Mètodes per al desenvolupament d'aplicacions mòbils

Robert Ramírez Vique

PID_00176740

Universitat Oberta de Catalunya
www.uoc.edu

Índex

Introducció

Aquest mòdul se centra en la problemàtica del desenvolupament d'aplicacions i serveis mòbils, i mostra a l'enginyer les oportunitats i dificultats pròpies de l'entorn des d'un punt de vista general.

Tot desenvolupament d'una aplicació o servei té una gran incertesa, i hi ha sistemes per a pal·liar els riscos que hi ha associats. En el cas de les aplicacions mòbils les dificultats, si és possible, són més grans. Ja hi havia algunes dificultats amb els primers desenvolupaments mòbils, com la fragmentació o les xarxes de telefonia i la qualitat del servei d'aquestes xarxes; amb el temps han anat apareixent noves problemàtiques com l'accés a informació d'entorn, o el control de les diferents capacitats dels dispositius. Al mateix temps, les oportunitats de negoci apareixen constantment, i permeten crear des de jocs de gran complexitat, reservats fins ara a consoles de gran potència, fins a aplicacions per a ajudar-nos a moblar la casa.

A causa d'aquesta situació es fa molt difícil poder donar una recepta màgica per al desenvolupament d'aplicacions mòbils, i per tant, es fa imprescindible aprendre i adaptar els mètodes i coneixements adquirits. En aquest mòdul us explicarem situacions, mètodes i estratègies per a minimitzar aquests riscos i implementar les solucions mòbils adequades, i també per a aconseguir el millor rendiment a les capacitats dels dispositius.

En el passat s'ha sentit parlar molt de les aplicacions mòbils, i a pesar que els mòbils ja tenien una gran penetració al mercat, i l'ús que se'n feia com a eina de treball o element de la vida diària era comú, les aplicacions mòbils no s'havien acabat d'enlairar. Les raons són diverses, des de l'intent infructuós d'aconseguir aplicacions executables a tots els dispositius, fins al cost que hi ha associat a aquestes; això ha fet que només algunes aplicacions hagin estat usades àmpliament, com per exemple l'SMS/MMS.

SMS

servei de missatges curts o *short message service*

MMS

servei de missatges multimèdia o *multimedia message system*

Avui dia més del 70% de la població disposa de dispositius mòbils i el nombre de telèfons intel·ligents o *smart phones* no para de créixer (el 90% dels nous dispositius són telèfons intel·ligents segons els estudis de Gartner); aquest és sens dubte el sector que més innovació i expectació està generant i generarà. Actualment es donen molts factors que fan que gairebé ningú no quedi fora de l'ecosistema mòbil; per tant, és un moment perfecte per a conèixer millor els racons d'aquest ecosistema. Alguns d'aquests factors són:

- Millores en les característiques de maquinari (*hardware*) dels dispositius mòbils, gràcies a la inclusió dels fabricants de l'electrònica de consum, que han vist un nínxol de negoci i no volen perdre l'oportunitat.

- Diversitat en les plataformes i dispositius, de manera que es pot cobrir un gran ventall de possibles consumidors. A més, estem vivint aparicions de novetats a un gran ritme, que no sembla decaure. Sens dubte hi ha un paper especial per a algunes aparicions, com són les de l'iOS (iPhone, iTouch i iPad) i l'Android, que han donat una perspectiva diferent.

- Familiarització de l'ús dels dispositius mòbils (telèfons intel·ligents, ordinadors de tauleta o *tablet PC*, televisors, etc.) en gran quantitat d'escenes quotidianes, que han entrat en molts mercats. El que abans semblava només reservat a les escenes de ciència-ficció avui dia està a l'abast de la mà.

- Popularització de les tarifes d'Internet mòbil per a aconseguir més quota de mercat, i sense parar de créixer.

- Aparició de gran quantitat de noves aplicacions dia a dia, disponibles per al gran públic gràcies les botigues d'aplicacions o mercats web (*market places*).

- Les noves formes o facilitats de monetització de les aplicacions, que fan més atractiu per a les empreses plantejar-se desenvolupar aplicacions per a aquest tipus de dispositius.

- L'aparició de les xarxes socials, el propòsit de les quals es veu complementat i potenciat amb les aplicacions mòbils.

Sens dubte això ens obliga, com a professionals del sector, a conèixer els reptes i possibilitats d'aquest entorn.

Durant el mòdul veurem inicialment una introducció a la situació del desenvolupament d'aplicacions mòbils, veient el que ho fa peculiar i ho diferencia d'altres processos de construcció d'aplicacions.

Després entrarem en més detall en un mètode de desenvolupament d'aplicacions mòbils, i intentarem exposar les millors pràctiques en cadascuna de les fases del desenvolupament.

Finalment farem un repàs a les opcions de negoci possibles dins del món mòbil.

Objectius

L'objectiu d'aquest mòdul és proporcionar un coneixement ampli i variat de les alternatives per al desenvolupament d'aplicacions mòbils. En concret, amb l'estudi d'aquest mòdul es pretén que l'estudiant aconsegueixi els objectius següents:

1. Aprendre la problemàtica dels desenvolupaments d'aplicacions per a mòbils.

2. Veure les restriccions i les possibilitats de les aplicacions esmentades.

3. Conèixer un mètode de desenvolupament, posant èmfasi en la problemàtica de les aplicacions per a dispositius mòbils.

4. Conèixer les eines necessàries per a aplicar aquest mètode a les noves tecnologies emergents.

5. Ser capaç de poder afrontar un projecte relacionat amb el desenvolupament d'aplicacions mòbils, sabent què cal fer en cadascuna de les fases, i tenint eines per a afrontar-lo amb garanties.

1. Ecosistema d'aplicacions mòbils

L'ecosistema mòbil es refereix al conjunt d'actors necessaris per a poder tenir els dispositius mòbils i finalment les aplicacions per a aquests dispositius. En concret en aquest ecosistema s'inclouen des dels operadors de telecomunicacions fins a tots els elements de programari que intervenen en l'execució de l'aplicació, passant pels fabricants de maquinari.

Totes les aplicacions s'executen dins d'un ecosistema; per tant, per a aconseguir un desenvolupament satisfactori és ideal conèixer-lo. Hi ha diversos factors que afecten l'ecosistema, com és la infraestructura de l'aplicació, el sistema operatiu, els mètodes d'entrada d'informació, els usuaris mateixos o els canals de distribució de l'aplicació.

Per exemple, en el cas de les aplicacions web un punt característic és que s'hi ha d'accedir utilitzant un navegador, cosa que condiciona moltes altres coses, i per a poder fer una bona aplicació web sens dubte s'ha de conèixer aquesta informació. En el cas de les aplicacions de taula tenim més control, però també tenim més diversitat a causa dels diferents sistemes operatius disponibles; succeeix el mateix amb els servidors amb les diferents xarxes o protocols que han de suportar.

En el cas de les aplicacions mòbils l'ecosistema és encara més heterogeni que en la resta de desenvolupaments; es poden executar en diferent tipus de dispositiu, sia un mòbil antic o bé un de nou, un telèfon intel·ligent o un ordinador de tauleta, o fins i tot coses menys evidents, com un televisor o una targeta intel·ligent. Aquests dispositius solen estar connectats a Internet mitjançant la connectivitat d'un operador per mitjà d'una xarxa gràcies a un contracte. Això finalment compon, com es veu a la taula següent, un ecosistema amb molts actors que cal tenir en compte per al desenvolupament d'aplicacions mòbils.

Targetes intel·ligents

Les targetes intel·ligents o *smart cards* són targetes amb un circuit integrat de mida de butxaca en què es pot programar algun tipus de lògica. Un exemple són les targetes de crèdit amb microxip.

Ecosistema dels dispositius mòbils

Les diferents capes d'actors que influeixen fins a aconseguir un servei com pot ser SMS o Internet mòbil.

Serveis
Aplicacions
Entorn de treball (*framework*) d'aplicacions
Sistemes operatius
Plataformes
Dispositius
Xarxes (GPRS, 3G, etc.)
Operadors

Sens dubte, l'ecosistema d'una aplicació per a dispositius mòbils és una cosa que no ens plantegem en un primer moment, però és una de les dificultats més grans del desenvolupament, que acaba causant entre altres coses més fragmentació de l'aplicació. Això finalment es tradueix en el fet que el desenvolupador ha de tenir en compte molts factors perquè l'aplicació funcioni tal com vol.

Dins d'aquest ecosistema tenim molta informació, que pot ser molt útil per a les nostres aplicacions. Des de la informació de la xarxa de dades actual per a adaptar els continguts, fins a la informació del dispositiu mateix, com pot ser la posició geogràfica. També trobem capacitats que en altres entorns no trobaríem, com per exemple la capacitat de localitzar altres dispositius en moviment, de donar informació del nostre entorn (localització, orientació, pressió atmosfèrica), aconseguir informació de l'usuari (contactes, calendari, etc.), o fins i tot mitjans de pagament molt més directes (per mitjà de l'operador, o del dispositiu mateix).

En aquest apartat veurem els reptes i oportunitats que ens ofereix aquest ecosistema mòbil.

1.1. Fragmentació

Un dels principis bàsics a l'hora de desenvolupar aplicacions és intentar tenir el codi més simple possible, per evitar complexitat, possibles errors i facilitar-ne el manteniment. Aquesta situació s'ha vist molt dificultada a causa de la fragmentació que hi ha en els entorns d'aplicacions més coneguts.

> La fragmentació és una situació, o condicionants de la situació, que no permeten compartir una mateixa aplicació entre diferents ecosistemes. És a dir, no es pot compartir l'aplicació sense necessitat de fer adaptacions en els ecosistemes.

Aquesta fragmentació pot venir per molts factors, que finalment provoquen diversitat i entropia dins de les aplicacions que vulguem crear. Aquestes raons per a la fragmentació poden ser:

- **Maquinari diferent**. Dispositius amb diferències en qualsevol component: mida o densitat de la pantalla, teclat, sensors, capacitat de procés, etc.

- **Programari diferent**
  - **Plataforma diferent**. Tant si és el cas d'una plataforma, com el d'un entorn de treball o el de sistemes operatius (o versions de qualsevol), poden generar fragmentació de les aplicacions.

  - **Diferències en les implementacions**. Per exemple, diferències en la implementació de l'estàndard, o bé errors coneguts de versions concretes.

- **Variacions de les funcionalitats**. Perquè és una versió amb menys privilegis (versió de pagament i versió gratuïta), o segons els papers dels usuaris mateixos de l'aplicació.

- **Preferències d'usuari**. La més habitual són les localitzacions de l'aplicació (idioma, orientació del text, etc.).

- **Diversitat de l'entorn**. Derivat de la infraestructura, com poden ser els operadors i les API, problemes de tallafocs, limitacions de les xarxes, itinerància, etc.

Itinerància

Itinerància o *roaming* és el concepte relacionat amb la capacitat de moure's d'un dispositiu d'una zona de cobertura a l'altra.

Aquesta fragmentació pot afectar tot el projecte de desenvolupament, des del model de negoci fins al desplegament, passant per la implementació i les proves. És imprescindible tractar-la molt seriosament.

Si no es tracta correctament, aquesta fragmentació pot causar molts problemes, com per exemple:

- Reduir la qualitat del producte. A causa de la complexitat més gran de les solucions fragmentades es poden generar més errors.

- Limitar el nombre de dispositius suportats. Per a evitar aquesta complexitat, es pot decidir suportar un nombre més petit de dispositius, amb possibilitats d'ampliacions en un futur.

- Allargar qualsevol fase del projecte, des de les fases inicials fins a la implementació, i sens dubte el manteniment. Aquesta dilatació en el temps significarà sobrecost i el possible fracàs del projecte.

- Grans costos associats a les proves sobre dispositius reals.

Sens dubte, la fragmentació ha estat i segurament serà la dificultat i risc més gran del desenvolupament d'aplicacions mòbils.

La fragmentació pot ser de diferents graus. No és el mateix atacar la fragmentació d'una aplicació que s'ha d'executar sobre un televisor i sobre un telèfon mòbil, que la fragmentació d'una aplicació sobre dues versions de la mateixa plataforma. Per això hi ha diferents estratègies per a combatre-la, i cadascuna té un sentit segons el cas concret.

1.1.1. Un desenvolupament per a cada escenari

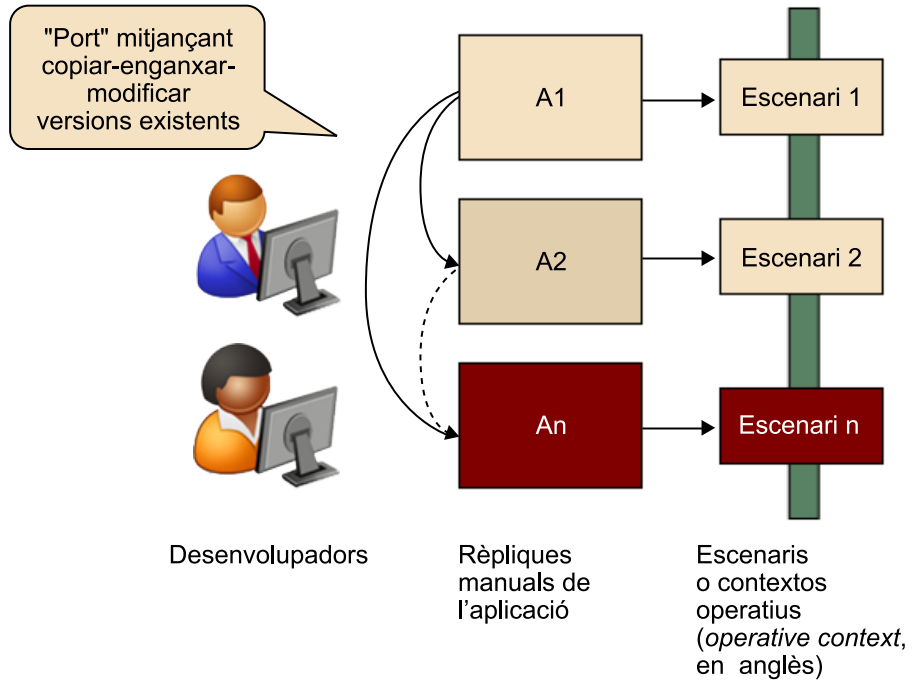
Un escenari és un cas de fragmentació a causa de qualsevol de les possibles causes de fragmentació.

És a dir, es fa tot un desenvolupament per a cada fragmentació que ens puguem trobar, sense compartir res. Això sol ser útil en casos en els quals els escenaris són molt diferents.

El procés d'adaptar l'aplicació a un nou escenari s'anomena *portar l'aplicació*.

Aquesta estratègia és la més costosa de totes, ja que no es pot aprofitar sense adaptar res o gairebé res del codi fet en altres escenaris. Com a contrapartida, podem aprofitar al màxim la capacitat del dispositiu i del llenguatge, i també aprofitar les últimes novetats.

Fragmentació d'aplicacions a causa de factors de diversitat



1.1.2. Part comuna i derivacions

La **derivació** és l'estratègia més habitual, en què tenim un informe de la nostra aplicació comuna a tots els nostres escenaris i per a cadascun podem definir la part específica que correspon.

En aquesta estratègia hi ha diverses variants, segons com es faci la derivació en els diferents escenaris. Aquestes derivacions el que fan són els canvis específics de cada escenari per a assemblar el funcionament en tots. Tenim, doncs, les opcions següents:

- **Derivació selectiva**. Les modificacions necessàries estan localitzades en uns elements concrets (tant si són classes del codi com fitxers de marcatge, estils o altres recursos), i hi ha un sistema o eina que genera les diferents versions agafant aquests elements i empaquetant-los per a cada escenari.

- **Derivació usant metaprogramació**. Es tracta de programar sabent que s'executarà en diversos escenaris; per a aconseguir els comportaments diferents hi ha diverses opcions:
  - Per mitjà de la injecció d'objectes o recursos (imatges, fitxers XML, etc.) en el codi, de manera que la nostra aplicació deixa aquests objectes buits, i en temps d'execució s'omplen amb els objectes específics de cada escenari. Aquesta estratègia es basa en el patró de disseny inversió de control o *inversion of control*.

  - Utilitzant preprocessadors, que s'encarreguen, abans d'executar, de canviar o bé ampliar el nostre codi per a adaptar-lo als diferents escenaris.

- **Generació automàtica**. El programari s'ha d'escriure d'una manera específica i només una vegada. *A posteriori* hi ha un procés que genera automàticament les aplicacions correctes per a cada escenari, normalment transformant la nostra aplicació a codi particular de cada escenari. En aquest punt hi ha més variants.

Aquesta estratègia pot ser menys costosa que l'anterior, ja que es pot aprofitar part del desenvolupament de les alternatives, i reduir així costos d'implementació. Però sovint requereix conèixer tant els llenguatges o entorns dels diferents dispositius com les eines, o fins i tot llenguatges, que ens serveixen per a dur a terme les adaptacions. En aquest cas se solen aconseguir aplicacions que aprofiten en gran manera el potencial dels dispositius, encara que segons l'estratègia escollida es pot donar el cas en el qual perdem control sobre el codi generat, i per tant, potència de desenvolupament.

1.1.3. Adaptació única

En aquest cas podem aconseguir una versió que funciona en tots els casos, i no és necessari fer més canvis. Hi ha diverses maneres d'afrontar aquesta estratègia:

- **Denominador mínim comú**. Es tracta d'aconseguir una aplicació, reduint els punts de fragmentació, de manera que no hi hagi la necessitat d'adaptar l'aplicació.

- **Tots en un**.
  - L'aplicació és capaç de conèixer la informació necessària per a poder adaptar-se a tots els dispositius. Per exemple, per a evitar el problema de diferents pantalles es genera una aplicació amb finestres autoescalables.

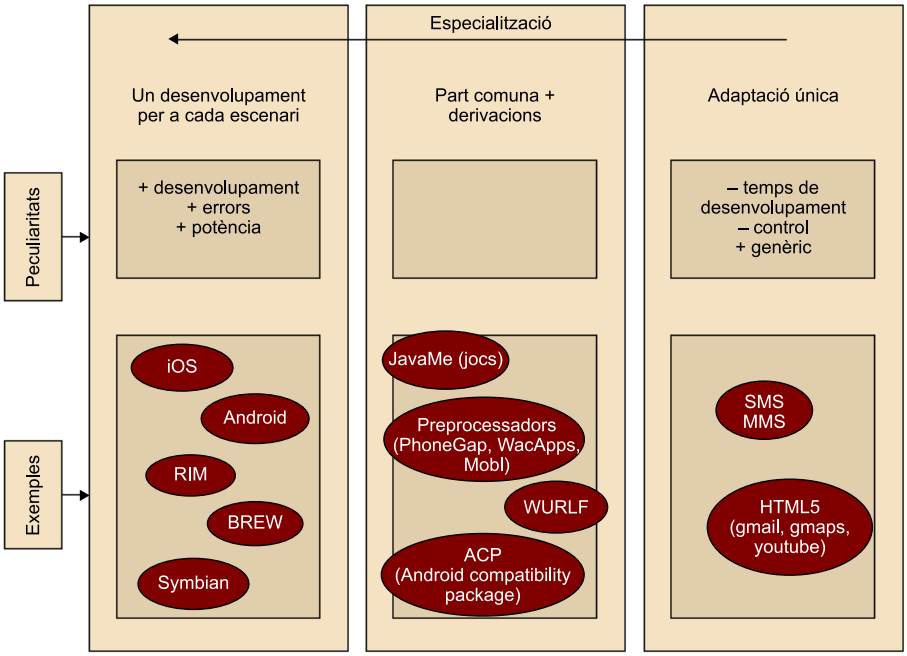
  - Són els dispositius els que s'adapten, és a dir, el programari s'escriu de manera abstracta i en el moment d'arribar al problema de la fragmentació es passa el testimoni al dispositiu, que sap com tractar-lo. Un exemple pot ser l'accés als contactes per mitjà d'API[1] abstractes, o a les trucades de telèfons.

[(1)]interfície de programa d'aplicació o *application program interface*

Aquesta estratègia és la més econòmica, quant a desenvolupaments diferents i coneixements requerits. Com s'intueix, també és la que es queda a la capa més superficial del potencial dels dispositius.

Malgrat això, no s'ha de pensar que no s'aconsegueixen coses potents, ja que amb aquest tipus d'aplicacions es poden fer una gran varietat d'aplicacions, des d'aplicacions molt simples i generals com pot ser una aplicació basada en SMS, fins a aplicacions com el Google Places que coneix la nostra localització, o altres que poden treballar amb dades sense connexió (mode fora de línia).

Categorització de tipus d'adaptacions als problemes de fragmentació



1.2. Context

El context es defineix com les informacions conjuntes de la situació actual, l'usuari, la informació del dispositiu i la informació d'altres aplicacions en un moment concret del temps.

Les aplicacions mòbils tenen la possibilitat d'aprofitar molt més el context en què s'estan executant, sobretot si les comparem amb aplicacions tradicionals. Això és a causa de diferents factors, començant per les noves capacitats dels dispositius i acabant per les capacitats que pot aportar l'entorn en què ens trobem o en el moment en què usem l'aplicació, passant per la capacitat d'accedir a la informació que el dispositiu mateix té de l'usuari, incorporant les capacitats o relacions socials que són implícites sobre l'usuari.

1.2.1. Capacitats dels dispositius

Els nous dispositius ens aporten molta informació que hi ha en el nostre entorn; per exemple, la més clara i coneguda és la posició geogràfica actual, que ens permet fer aplicacions basades en la localització (LBS). A més, hi ha altres informacions, com la informació d'altres sensors: d'orientació, de pressió, de llum, etc. El fet de poder gravar imatges, vídeos i àudio també ens aporta més informació de context, com per exemple les aplicacions que reaccionen a la parla o bé les aplicacions de realitat augmentada, que són exemples d'aplicacions que aprofiten aquest tipus de context. O coses més òbvies, com és saber l'hora actual, l'idioma, o la zona horària, ajuden a fer aplicacions més contextualitzades. Un altre punt que ha fet millorar la situació són sens dubte les millors comunicacions sense fils que han aparegut, des del 3G o UMTS fins a les millores de Bluetooth o nous estàndards de Wi-Fi.

Aplicacions basades en la localització

Les aplicacions basades en la localització o *location based service* (LBS) són serveis que intenten donar un valor afegit gràcies al coneixement de la ubicació geogràfica de l'usuari.

Realitat augmentada

Realitat augmentada (RA) és el terme que s'usa per a definir una visió directa o indirecta d'un entorn físic del món real, els elements del qual es combinen amb elements virtuals per a la creació d'una realitat mixta a temps real.

Moltes de les aplicacions aprofiten diverses capacitats; així, per exemple, una aplicació de realitat augmentada n'està aprofitant diverses:

- GPS, per a conèixer la posició geogràfica i així saber què s'ha de mostrar.
- Brúixola, per a conèixer l'orientació actual.
- Acceleròmetre, per a saber quina és l'orientació exacta del nostre dispositiu i sobreposar les capes.
- Càmera, per a poder captar el nostre voltant i així ampliar la informació. Fins i tot de vegades hi ha diverses càmeres.

- Connexió a Internet per a poder obtenir la informació per a ampliar la nostra realitat. Aquesta connectivitat pot venir per mitjà d'Internet mòbil o Wi-Fi, o d'altres.

- Capacitat de processament gràfic molt millorada, amb xips d'acceleració gràfica potents.

A més, estan apareixent nous protocols o capacitats que ajudaran a millorar les aplicacions mòbils, com és el cas d'NFC. NFC, a més de permetre fer pagaments amb el mòbil, permetrà comunicar informació d'entorn, com es pot fer ara amb RFID, per a arribar a aplicacions mòbils; per exemple, una visita guiada a un museu amb informació constant del que estem veient.

Sens dubte, el salt en els últims anys ha estat molt gran respecte als mòbils que "únicament" eren capaços de trucar i enviar missatges. Ara tenim a l'abast de la mà moltes funcions que aporten possibles aplicacions per a conèixer molt millor el nostre entorn.

1.2.2. Ubiqüitat

La ubiqüitat (o omnipresència) es defineix com la capacitat d'accedir a tota la informació o serveis que necessita l'usuari en qualsevol moment i circumstància, per mitjà del dispositiu que tinguem actualment.

Amb la situació actual de les aplicacions mòbils, estem molt a prop d'aconseguir aquesta ubiqüitat, ja que les aplicacions mòbils poden anar amb l'usuari contínuament. De fet, s'han convertit en un accessori diari, tant a escala professional com personal. Gràcies a les noves capacitats tant de connectivitat com de procés, es pot accedir gairebé a tots els serveis disponibles a Internet. L'accés a aquests serveis pot arribar a ser més pràctic que des d'un ordinador personal, i en això se centra aprofitar la ubiqüitat.

En aquest sentit, en qualsevol moment podem rebre informació útil per a nosaltres en el moment actual, com per exemple rebre un correu electrònic o un missatge instantani, però també pot ser informació sobre el lloc pel qual estem passant actualment, o recordatoris relacionats amb el lloc o la persona amb qui som actualment.

També podem conèixer estats del temps, trànsit i molts altres paràmetres en el moment i lloc que ens interessa. O si parlem d'aplicacions per a un altre tipus de dispositius, com un televisor, també podem tenir informació contextualitzada, com poden ser jocs, concursos o simplement comentaris relacionats amb el programa que estiguem veient en aquell moment.

Per tant, amb les aplicacions per a dispositius mòbils tenim una sèrie de beneficis relacionats amb la ubiqüitat:

- El dispositiu mòbil és el primer mitjà de comunicació massiu real, ja que és capaç d'arribar a gairebé tots els usuaris i a tota hora.

Mitjà massiu

Un mitjà massiu o mitjà de comunicació de massa és el mitjà de comunicació rebut simultàniament per una gran audiència.

- El primer mitjà de comunicació sempre encès. És capaç de captar i enviar informació encara que estigui apagat (apagat en el sentit de l'usuari, és a dir, tancat però no totalment apagat).

- El primer mitjà que és sempre amb l'usuari.

- El mitjà massiu que té incorporat un sistema de pagament, en aquest cas per mitjà de l'operador.

1.2.3. Context social

Un altre punt important que cal tenir en compte, i que sens dubte ajuda molt que una aplicació triomfi al sector actualment, és el component social que té. Això significa la possibilitat d'interactuar per mitjà de les nostres aplicacions amb els nostres amics, família i altres coneguts (o fins i tot desconeguts).

Exemples d'interacció

Els següents són alguns exemples d'interacció:

- Compartir la nostra puntuació en un joc.
- Veure usuaris que estan interactuant actualment amb l'aplicació o bé amb el tema que ens interessa actualment; per exemple, mentre mirem una sèrie de televisió poder comentar-la.
- Publicar en qualsevol moment el que s'està fent actualment a la nostra xarxa social i veure el que fan els nostres amics.
- Rebre avisos de quan els nostres amics són a prop, arriben a un lloc concret o fan alguna acció destacable.

Òbviament el gran creixement de les xarxes socials actuals és el brou de cultiu ideal per a aquest tipus d'aplicacions socials mòbils (MoSoSo[2]), ja que actualment hi ha una gran penetració de les xarxes a Internet a escala mundial. Fins i tot hi ha casos com l'Android, en què per a treure més partit del nostre dispositiu és necessari accedir amb unes credencials de Google i la seva capa social, cosa que automàticament ens connecta amb tots els nostres amics d'aquest servei i permet treure el màxim profit de moltes de les nostres aplicacions.

(2) *mobile social software*

Les aplicacions socials tenen gran projecció al mòbil per la capacitat d'omnipresència (o ubiqüitat) d'aquestes aplicacions, és a dir, que ens acompanyen en el moment adequat. Per exemple, mentre es fan les fotos es poden pujar a la nostra xarxa social, o quan accedim a informació de contactes al nostre voltant, tenim informació dels nostres contactes.

1.2.4. Costos

Sens dubte que estigui contextualitzada és un gran punt a favor per a la nostra aplicació, però, com tota aportació, té la seva contrapartida, i en aquest cas ve en forma de costos. Així per a poder tenir aquest context es pot necessitar:

- Accés a Internet per mitjà de xarxes Wi-Fi o MWWAN.
- Proximitat d'altres dispositius per a compartir informació.
- Estar constantment connectat a les xarxes socials. Això requereix disposar de comptes d'aquestes xarxes socials, i tenir activat l'accés, amb els possibles problemes de privacitat.
- Connexions a altres xarxes sense fils: Bluetooth, GPS, NFC, etc.
- Grans capacitats de procés als nostres telèfons per a fer les accions, sia capacitat de procés intern o bé per mitjà d'accessoris.
- A causa d'innovacions i canvis, hem de pagar el cost d'actualitzar la nostra aplicació. És a dir, durant el llançament de la nostra aplicació o després, poden aparèixer noves fonts de fragmentació de maquinari o programari que provoquin que la nostra aplicació s'hagi d'actualitzar. Això succeeix dins de tots els desenvolupaments però especialment en el desenvolupament d'aplicacions per a dispositius mòbils a causa de la gran velocitat del canvi en aquest sector.

Els costos finals es tradueixen en el següent:

- **Limitació de la vida de les bateries**. Això significa que en utilitzar moltes d'aquestes capacitats, per exemple el GPS, fem que el nostre terminal perdi autonomia, ja que es necessita destinar energia a aquests perifèrics, i succeeix el mateix amb la resta de capacitats. En la matèria que ens pertoca, el desenvolupament d'aplicacions, hem de tenir això molt present a l'hora de dissenyar i escriure les nostres aplicacions, ja que en pot afectar molt el rendiment.
- **Vulneració de la privacitat**. Cada dia més els nostres dispositius mòbils contenen informació més personal i confidencial i requereixen accés a aquesta informació per a poder treure'n més partit i aconseguir integrar-se en el context. Per això sempre que fem una aplicació que requereix accés a

informació o accedeix a dades d'altres fonts, com les xarxes socials, ha de tenir el permís explícit de l'usuari, i aquest permís s'ha de poder revocar.

- **Necessitats de maquinari**. Per exemple, la necessitat de més velocitat de transmissió: si ens trobem en una zona de poca cobertura per a la nostra xarxa de transmissió de dades, pot significar que les nostres aplicacions no funcionin correctament.

- **Necessitats d'inversions no previstes a causa de novetats del mercat**. Si apareix una font de fragmentació nova, com per exemple un nou dispositiu, s'ha d'invertir a donar suport a aquest nou dispositiu. Aquesta inversió pot voler dir des de no haver de fer res, fins a fer un desenvolupament nou.

1.2.5. Conclusions

Sens dubte un punt diferenciador de qualsevol aplicació mòbil ha de ser l'ús del context, d'una o diverses maneres, ja que això constitueix un valor diferenciador respecte a les aplicacions d'altres suports.

2. Característiques d'un projecte de desenvolupament per a dispositius mòbils

Com hem vist anteriorment, els desenvolupaments d'aplicacions sobre dispositius mòbils tenen grans oportunitats i possibilitats, però també algunes dificultats afegides que poden arribar a ser un risc per a aconseguir que els projectes siguin un èxit.

Per tant, a l'hora d'afrontar un projecte de desenvolupament de programari per a dispositius mòbils, o bé projectes en què una part estigui orientada a dispositius mòbils, haurem de tenir un mètode que, a més de suportar la problemàtica habitual del desenvolupament de programari, s'encarregui de donar solucions i minimitzar riscos per al cas concret del desenvolupament de programari mòbil.

En aquest apartat veurem una visió general del tipus d'aplicacions per a dispositius mòbils que ens podem trobar, comparant-les per a poder triar la millor alternativa quan ens enfrontem a un projecte. Aquest punt és molt important perquè condiciona totes les fases del desenvolupament; de fet, es podria dir que segons el tipus d'aplicació els desenvolupaments són molt diferents entre si.

Després de veure els tipus d'aplicacions que hi ha, és necessari conèixer les opcions disponibles per a desenvolupar aquesta aplicació. En aquest punt farem un repàs a les diferents estratègies, la qual cosa ens ajudarà a entendre millor el món del desenvolupament per a dispositius mòbils.

El següent que veurem en aquest apartat són els mètodes de desenvolupament existents aplicats al desenvolupament d'aplicacions mòbils, i veurem les peculiaritats generals del desenvolupament mòbil i com es poden utilitzar aquests mètodes per a solucionar les problemàtiques.

A continuació veurem les fases del desenvolupament, posant èmfasi en les diferències de les fases d'un desenvolupament d'aplicació normal respecte al desenvolupament per a dispositius mòbils.

Per a poder abordar els projectes, en aquest apartat veurem una introducció a les estratègies que hi ha en el desenvolupament d'aplicacions mòbils, per a conèixer així les diferents alternatives.

Després veurem amb més detalls un mètode de desenvolupament i les fases que té, posant el focus especialment en les peculiaritats del desenvolupament d'aplicacions mòbils.

2.1. Tipus d'aplicacions

Hi ha molts tipus d'aplicacions, ja que el tipus de dispositiu que tenim en ment pot ser molt versàtil. A més, hi ha aplicacions en què segurament no pensem, com poden ser les aplicacions per a dispositius especials, com un televisor o una consola, ja que, malgrat que no són dispositius mòbils, solen ser programats amb les mateixes tecnologies i limitacions.

Les aplicacions es poden dividir segons la utilitat que els vulguem donar, o bé segons les necessitats de dispositiu i de la complexitat de l'aplicació mateixa.

Quan ens enfrontem a un projecte per a dispositius mòbils, és important conèixer les opcions que tenim, els punts forts i febles d'aquests dispositius. Amb aquesta informació, podrem triar millor l'aplicació que farem.

A continuació mostrem una divisió segons el tipus de desenvolupament.

2.1.1. Aplicacions bàsiques

Les aplicacions bàsiques són aplicacions d'interacció bàsica amb el dispositiu, que no requereixen res més que enviar o rebre informació puntual de l'usuari.

Les aplicacions bàsiques es poden gestionar simplement amb la tramesa de missatges de text (SMS o MMS). Aquest tipus d'aplicacions existeixen des de fa molt temps i, encara que han tingut gran acceptació i ús, avui dia estan començant a deixar pas a aplicacions més complexes.

Les aplicacions bàsiques tenen diversos avantatges:

- Simplicitat.
- Facilitat de monetització.
- Gran quantitat d'usuaris potencials.

Per contra, tenen alguns desavantatges:

- Poca o gairebé nul·la capacitat de processament del context.
- Molt baixa complexitat de les aplicacions fetes.
- Limitacions imposades per la tecnologia sobre els dissenys de les aplicacions: 160 caràcters de text.

Exemple d'aplicació bàsica

Un exemple d'aplicació és una aplicació per a conèixer quin és el temps d'arribada d'un autobús, enviant un SMS a un número concret i rebent una resposta.

2.1.2. Webs mòbils

Els **webs mòbils** són aquells webs que ja existeixen actualment i que són adaptats específicament per a ser visualitzats als dispositius mòbils, adaptant l'estructura de la informació a les capacitats del dispositiu, de manera que no saturin els usuaris i es puguin usar correctament des d'aquests dispositius.

Llenguatge de marques

Els llenguatges de marcatge o llenguatges de marques (*markup language*) són els llenguatges que estructuren la informació per mitjà de marques o etiquetes (*tags*).

Depenent dels dispositius als quals vulguem arribar s'haurà d'adoptar un llenguatge de marcatge o un altre, ja que hi ha dispositius que només suporten un tipus de marques. En els dispositius de tipus telèfon intel·ligent actuals, es pot arribar a poder veure un web malgrat no estar adaptat a dispositius mòbils, però en aquest cas no estaríem parlant d'una aplicació per a dispositius mòbils, sinó d'una capacitat del dispositiu.

Aquest tipus d'aplicacions són aplicacions bàsiques, generalment sense ús d'objectes dinàmics com JavaScript; per tant, no tenen tot el potencial d'un navegador web de taula. S'utilitzen estàndards web com per exemple XHTML, WML, XHTML-MP o C-HTML, i en general versions prèvies a la nova versió de l'estàndard HTML: HTML5. Estan pensades per a donar suport a dispositius de gamma mitjana i baixa.

Els avantatges de les webs mòbils són els següents:

- **Fàcil implementació, verificació i actualització**. Fins i tot es pot fer gran part del desenvolupament sense necessitat d'utilitzar dispositius mòbils ni emuladors, fins a arribar a les fases finals del desenvolupament.
- **Llenguatge conegut i estàndard**. Els llenguatges de marques són molt coneguts avui dia per la majoria dels desenvolupadors, i en la majoria dels casos es tracta de subconjunts de llenguatges coneguts.

WML

Gairebé tots els dispositius mòbils d'avui dia tenen suport per a algun tipus de llenguatge de marcatge; inicialment era WML (llenguatge d'etiquetatge sense fils o *wireless markup language*), però hi ha altres llenguatges de marcatge que estan suportats.

- Poden suportar múltiples dispositius amb un únic codi font. Si es vol suportar fragmentació entre dispositius s'han d'utilitzar tècniques especials com el WURLF.

WURLF

WURLF és un repositori d'informació per a poder identificar a partir de la metainformació d'una petició web d'un dispositiu mòbil les capacitats i limitacions que té. Per a saber més sobre WURLF podeu visitar la pàgina web següent: http://wurfl.sourceforge.net/

Els inconvenients dels webs mòbils són diversos:

- És difícil suportar múltiples dispositius, i també aconseguir la mateixa experiència d'usuari amb diversos tipus de navegadors.

- Ofereixen grans limitacions a l'hora de fer programes, tant de procés com d'accés a la informació del dispositiu i de l'usuari. Per tant, és difícil aconseguir aplicacions contextualitzades.

- En molts casos estan pensats per a ser visualitzats amb connexions lentes, però aquestes connexions poden ser massa lentes, i provocar una experiència d'usuari molt baixa.

- Actualment la majoria de dispositius nous estan incorporant estàndards més nous (com HTML5), per la qual cosa no s'està treballant a millorar aquests estàndards.

- El nombre de dispositius que només poden veure una pàgina web amb aquest tipus de llenguatges de marques està disminuint.

Exemples de webs mòbils

Qualsevol web públic amb informació per al contribuent és un bon exemple de web mòbil. Per exemple: wap.bcn.cat.

2.1.3. Aplicacions web sobre mòbils

Les aplicacions web sobre mòbils són aplicacions que no necessiten ser instal·lades al dispositiu per a poder executar-les. Estan basades en tecnologies HTML, CSS i JavaScript i s'executen en un navegador. A diferència dels webs mòbils, l'objectiu bàsic de les quals és mostrar informació, aquestes aplicacions tenen com a objectiu interactuar amb el dispositiu i l'usuari, i poden treure més partit de la contextualització.

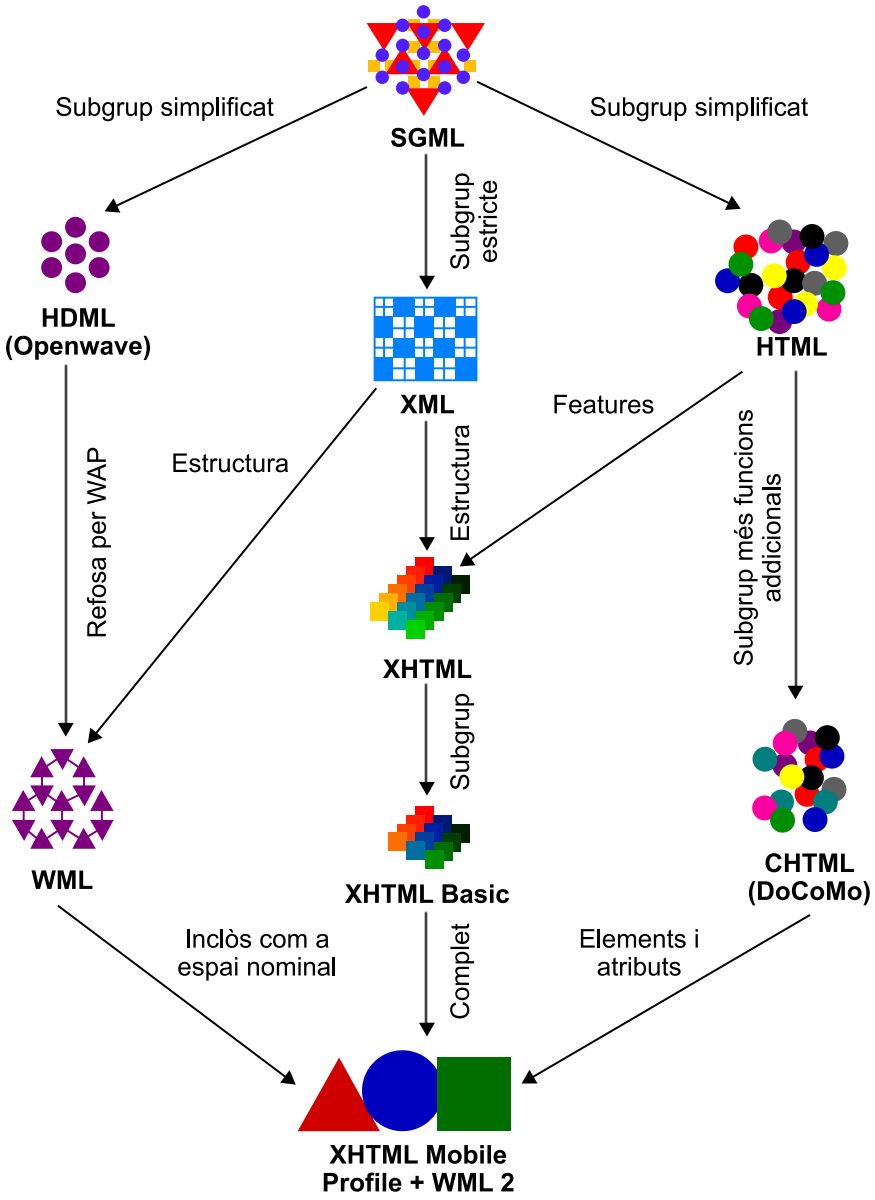
Es tracta d'aplicacions especialment dissenyades per a treballar en el mòbil, i que intenten aprofitar al màxim les possibilitats que tenen. De vegades accedeixen a dades del context: posició geogràfica, dades desades, etc.

Aquest tipus d'aplicacions han existit des de fa temps, però amb l'aparició de navegadors més potents (per exemple els basats en WebKit, com són els navegadors de l'Android, l'iOS o el Nokia S60 Series) s'ha passat bàsicament d'XHTML a suportar HTML i sobretot HTML5. Gran part d'aquest avenç s'aconsegueix amb HTML5 i CSS3, que, entre altres coses, ens ofereixen la possibilitat de tenir aplicacions molt semblants i de molt potencial respecte a les aplicacions natives.

WebKit

WebKit és una plataforma d'aplicacions que funciona com a base de diversos navegadors com són el Google Chrome o el Safari. Està basada en el motor de renderització KHTML del navegador Konqueror del KDE.

A continuació presentem l'evolució dels llenguatges de marcatge, partint de l'original SGML. Veurem que, finalment, s'arriba al que fins ara era el llenguatge més usat en el desenvolupament de webs per a dispositius mòbils: l'XHTML-MP (*mobile profile*). Com es pot observar, hi ha hagut moltes alternatives, i algunes que no hem vist, la qual cosa provocava una gran dificultat per a desenvolupar aquests webs:



Els avantatges de les aplicacions web sobre mòbils són diversos:

- Possibilitat d'accés a molta informació del dispositiu per a fer aplicacions relativament complexes.
- Desenvolupament, distribució i proves senzills.
- Convergència entre aplicacions de taula i de dispositius mòbils. Això té moltes implicacions, com que els desenvolupadors només han de conèixer una tecnologia.
- Ús d'estàndards del Web, clarament definits.
- Àmpliament suportat per la indústria, de tal manera que la majoria dels nous dispositius tenen suport per a aquest tipus d'aplicacions.

Per contra, també tenen inconvenients:

- Es necessita un navegador que tingui suport per a aquest tipus de tecnologia, i encara que la majoria dels nous dispositius n'incorpora, els antics no en tenen.
- El rendiment és més baix respecte a aplicacions natives, ja que s'executa tot per mitjà del JavaScript del navegador, la potència del qual és limitada.
- Impossibilitat d'accedir a totes les possibilitats del dispositiu, ja que no es pot accedir al maquinari ni tampoc a molts perifèrics com els sensors o les càmeres. No es pot accedir a molta de la informació de l'usuari, com els contactes o les cites.

Exemple d'aplicació mòbil

Actualment hi ha molts exemples d'aplicacions web fetes per a executar-se en taula, que han estat ràpidament portades als dispositius mòbils, com són mobile.twitter.com, facebook.com o maps.google.com.

Introducció a l'HTML 5

L'HTML5 és un gran pas endavant per a millorar els estàndards web i aconseguir desenvolupaments per a dispositius mòbils més sostenibles. És per això que veurem una petita introducció a aquest estàndard.

Abans de començar a definir-se l'HTML 5, el consorci W3C[3] ja estava treballant en un estàndard per a substituir la gran varietat d'estàndards web que hi havia: l'XHTML 2.0.

(3)El World Wide Web Consortium (W3C) és un consorci internacional que produeix recomanacions, no estàndards, per al World Wide Web (WWW). El creador, i actual cap, d'aquest consorci, és Tim Berners-Lee, considerat el creador del Web, concretament el creador d'URL, HTTP i HTML.

Logotip oficial de l'estàndard HTML 5

L'XHTML 2.0, igual que el seu predecessor XHTML 1.0, contenia elements XML, per la qual cosa era incompatible amb l'HTML 4. A causa de la lentitud del W3C per a generar nous estàndards (l'HTML 4.01 va ser aprovat el 1999), el WHATWG[4] va decidir prendre com a referència HTML 5 i abandonar l'estructura HTML en favor de l'XML. Finalment, el 2006, el W3C decideix participar en l'elaboració de l'HTML 5. Es desenvolupen en paral·lel l'XHTML 2.0 i l'HTML 5. D'aquesta manera, s'ha aconseguit el suport dels principals navegadors i es continua essent compatible amb les versions anteriors.

(4)Web Hypertex Application Technology Working Group. Aquest grup el formen les empreses responsables dels principals navegadors web: Apple, Opera, Mozilla, Microsoft i Google.

Amb HTML 5 es vol aconseguir:

- Afegir novetats als HTML i CSS actuals.
- Donar suport a tots els navegadors, incloent els navegadors mòbils i així evitar una ruptura total amb les versions actuals de l'estàndard.
- Aprofitar les característiques dels dispositius actuals, com poden ser acceleració gràfica o múltiples processadors.
- Tenir més llibertat de desenvolupament i així evitar al màxim la necessitat d'extensions de propietat que hi ha actualment.

L'HTML 5 s'engloba dins de l'estàndard Web Applications 1.0, que incorpora alguns estàndards més. Aquest estàndard dóna suport, per exemple, a les versions XHTML i HTML de l'estàndard i defineix les API per poder fer les accions dinàmiques (amb ECMAScript[5]).

(5)Estàndard sobre el qual està basat el popular llenguatge Javascript.

Actualment algunes de les coses que es poden fer amb les aplicacions mòbils són:

- Visualització d'informació i interacció amb aquesta informació, per mitjà de les teles (*canvas*) i els formularis (*web forms*).
- Mode sense connexió. Es desa la informació de l'usuari al dispositiu físic, mitjançant l'emmagatzemament local, per exemple, per a suportar falta de connectivitat.
- Emmagatzematge de dades en el navegador com una base de dades.
- Accés a dades com la posició geogràfica.
- Renderitzatge d'objectes 2D i 3D aprofitant la potència de les targetes gràfiques dels dispositius, per a millorar-ne així el rendiment, gràcies en part al suport d'SVG[6].

[6]SVG (*scalable vector graphics*), especificació per a definir gràfics 2D, tant estàtics com animats. Des de 2001, és una recomanació del W3C. L'SVG està integrat en els estàndards web, per la qual cosa es permet definir els objectes gràfics amb etiquetes estàndard HTML.

- Suport per a vídeo i àudio (etiquetes *video* i *audio*) sense necessitat d'extensions de propietat.
- Noves etiquetes semàntiques, com *section, article, header, nav*, etc.
- API per a agafar i arrossegar (*drag & drop*).
- Treballs pesants en segon pla (*Web workers*).
- Gestió de connexions remotes (a través dels *Web Sockets*).

Enllaços d'interès

En l'adreça web següent trobareu diversos exemples de les possibilitats d'HTML5: http://html5rocks.com

I en aquesta adreça trobareu el suport actual dels navegadors per als diferents estàndards: http://html5readiness.com/

Paral·lelament a la definició dels estàndards HTML, s'han anat actualitzant els estàndards corresponents a l'estil de les pàgines; en concret la nova versió és CSS3. Aquesta versió està dividida en mòduls que s'han aprovat com a especificacions en moments diferents. Alguns dels més importants són *CSS selectors*, *CSS colors*, *backgrounds & borders*, *CSS multi-column layout*, *CSS namespace*, *media queries*, *CSS speech*, *CSS animations*, *CSS 3D transformations*, etc. Tots aquests estàndards també contribueixen a aconseguir que l'experiència dels usuaris de webs per a dispositius mòbils sigui més completa i la programació pugui ser més estructurada.

2.1.4. Aplicacions web mòbils natives

Hi ha un tipus d'aplicacions, anomenades *aplicacions web mòbils natives*, que no són aplicacions web pròpiament ni tampoc són natives. S'executen amb un navegador, o més ben dit, un component natiu que delega en un navegador, però que tenen alguns dels avantatges de ser natives.

Aquest tipus d'aplicacions es basen a poder ser instal·lades al dispositiu, amb la qual cosa, per exemple, es poden distribuir pels canals estàndards de distribució d'aplicacions natives o es poden incorporar com a accessos directes igual que la resta d'aplicacions, però amb la peculiaritat que finalment les aplicacions no tenen la potència de les aplicacions natives, sinó que simplement executen codi en un navegador incrustat, generalment amb HTML5.

Els avantatges de les aplicacions web mòbils natives són els següents:

- Tots els punts a favor de les aplicacions web mòbils.
- Es poden considerar en termes d'instal·lació i distribució com a aplicacions natives.

En canvi, els inconvenients que tenen són:

- La majoria dels punts en contra de les aplicacions web mòbils, a excepció de la instal·lació en el client.

- L'experiència de l'usuari de vegades és contradictòria, ja que malgrat tractar-se d'una aplicació nativa, requereix connexió a Internet per a poder treballar i es comporta amb els temps de resposta d'un navegador.

Exemple d'aplicacions mòbils natives

Aplicacions d'accés a informació confidencial, com són les entitats financeres en línia (*on-line*), que distribueixen l'aplicació, però en les quals per la naturalesa que tenen no és recomanable tenir dades al dispositiu físic.

Hi ha també els anomenats *mobile widgets*, que es poden englobar com un subtipus d'aplicació web mòbil. Es basen a executar aplicacions web com a aplicacions natives, però com un *widget*, és a dir, que es poden executar contínuament o estan integrades en la plataforma de destinació. Exemples d'aquest tipus d'aplicacions són un cercador de resultats a la Wikipedia o un traductor. Tenen la majoria dels avantatges de les aplicacions mòbils amb les limitacions de dimensió, ja que solen ser executades en una part limitada de la pantalla.

Exemples de *mobile widgets*

Alguns exemples de *mobile widgets* són WRT (*web runtime widgets* de Symbian) o BlackBerry Widget SDK.

2.1.5. Aplicacions natives

Les aplicacions natives són les aplicacions pròpies de cada plataforma, i han de ser desenvolupades pensant en la plataforma concreta. No hi ha cap tipus d'estandardització ni en les capacitats ni en els entorns de desenvolupament, per la qual cosa els desenvolupaments que pretenen suportar plataformes diferents solen necessitar un esforç extra.

Aquestes aplicacions són les que tenen més potencial, ja que aprofiten al màxim els dispositius, i s'aconsegueix una experiència d'usuari millor.

Hi ha moltes plataformes, gran part lligades al tipus de dispositiu, encara que també hi ha plataformes, com l'Android, que existeixen per a diferents tipus de dispositius.

Algunes de les més conegudes són iOS, Android, BlackBerry, bada, Java ME, Windows Phone (abans Windows Mobile o Windows CE), Symbian, Web OS o BREW. Totes tenen diferents tipus de dispositius amb una base comuna.

Els avantatges de les aplicacions natives són els següents:

- Accés total al context, amb totes les possibilitats que això té. S'aconsegueixen les millors experiències d'usuari.

- Possibilitat de gestió d'interrupcions en l'aplicació o en les capacitats del dispositiu. Des de saber si tenim connectivitat de dades, o bé connectivitat de localització, fins a tenir informació sobre la bateria.

- Relativament fàcils de desenvolupar si només es preveu una plataforma.

- Es poden distribuir pels canals coneguts d'aplicacions que permeti la plataforma, amb la qual cosa es poden monetitzar més fàcilment.

- Totes les novetats arriben primer a aquest tipus d'aplicacions, ja que són en aquest tipus d'aplicacions on es proven.

En canvi, tenen inconvenients:

- Portar aplicacions és costós. En cas de voler fer una aplicació per a més d'una plataforma, es complica el desenvolupament a causa dels problemes de la fragmentació.

- Depenent de la plataforma elegida, hi pot haver fragmentació dins de cada plataforma, a causa dels diferents tipus de dispositius o versions de la plataforma.

- No hi ha un estàndard, per la qual cosa cada plataforma oferirà les seves peculiaritats.

- Normalment poder desenvolupar, distribuir o verificar aquestes aplicacions en dispositius reals sol requerir una llicència de pagament, depenent de la plataforma.

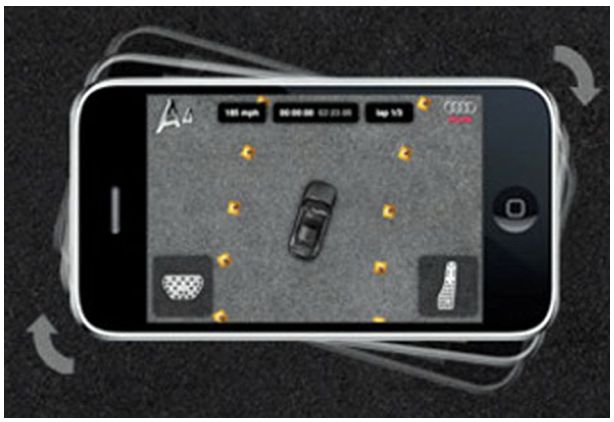
- Els guanys per aquestes aplicacions se solen dividir entre el creador de l'aplicació i la plataforma de distribució.

Exemples d'aplicacions natives

Alguns exemples d'aplicacions natives són:

- Aplicacions per a la gestió de l'agenda, o per a trobar amics de l'agenda amb la seva posició geogràfica.
- Alarmes o aplicacions correu electrònic.
- Aplicacions socials, com les aplicacions de les xarxes socials, i també altres aplicacions que permeten utilitzar l'accés a aquestes aplicacions, com són els agregadors de xarxes socials.

Les imatges següents representen aplicacions natives que aprofiten les funcionalitats del dispositiu, com són aplicacions de realitat augmentada o bé jocs.



2.2. Estratègies de desenvolupament d'aplicacions mòbils

A l'hora d'emprendre un projecte mòbil una de les coses importants que hem de conèixer són les alternatives. De vegades és inviable conèixer-les totes i normalment molt difícil conèixer-les totes en detall, però sí que és bo tenir una visió general de les opcions i alternatives del mercat. En aquest mòdul es donarà una visió general ampliant algun dels detalls.

Atesa la gran fragmentació de plataformes i tipus d'aplicacions que hi ha, el primer que hem de fer és intentar minimitzar al màxim el ventall de possibilitats, i per a això veurem els tipus d'aplicacions a què ens enfrontarem.

Una vegada tinguem clar el tipus d'aplicació que volem fer, hi ha diferents estratègies per a portar-la al nostre dispositiu i dins d'aquestes estratègies hi ha moltes alternatives concretes. Veurem també els punts forts i febles de cada alternativa.

Així podrem veure les diferents estratègies per a desenvolupar al nostre mòbil.

2.2.1. Desenvolupaments web

Dins d'aquest subapartat englobem totes les aplicacions que estan basades en llenguatges de marques, la qual cosa afegeix la facilitat de poder programar i verificar sense necessitat d'un emulador o un dispositiu real. A aquestes aplicacions s'hi accedeix directament per mitjà de la Xarxa, i queden excloses les aplicacions que requereixen un preprocés per a poder ser distribuïdes.

En aquest punt també s'inclouen aplicacions web basades en aplicacions de propietat, com pot ser Flash i Flash Lite. Aquestes aplicacions tenen el mateix model de desenvolupament.

1) Prerequisits. En general es pot utilitzar qualsevol entorn de desenvolupament conegut. En el cas d'aplicacions de tipus *widget* o entorns d'execució de propietat, els fabricants solen donar suport per a aquests desenvolupaments per mitjà d'entorns de desenvolupament específics.

2) Fragmentació. La fragmentació en aquest tipus d'aplicacions existeix, encara que sol ser més petita que en la resta. Per a poder adaptar la nostra aplicació a les capacitats del dispositiu, i com que estem en una arquitectura de navegador web, la millor opció és intentar reconèixer el dispositiu en el moment de rebre la primera petició. També es pot intentar mostrar de manera diferent la informació al navegador, però normalment les capacitats del navegador fan inviable aquesta opció.

Una vegada que sabem el dispositiu n'hem de conèixer les capacitats. Hi ha diverses maneres d'adaptar la nostra aplicació al dispositiu:

a) Tenint-ne coneixement, com per exemple amb el projecte WURLF, que consisteix en una base de dades de tots els dispositius i les capacitats que tenen.

b) Utilitzant servidors que incorporen aquest coneixement i l'automatitzen, i arriben a fer transformacions automatitzades. Aquest cas té el problema que les últimes versions no solen estar suportades.

Exemple

El següent exemple és d'un codi de validació de les capacitats d'un dispositiu, utilitzant la biblioteca de WURLF per a reconèixer el dispositiu i les capacitats que té.

```
public class HelloWorld extends HttpServlet {
      ...
      protected void processRequest(
          HttpServletRequest request,
          HttpServletResponse response)
          throws ServletException, IOException {

              WURFLHolder wurflHolder = (WURFLHolder) getServletContext()
                                  .getAttribute("net.sourceforge.wurfl.core.WURFLHolder");

              WURFLManager wurfl = wurflHolder.getWURFLManager();

              Device device = wurfl.getDeviceForRequest(request);

              MarkUp markUp = device.getMarkUp();
              request.setAttribute("markUp", markUp);
              request.setAttribute("device", device);
         ...
         }
     }
```

3) Proves. Les proves es poden començar amb navegadors d'escriptori que suportin HTML5 o el llenguatge de marcatge corresponent, però després s'hauran de fer proves reals amb emuladors i, molt recomanable, amb dispositius reals. Cal parar especial atenció a aquelles parts dels estàndards que no són suportades per tots els dispositius.

En el món del desenvolupament web hi ha moltes eines per a verificar les aplicacions web i la gran majoria es pot aplicar a aquest tipus d'aplicacions.

4) Distribució. La distribució és el més senzill, ja que es troba en el mateix nivell que la distribució d'una aplicació web qualsevol. Només en cas que tingui canvis incompatibles amb les dades desades fora de línia d'una aplicació, s'hauran de fer accions informatives perquè l'usuari actualitzi aquestes dades.

2.2.2. Entorns de desenvolupament natius

Com ja hem vist, les aplicacions natives són les que ofereixen una experiència d'usuari millor; són aquelles que estan especialment dissenyades i implementades per al context d'execució (plataforma o dispositiu) on s'executaran, i poden treure partit de totes les capacitats d'aquests dispositius. De vegades, també són subjectes a normes específiques dels fabricants de dispositius o els responsables de les plataformes.

1) Prerequisits. Per a poder desenvolupar per a una aplicació nativa, generalment es necessita l'entorn de desenvolupament o IDE de cada plataforma. Per exemple, per a l'Android en necessitem l'SDK i és recomanable usar l'Eclipse i afegir-hi alguns connectors; en el cas de l'iOS, en canvi, necessitem l'xCode, per a BlackBerry *apps* en necessitem també l'SDK, per al Windows Phone es necessita el Microsoft Visual Studio, etc. Aquests IDE poden tenir una llicència de pagament, que depèn de cada plataforma.

IDE

L'entorn integrat de desenvolupament o *integrated development environment* (IDE) és un entorn de desenvolupament que incorpora totes o gairebé totes les eines de desenvolupament necessàries, com poden ser eines de modelatge o disseny, o eines de depuració.

Alguns d'aquests IDE no són multiplataforma, per la qual cosa requereixen disposar d'un equip de desenvolupament específic, com pot ser el cas del Windows Phone o l'iOS.

Aquests IDE solen proporcionar tot el necessari per a cobrir tot el desenvolupament de l'aplicació, incloent-hi els emuladors necessaris per a provar la nostra aplicació mentre la desenvolupem.

2) Implementació. Totes les implementacions són diferents, i cada sistema utilitza el seu mètode i els seus patrons propis, però hi ha coses comunes:

a) Es disposa d'un emulador per a poder provar les nostres aplicacions. Però de vegades l'emulador no permet emular totes les accions d'usuaris o no és prou àgil l'emulació, i es necessita un dispositiu real.

b) Separació de presentació i lògica, per a poder aprofitar així al màxim els components.

c) Possibilitat de depurar la nostra aplicació per a poder tenir més control.

d) Generalment hi ha eines per a facilitar la construcció de les interfícies gràfiques o *user interface* (UI).

3) Proves. Per a poder fer proves, cada IDE té les seves eines, des de les típiques tecnologies de proves unitàries, fins a sistemes més complexos com són el Monkeyrunner de l'Android per a poder fer proves d'estrès de les aplicacions. També hi ha eines per a fer proves d'acceptació contra la UI utilitzant llenguatges d'alt nivell, com és el cas de la UIAutomation sobre l'iOS.

Monkeyrunner

Les eines Monkeyrunner i Monkey serveixen per a llançar molts esdeveniments d'usuari sobre l'emulador; són esdeveniments aleatoris, i per això s'usa la metàfora d'una mona.

Sens dubte, les possibles proves que es poden fer sobre les aplicacions natives són molt més extenses i controlades respecte a les que es puguin fer en un altre tipus d'aplicació, ja que tenim les eines pròpies de la plataforma.

UIAutomation

La UIAutomation serveix per a definir en JavaScript les accions d'usuari que finalment s'executen en el dispositiu o l'emulador.

4) Signatura i distribució. Per a poder distribuir l'aplicació o fins i tot executar-la en un terminal per a fer proves, pot ser necessari signar-la amb un certificat digital que ens identifiqui com a desenvolupadors.

Si la distribució és per mitjà de tercers, com poden ser els mercats d'aplicacions o *marketplaces*, és encara més necessari per a poder acreditar que tenim el dret de publicar aplicacions, i també per a fer-nos-en responsables.

Segons la plataforma, els models de distribució són simplement sistemes de baixada, bé amb sistemes OTA (actualització aèria, *over-the-air*) o també per mitjà de mercats d'aplicacions. Aquest últim sistema està tenint una gran acollida perquè permet monetitzar fàcilment l'aplicació i arribar a un gran nombre d'usuaris potencials ràpidament, encara que pot tenir unes fortes restriccions a l'hora d'aconseguir publicar la nostra aplicació.

2.2.3. Entorn de desenvolupament multiplataforma

Com hem estat veient, les aplicacions natives són molt potents, però alhora requereixen un esforç de desenvolupament per a suportar tan sols una plataforma, i així amb cadascuna de les plataformes que vulguem suportar. Per a poder suportar totes les plataformes necessitaríem saber molts llenguatges, ja que caldria portar les aplicacions entre plataformes. En concret, actualment hi ha almenys cinc llenguatges diferents que són necessaris per a poder fer aplicacions sobre les plataformes més actuals: C, C++, Java, C#, JavaScript i Objective-C. A més dels diferents IDE necessaris i les biblioteques específiques corresponents.

En canvi, les aplicacions web ens permeten arribar a moltes plataformes amb un mateix codi, i sense necessitat de portar el codi, però sense poder portar a l'usuari la mateixa experiència que s'aconsegueix amb les aplicacions natives.

Llavors si hi hagués la possibilitat de fer aplicacions natives des d'una mateixa línia de codi, tindríem la millor de les dues aproximacions, i aquí és on entren en joc les estratègies d'aplicacions multiplataforma o *cross-platform*, també conegudes com a *aplicacions híbrides*.

Des de fa temps ja s'està intentant aquesta capacitat multiplataforma amb intents com el Java ME, però no han donat els resultats esperats. Amb l'arribada de l'HTML5 i l'explosió de nous telèfons intel·ligents, sembla que han aparegut noves alternatives que s'han de tenir en compte. Moltes d'aquestes alternatives aprofiten l'HTML5 com a base i construeixen al seu voltant maneres d'accedir a les capacitats que l'HTML5 no dóna de partida, pràcticament sempre per

mitjà d'objectes JavaScript. Usar elements 100% estàndard com són HTML5, CCS i JavaScript ofereix un gran punt a favor, ja que es tracta de tecnologies àmpliament conegudes.

N'hi ha que simplement es queden en un *wrapper* de l'aplicació HTML5 afegint aquests punts d'accés; en canvi, d'altres fan preprocessament per a acabar generant aplicacions 100% natives. Hi ha altres alternatives que proporcionen la seva pròpia arquitectura i llenguatges diferents, i també per mitjà d'un sistema de compilació o execució via màquina virtual aconsegueixen tenir aplicacions natives.

Hi ha coses que aquestes aproximacions no poden evitar de manera senzilla (tret que tinguin codi condicional específic per a cada plataforma):

- **Pèrdua de controls específics d'una plataforma**. Si es té un control de la UI o una funcionalitat concreta que només hi ha en una plataforma, no es podrà generar de manera única per al nostre desenvolupament multiplataforma.

- **Integració en l'escriptori del dispositiu**. Segons la plataforma les possibilitats d'afegir elements a l'escriptori de cada usuari varien. Per exemple, en l'Android o el Symbian es poden crear *widgets* potents per a millorar la usabilitat de la nostra aplicació, mentre que en el Windows Phone només és possible afegir icones de l'aplicació.

- **Gestió de la multitasca**. Com que es tracta de conceptes de baix nivell de cada plataforma, cadascuna el tracta de manera diferent, amb restriccions diferents, per la qual cosa no serà fàcil fer codi comú per a totes sense perdre molta potència.

- **Consum de la bateria**. Aquestes aproximacions requereixen una capa d'abstracció sobre el nostre dispositiu, que provoca problemes, com la multitasca; de la mateixa manera, el control sobre el consum de bateria es fa més difícil quan no es tenen les capacitats concretes de la plataforma. També afecta el fet de no tenir control sobre la multitasca, ja que aquesta és una de les maneres d'estalviar consum de bateria.

- **Serveis de missatgeria asíncrona o *push services***. Serveixen per a implementar coses com la missatgeria instantània, però com que cada plataforma els implementa d'una manera, es fa complicat atacar-los conjuntament.

Els següents són alguns exemples que hi ha:

- **PhoneGap**. Es tracta d'aplicacions fetes en HTML5 que tenen objectes JavaScript que permeten l'accés via enllaços a les funcions natives per a les

capacitats que HTML5 no ofereix. Les aplicacions s'executen sobre un component que conté un navegador.

- **Rhodes**. Es tracta d'aplicacions escrites en Ruby utilitzant un patró de disseny MVC[7] i executades en màquines virtuals específiques de Ruby per a cada plataforma; per tant, són aplicacions natives. Inclou sistemes per a sincronitzar dades de manera senzilla, i així aconseguir el canvi de mode en línia a fora de línia sense gaire cost.

[(7)]model vista controlador

- **Appcelerator**. Es tracta d'aplicacions escrites en HTML5, que és compilat en aplicacions natives. També són capaços de generar aplicacions de taula clàssiques, executables sobre Mac, Linux o Windows.

- **Wacapps**. Es tracta d'aplicacions HTML5, amb el suport per a la distribució dels operadors.

- **Flash**. Són aplicacions que corren sobre un reproductor de propietat, i si n'hi ha, no necessiten ser portades. S'escriuen de la mateixa manera que les aplicacions de taula, i tenen les mateixes restriccions.

- **Java ME**. Són aplicacions escrites en Java, amb totes les peculiaritats dels perfils J2ME, i s'executen per mitjà d'una màquina virtual amb les restriccions corresponents. Actualment es troben en gairebé el 80% dels dispositius mòbils del mercat.

- **Unity3D**. Entorn per a desenvolupar jocs natius o web per a cada plataforma, incloent-hi plataformes com la Play Station, el Wii, PC o Mac, i suport per a l'Android i l'iPhone.

Aquestes són només algunes de les existents, però n'hi ha moltes més i continuen sortint alternatives.

Aquests desenvolupaments ens ofereixen beneficis dels desenvolupaments previs:

- Només una línia de desenvolupament per a diverses plataformes.
- Aplicacions instal·lables en els nostres dispositius, i amb la possibilitat de distribuir-les per mitjà dels mercats d'aplicacions.
- Depenent del cas, el resultat són aplicacions natives que aprofiten en gran manera el potencial del dispositiu i ofereixen un disseny i una experiència d'usuari idèntics a les aplicacions desenvolupades només per a una plataforma.

Però també té punts negatius:

- En diferents graus, tenen accés parcial als recursos del dispositiu. Això es tradueix que segons el grau no poden interactuar amb algunes capacitats, i

en més o menys grau seran aplicacions menys eficients que les aplicacions natives.

- Per a poder suportar diversos dispositius, se sol haver de fer el mínim comú denominador de les capacitats; així, si una plataforma aporta una nova funcionalitat (per exemple, un sistema de comunicació que la resta no té) no es podrà implementar sense afegir codi específic.

- Suport parcial. No tenen suport absolut per a totes les plataformes i totes les versions.

- Les noves funcionalitats triguen a ser suportades. Com que les novetats en les plataformes apareixen a un ritme molt elevat i en curts períodes de temps, si es desenvolupa utilitzant aquests entorns cal esperar un temps per a poder fer ús d'aquestes eines, o en el millor dels casos, si es tracta de codi lliure o ampliable, es pot fer una implementació per a tenir accés a aquesta funcionalitat.

- Es requereix pagar la llicència, si en té, per a l'entorn de desenvolupament, més la llicència pròpia de cada plataforma per a poder distribuir l'aplicació.

Com veiem, aquesta alternativa és sens dubte per a tenir en compte per a decidir quina estratègia de desenvolupament volem per al nostre projecte web, però no sempre serà la millor.

1) Prerequisits. En general cadascun dels entorns proporciona el seu entorn de desenvolupament complet per a poder aconseguir les aplicacions natives.

En molts casos la manera de provar les aplicacions és per mitjà dels emuladors de les plataformes natives, de manera que es necessita instal·lar l'IDE propi de l'entorn de desenvolupament i els emuladors, o de vegades els SDK.

SDK

L'equip de desenvolupament de programari (SDK) són les eines necessàries per a poder dur a terme el desenvolupament d'aplicacions en una plataforma.

2) Implementació. La implementació variarà molt segons cada plataforma, ja que n'hi ha algunes que podran aprofitar totes les eines de desenvolupament, altres que no ho necessiten en excés, i algunes en les quals el desenvolupament serà més difícil.

Però en general, les eines faciliten el procés de desenvolupament, encara que sense arribar al nivell de les eines de desenvolupament natives.

3) Signatura i distribució. La distribució de vegades es pot fer directament per mitjà dels IDE de les plataformes de desenvolupament, però en la majoria dels casos és necessari fer-ho per mitjà dels canals habituals per a les aplicacions natives.

2.3. Mètodes aplicats al desenvolupament d'aplicacions mòbils

En el món del desenvolupament de programari hi ha molts mètodes de desenvolupament, cadascun amb punts forts i punts febles. En el cas del desenvolupament d'aplicacions mòbils succeeix el mateix, i quan ens plantegem quin mètode volem triar haurem de saber escollir segons les nostres necessitats.

Alguns dels mètodes més coneguts són:

- Model *waterfall*.
- Desenvolupament ràpid d'aplicacions.
- Desenvolupament àgil (qualsevol de les variants que té).
- Mobile-D.

Una de les característiques importants de la gran majoria de desenvolupaments mòbils és una durada curta, i això es deu a diversos factors com són la gran competència en el sector o els canvis, amb l'aparició de novetats tant de programari com de maquinari gairebé constants; també el fet que moltes aplicacions neixen amb un desenvolupament precoç en forma de prototip i van evolucionant, o fins i tot la simplicitat de les aplicacions, que no requereixen grans desenvolupaments. Això sol ser, amb excepcions, la norma dels desenvolupaments d'aplicacions per a dispositius mòbils.

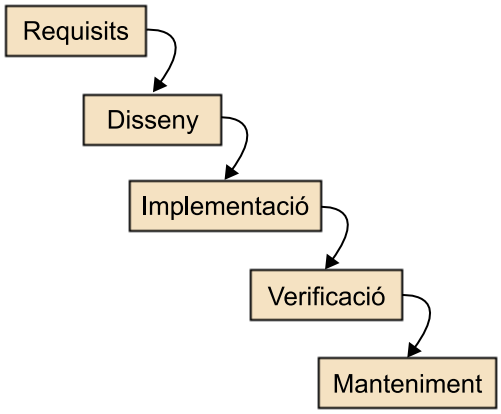
"It can take up to nine months to deploy an entertainment (mobile) application, but that's the duration of a cell phone in this market."

Craig Hayman (IBM)

2.3.1. Model *waterfall*

El model *waterfall* és el model més estàtic i predictiu. És aplicable en projectes en què els requisits estan fixats i no canviaran durant el cicle de vida del desenvolupament. Aquesta aproximació divideix el projecte en fases estanques totalment seqüencials. En aquest model, el desenvolupament s'interpreta com l'aigua que va caient d'un estany al següent. Es posa molt èmfasi en la planificació, els temps i les dates límit i el pressupost.

Exemple de fases d'un projecte de programari en un model *waterfall*



En el context del desenvolupament d'aplicacions mòbils, el model *waterfall* pot ser aplicable a projectes realment controlats i previsibles, en què no hi ha gaire incertesa pel que es vol fer i no són importants els canvis constants en la indústria.

2.3.2. Desenvolupament ràpid d'aplicacions

El desenvolupament ràpid d'aplicacions és un mètode de desenvolupament iteratiu, centrat molt a aconseguir prototipus com abans millor i anar millorant-los. Se sol prioritzar la implementació sobre la planificació, utilitzant molts patrons de disseny coneguts, per a poder adaptar-se tan bé com es pugui a canvis en els requisits.

El desenvolupament ràpid d'aplicacions és un mètode molt útil per al desenvolupament de projectes realment urgents amb temps de lliurament molt curts.

2.3.3. Desenvolupament àgil

El desenvolupament àgil és un model de desenvolupament basat en iteracions, en què en cada iteració es fan totes les fases del cicle de desenvolupament.

Manifest àgil

El manifest àgil va ser publicat el 2001 per disset desenvolupadors de programari representants dels mètodes de desenvolupament més populars llavors, que es passarien a conèixer com a *àgils* (*extreme programming*, *crystal clear*, DSDM o ASD, entre d'altres). El manifest defineix els dotze principis i quatre valors ètics per als desenvolupadors. Es pot veure a http://agilemanifesto.org.

El desenvolupament àgil es basa en els principis del manifest àgil i els seus valors ètics tracten de donar més valor a uns conceptes que a d'altres, però no deixen de fer aquests altres:

1) Valorar més els individus i les seves interaccions que els processos i eines.

2) Valorar més el programari que funciona que la documentació exhaustiva.

3) Valorar més la col·laboració amb el client que la negociació contractual.

4) Valorar més la resposta al canvi que el seguiment d'un pla.

Amb aquests valors s'intenta entre altres coses aconseguir lliurar valor al més aviat possible, sense tenir problemes amb els canvis de requisits. Això ho fa molt vàlid per a projectes canviants, tant si són grans com petits, i intenta mitigar els riscos amb aquests valors. Per a poder aconseguir projectes que puguin canviar fàcilment es fa esment a la qualitat dels productes aconseguits, cosa que és realment important en el cas dels projectes de programari per a dispositius mòbils. Per a aconseguir això es basen en les proves de l'aplicació, sovint automatitzant-les.

Mètodes derivats

Hi ha diversos mètodes derivats d'aquest model de desenvolupament àgil, com són Scrum, Kanban, Lean, AUP o *extreme programming*.

Els mètodes àgils solen ser molt adequats per al desenvolupament d'aplicacions mòbils per diverses raons:

- **Alta volatilitat de l'entorn**. Amb canvis en entorns de desenvolupament, de nous terminals i noves tecnologies a un ritme molt més elevat que en altres entorns de desenvolupament.

- **Equips de desenvolupament petits**. Com que els desenvolupaments mòbils solen ser projectes relativament petits, els equips no solen ser gaire grans. Generalment són duts a terme per desenvolupadors individuals o pimes.

- **Programari no crític**. No solen ser aplicacions d'alt nivell de criticitat, ja que solen ser aplicacions per a entreteniment o gestió empresarial no crítica.

- **Cicles de desenvolupament curts**. Atesa l'evolució constant de la indústria, es requereixen cicles de vida realment curts per a poder donar sortida a les aplicacions a temps.

2.3.4. Mobile-D

El mètode Mobile-D es va desenvolupar juntament amb un projecte finlandès del 2004, principalment fet per investigadors de la VTT (institut d'investigació finlandès), i malgrat que és un mètode antic continua en vigor, ja que és utilitzat en projectes amb èxit, a més d'estar basat en tècniques que funcionen.

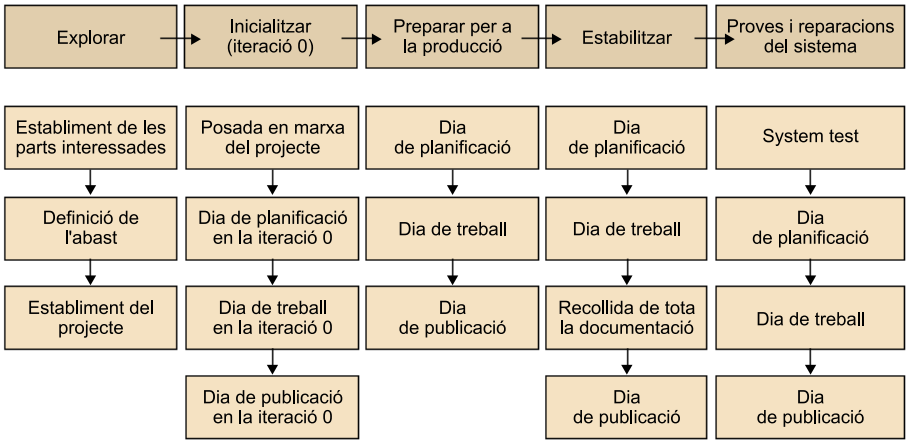
Enllaç d'interès

Per a saber més sobre Mobile-D podeu visitar la pàgina web de la VTT: http://agile.vtt.fi/mobiled.html.

La motivació és aconseguir cicles de desenvolupaments molt ràpids en equips molt petits (no pas més de deu desenvolupadors) treballant en un mateix espai físic. Segons el que advoca el mètode, treballant tal com diu, s'han d'aconseguir productes totalment funcionals en menys de deu setmanes.

Es basa en solucions conegudes i consolidades: *extreme programming* (XP), *crystal methodologies* i *rational unified process* (RUP). XP per a les pràctiques de desenvolupament, *crystal* per a escalar els mètodes i RUP com la base en el disseny del cicle de vida.

Cicle de desenvolupament de Mobile-D



Cada fase (excepte la inicial) té sempre un dia de planificació i un altre de lliurament. Les fases són:

- **Exploració**. Es dedica a la planificació i a conceptes bàsics del projecte. És una mica diferent de la resta de fases.

- **Inicialització**. Es preparen i identifiquen tots els recursos necessaris. S'estabilitza l'entorn tècnic.

- **Producció** o **fase de producte**. Es repeteixen iterativament les subfases amb un dia de planificació, un de treball i un de lliurament. Aquí s'intenten utilitzar tècniques com el desenvolupament dirigit per les proves o *test driven development* per a aconseguir la millor qualitat.

Desenvolupament dirigit per les proves

El desenvolupament dirigit per les proves o *test driven development* (TDD) indica que abans de fer una funcionalitat hi ha d'haver una prova que en verifiqui el funcionament.

- **Fase d'estabilització**. Es duen a terme les accions d'integració per a assegurar que el sistema complet funciona correctament.

- **Fase de proves i reparació**. Té com a meta la disponibilitat d'una versió estable i plenament funcional del sistema segons els requisits del client.

2.4. Fases dels projectes de desenvolupament d'aplicacions mòbils

Hem vist que hi ha diversos mètodes existents al mercat per al desenvolupament d'aplicacions mòbils, i tots es divideixen en diferents fases. Cadascun distingeix què cal fer en cada fase, i també en quin nivell o quins resultats es requereixen, però aquestes fases aplicades al desenvolupament d'aplicacions mòbils tindran problemàtiques comunes i solucions comunes, i per això les repassarem.

2.4.1. Planificació

En la fase de planificació s'intenta distribuir el temps i els recursos necessaris per a poder dur a terme el projecte, sia en una única planificació completa o bé en planificacions més dividides.

En les fases de planificació sempre s'intenta aconseguir la màxima precisió de les estimacions, contra els riscos associats al projecte. En el cas dels projectes d'aplicacions mòbils hi ha riscos implícits que s'han de tenir molt en compte:

- **Dificultats pel desconeixement de la tecnologia**. Se sol tractar de tecnologies noves, de vegades desconegudes per als desenvolupadors, i això repercuteix en temps d'aprenentatge més possibles desviacions per falta de contratemps no coneguts.

- **Disposar de dispositius reals**. Per a poder fer un bon projecte de desenvolupament d'aplicacions mòbils, sens dubte és necessari poder provar-ho sobre dispositius reals. Per a això s'ha de planificar una fase de proves reals.

- **Temps de sortida al mercat** o ***time-to-market***. A l'hora de planificar s'han de tenir molt compte els temps en els quals se sol portar una idea al mercat, i intentar reduir-los al màxim, ja que la competència és molt gran.

- **Prototipatge**. És important planificar quan s'aconseguirà el primer prototip, i potser la primera sortida en beta, ja que per a aquest tipus d'aplicacions el prototipatge ràpid pot ser molt útil.

2.4.2. Presa de requisits

Com en tot projecte de programari, cal conèixer els requisits, tant els funcionals com els no funcionals. Dins dels requisits no funcionals haurem de tenir molt present coses en l'àmbit d'ús; és a dir, coses com:

- Qui serà el nostre usuari? En quin moment utilitzarà la nostra aplicació?
- Quins requisits mínims de maquinari són necessaris?
- Necessitem gestionar els modes en línia i fora de línia?
- Hi ha d'haver dades de tercers, des d'un mapa, fins a dades del dispositiu mateix?

Una vegada tinguem tots els requisits serà important dedicar un temps a intentar ordenar-los, la qual cosa ens ajudarà més a prendre decisions.

Un objectiu important d'aquesta fase és aconseguir el pla de dispositius o *device plan*.

Pla de dispositius (*device plan*)

El pla de dispositiu o *device plan* és una llista ordenada de tots els dispositius o grup de dispositius que es volen suportar.

Per a elaborar el pla de dispositius agrupem els dispositius el desenvolupament dels quals es pugui atacar conjuntament (per exemple, tots els que tinguin l'Android 2.1 o superior o tots els que tinguin GPS). A partir d'aquí donem un valor de negoci a cada classe de dispositius i el cost associat a desenvolupar l'aplicació per a aquest grup. Finalment s'aconsegueixen els beneficis esperats de cada classe, tal com es mostra a continuació:

	Cost estimat	Ingressos estimats	Beneficis previstos
Dispositius de classe A	1	5	4
Dispositius de classe B	2	4	2
Dispositius de classe C	3	3	0
Dispositius de classe D	4	2	–2
Dispositius de classe E	5	1	–4

A l'hora de decidir el cost, s'han d'afegir tots els possibles costos tècnics o no tècnics que es puguin deure als requisits que hi ha associats.

Exemple

Si volem fer un joc 3D, s'ha d'afegir el cost associat a desenvolupar-lo si alguna plataforma no té biblioteques conegudes per a aquest tipus de modelatge gràfic; o si volem fer una aplicació per a gestió empresarial d'inventari, per exemple, i volem l'aplicació per a l'Android i l'iOS; i en cas de voler suportar els ordinadors de tauleta corresponents, s'ha de tenir en compte el cost d'adaptar la interfície d'usuari.

Definició de l'arquitectura

Sempre que es té una aplicació, sia mòbil o no, hi ha diverses opcions d'arquitectures. En el cas dels dispositius mòbils hi ha encara més alternatives. Ja hem vist que hi ha la possibilitat de tenir pàgines web mòbils, o aplicacions web mòbils o, és clar, aplicacions natives, però això és només una part del problema, ja que en cas de tenir aplicacions mòbils hi ha moltes arquitectures d'aplicació possibles.

De vegades per a poder prendre una bona decisió sobre l'arquitectura és necessari fer petits prototips; això depèn de la mida del projecte i del coneixement de la tecnologia.

Poder decidir quina és l'arquitectura de la nostra aplicació sens dubte ajudarà a poder saber quin tipus de desenvolupament farem: una aplicació web, una aplicació nativa o potser una d'híbrida. Per tant, si no és clara l'arquitectura poden quedar afectades moltes de les fases següents del projecte.

A continuació repassem les arquitectures més habituals en el desenvolupament d'aplicacions per a dispositius mòbils, com són les aplicacions en línia i fora de línia, o les aplicacions de sincronització, i finalment les aplicacions que necessiten connexió però entre dispositius.

1) Aplicació fora de línia

Les aplicacions fora de línia són aplicacions que una vegada baixades no requereixen per a res la connexió (a excepció d'actualitzacions) per a poder funcionar. Aquestes aplicacions només necessiten desenvolupar l'aplicació del dispositiu mòbil, i no són necessaris més components.

Tenen com a avantatge que es poden utilitzar tant amb connexió com sense, però com a contrapartida solen ser aplicacions de les quals una vegada instal·lades es perd el rastre.

Exemples d'aplicacions fora de línia

Els següents són alguns exemples d'aplicacions fora de línia:

- Molts tipus de jocs, que no comparteixen cap tipus d'informació.
- Aplicacions de productivitat com un gestor de tasques, o alarmes.

2) Aplicació totalment en línia

Les aplicacions totalment en línia són aplicacions que no poden funcionar sense connexió a Internet. Aquestes arquitectures requereixen sense cap dubte una part de servidor i estan pensades per a mantenir-hi una comunicació constant.

Tenen com a contrapartida que quan l'usuari està sense connexió no pot utilitzar l'aplicació, però disposen d'informació constant de les interaccions de l'usuari. És necessari desenvolupar almenys la part servidora, potser una part de desenvolupament en el client i de vegades la comunicació entre totes dues. Com que necessiten estar sempre connectades tenen un consum extra de bateria.

De vegades la part servidora no es necessita fer perquè es tracta d'aplicacions anomenades *remescles* (o *mash-ups*), que aprofiten API existents a la Xarxa per a interactuar amb dades, com pot ser l'API de Twitter, o dades públiques de l'Estat.

Exemples d'aplicacions totalment en línia

Els següents són alguns exemples d'aplicacions totalment en línia:

- Totes les aplicacions que es poden utilitzar al mòbil via el navegador web corresponent, com són les xarxes socials, accessos a correu electrònic web (*webmails*), accessos a *webmails*, etc.).
- Aplicacions web mòbils natives, que són aplicacions web però que tenen aparença i característiques d'una aplicació nativa.
- Aplicacions natives que requereixen la connexió per a poder estar autentificades o per a obtenir les dades.
- Aplicacions de xarxes socials o de temps real en què la connectivitat per a tenir la informació actualitzada és gairebé imprescindible.
- Qualsevol tipus de xat, aplicació de trucades o videoconferències.

3) Aplicacions de sincronització

Les aplicacions de sincronització són aplicacions que poden funcionar de totes dues maneres, en línia i fora de línia, i permeten fer les mateixes accions o molt semblants de totes dues maneres. L'aplicació s'ha d'encarregar de sincronitzar les dades de la situació fora de línia quan es trobi en línia, i gestionar els possibles conflictes. Això repercuteix en un benefici per a l'usuari, ja que pot treballar en qualsevol punt i tenir la informació tan actualitzada com sigui possible.

La sincronització pot ser en un servidor propi o bé amb objectes o API per a la sincronització al núvol, de manera que en facilita el desenvolupament i redueix costos. Hi ha diferents tipus de sincronització:

- **Unidireccional**. Són aplicacions per a poder sincronitzar els canvis amb algun servidor extern, o bé perquè en un dels dos extrems només és possible llegir informació o bé perquè serveixen com a còpia (*backup*) d'informació.

- **Bidireccional**. Es tracta d'aplicacions que poden tenir modificacions tant en el servidor com en el client, i la part servidora i client s'han de comunicar per a fer la sincronització.

Exemples de sincronització bidireccional

Els següents són alguns exemples de sincronització bidireccional:

- Tramesa d'estadístiques de jocs o d'aplicacions esportives.
- Una informació de visualització de mapes o de fulls de ruta per a repartidors.
- Aplicacions de tramesa i recepció de correu electrònic.
- Calendaris que es puguin modificar externament a l'aplicació mòbil.
- Aplicacions de subscripció i lectura d'agregadors de continguts (RSS).

4) Aplicacions per a comunicació entre dispositius

Les aplicacions per a comunicació entre dispositius són aplicacions per a interconnectar dos (unidestinació, *unicast*) o més dispositius (multidestinació, *multicast*) i intercanviar informació.

Aquest tipus d'aplicacions requereixen desenvolupar la part del client a més de la comunicació i la recepció de la informació, i també s'ha de desenvolupar la fase de cerca i enllaç entre els dispositius. Aquest tipus d'aplicacions solen utilitzar comunicacions WPAN, de tipus Bluetooth o NFC.

Exemples d'aplicacions per a comunicació entre dispositius

Els següents són alguns exemples d'aplicacions per a comunicació entre dispositius:

- Aplicacions d'intercanvi de contactes.
- Jocs entre dos o diversos jugadors.

2.4.3. Especificació i disseny

A diferència del que succeeix amb altres fases del desenvolupament d'aplicacions, en el cas de l'especificació aquesta no té grans diferències respecte a les aplicacions de taula normals. Únicament cal puntualitzar que s'ha de tenir en compte que moltes vegades la fase d'especificació queda encavalcada amb el disseny.

Quant al disseny, hi ha alguns patrons de disseny, àmpliament coneguts en el desenvolupament d'aplicacions, que solen ser implementats en les aplicacions per a dispositius mòbils:

- **Model vista controlador** (MVC, *model view controler*). S'utilitza per a poder separar al màxim la lògica de la visualització i interacció i així poder donar

suport a més escenaris, com pot ser el cas d'una aplicació per a telèfon intel·ligent i la mateixa per a ordinador de tauleta.

- ***Threading***. Es refereix a l'ús de fils en segon pla per a fer tasques llargues que bloquegin l'usuari.

- ***Delegation***. Es tracta de delegar una part del treball envers un altre objecte sense que aquest hagi de ser una subclasse del primer. En el cas dels desenvolupaments mòbils és molt útil per a delegar treballs relacionats amb la interfície, i així s'aconsegueix treballar amb esdeveniments de manera molt senzilla i sostenible.

- **Model de memòria gestionada**. En general les aplicacions no s'executen directament sobre la plataforma sinó que hi sol haver una capa intermèdia o programari intermediari (*middleware*), i aquest se sol preocupar de gestionar la memòria, però per a poder fer-ho de manera eficient és necessari dur a terme algunes accions o convencions.

A més dels patrons orientats a aconseguir un programari de més qualitat, també hi ha patrons de disseny per a maximitzar la usabilitat de la nostra aplicació. Aquests patrons han de ser interpretats de manera diferent segons la plataforma per a la qual desenvolupem, ja que al final intenten fer ús dels comportaments comuns a la resta d'aplicacions, i pot ser que aquest comportament variï segons la plataforma.

Exemple

En l'Android es recomana fer que les navegacions via pestanyes se situïn a la part superior de la pantalla, mentre que en l'iOS es recomana que siguin a la part inferior.

En general hi ha uns principis bàsics aplicables a qualsevol disseny, com són la claredat per sobre de la simplicitat o donar més valor al contingut que al continent. Per a això hi ha alguns patrons de disseny de la UI que ajuden a resoldre problemes coneguts d'una bona manera, i que han estat provats àmpliament.

Alguns dels més coneguts són *dashboard*, *actionbar*, *dynamic list*, *pager*, *popups* i *alerts*.

A continuació n'expliquem algun, exposant el problema que solucionen, com el solucionen, i si és necessari algunes recomanacions per a tenir en compte.

Quadre de control o *dashboard*

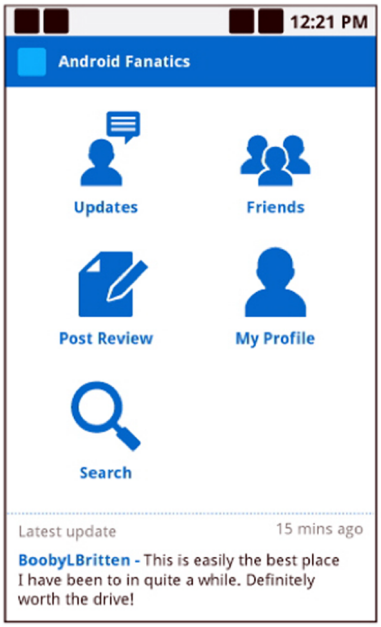
Problema: accedir de manera ràpida, clara i senzilla a les funcionalitats principals de l'aplicació. En el cas del mòbil és important per la resposta que requereix l'usuari.

Solució: tenir una pàgina d'arribada amb la informació clara de l'última informació de l'aplicació i les accions més importants.

Es recomana:

- Destacar el que és nou.
- Focalitzar-se en 3-6 característiques importants.

Exemple de patró d'interfície de quadre de control

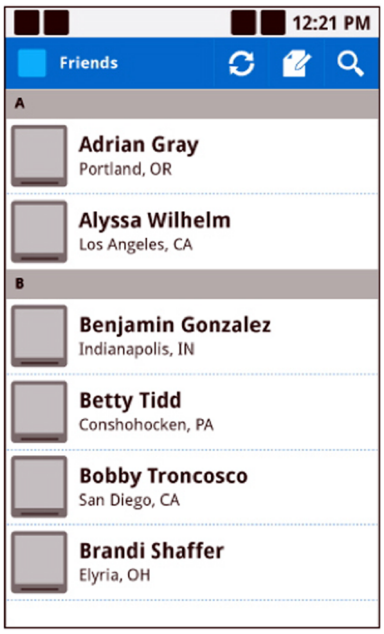


Barra d'acció o *actionbar*

Problema: la limitació d'espai de les pantalles fa que mostrar les accions que l'usuari pot fer en una pantalla pugui ser molt complicat, i pot provocar molta pèrdua d'espai útil si s'introdueix un objecte visual botó per cada acció.

Solució: s'agrupen totes les accions que es poden fer a la pantalla actual en una zona, a la part superior o inferior, depenent de la plataforma, per a aprofitar millor l'espai i tenir més cohesió entre aplicacions. A més, aquest espai es pot aprofitar per a indicar el lloc de navegació on ens trobem.

Exemple de patró de barra d'acció



Menús contextuals o *quick action menu*

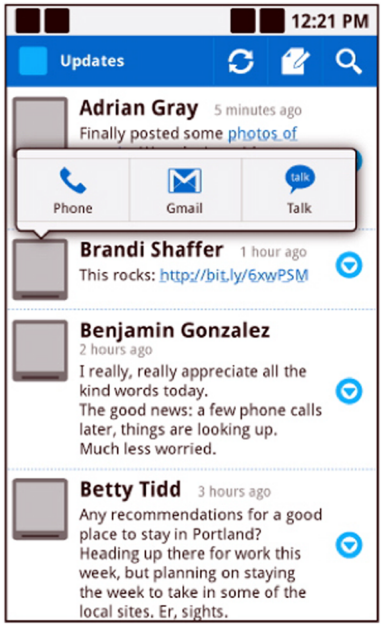
Problema: les limitacions d'espai per a les accions pròpies d'un element de la pantalla ocuparien potencialment molt espai.

Solució: es mostra un menú contextual en polsar l'objecte, generalment en la imatge (per exemple, en les imatges dels contactes), i d'aquesta manera s'evita haver d'afegir més soroll a la interfície, i dins d'aquest menú es poden agrupar totes les accions corresponents a l'objecte en qüestió.

Es recomana:

- Usar-ho només per a les accions més òbvies i importants.
- Usar-ho quan l'objecte no tingui una vista especialment dissenyada per a aquest; en aquest cas cal anar a aquesta vista i mostrar allà la informació.
- No usar-ho en contextos en què se suportin múltiples seleccions.

Exemple d'ús de patró de menú contextual

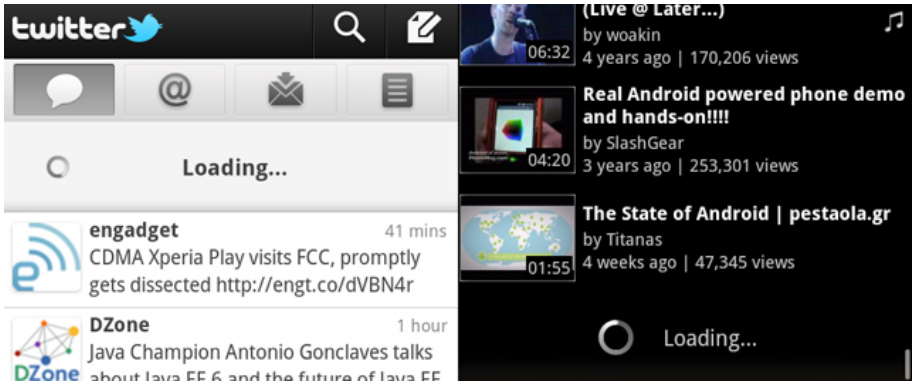


Llista dinàmica o *dynamic list*

Problema: carregar moltes dades en una llista pot ser molt lent, especialment amb problemes de xarxa. Si hem d'esperar tota una llista (que de vegades no acaba o és molt gran) per a mostrar-la, això pot causar una molt mala experiència d'usuari.

Solució: en lloc d'esperar que la llista es carregui, cal mostrar les dades rellevants i carregar la llista immediatament. Les dades que falten es poden carregar quan l'usuari ho demani o bé l'aplicació prevegi que serà així, per exemple quan es desplaci cap al final de la llista.

Exemple d'ús de patró de llista dinàmica



Missatges d'alerta

Problema: succeeix un esdeveniment en el sistema o l'aplicació que és rellevant per a l'usuari.

Solució: mostrar un missatge d'alerta, el qual adquireix el focus de l'usuari per a advertir-lo de la situació. Per exemple, quan hi ha problemes de bateria o quan la nostra aplicació ha detectat una pèrdua de connectivitat.

Exemple de patró de missatge d'alerta

Es recomana:

- Usar-ho només per a missatges importants i necessaris. Aquest tipus de missatges són molt intrusius i en cas d'aparèixer excessivament poden arribar a ser ignorats.

- En cas de ser necessari fer alguna acció, cal mostrar una de les opcions com a drecera per a aquesta acció. Per exemple, si hi ha un problema d'espai en disc, una de les accions ens anirà al gestor d'aplicacions instal·lades per eliminar les que no necessitem.

2.4.4. Implementació i proves (*testing*)

La implementació d'aplicacions per a dispositius mòbils s'assembla molt a la de la resta d'aplicacions, encara que, generalment, es tracta d'aplicacions més petites, o bé que tenen cicles de desenvolupament més curts que les aplicacions tradicionals. Això es deu tant a la naturalesa mateixa de l'aplicació o a les necessitats del mercat, que demana aconseguir prototips o proves de concepte ràpides.

Una particularitat de les proves és la necessitat de tenir un emulador o entorn de proves per a poder provar allò que estem desenvolupant. Aquesta necessitat provoca dificultats (no es pot fer la prova si no es disposa d'un dispositiu) i ser lentes (per la lentitud d'execució sobre els dispositius o emuladors). Per aquest motiu, s'acudeix a les proves unitàries, que permeten dividir el desenvolupament, amb la qual cosa es pot provar de manera desacoblada i desenvolupar parts de la nostra aplicació sense la necessitat de fer proves en l'emulador, com poden ser els accessos a bases de dades o bé la lògica de negoci.

Amb vista a la implementació, hi ha una sèrie de factors que prenen especial importància en el cas de les aplicacions per a dispositius mòbils, i seran primordials per a l'èxit del projecte:

- **Usabilitat**. La usabilitat de l'aplicació s'ha de tenir molt en compte (hem de tenir present que la majoria d'usuaris no tindran temps ni ganes de

llegir els manuals). Una bona pràctica per a millorar la usabilitat és adaptar les aplicacions als comportaments estàndard de la plataforma, de manera que l'usuari pugui aprofitar regles mnemotècniques o hàbits adquirits.

- **Responsivitat**. L'aplicació ha de respondre a les accions d'usuari tan bé com sigui possible i de manera àgil. És un punt molt important, ja que l'usuari de dispositius mòbils sol ser més exigent que el d'aplicacions tradicionals en situacions semblants. Algunes plataformes, com l'Android, són molt estrictes a l'hora d'aconseguir que l'aplicació sigui prou responsiva. Si l'aplicació no és prou àgil a l'hora de respondre l'usuari, el sistema l'ha d'avisar d'aquesta circumstància i permetre-li tancar-la immediatament.

Exemple de no-responsivitat

Una aplicació que, mentre du a terme una tasca costosa, bloqueja la interfície d'usuari i, a més, no l'informa del fet.

- **Optimització de recursos**. Els dispositius mòbils, fins i tot els actuals, tenen uns recursos molt més reduïts que les aplicacions de sobretaula o les pàgines web. Això vol dir que hem de fer un bon ús de la memòria i del processador del dispositiu, per la qual cosa és convenient tancar els recursos que no es necessitin per a evitar els problemes associats.
  També és important prestar molta atenció al consum de bateria de l'aplicació. Per això, és recomanable evitar càlculs excessius (com els càlculs de coma flotant), l'ús de les funcions de vibració o l'ús de les connexions sense fil. Cadascuna d'aquestes recomanacions s'ha d'adaptar a la plataforma específica on treballem, ja que els fabricants ens donaran les recomanacions específiques.
  Una altra bona pràctica, en aquest sentit, és perfilar l'aplicació amb les eines pròpies de la plataforma per a trobar problemes.

- **Accessibilitat de l'aplicació**. Hem de tenir en compte en dissenyar l'aplicació que els usuaris poden necessitar accedir-hi de diferents maneres. Per exemple, si tenim una aplicació amb formularis, és ideal que sigui accessible tant per mitjà de les pantalles tàctils com dels ratolins de bola (*trackballs*).

Nota

Hi ha eines que tradueixen l'aplicació a text, i la fan accessible a persones cegues. Per a això, s'ha de desenvolupar de manera correcta.

Importància de les proves unitàries

Les proves unitàries serveixen, com ja hem comentat, per a provar el funcionament correcte d'una part del codi. Aquestes proves tenen com a característiques més destacades que han de ser automatitzables (no és obligatori, però sí molt recomanable), completes, reutilitzables o repetibles al llarg del temps, independents entre si i tan professionals com el codi mateix.

Si la prova se centra en una part de l'aplicació que depèn del dispositiu en què s'executa, es denomina *prova d'integració*.

Les proves unitàries agiliten el desenvolupament perquè se centren en una part del desenvolupament i, per tant, no caldrà provar aquesta unitat dins de l'emulador o emuladors, sempre que es tracti d'una part realment unitària. A més, reforcen la fiabilitat del desenvolupament perquè es fan al mateix temps

que la prova que ho verifica. I els avantatges es multipliquen quan la prova és automatitzada, ja que evita l'aparició d'errors en un futur en provar l'aplicació de manera més exhaustiva.

Proves d'integració contextualitzades

Les proves que fem han d'estar contextualitzades, és a dir, pensant a reproduir el que realment ocorre a l'usuari quan utilitza la nostra aplicació. Llavors, no és el mateix provar un gestor de rutes per al cotxe, que provar una aplicació per a controlar les despeses de l'empresa, ja que s'usen en moments i amb objectius molt diferents.

Hi ha una sèrie de preguntes que s'ha de fer la persona que provi l'aplicació per a poder fer correctament aquestes proves, com per exemple:

- Quina és l'experiència d'usuari amb l'aplicació?
- Es carrega ràpidament, l'aplicació? Es necessita alguna barra de progrés de l'aplicació? I amb connectivitat reduïda?
- Canvia l'aplicació en moure el dispositiu? És a dir, s'han de tenir en compte els sensors d'acceleròmetres? Reacciona àgilment a aquests canvis?
- L'aplicació, accepta correctament les interrupcions, com per exemple les trucades? L'aplicació, acaba correctament, és a dir, tanca correctament tots els recursos utilitzats?
- En cas que utilitzi connectivitat o geolocalització, o bé qualsevol altre servei, sabem quin és el comportament esperat de l'aplicació davant de la falta d'aquesta capacitat?

Per a poder respondre aquest tipus de preguntes s'hauran de generar una sèrie de proves que portaran a contextualitzar-la. Sense aquestes proves, l'aplicació pot no funcionar bé en molts casos. La millor manera de dur a terme aquestes proves és, sens dubte, sobre els dispositius mateixos. El tipus, el nombre i la diversitat de les proves dependrà dels requisits de l'aplicació, però es fa totalment imprescindible haver fet proves amb aquests dispositius abans de poder distribuir l'aplicació.

Exemple

http://www.perfectomobile.com/ és un exemple de servei que permet provar aplicacions sobre dispositius reals, per mitjà d'una granja. A més, permet automatitzar-les, la qual cosa redunda en una millora de la qualitat.

Continuïtat de les proves

Com en qualsevol desenvolupament, és important fer proves en cada nou cicle, però en aquest cas ho és més que en altres entorns amb més facilitat de canvi, ja que generalment (i en especial amb aplicacions natives) els desplegaments de l'aplicació i de les actualitzacions no solen estar controlats pels desenvolupadors. Això vol dir que qualsevol canvi, error o mal funcionament

té més repercussió, i s'ha d'anar amb molta cura per a evitar la regressió de problemes ja solucionats. Una manera de combatre aquest problema és mantenint un pla de proves i executant-lo a cada nova versió.

En cas de tractar-se d'una aplicació web per a mòbils és molt més fàcil fer canvis. En una aplicació nativa les noves versions poden ser actualitzades per mitjà del sistema mateix, depenent de l'entorn operatiu (OC).

És important tenir sistemes que assegurin la fiabilitat i la no-regressió d'errors abans de llançar una nova versió. Tant si són automatitzats, com els sistemes d'integració contínua, com si són més manuals, com plans de proves complets i reproduïbles.

3. Negoci

Un dels punts que està afavorint el creixement del desenvolupament d'aplicacions mòbils és la facilitat per a poder-hi fer negoci. Portar una idea a una cosa rendible avui dia està molt més a prop de la mà que fa uns anys.

Això s'ha aconseguit gràcies a la millora del canal de distribució, principalment amb l'aparició dels mercats d'aplicacions, que en faciliten la distribució i compra.

Així anteriorment els negocis en dispositius mòbils es basaven en la venda directa d'aplicacions a mesura o en serveis de micropagament amb SMS. En canvi, avui dia han aparegut nous models de negoci que aprofiten la presència del nostre dispositiu mòbil en el dia a dia, com per exemple el pagament en aplicacions mòbil per béns virtuals, o el pagament per subscripció.

En aquest apartat veurem una classificació dels possibles negocis que es poden fer amb les aplicacions per a dispositius mòbils, per a entendre les possibilitats existents. Després veurem amb més detall com és algun d'aquests negocis, en especial aquells d'aparició recent. També veurem els passos necessaris per a aconseguir publicar aplicacions que es venen i distribueixen per mitjà de mercats d'aplicacions i les coses que són importants per a arribar a més usuaris.

3.1. Possibilitats de negoci

Hi ha moltes maneres de fer negoci amb les aplicacions mòbils. Entre els models clàssics de negoci amb el mòbil destacarem els següents, encara que cal tenir en compte que algun ha experimentat una caiguda en les vendes, en part per l'abús (principalment missatgeria) i la saturació del mercat:

- SMS prèmium. Missatges de text amb un cost extra, també coneguts com a *serveis de tarifació addicional* (STA).

- SMS per a baixar aplicacions o melodies.

- Desenvolupament d'aplicacions a mida.

En contrapartida, han aparegut nous models de negoci, entre els quals destaquem els següents:

Nota

El 2009 entra en vigor el codi de conducta dels STA que, entre altres mesures per a regular el sector, obliga tots els serveis basats en missatges prèmium a avisar l'usuari del cost (sempre que sigui més d'1,2 euros). Aquest fet es coneix com a *alta voluntària* (*opt-in*).

- Vendre l'aplicació per mitjà d'una botiga d'aplicacions. És el model més clàssic, encara que no sempre és el més eficaç.

Terminologia

Les aplicacions gratuïtes que tenen una versió de pagament se solen conèixer com a *versió lite* i, de vegades, la versió de pagament com a *versió pro*.

- Oferir versió gratuïta i amb la distribució aconseguir màrqueting de l'aplicació o dels serveis relacionats, com pot ser un web immobiliari.

- Oferir una versió gratuïta i una versió de pagament. La versió gratuïta no conté totes les funcionalitats de l'aplicació; d'aquesta manera, l'usuari pot provar-la i, si li convenç i vol obtenir les funcionalitats extra, hi haurà de pagar.

- Vendre publicitat. Diverses tecnologies permeten incloure en les aplicacions, tant en les basades en tecnologia web com en les natives, anuncis contextualitzats. De vegades la publicitat només apareix en la versió *lite*.

- Vendre paquets de continguts extres. Es tracta de vendre l'aplicació per parts, cadascuna amb un cost. Això és molt habitual en els jocs, per exemple, que ofereixen paquets amb diferents nivells.

- Estendre un negoci existent al món mòbil:
  - Millorant una línia de negoci, per exemple afegint la localització geogràfica al nostre servei.
  - Intentant aconseguir nous clients. Per exemple, aconseguir clients "compulsius", és a dir, aquells que si haguessin d'anar a l'ordinador no comprarien però tenint-ho a la butxaca sí que ho farien.

Exemple

Pensem, per exemple, en una aplicació d'informació meteorològica o de trànsit oferta via web que concreti la informació en funció de la localització de l'usuari.

- Construint una aplicació com un servei, com per exemple el Gootaxi, que permet accedir al servei de taxis per mitjà del mòbil.

Exemple

Pensem, per exemple, en informació turística durant un viatge, o en informació de descomptes del comerç on és l'usuari.

- Construint la nostra aplicació com una subscripció. Per exemple, la reproducció en temps real de partits de l'NBA que ofereix el portal http://www.nba.com.

- Mobilitzant una tecnologia existent. Per exemple, un CRM.

- Creant una aplicació que estengui un negoci en línia. Principalment aprofitant API públiques d'aquests negocis en línia.

- Aplicacions amb sistemes *pay-in*. Això significa que l'aplicació ens pot demanar pagaments en canvi d'opcions de l'aplicació mentre s'està executant. Això pot funcionar per a aplicacions com jocs, per a comprar elements que ajudin el joc, o aplicacions de modelatge, per a comprar elements especials, o aplicacions normals que demanen pagaments per a serveis addicionals.

Segons el públic objectiu de l'aplicació, l'acceptació que sigui de pagament serà molt diferent: hi ha plataformes en què és probable que l'usuari estigui disposat a pagar i unes altres en què la publicitat és el millor mitjà per a aconseguir ingressos per l'aplicació.

A continuació veurem amb més detall dos dels models apuntats en aquest apartat:

- El model d'aplicació gratuïta
- El model de pagament directe o indirecte

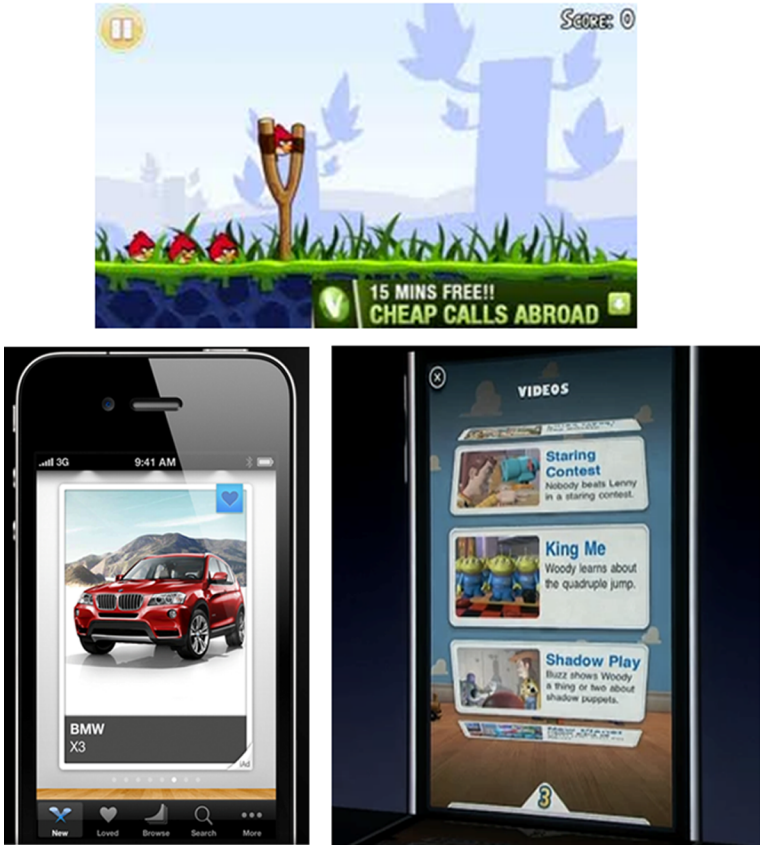
3.1.1. Model d'aplicació gratuïta

Hi ha molts tipus d'aplicacions gratuïtes, la majoria senzillament pretén atreure usuaris perquè acabin comprant la versió de pagament de l'aplicació o productes relacionats.

Terminologia

Aquest model d'aplicacions rep el nom de *freemium*.

Una altra tipologia d'aplicació gratuïta, a la qual també ens hem referit abans, és la que obté els beneficis a partir de la publicitat, generalment contextualitzada. Aquest model de negoci és molt proper al de les pàgines web que obtenen beneficis pel nombre de visites o pel nombre de clics en els anuncis que incorporen. Hi ha moltes empreses que ofereixen aquesta publicitat i les plataformes mateixes incorporen suport directe, com per exemple iAd per a iOs, adMob per a Android, Microsoft Advertising Exchange per a Windows Phone 7 o Blackberry Advertising Service per a Blackberry, entre d'altres. I moltes es poden usar en altres plataformes, la qual cosa fa una mica més fàcil portar les aplicacions. També hi ha plataformes de publicitat específica per a dispositius mòbils que ofereixen la publicitat en el format correcte i amb més contextualització.

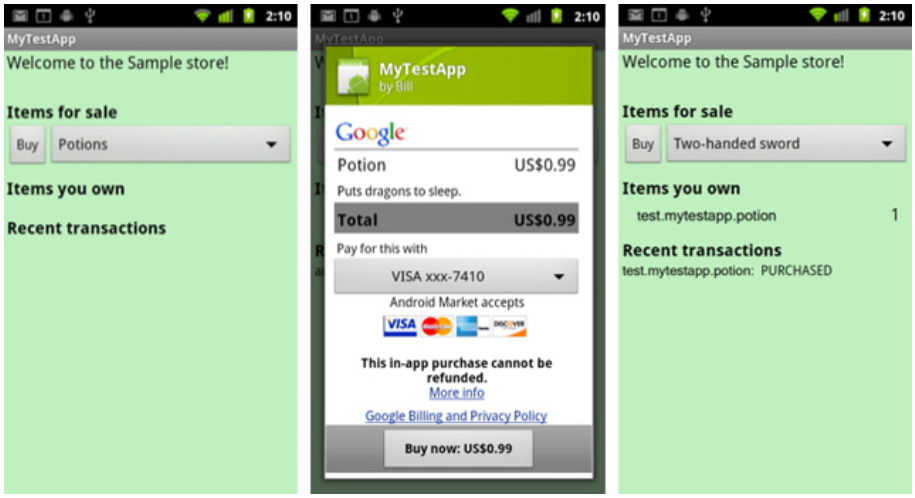


Exemple de publicitat en aplicacions per a dispositius mòbils

Aquest tipus de negoci té un benefici ocult, que és el seguiment (*tracking*) dels usuaris de l'aplicació. Aquesta informació variarà segons la plataforma de publicitat utilitzada, però sempre ajuda a definir el perfil dels usuaris, la qual cosa redunda immediatament en possibilitats de millora de l'aplicació.

3.1.2. Pagament directe o indirecte

Les aplicacions que s'han de pagar, ja sigui en el moment de la descàrrega o bé en el moment d'utilitzar algun servei concret, formarien aquest grup, que també engloba les aplicacions que tenen una versió gratuïta i una altra de pagament.



Exemple d'aplicació de tipus *pay-in*

El primer pas per a aconseguir que una aplicació pugui ser venuda una vegada desenvolupada és incloure-la en una botiga d'aplicacions. Actualment hi ha moltes botigues oficials, tant de les plataformes com dels fabricants (si són diferents) i també botigues alternatives o extraoficials.

Les exigències per a poder incloure una aplicació en una botiga poden variar molt segons la botiga; el que sembla indubtable és que el nombre de compradors potencials sol ser més elevat com més gran són aquestes exigències: en una botiga oficial l'aplicació haurà de passar uns controls de qualitat que, segurament, no tindrem en una botiga alternativa. Evidentment, el nombre de compradors potencials d'una botiga oficial és molt més gran que el d'una botiga alternativa.

En resum, si pretenem que la nostra aplicació tingui una perspectiva de descàrregues acceptable, l'haurem d'incloure en alguna de les botigues oficials esmentades abans, i per a això haurem de complir uns requisits:

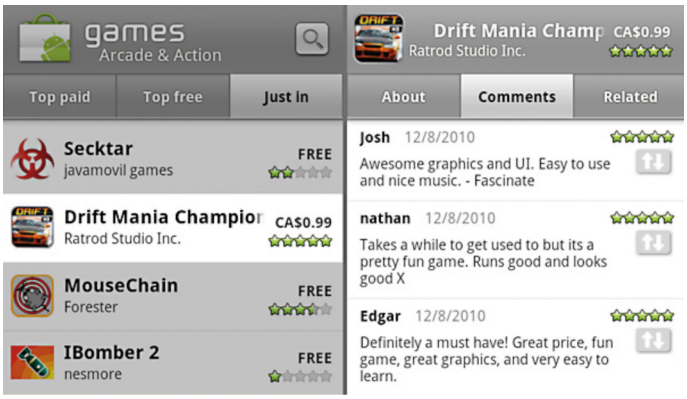
- Registrar-nos, i generalment pagar, per a aconseguir un compte de desenvolupador que ens acrediti com a responsables de les aplicacions.

- Pujar l'aplicació a la botiga per mitjà de la zona de desenvolupador.

- Superar el procés d'aprovació, la qual cosa pot trigar dies o setmanes. En totes les botigues, és possible que la nostra aplicació sigui eliminada en cas de causar problemes o bé incomplir alguna de les normes d'aquesta botiga.

Una vegada l'aplicació és a la botiga, podrem controlar la informació relativa a aquesta, com el nombre de descàrregues, comentaris, valoracions, etc.

Hem de tenir present que l'èxit de les aplicacions depèn que els usuaris les puguin trobar fàcilment, per la qual cosa és primordial situar-se en la part alta de la llista d'aplicacions de la botiga, i ocupar aquestes posicions sol estar associat a les descàrregues i a la valoració dels usuaris, de manera que és molt important tenir en compte les seves opinions.



Exemple d'opinions sobre les aplicacions

Resum

L'objectiu principal d'aquest mòdul didàctic ha estat veure i aprendre sobre les peculiaritats del desenvolupament d'aplicacions per a dispositius mòbils i, per a això, hem presentat tant les principals dificultats com els possibles beneficis derivats del tipus de dispositius sobre el qual s'executaran aquestes aplicacions i s'ha aprofundit en les estratègies de desenvolupament possibles.

Pel que fa a les dificultats, hem destacat especialment la fragmentació, i se n'ha estudiat l'origen i com cal combatre-la. Quant als beneficis, hem vist que la contextualització de l'aplicació i la ubiqüitat són els dos grans punts que cal tenir en compte.

També hem vist algunes pinzellades sobre els mètodes utilitzats en projectes orientats a aconseguir aquest tipus d'aplicacions, amb els seus punts forts i punts febles. Hem parat esment a fases que són especialment delicades en aquest tipus de projectes, com el disseny de la interfície i la selecció del tipus d'aplicació (web, nativa, etc.), o fins i tot úniques per a aquest tipus d'aplicacions, com són la distribució basada en *market places*.

Finalment, hem presentat les possibilitats de negoci per a aquest tipus d'aplicacions.

Activitats

1. Avalueu l'estat de l'art de les aplicacions mòbils en 3D, investigant les opcions de desenvolupament, i les API que hi ha per a dues de les plataformes següents:

a) Android
b) iPhone
c) Java ME
d) BREW
e) Symbian

2. Instal·leu i desenvolupeu una aplicació qualsevol amb, almenys, dues eines de desenvolupament multiplataforma: *phonegap*, *rhomobile*, *wapps*, *appcelerator*, etc.

3. Planifiqueu un projecte d'una aplicació senzilla per a gestionar tasques compartides. La planificació ha d'incloure temporització, recursos humans i materials (dispositius, per exemple).

4. Desenvolupeu un pla d'implantació de dispositius per als casos següents:

a) Una aplicació per a llegir signatures en documents oficials. Les signatures es fan amb el dit o un bolígraf sense punta sobre la pantalla del dispositiu.

b) Una aplicació per a mostrar la situació dels nostres amics d'una xarxa social i seu últim comentari.

5. Identifiqueu entre les aplicacions que useu diàriament quines podrien ser mòbils i quines no. Expliqueu-ne els motius.

6. Esbrineu els costos associats al pagament de les aplicacions.

7. Comptabilitzeu el nombre total d'aplicacions venudes i els beneficis generats per a la plataforma i per als desenvolupadors de tres de les principals plataformes del mercat amb botigues d'aplicacions.

8. Indiqueu com es pot evitar la fragmentació amb iOs.

9. Expliqueu com funciona la gestió de la memòria cau del navegador en l'HTML5.

10. Expliqueu la diferència que hi ha en l'HTML5 entre base de dades local i gestió fora de línia de l'aplicació.

11. Especifiqueu els components que s'utilitzen per a aconseguir aplicacions web mòbils natives. Escriviu un exemple per a carregar la pàgina principal de la UOC.

12. Indiqueu quines són les possibles fonts de fragmentació actuals.

13. Enumereu les fonts de fragmentació que s'utilitzen en una aplicació de visualització de fotografies per a dispositius Android.

14. Poseu un exemple d'aplicació LBS en el context de la restauració.

Glossari

accessibilitat *f* Grau en el qual les persones poden utilitzar un objecte, visitar un lloc o accedir a un servei, independentment de les seves capacitats tècniques, cognitives o físiques.

API (*application program interface*) *f* Interfície exposada per a ser utilitzada dins d'una aplicació amb l'objectiu de donar accés a biblioteques o funcions externes a l'aplicació.

connector (*plug-in*) *f* Aplicació afegida a una aplicació principal per a ampliar o canviar-ne la funcionalitat.

derivació de programari (*device plan*) *f* Modificació sobre el programari original per a poder adaptar-se als canvis de la fragmentació.

ecosistema mòbil *m* Conjunt d'actors necessaris per a poder tenir els dispositius mòbils i finalment les aplicacions per a aquests dispositius. En concret en aquest ecosistema s'inclouen les operadores de telecomunicacions, els fabricants de maquinari i els elements de programari que intervenen en l'execució de l'aplicació.

entorn de treball d'aplicacions *m* Marc de desenvolupament que permet fer aplicacions de manera més senzilla, ordenada i mantenible.

escenari d'una aplicació *m* Cas de fragmentació a causa d'una o diverses de les possibles causes de fragmentació.

fragmentació *f* Situació, o condicionants de la situació, que no permet compartir una mateixa aplicació entre diferents ecosistemes sense fer-hi les adaptacions pertinents.

freemium *m* Model de negoci gratuït en contraposició al model prèmium.

CSS (*cascade stylesheet* o full d'estil en cascada) *m* Llenguatge usat per a definir la presentació d'un document estructurat amb XML o HTML.

CRM (*customer relationship management* o gestió de la relació amb els clients) *f* Programari per a gestionar la relació amb els clients.

IDE (*integrated development environment*) *m* Entorn de desenvolupament que incorpora totes, o gairebé totes, les eines necessàries (eines de modelatge o disseny, eines de depuració, etc.).

inverse of control *m* Patró de disseny utilitzat per a definir les dependències des d'un contenidor extern.

iOs *m* Sistema operatiu per a dispositius mòbils de l'iPhone.

itinerància (*roaming*) *f* Capacitat d'un dispositiu de moure's d'una zona de cobertura a una altra.

Javascript *m* Dialecte de l'estàndard ECMAScript, utilitzat per a donar dinamisme a les aplicacions web.

llenguatge de marcatge o llenguatge de marques *m* Llenguatge que estructura la informació per mitjà de marques o etiquetes (*tags*).

LBS (*location based service*) *m* Servei que intenta oferir un valor afegit gràcies al coneixement de la ubicació geogràfica de l'usuari.

mash-up *f* Aplicació que aprofita API existents a la Xarxa per a interactuar amb dades, com pot ser l'API de Twitter, o dades públiques de l'estat.

mètode *m* Definició dels passos que cal seguir per a aconseguir un objectiu.

MoSoSo (*mobile social software*) *m* Programari amb una capa social afegida per a aprofitar les connexions socials de l'usuari.

MMS *m* *Multimedia message system*.

OTA (*over-the-air*) *m* Mètode per a distribuir actualitzacions o altres funcions accessibles per mitjà d'Internet.

pla de dispositius (*device plan*) *m* Llista ordenada de tots els dispositius o grups de dispositius que volem suportar.

preprocessador *m* Eina que permet fer canvis sobre el codi amb l'objectiu d'adaptar-lo a unes necessitats específiques.

prova unitària *f* Prova que serveix per a comprovar el funcionament correcte d'una part del codi. És recomanable que aquest tipus de proves siguin automatitzables, completes, reutilitzables (repetibles al llarg del temps), independents entre si i tan professionals com el codi mateix.

responsivitat *f* Capacitat de resposta d'una aplicació davant les accions d'usuari.

SDK (*software development kit*) *m* Conjunt d'eines necessàries per a poder fer el desenvolupament d'aplicacions en una plataforma.

SGML *m* *Standard general markup language*.

targeta intel·ligent (*smart card*) *f* Targeta amb un circuit integrat de mida butxaca en què es pot programar algun tipus de lògica. Un exemple són les targetes de crèdit amb microxip.

TDD (*test driven development*) *m* Desenvolupament dirigit per les proves: abans de fer una funcionalitat hi ha d'haver una prova que en verifiqui el funcionament.

ubiqüitat *f* Capacitat d'accedir a tota la informació o serveis que necessita l'usuari en qualsevol moment i circumstància, per mitjà del dispositiu que tingui actualment.

usabilitat *f* Facilitat amb què les persones poden utilitzar una eina particular o qualsevol altre objecte fabricat per humans amb la finalitat d'aconseguir un objectiu concret.

W3C (World Wide Web Consortium) *m* Consorci internacional que produeix recomanacions, que no estàndards, per al World Wide Web (WWW).

WHATGW (Web Hypertext Application Technology Working Group) *m* Grup format per les empreses responsables dels principals navegadors web: Apple, Opera, Mozilla, Microsoft i Google.

WML *m* *Wireless markup language*.

WURLF *m* Repositori d'informació que permet identificar les capacitats i limitacions d'un dispositiu mòbil per mitjà de la metainformació d'una petició.

Bibliografia

Ahonen, Tomi (2007). *Mobile the 7th Mass Media is to internet like TV is to radio*.

Allen, Sarah; Graupera, Vidal; Lundrigan, Lee (2010). *Pro Smartphone Cross-Platform Development*. Apress.

Blanc, Pablo; Camarero, Julio; Fumero, Antoni; Werterski, Adam; Rodriguez, Pedro (2009). *Metodología de desarrollo ágil para sistemas móviles*. Universitat de Madrid

Fling, Brian (2009). *Mobile Design and Development*. O'Reilly.

Lehtimaki, Juhani. "Android UI Design Patterns".

Rajapakse, **Damith C.** (2008). *Fragmentation of mobile applications*

Spataru, Andrei Cristian (2010). *Agile Development Methods for Mobile Applications*. Universitat d'Edimburg.

Virkus, R.; Gülle, R.; Rouffineau, T. i altres (2011) *Don't panic Mobile Developer's guide to Galaxy*. Enought Software Gmb H + Co. KG.

Wong, **Richard** (2010), *In Mobile, Fragmentation is Forever. Deal With*.

Woodbridge, Rob (2010). 9 Mobile Business Modesl taht yo ucan use right now to generate revenue.

Enllaços d'Internet

http://www.comp.nus.edu.sg/~damithch/df/device-fragmentation.htm

http://techcrunch.com/2010/03/04/mobile-fragmentation-forever/

http://communities-dominate.blogs.com/brands/2007/02/mobile_the_7th_.html

http://www.versionone.com/pdf/mobiledevelopment.pdf

http://www.adamwesterski.com/wp-content/files/docsCursos/Agile_doc_TemasAnv.pdf

http://www.inf.ed.ac.uk/publications/thesis/online/IM100767.pdf

http://developer.smartface.biz/documents/Application_Development_Methodology.pdf

http://untether.tv/ellb/blog/8-mobile-business-models-that-you-can-use-right-now-to-generate-revenue/

http://en.wikipedia.org/

http://www.mit.jyu.fi/opetus/kurssit/jot/2005/kalvot/agile%20sw%20development.pdf

http://developer.android.com

http://developer.apple.com