TREBALL DE FI DE CARRERA

TÍTOL DEL TFC: Codis QR i Android

TITULACIÓ: Enginyeria Tècnica de Telecomunicació, especialitat Telemàtica

AUTOR: David Hernàndez Parramón

DIRECTOR: Dolors Royo Vallés

DATA: 25 de febrer de 2011

Títol: Codis QR i Android

Autor: David Hernàndez Parramón

Director: Dolors Royo Vallés

Data: 25 de febrer de 2011

Resum:

Durant aquest projecte realitzarem una aproximació a Android, un dels sistemes operatius per a dispositius mòbils més populars del moment amb el permís de IOS, el sistema operatiu de Apple. Android és un sistema de codi obert i basat en Java, que ara mateix disposa d'una gran popularitat, gracies a tot això la seva comunitat ha crescut moltíssim, i la mateixa ha creat infinitat d'informació, tutorials i exemples, és per això que es tracta d'un sistema operatiu on la corba d'aprenentatge és molt assequible.

Em fet us, també, dels codis QR com a marcadors que ens permetrà obtindre més informació relacionada amb la classe a la que esta associat, mostrant-nos qui esta donat la classe a aquella aula i qui donarà la següent classe.

La idea principal d'aquest projecte és la de oferir a professors i alumnes una ajuda per conèixer més informació respecte al entorn que els rodeja, en aquest cas el Campus, mitjançant el seu dispositiu mòbils.

També podrem accedir per mitjà de l'aplicació a una base de dades de professors, assignatures i classes, que ens mostrarà informació sobre els mateixos, donant-nos informació extra respecte als mateixos.

Títol: Maqueta de TFC/PFC

Autor: David Hernàndez Parramón

Director: Dolors Royo Vallés

Data: 25 de febrer de 2011

Overview:

This project is to provide a tool for teachers and students to have a better understanding of the campus, using their Android device.

During this project we will do a aproximation to Android operating sistem, one of the most popular sistems in movile devices, at this moment together with IOS, Apple's operating sistem. Android is a open code sistem and Java based, that and his popularity created a strong scene and plenty of information about, and plenty of experimented users.

To acomplish our objective, used also the QR codes as marquers that will allow us obtain more info related with the class related with it, showing us the subject that is being done at that moment and who will do the next one.

Also, thanks to this aplication we will acces to a database of teachers, subjects and classes, and information related, it will be a reduced version of the virtual campus.

At the end of this project, we intend to do a aproximation at the design proces, that as enginiers are not completly aware, but of high importance due that simplifies our work helping to stablish a visual stile for our aplication, a comon and standarized look in the diferent screens of the aplication and to stablish the limits of it without having to code anithing.

Index de continguts:

Capítol 0: Introducció

Existeixen molts programes de posicionament al mercat, programes que ens permeten situar-nos a un espai desconegut i descobrir més informació sobre el nostre entorn, programes com ara Layar, GoogleMaps,.... Aquests programes existeixen per a pràcticament totes les plataformes del mercat, Windows, Mac, Android, IOS, Blackberry,... i fan us de GPS o el posicionament basat en la triangulació, així com ara solucions online o offline.

Som a l'era del web 2.0, on es important que el nostre dispositiu conegui la seva situació física per a compartir-la, i poder fer us de la basant social dels nostres dispositius, això comporta també una reducció a la nostra privacitat, però també obra un mon de possibilitats. Per tot això els nostres dispositius s'han omplert de sensors (com ara GPS o acceleròmetres) que permeten fer us de totes aquestes possibilitats.

Però per fer us d'un GPS és necessari no tindre gaire obstacles entre nosaltres i el satèl·lit. Aquest fet fa que ens sigui impossible usar el GPS per orientar-nos dins grans edificis, donat que normalment tindrem metres de ciment per sobre nostre.

És per això que els codis QR poden ser-nos útils, si ve no és tracta d'un sistema de posicionament, poden ser usats com a marcadors per a ser escanejats amb un mòbil amb càmera, que ens retornarà la informació associada, com ara davant de quina classe som o quina assignatura s'està donat.

Aquest treball pretén, per tant, donar una utilitat pràctica a la càmera del nostre dispositiu mòbil, i proveir a l'alumne, al professor o un visitant, d'una eina per a poder conèixer més informació sobre el seu entorn, en aquest cas la universitat o si fos necessari en qualsevol tipus de recinte on es faci necessari conèixer més informació sobre el que ens rodeja i no disposem de cobertura de GPS.

El programa que crearem haurà de ser capaç de, tocant l'opció desitjada, mostrar-nos l'objectiu de la nostra càmera i escanejar el codi QR, per immediatament després mostrar-nos l'informació associada.

Capítol 1: Tecnologies Emprades

El nostre objectiu és obtindre una aplicació mòbil per a obtindre més informació del nostre entorn i fer us dels codis QR, i per tant l'elecció de les eines que usarem és prou clara, però tot i això hem tingut que realitzar una serie de eleccions per definir sistema operatiu per al que treballarem, la base de dades que usarem i el sistema d'escaneig que integrarem. La estació de treball que usarem, serà un PC amb Windows 7, i Eclipse (aquest bé determinat per el sistema operatiu per a dispositius mòbils que usarem).

En els pròxims capítols veurem amb més detall totes aquestes eleccions.

1.1 Sistema operatiu:

La primera elecció és evidentment, el sistema operatiu, tenim clar que és tractarà d'un sistema operatiu per a dispositius mòbils, donat a que els requeriments de mobilitat que ens hem imposat ho exigeixen, però de totes maneres disposem de diferents opcions igualment interessants, Android, IOS, Blackberry OS, Windows 7,...

La primera elecció potser, seria el sistema operatiu més estès potser, o sigui IOS, el sistema operatiu d'Apple que implementa als seus dispositius mòbils (com ara el Iphone o Ipad), el problema és que és que es tracta d'un sistema força restrictiu, que només és pot programar de forma nativa per a IOS des de Mac, fet que ens condiciona donat que no disposem de cap estació de treball Apple, ni de cap dispositiu Apple de proves.

Si disposem en canvi d'estacions de treball PC (en el nostre cas amb Windows 7) i també un dispositiu Android, a més és tracta del segon sistema més estes a tot el mon, seguint molt aprop IOS. A més és un sistema obert basat en Java i amb una gran comunitat, fet que ens facilita l'obtenció d'informació.

Les altres opcions son igualment interessants però donat que no disposen de una base de usuaris i creadors tan gran i que a més son opcions més restrictives i que tampoc disposem de dispositius de provés, escollir Android ens va semblar la millor opció.

1.2 Base de dades:

La següent qüestió que ens varem plantejar va ser quin sistema hauríem d'usar per accedir a les dades necesaries per a la aplicació, teníem clar que tenia que ser una base de dades en SQL, donat que és el sistema més estès, que més s'usa a Android , i del que podem trobar més tutorials i documentació. Segons hem descobert durant el projecte, existeix una versió de SQL anomenada SQLite que és més lleugera per a dispositius mòbils, permeten millorar el rendiment general de l'aplicació que serà perfecte per a l'aplicació.

1.3 Escaneig de codis QR:

Un cop decidit el sistema operatiu i com emmagatzemarem les dades hauríem de decidir quin serà el mètode per a escanejar els codis QR. Podríem crear un sistema d'escaneig i descodificació d'imatges per el projecte, però això representaria dedicar tota l'extensió d'aquest per a la creació d'aquesta part i em preferit aprofundir més en Android. Per tant, per aquesta tasca hem optat per a una solució externa, que ens proporciona Google, anomenada Qrdroid que usa la llibreria Zxing, aquesta aplicació escanejarà els codis i els descodificarà per a que mes tard la nostra aplicació busquí l'entrada i mostri per pantalla l'informació associada.

Capítol 2: Android

2.1 Historia:

La marca Android va néixer al octubre de 2003 a Palo Alto, Califòrnia de la ma de Andi Rubin (**Fig 2.1**), Rick Miner, Nick Sears i Chris White, que per llavors no tenien res a veure amb Google. Cal tindre en compte que Rubin va treballar com a enginyer a Apple i que a més va crear WebTV mes tard adquirit per a Microsoft (actualment anomenat MSN TV) i Danger Inc., companyia també absorbida per Microsoft.

Fig. 2.1 Andi Rubin, un dels principals creadors de Android

L'agost del 2005 Google adquireix Android, per a diversificar i estendre els seus serveis online dins, el tot just, jove mon de els telèfons mòbils intel·ligents (o Smartphones), que gracies a la seva capacitat de procés i mobilitat començaven a fer l'us de programes socials o Web 2.0.

La primera Beta del SDK d'Android apareix el 12 de novembre de 2007, però no va ser fins el 23 de setembre de 2008 que es va alliberar, per fi, la versió 1.0. Un sistema operatiu per a dispositius mòbils basat en Linux i una versió modificada de Java.

Apple crea IOS (el sistema operatiu per a dispositius mòbils d'Apple) integrat dins el famós Iphone, el 9 de gener 2007, fet que ens mostra la competència descarnada entre els gegants del software i les telecomunicacions, podem veure que amb pocs mesos de diferencia presenten productes molt similars per intentar aconseguir el lideratge en el sector, disputat majoritàriament per Google, Apple i en menor mida Microsoft.

Google ha donat prioritat a aquest sistema operatiu, i sempre ha estat molt ràpid en la correcció de errors, i implementació de millores. Després de la versió 1.0 es segueixen tot de rapides actualitzacions (1.1, 1.5, 1.6, 2.0, 2.1, 2.2 i 2.3, 2.3.3) fins el 6 de desembre de 2010 on es alliberada la versió 3.0, la primera versió del sistema optimitzada per a tauletes (el primer IPAD va ser presentat el 27 de gener del mateix any).

Recentment va ser alliberada la versió 3.1, 3.2 i la versió 4.0 i 4.0.3 (ultima versió presentada a dia de la realització de aquest informe), aquestes dos ultimes per a Smartphones de ultima generació, amb suport per a microprocessadors de múltiples nuclis i amb processadors gràfics dedicats.

Android, es un sistema operatiu extremament adaptable, això i la proliferació de processadors de altes prestacions, baix cost i baix consum, ens permet instal·lar-lo en multitud de dispositius, telèfons intel·ligents, ordinadors, tauletes tàctils, televisors, i fins i tot neveres.

2.2 Fragmentació:

La versatilitat d'Android i tot aquest historial de versions, 15 versions en total en només 4 anys, i que l'adaptació de cada versió és responsabilitat del fabricant no de Google, provoca el problema de la fragmentació de versions, on existeixen una gran quantitat dispositius al mercat que moltes vegades no disposen de la ultima versió del sistema.

Tota aquesta fragmentació fa que els programadors de aplicacions hagin de tindre en compte tot tipus de requeriments multiplicant el treball ja de per si extens dels professionals, que si volen tindre una màxima distribució han de tindre en compte no només programar per a Android amb totes les seves versions, sinó també per IOS, el nou Windows 7, Palm OS,...

Donat que l'actualització dels sistemes operatius recau en mans dels fabricants de dispositius, i no en Google, provoca que si un fabricant prefereix destinar els recursos a un nou terminal els usuaris de aquell dispositiu no tindran la possibilitat de actualitzar, i per tant no tindran accés a les millores de la nova versió, limitant per tant l'accés a les noves aplicacions.

Segons la web oficial dedicada a creadors de aplicacions de Android que posa a la nostra disposició Google, la fragmentació de Android, a 5 de març de 2012, és de la següent manera:

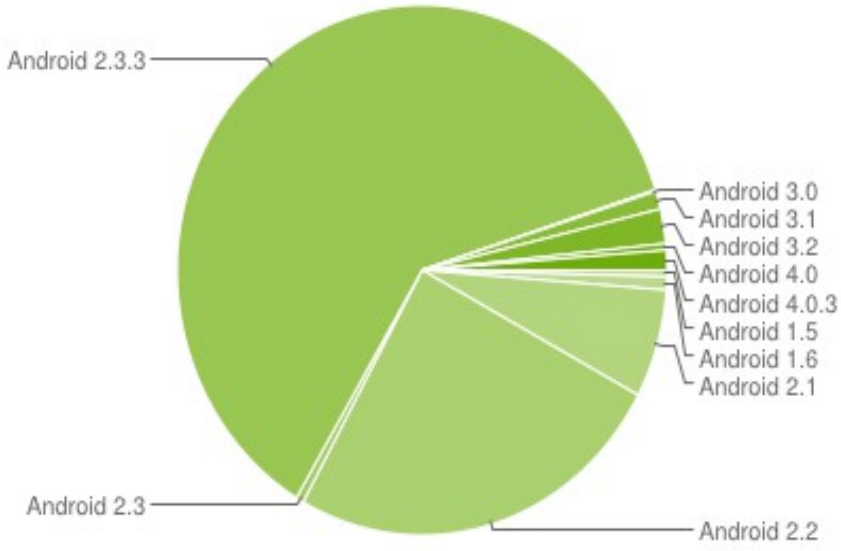


Fig. 2.2 Distribució de versions d'Android
en els terminals en actiu

Com podem comprovar a la distribució (**Fig 2.2**), el nombre més gran d'usuaris es troba actualment a la versió 2.3.3 i a la versió 2.2, tinguen entre les dues gairebé el 87% del total, però aquesta distribució varia amb cada nova versió, o amb la posada en mercat d'una nova generació de terminals, desplaçant-se molt cada pocs mesos.

A continuació veiem una distribució en el temps des de setembre del 2011 fins al març de 2012 de les diferents versions:

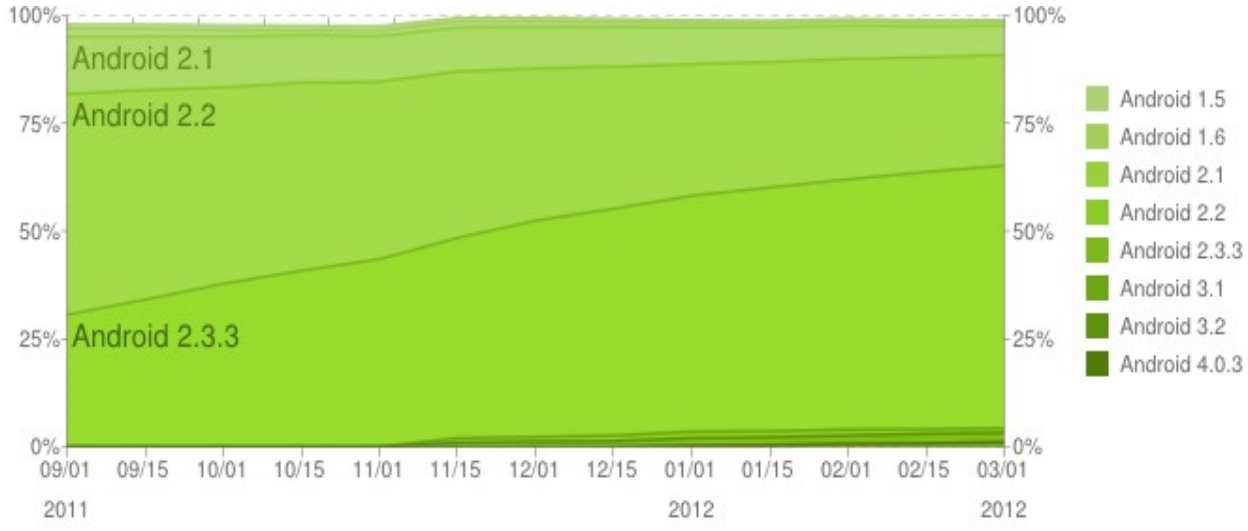


Fig. 2.3 Distribució temporal de versions

Com podem veure (**Fig 2.3**) les versions 2.x són les que els últims mesos han tingut una quota major, fet que pot canviar amb l'aparició i posterior implantació de la versió 4.x els pròxims mesos i l'adaptació gradual dels fabricants.

2.2.1 Mides i densitats de pantalles:

Si volem crear una aplicació que tingui el mes ampli espectre d'usuaris i que es vegi el millor possible en tot tipus de dispositius, apart de tindre en compte la fragmentació, seria convenient tindre en compte el espectre de mides i densitats de pantalla, donat que pot canviar molt la forma en que la nostra aplicació és usada i visualitzada. Per sort Android ha tingut en compte aquest fet i ha proveït als creadors d'un seguit d'ajudes per aquest fi.

El sistema operatiu escull en funció del dispositiu un dels repositoris de recursos que hem de crear, donat la gran varietat de definicions i densitats de pantalla que existeixen actualment s'han creat 4 mides prefixades (small, normal, large i xlarge) i 4 densitats prefixades (ldpi, mdpi, hdpi i xhdpi), de manera que ens proveïxen interficies més flexibles per poder usar en diferents tipus de dispositius.

A continuació veiem la distribució dels mides i densitats de pantalles (**Fig 2.4**):

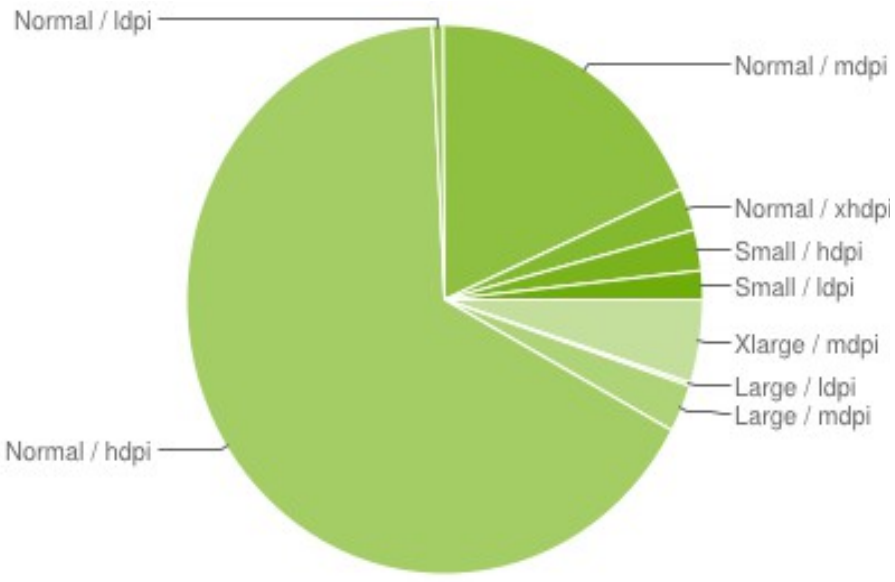


Fig 2.4 Distribució de les diferents densitats i mides de pantalles

Com podem observar la mida mes estesa es la Normal, amb les densitats per polsada son hdpi i mdpi, donat a un ajustat temps de producció si necessitem ajustar-nos a una

determinada mida aquests serien una bona opció donat que compten amb una quota de mercat de gairebé el 85%, per suposat que aquest percentatge varia tant ràpid com la quantitat de terminals que apareixen al mercat.

Aquesta distribució no varia amb tanta velocitat com ara la de versions de Android degut a la més “lenta” evolució de les pantalles i per tant no influeix tant, però es un fet que cal tindre en compte.

2.3 Conseqüències:

Tota aquesta variabilitat de versions, densitats i resolucions del sistema, i hardware (p.e: dispositius que disposen de acceleració gràfica i d'altres que no) fa que trobem aplicacions diferents per a diferents dispositius dins el Marquet del cadascun (el repositori online del sistema operatiu), donat que molts creadors prefereixen dedicar-se a determinat sistema i optimitzar-lo a crear-ho per tot tipus de dispositius.

En Android, a més, son els distribuïdors de dispositius els encarregats de adaptar el sistema al seu terminal, per tant quant hi ha una actualització de aquest per part de Google depengui de la voluntat i els recursos del fabricant l'actualització, fet que fragmenta més el mercat, ja que aquests solen crear “customitzacions” dels sistema, com ara HTC o Samsung.

La versió 3.0, optimitzada per a Tablets pc, va ajudar a limitar aquesta fragmentació, però no la va eliminar. Molts experts coincideixen que és el punt més negre d'Android i on Apple, el mes directe competidor d'Android, te les de guanyar. Apple compta també amb l'avantatge de ser ell l'encarregat d'actualitzar el dispositiu, fet que simplifica l'actualització.

De totes formes, el fet que sigui un sistema de codi obert, i que disposi d'una scene (comunitat que es dedica de forma desinteressada a desenvolupar el sistema) molt activa, fa que hagin aparegut a Internet versions modificades del sistema operatiu, anomenades ROMs, que no són més que adaptacions fetes per aficionats on prenen el paper de les companyies, i actualitzen a la ultima versió els seus dispositius, de aquesta manera podem tindre la ultima versió del sistema al nostre terminal, malauradament Google o els fabricants no es fan càrrec d'aquestes “actualitzacions” i per tant perdem cap mena de garantia sobre el terminal.

2.4 Contextualització:

Com a terminal de proves per al projecte hem usat un HTC Hero, amb la Rom, força popular dins la Scene, anomenada Salsasense 4.0, que integra la versió 2.3.3 d'Android i la versió 4.0 de Sense (**Fig 2.5**), interfície propietària de HTC que canvia força l'aspecte del sistema, afegint funcionalitats. Per a la creació del programa haurem d'activar l'opció “Depuración USB” dins “Ajustes/Aplicaciones/Desarrollo” que ens permet debugar aplicacions dins el terminal.

Fig 2.5 Interfície sense d'HTC

Donat que la versió al començament del projecte que te mes projecció actualment és la 2.1, i a més aquest integra certes funcionalitats extra a la càmera, com ara el zoom digital, que ens seran útils per a la captura dels codis QR, a més aquesta versió es completament compatible amb el terminal de provés.

Com a densitat de pantalla em usat la mdpi, primera, per ser una de les més esteses i segona per ser a la que correspon el nostre terminal, l'HTC Hero, que te una resolució de 320 x 480 pixels.

2.5 Fonaments d'Android:

No és l'objectiu d'aquest projecte servir com a manual sobre el sistema operatiu, però si que considerem que és important tindre en compte una serie conceptes dins la filosofia d'aquest sistema operatiu.

2.5.1 Components de una aplicació:

Dins Android existeixen 4 components fonamentals que cal tindre en compte donada la seva importància per a l'arquitectura de aquest sistema operatiu:

2.5.1.1 Activitat

Hem de tindre una idea clara de que es una Activity (activitat), una activitat es un component de l'aplicació que proporciona a l'usuari una pantalla on pot interactuar, i per on el dispositiu ens mostra la informació, com per exemple trucar, posar una cançó, veure un vídeo,...

Una aplicació normalment consisteix en una serie d'activitats unides d'alguna manera, la nostra aplicació per exemple consisteix en les següents activitats

2.5.1.2 Content Provider:

Un *Content Provider* es la forma que Android accedeix a informació estructurada, com ara una base de dades, es tracta d'una "porta" d'accés a l'arxiu (o servidor) que conte l'informació, aquí és on accedirem, obrirem, tancarem o actualitzarem la base de dades, a més és aquí on escriurem, en el nostre cas, les ordres de consulta de la base de dades. Els *Content Providers* encapsulen l'informació i proveïxen de mecanismes per a la seguretat en les dades que manegem.

Mitjançant un accés al nostre *Content Provider* tindrem al nostre abast la informació emmagatzemada a la base de dades que em creat per a l'us dins la nostra aplicació, dins el *Content Provider* fem la consulta SQL i emmagatzemem el resultat a una variable o en cas que sigui una llista en un vector.

Quant volem accedir a la informació d'un *Content Provider* hem d'usar un *Content Resolver* al context de l'aplicació. El *Content Resolver* es el que comunica l'activitat amb el *Content Provider,* una instancia de una classe que conte el Content Provider, per ser més exactes, un cop el *Content Provider* rep aquesta sol·licitud realitza les operacions demanades i retorna l'informació requerida al *Content Resolver*.

2.5.1.3 Serveis:

Un servei es un component que funciona en segon pla per a realitzar operacions que estaran funcionant molt de temps o treballar amb processos remots. Podríem fer-ho servir per exemple per a mantindre la musica funcionant d'un reproductor de musica mentre realitzem d'altres feines amb el dispositiu. També es tracta del component que hauríem de implementar per a realitzar consultes a una base de dades remota i esperar els resultats.

2.5.1.4 Broadcast Reciver:

Un *Broadcast Reciver* es un component que respon a un ampli ventall d'anuncis broadcast del sistema.

S'usa per a mostrar notificacions, com ara que la bateria s'esta esgotant, que ha arribat un missatge o que s'ha finalitzat la descarrega d'un arxiu.

2.6 Cicle de vida d'una aplicació a Android:

Degut a la limitació en els recursos dels dispositius que gestiona Android (i també per ser el mes eficients possible), Android treballa d'una forma una mica diferent a la de sistemes operatius mes “tradicionals”. Les Activitats són tractades en una “pila d'Activitats”. Quant una Activitat és iniciada per l'usuari, és posada directament a sobre de tot de la pila (al onCreate()). Si existeix una activitat prèvia, aquesta ocupara el segon lloc a la pila i no tornarà a ocupar el primer lloc mentre la primera Activitat existeixi.

Una Activitat te bàsicament 4 estats:

- Si una activitat esta en primer pla de la pantalla, al primer lloc de la pila d'activitats, esta activa o corrent (*active* o *running*).

- Quant una Activitat perd protagonisme, per exemple quant una Activitat que no ocupa tota la pantalla o una Activitat amb un fons transparent, l'Activitat esta en *pausa (pause)*. Això vol dir que la Activitat esta completament viva (manté l'estat i la informació de membre i es manté unit al gestor de finestra), però en cas que el sistema es quedí sense memòria podrà tancar les aplicacions en *pausa*.

- Si una Activitat esta completament tapada per una altra Activitat aquesta Activitat es troba parada o Stopped. Manté tota la informació de membre i d'estat, però no es visible per l'usuari i moltes vegades serà matada per el sistema en quan necessití memòria per alguna altre feina.

- Si una Activitat es pausada o parada (Paused o Stopped), el sistema pot preguntar si l'usuari vol finalitzar l'Activitat o el sistema pot simplement matar el procés. Quant l'usuari torna a l'activitat, ha de ser completament reiniciada i restaurada al seu estat previ.

El següent esquema (**Fig 2.10**) representa el cicle de vida de una activitat, els quadres de colors són els estats en que una aplicació pot estar, i els quadres blancs són els mètodes que podem usar com a programador per a controlar tot el procés.

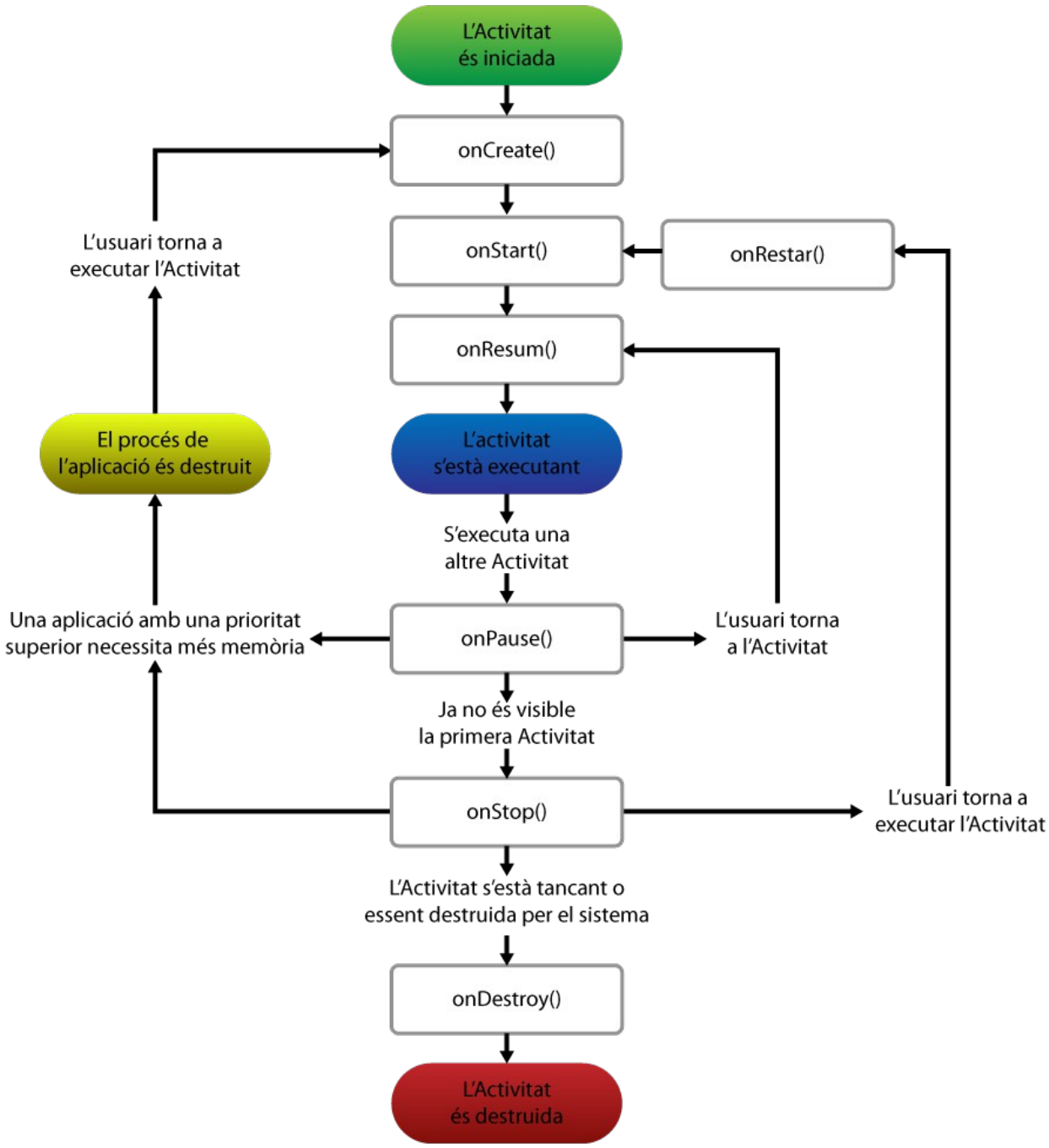


Fig 2.10 Cicle de vida d'una aplicació en Android

Tot el cicle de vida de l'activitat succeeix entre el mètode onCreate() i el mètode onDestroy(). Al estat onCreate() hem de fer tota la configuració de recursos i al estat onDestroy() em de alliberar-los. Per exemple, si em de descarregar informació de la xarxa i ho em de fer en segon pla, crearem el threat al onCreate(), i el pararem al onDestroy.

El cicle de vida visible per a l'usuari passa entre la crida onStart() i la crida al onStop().

Quant entrem a l'estat onStart() l'usuari veu la primera “pantalla” de l'aplicació, i quant el usuari surt de l'aplicació entrem al estat onStop() (quant el usuari pitja la tecla de tornar enrere), però si l'usuari torna a executar l'aplicació passarem a l'estat onResume() i seguidament a l'estat onStart().

Tot aquest procés permet al desenvolupador un major control dels recursos i quant aquests s'alliberen i tan mateix permet al sistema operatiu alliberar recursos que estan a l'estat onStop() o passar de l'estat onPause a l'estat onStop() quant sigui necessari. En cas que l'aplicació sigui destruïda (onDestroy()) i l'usuari torna a executar l'aplicació tornarà a l'estat onCreate() en comptes de l'estat onRestart().

3.3 Bones pràctiques

Per finalitzar, Google posa a la nostra disposició una serie de documents que fan referencia a bones pràctiques a l'hora de programar i dissenyar per Android, en la mida del possible em fet us d'aquesta documentació i l'hem aplicat a la nostre aplicació. A continuació fem esment de algunes d'aquestes pràctiques:

3.3.1 Disseny per a una major compatibilitat

Android, com ja em comentat és un sistema dissenyat per a funcionar en múltiples dispositius amb diferents hardwares, tals com diferent pantalla, amb o sense càmera, amb o sense GPS, ... Si volem que la nostra aplicació tingui el més ampli espectre de potencials usuaris haurem de tindre tot això, Android ens facilita aquesta tasca, però tot i amb això haurem de crear la nostra aplicació (en la mesura del possible) per a que això no sigui un obstacle per als usuaris.

Tenim diferents eines per a obtindre una major compatibilitat, com ara que Android pot disposar de diferents “jocs” d'imatges que usarà depenguen de la mida de la pantalla del dispositiu que esta executant l'aplicació.

Si el que volem fer es limitar els usuaris que poden executar l'aplicació que no disposin del Hardware necessari, podem especificar els requeriments dins el manifest de l'aplicació.

3.3.2 Disseny per a un millor rendiment

Donat que els dispositius mòbils tenen uns recursos inferiors (tot i que sembla que cada cop estan més propers) als dels típics ordinadors de sobretaula o portàtils (tot i que la diferencia entre uns i altres és fa fent mes petita), cal dissenyar les aplicacions de manera que facin el menor us d'aquest recursos, per poder d'aquesta manera distribuir la nostre aplicació a un ventall major de dispositius. A més una aplicació optima a nivell de rendiment farà un menor us del processador, que vol dir un menor us de la bateria, un bé

molt escàs i preuat en aquests dispositius.

3.3.3 Disseny per a una millor accessibilitat

Existeixen usuaris amb necessitats especials que fan que hagin d'interactuar de manera diferent amb els dispositius Android, per tant seria recomanable realitzar les aplicacions tenint en compte els següents preceptes.

- Permetre i controlar la navegació amb el controlador direccional.
- Etiquetar tots els nostres Widgets d'entrada.
- Seguir les regles de bones practiques per a disseny d'interfícies.

3.3.4 Disseny per a una major sensibilitat

Si durant la execució d'una aplicació passen llargs períodes de carrega, és queda penjada o congelada durant llargs períodes o triga massa a l'hora de executar cert procés poden passar dues coses, o que el propi sistema operatiu mostri un missatge de “Aquesta aplicació no esta responent” (avís ANR) o que el usuari se'n afarti i vulgui finalitzar l'aplicació.

Normalment les aplicacions funcionen en un sol thread (el main), això vol dir que qualsevol aplicació que estigui durant massa temps executant el thread main llençara un avís ANR. Per evitar això caldria en la mesura del possible fer us dels estats del cicle de vida d'Android. Però en cas que necessitem que un thread treballi durant el temps que necessiti sense llençar un ANR, podem fer us de Thread.wait() o Thread.sleep.

Si el que volem es que el usuari no es cansi d'esperar existeixen certs “trucs” que passem a comentar:

- Si l'aplicació esta fent cert treball en segon pla, podem fer us de una barra de progrés que ens informa del percentatge de informació processada, d'aquesta manera donem informació a l'usuari de quant porta esperant i quant de temps més haurà d'esperar.
- Per a jocs, en concret, fer càlculs de cada moviment en un thread fill diferent, així si es bloqueja un thread, podrem continuar executant els altres.
- Si l'aplicació te un alt temps de carrega a l'inici podríem considerar el fet de mostrar una pantalla de benvinguda. Tot i amb això em d'indicar en la mesura del possible a l'usuari la raó per la qual l'aplicació no respon.

3.3.5 Disseny per una major uniformitat

Encara que la nostra aplicació sigui sensible als temps de carrega o suficientment rapida, certes decisions en el disseny poden arribar a esser un problema per al usuari (com ara interaccions amb altres aplicacions o missatges, pèrdues d'informació, bloquejos

inesperats, etc...). Per evitar tot el possible aquests casos em de tindre en compte els següents punts:

3.3.5.1 Guardar l'informació

Posem per cas que l'usuari estigui editant informació i una trucada arribi al telefon mòbil, si no guardem aquesta informació fent us dels mètodes OnSavedInstanceState() o Onpause() aquesta és perdrà.

3.3.5.2 No exposar la informació original

Si exposem l'informació original sempre correm el risc de que es perdi, és per aquesta rao que el més recomanable és l'us de Content Providers, a més es la manera més endreçada de realitzar una conversió de formats.

3.3.5.3 No interrompre l'usuari

Si l'usuari executa una aplicació (com ara la del telefon per a trucar) podem assumir que ho ha fet a propòsit. És per això que em de evitar en la mesura del possible que la nostra activitat n'executi una altra a menys que l'usuari ho vulgui, i per tant només en cas de que rebem un input de l'usuari.

3.3.5.4 Tens molt a fer? Fes-ho en un Thread

Si la nostra aplicació ha de realitzar càlculs molt costosos o que portin molt de temps, ho hauríem de moure a un Thread. Això evitara el avís de “l'Aplicació no esta responguen”.

3.3.3.5 No sobrecarregar una activitat

La majoria de aplicacions que valen la pena disposen de múltiples activitats. Amb això volem dir que no cal posar informació o opcions que no necessitem

3.3.4 Disseny per a una millor seguretat

Aquests són alguns dels mètodes que integra Android per ajudar als creadors a crear aplicacions segures:

- El Sandbox per a aplicacions d'Android aïlla la informació i el codi en funció de cada aplicació.
- El Framework de les aplicacions d'Android disposa d'implementacions robustes de seguretat com ara criptografia, gestió de permisos, i una comunicació segura entre

processos dins una aplicació.

- Integra les següents tecnologies, ASLR, NX, ProPolice, safe_iop, OpenBSD dlmalloc, OpenBSD calloc, i Linux mmap_min_addr per a mitigar riscos associats amb errors comuns de gestió de memòria.
- Un sistema d'arxius encriptat que es pot activar per a protegir informació en casos de pèrdua o robatori del dispositiu.

Capítol 3: Codis QR

3.1 Introducció:

Podem definir el codi QR, com a una evolució dels codis de barres (**Fig 3.1**), que eren (i encara ara són) una representació gràfica d'una informació associada a un producte o objecte, per d'aquesta manera, amb un escàner d'infrarojos poder obtindre la informació associada. Els codis de barres són àmpliament usats a supermercats, magatzems o per a emmagatzemar productes, consultar el seu preu i obtindré més informació sobre el mateix.

Fig. 3.1 Codi de barres tradicional

La diferencia més evident entre els codis QR (**Fig 3.2**) i els codis de barres, és el seu aspecte, podríem dir així, que els codis de barres tradicionals son unidimensionals, on cada número és codificat per un nombre determinat de barres blanques i negres, però en canvi els codis QR, són bidimensionals, on tenim una superfície tal com un quadrat on s'hi representen petits quadres (no necessàriament negres, donat que existeixen versions en color) sobre una superfície plana, on cada petit quadre representa un 1 i un espai blanc representa un 0, donant-nos la possibilitat de encabir-hi mes informació, atès que fa un us òptim de l'espai.

Fig. 3.2 Codi QR

Els nous telèfons intel·ligents o tauletes tàctils disposen d'una capacitat de procés cada cop superior, i pràcticament tots disposen a més d'una càmera integrada al dispositiu, per tant són capaços de realitzar una captura del codi amb la càmera, realitzar un anàlisis visual i descodificar-lo de manera eficaç (codis de barres o QR), donant-nos així la capacitat a qualsevol usuari que disposí d'aquests dispositius una eina per a veure aquesta informació associada. El dispositiu accedirà d'aquesta manera a una base de dades on està continguda la informació associada, i pintarem per pantalla l'informació associada al codi QR.

Un us alternatiu dels codis QR és el de marcadors virtuals (**Fig 1.3**), una “guia” per al dispositiu que permet fer un anàlisis del que veu el dispositiu, i per tant oferir una capa d'informació al respecte, o integrar objectes 3D dins el visor de la càmera. Aquest tipus de marcadors s'usen moltes vegades per realitat augmentada.

Fig. 3.3 representació 3D on s'usa un codi QR com a marcador de referencia

Capítol 4: Aplicatiu de la EEUTIC

4.1 Objectius

L'objectiu d'aquest projecte és crear una aplicació que mitjançant codis QR ens doni informació extra del campus de Castelldefels, per d'aquesta manera tindre una guia del mateix al nostre dispositiu, i obtindre d'aquesta manera un grau de mobilitat extra al no tindre que dependre del nostre portàtil o ordinador.

Pretenem crear una versió simplificada del campus digital, on podem consultar informació relativa a professors, aules o assignatures, així com també un programa que no depenguí del GPS per donar-nos informació extra del nostre entorn, dispositiu completament inútil en interiors, degut a que requereix d'una enllaç directe entre nosaltres i el satèl·lit, sense obstacles físics.

4.2 Introducció

Per a aquest projecte pretenem simular també el procés de creació d'una aplicació a nivell professional, això vol dir:

- Crear un esborrany del projecte que cobreixi les necessitats del client, sense posar-nos a programar encara, per tindre una idea aproximada.

- Fer una llista dels requeriments d'aquest, on realitzarem una recerca de informació i recursos que seran necessaris per al nostre projecte, el nostre cas, quins programes necessitem per a la creació.

- Dissenyar un layout amb l'ajuda d'un programa de disseny, com ara Photoshop, per a obtindre una referencia visual del mateix, crear els recursos visuals necessaris per a l'aplicació i com a ajuda visual per a que el client pugui fer-se una idea de com serà aquesta aplicació.

- Crear una primera versió operativa del nostre aplicatiu que integri les opcions desitjades.

- Un cop tinguem una primera versió de l'aplicació haurem de portar a terme una fase de testeig per a poder detectar errors i possibles modificacions dins la nostra aplicació.

4.2.1 Esborrany

En aquesta fase és tracta de dimensionar en certa mida el nostre projecte, veure quins son els requeriments i llistar-los de manera que puguem fer-nos una idea de l'abast del projecte. A més d'aquesta manera podrem crear en la següent fase una llista dels

recursos del mateix.

Pretenem crear una aplicació funcional que mitjançant una ajuda visual ens permeti obtindre mes informació sobre el nostre entorn. La informació que haurem de mostrar serà:

- Aula associada.
- Assignatura que s'està donant en aquell moment a l'aula associada i horari de aquesta.
- Professor associat a aquella assignatura.
- Assignatura que es donarà un cop finalitzi la Assignatura actual.
- Professor assignat a l'assignatura que es realitzarà.
- Horari de l'assignatura que es realitzarà.

Pretenem posar a disposició del usuari una llista de Professors, assignatures i aules, cadascun disposarà d'una fitxa que mostrarà l'informació associada a aquests.

Per a la fitxa dels professors, volem que serveixi als alumnes com a mitjà de contacte amb els professors, per tant haurem de crear accessos directes als telèfons, correus electrònics i webs del mateixos, així com també:

- Nom.
- Despatx.
- Assignatures que imparteix.

Per a la fitxa de l'assignatura, seguirem la base del campus digital, on podem trobar:

- Nom de l'assignatura.
- Professors que l'imparteixen.
- Coordinador.
- Numero de crèdits.
- Competències.
- Nom de la matèria associada.

4.2.2 Requeriments

En aquesta fase, dimensionarem els requisits del projecte, escollint les eines necessàries per al mateix. També haurem de dimensionar i establir les taules i les relacions entre taules de la nostre base de dades, que per agilitzar la lectura d'aquest projecte estan ubicades als annexos del mateix.

4.2.2.1 SQLite

Per a la creació de la base de dades usarem SQLite, una versió de SQL lleugera per a

dispositius mòbils que ens permet tindre una base de dades que funciona de manera optima en la nostra aplicació.

SQLite esta basat en ACID (en anglès atomicitat-consistència-isolació-durabilitat) i es un sistema de gestió de bases de dades incrustada relacional que ens permet tindre una llibreria que ocupa poc (275kB aprox.).

Es tractarà d'una base de dades local (allotjada al propi dispositiu), i a la que accedirem a l'aplicació des de un Content Browser, des de on la obrirem, la actualitzarem, la tancarem, i realitzarem les consultes necessàries.

Donat que en un principi no es tracta d'una base de dades gaire extensa no hi ha cap problema donat que són 14 Kb, que tot i que pot créixer força, és tracta de valors completament assumibles per als dispositius actuals.

4.2.2.2 ZXING

Per realitzar l'escanejat de codis QR usarem una aplicació exterior, donat que per a fer-ho nosaltres hauríem de aplicar tècniques de anàlisis d'imatges i descodificació, i l'extensió del projecte es multiplicaria en excés.

La funció de descodificació de codis la deixarem per al programa Qrdroid (que integra la llibreria Zxing), que ens permet capturar una imatge del codi QR amb la càmera del dispositiu i descodificar-la per retornar al nostre programa el valor descodificat.

L'aplicació s'encarregarà d'accedir a la càmera i descodificar el codi escanejat, l'aplicació a més podria descodificar un codi des de una imatge guardada, una Url o introduir un numero de codi de barres, per ultim l'aplicació també ens permet compartir un codi.

Per integrar-lo a la nostre aplicació haurem de crear un Intent que cridi a la aplicació i que com a resposta de aquesta aplicació (un cop l'usuari escaneja el codi) llencí un altre Intent a la classe que correspon aquell codi QR.

Per tant hem de crear codis QR per a cada classe, els codis seran creats per a una aplicació codificadora de codis per exemple: http://qrcode.kaywa.com/ on podem introduir un text per a ser codificat en una imatge d'un codi QR que pot ser fàcilment guardada i compatible amb qualsevol visualitzador d'imatges. D'aquesta manera l'únic que em de fer és introduir el numero de classe al codificador, indexar-lo a la base de dades i ja el podem usar a l'aplicació.

4.2.3 Disseny

És recomanable en tota aplicació realitzar un disseny de la aplicació, podem usar un programa com Photoshop per el cas, pot semblar que realitzar un disseny previ pot ser una pèrdua de temps, però crear un disseny en Photoshop ens servira per:

a) Fer-nos una idea i conèixer la futura disposició del layout, disposant així d'una eina que ens permetrà conèixer la distancia exacte a la que estan els diferents elements i les dimensions d'aquests.

b) Disposar de les imatges dels diferents elements del layout per poder integrar més tard al programa aquestes imatges, i aconseguir d'aquesta manera un resultat més personalitzat i professional.

c) Si em de realitzar canvis a l'aplicació, tals com afegir alguna funcionalitat o canviar la disposició de algun element podem veure com afecta als altres elements sense necessitat de realitzar cap canvi al codi.

És recomanable fer us de les capes (layers) dins Photoshop, degut a que ens permet amagar totes les altres i treballar o guardar una determinada capa

4.2.3.1 Interfície

El primer que hauríem de plantejar-nos, un cop tenim una idea del que volem fer, és decidir quines opcions volem que el nostre programa tingui, que volem que faci.

Primer crearem un disseny del layout (distribució) de la Activitat principal, la que volem que sigui la primera pantalla que vegi l'usuari. En el nostre cas volem que el usuari poguí escollir entre fer una recerca de professors, aules o assignatures, i que a més tinguí l'opció de poder escanejar un codi QR. Es per tant raonable crear un boto a l'activitat per a cadascuna d'aquestes accions, i a pesar de que escanejar seria una acció diferent per qüestions de unificació crearem un boto també per a escanejar.

Fig. 4.1 Disseny aplicat a la primera activitat

No cal que programem exactament aquest disseny (**Fig 4.1**), es més en ocasions no ens serà possible, però ens ajuda a fer-nos una idea de l'espai.

Com es pot veure em afegit a més de text ajudes visual, per fer l'aplicació més entenedora i més immediata.

Per a la següent activitat hauríem de tindre en compte que un cop toquem un dels botons realitzarem dos tipus d'accions, si fem clic als tres primers botons (la cerca de professors, assignatures o classes) hauríem d'anar a parar a una pantalla de selecció on escollirem quin ítem volem veure en detall, per aquest tipus d'activitat hem escollit usar simplement una llista amb el fons usat a l'activitat anterior, per mantenir l'aplicació uniforme. L'altre tipus d'acció seria l'escanejat de codi, que en el nostre cas ens portarà a una altre aplicació per realitzar l'escanejat i posterior descodificació, i després ens portarà a la pantalla on veurem la informació associada a aquell codi QR.

Per ultim haurem de realitzar el disseny les vistes de detall dels items, aquestes són:

Per al layout de classe (**Fig 4.2**) amb l'acció d'escanejar i de buscar per llista), volem que s'ens mostri el nom del professor que esta realitzant classe a aquella aula, l'assignatura

que esta donant, de quina a quina hora es farà aquella classe, el professor que haurà de fer classe després de la classe actual, l'assignatura impartida i de quina a quina hora es farà aquesta classe. A més a mode ajuda visual hem afegit el numero de classe i el fons canvia depenguen de a quina torre esta situada la classe.

Fig. 4.2 Layout de classes

Per al Layout de Assignatura (**Fig 4.3**), volem tindre obtindre la informació relativa a cada assignatura, hem seguit bàsicament l'estructura del campus virtual, o sigui: Nom de l'assignatura, el coordinador, els professors associats a aquella assignatura, el numero de crèdits de l'assignatura, les competències i el nom de la matèria.

Fig. 4.3 Layout de assignatures

Per ultim per al layout de professor (**Fig. 4.4**), volíem donar una mica mes d'interactivitat a l'activitat i es per això que a part de el nom del professor, el despatx i l'assignatura que imparteix em creat accessos directes per a poder, trucar al professor, visitar la web del professor (en cas que tingui web o/i telefon) i enviar un correu.

A més s'han creat uns botons amb icones per a fer els accessos mes entenedors per als usuaris.

Fig. 4.4 Layout de professors

Donat que els botons de Android tenen 3 estats, normal (**Fig 4.5**), onFocus (**Fig 4.6**), quant el cursor esta senyalant el boto amb el cursor, o quant es clicat (**Fig 4.7**), o tocat, donat que parlem de pantalles tàctils, hauríem de crear una versió del boto per a cada estat.

En el nostre cas, hem aplicat estils de capa, una opció de Photoshop que ens permet crear ombres, degradats i d'altres de forma paramètrica, que podem copiar i enganxar a d'altres capes del document.

Per exemple per al boto de l'exemple de mes abaix hem creat un requadre on em afegit l'icona i el caràcter “>”, a aquests dos em aplicat l'estil de capa que ens permet crear un brillo a la capa actual.

Hem canviat alguns de aquests valors, per exemple, per al boto onFocus, em canviat el traç del boto a un color blau mes clar, i per al boto clicat, em canviat la direcció del degradat de fons. El text s'afegeix després, per codi, donat que d'aquesta manera tenim la possibilitat de canviar-lo sense necessitat de modificar el disseny.

A continuació es pot veure la diferencia entre els tres tipus de botons:

Fig. 4.5: Boto en estat "normal"

Fig 4.6: Boto en estat "OnFocus"

Fig 4.7: Boto en estat "tocat"

No afegim cap text donat que l'afegirem per codi, això ens permet canviar el text, si ho requerim, del boto per codi, sense tindre que editar la imatge i d'una forma molt mes àgil.

A més, hauríem de crear els icones del menú inferior, per als quals hem seguit les directrius de Google (**Fig 4.8**).

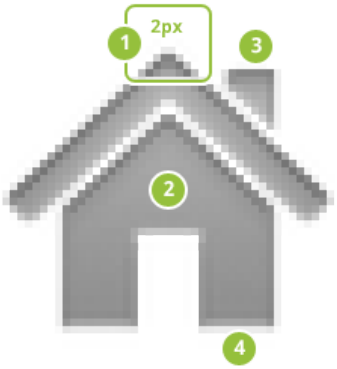


Fig. 4.8 Exemple extret de la pagina de Android per a creadors

- Arrodoniment de cantonades, 2 px de radi de cantonada, quant convingui.
- Omplir amb un gradient, a 90º des de el color `#8C8C8C` al color `#B2B2B2`.

3. `Ombrejat interior amb el color #000000, 20% d'opacitat, angle 90°, distancia 2px, tampany 2px.`

4. `Bisell interior: profunditat 1%, direcció cap abaix, mida 0px, angle 90°, altitud 10°, color del remarcat: #ffffff a 70% d'opacitat, color de l'ombra: #000000 a 25% d'opacitat.`

Per ultim l'únic que ens queda dissenyar es boto de l'icona de l'aplicació, per a la qual em seguit també les directrius de Google. Segons Google, aquesta icona hauria de tindre 3 objectius principals:

- Promoure la marca i explicar l'historia de l'aplicació.
- Ajudar als usuaris a distingir l'aplicació dins el Market.
- Funcionar bé al escriptori.

És per tot això que em decidit usar el logo de la UPC (**Fig 4.9**), donat que ens dona una idea per a qui esta destinada l'aplicació, que es lo que ens ofereix, es fàcilment distingible donat que tot alumne ha vist el logo, i funciona perfectament amb el fons negre i la resta d'icones del escriptori.

Fig. 4.9 Icona escollida

A més hem de deixar certa distancia amb el limit de la imatge per a mantenir una imatge consistent amb la resta d'icones del mòbil.

4.2.4 Codificació

Com ja s'ha explicat, l'objectiu d'aquesta aplicació és conèixer la matèria que s'esta donant a una classe en un determinat moment, per a realitzar aquesta tasca em cregut adient fer us dels codis QR, i d'aquesta manera fer saber al dispositiu de quina classe volem conèixer la informació. A més volem introduir també l'opció de poder buscar una determinada classe, assignatura o professor per mitja de llistes de selecció.

Existeixen diferents elements que em considerat especialment necessaris per a la realització d'aquest projecte dins Android:

- Fer us de Content Providers per accedir a la base de dades i d'aquesta manera omplir les llistes i les fitxes de classes, assignatures i professors.
- L'us de consultes SQL per a trobar els valors que requereix l'aplicació dins la base de dades.
- L'us d'Intents, per accedir a d'altres activitats creades per nosaltres o, en el cas de Zxing, creades per tercers.
- L'us de menús inferiors, per a un accés ràpid a diferents parts de l'aplicació amb només un click.
- La utilització de layouts en format XML per a la distribució d'elements d'una activitat.
- L'us de Layouts personalitzat per a mostrar correctament l'activitat d'inici.
- Llistes (List Views), per presentar tots els valors que pot seleccionar l'usuari, d'una forma entenedora.

4.2.4.1 Organització de l'aplicació

Dins l'aplicació, el primer que ens trobem és el menú principal, on podem accedir a 4 botons: Buscar classe, Buscar assignatura, Buscar Professor i escanejar codi QR.

Si fem clic a qualsevol dels 3 primers botons accedirem a una llista, que conté una llista de totes les classes, assignatures o professors, aquests valors s'obtenen omplint un vector amb tots els valors necessaris de la base de dades, per tant des de l'activitat actual (l'activitat que conté la llista) hem d'accedir al Content Provider que s'encarrega de realitzar totes les consultes a la base de dades de dades i retornar-les en forma de vector, que ens serveix per a omplir la llista corresponent, llavors el usuari escull el valor desitjat, es fa una nova serie de consultes i es pinta la activitat corresponent.

Per a l'escaneig dels codis QR, realitzem una crida a una aplicació exterior (Qrdroid), que s'encarrega de fotografiar el codi, descodificar-lo i retornar-lo a la nostra aplicació, aquest codi en aquest cas es tracta d'un numero de classe, però podríem codificar el nom d'un professor, una assignatura. Un cop tenim el numero de classe realitzem una consulta a la nostre base de dades, i aquesta ens retorna els valors que em de pintar, que en aquest cas és el mateix que ens apareixeria després de clicar dins la llista de classes.

4.2.5 Testeig

L'objectiu d'aquesta pràctica és trobar possibles errors o problemes en el programa i corregir-los de forma que obtinguem una aplicació més robusta i resistent a fallades.

En un projecte professional el nombre de probes i el temps que podem dedicar a aquestes esta lligat al nombre de recursos, però és tracta d'una fase que les companyies cada cop tenen més en compte, donada la seva importància. Donat que el projecte només l'ha fet una persona ha estat impossible realitzar una bateria de test extensa, però suficient per a obtindre una aplicació robusta.

De totes maneres, és tracta d'una aplicació que si ve disposa de diferents opcions, les maneres en que l'usuari interacciona amb elles son limitades, i per tant no gaire susceptible d'errors.

Capítol 5: Conclusions, millores i bibliografia

5.1 Conclusions:

L'objectiu principal era aconseguir un sistema que ens permetés orientar-nos dins un edifici, donat la impossibilitat d'usar GPS, creiem que l'us de marcadors QR podria esser una opció de baix cost i que no requereix de hardware especial, a part de l'us de càmera de fotografies, a diferencia de altres tecnologies com ara NFC. Això permet que tot tipus de dispositius de baix cost amb Android puguin fer us d'aquesta tecnologia.

També volíem donar a aquest projecte un enfoc una mica diferent aplicant el disseny d'interfície, que ens permet obtindre una aplicació més atractiva, i permetre'ns realitzar una planificació més acurada de tots els aspectes d'aquesta, així com donar-nos una visió d'una part del procés que no es dona a la nostra carrera però estretament relacionada amb el que ens ocupa.

A més aquest projecte ens ha permès aprendre una tecnologia, molt estesa avui en dia (i d'un gran creixement), com es Android, un sistema operatiu basat en Java i Linux, tecnologies molt importants dins el mon de les telecomunicacions.

Android, un camp molt interessant, que val la pena aprofundir un cop finalitzat el projecte, i una interessant opció de futur donat que el mon de les aplicacions mòbils es dels pocs on la crisis no sembla tindre molt efecte, tal i com podem veure en fires com ara la Movile World Congress, que es celebra a Barcelona.

5.2 Millores:

Tot i haver complert àmpliament els objectius inicials, creant no només una aplicació que ens permet descodificar un codi QR i mostrar l'informació associada, sinó a més creant una versió reduïda del nostre campus digital, creiem que podríem dur a terme una serie de millores que podrien convertir aquesta aplicació en una aplicació molt útil per a professors i alumnes del campus de Castelldefels. Aquestes millores, podrien ser:

- Integrar el sistema de qualificacions, de manera que qualsevol professor pogués, per exemple corregir exàmens i/o treballs i penjar-los des de on fos, només amb l'ajuda del seu Smartphone. O permetre a qualsevol estudiant consultar les seves notes actualitzades en temps real integrant per exemple un sistema de notificacions a la barra superior del sistema operatiu.

- Podríem accedir de forma remota a la base de dades de la universitat, per d'aquesta manera obtindre informació actualitzada dels camps requerits. Suposant que la base de dades de la universitat esta en format SQL, no hauríem de realitzar grans canvis, a part de integrar un Webservice a la nostre aplicació per accedir-hi. Potser alentiria el rendiment de la nostra aplicació, però no radicalment

- Podríem crear un mapa del campus, per situar-nos quan siguem a l'exterior, or per consultar la situació d'una classe o despatx

- Per fer l'aplicació més social podríem implementar un xat entre estudiants o professors, per d'aquesta manera poder treballar de manera remota en treballs en grup.

- En una possible versió 2.0 es podria integrar una utilitat tipus layar, on a l'exterior dels edificis ens mostres la ubicació de diferents localitzacions, com ara la biblioteca, el bar o determinat despatx o classe, aquesta opció hauria de fer us del GPS, i per tant seria només útil en exteriors.

5.3 Bibliografia:

Enllaços útils per a conèixer l'historia de Android:

http://mashable.com/2011/07/26/android-history-infographic/
http://en.wikipedia.org/wiki/Android_%28operating_system%29

Pàgina de la Wikipedia que ens explica l'historia d'Andy Rubin un dels pares de Android:

http://es.wikipedia.org/wiki/Andy_Rubin

Pàgina de la Wikipedia que ens parla sobre l'historia de l'Ipad un dels mes directes competidors de les taules tàctils d'Android:

http://es.wikipedia.org/wiki/IPad

Enllaços que enumera les millores contingudes en la versió 2.0 d'Android:

http://developer.android.com/sdk/android-2.0.html#api

Distribució actualitzada de les diferents versions d'Android basada en els accessos a Google Play des de els diferents terminals:

http://developer.android.com/resources/dashboard/platform-versions.html

Web de descarrega del Android SDK:
http://developer.android.com/sdk/index.html

Fonaments sobre les aplicacions dins Android:

http://developer.android.com/guide/topics/fundamentals.html

Fonaments sobre un tipus de components de aplicacions dins Android anomenats activitats, on s'inclou el cicle de vida de la mateixa:

http://developer.android.com/guide/topics/fundamentals/activities.html

Web que ens permet crear codis QR, útil per a la realització dels codis QR de les diferents aplicacions:

http://qrcode.kaywa.com/

Informació útil sobre Sqlite, un estàndard basat en SQL especialment dissenyat per a dispositius mòbils, i que a més és el que hem usat:

http://en.wikipedia.org/wiki/Sqlite

Aplicació per a Android que ens permet escanejar i descodificar codis QR:

http://code.google.com/zxing

Visor simple de bases de dades que ens permetrà crear la base de dades per al projecte per mes tard importar-la des de eclipse:

http://sqlitebrowser.sourceforge.net

Pàgina creada per Google que ens proporciona tota la informació necessària per introduir-nos dins el mon d'Android i començar a programar:

http://developer.android.com/index.html

Pàgina que ha estat de gran ajuda per a aquest projecte, es tracta de una comunitat de programadors per a tot tipus de llenguatges:

http://stackoverflow.com/questions/tagged/android

Pàgina oficial de Eclipse, un dels entorns més estesos de programació existent:

http://www.eclipse.org

Informació sobre codis QR:

http://en.wikipedia.org/wiki/QR_code
http://searchengineland.com/what-is-a-qr-code-and-why-do-you-need-one-27588

Informació sobre codis tradicionals de barres:

http://es.wikipedia.org/wiki/Código_de_barras

Informació sobre les bones pràctiques en Android:

http://developer.android.com/guide/practices/index.html

Enllaç que ens permet conèixer més sobre l'API d'Android:

http://developer.android.com/guide/components/index.html

Web per a conèixer la millor manera per a instal·lar i començar a programar Android:

http://developer.android.com/tools/index.html

Capítol 6: Annexos

6.1 Taules

A continuació fem esment de les diferents taules creades per a la base de dades i la seva relació:

6.1.1 Taula: Assignatures

Usarem aquesta taula bàsicament per a pintar l'activitat Llista_assignatura i controlador_assignatura, conte tota la informació relacionada amb les Assignatures impartides.

Camp	**Tipus**	**Comentari**
NAssignatura	Numeric	Index per a identificar el numero d'assignatura
Assignatura	Text	Nom de l'assignatura
Profe1	Integer	Professor assignat a la assignatura
Profe2	Integer	Professor assignat a la assignatura
Profe3	Integer	Professor assignat a la assignatura
Coordinador	Integer	Coordinador assignat a la assignatura
NCredits	Integer	Nombre de crèdits de l'assignatura
Competencies	Text	Competències de l'Assignatura
NomMateria	Text	Nom de la matèria que te assignada la assignatura

6.1.2 Taula: Classes

Aquesta taula ens serveix per a trobar quin color te associat determinada classe, es per tant una taula bastant simple.

Camp	**Tipus**	**Comentari**
NumClasse	Numeric	Numero de clase

Color	Integer	Color a que correspon determinada classe

6.1.3 Taula: Horaris

Relació dels horaris associats a cada classe, on es mostren les hores d'inici i finalització de cada assignatura i els dies en que s'imparteixen.

Camp	**Tipus**	**Comentari**
NumClasse	Numeric	Index per indicar el numero de classe, obtindrem el color de la classe consultant la taula Classes
NAssignatura	Integer	Numero de assignatura, obtindrem el seu valor en text consultant la taula Assignatures
HInici	Integer	Hora d'inici d'una assignatura
MInici	Integer	Minut d'inici d'una assignatura
HFi	Integer	Hora de finalització d'una assignatura
MFi	Integer	Minut de finalització d'una assignatura
Dia	Text	Dia en que s'imparteix l'assignatura

6.1.4 Taula: Profes

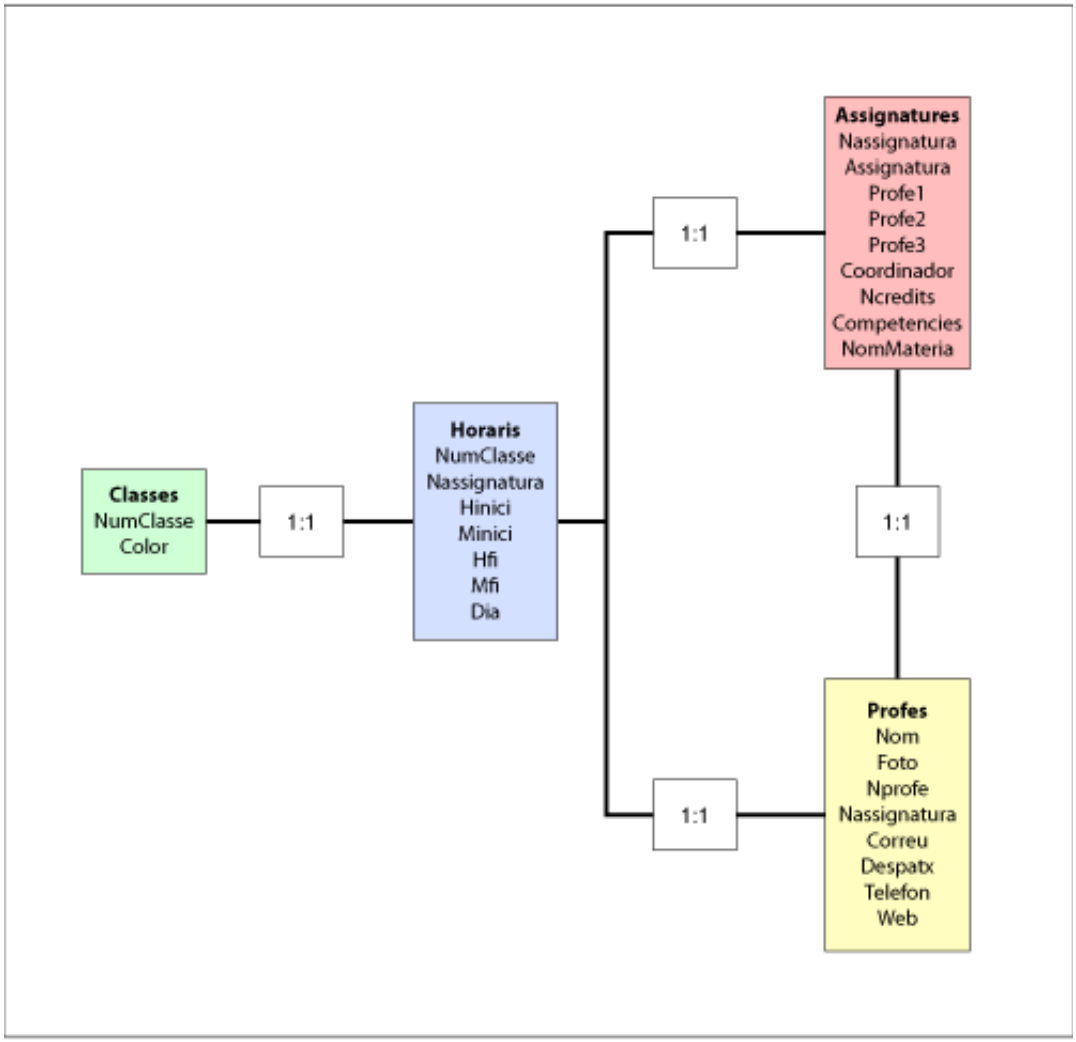
En aquesta taula tenim els camps relacionats amb les activitats de Llista_profes i controlador_profes, trobarem la informació de cada professor, que ens permetrà a més mostrar el nom a d'altres activitats on tenim només el index referent al professor, com ara controlador_assig, o controlador_classe.

Camp	**Tipus**	**Comentari**
Nom	Text	Nom del professor en qüestió
Despatx	Text	Despatx del professor
Web	Text	Web del professor (si la te)
Telefon	Integer	Telefon del professor (si el te)
Correu	Text	Correu electrònic del professor (si el te)

NAssignatura	Integer	Numero d'assignatura assignada a aquest professor
Foto	Text	Nom de Fotografia, funcionalitat que és podria afegir en un futur
NProfe	Text	Numero de professor, per a buscar a altres taules

Per ultim és necessària una taula extra, que requereix l'especificació de SQLite, aquesta taula tindrà un únic valor “locale=es_ES”, ens serveix per a especificar la localització del telefon mòbil, així com l'idioma del mateix.

6.2 Relacions:



6.3 Fulla del resultat de les proves de qualitat:

Per a la realització d'aquest projecte varem realitzar una serie de proves de qualitat amb els següents errors o possibles millores i els seus resultats:

	Tipus:	Gravetat:	Estat:	Comentari
Al girar el telefon (mode apaïsat) queda amagat el boto de buscar per classe dins el menú principal.	Error	Mitjà	Ok	S'ha creat un nou layout per al menú principal en mode apaïsat.
Quant s'envia un correu a un professor que automàticament aparegui el Email al camp destinatari de l'aplicació de correu.	Error	Mitjà	Nok	Millora per a una possible versió 1.1.
La icona del programa és mostra tallada.	Error	Baixa	Ok	S'ha de deixar cert marge per a que el sistema no talli la imatge.
Dins LogCat a Eclipse apareix NullPointException, i l'aplicació dona error.	Error	Alta	Ok	El camp Locale=”Us_us” era erroni, ha de ser Locale=”ES_es”.
S'ha de donar algun tipus d'input visual per a que l'usuari sàpiga quant un boto es polsat o no.	Millora	Mitjà	Ok	S'han creat imatges personalitzades per a cada boto que canvien en funció de l'estat del boto (en repòs, on focus, boto tocat)
Crear icona per retornar ràpidament al menú principal des de qualsevol part de l'aplicació.	Millora	Mitjà	Ok	S'ha creat un nou boto actiu a tota l'aplicació menys al menú principal.
Crear enllaços a cada controlador per anar als altres controladors.	Millora	Baixa	Ok	S'ha creat un menú inferior per a poder accedir ràpidament a d'altres activitats.
Integrar escanejar codi a l'aplicació per no tindre que instal·lar cap aplicació.	Millora	Mitjà	Nok	Millora per a una possible versió 1.1.
Afegir ancores per a una navegació rapida dins les	Millora	Baixa	Nok	Millora per a una possible versió 1.1.

llistes				

6.6 Instal·lació d'Android:

Apunt*: per a la instal·lació em considerat un entorn Windows 7, donat que es sistema instal·lat en el ordinador en el que em creat el programa, existeix tot tipus de documentació per a realitzar-ho per a entorns MacOS o Linux, però considerem que no és tasca d'aquest projecte ser un manual d'instal·lació per als diferents sistemes operatius.*

6.6.1 Instal·lació de JDK:

Per a poder començar a programar Android, sistema operatiu basat en Java, necessitem descarregar i instal·lar el software que conté les eines de desenvolupament per a la creació en Java (o JDK) així com també una maquina virtual de Java (o JRE), per ultim descarregarem Android SDK que ens proporciona l'emulador així com una serie d'eines per a treballar en aquest. L'únic que ens manca és un entorn on programar, l'escollit en aquest cas és Eclipse, donat que es un dels més estesos en la comunitat, és gratuït, de codi lliure i per el que hi ha més documentació oficial.

La instal·lació del JDK i el JRE és com la instal·lació de qualsevol altre programa, fer doble click al arxiu descarregat i fer click a next de les diferents pantalles, un cop instal·lats, passarem a instal·lar el Android SDK, tasca potser un pel més complicada.

Descarreguem i executem el Android SDK de la web d'Android recomanat per al nostre sistema operatiu (http://developer.android.com/sdk/index.html), la versió exe ja ens farà el cas.

Esperem a que acabi d'instal·lar-se i un cop finalitzat, hem de fer clic esquerra sobre “Mi PC”, seleccionar propietats, i clicar a “Configuración avanzada del sistema” (**Fig. 2.6**).

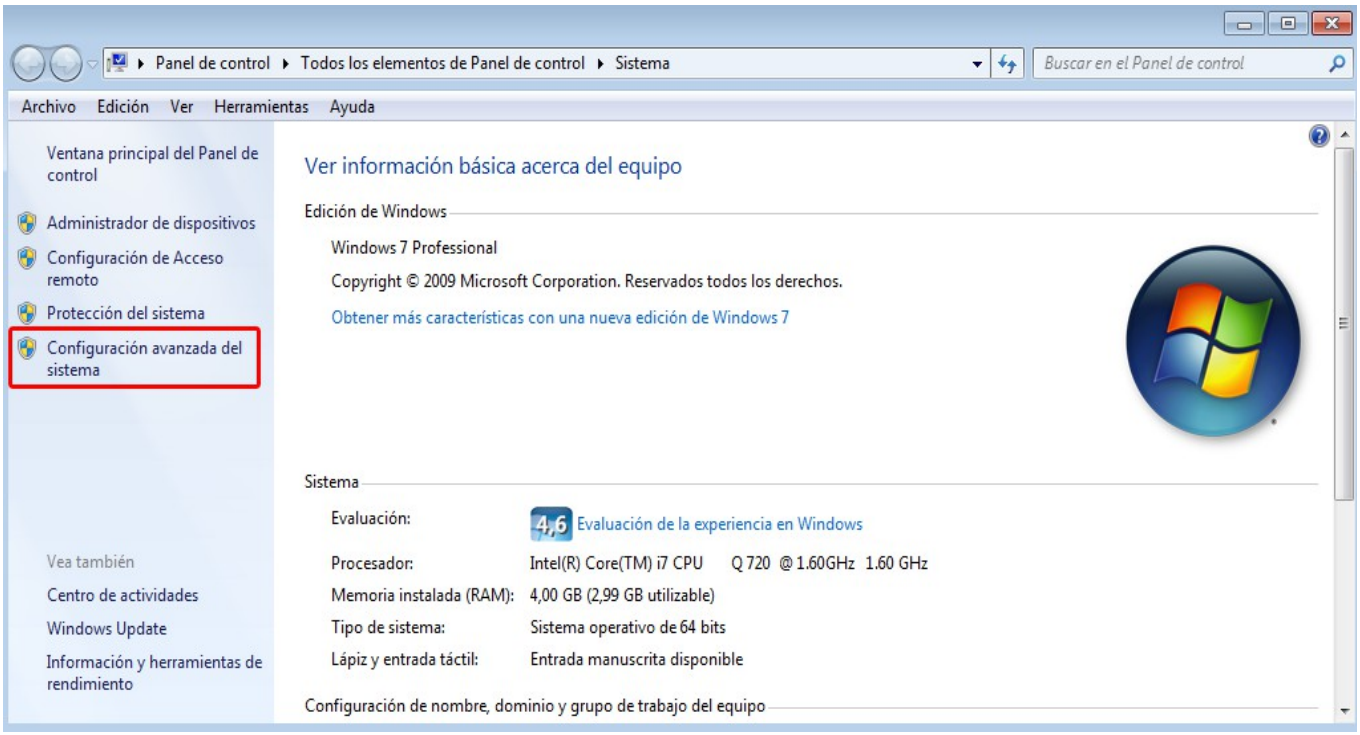


Fig. 6.6 Configuració avançada del sistema

A continuació fem clic a “Variables de entorno...” (**Fig. 6.7**).

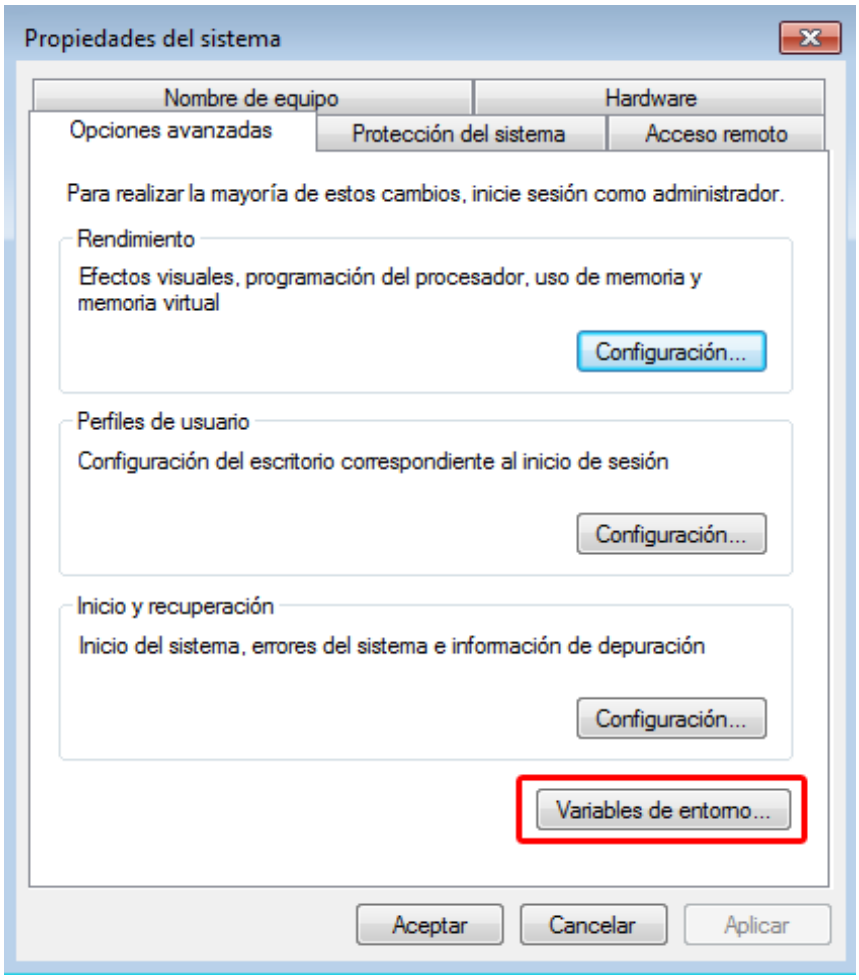


Fig 6.7 Propietats del sistema

A “Variables de usuario” fem doble clic a la variable PATH, i em de afegir la ruta on em instal·lat l'Android SDK (**Fig. 6.8** i **Fig 6.9**).

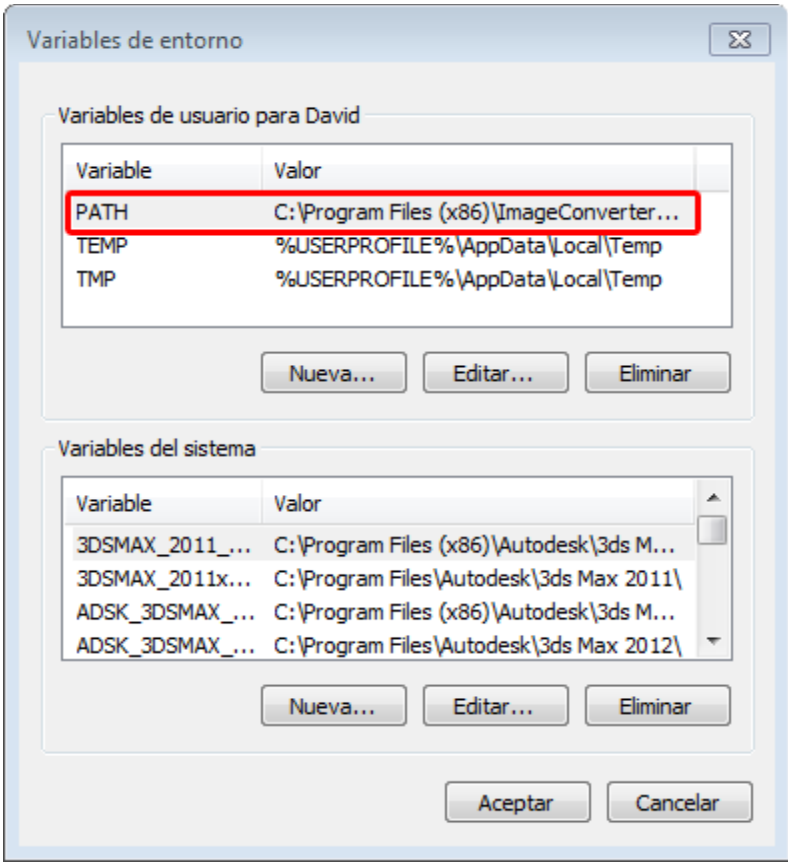


Fig. 6.8 Variables d'entorn

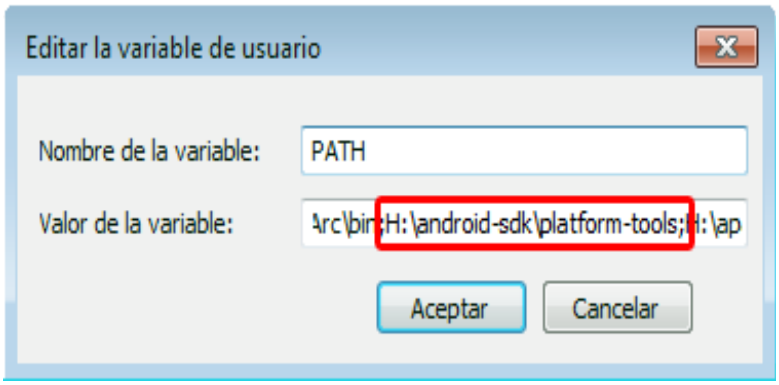


Fig. 6.9 Variables PATH

Això permetrà al sistema conèixer la ubicació del SDK, per a futures actualitzacions del sistema.

A continuació instal·larem Eclipse, l'entorn de programació. Com a primer pas haurem de descarregar l'ultima versió del “Eclipse IDE for Java EE Developers”, la versió de 64 bits en cas que el nostre processador sigui compatible, la gran majoria de processadors actuals ho són, així com el equip usat per al projecte, en cas contrari haurem d'escollir la de 32 bits, des de la web http://www.eclipse.org/downloads/.

Un cop descarregat l'arxiu, l'executarem i l'instal·larem. Escollim la ruta de destí del programa i esperem a que s'instal·li.

Iniciem l'Eclipse, i un cop escollit la ruta del nostre Workspace, haurem d'instal·lar el plugin per a Eclipse d'Android. Dins Eclipse anem a “Help/Install new software” i fem clic a Add.

Al quadre de diàleg, a name posem "ADT Plugin" i a location posem la següent adreça:

https://dl-ssl.google.com/android/eclipse/

En cas que tinguem problemes amb aquesta adreça web, substituirem l'https per http, ja que en determinats navegadors pot ocasionar problemes, es recomanable per temes de seguretat usar el protocol https, però no es necessari.

Fem clic a ok, i a al següent quadre, seleccionem el checkbox de “developer tools” i next.

Acceptem l'acord de llicencia i fem clic a “finish”. Un cop fet tot això haurem de reiniciar l'Eclipse.

Ara configurarem el plugin ADT:

1. Anem a “Window/preferences...” per obrir les opcions d'Eclipse.
2. Seleccionem Android al panell de l'esquerra.
3. A “SDK location” farem clic a Browse i buscarem on hem instal·lat l'SDK d'Android.
4. Fem clic a “Apply” i finalment Ok.

Per ultim, actualitzarem l'SDK.

1. Fem clic a “Window/Android SDK Manager”
2. Fem clic a “Select New/updates”
3. Per ultim fem clic a “Install packages”.

Amb aquesta opció ens instal·larà totes les noves actualitzacions de Android, procés que porta una bona estona. Un cop finalitzat ja estarem llestos per a començar a programar.

6.4 Tractament de la base de dades:

Les activitats que requereixin informació de la base de dades (com les llistes, o les fitxes de assignatures, professors o classes), accediran a Content Browser, a la funció corresponent i aquesta farà la consulta a l'arxiu amb la informació (la base de dades) i la retornarà en forma de array o variable.

Per a poder accedir a les diferents funcions encarregades de la cerca d'items dins la base de dades (de professors, classes o assignatures) haurem de seguir gairebé sempre el mateix patró, per exemplificar-ho veure'm l'exemple de com hem creat una llista (en aquest cas la dels professors):

```
public class Llista_profes extends ListActivity {
	BaseDatosHelper miBBDDHelper;

	private class ProfeAdapter extends ArrayAdapter<Info> {
		ArrayList<Info> items;

		public ProfeAdapter(Context context, int textViewResourceId,
				ArrayList<Info> items) {
			super(context, textViewResourceId, items);
			this.items = items;
		}

		@Override
		public View getView(int position, View convertView, ViewGroup
parent) {
			View v = convertView;
			if (v == null) {
				LayoutInflater vi = (LayoutInflater)
getSystemService(Context.LAYOUT_INFLATER_SERVICE);
				v = vi.inflate(R.layout.llista_item_profes, null);
			}

			Info classe = items.get(position);
			if (classe != null) {

				TextView tNomProfe = (TextView)
v.findViewById(R.id.item);
				tNomProfe.setText("" + classe.GetNomProfe());

			}
			return v;
		}
	}

	public void crearBBDD() {
		miBBDDHelper = new BaseDatosHelper(this);
		try {
			miBBDDHelper.crearDataBase();
		} catch (IOException ioe) {
			throw new Error("Unable to create database");
		}
	}
```

```
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.llista_profes);

		crearBBDD();
		ArrayList<Info> Profes = getItems();
		setListAdapter(new ProfeAdapter(this, R.layout.llista_item_profes,
				Profes));

		ListView lv = getListView();
		lv.setClickable(true);

		lv.setOnItemClickListener(new OnItemClickListener() {

			public void onItemClick(AdapterView<?> list, View view,
					int position, long id) {
				TextView tv = (TextView) view.findViewById(R.id.item);

				BaseDatosHelper.tProfe.SetNomProfe("" + tv.getText());
				carregarActivitat(Controlador_profe.class);

			}
		});

	}

	public ArrayList<Info> getItems() {
		miBBDDHelper.abrirBaseDatos();
		ArrayList<Info> listaClasses = miBBDDHelper.GetLlistaProfes();
		miBBDDHelper.close();
		return listaClasses;
	}
}
```

Les funcions mes importants són:

```
public void crearBBDD()

private class ProfeAdapter extends ArrayAdapter<Info>

public ArrayList<Info> getItems()
```

Obrim la base de dades, emmagatzemem el resultat de la funció miBBDDHelper.GetLlistaProfes() dins un Array temporal, tanquem la base de dades i retornem l'array amb les dades, aquest el pasem a la funció ProfeAdapter, i aquesta pinta la llista amb l'array de resultats per a poder mostrar-ho per pantalla.

La funció CrearBBDD ens serveix per a crear l'objecte.

Dins el Content Browser, haurem de crear el codi que ens permetí accedir a la base de dades, realitzar la consultar corresponent i retornar un array (o variable en cas que sigui un sol valor) amb els resultats corresponents:

```
public BaseDatosHelper(Context contexto) {
            super(contexto, DB_NAME, null, 1);
            this.myContext = contexto;
      }

public ArrayList<Info> GetLlistaProfes() {
            ArrayList<Info> llistaProfes = new ArrayList<Info>();

            Cursor c = myDataBase.rawQuery("SELECT " + TABLE_KEY_PROFE + ","
                        + TABLE_KEY_FOTO + " FROM " + TABLE_PROFES + " ORDER BY
"
                        + TABLE_KEY_PROFE, null);
            // Iteramos a traves de los registros del cursor
            c.moveToFirst();
            while (c.isAfterLast() == false) {
                  Info classe = new Info();

                  classe.SetNomProfe(c.getString(0));
                  classe.SetFoto(c.getString(1));

                  llistaProfes.add(classe);
                  c.moveToNext();
            }
            c.close();
            return llistaProfes;
      }

public void crearDataBase() throws IOException {

            boolean dbExist = comprobarBaseDatos();

            if (dbExist) {
                  // Si ya existe no hacemos nada
            } else {
                  this.getReadableDatabase();
                  try {
                        copiarBaseDatos();
                  } catch (IOException e) {
                        throw new Error("Error al copiar la Base de Datos");
                  }
            }
      }

private boolean comprobarBaseDatos() {
            SQLiteDatabase checkDB = null;
            try {
                  String myPath = DB_PATH + DB_NAME;
                  checkDB = SQLiteDatabase.openDatabase(myPath, null,
                              SQLiteDatabase.OPEN_READWRITE);

            } catch (SQLiteException e) {
                  throw new Error("Imposible abrir base de datos");// No existe
            }
```

```
        if (checkDB != null) {
              checkDB.close();
        }

        return checkDB != null ? true : false;
  }

  public void abrirBaseDatos() throws SQLException {
        String myPath = DB_PATH + DB_NAME;
        myDataBase = SQLiteDatabase.openDatabase(myPath, null,
                    SQLiteDatabase.OPEN_READWRITE);

  }
```

Les funcions importants dins el Content Browser són:

```
public void crearDataBase() throws IOException

private boolean comprobarBaseDatos()

public void abrirBaseDatos() throws SQLException

public ArrayList<Info> GetLlistaProfes()
```

El primer que fem es obrir el fitxer que conté la base de dades, controlant que efectivament sigui possible obrir-lo, creem un array amb l'estructura de la funció Info, on emmagatzemem el resultat del punter que ens dona la consulta SQL i finalment tornem aquest array per que l'activitat que l'ha sol·licitat pugui fer-ne us (mostrar-ho per pantalla en el nostre cas).