació la podríem compilar per les altres plataformes.

Preparació del entorn de desenvolupament

La plataforma base de Titanium és software lliure sota llicencia Apache 2 i és gratuïta tant per us personal com comercial. A part de les avantatges del cost de desenvolupament, tenim accés al codi font i això ens permet verificar que no hi ha cap tipus de codi maliciós injectat a la nostra aplicació.

Un altre gran avantatge és programar en JavaScript. Els desenvolupadors poden aprofitar els seus coneixements existents en aquest llenguatge i aplicar-los per crear aplicacions mòbils natives. Això sobretot és molt avantatjós per els programadors d'iOS degut a la joventut de la plataforma.

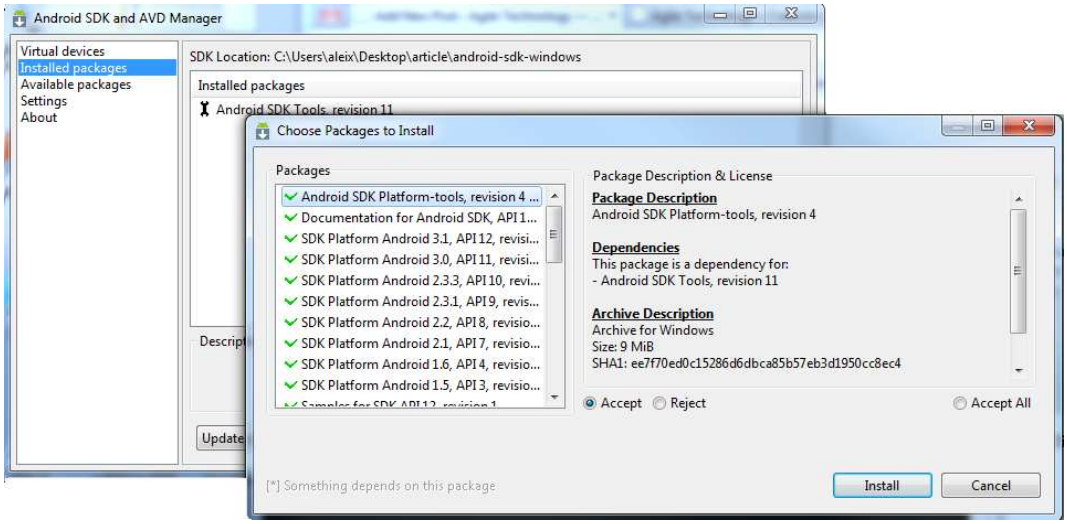
Per poder desenvolupar per Android amb Titanium necessitem un IDE com **Titanium Developer** (*http://developer.android.com/sdk/index.html*) o **Titanium Studio** (*http://preview.appcelerator.com/studio/*) i com per altre banda el **SDK d'Android** (*http://developer.android.com/sdk/index.html*). Si volem desenvolupar per iOS necessitarem com ja sabem un equip amb Mac OS i tenir instal·lat **XCode**.

Instal·larem Titanium Studio ja que Titanium Developer només porta eines per generar codi però no porta cap editor, en canvi, Titanium Studio és un IDE basat en Eclipse i ho té tot integrat.

Per instal·lar Titanium Studio només hem de seguir els passos de el instal·lador i finalment registrar-nos amb una compta d'Appcelerator que és gratuïta. Per instal·lar el SDK d'Android, com ja hem vist anteriorment, només cal descomprimir el fitxer en la carpeta on volem instal·lar-lo.

Configuració del SDK d'Android

Una vegada descomprimit el SDK d'Android, tenim que descarregar les diferents versions que ens interessi utilitzar. Per això executem l'aplicació *SDK Manager*.



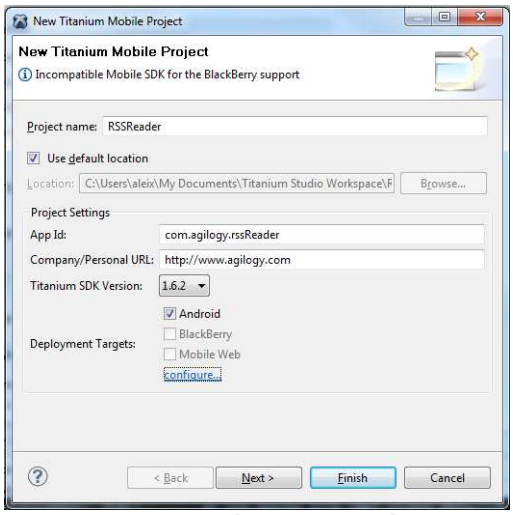
SDK Manager

Si no necessitem una versió especifica podem deixar marcat aquells que venen per defecte i continuar amb el botó *Install*. Tenim que diferenciar els SDK d'Android i els SDK d'Android de Google. Aquests últims tenen els components de Google com per exemple Google Maps.

És important tenir en compte que cada vegada que instal·lem nous SDK s'han d'eliminar els dispositius virtuals (Virtual Devices) que existeixin i tornar-los a crear perquè funcioni tot correctament.

Creació del primer projecte

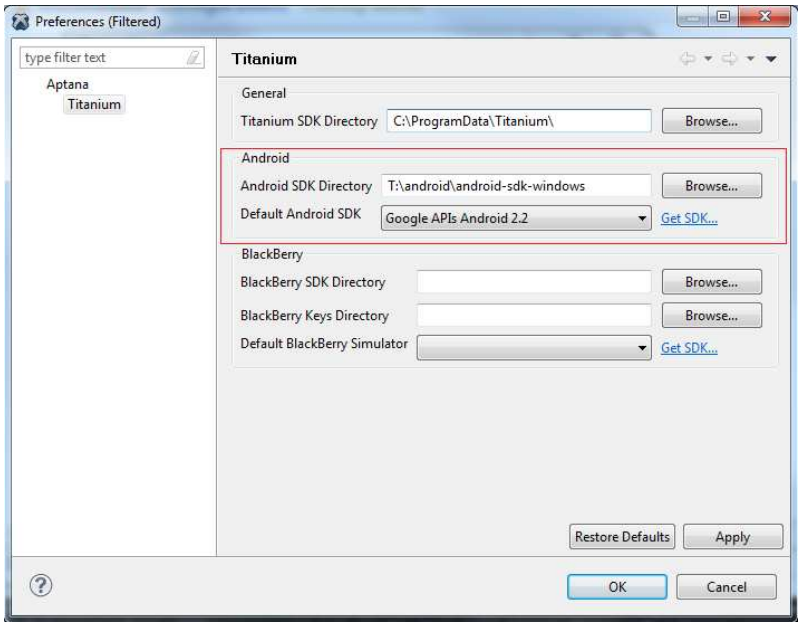
Arranquem Titanium Studio i creem el primer projecte des de *File -> New Titanium Mobile Project.*



Nou projecte de Titanium Studio

Omplim la informació bàsica del nou projecte.

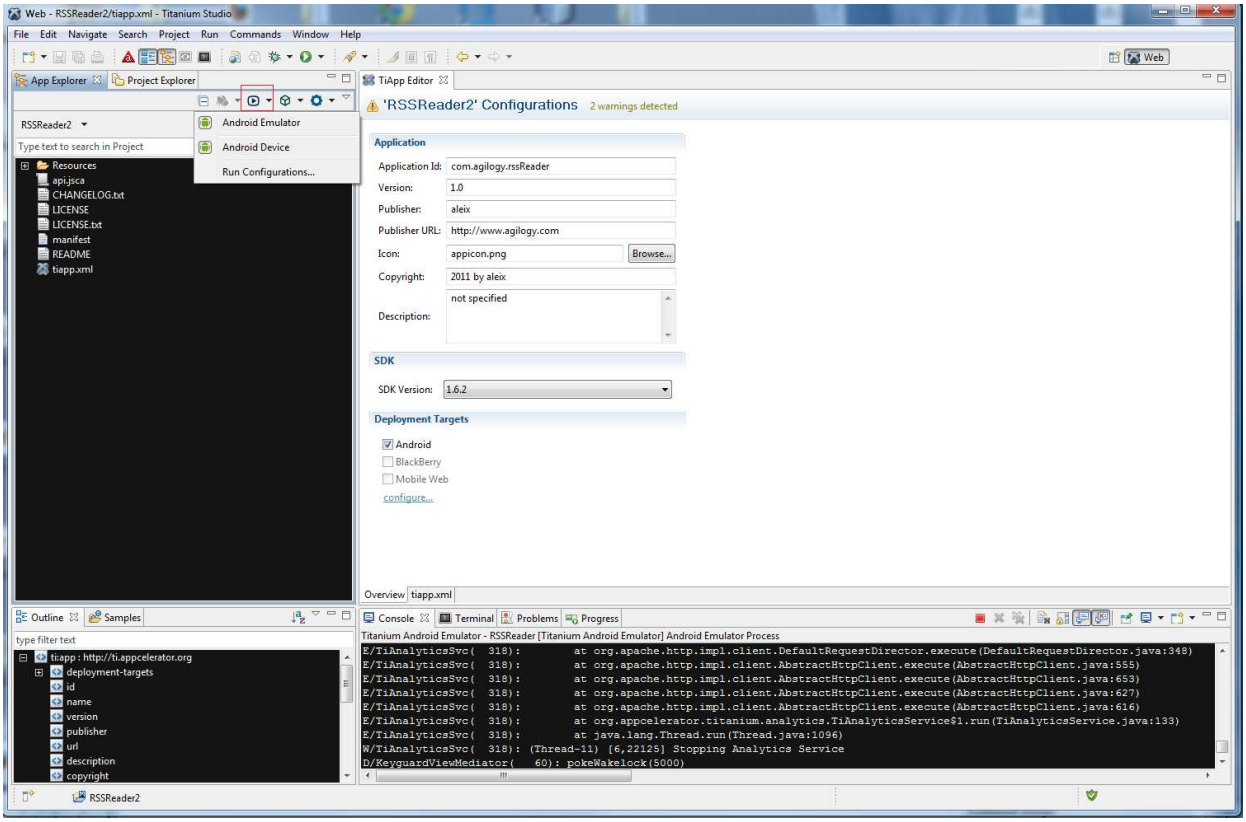
1. Project name: nom del nostre projecte.
2. App id: identificador de l'aplicació. Encara que si creem una aplicació per iOS no fa falta, en el cas d'Android el identificador ha de ser del tipus paquet de Java.
3. Titanium SDK Version: La versió del SDK de Titanium que utilitzem.
4. Deployment Targets: Els diferents dispositius als que va destinada la Nostra aplicació. Podem configurar cada una de les plataformes donant a '*configure…*'.



Configuració de plataformes

Si ens fixem en la part corresponent a Android, tenim que especificar el directori on hem instal·lat el SDK anteriorment i seleccionar la versió del SDK a utilitzar. **Per tal d'arribar al major número de dispositius tenim que utilitzar la versió mínima que necessitem**.

Una vegada creat el projecte passem a la vista general del entorn:



Vista General de Titanium Studio

- La part superior esquerra de la pantalla mostra els fitxers del projecte.
- La part superior dreta és la zona d'edició. En la captura es mostra el fitxer 'tiapp.xml' que és el fitxer de configuració de l'aplicació.
- En la part inferior esquerra es mostra l'estructura de fitxers oberts i es pot accedir a un altre pestanya amb alguns exemples.
- Finalment, en la part inferior dreta tenim les típiques pestanyes consola, sortida,... d'Eclipse.

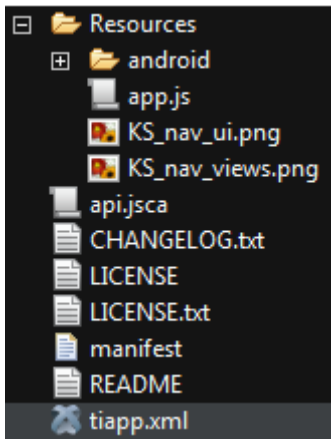
El propi assistent al crear un projecte nou ja genera una sèrie de fitxers d'inici que ja poden ser executats. Per fer-ho, es pot seleccionar la icona 'play' i

seleccionar *Android Emulator* per executar-lo amb el emulador o *Android Device* per executar-lo en un dispositiu físic. Provem a fer-ho amb el emulador:

Execució amb l'emulador

Estructura de fitxers del projecte

L'estructura de fitxers i directoris inicials és la següent:



Estructura de fitxers del projecte

- **tiapp.xml**: conté la configuració bàsica de l'aplicació.
- **app.js**: és el fitxer que s'executa al iniciar l'aplicació.
- Carpeta Android: conté els fitxers de recursos exclusius per Android. Si la aplicació es configura per executar-se també per iPhone tindríem un altre carpeta iPhone amb els recursos per iOS.

En general, podem posar fitxers amb el mateix nom a la carpeta Android i iPhone. Si ho fem, quan generem l'aplicació per un dispositiu concret s'utilitzarà el fitxer corresponent.

El fitxer app.js

Com s'ha indicat el fitxer app.js és el fitxer que s'executa inicialment. A partir d'aquí, i utilitzant la funció *Titanium.include(' ')* podem incloure els fitxers que vulguem i crear l'estructura de directoris més apropiada per nosaltres.

El codi de l'aplicació d'inici és el següent:

```
// this sets the background color of the master UIView (when there are no windows/tab
groups on it)

Titanium.UI.setBackgroundColor('#000');

// create tab group

  var tabGroup = Titanium.UI.createTabGroup();

//

// create base UI tab and root window

//

  var win1 = Titanium.UI.createWindow({

  title:'Tab 1',

  backgroundColor:'#fff'

});

var tab1 = Titanium.UI.createTab({

icon:'KS_nav_views.png',

title:'Tab 1',

window:win1

});

  var label1 = Titanium.UI.createLabel({

  color:'#999',

  text:'I am Window 1',

  font:{fontSize:20,fontFamily:'Helvetica Neue'},

  textAlign:'center',

  width:'auto'

});

win1.add(label1);

//
```

```
// create controls tab and root window
//
var win2 = Titanium.UI.createWindow({
title:'Tab 2',
backgroundColor:'#fff'
});
var tab2 = Titanium.UI.createTab({
icon:'KS_nav_ui.png',
title:'Tab 2',
window:win2
});
var label2 = Titanium.UI.createLabel({
color:'#999',
text:'I am Window 2',
font:{fontSize:20,fontFamily:'Helvetica Neue'},
textAlign:'center',
width:'auto'
});
win2.add(label2);
//
//  add tabs
//
tabGroup.addTab(tab1);
tabGroup.addTab(tab2);
// open tab group
tabGroup.open();
```

El codi és molt senzill. Totes les crides a les APIS de Titanium estan a l'objecte Titanium per lo tant comencen tenen l'estructura Titanium.xxx. Per abreviar es pot utilitzar l'estructura Ti.xxx.

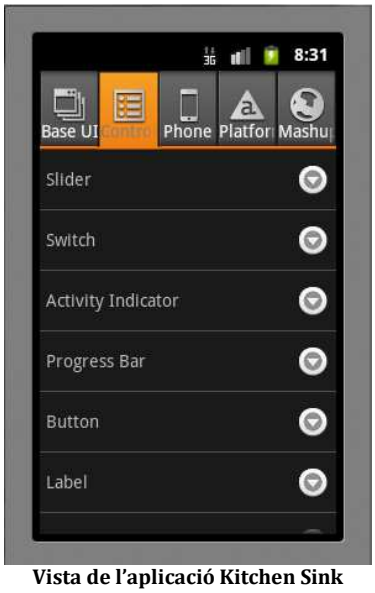
En termes generals, en una aplicació Titanium es crea la finestra amb la crida a **Titanium.UI.createWindow** i a continuació, es van afegint els diferents controls. Per veure en detall totes les funcions i paràmetres disponibles es pot consultar tota la informació sobre l'API de Titanium a la següent direcció *http://developer.appcelerator.com/apidoc/mobile/latest/Titanium-module*.

En totes les funcions de l'API es poden passar com a paràmetre objectes amb els diferents atributs que ens interessin.

Un exemple amb el que s'han realitzat proves per veure les possibilitats de Titanium és **Kitchen Sink**. Es pot descarregar el codi font de *https://github.com/appcelerator/KitchenSink/.*

Aquest exemple mostra diferents vistes, controls, etc. per veure els diferents aspectes segons cada dispositiu. Un mateix control creat amb Titanium utilitza el control natiu de cada dispositiu per tant es pot mostrar diferent al dispositiu Android que al dispositiu amb iOS.

Per tal de poder executar l'aplicació hem de tenir en compte que un dels controls que utilitza Kitchen Sink és el mapa, així que és necessari configurar l'aplicació amb l'API de Android de Google.



Vista de l'aplicació Kitchen Sink

3.3 Cas d'estudi 3: Adobe Air Mobile

Les aplicacions d'Air Mobile s'implementen com aplicacions natives. Utilitzant el format d'aplicació del dispositiu i no el format d'arxiu d'Air. Actualment AIR admet paquets APK d'Android i paquets IPA d'iOS. Una vegada creada la versió oficial del paquet de l'aplicació, aquesta és pot distribuir

mitjançant el mecanisme de cadascuna de les plataformes, per Android com ja sabem Android Market i per iOS App Store d'Apple. És pot utilitzar el SDK d'Air i Flash Professional, Flash Builder o altres eines de desenvolupament d'ActionScript per tal de crear aplicacions Air per dispositius mòbils. Les aplicacions d'Air Mobile basades en HTML no s' admeten en aquest moment. Amb Air Mobile també tenim la possibilitat de desenvolupar aplicacions per la tableta BlackBerry PlayBook de RIM, en aquest cas és la propia RIM qui proporciona el seu propi SDK per el desenvolupament d'Air. Per obtenir informació més detallada sobre el desenvolupament per aquest dispositiu podeu visitar el lloc web *http://us.blackberry.com/developers/tablet/adobe.jsp*.

A continuació configurarem l'entorn de desenvolupament i realitzarem la nostra primera aplicació amb Air Mobile.

Per tal de conèixer els requisits mínims que Air requereix en cadascun dels dispositius visiteu el lloc web *http://www.adobe.com/es/products/air/tech-specs.html*. Destaquem d'aquests requisits la necessitat d'uns mínims de maquinari bastant potents en el cas d'Android.

El flux de treball per crear una aplicació d'Air per dispositius mòbils és molt semblant al utilitzat per la creació d'una aplicació d'escriptori. Les diferencies principals es produeixen quan arriba el moment d'empaquetar, depurar i instal·lar una aplicació. Per exemple, les aplicacions AIR per Android utilitzen el format de paquet natiu **APK** d'Android en lloc del format de paquet d'AIR.

Veiem a continuació dos exemples de flux de treball en Android i en iOS. El desenvolupament per Android segueix els següents passos:

1. Escriure el codi ActionScript o MXML.
2. Crear un arxiu descriptor de l'aplicació d'AIR.
3. Compilar l'aplicació.
4. Empaquetar l'aplicació com un paquet Android (.apk).
5. Instal·lar el motor d'execució d'AIR en el emulador d'Android o el dispositiu (si encara no el tenim instal·lat).
6. Instal·lar l'aplicació en el dispositiu o emulador d'Android.

7. Iniciar l'aplicació en el dispositiu.
8. Una vegada finalitzada l'aplicació empaquetarem definitivament com fitxer APK i ja es pot distribuir a l'Android Market o per altres medis.

A l'hora de desenvolupar per iOS aquests són els passos a seguir:

1. Instal·lar iTunes.
2. Generar els ID i els fitxers del desenvolupador necessaris en *Apple iOS Provisioning Portal*. Aquest elements inclouen: Certificat de Developer, ID de l'aplicació i fitxer de subministrament. És necessari incloure aquests ID en qualsevol dispositiu de prova en el que s'hagin d'instal·lar l'aplicació al crear el perfil de subministrament.
3. Transformar el certificat de desenvolupament i la clau privada en un fitxer de magatzem de claus P12.
4. Escriure el codi MXML o ActionScript de l'aplicació.
5. Crear gràfics d'icones i de la finestra inicial de l'aplicació.
6. Crear el descriptor de l'aplicació.
7. Empaquetar el fitxer IPA utilitzant ADT.
8. Utilitzar iTunes per situar el perfil de subministrament en el dispositiu de prova.
9. Instal·lar i provar l'aplicació en el dispositiu amb iOS. Pot utilitzar iTunes per instal·lar l'arxiu **IPA**. Una vegada finalitzada l'aplicació d'AIR, pot tornar a empaquetar utilitzant un certificat de distribució i perfil de subministrament. Per últim ja es pot enviar a l'Apple App Store.

Com es pot comprovar Apple posa bastants entrebancs a l'hora d'empaquetar, executar i distribuir les seves aplicacions.

Configuració del entorn de desenvolupament

Com ja hem dit hi ha diferents formes de desenvolupar amb Air però la més còmode és utilitzar **Flash Builder**. Hi ha dos versions 4.6, *Standard* i *Premium* amb un cost de 199€ i 549€ respectivament. Nosaltres utilitzarem una versió de prova que es pot fer servir durant 30 dies. També existeix la possibilitat de desenvolupar només amb el SDK de Flex que és gratuït i utilitzant editors externs però

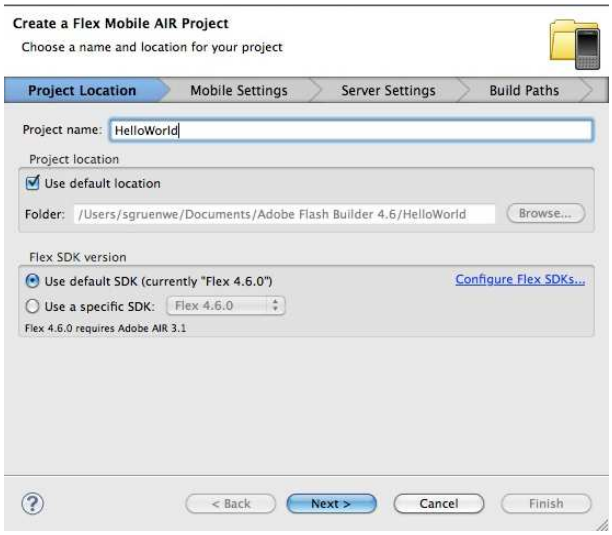
l'experiència no és tan profitosa ja que aquest Builder disposa de molts assistents que et faciliten molt la feina a l'hora de desenvolupar, compilar, empaquetar, etc...

Són necessari alguns requisits segons la plataforma per la que estem desenvolupant. Per Android és necessari el SDK d'Android per tal de crear i executar un emulador d'Android i és important configurar la variable d'entorn AIR_ANDROID_SDK_HOME amb la ruta del SDK d'Android instal·lat. Per tal de desenvolupar per iOS és necessari registrar-se al iOS Developer d'Apple (servei gratuït). Una vegada registrat pots accedir al *iOS Provisioning Portal* on podràs descarregar els següents elements que són indispensables per instal·lar una aplicació en un dispositiu per proves i per la distribució posterior. Aquests elements són: els certificats de distribució i desenvolupament, ID de l'aplicació i els fitxers de subministrament de distribució i desenvolupament.

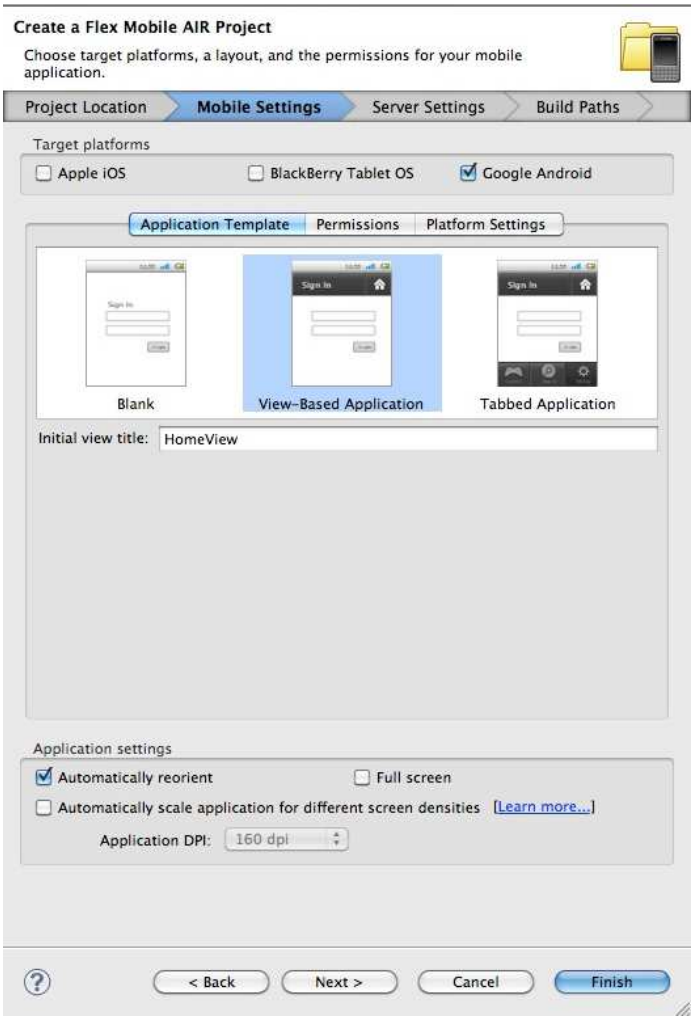
Creant un nou projecte per Android

Crear un nou projecte amb Flash Builder 4.6 és senzill. El procés de desenvolupament és pràcticament igual per la resta de plataformes, com ja hem vist la diferencia es fa evident en els passos posteriors al desenvolupament. Veiem quins passos són necessaris:

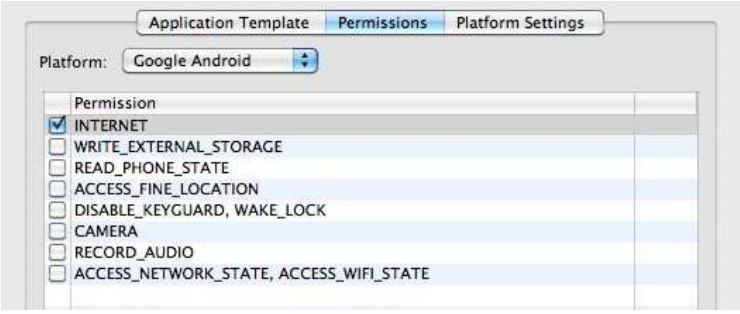
1. Anem al menu File> New > Flex Mobile Project per obrir el asistent.
2. Especifiquem el nom del nostre projecte, en aquest cas **HelloWorld**.
3. Deixem la resta de atributs amb els valors per defecte i fem clic a *Next*.



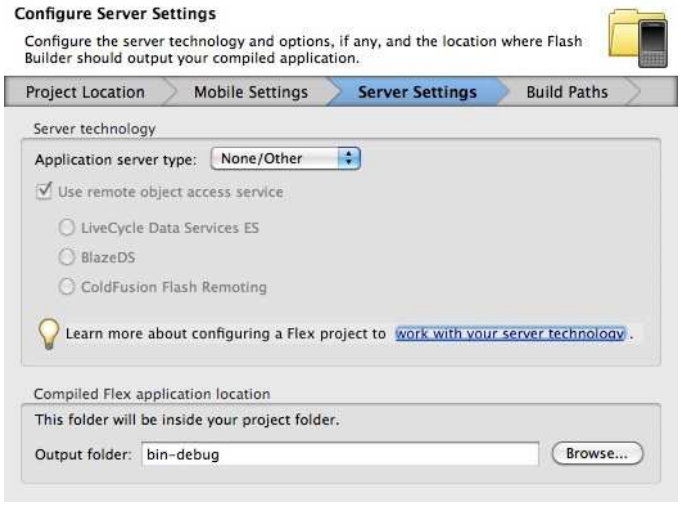
4. En la configuració del dispositiu mòbil marquem Google Android com a plataforma de destí. Hi ha tres possibles estils de disseny, seleccionarem *View-Based Application* i deixem la resta d'atributs amb els valors per defecte en aquest exercici.



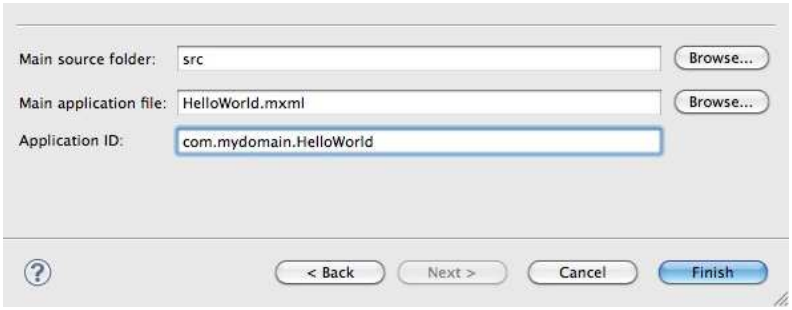
5. Opcionalment a la pestanya de Permisos es poden configurar els permisos d'accés que donem a aquesta aplicació. Per exemple en aquest cas donem accés a Internet. En part, aquest permís és necessari per depurar l'aplicació al nostre dispositiu.



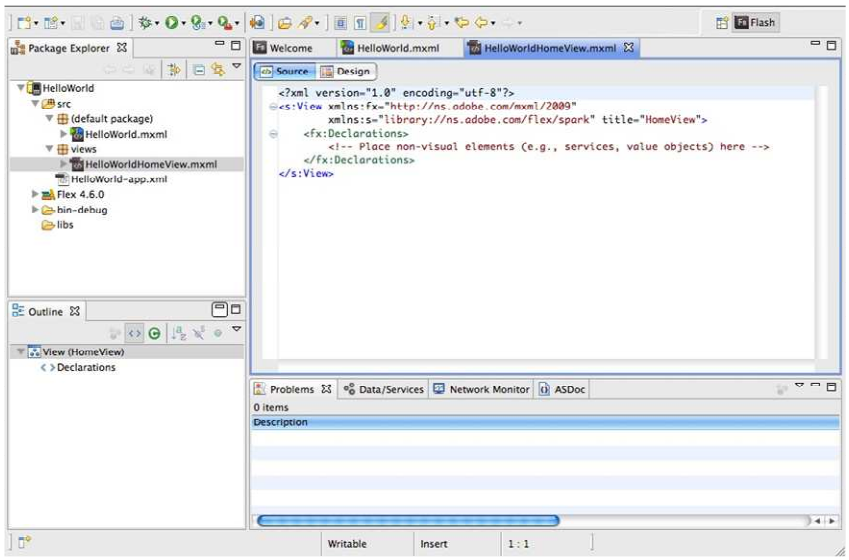
6. Deixarem la configuració de Servidor amb els valors per defecte i farem clic a *Next..* de nou.



7. A continuació Flash Builder crearà el projecte i actualitzarà la configuració de rutes des d'un assistent. En aquesta finestra podem afegir llibreries o qualsevol cosa que necessitarem posteriorment a la nostre aplicació. També podem canviar algun noms de fitxers i paràmetres. En el nostre cas canviarem el *Application ID* que és necessari per identificar la nostra aplicació. En aquest cas que no és un projecte real, podem posar qualsevol nom però en una aplicació real la aplicació ha de ser semblant a un *Fully Qualified Domain Name*, per exemple **com.mydomain.MyApp**, on MyApp és el nom de la nostra aplicació. En aquest exemple introduirem *com.mydomain.HelloWorld* com Application ID.



En aquest punt, fem clic a *Finish* i Flash Builder carrega el nostre projecte, obre el editor de codi i ens mostra la nostra nova aplicació.



Quan iniciem un nou projecte, Flash Builder generar els següents fitxers (on ProjectName és el nom que s'ha indicat al primer pas del assistent)

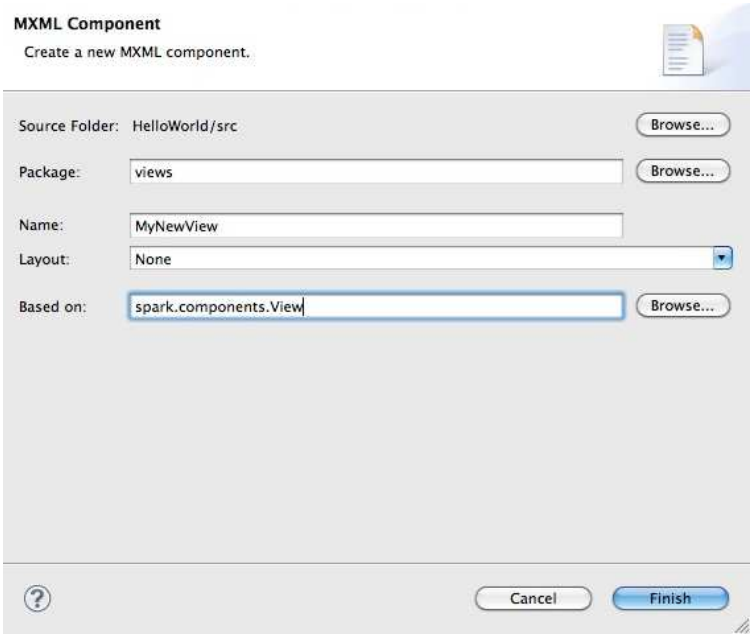
- **ProjectName.mxml**: Aquest és el fitxer de l'aplicació del projecte. Normalment no s'afegeix directament contingut a en aquest fitxer.
- **ProjectNameHomeView.mxlm**: Aquest fitxer representa la vista o finestra inicial del nostre projecte. Flash Builder col·loca aquest fitxer al paquet anomenat *Views*.

Fins ara hem creat una aplicació base, a continuació crearem la nostra aplicació d'exemple. Farem clic damunt del fitxer **HelloWorldHomeView.mxml** des del Package Explorer, al costat esquerra de la finestra del nostre projecte. Abans de tancar l'etiqueta *</s:View>* afegirem el següent codi:

```
<s:VGroup width="100%" height="100%" verticalAlign="middle"
horizontalAlign="center"> <s:Label text="Hello, World!"/> <s:Button
label="Continue" click="navigator.pushView(MyNewView)"
styleName="next"/> </s:VGroup>
```

Com podem veure s'utilitza un sistema d'etiquetes molt senzill on afegim una etiqueta amb un text i un botó que ens porta a una nova vista anomenada **MyNewView**.

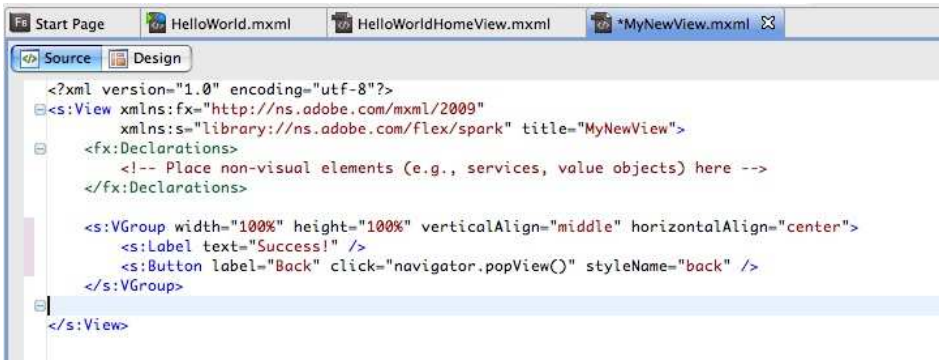
A continuació anem a crear la nova vista, abans guardarem els canvis realitzats a la anterior vista. Al menú principal anem a File > New > MXML Component. Veurem un petit assistent on afegirem les següents dades.



Com podem veure posem el nom de la nova vista i l'afegim al paquet *Views*. Anem a introduir el codi en aquesta nova vista, per això, obrim des del Package Explorer el fitxer MyNewView.mxml sinó s'ha obert automàticament i introduïm el següent codi abans de l'etiqueta *</s:View>.*

```
<s:VGroup width="100%" height="100%" verticalAlign="middle"
horizontalAlign="center"> <s:Label text="Success!" /> <s:Button
label="Back" click="navigator.popView()" styleName="back" />
</s:VGroup>
```

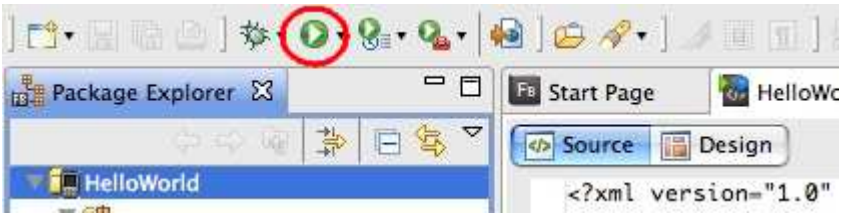
Com podem veure mostrarem una etiqueta amb el text **Success!** i un botó per tornar enrere. Guardem aquest fitxer i finalment ja podem provar la nostra aplicació.



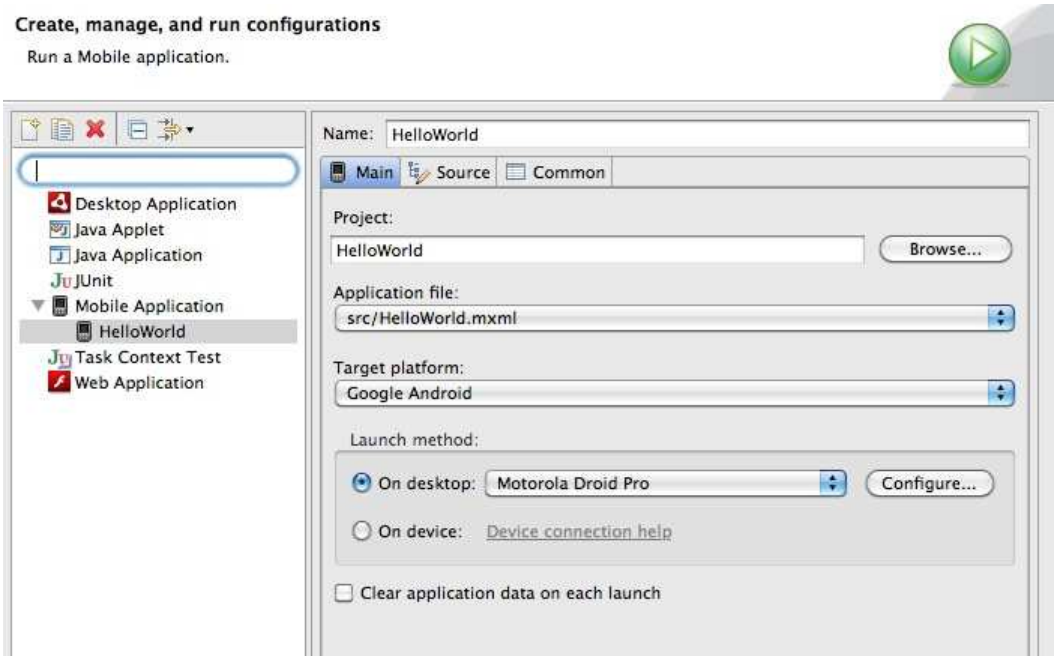
Provant la nostra aplicació

Es possible provar la nostra aplicació en un dispositiu físic com ja sabem però Flash Builder ens facilita aquesta feina deixant-nos executar i depurar la nostra aplicació des del propi entorn de desenvolupament.

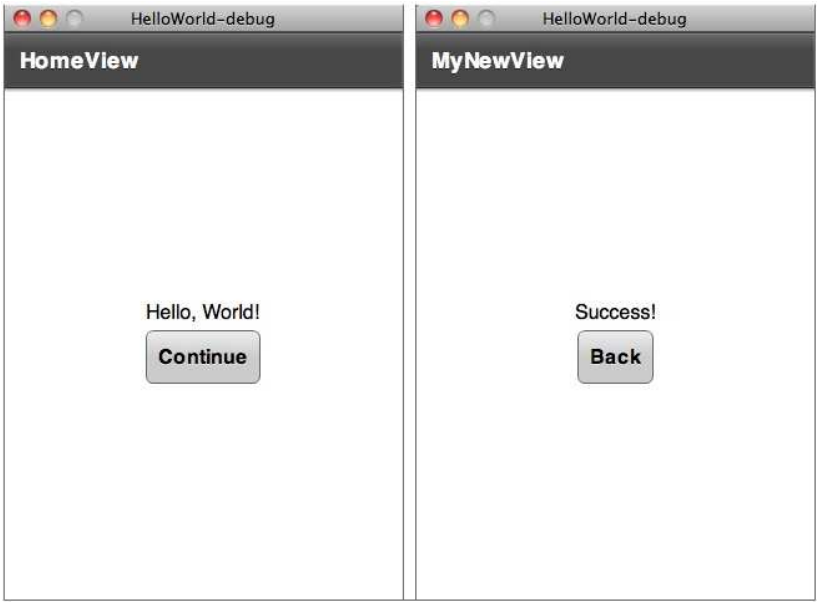
1. Ens assegurem que el fitxer HelloWorld.mxml està obert i actiu al editor de codi.
2. Fem clic damunt del botó **Run** de la barra d'eines.



3. Ens assegurem que la plataforma de destí és Google Android.
4. Com a *Launch Method* seleccionem *On Desktop* i posteriorment seleccionem un dispositiu per simular, per exemple el Motorola Droid Pro.



5. Fem clic a *Run* per acceptar la configuració i llençar la aplicació. A continuació el llançador d'aplicacions anomenat **ADL** ens mostra la finestra anomenada **HelloWorld-debug**. Podem interactuar amb el botó *Continue* per veure l'altra vista que hem creat.



Finalment guardem el nostre projecte i així hem creat la nostra primera aplicació Flex Mobile. Veiem a continuació el procés d'empaquetament de la nostra aplicació i posterior distribució.

Empaquetament del nostre projecte

Imaginem que ja tenim llesta la nostra aplicació per distribuir-la per exemple al Android Market. Podem exportar-la des del mateix Flash Builder.

1. Amb el projecte obert a Flash Builder, seleccionem al menú *Project > Export Release Build*.
2. En el quadre que s'obre ens fixem que el projecte HelloWorld apareix en el selector d'aplicacions.
3. Connectem el nostre dispositiu Android al ordinador si no ho hem fet encara.
4. Especifiquem el nom base del paquet Android (APK) o deixem el que hi ha per defecte.
5. Ens assegurem que tenim seleccionat l'opció '*Signed Packages for each target platform'* i fem clic a *Next* per començar el procés d'exportació.
6. Ara hem de signar digitalment la nostra aplicació. Les signatures segures s'obtenen de les autoritats certificadores però podem crear una signatura pròpia en aquest moment. Fem clic en *Create* i omplin la informació amb el nostre nom i un Password. Fem clic en *Browse* per guardar el certificat al nostre equip i finalment fem clic a *OK* per tornar al quadre anterior.

7. El següent pas és opcional. Podem afegir contingut al nostre paquet que sigui necessari per poder executar la nostra aplicació. Això fa créixer la mida del nostre paquet així que hem de tenir en compte quina quantitat de fitxers afegim.
8. Fem clic a la pestanya *Deployment* i seleccionem l'opció '*Install and launch application on any connected devices*'. Amb aquesta opció el nostre fitxer APK automàticament s'instal·larà al nostre dispositiu Android connectat. Fem clic a *Finish*.

Podem trobar el fitxer APK en la ruta on Flash Builder guarda els projectes per defecte:

- **Windows:** C:\Users\<username>\Adobe Flash Builder 4.6\<project folder>
- **Mac OS:** HD/Users/<username>/Documents/Adobe Flash Builder 4.6/<project folder>

Ara ja pots enregistrar la teva aplicació en la botiga en línia o en un altre medi de distribució.

El procés de empaquetament per la resta de plataformes es basa en els requeriments de cada empresa. Podem seguir aquests dos enllaços on tenim informació per la resta de plataformes.

- Apple iOS:

http://www.adobe.com/devnet/air/articles/packaging-air-apps-ios.html

- BlackBerry Tablet OS:

http://www.adobe.com/devnet/air/articles/packaging-air-apps-blackberry.html

Per aprendre més sobre Air Mobile tenim un parell de tutorials amb el desenvolupament de aplicacions que ens poden anar molt be. El primer d'ells és **Build your first Mobile Flex application – Twitter Trends** (*http://www.adobe.com/devnet/flex/articles/twitter-trends.html*), on incorporem els últims *Trending Topics* des de Twitter. Un altre tutorial que està molt be és

Build an application in an hour (*http://www.adobe.com/devnet/flex/testdrive.html*) amb seqüències de vídeo, codi d'exemple i petits tutorials.

El **Adobe Developer Connection** (*http://www.adobe.com/devnet.html*) ofereix molts recursos que ens ajuden a familiaritzar-nos amb el desenvolupament amb Adobe Mobile i a endinsar-nos en les possibilitats de les seves eines.

3.4 Estudi comparatiu dels casos d'estudi

En primer lloc després de conèixer les tres tecnologies i de treballar amb elles anem a veure els punts fort i febles de cadascuna. Posteriorment les compararem entre elles per veure quina de elles ens pot ser més convenient per el nostre projecte.

PhoneGap

En resum podem veure que una aplicació de PhoneGap és una sèrie de pàgines web que estan emmagatzemades i empaquetades dintre d'una aplicació mòbil, que es pot visualitzar en un navegador web i que té accés a la majoria d'APIs del mòbil. Això converteix en aquesta plataforma en una alternativa molt senzilla per crear aplicacions.

Per treballar amb cada plataforma s'ha d'utilitzar un sistema distint: per iOS és necessari utilitzar Xcode, disposar d'un Mac per poder programar i una plantilla de projecte que proporciona PhoneGap. Per Android és necessari Eclipse (Windows, Mac i Linux) i una altre plantilla de projecte específica. Per BlackBerry no hi ha cap entorn definit, nomes JAVA SDK, BlackBerry SDK i Apache Ant.

Podem veure una taula de les APIs suportades per PhoneGap en cada tipus de dispositiu a *http://phonegap.com/about/features/* , algunes aplicacions fetes a *http://phonegap.com/apps/* i algunes utilitats i eines que podem utilitzar per ajudar-nos al desenvolupament a *http://phonegap.com/tools/.*

Principals avantatges

- Es la solució, juntament amb Rhomobile, que suporta un número més gran de plataformes mòbils, ja que corre dintre d'un navegador web.

A part de iOS i Android, funciona en Palm, Symbian, WebOS, Windows 7, BlackBerry i Bada.

- Es molt senzill desenvolupar amb ell i proporciona gran llibertat als que tenen coneixements HTML i JavaScript.
- Portabilitat de aplicacions web de la nostra a empresa a plataforma mòbil de forma gairebé directa. Només cal adaptar les vistes a la mida d'un dispositiu mòbil.
- Hi ha bona documentació i bastants exemples. La pàgina web ens dona molta informació en un format molt clar.
- Es gratuït amb suport de pagament. Llicencia BSD.

Principals inconvenients

- Requereix diferents instal·lacions i configuracions per cadascuna de les plataformes. Això complica una mica les coses sobretot en el cas de iOS on cal, a més a més, disposar d'un Mac amb Xcode per poder empaquetar les aplicacions.
- L'aplicació no és més que una pàgina web, per tant, l'aspecte dependrà del framework web utilitzat. Necessitarem l'ús de frameworks HTML mòbils com Sencha Touch, jQuery Mobile, Jo, Sproutcore, XUI, jQTouch si volem que sembli una aplicació nativa.
- No arriba al rendiment d'una aplicació nativa, ja que HTML, CSS i JavaScript ha de ser llegit i interpretat per el *engine* del navegador cada vegada que arranca.

Titatium Appcelerator

Com hem vist amb Appcelerator és possible crear aplicacions per Android, iPhone i BlackBerry. Per programar proporciona Titanium Studio, que com hem vist és un IDE basat en Eclipse on podem crear projectes, editar fitxers, etc.. Permet utilitzar com llenguatge de programació a més de JavaScript, PHP, Ruby i Python transformant posteriorment el codi a JavaScript amb els frameworks Phpjs, Skulpt i Ruby.js.

Les aplicacions es programen integrament en JavaScript, creant i colocant a mà tots els controls, utilitzant un llibreria que fa de pont entre la nostra aplicació

JavaScript i els controls del sistema. Això significa que les finestres i la resta de controls visuals són natius (botons, llistes, menus, etc.). Per exemple quan afegim un botó, es crea un botó de sistema i s'afegeix a la vista. Això fa que renderitzi més ràpidament i que la resposta del usuari sigui també més rapida. A diferencia de PhoneGap, en Appcelerator no hi ha DOM, per tant, no es poden utilitzar llibreries com jQuery que serveixen per manipular-lo ja que el context d'execució és pur no encapsulat dintre d'un document HTML.

Aquesta és una de les característiques que sembla més interessant d'Appcelerator, de fet, es ven com un generador d'aplicacions natives ja que quan empaquetem l'aplicació el JavaScript és transformat i compilat. Quan posteriorment executem l'aplicació en el mòbil, el codi s'executa dintre d'un *engine* JavaScript: **JavaScriptCore** en iOS i **Mozilla Rhino** en Android/BlackBerry.

El fet de que JavaScript sigui compilat i que els controls siguin natius, fa que el rendiment sigui el millor possible, arribant pràcticament al nivell de les aplicacions natives.

Com s'ha indicat no suporta DOM i és molt complicat maquetar ja que no existeix un HTML inicial on afegir els controls com per exemple a PhoneGap i s'han de crear els components a mà amb JavaScript. Tot i així es poden afegir components WebKit on introduir codi HTML però solament es recomana fer quan sigui absolutament necessari per mostrar per exemple un text amb format. També és possible crear un únic component Webkit a pantalla complerta i treballar de manera semblant a com ho fa PhoneGap. Podem veure un exemple a *http://developer.appcelerator.com/question/71/what-happened-to-html--css*.

A diferencia de PhoneGap, que solament té una llibreria JavaScript per accedir a les característiques especials del sistema, Appcelerator necessita, a més a més, llibreries per utilitzar els controls natius i la seva disposició en pantalla, per lo que fa el desenvolupament en general més costos.

Per desenvolupar per iOS és necessari tenir un Mac amb Xcode instal·lat. Titanium Studio genera un projecte Xcode amb el JavaScript transformat juntament amb totes les llibreries necessàries. Després és possible llençar el

simulador amb l'aplicació en Xcode sense sortir de Titanium Studio. Una vegada generat el projecte, aquest es pot obrir en Xcode i continuar empaquetant-lo i configurant-lo per la seva distribució (certificats, *provisioning*, logos, *splash screen*, etc.). Des de Xcode no es pot editar el JavaScript per això caldria tornar a Titanium Studio i tornar a generar el projecte Xcode un altre cop.

Per Android tant per provar el simulador com per empaquetar l'aplicació cal tenir el SDK d'Android instal·lat.

Principals avantatges

- Multiplataforma mòbil i també d'escriptori.
- Aspecte i controls natius. Sembla que estem executant una aplicació nativa dintre del nostre dispositiu.
- El millor rendiment de totes les tecnologies analitzades.
- Molt bons exemples, com el **KitchenSink** que hem utilitzat on tenim pràcticament tot el que es pot fer.
- Gratuït amb suport de pagament i llicencia Apache.

Principals inconvenients

- Requereix un Mac i Xcode per empaquetar les aplicacions per iOS.
- Definició dels components visuals i controls a mà, en PhoneGap tenim HTML i en Flex MXML.
- Molta documentació però poc actualitzada i descol·locada dintre del lloc web, tutorials desfasats i poc professionals.
- El IDE i les aplicacions fallen a vegades.
- Les aplicacions d'escriptori es distribueixen sense compilar per tant amb accés al codi HTML, JS, CSS o a les imatges.

Adobe Air Mobile

En resum, Adobe Air Mobile funciona amb Flex i suporta iOS, Android i BlackBerry Tablet. També és possible executar en sistemes d'escriptori Windows, Mac i Linux mitjançant un *runtime*. Les aplicacions d'escriptori no són compatibles directament amb les de mòbil ja que els controls visuals canvien.

Utilitza com a llenguatge ActionScript i disposa d'un IDE oficial anomenat Flash Builder. Aquest és molt potent però de pagament. És possible compilar i empaquetar les aplicacions amb Flex SDK (gratuït) però és moltíssim més fàcil amb Flash Builder. Els controls visuals utilitzats no són originals de cada plataforma sinó específics de Flex. Això és bo per que les aplicacions sempre tindran el mateix aspecte i comportament però dolent ja que no aniran tan suaus com els natius.

Adobe Air requereix que el processador del dispositiu sigui amb arquitectura Arm7, que són els de gama alta avui en dia. Amb iOS això no és necessari. La raó és que en Android les aplicacions van sobre Air Mobile i a iOS, degut a que Apple ho prohibeix, corren sobre CocoaTouch directament.

Es poden depurar les aplicacions en remot, és a dir, es pot instal·lar i executar l'aplicació al teu dispositiu i depurar-la des de l'ordinador. Aquesta és la única tecnologia que no requereix ni d'Android SDK ni de Xcode en Mac per executar i crear aplicacions.

Principals avantatges

- Multiplataforma mòbil, encara que no suporta tantes plataformes com la resta de tecnologies, i també possibilitat de crear aplicacions d'escriptori.
- ActionScript és un llenguatge molt potent que permet l'ús de patrons i estructures de desenvolupament complexes. A més, hi ha una gran comunitat que sap programar en aquest llenguatge.
- Desenvolupament i definició de les vistes molt intuïtives gracies a l'editor visual de MXML amb Flash Builder. En general l'IDE i Flex estan molt avançats i són molt potents. La documentació és realment molt bona, ho hem pogut veure en el transcurs del document i tenim una gran empresa darrera que és Adobe fent un gran esforç amb aquest sistema.
- Les ultimes versions de Flash Builder no requereixen l'ús de Xcode ni d'un ordinador Mac.

- Tenim la possibilitat de fer una depuració remota veient directament el resultat al nostre dispositiu.

Principals inconvenients

- El preu de Flash Builder. Encara que hi ha més eines i es pot utilitzar el SDK gratuït, però com hem vist al apartat anterior aquest IDE és el punt més fort d'aquesta tecnologia.
- No funciona en tots els dispositius Android, només en aquells que tinguin una arquitectura Arm7. Podem veure el llistat a *http://www.adobe.com/flashplatform/certified_devices/.*
- Rendiment regular, la renderització no és molt suau en iOS. Les aplicacions d'escriptori consumeixen molta CPU, sobretot en Mac.
- Aspecte de les interfícies no natives encara que homogènia entre totes les plataformes. Això personalment és una desavantatge per mi però pot ser un avantatge per algú que vulgui que la seva aplicació sigui igual en totes les plataformes.

Una vegada analitzats els punts forts i febles, anem a realitzar les comparacions entre tecnologies. S'han seleccionat 10 punts per tal de realitzar la valoració, aquests punt són els següents:

1. Llenguatges de programació
2. Maquetació i disseny d'aplicacions
3. Entorns de desenvolupament (IDE)
4. Components necessaris per la programació
5. Rendiment de les aplicacions
6. Fiabilitat de les eines
7. Documentació disponible de la tecnologia
8. Seguiment de la comunitat
9. Distribució i venda de aplicacions
10. Preu de desenvolupament i distribució

Amb tota aquesta informació podrem veure en quins punts són més forts unes o les altres i donar una valoració a cadascuna de les tecnologies analitzades.

Llenguatges de programació

Comencem amb la comparació dels llenguatges de programació de cadascuna d'aquestes tecnologies. Tenim principalment ActionScript i JavaScript, encara que Appcelerator ens dona la possibilitat d'escriure amb llenguatges com PHP, Ruby i Python.

Entre ActionScript i JavaScript podríem donam com a guanyador ActionScript encara que els dos tenen el mateix origen en el ECMAScript. JavaScript té una codificació dinàmica dèbil i ActionScript una codificació estàtica forta. Els dos estan orientat a objectes però l'herència en JavaScript està basada en prototipus i la de ActionScript orientada a objectes. En definitiva, JavaScript és molt bon llenguatge, això es demostra en la gran quantitat de llocs que s'utilitza, però a l'hora de desenvolupar una gran aplicació és més difícil estructurar-la amb JavaScript que amb ActionScript degut sobretot a l'herència basada en classes. Potser inicialment les aplicacions mòbils eren molt senzilles i JavaScript funcionava perfectament, però cada cop evolucionen a aplicacions més complexes i necessitem llenguatges com ActionScript, Java o Objective-C que s'utilitza en els entorn natius. En aquest punt donem com a guanyador a Air Mobile davant de Appcelerator i PhoneGap, sempre i quan la nostra aplicació requereixi d'un nivell de estructuració alt, sinó utilitzarem aquell que ens sigui més familiar.

Maquetació i disseny d'aplicacions

En aquest punt tenim un clar perdedor, Appcelerator, ja que no té cap sistema de maquetació ni de disseny de vistes i s'ha de codificar i col·locar els components a mà. PhoneGap és HTML/CSS3 i Flex té MXML i la seva eina Flash Builder amb un editor visual molt potent. Aquí potser és una mica qüestió de gustos i sobretot si estem familiaritzat amb la quantitat de frameworks disponibles per treballar amb HTML5 i CSS3. Clarament MXML està més integrat amb el propi llenguatge i amb el propi IDE, amb l'altre és més complicat ja que són una sèrie de enllaços, formularis, Ajax amb els que manipulem el DOM. Tot això fa que de nou ens decantem per Air Mobile i després per PhoneGap, encara que personalment tinc més experiència amb frameworks com jQuery i no sem fa més complicat.

Entorn de desenvolupament

Després de les dos anteriors resolucions està clar que guanyarà Air Mobile amb el seu Flash Builder. Com ja sabem és un entorn basat en Eclipse molt avançat i potent, Adobe ha invertit molt temps i esforç en crear un IDE realment adaptat per incrementar la velocitat de desenvolupament en Flex, ple d'assistents, editors i amb una depuració molt avançada. Tot això té en contra seu que no és gratuït i al damunt és bastant car, així que és normal que estigui per damunt de la resta. Molt a prop de Flash Builder tenim a Titanium Studio de Appcelerator que també és un Eclipse amb el que podem provar les nostres aplicacions i que detecta errors de codi JavaScript. En últim lloc deixem PhoneGap que només és una mena de plantilla per treballar en Eclipse per Android i Xcode per iOS, però podem crear el nostre HTML i JavaScript amb el nostre editor preferit.

Si els diners són un impediment per nosaltres, la nostra elecció serà Appcelerator sinó Air Mobile serà l'elecció més encertada.

Components per la programació

Appcelerator és l'únic que permet creat controls natius per cada plataforma. PhoneGap usa controls de formularis en HTML, i Flex té el seu propi sistema de components. En aquest cas el guanyador és Appcelerator per permetre crear controls natius amb la seva conseqüent fluïdesa de l'aplicació que no s'arriba a aconseguir amb Flex amb el seu sistema propi o amb PhoneGap i els seus frameworks externs que fan que l'aplicació vagi al gust del desenvolupador.

Rendiment de les aplicacions

Encara que cada cop els dispositius mòbils són més potents tenim que tenir en compte quina de les tecnologies estalvia més recursos o fa un ús més acurat. En aquest cas torna a guanya Appcelerator al utilitzar controls natius i proporcionar més suavitat a la seva interfície. Seguida d'aquest tenim PhoneGap que simula HTML5 i per últim Flex que és interpretat i no té la mateixa suavitat.

Fiabilitat de les eines

Appcelerator i Phonegap són tecnologies joves que encaren necessiten evolucionar per tal d'estar totalment acabades, a vegades fallen, encara que hi ha una comunitat darrera treballant en elles. Flex porta molts anys al mercat i té darrera seu a Adobe treballant dur. En aquest cas com a fiabilitat de les eines té un clar guanyador amb Air Mobile i el sobretot el seu Flash Builder.

Documentació disponible de la tecnologia

Torna a guanyar Air Mobile ja que hi ha una gran documentació, encara que Phonegap té també molt bona documentació i molts exemples per els desenvolupadors. En últim lloc Appcelerator que disposa d'una documentació una mica antiquada i molt dispersa.

Seguiment de la comunitat

Adobe disposa del *Adobe Developer Connection* on la comunitat té un munt de recursos i eines al seu abast. En aquest cas gracies a l'empresa que té darrera i que fa molts anys que està en el mercat, la comunitat més potent per tant és la de Air Mobile.

Distribució i venda de aplicacions

Appcelerator i PhoneGap requereixen tenir un Mac i conèixer Xcode per tal de empaquetar les aplicacions per la posterior distribució. Amb Android no hi ha cap problema es pot utilitzar el SDK que és Java i multiplataforma. Com a punt negatiu per PhoneGap, recordem que les seves distribucions poden ser obertes i es pot modificar el seu codi font, fent perillar la integritat de l'aplicació. Flash Builder permet crear directament les aplicacions per iOS, Android i BlackBerry Tablet OS sense utilitzar ni Xcode ni el SDK Android. Definitivament Flash Builder posa les coses molt senzilles, guanyador un cop més Air Mobile.

Preu de desenvolupament i distribució

Appcelerator i PhoneGap són gratuïts nomes s'ha de pagar un suport de formació. Flash Builder, com hem dit, té un preu al voltant dels 600€, encara que

hi ha SDK lliure però si l'utilitzem Air Mobile perdria la majoria de punts positius que ha tret en aquest anàlisi. En aquest cas guanyen totes menys Air Mobile.

Resolució de la comparació

En general he intentat fer una comparació entre tecnologies bastant subjectiva encara que he estat objectiu en alguns detalls. Donar una resolució definitiva no és possible ja que la selecció d'una o altre tecnologia és molt objectiva.

Per començar veurem cap a quines tecnologies ens volem dirigir, en el cas de fer una aplicació per alguna plataforma que no sigui iOS o Android, ens anirem cap a PhoneGap ja que ens posa a l'abast un munt de plataformes. També utilitzarem PhoneGap si l'aplicació ja la tenim desenvolupada en HTML5 i volem portar-la als dispositius mòbils, ja que la portabilitat serà gairebé directa.

Mirarem les característiques del dispositiu mòbil al que hem vull dirigir i veurem quines necessitats tenim i quina tecnologia pot cobrir-les, per això mirarem el llistat d'APIs que suporta cadascuna. També descartarem Air Mobile si el dispositiu no disposa d'una arquitectura Arm7 o no compleix alguns dels requisits mínims que aquesta tecnologia requereix. Una vegada passat tots aquestos requisits, analitzaríem si volem una aplicació ben dissenyada i robusta amb APIs que no fallaran però amb una pèrdua de suavitat, utilitzant Air Mobile o si és una aplicació més senzilla i vull velocitat i suavitat, utilitzant Appcelerator o PhoneGap. Diferenciaríem entre aquest dos si volem un acabat més natiu a costa de no disposar d'eines de disseny visual, utilitzant Appcelerator, o si volem un acabat personalitzat per totes les plataformes i amb disponibilitat de frameworks per dissenyar les vistes, utilitzant PhoneGap.

Un cas especial a remarcar és si volem desenvolupar per iOS. En aquest cas, si no disposem d'un Mac necessitaríem Air Mobile, ja que és l'únic que permet creat aplicacions iOS sense Mac. Encara que sempre queda l'opció de virtualitzar Mac OS.

La meva opinió personal és que descartaria Air Mobile per el seu alt preu i per requerir terminals d'un model específic amb processador Arm7. També

PhoneGap, si no vull desenvolupar per moltes plataformes, per distribuir les aplicacions amb el codi obert. Per tant, hem quedaria amb Appcelerator ja que és bastant complert i disposa d'un IDE és bastant potent. Si els diners no fossin un problema treballaria amb Air Mobile per tots els punts positius que hem indicat durant l'anàlisi.

3.5 Avantatges i desavantatges de les tecnologies multiplataforma

Analitzem finalment quins avantatges i quins inconvenients ens presenten les tecnologies multiplataforma.

Com ja hem vist, gracies a les tecnologies multiplataforma tenim el principal avantatge de que programant una aplicació una sola vegada podem executar-la en diferents plataformes, amb la conseqüent major extensió de la nostra aplicació i majors beneficis si aquesta és de pagament.

Veiem altres avantatges que tenen aquestes tecnologies respecte les natives:

- Facilitat en la implementació, ja que només utilitzant el nostre llenguatge favorit realitzarem aplicacions per tots els terminals. Això pot estalviar molts diners en petites empreses amb poc personal que no poden invertir el temps necessari en aprendre nous llenguatges per cadascuna de les plataformes on volem intervenir.
- Facilitat en l'actualització de la nostra aplicació, ja que tenim l'aplicació centralitzada i un canvi o actualització pot ser actualitzat a totes les plataformes.
- Control d'accés a la teva aplicació on pots actualitzar sense demanar permís a ningú. Distribuint l'aplicació des de la nostra pàgina web no hi haurà cap botiga que controli si es pot vendre o no, com a l'App Store on controlen una sèrie de requisits que s'han de complir.
- Possibilitat de crear vistes natives o crear les nostres vistes personalitzades, donant-nos la possibilitat de mostrar la nostra aplicació igual en totes les plataformes.

- Poder programar per iOS sense conèixer Objective-C, Xcode i sense disposar d'un Mac obligatòriament.

Un dels principals inconvenients és la velocitat en l'execució, com ja sabem renderitzar HTML o interpretat JavaScript és sense cap dubte més costos que executar una aplicació nativa, cada petició que fem en la nostra aplicació implicarà una recarrega de la pàgina o un accés contra el nostre servidor. Sobretot en els últims temps, una petita espera impacta en l'experiència del usuari. L'aplicació nativa té tots els recursos i processos localment i només accedeix al servidor per obtenir o enviar dades si les necessiten. Per tant, una aplicació multiplataforma no té la fluïdesa ni velocitat que una aplicació nativa encara que les tecnologies multiplataforma que compilen les seves aplicacions com a natives s'acosten bastant al rendiment de les realment natives.

- Un dels inconvenients del que actualment no és salva cap tecnologia multiplataforma són les APIs natives. Encara que ja hem vist que la llista és llarga en cadascuna de les tecnologies analitzades sempre hi ha algunes que no són accessibles des d'una o altre tecnologia o no s'accedeix de la mateixa forma. Per exemple, potser tenim accés a les coordenades del GPS però no en temps real per exemple. Sobretot en iOS tenim denegat l'accés a algunes de les tecnologies del seus dispositius, si la nostra aplicació és senzilla i no fa us de cap d'aquestes APIs, aquest punt no ens importarà però en la resta de aplicacions si, sobretot en jocs on poden entrar en lloc accés a Internet, acceleròmetre o brúixola.
- Possibles dificultats en la distribució si decidim no distribuir la Nostra aplicació mitjançant les botigues de cadascuna de les plataformes. Aquestes donen un fàcil accés als usuaris i gran publicitat, fent descomptes o rankings de les aplicacions més venudes. L'usuari amb una simple pulsació pot instal·lar una aplicació al seu dispositiu i sempre serà avisat de les continues actualitzacions. En canvi, si decidim distribuir per el nostre compte l'aplicació, segurament farem us de campanyes de publicitat si volem que la nostra aplicació arribi al mateix número de públic. A part és molt més senzill entrar, per

exemple, a l'App Store, seleccionar el programa, introduir un password i ja el tenim al dispositiu sense cap falta de registrar-nos a cap lloc ni donar les nostres dades, targetes de crèdit, etc.

Conclusió

Com a conclusió del nostre estudi comparatiu, veiem que tenen molt més avantatges que inconvenients, aquestes tecnologies multiplataforma, però en contra els inconvenients són suficientment potents com per desestimar-les segons el tipus d'aplicació que vulguem desenvolupar. Personalment crec que si l'aplicació és senzilla i no requereix d'un alt nivell de procés, la millor opció és programar amb tecnologies multiplataforma però si volem desenvolupar una aplicació amb un acabat més professional i amb accés a totes les APIs del dispositiu, utilitzarem les tecnologies natives.

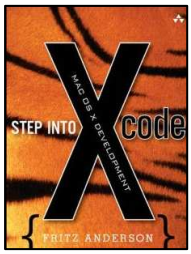
Durant el recorregut d'aquest projecte hem après una mica com hem arribat al moment actual fent una repassada a la historia dels dispositius mòbils per continuar fent un estudi de l'art dels dispositius i les tecnologies disponibles actualment. Hem vist en profunditat tres tecnologies multiplataforma i hem realitzat una comparativa teòrica i practica entre elles i entre elles i les tecnologies natives.

Com a detall dir que s'havia previst realitzar unes proves practiques amb Android natiu però aquestes eren molt semblants a les realitzades amb PhoneGap i s'han desestimat finalment per no trobar-les necessàries per el nostre estudi comparatiu.

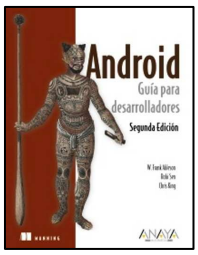
Personalment ha estat un projecte molt enriquidor de coneixements i que obre una porta que m'interessava per endinsar-me professionalment en el desenvolupament d'aplicacions per dispositius mòbils. Ara tinc la capacitat d'estudiar el mercat i veure cap on he de dirigir els meus projectes.

Annexos

Bibliografia



Step into Xcode: Mac OS X Development *Fritz Anderson,* Addison-Wesley Professional (February 9, 2006)



ANDROID: Guia para desarrolladores (2ª ED.) *Frank Ableson, Robi Sen, Chris King*, ANAYA MULTIMEDIA, 2011

Fonts Electròniques

Appcelerator

1. http://www.agilogy.com/creacion-de-una-aplicacion-para-android-con-appcelerator-titanium.html
2. http://nosoloweb.es/siete-dias-con-titanium-dia-0-introduccion/
3. http://www.sg.com.mx/content/view/1171
4. http://apirola.wordpress.com/2010/11/15/appcelerator-titanium/
5. http://androideity.com/2011/09/19/appcelerator-titanium-para-crear-aplicaciones-android/

Adobe Air

6. http://www.adobe.com/devnet/flash-builder/articles/hello-world.html

7. http://help.adobe.com/es_ES/air/build/air_buildingapps.pdf

Phonegap

8. http://www.atsevw.info//index.php?option=com_content&task=view&id=59&Itemid=9&mosmsg=Item+guardado+satisfactoriamente
9. http://www.ellibrogordodepetete.com/p/ejemplos-phonegap-codigo-fuente.html
10. http://wiki.phonegap.com/w/page/30862722/phonegap-android-eclipse-quickstart
11. http://revolucion.mobi/2011/11/23/videotutorial-como-instalar-phonegap-en-eclipse/
12. http://phonegap.com/start#android
13. http://blog.jorgeivanmeza.com/2011/05/primeros-pasos-con-phonegap-para-android/

Informació general

14. http://developer.apple.com/devcenter/ios/index.action
15. http://developer.android.com/sdk/index.html
16. http://us.blackberry.com/developers/
17. http://symbian.nokia.com/
18. http://en.wikipedia.org/wiki/List_of_digital_distribution_platforms_for_mobile_devices
19. http://en.wikipedia.org/wiki/Mobile_application_development

Blogs

20. http://www.idg.es/computerworld/El-desarrollo-de-aplicaciones-moviles-camina-hacia/seccion-actualidad/noticia-104453
21. http://mashable.com/2010/08/11/cross-platform-mobile-development
22. http://www.ticbeat.com/novedades/llega-rhodes-20-nuevo-entorno-desarrollo-multiplataforma-codigo-abierto-smartphones/
23. http://albertovilches.com/
24. http://javiercancela.com/