Títol: *Integració del Racó de l'estudiant a l'entorn Android*

Volum: *1*

Alumne: *Roger Sala Angordans*

Director/Ponent: *Hugo Hernández Pibernat / Anna Río Doval*

Departament: *Llenguatges i Sistemes d'Informàtics*

DADES DEL PROJECTE

Títol del Projecte: Integració del Racó de l'estudiant a l'entorn Android

Nom de l'estudiant: Roger Sala Angordans

Titulació:Enginyeria en Informàtica

Crèdits: 37,5

Director/Ponent: Hugo Hernández Pibernat

Departament: Llenguatges i Sistemes Informàtics (LSI)

MEMBRES DEL TRIBUNAL (*nom i signatura*)

Presidenta: Maria José Serna Iglesias

Vocal: Estanislau Llanta Salleras

Secretari (ponent): Anna Rio Doval

Codirector: Jaume Moral Ros

QUALIFICACIÓ

Qualificació numèrica:

Qualificació descriptiva

Data: Dimecres, 21 de desembre de 2011

Contribucions del Projecte

El projecte que es descriu en aquest document contempla el desenvolupament d'una aplicació per a dispositius Android per al Laboratori de Càlcul de la Facultat d'Informàtica de Barcelona. L'objectiu principal del projecte, la creació d'aquesta aplicació anomenada *Racó Mobile*, ha estat assolit amb èxit. De fet, l'aplicació ja es troba al principal centre de distribució d'aplicacions mòbils de Google, l'*Android Market*, i als telèfons d'estudiants i altres membres de la facultat que ja l'han descarregat.

A més a més, la col·laboració amb el LCFIB ha permès col·laborar en el desenvolupament i posada en marxa de l'*API* pública que permet a desenvolupadors externs consultar part de la informació que hi ha disponible al Racó de l'estudiant. Aquesta *API* també està ja disponible per tot aquell que la vulgui utilitzar. Cal dir, que es tracta d'un projecte molt novadors, ja que no és gens habitual, trobar una universitat que permeti accedir als seus serveis en forma de recursos web. De fet, aquesta afirmació no només és certa a l'àmbit nacional o estatal, sinó que es fa extensible a nivell mundial. Fruit de la innovació i caràcter institucional del projecte, ha sorgit la oportunitat de presentar-lo i defensar-lo en dues jornades d'àmbit universitari:

- Hugo Hernández, Jaume Moral, Marcel Arbó i Roger Sala. *Mobilitzant el Racó de l'estudiant de la Facultat Informàtica de Barcelona.* TSIUC[1] - Trobada dels Serveis Informàtics de les Universitats de Catalunya. Organitzat pel Centre de Serveis Científics i Acadèmics de Catalunya (CESCA). 2011.
- Jaume Moral, Hugo Hernández, Marcel Arbó i Roger Sala. *Movilizando la intranet de la Facultad de Informática de Barcelona.* Jornadas Técnicas de RedIRIS[2]. 2011.

Addicionalment, la ponència a les TSIUC va donar lloc a una invitació per publicar un resum del projecte a una revista de divulgació científica:

- Hugo Hernández, Jaume Moral, Marcel Arbó i Roger Sala. *Mobilitzant el Racó de l'estudiant de la Facultat Informàtica de Barcelona*. Teraflop. ISSN: 1134-6671. CESCA. Article acceptat i pendent de publicació.

[1] http://www.cesca.cat/ca/trobada-dels-serveis-informatics-de-les-universitats-de-catalunya/mobilitat-i-ubiquitat
[2] http://www.rediris.es/jt/jt2011/

Agraïments

En primer lloc m'agradaria agrair a l'Hugo Hernández que acceptés dirigir el projecte. El seu coneixement, la seva implicació, voluntat i ganes de realitzar una bon projecte han estat essencials perquè la seva finalització sigui un èxit. Gràcies Hugo.

En segon lloc, donar les gràcies al Laboratori de Càlcul de la Facultat d'Informàtica de Barcelona per donar-me la possibilitat de realitzar un projecte al costat de professionals en l'àmbit. Concretament, destacar a Jaume Moral, una de les millors persones que fins ara he conegut. La seva implicació, dedicació i el seu gran coneixement en l'àmbit informàtic han contribuït a que la realització d'aquest projecte fos pràctica i amb el mínim d'entrebancs possibles. A part de la seva flexibilitat alhora d'escoltar i acceptar noves propostes. Gràcies Jaume.

En tercer lloc, donar les gràcies a l'Erola Rabat. Des de un principi va mostrar un gran interès pel projecte i va demostrar tenir grans coneixements en el disseny d'aplicacions mòbils. El disseny que es presencia en l'aplicació ha estat ideat i portat a terme per ella. Gràcies a Erola.

En quart lloc, agrair a les persones que han estat al meu costat al llarg d'aquests anys. Els meus pares i la Laia. Amb ells he compartit moments d'alegria i tristesa. Els d'alegria els he intentat aprofitar el màxim, per contra, els de tristesa els he pogut superar gràcies als seus consells i sobretot, la seva paciència. Així doncs, m'agradaria dedicar aquest projecte a ells, ja que sense el seu suport ara mateix no seria on sóc. Gràcies pares. Gràcies Laia.

Per últim, una persona molt especial per mi, el meu avi. Ell sempre ha estat al meu costat i encara que en moltes ocasions no sabés molt bé de que li parlava sempre m'escoltava. El seu objectiu era que acabés la carrera el més aviat possible. Per això quan vaig començar el projecte em va regalar una figura que representa un titulat amb el títol sota el braç, com que no tenia el projecte vam treure el diploma. Ara avi, per fi, ja li podem posar. Gràcies avi, sempre junts.

ÍNDEX

ÍNDEX DE FIGURES

Capítol 1

INTRODUCCIÓ

Els telèfons mòbils han estat un dels focus de la indústria de les telecomunicacions al llarg de la última dècada. En pocs anys hem passat d'usar els telèfons únicament per realitzar trucades a disposar de terminals amb la mateixa capacitat de càlcul que un ordinador que permeten accedir a diaris i revistes, connectar-se a xarxes socials, llegir llibres en format electrònic i, desenvolupar aplicacions *ad-hoc* per cobrir necessitats específiques d'usuaris i empreses.

Alguns coneguts exemples d'aplicacions d'aquest tipus són les aplicacions desenvolupades per la majoria de bancs i caixes per a facilitat l'accés al seus clients a l'estat dels comptes oberts a l'entitat. També la major part de diaris i revistes amb presència a Internet disposen d'una aplicació per a poder accedir als continguts de forma més adequada des d'un dispositiu mòbil.

És important remarcar que els dispositius mòbils es divideixen en dues famílies clarament diferenciades. Els primers, són els *Smartphones*[3], van sorgir com a evolució natural dels telèfons mòbils. Disposen de totes les característiques abans esmentades però mantenen una mida propera a la d'un telèfon mòbil convencional. El segon tipus, que ha agafat força renom en els últims dos anys, és la família dels *Tablets*.

La característica distintiva dels *Tablets* és que disposen d'una pantalla molt més gran. De les 3.3 polzades de mitja de que disposa la pantalla d'un Smartphone, passem a les 10 polzades en el cas d'un *Tablet*. Aquest fet permet utilitzar el dispositiu per executar aplicacions amb més quantitat de contingut. L'increment del tamany de la pantalla ha permès augmentar els àmbits d'aplicació dels dispositius mòbils que ara es comencen a utilitzar, per exemple, en hospitals o centres educatius.

Actualment al mercat hi ha quatre grans entorns pel desenvolupament d'aplicacions mòbils: Android (Google) [5], iOS (Apple) [6], BlackBerry OS (RIM)[7] i Windows Phone (Microsoft) [8]. Cadascun disposa del seu propi sistema operatiu i són totalment incompatibles entre si. El projecte que es presenta a continuació està enfocat en el desenvolupament

[3] S'anomenen telèfons intel·ligents

d'aplicacions per a sistemes que incorporin el sistema operatiu Android. Concretament, s'ha col·laborat amb el Laboratori de Càlcul de la Facultat de Informàtica de Barcelona desenvolupant l'aplicació *Racó Mobile*. L'aplicació permet l'accés als serveis i informació proporcionada per la facultat als estudiants en un format adaptat a les propietats dels sistemes Android. L'aplicació ofereix continguts per a usuaris no estudiants, que estiguin interessats en la facultat, hagin d'assistir-hi o posar-s'hi en contacte.

En l'àmbit nacional es poden trobar aplicacions Android per a universitats. Múrcia i Navarra en són dos exemples. La universitat de Múrcia ofereix accés a la informació bàsica i localització, accés a les notícies, localització del mapa d'edificis dels diferents campus mitjançant realitat augmentada i accés a canals com el Twitter de la universitat. L'aplicació de la universitat de Navarra ofereix informació sobre les notícies de la universitat, l'agenda, vídeos, àudios, i un espai per a futurs estudiants.

En els últims anys però, la forma d'accedir a continguts web ha canviat força. L'aparició, i posterior popularització, de dispositius mòbils amb connexió a Internet com Android, iPhone o Blackberry han donat lloc a un gran conjunt d'aplicacions que permeten l'accés a aquests serveis web des de diferents interfícies. Les grans empreses d'Internet com Google, Amazon o Twitter, ja ofereixen *APIs* que permeten a desenvolupadors d'aplicacions externes obtenir dades dels seus servidors per a mostrar-les a les pàgines i aplicacions de tercers. Aquesta oferta ha propiciat un canvi de paradigma en el que no només hi ha proveïdors i consumidors si no que apareix la figura d'un tercer agent que és consumidor i proveïdor al mateix temps. Gràcies a aquestes llibreries, els desenvolupadors poden millorar el comportament d'aplicacions ja existents o integrar en una única aplicació serveis relacionats amb dos proveïdors diferents. Des del punt de vista de les empreses també resulta molt favorable, ja que els desenvolupadors d'aplicacions externes fan més popular l'ús de la seva tecnologia.

Degut a la realització d'aquest projecte final de carrera s'ha aprofitat per poder millorar l'oferta de serveis online de la facultat, i facilitar l'accés a l'informació que existia en la actual arquitectura del Racó. Així doncs, el Laboratori de Càlcul de la FIB (LCFIB) ha decidit crear una capa de serveis web per a usuaris externs. L'objectiu d'aquest canvi és poder oferir una *API* completa que permeti accedir als estudiants a la seva informació en formats estàndard com *XML* o *JSON* per tal de poder usar aquesta informació en altres aplicacions que en millorin la visualització o utilitat. Es tracta d'un objectiu complicat degut a la gran quantitat de serveis que ofereix actualment el Racó i també degut a que aquest integra informació de moltes naturaleses i orígens diferents. Per aquest motiu, s'ha optat per donar un primer pas creant aplicacions per adonar accés als estudiants des de dispositius mòbils Android o iPhone. Per fer-ho, el LCFIB ha donat suport en la generació dels serveis web necessaris mentre es realitzava el desenvolupament de les aplicació.

A la resta d'aquest capítol, s'explicarà en primer lloc les característiques principals de tot el què Android engloba (veure Secció 1.1) , és a dir, història, versions i estat en l'actualitat. En segon lloc, es detallarà la base en què està desenvolupat Android (veure Secció 1.2). En tercer lloc, es presentarà quina ha estat l'arquitectura i patrons de disseny escollits per a realitzar la implementació (veure Secció 1.3). Com a últim punt, es detallarà la descripció, objectius i metodologia de treball del projecte (veure Secció 1.4).

1.1. EL MÓN ANDROID

Abans de començar a entrar en detalls tècnics, s'ha realitzat un breu apartat introductori a tot el que Android envolta. En aquest es veurà, com va sorgir Android [8], les diferents versions que hi ha hagut fins l'actualitat [9], quins dispositius mòbils actualment l'integren i, finalment, quin és el seu estat actual en el mercat [10].

UNA MICA D'HISTÒRIA

Android és un sistema operatiu per a dispositius mòbils *SmartPhones* i *Tablet PC*. Fou desenvolupat per la *Open Handset Alliance* [11] liderat per Google. Està constituït per un consorci de 84 empreses per a desenvolupar estàndards oberts per a dispositius mòbils. Les empreses que la composen són Google, HTC, Sony, Dell, Intel, Motorola, Qualcomm, Texas Instruments, Samsung, LG, T-Mobile, Nvidia, i Wind River Systems.

Google a l'agost de 2005 va adquirir el desenvolupador inicial del *software* Android anomenat, *Android Inc*. Ara mateix, es considera una subsidiària totalment propietat de Google. No se sabia molt sobre *Android Inc*. en el moment de l'adquisició, però molts van assumir que Google estava planejant entrar al mercat de la telefonia mòbil amb aquest moviment.

A Google, l'equip dirigit per Rubin va desenvolupar una plataforma de dispositius mòbils impulsats pel nucli de Linux. Andrew Rubin és un pioner de la tecnologia, cofundador i ex CEO de *Danger Inc* i *Android Inc.* En l'actualitat és vicepresident sènior de mòbils de Google, on supervisa el desenvolupament d'Android. Tornant a Google, va comercialitzar la plataforma als fabricants de telèfons i operadors amb la premissa de proporcionar un sistema flexible i actualitzable. Google havia creat una sèrie de components i va informar als distribuïdors que estava obert a diversos graus de cooperació per la seva part.

Les especulacions sobre la intenció de Google d'entrar en el mercat de les comunicacions mòbils va continuar fins a desembre de 2006. En els mitjans impresos i punts de venda online es rumorejava que Google estava desenvolupant un telèfon marca *Google*. Al setembre de 2007, *InformationWeek* va realitzar un estudi on afirmava que Google havia presentat diverses sol·licituds de patents en l'àrea de la telefonia mòbil.

El primer llançament de la distribució d'Android el 5 de novembre de 2007 es va anunciar amb la fundació de *l'Open Handset Alliance*. Google va llançar la major part del codi sota la *llicència Apache*[4]. El projecte d'Android Open Source (AOSP) s'encarrega de la conservació i el desenvolupament.

Actualment, el sistema operatiu Android s'usa en telèfons intel·ligents, ordinadors portàtils, *netbooks*, ordinadors *Tablet*, Google TV, rellotges de polsera, auriculars i altres dispositius.

El primer telèfon disponible al mercat per executar Android va ser l'HTC Dream, presentat el 22 d'octubre de 2008. A principis de 2010 Google va col·laborar amb HTC per llançar el seu producte estrella, el Nexus One. El va seguir el 2010 el Samsung Nexus S, Samsung Galaxy I i II i ja el 2011 amb el Nexus Galaxy.

[4] La llicència *Apache* és una llicència de programari lliure *copyfree* escrit per la Apache Software Foundation (ASF). La llicència *Apache* requereix la preservació de la nota de *copyright* i una renúncia. Per una renúncia entenem qualsevol declaració de la intenció d'especificar o delimitar l'abast dels drets i obligacions que poden ser invocats i aplicats per les parts en una relació legalment reconeguda.

VERSIONS I PRINCIPALS CARACTERÍSTIQUES

En Android les versions porten nom d'aliments dolços. Quan surt una nova versió també es té en compte l'ordre alfabètic. Actualment, els codis de les versions que s'han desenvolupat són (en anglès): *Cupcake, Donut, Eclair, Froyo, Gingerbread, HoneyComb, Ice Cream Sandwich.* En aquest document, per veure amb claredat les característiques principals de cadascuna i millores que s'incorporen, no es presenten totes les subversions que ha sorgit de cadascuna. Per exemple, de la versió 2.0 hi ha la 2.0.1, i la 2.0.2, amb lo qual es presentaria un resum del més destacat de les tres.

Prèviament a les versions esmenades es va presentar, tal i com es pot apreciar a la *Figura 1* les versions 1.0, i 1.1, l'any 2008, i 2009 respectivament. Aquestes són les primeres versions Android que es van realitzar. Com a destacat, a part de les diferents funcionalitats que qualsevol sistema operatiu per a mòbils pot incorporar com: missatges, trucades, alarmes, etc. incorporava *Android Market* [5][28], un navegador per Internet, les diferents funcionalitats que Google ofereix com: Gmail, Contactes, Calendari, Maps, Sync, GTalk, i notificacions a la barra d'estat.

Al llarg de 2009, la versió 1.1 fou millorada i a l'abril del mateix es va publicar una versió estable anomenada *Cupcake* (versió: 1.5) i al mes d'octubre *Donut* (versió: 1.6). Els principals canvis i característiques que van aportar foren:

- Millores de rendiment. La càmera tenia una arrencada ràpida i el GPS una localització, també, més ràpida.
- S'incorporava el teclat en pantalla i *widgets* simples, tal i com es pot veure en les imatges de la *Figura 1*.
- Permetia publicar vídeos al Youtube i fotos a Picasa.
- Per realitzar cerques en el mòbil es va incloure una ajuda de teclat.
- Es podia consultar l'indicador de bateria on es mostrava quin era l'històric d'ús.
- *Android Market*, s'actualitzava amb una nova interfície per a l'usuari.

Eclair (versió: 2.0) i *Froyo* (2.2) es van publicar al llarg de 2010 (veure Secció Mercat i actualitat)**.** *Froyo* ha estat fins al novembre de 2011 la versió que més acceptació ha tingut, liderant amb diferència respecte les altres versions. Per aquest motiu es va decidir a l'inici del projecte desenvolupar per aquesta versió l'aplicació del *Racó Mobile*. A diferència de les anteriors descripcions, en aquesta, entrarem en més detall en les característiques.

La **pantalla d'inici** ha incorporat consells per ajudar als nous usuaris sobre com configurar la pantalla inicial amb accessos directes, *widgets* i com fer ús de múltiples pantalles d'inici. El telèfon, el llançador d'aplicacions i el navegador es troben en accessos directes a la pantalla inicial. De manera que és fàcil accedir-hi des de qualsevol dels 5 panells de la pantalla d'inici.

L'**Exchange Support** també s'ha vist millorat en diferents aspectes. La seguretat amb l'addició de PIN numèric o alfanumèric de les opcions de contrasenya per desbloquejar el dispositiu. També l'eliminació remota de dades. De forma remota es pot reiniciar el dispositiu als paràmetres de fàbrica per assegurar les dades en el dispositiu de cas de pèrdua o robatori.

[5] L'*Android Market* és una botiga *Online* d'aplicacions dissenyades per a dispositius Android. Els usuaris obtenen accés a l'*Android Market* a través del seu lloc web o bé mitjançant l'aplicació.

L'aplicació Calendari suporta diferents calendaris. Un altra aspecte rellevant és l'addició de l'auto-descobriment, només cal saber el nom d'usuari i contrasenya per a configurar fàcilment i sincronitzar amb un compte d'*Exchange*. Finalment i per acabar aquest punt, les llistes globals d'adreces *look-up* està disponible en l'aplicació de correu electrònic, facilitant als usuaris els noms dels destinataris. Emplenament automàtic al directori.

La galeria permet mostrar i seleccionar imatges mitjançant el zoom. Els botons de la càmera en pantalla faciliten l'accés a una nova interfície d'usuari per al control de zoom, flash, balanç de blancs, geo-etiquetatge, l'enfocament i l'exposició. **La càmera** també proporciona una manera fàcil de configurar la mida del vídeo / qualitat dels *MMS* i YouTube.

El flaix LED està habilitat per la càmera de vídeo. Els vídeos es poden gravar en la nit o en ambients amb poca llum.

Pel què fa a la utilitat de ***Portable HotSpot*** cal destacar, que certs dispositius com el Nexus One es poden convertir en un punt d'accés *Wi-Fi* que es pot compartir amb fins a 8 dispositius.

Es pot utilitzar el telèfon com una connexió 3G per a un portàtil amb Windows o Linux mitjançant la connexió del telèfon a l'ordinador amb un cable USB. La connexió es comparteix entre els dos dispositius.

La funcionalitat **multilingüe,** també s'ha millorat ja que els usuaris poden afegir diversos idiomes per al teclat i canviar entre els múltiples idiomes d'entrada basada en el llatí.

A principis de 2011 es va publicar *Gingerbread* (versió: 2.3). En quan a les novetats que va aportar, es detallen a continuació:

- La interfície d'usuari es refina en molts aspectes en tot el sistema. És més fàcil d'aprendre, ràpid d'usar i més eficient energèticament.
- Canvis en els menús i ajustos que fan que sigui més fàcil per a l'usuari navegar i controlar les funcions del sistema i el dispositiu.
- El teclat afegeix la possibilitat de corregir paraules introduïdes a partir de suggeriments al diccionari. També mostra el caràcter actual, suggeriments del diccionari més grans i l'estil més viu cosa que fa més fàcil de llegir.
- Selecció de paraules, copiar i enganxar.
- El sistema Android té un paper més actiu en la gestió d'aplicacions que mantenen el dispositiu despert durant molt temps o que consumeixen *CPU* mentre s'executa en segon pla. El sistema també proporciona informació als usuaris sobre el consum dels components del sistema i les aplicacions en execució.
- Quan l'usuari entra en l'administració d'aplicacions, una nova fitxa *Running* mostra una llista d'aplicacions actives, l'emmagatzematge i la memòria utilitzada per cada una.
  Una aplicació *NFC Reader* permet a l'usuari llegir i interactuar amb comunicació de camp proper (*NFC*). Per exemple, l'usuari pot "tocar" una etiqueta *NFC* integrada en un cartell, adhesiu, o publicitat. A continuació, actuar sobre les dades de lectura de l'etiqueta. Un ús típic seria llegir una etiqueta en un restaurant, botiga o esdeveniment, i llavors saltar a una pàgina web. La *URL* s'inclou en les dades de l'etiqueta. La comunicació *NFC* es basa en la tecnologia sense fils del dispositiu. El suport per les

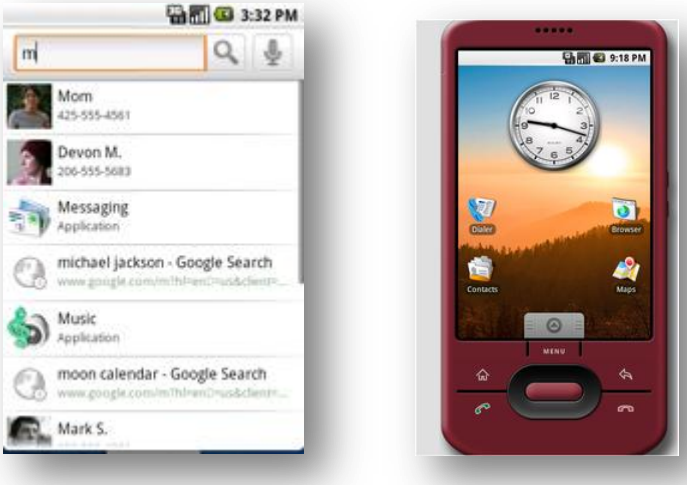

Figura 1. Imatges del dispositiu en la versió 1.0 i 1.1

característiques de la plataforma *NFC* en dispositius específics està determinat pels seus fabricants.

- L'aplicació permet a l'usuari accedir a diverses càmeres del dispositiu, incloent una càmera frontal, si està disponible.
- L'usuari pot fer trucades de veu a través d'Internet a altres usuaris que tenen comptes *SIP*. L'usuari pot afegir número de telèfon de Internet (una adreça *SIP*) a qualsevol contacte i pot iniciar una trucada de contacte ràpid.

Actualment també es dóna suport per *Tablet PC*. Les versions corresponents són *HoneyComb* (versió: 3.0) i des de principis d'octubre de 2011, *Ice Cream Sandwich* (versió: 4.0). Aquesta última, és una versió compartida per als dos dispositius. Si es desitja veure en més detall sobre cada una de les versions explicades es pot consultar la pàgina web de desenvolupadors d'Android [5].

Android ha realitzat una notable evolució des de la versió 1.0 fins a l'actual *Gingerbread*. A nivell d'usuari els canvis són notables ja que per exemple, ara és un sistema operatiu més configurable per a l'usuari. Aquest té més control sobre quines aplicacions hi ha actives i quines no, el sistema *Android Market* té una interfície molt més amigable, entenible i s'ha inclòs el sistema *NFC*, entre d'altres. Destacar també, que totes aquestes funcionalitats són accessibles per a desenvolupadors. Mitjançant l'*SDK* d'Android, el qual es veurà més endavant, es pot accedir a aquestes funcionalitats i adaptar-les a les nostres aplicacions.

Un cop vistes les diferents versions i tenint el coneixement del potencial que ofereix Android, es mostra a continuació, la seva situació en el mercat.

ACTUALITAT I MERCAT

A l'inici del projecte es va realitzar l'estudi previ de les versions Android i el seu establiment en els dispositius. L'objectiu era desenvolupar per la versió més estesa per tal de garantir la màxima acceptació del projecte als usuaris finals. També es volia valorar si la versió escollida suportaria les diferents funcionalitats que es volien incorporar en el projecte.

Plataforma	API (nivell)	Distribució en %
Android *HoneyComb* 3.0	11	0,3%
Android Gingerbread 2.3.3	10	3,0%
Android Gingerbread 2.3	9	1,0%
Android Froyo 2.2.x	***8***	***65,9%***
Android Eclair 2.0. x / 2.1. x	7	24,5%
Android Donut 1.6	4	3,0%
Android Cupcake 1.5	3	2,3%

Figura 2. Establiment del sistema operatiu Android. Dades oficials del dia 2 de maig de 2011

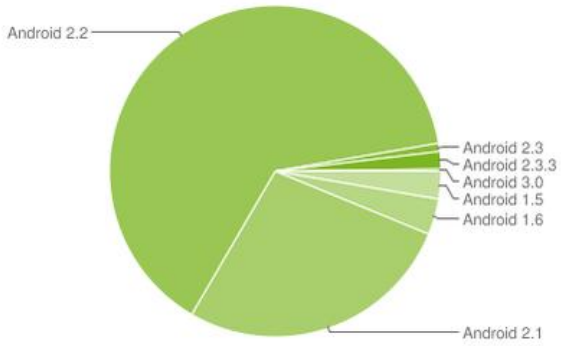

Figura 3 . Establiment del sistema operatiu Android. Dades oficials del dia 2 de maig de 2011

En la *Figura 2* es pot veure un quadre detallat amb tant per cent de la distribució actual de les aplicacions. La columna central *API* indica el mínim nivell que ha de suportar el telèfon per poder disposar de la distribució. En la *Figura 3* es pot observar un gràfic on es veuen les diferents versions d'Android que hi havia en aquell moment en el mercat. Android va tenir una fort creixement en el mercat a l'inici de 2011, i com a conseqüència va fer que la versió 2.2 tingués més èxit. A principis de març es va publicar la versió 2.3 i encara no estava del tot instaurada al mercat.

Al mes de novembre de 2011 s'han consultat de nou les dades per veure quin era l'estat d'acceptació de les versions. L'aplicació *Racó Mobile* al llarg del mes de desembre serà publicada a l'*Android Market*. Aquest fet, provoca la necessitat de saber quins usuaris podran descarregar i instal·lar l'aplicació.

Com es pot veure a la *Figura 4 Gingerbread* ha guanyat molt de terreny, actualment té un 43,9% del mercat davant del 40,7% de *Froyo*. No obstant, no es considera una adversitat ja que les funcionalitats que actualment oferirà el *Racó Mobile* són compatibles amb les versions posteriors a *Froyo*. En la *Figura 5* s'observa en detall l'estat actual de la resta de versions.

Destacar que les versions prèvies a la 2.X han perdut participació en el mercat respecte al mes de febrer. No obstant, la més perjudicada ha estat la versió 2.2. Aquest fet ha estat degut a que el març, tal i com s'ha mencionat anteriorment, va aparèixer la nova versió. Fins a l'actualitat han passat 9 mesos i les companyies han actualitzat els seus dispositius.

Plataforma	API (nivell)	Distribució en %
Android *HoneyComb* 3.0	11	10,1%
Android Gingerbread 2.3.3	***10***	***43,9%***
Android Gingerbread 2.3	9	0,5%
Android Froyo 2.2.x	***8***	***40,7%***
Android Eclair 2.0. x / 2.1. x	7	10,4%
Android Donut 1.6	4	1,4%
Android Cupcake 1.5	3	0,9%

Figura 4. Establiment del sistema operatiu Android. Dades oficials del dia 3 de novembre de 2011

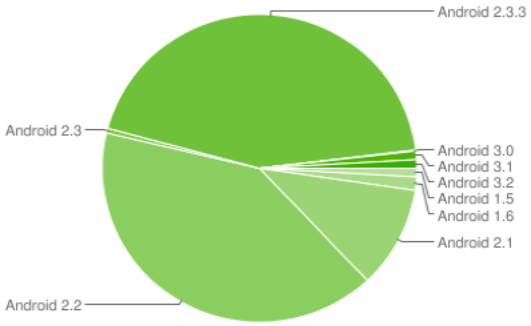

Figura 5. Establiment del sistema operatiu Android. Dades oficials del dia 3 de novembre de 2011

1.2. PRELIMINARS PEL DESENVOLUPAMENT EN ANDROID

En el sistema operatiu Android podem destacar dues bases principals. La primera d'elles és el seu nucli basat en Linux. La segona, el llenguatge de programació és bàsicament Java. Sobre aquests dos fets en parla el llibre "*Android in Practice*" [1]**,** i en destaca diferents aspectes que es presenten a continuació.

ANDROID I LINUX

L'afirmació, Android està basat en un sistema Linux, és relativa ja que, Linux és un sistema operatiu basat en el Kernel de Linux [11], és a dir, la distribució *GNU*/Linux. Aquesta, inclou el Kernel de Linux i el conjunt estàndard d'aplicacions del sistema específiques per aquest. Exemples dels quals poden ser Ubuntu[6] o Red Hat[7] els quals contenen el Kernel i les aplicacions *GNU*.

Pel què fa a Android, difereix d'aquesta definició ja que no inclou moltes de les aplicacions que tot sistema *GNU*/Linux incorpora (en especial X11, que gestiona la part gràfica de l'entorn). De fet, Android no conté la llibreria estàndard del llenguatge *C* que conté *GNU*. Android porta una implementació pròpia, optimitzada i adaptada per a mòbils anomenada *Bionic*. Les aplicacions realitzades per *x86 GNU*/Linux, no es poden incorporar per defecte a Android, a menys que, hagin estat compilades prèviament per la llibreria *C* d'Android (*Bionic*).

[6] http://www.ubuntu.com/
[7] http://www.redhat.com/

Directori	Descripció
/sdcard	Aquest és el punt de muntatge de la tarja *SD* la qual es pot trobar a molts dispositius Android. Aquí és on hi ha emmagatzemada tota la informació de la tarja *SD* (fitxers, imatges, àudio, etc.)
/data/app	Aquí és on Android guarda totes les aplicacions instal·lades. Els arxius *.APK*
/data/data	Aquí és on Android guarda les dades específiques de les aplicacions. Per exemple, les preferències, llibreries pròpies, etc.

Figura 6. Principals directoris del sistema operatiu Android

Pel què fa al sistema de fitxers Linux utilitza una estructura en forma d'arbre, el qual l'element "arrel" s'anomena "*root*" o bé "*/*". Android, també conté un sistema similar a aquest, amb la navegació en forma d'arbre i utilitzant les mateixes comandes que s'usen en Linux. La *Figura 6* mostra els principals directoris que es troben en un sistema Android.

Un tema que també cal destacar és els permisos d'accés als directoris. Els permisos en Android, si s'opera mitjançant la línia de comandes tenen el mateix format que el sistema operatiu Linux. Cal destacar que, quan una aplicació s'instal·la al dispositiu, es crea un nou compte per a aquesta aplicació, la qual té accés únicament als fitxers que a ella fa referència. Els accessos als fitxers de sistema i de les altres aplicacions els hi són restringits. Amb lo qual, aquí tenim el motiu de perquè les aplicacions en Android, per defecte i normalment, no poden compartir informació.

El sistemes operatius actuals, permeten que hi hagi varis processos actuant en paral·lel. Aquests operen segons la quantitat de memòria i espai que se'ls assigna. Android també conté aquesta característica, amb la diferència que l'usuari només pot veure'n en pantalla un d'ells. El fet de que sigui multitasca és important en plataformes on la interacció entre aplicacions és un requeriment que l'usuari cada vegada sol·licita més. Android dóna preferència a les aplicacions d'usuari que en aquell moment estan interactuant amb ell, o que han estat usades recentment. Totes aquestes aplicacions estan en la pila d'execució. Val a dir, que quan el sistema necessita espai i/o memòria, és capaç de tancar les aplicacions automàticament.

ANDROID I JAVA

Qualsevol entorn Java el podem identificar per la llibreria que conté les classes on hi ha la plataforma Java [12], incloent les utilitats dels llenguatges, xarxa o concurrència entre d'altres. La segona part fa referència a la màquina virtual de Java, la qual, serveix per interpretar el conjunt de classes (*.class*) generades un cop compilades pel compilador Java, i que s'explica en l'apartat "l'Emulador".

L'entorn de desenvolupament d'Android segueix aquest patró. No obstant, l'intèrpret no genera arxius *.class* sinó que interpreta un altra tipus d'arxius. De totes maneres, permet utilitzar gran part de les llibreries que Java proporciona en la seva plataforma. Aquestes llibreries estan basades en l'*Apache Harmony Project.* La màquina que es fa servir per executar les aplicacions en Java s'anomena *Dalvik* [16].

El fet d'estar basades en aquest projecte no implica que estan incloses en la seva totalitat, sinó que només s'han inclòs les llibreries principals per poder desenvolupar per a dispositius mòbils. En el llibre "*Android in Practice"* es poden trobar en detall un recull de les llibreries que han estat incloses i les que no. En la *Figura 7* es mostra un resum.

Implementat	Parcialment implementat	No implementat
java.io	java.awt	java.applet
java.math	java.beans	java.rmi
java.nio	java.lang	java.accessibility
java.security	java.net	javax.activation
java.sql	java.util	javax.activity
java.text	javax.security	javax.annotation
java.util.concurrent		javax.imageio
java.util.logging		javax.jws
java.util.regex		javax.lang.model
javax.crypto		javax.management
javax.net		javax.naming
javax.sql		javax.print
javax.xml		javax.rmi
org.w3c.dom		javax.script
org.xml.sax		javax.sound
		javax.swing
		javax.tools
		javax.transaction
		org.ietf.jgss
		org.omg

Figura 7. Estat dels paquets de Java en les llibreries Android

Com es pot veure a la Figura anterior les llibreries d'interfícies com *AWT* i *Swing*, no estan incloses degut a que no són compatibles amb Android. Android proveeix la seva pròpia llibreria d'interfície, que ja es pot avançar que és en *XML* (veure Secció 4.2). Pel què fa a les llibreries que fan referència a continuació, a part de comentar si s'han inclòs o no en el sistema Android i realitzar una breu descripció, són les que al llarg de la implementació s'han utilitzat.

La llibreria d'execució *java.lang* s'ha inclòs. També el paquet de *java.util* que conté les estructures de clau valor que es poden necessitar. Els paquets *java.io* i *java.net* per tal de llegir les dades de fitxers i d'Internet. I els paquets *java.nio* que permeten llegir i escriure dades de manera asíncrona.

Els paquets *java.sql* i *javax.sql* permeten connectar a base de dades relacionals. Mitjançant les llibreries *org.xml.sax* i *org.w3c.dom* es poden parsejar dades en *XML*. No obstant, en alguns casos pot ser que s'hagi d'incloure llibreries externes (*org.xmlpull.v1*) per tal de parsejar alguns fitxers *XML* o bé la llibreria (*json.org*) per a fitxers *JSON*.

Finalment per, poder realitzar connexions tenim la llibreria *java.net*. No obstant, si s'ha d'interactuar amb *cookies*, redireccions, autenticació, etc. Es pot considerar usar la classe *HttpClient*.

Arribats aquest punt ja s'ha assolit una perspectiva del sistema operatiu Android i les seves principals components. En el punt que es detalla a continuació, s'explica els detalls sobre quin és l'entorn de desenvolupament i l'*IDE*[8] [15] recomanat per desenvolupar aplicacions.

[8] Integrated Development Environment

1.2.1. CARACTERÍSTIQUES DE L'ENTORN DE DESENVOLUPAMENT

Android és un sistema multiplataforma que permet desenvolupar aplicacions des de qualsevol tipus d'entorn (Linux, Windows o Mac). En un principi, l'*SDK* (el qual es descriu més endavant) només estava suportat per a Linux i Windows, més endavant, es va alliberar una versió igual a la que ja hi havia en el mercat, per poder desenvolupar des de sistemes operatiu de d'Apple.

Pel què fa l'*IDE* de desenvolupament no està restringit a cap estrictament. Ara bé, les prestacions que es poden extreure d'uns o altres varien notablement. A continuació, es presenta quin *IDE* és el que s'ha fet servir, que alhora, és el mateix que es recomana a la pàgina web d'iniciació a Android i els diferents llibres que s'han consultat (per exemple, *Hello, Android Introducing Google's Mobile Development Platform* [2]). *Eclipse*, és el software que els desenvolupadors de Google han escollit per donar suport. Per aquest *IDE* existeix un *Plugin* que es pot descarregar gratuïtament. Aquest, inclou totes les eines necessàries per poder desenvolupar aplicacions Android. També, inclou projectes de mostra amb el codi font disponible, un emulador i les biblioteques necessàries per crear aplicacions Android.

A continuació es presenta l'eina que permet realitzar les proves emulant el dispositiu.

L'EMULADOR

Dalvik, és l'eina que fou dissenyada i desenvolupada per compartir i optimitzar l'execució per sistemes amb dispositius mòbils. *Dalvik* està basada en un sistema UNIX incloent *Vanilla Linux*, *BSD*, i *MacOS X*. Els tres aspectes bàsics pel què fou dissenyada foren:

- Ha de ser ràpid, fins i tot amb *CPU*'s amb poca potència.
- Ha d'executar-se en sistemes amb poca memòria.
- Ha d'executar-se amb eficiència d'energia.

La principal funció que se li aplica és la de depurar les aplicacions, afegir i eliminar fitxers a directoris del sistema operatiu de l'emulador. Es poden aplicar dues tècniques:

1. El *Android Debug Bridge* també conegut com a *ADB*. Permet mitjançant la línia de comandes comunicar-se amb l'emulador o el dispositiu físic connectat a l'ordinador via *Telnet*. És un programa de tipus *client-servidor* que inclou tres components:
    - El client que està en la màquina de desenvolupament de l'aplicació.
    - El servidor que s'executa en un segon pla, en el mateix equip on es desenvolupa el projecte. Aquest, gestiona la comunicació entre el client i el *ADB* que està en l'emulador.
    - Un *daemon*, que s'executa en la part de l'emulador del dispositiu.

    Destacar que per a sistemes operatius Linux, és una eina molt recomanable, ja que se li pot extreure un gran potencial a nivell de depuració.

2. El *Dalvik Debug Monitor Server* també conegut com a *DDMS*. Aquesta eina permet depurar l'aplicació, extreure informació sobre l'estat de memòria i el consum, realitzar captures de pantalla, simular trucades entrants i missatges mitjançant la suplantació d'identitat i falsificació de dades de localització, entre d'altres opcions. Destacar que aquesta eina està incorporada al *Plugin* que s'instal·la a l'*Eclipse*.

Per tan doncs, tenim que l'*SDK* d'Android inclou un emulador de dispositiu virtual de mòbils que s'executa a l'ordinador. Permet realitzar prototipatge, desenvolupament i prova de les aplicacions per Android sense necessitat d'utilitzar un dispositiu físic. Com s'ha pogut veure l'emulador imita totes les característiques de maquinari i programari d'un dispositiu mòbil normal, excepte que no pot fer trucades reals. Ofereix una varietat de tecles de navegació i control, que es pot "pressionar" amb el ratolí o el teclat per generar esdeveniments a l'aplicació. També ofereix una pantalla en què es mostra l'aplicació, juntament amb les altres aplicacions d'Android.

L'emulador és una aplicació basada en *QEMU*[9] [17], que proporciona un dispositiu mòbil *ARM* virtual en el qual es poden executar les aplicacions d'Android. S'executa des de la pila d'Android, fins al nivell de nucli, que inclou un conjunt d'aplicacions preinstal·lades (com el marcador de text o telèfon) que es pot accedir des de les aplicacions. Es pot triar quina és la versió del sistema Android que desitja executar-se mitjançant la configuració de l'emulador en l'*AVD*. Es poden configurar varis emuladors alhora i posar-los en marxa al mateix temps. Cada un d'ells pot tenir un tamany diferent, i es pot comprovar com es veu l'aplicació amb les diferents pantalles que actualment pot tenir instal·lat un sistema operatiu Android.

Les capes de *QEMU* poden proporcionar serveis de traducció binària de manera dinàmica del codi de la màquina *ARM* pel sistema operatiu i l'arquitectura del processador de l'equip de desenvolupament.

Un cop vist, com és i què ofereix l'emulador creat per desenvolupar aplicacions en Android, es presenta a continuació una de les *APIs* d'Android que s'ha usat en el projecte i que el *Plugin* d'Android no porta incorporat. No obstant, fàcilment es poden instal·lar i permetre als usuaris integrar fàcilment funcionalitats a la seva aplicació.

L'API DE GOOGLE MAPS

En el *Racó Mobile*, s'han usat dues funcionalitats que Android ofereix als desenvolupadors. Aquestes són la geolocalització i la incorporació de Google Maps [18] al projecte. Amb l'*API* de geolocalització es poden realitzar aplicacions de cerca de posició de l'usuari. Com es veurà més endavant, aquesta funcionalitat s'ha aplicat en el projecte en el moment de mostrar quins són els lloc més comuns a visitar per a persones que assisteixen a la FIB o el campus Nord per primera vegada. Pel què fa a la *API* de Google Maps, permet mostrar el mapa tal i com es pot veure mitjançant el navegador web. També afegir punts estàtics per a mostrar informació sobre aquests. Quan es crea el projecte s'ha de tenir en compte la versió amb què serà desenvolupada l'aplicació i aquesta, indicar-li que es vol fer servir la *API* de Google Maps. De la mateixa manera, quan es crea l'emulador per realitzar les proves corresponents s'ha de tenir en compte de la versió i configurar-lo perquè executi Google Maps.

Un cop configurat el projecte veiem com gestionar la funcionalitat mitjançant el codi. Alhora de crear la vista que mostrarà el mapa, s'ha d'implementar una classe Java que hereti d'un *MapView*. Aquesta realitza el procés de mostrar el mapa. Per tal de manipular-lo es poden

[9] (**Q**uick **EMU**lator) programari de codi obert per a la creació de l'emulació i els entorns de màquines virtuals que s'utilitza per executar sistemes operatius i les aplicacions escrites per a una altra plataforma, per exemple, executar el programari ARM en un PC basat en x86.

```
private class Localitzacio extends Mapview implements LocationListener
```

Figura 8. Capçalera de la classe Localització del projecte

sobreescriure els mètodes que es creguin convenients. Fins i tot, afegir-ne per tal de realitzar les funcions que es creguin necessàries.

Pel què fa a la geolocalització, es basa en l'emissió d'un senyal que es pot escollir entre via GPS o via el proveïdor d'Internet del telèfon. Gestionar correctament quan començar o deixar de rebre el senyal és molt important. Aquest tipus de funcionalitat consumeix més bateria de lo normal, per tant, s'ha de saber gestionar eficientment. En quan a escollir l'ús de GPS o Internet depèn de lo acurats que es vulgui ser. El *GPS* és més precís, només funciona a l'aire lliure, consumeix la bateria ràpidament, i no actualitza la posició tan aviat com els usuaris voldrien. Mitjançat la ubicació de xarxa es determina la ubicació de l'usuari utilitzant torres de telefonia mòbil i els senyals *Wi-Fi*, proporcionant informació sobre la ubicació tan en interiors com en exteriors, respon més ràpid i utilitza menys bateria, per contra, és bastant menys precís. Pel què fa a la implementació de la geolocalització, s'ha d'heretar d'una classe que s'anomena *LocationListener* la qual permet usar els mètodes per poder capturar els events de canvi de posició i activar/desactivar la captura de posició. Així doncs, la capçalera de la classe seria equivalent a la *Figura 8*.

1.3. PATRONS DE DISSENY

Fins ara hem repassat les característiques més importants de l'entorn Android. No obstant, quan es planteja el desenvolupament d'una projecte és recomanable disposar d'una bona planificació i estructura de l'aplicació que es vol crear. Afortunadament, existeixen diferents patrons de disseny que faciliten la presa de decisions en funció del tipus de projecte. En la Iteració 1 (veure Secció 3.4) es pot veure en detall quines van ser les opcions que es van valorar alhora de planificar el projecte.

Els patrons de disseny són: *un conjunt de solucions a problemes que generalment trobes en el disseny d'un programa. Cada patró explica com resoldre un determinat problema, sota determinades circumstàncies, i els avantatges i desavantatges del seu ús* [2].

Es poden usar els patrons de disseny com a guia per resoldre problemes comuns en programació. Un cop el desenvolupador coneix els diferents patrons de disseny que existeixen o simplement un ampli ventall d'opcions a usar, és capaç d'identificar correctament els problemes que es presenten o que puguin aparèixer. Podrà identificar quina solució és la que millor s'aplica al problema, treballant, en aquest sentit, de manera proactiva. També s'ha de ser conscient que cap solució és perfecta, ja que, es pot aplicar una solució que pot perjudicar el projecte per algun altra sentit. Per exemple, ens pot treure flexibilitat en alguna part del programa.

Finalment, a nivell professional conèixer una gran varietat de patrons de disseny pot ajudar a tenir un llenguatge comú amb altres programadors. Quan es realitza un treball juntament amb altres persones, quan s'hagi de modificar un programa l'autor del codi segurament haurà d'explicar què va ser el que va fer, com funciona el seu codi i per què va fer

aquest o aquell canvi. Aquest fet pot demorar-se en hores. Aquesta conversa pot escurçar-se si diu: "Aquí he utilitzat aquest patró", per exemple. Darrere d'aquesta sola paraula, ja es coneix els conceptes, i se sap com ha estat realitzat el programa.

QUAN UTILITZAR PATRONS DE DISSENY

Els patrons de disseny són un catàleg de solucions ja provades i que ens permeten millorar el cicle de vida i desenvolupament del software. De la mateixa manera que no és aconsellable optimitzar prematurament, no s'han d'utilitzar patrons de disseny abans d'hora. Així doncs, és en l'etapa de disseny on es pot plantejar la seva aplicació en el projecte. En la majoria d'ocasions és recomanable implementar un prototipus primer i assegurar-se que funciona, per després utilitzar el patró de disseny per a millorar les flaqueses.

El seu ús pot incrementar o disminuir la capacitat de comprensió d'una implementació, augmentar la quantitat de codi, disminuir la modularitat i separar millor els conceptes. La majoria de desenvolupadors utilitzen patrons de disseny quan perceben un problema en el seu projecte el qual hauria de resultar senzill i no ho és, per exemple, el rendiment. La referència més utilitzada en el tema dels patrons de disseny és el llibre de la "*banda dels quatre*", *Design Patterns: Elements of Reusable Object-Oriented Software* realitzat per Erich Gamma, Richard Helm, Ralph Johnson i John Vlissides, Addison- Wesley, 1995. Destacar, que els patrons de disseny són molt populars en l'actualitat, de manera que no deixen d'aparèixer nous llibres. Un altra llibre a destacar també, és el que s'ha utilitzat per a realitzar el projecte "*Applying UML and Design Patterns*" [3].

En resum els patrons de disseny són:

- Una solució estàndard per un problema comú de programació.
- Una tècnica per flexibilitzar el codi mitjançant la satisfacció de criteris.
- Un projecte o estructura d'implementació que aconsegueix una finalitat determinada.
- Una forma pràctica de descriure certs aspectes de l'organització d'un programa.

A continuació es presenta l'arquitectura Model-Vista-Controlador i alguns dels patrons de disseny més habituals.

1.3.1. ARQUITECTURA MODEL – VISTA – CONTROLADOR

És l'arquitectura més comuna i usada en el món del desenvolupament d'aplicacions per a dispositius mòbils. Concretament, és el que s'usa en iOS i Android. Es tracta de realitzar un disseny que separi la vista del model, amb la finalitat de millorar la reusabilitat. D'aquesta manera les modificacions en les vistes no impacten en la lògica de negoci i de dades.
En la *Figura 9* es mostra l'esquema típic del *MVC*. Els elements bàsics són:

- Model: Dades i regles de negoci.
- Vista: Mostra la informació a l'usuari
- Controlador: Gestiona les entrades del usuari i fa d'enllaç entre les vistes i el model.

El *model* és el responsable de:

- Accedir a la capa d'emmagatzament de dades. El cas ideal és que el model sigui independent del sistema de dades.

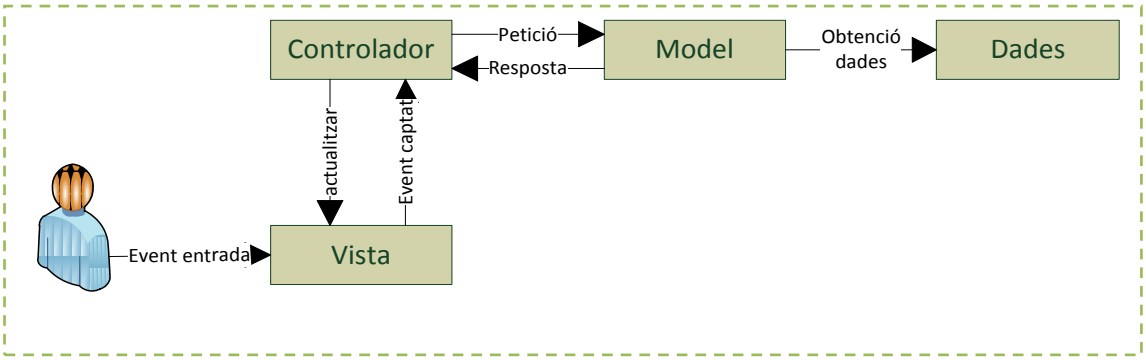

Figura 9. Arquitectura Model-Vista-Controlador

- Defineix les regles de negoci (funcionalitat del sistema).
- En cas que el model sigui actiu, notificarà els canvis que en les dades pugui produir un agent extern.

El *controlador* és el responsable de:

- Rebre els events d'usuari transmesos per les vistes, (un clic, un canvi en un camp de text,etc).
- Conté regles de gestió d'events, del tipus "Si Event X, llavors acció Y". Aquestes accions poden suposar peticions al model o a les vistes. Una d'aquestes peticions a les vistes pot ser una crida a un mètode "Actualitzar()". Una petició al model pot ser "Obtenir informació d'una assignatura".

Les *vistes* són responsables de:

- Rebre les dades del model a través dels controladors i mostrar-les a l'usuari.
- En Android i iOS entre d'altres, el controlador té el registre sobre quina vista té associada.
    - Poden donar el servei de "Actualització()", perquè sigui invocat pel controlador.
    - En el *MVC* el controlador s'encarrega de donar el servei d'actualització a les vistes. En Android les vistes són fitxers *XML* (veure Secció 4.2).

1.3.2. Altres Patrons de disseny

A nivell global i d'estructura de projecte s'ha aplicat l'anterior patró explicat, no obstant, també s'han usat altres patrons de disseny. Per tal de facilitar la lectura i comprensió de cada patró s'ha seguit una estructura detallant, en quin context passa, quin és el problema, quina és la solució que ofereix aquest i a on s'ha aplicat del projecte [2] [4].

Patró controlador

- Context:
    - Els (sub)sistemes software reben esdeveniments.
    - Un cop interceptats aquests esdeveniments, algun objecte del sistema ha de rebre'ls i executar les accions corresponents.
- Problema:
    - Quin objecte és el responsable de rebre un esdeveniment?

- Solució:
  - Assignar aquesta responsabilitat a un controlador. Els clients del sistema desconeixen l'estructura interna del sistema.
  - Un controlador és un objecte d'una certa classe. El controlador delega sobre un o més objectes del sistema el tractament de l'esdeveniment.
  - L'objecte que tracta l'esdeveniment no té coneixement sobre l'existència o el tipus de controlador.
- Variants analitzades i que s'han practicat en el projecte:
  - Façana: Un objecte que representa tot el sistema.
  - Cas d'ús: Un objecte que representa una instància d'un cas d'ús.

CONTROLADOR FAÇANA

Partint de la base que el context i el problema són comuns a l'anterior, passem a detallar la solució:

- Classe singletó
  - Tantes operacions com esdeveniments ha de capturar el sistema.
  - Eventualment poden incloure's atributs per compartir informació
- Controladors inflats si hi ha molts esdeveniments, poca cohesió.
- Aplicat:
  - A cada cas d'ús se li ha creat el seu controlador particular.

CONTROLADOR CAS D'ÚS

Partint de la base que el context i el problema són comuns a l'anterior, passem a detallar la solució:

- S'associa un *controlador cas d'ús* a cada cas d'ús definit al sistema.
- Aspecte estàtic: tantes noves classes com casos d'ús té el sistema.
  - Cada classe declara les operacions del diagrama de seqüència corresponent.
  - Eventualment, poden incloure's atributs per compartir informació (estat del cas d'ús).
- Aspecte dinàmic: similar al cas anterior.
  - El no ser *singletó*, cal crear-los i destruir-los quan es necessiten.
- Millora la cohesió del sistema.
- Aplicat:
  - A cada cas d'ús se li ha creat el seu controlador particular.

PATRONS D'ASSIGNACIÓ DE RESPONSABILITATS

- Context:
  - S'ha de determinar quin objecte és el responsable de captar l'esdeveniment extern i donar una resposta.
- Problema:
  - Qui s'encarrega de realitzar-ho? A qui li destinem la responsabilitat?
- Solució:
  - Destinar-ho a qui pot gestionar millor l'esdeveniment. A qui té la informació necessària per poder o bé delegar-la o bé gestionar-la per si mateix.
- Variants analitzades i que s'han practicat en el projecte:
  - Patró Expert.

- Patró Plantilla.

PATRÓ EXPERT

Pel què fa al context i el problema és el mateix que la descripció anterior. No obstant, la solució és pot concretar en més detall. A continuació l'explicació:

- Solució:
  - Assignar una responsabilitat a la classe que té la informació necessària per realitzar-la.
  - L'aplicació del patró requereix tenir clarament definides les responsabilitats que es volen assignar (postcondicions de les operacions).
  - No sempre existeix un únic expert, sinó que poden existir diversos experts parcials que hauran de col·laborar.
- Beneficis:
  - Es manté l'encapsulament i baix acoblament.
  - Conducta distribuïda entre les classes que tenen la informació alta cohesió.
- Aplicat:
  - Cada controlador de l'aplicació és expert d'ell mateix, és a dir, s'autogestiona.

PATRÓ PLANTILLA

En aquest cas, el context, problema i solució té la mateixa idea que la inicial plantejada, no obstant, preferim concretar-la més ja que es tracta d'un cas particular.

- Context:
  - La definició d'una operació en una jerarquia té un comportament comú a totes les subclasses i un comportament específic per cadascuna d'elles.
- Problema:
  - Repetir el comportament comú a totes les subclasses implica duplicació de codi i un manteniment més costós.
- Solució:
  - Definir l'algorisme (l'operació) a la superclasse, invocant operacions abstractes (amb signatura definida en la superclasse i mètode a les subclasses) que són redefinides a les subclasses.
  - L'operació concreta s'anomena *plantilla.*
  - Les noves operacions s'anomenen *primitives.*
  - L'operació de la superclasse defineix la part del comportament comú i les operacions abstractes la part del comportament específic.
- Aplicat:
  - Aplicat a la classe de gestió de la connexió. Primer s'obté la informació i posteriorment cada classe que hereta d'aquesta pot realitzar l'actualització de la vista i de dades de la manera més adient.

PATRÓ OBSERVADOR

- Context:
  - Quan una abstracció té dos aspectes: una dependent de l'altra.
  - Quan un canvi en un objecte requereix canviar-ne d'altres, i no se sap quants.
  - Quan un objecte ha de poder avisar altres objectes, sense fer supòsits sobre qui són.

- Problema:
  - Sovint passa que les dades canvien en un lloc, però altres components depenen d'aquestes dades.
  - Això es podria resoldre introduint una crida directa als objectes dependents: inflexible i no reusable.
  - Forces a equilibrar:
    - Els canvis d'estat d'un component s'han de notificar a altres components.
    - El nombre i la identitat dels dependents són desconeguts, i poden canviar.
    - La requesta explícita de nova informació per part dels dependents no és factible.
    - El publicador i els seus dependents no haurien d'estar acoblats fortament.
- Solució:
  - Els objectes dependents *(observadors*) se subscriuen per rebre notificacions.
  - L'objecte independent *(subjecte*) notifica els seus canvis d'estat als observadors.
  - S'ha d'assegurar que l'estat és consistent.
  - Quan un observador rep notificació de canvi d'estat, actualitza el seu estat. O bé li arriba informació amb la notificació (empenta), o bé li demana al subjecte (estirada).
- Aplicat:
  - Aplicat a la classe gestió connexió i controladors. Realitzen crides asíncrones. El controlador pot continuar gestionant events, mentre es processa la informació amb el servidor. El servidor un cop té les dades avisa al controlador i si la vista encara està activa es refresca la pantalla, altrament, només es guarda la nova informació a la base de dades.

1.4. OBJECTIUS DEL PROJECTE

Tot projecte té una finalitat o finalitats. Aquest fet és el que determina l'èxit o el fracàs del projecte. Els objectius, han d'estar ben definits quan el desenvolupament d'aquest comença. Les diferents parts que participen en la definició dels objectius, s'entén que tots persegueixen un mateix objectiu final, han d'estar d'acord amb tots els subobjectius definits prèviament. En cas que no sigui així, s'entra en un procés iteratiu on les diferents parts involucrades en el projecte han de posar-se d'acord tant amb els objectius individuals com comuns.

Pel què fa el projecte, la definició dels objectius va quedar molt clara des d'un inici, amb lo qual, no fou necessari entrar en el procés iteratiu anteriorment mencionat. A continuació, es presenta la descripció dels diferents objectius que ha comportat la realització del *Racó Mobile*.

1.4.1. DESCRIPCIÓ I OBJECTIUS

La realització d'aquest projecte engloba varis objectius personals i institucionals. L'objectiu principal i que involucra les dues parts del projecte seria: Aconseguir que el Racó de

Usuari	Servei
Nous Estudiants i externs	a. Notícies de la facultat b. Localització geogràfica de la facultat. c. Assignatures que s'imparteixen a a la facultat.
Estudiant	a. Tots els anteriors. b. Accés al Racó: consulta d'assignatures que està cursant, el correu personal, ocupació de les aules, horari personal i agenda personal.

Figura 10. Usuaris a qui va dirigida l'aplicació i serveis que podran usar

l'estudiant de la Facultat d'Informàtica de Barcelona disposi d'una aplicació per a telèfons mòbils. Concretament, *Smartphones* amb el sistema operatiu Android.

Altres objectius que complementen l'anterior els trobem descrits a continuació, comencem pels objectius institucionals, és a dir, de la facultat.

La Facultat d'Informàtica vol donar als seus estudiants, futurs estudiants o bé persones interessades en la facultat, un servei actualitzat, mitjançant el telèfon mòbil, tal i com ja estan realitzant altres universitats.

Donar el màxim de facilitats en l'accés a la consulta de l'actualitat de la facultat des de medis diferents, en aquest cas el mòbil. Tal i com es pot veure a la *Figura 10* diferenciarem tres tipus d'usuari segons el servei que poden usar.

A nivell personal també podem destacar vàries motivacions per a la realització del projecte:

- Elaboració d'un projecte propi, partint de la negociació amb el client, passant pels successius passos que la realització d'un projecte comporta fins arribar a la implementació i entrega del mateix.
- Treballar amb tecnologia actual i que des de fa relativament poc temps ha sorgit al mercat, com és la de desenvolupar aplicacions per sistemes Android.
- Col·laborar amb un projecte que afecta a la facultat on un mateix ha estudiat els darrers anys. I així, poder ajudar a que la facultat sigui més accessible als estudiants, nous estudiants i externs.
- La possibilitat de posar en pràctica tot el coneixement que s'ha adquirit al llarg de la carrera i així, demostrar-se a un mateix que l'objectiu ha estat assolit amb èxit.

Tot projecte, per tal de tenir èxit i obtenir un bon resultat, ha d'estar degudament planificat i amb una metodologia de treball definida. A continuació es presenta la metodologia que s'ha seguit en el desenvolupament del projecte.

1.4.2. METODOLOGIA DE TREBALL

Durant els primers contactes entre el director del projecte i un mateix, es va definir el període de reunions i les principals reunions que s'esdevindrien en el projecte.

El primer de tot que es va establir fou que hi hauria una reunió setmanal, concretament, cada dijous. En aquesta s'exposaria la feina pactada en la reunió prèvia i s'establiria l'objectiu de la següent setmana. Utilitzant aquesta metodologia s'ha aconseguit i/o s'aconseguiria la divisió de grans objectius en objectius menors. Amb lo qual, fou totalment assequible assolir els

principals punts per realitzar el projecte en les dates establertes, sempre assolint, que podia haver-hi un marge de demora.

Així doncs, s'ha treballat durant cada dia de la setmana intensament per poder assolir l'objectiu setmanal marcat. En les reunions, es parlava dels problemes trobats, com s'havien solucionat (sempre hi quan s'hagués trobat la solució), i quines possibles evolucions podria tenir aquell objectiu. En cas que l'ampliació fos necessària es realitzava, en cas de que fos prescindible, es deixava a la llista de tasques pendents.

Cap al mes de setembre, quan el projecte es va començar a documentar de manera profunda, es va decidir que les reunions es realitzarien depenen de la conveniència i l'evolució d'aquesta. Aquest fet, es va produir degut a que en una setmana no es podien corregir els errors que es reportaven i alhora realitzar documentació suficient com per presentar-li al director. Amb lo qual, via correu electrònic s'anava informant de quin era l'estat, i aproximadament, cada dues o tres setmanes es produïa la reunió on s'entregaven les parts pactades. Si es donava el cas, el director retornava la part de documentació corregida que s'havia entregat en la reunió o per correu prèviament.

1.5. ESTRUCTURA DEL DOCUMENT

Arribats aquest punt, en la introducció s'ha pogut veure diferents aspectes sobre a tot el que a Android envolta, característiques principals del sistema, algunes de les decisions que s'han pres al llarg del projecte i finalment quins han estat els objectius i la metodologia de treball que s'ha portat a terme.

En els següents capítols doncs, s'explicarà en primer lloc quin ha estat i com s'ha portat a terme el procés d'anàlisis del projecte (veure Capítol 2). En segon lloc, com s'ha realitzat i quin ha estat el resultat de l'especificació i disseny d'aquest (veure Capítol 3). En tercer lloc, quins són els principals aspectes a destacar sobre la implementació del projecte (veure Capítol 4). En quart lloc, es detalla com s'ha realitzat el testeig de l'aplicació (veure Capítol 5). Per últim, quines són les conclusions que s'han obtingut en la realització del projecte (veure Capítol 7).

Per tal de facilitar la lectura i comprensió d'aquest document en els Annex I i II s'hi ha afegit diferents aspectes que en algun moment poden ser feixucs de llegir. En l'Annex I, es mostren les diferents taules que es van realitzar durant el procés d'especificació. En l'Annex II, els resultats de les enquestes realitzades als estudiants que van realitzar l'etapa de testeig.

El projecte *Racó Mobile* ha estat posat a disposició del usuaris mitjançant *l'Android Market.* En l'Annex III, s'han detallat els passos per realitzar la publicació.

Capítol 2

PROCÉS D'ANÀLISI

Per a qualsevol tipus de projecte aquest primer procés és molt important, ja que és aquí on s'estableix l'abast inicial del projecte. L'abast, permet establir què ha d'incloure el projecte. Ha de quedar clar que l'abast és tot allò que es creu que necessita el projecte per poder ser viable i tenir èxit. Servirà per determinar el perímetre que delimitarà el projecte. Amb la seva definició es poden extreure quins requeriments hi ha involucrats en el projecte i quins no.

Amb la definició de l'abast es pot obtenir un "contracte" amb el client, el qual indicarà, com ja hem dit, l'extensió i complexitat del projecte. No obstant, sempre s'ha d'estar disposat i obert a valorar possibles millores o noves idees que puguin augmentar la qualitat del treball final. D'aquesta última afirmació, es dedueix que aquest apartat és cíclic i segurament variable (dins un límits establerts prèviament), i que estarà vigent fins a la fase de disseny del projecte.

En la *Figura 11* es presenta un esquema de com, idealment, s'hauria de realitzar un anàlisi d'un projecte. Com a resultat final es podrà tenir la seguretat de que els requeriments queden ben definits.

En l'elaboració del projecte, s'ha aplicat la metodologia descrita en la *Figura 11* tan estrictament com s'ha pogut. No obstant, s'han introduint certes variants que es detallen a continuació:

- Degut al contacte diari amb el personal del LCFIB (Veure Secció 2.1.2), s'anaven realitzant 1 o 2 reunions per setmana, a nivell informal, on es comentaven petits detalls (com quines funcionalitats serien importants i quines es podrien obviar).
- Es realitzaven reunions mensuals, 1 o 2 segons conveniència, per tal de concretar els principals requisits, i alhora es debatien els anteriors. També es presentava l'evolució del projecte.
- El desenvolupador i analitzador en aquest cas era el projectista, qui parlant amb el director de projecte acordaven les decisions finals.

- Després d'uns 2-3 mesos aproximadament, es va acotar el projecte i es va continuar prosperant a nivell de documentació i implementació, com ja s'havia començat paral·lelament.

Iteració	Accions a realitzar	Durada
Previ a iteració 1 (1)	Primer contacte amb el client, per començar a detallar els requisits. L'equip propi de cadascú hi hauria de ser present. • El matí del primer dia, l'objectiu és acordar els principals requeriments (casos d'ús) i els requeriments no funcionals. • Amb el propi equip, decidir un 10% dels casos, per tal de realitzar el disseny i veure l'arquitectura que comportarà. • El 1.5 dies que resten, s'ha de treballar amb els requeriments funcionals i no funcionals. Després d'aquest pas, tindrem un 10% analitzat amb certa profunditat, i la resta a alt nivell.	2 dies
Previ a iteració 1 (2)	Analitzar, dissenyar i implementar els casos d'ús que s'han escollit prèviament, per veure	4 setmanes
Iteració 1	• Els primers dos dies s'ha de realitzar el model i dissenyar el model conceptual (UML) que en un primer instant s'entén que el projecte hauria de ser. • Començar a programar, integrar i testejar de forma continua el projecte que es va creant. Entenen que el disseny UML, en un principi, només és una idea per començar i que anirà variant. • Una setmana abans de la segona reunió, reunir a l'equip i analitzar si el projecte és viable; en cas que no ho sigui refer l'abast, i les tasques secundàries passen a formar de la llista de pendents. • Dos dies abans de la segona reunió testejar el codi i integrar-lo . • Realitzar les demostració al client, per demostrar els progressos.	3-4 setmanes
Previ iteració 2	Realitzar una segona reunió per tal de reajustar els requisits. Tornar a seleccionar el 10% dels requisits que es considerin més rellevant i dedicar-hi dos dies (1). Un cop realitzat, potser un 25% dels casos d'ús no funcionals han de ser escrits. No cal que estiguin perfecte.	2 dies
Previ iteració 2	Realitzar els mateixos passos que en "Previ iteració 1 (2)	4 setmanes
Iteració 2	Realitzar els mateixos passos que en "Iteració 1"	3-4 setmanes
...	Iteració del tres primers punts: Previ iteració (1)(2) i iteració 1	8 setmanes i 4 dies
Iteració 4	El 80% o 90% dels requeriments haurien d'estar redactats, i un 10% del projecte implementat. Arribats en aquest punt, tindrem un 20% del projecte cobert.	
Fase final	En aquest punt ja tenim els requisits definits i estabilitzats cosa que fa que ja no calgui de dedicar-hi més temps.	

Figura 11. Detall de l'evolució de l'anàlisi de requisits. Cas ideal. [2]

Per tal de garantir l'èxit del projecte s'ha cregut convenient afegir a la metodologia, una tècnica destinada a l'usuari final que usarà l'aplicació. Aquesta consisteix en utilitzar "l'estratègia de participació activa per part dels stakeholders[10]", així doncs, assegurem el màxim de satisfacció per part dels actors finals. Aquesta tècnica a diferència d'altres com podrien ser, partir d'un software existent, descobrir-ho mitjançant experimentació, etc. ens permet tenir un nivell d'incertesa menor. La interacció amb l'usuari final és directe i degut a l'elevat nombre d'actors finals s'ha aplicat la metodologia mitjançant enquestes.

Els individus que han format part de l'enquesta han estat una classe de tercer curs de la FIB. En total s'han realitzat trenta enquestes. Les preguntes que s'han formulat han estat basades amb el servei que s'espera que una aplicació com el *Racó Mobile* pugui oferir. En les *figures 12* i *13* es presenten els resultats.

Com es pot veure, a la *Figura 12* s'ha realitzat la divisió percentual sobre la disposició de telèfons intel·ligents. Es pot observar que un 90% d'ells disposen d'un *Smpartphone*. Aquest fet, demostra l'avenç tecnològic que s'han experimentat els últims anys. D'aquest 90% es pot observar com un 50% són Android o Iphone, siguent el primer el telèfon més escollit. Analitzant les respostes d'aquesta pregunta es pot veure com realitzar l'aplicació per a sistemes Android és una decisió, a priori, encertada.

Si s'observa la *Figura 13* es distingeixen dos enfocaments en les preguntes. El primer grup s'ha destinat a preguntes sobre funcionalitats que es podrien oferir. El segon grup, ha estat destinat a preguntes sobre rendiment, aspecte, usabilitat, etc.

Pel què fa al primer grup de preguntes es pot apreciar que les funcionalitats que es podrien considerar bàsiques per tenir èxit seria incorporar:

- L'horari Setmanal.
- El calendari d'exàmens finals (actualment l'agenda del nou Racó ja ho ofereix).
- Els avisos de les assignatures.
- Les notes i els exàmens (estarien incorporats als avisos).

En un segon pla, menys destacades però també importants:

- El correu de la FIB.
- L'ocupació de les aules.
- Descarrega del currículum.

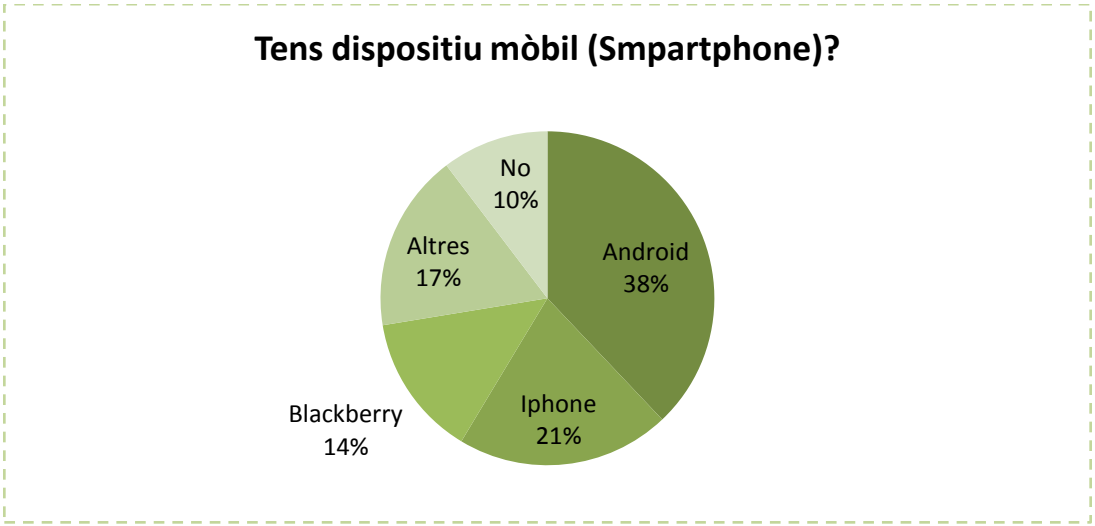

Figura 12. Resposta a la pregunta 1 de l'enquesta realitzada als estudiants

[10] Usuaris del sistema global i que es veuen afectats pel problema

En quan al segon tipus de pregunta es pot veure que les qualitats que més s'han valorat d'una aplicació mòbil són:

- Usabilitat (fàcil d'utilitzar, intuïtiva, còmode, etc).
- Velocitat de transmissió de les dades.
- Funcionalitats que s'ofereix.

Un cop s'ha analitzat l'enquesta i obtinguts els principals objectius que hauria d'assolir l'aplicació es pot passar al pas següent. En el següent apartat, es presenta la idea de requisit d'un projecte i quins hauria de complir el *Racó Mobile*.

Enquesta realitzada als estudiants de la FIB		Respostes (1 poc satisfet – 1 molt satisfet)					Número de respostes
		1	**2**	**3**	**4**	**5**	
Preguntes	2.- Creus que et seria útil disposar d'una aplicació de la Facultat pel mòbil?	-	1	2	12	15	30
	3.- T'agradaria poder consultar el teu horari setmanal?	-	1	4	7	18	30
	4.- T'agradaria consultar l'horari d'exàmens finals?	-	1	3	7	19	30
	5.- T'agradaria poder consultar el calendari lectiu?	1	-	4	8	17	30
	6.- T'agradaria consultar l'ocupació de les aules?	-	2	6	9	13	30
	7.- T'agradaria consultar els avisos de les assignatures que estàs cursant?	-	-	2	7	21	30
	8.- T'agradaria poder consultar la Guia docent de les assignatures que estàs cursant?	-	4	10	8	8	30
	9.- Trobaries interessant que es pogués accedir als vídeos del mediaFIB, youtube i assignatures?	1	7	11	7	4	30
	10.- Creus que seria interessant que hi hagués un mapa detallat del campus?	-	1	10	7	12	30
	11.- T'agradaria poder consultar les notes dels exàmens?	-	-	1	6	23	30
	12.- Voldries poder descarregar el teu currículum?	1	-	9	13	7	30
	13.- Creus que et seria útil poder accedir al correu de la FIB?	-	1	7	11	11	30
	14.- T'agradaria poder saber en quin ordinador estan fent pràctiques els teus companys?	2	6	11	5	6	30
	15.- T'agradaria que es donés accés al fòrum de la FIB?	-	6	12	8	4	30
	16.- Dins d'una aplicació mòbil, com valores la qualitat gràfica i visual?	-	1	8	12	9	30
	17.- Dins d'una aplicació mòbil, , com valores la velocitat de funcionament?	-	1	1	9	19	30
	18.- Dins d'una aplicació mòbil, com valores la utilitats de les funcionalitats?	-	-	2	13	15	30
	19.- Dins d'una aplicació mòbil, com valores la usabilitat?	-	-	1	12	17	30
	20.- Estàs d'acord en que un aplicació mòbil pot reduir la quantitat de continguts respecte a una versió no mòbil de la mateixa aplicació per tal de millorar-ne la facilitat d'ús?	-	1	8	11	10	30

Figura 13. Respostes de l'enquesta realitzada als estudiants

2.1. REQUISITS DEL PROJECTE

Un requisit es defineix com les capacitats i condicions que un sistema –i més aviat un projecte– han de complir. Aquesta definició permet extreure una sèrie de requisits o qualitats, que la gran majoria de projectes haurien de complir implícitament.

A continuació s'explica en més detall cadascun dels factors de qualitat [1][4] que hi ha descrits a la *Figura 14*. Remarcar que tots ells formen part dels requisits no funcionals:

Eficiència: Aquest factor és un dels més importants. Determina la durada i el cost del projecte. Si les previsions de les diferents etapes han estat correctament planificades, és a dir, es tenen en compte els diferents problemes o desviacions que puguin sorgir, el projecte tindrà un èxit assegurat. Altrament, una mala planificació pot fer que el projecte es demori amb les dates pactades. Aquest fet, pot generar desconfiança o descontentament per part del client.

Així doncs, s'han de concretar factors com, el període de reunions i seguiment, què s'ha de mostrar en cada un d'ells i qui ha d'assistir-hi.

En cada una de les parts del projecte s'ha de gestionar molt acuradament qui participarà en el projecte i quin serà el seu rol. Una mala gestió d'aquesta part podria implicar un increment en el cost del projecte o un endarreriment de l'entrega.

Flexibilitat: En la planificació del projecte s'ha de valorar la possibilitat de que les diferents tecnologies o estratègies que en un principi s'havien pensat siguin modificades. Aquest fet, és probable que passi ja que, els llenguatges de programació i servidors es veuen afectats per diferents actualitzacions, per exemple, errors de seguretat que es troben. En Android, des de 2008 fins al 2010 s'ha passat per, fins a 7 versions diferents excloent l'aparició de *Honeycomb* i *Ice Cream Sandwich*.

Un altra fet important a destacar en aquest punt, és la possibilitat d'un error en l'estudi previ que afectaria al canvi total de tecnologies i, en el pitjor dels casos, començar el projecte de nou.

Integritat: La privacitat i la seguretat en qualsevol projecte també és molt important. S'ha de garantir que totes les dades enviades i guardades al telèfon són segures i d'accés restringit. Per tal de realitzar aquesta acció, existeixen mètodes d'emmagatzement de dades que permeten l'encriptació d'aquestes. Les dades que s'envien per Internet (entre dispositiu i servidor) també es disposa de sistemes d'encriptació.

Un "forat de seguretat" en l'aplicació donaria pas a una desconfiança general de l'aplicació fet que podria provocar la seva desinstal·lació i conseqüentment el fracàs.

Mantenibilitat: Un projecte ha de tenir el seu pla de contingència per tal de poder reportar els errors de la manera més ràpida i efectiva possible. Un pla de contingència inclou, diferents apartats que serien: què hem de fer?, a qui hem d'avisar?, com l'hem d'avisar?, amb quina informació?. Aquest fet, ens ajuda a reparar els errors ràpidament i pràcticament de forma transparent a l'usuari, ja que amb un correu on s'informi de la fallada del sistema seria suficient. Una sèrie d'errors consecutius (valorant la seva gravetat dins l'aplicació), podria propiciar l'aparició d'una nova versió de l'aplicació.

Requisit	Descripció
Eficiència	Temps i espai
Flexibilitat	Adaptar-se a millores
Integritat	Accés restringit
Mantenibilitat	Fàcil de solucionar els errors
Fiabilitat	Tolerància a fallades
Actualitat	Tècniques actualitzades
Reusabilitat	Components es puguin reutilitzar
Testabilitat	Fàcil de prova el funcionament
Usabilitat	Fàcil d'aprendre usar

Figura 14. Factors de qualitat que ha tenir un software [4]

Fiabilitat: El sistema ha de preveure contingències que poden afectar a la prestació estable i permanent del sistema. El sistema haurà de tenir en compte i minimitzar sobrecàrregues del servidor, ja sigui processos, transaccions etc. En quan a l'aplicació, s'haurà de tenir en compte les múltiples peticions que l'usuari pugui realitzar. Les actualitzacions i la navegació per pantalles que requereixen d'actualització automàtica o d'accés a la base de dades.

Actualitat: Realitzar el projecte amb les versions més actuals i fiables que estan el mercat. S'ha de valorar la possibilitat de que pot haver-hi tecnologies on l'última versió encara estigui en fase de proves o que faci poc que ha sortit al mercat. En aquest cas, seria bo començar a treballar amb la versió anterior, tenint en compte que depenen de la durada del projecte s'hagi de realitzar un migració cap a la versió actual.

En l'aplicació s'ha decidit treballar amb la versió 2.2 ja que era la versió estandarditzada (com s'ha pogut veure anteriorment) i la que la majoria de dispositius actualment portava instal·lada quan es va començar el projecte. En quan al llenguatge de programació Java s'ha usat la versió 1.6.

Reusabilitat: Aprofitar parts d'un projecte per aplicar-ho a altres. Aquest, és un factor que pot optimitzar la feina de l'empresa en aspectes de temps i cost. Aquest fet, provoca que alhora de dissenyar i implementar una aplicació, es valorin aspectes com estandarditzar classes. Encara que en un principi puguin endarrerir o semblar que porten més feina del compte a la llarga optimitzaran moltes altres situacions.

En el cas d'Android ens trobem que els fitxers *XML* que es creen per realitzar les vistes, són estàndards, fet que provoca que es puguin reaprofitar per vàries vistes. Les classes que mostren la informació de les llistes en un format específic també es poden manipular de manera que es puguin compartir.

Testabilitat: Qualsevol projecte abans de ser entregat o publicat comercialment s'ha d'haver provat i simulat en tots els casos possibles. L'objectiu és minimitzar el nombre d'errors que puguin sorgir en l'aplicació. Per tal de que sigui possible, l'aplicació ha de ser fàcil de provar i funcionar. Per realitzar un "testeig" complet, l'aplicació s'ha de distribuir entre usuaris

experts en l'àmbit i també a usuaris no experts, ja que així, es pot assegurar que a priori totes les possibles situacions siguin experimentades.

En el projecte, l'aplicació es distribuirà als membres del Laboratori de Càlcul que disposin d'un mòbil Android. Alguns han desenvolupat aplicacions per Android i d'altres són experts en altres camps. Així doncs, s'ajustarien els dos perfils mencionats anteriorment.

Usabilitat: Aquest punt, determina la futura acceptació del projecte. Si es fa un projecte intuïtiu, simple i que cada acció que hi ha en ell sigui l'esperada per l'usuari, aquest tindrà èxit. Factors com la velocitat en la resposta de dades també afecten a la usabilitat, per tant, serà important minimitzar el temps de resposta entre servidor – dispositiu i temps de processament de dades.

Cal tenir clar a qui va dirigida l'aplicació. En aquest cas, va dirigida a qualsevol tipus d'usuari que no té perquè tenir coneixements en informàtica. Per tant, cada pantalla per si mateixa ha de ser auto explicativa des de els noms dels botons, les descripcions dels ítems, com els títols que en ella hi hagi, etc.

L'usuari no hauria de necessitar més d'un dia per copsar el funcionament de l'aplicació, sense necessitat d'un manual.

Arribats aquest punt s'ha de ser capaç de concretar els diferents requisits que el sistema haurà de complir. Per això, s'ha distingit entre els requisits funcionals i no funcionals del projecte.

2.1.1. REQUISITS FUNCIONALS

Són aquells que descriuen què ha de fer el software. Relacionen la interacció entre les dades que es consultaran i el procés o tractament d'aquestes fins que arriben a estar disponibles per part de l'actor client.

Degut a que l'aplicació contindrà una part pública, que afecta als estudiants i no estudiants, s'ha optat per dividir els requisits en públics i privats.

2.1.1.1. PÚBLICS – FACULTAT D'INFORMÀTICA

RF1. Notícies

Conté les funcionalitats relacionades amb les notícies de la facultat.

- Llistat de Notícies: Veure totes les notícies disponibles.
- Consultar informació particular: Seleccionar una notícia i veure la informació d'aquesta.
- Actualitzar: L'usuari pot actualitzar la informació de les notícies que s'estan mostrant.

RF2. Situació

Conté les funcionalitats relacionades amb la situació geogràfica de la facultat.

- Localització: Situació de la facultat en el mapa.
- Geolocalització: Determinar la posició en el mapa de l'usuari.

RF3. Assignatures del Grau

Conté les funcionalitats relacionades amb la informació de les assignatures del Grau.

- Llistat d'assignatures: Veure totes les assignatures que s'imparteixen en el Grau.
- Consultar informació particular: Seleccionar una assignatura i consultar la informació.
- Cerca intel·ligent: Selecció de l'assignatura mitjançant mnemotècnics.

2.1.1.2. PRIVADA – INTRANET DE LA FACULTAT D'INFORMÀTICA: EL RACÓ

RF4. Login

Conté les funcionalitats relacionades amb el registre de l'usuari.

- Login: Permet entrar al racó de l'estudiant.
- Logout: Permet sortir del racó de l'estudiant.

La contrasenya i el nom d'usuari es guarden automàticament.

RF5. Events conjunts

Conté les funcionalitats relacionades amb les notícies, assignatures cursant per l'estudiant en aquell quadrimestre i correu personal. Es mostraran conjuntes per facilitar i agilitzar les seves consultes.

- Llistat conjunt: Veure els tres tipus d'events ordenats cronològicament.
- Consultar informació particular: Seleccionar un event i consultar la informació.
- Actualitzar: L'usuari pot actualitzar la informació dels events que s'estan mostrant.

RF6. Preferències

Conté les funcionalitats relacionades amb les preferències que pot tenir l'usuari.

- Conjunt d'events a mostrar: Seleccionar la quantitat d'events a mostrar de cada tipus.
- Notificacions: Seleccionar si es volen rebre o no notificacions de les assignatures que l'estudiant imparteix.

RF7. Agenda

Conté les funcionalitats relacionades amb el calendari de l'estudiant.

- Llistat d'events: Veure el calendari d'exàmens, i altres events manuals que pugui afegir l'usuari. Es mostren ordenats cronològicament i amb una separació d'events futurs i passats.

RF8. Horari acadèmic personal

Conté les funcionalitats relacionades amb les classes que ha d'assistir l'estudiant.

- Mostrar horari: Es mostrarà l'horari del dia actual.
- Consultar determinada: L'usuari podrà consultar l'horari per un dia determinat.

RF9. Ocupació de les aules

Conté les funcionalitats relacionades amb l'ocupació de les aules A5, B5 i C6.

- Llistat aules: Veure totes les aules, mostrades segons l'edifici on pertanyen, la seva ocupació (quantitat d'ordinadors lliures) i si hi ha classe o no en aquell instant.
- Consultar mapa de les aules informàtiques: Seleccionar l'edifici el qual es vol veure la informació mitjançant una imatge.
- Actualitzar: L'usuari pot actualitzar la informació de la vista.

RF10. Assignatures cursant

Conté les funcionalitats relacionades amb les assignatures que té matriculades l'alumne.

- Llistat d'assignatures: Veure totes les assignatures que té matriculades amb els seus avisos recents.
- Consultar informació particular: Seleccionar l'assignatura o l'avís i consultar la informació.

RF11. Correu

Conté les funcionalitats relacionades amb el correu de l'usuari.

- Llistat del correu: Veure els correus del compte personal de la facultat.

RF12. Notificacions

Conté les funcionalitats relacionades amb les notificacions d'assignatures.

- Rebre Notificació: Quan hi hagi un nou avís d'una assignatura l'usuari rebrà una notificació de la publicació.
- Veure Notificació: L'usuari podrà veure l'avís que s'ha publicat.

RF13. About

Conté les funcionalitats relacionades amb dades dels participants en el projecte.

- Veure informació: Quan l'usuari ho desitgi podrà accedir a la informació de qui ha realitzat el projecte.
- Reportar errors: L'usuari tindrà a la seva disposició una direcció de correu per tal de comunicar els errors que detecti en el projecte.

2.1.2. REQUISITS NO FUNCIONALS

Aquests tipus de requisits defineixen les qualitats que el sistema a desenvolupar ha de tenir independentment de la seva implementació. Aquests requisits estan compostos per econòmics, estructurals, polítics i de qualitat. A continuació es detallen els requisits que un sistema com el nostre ha de complir:

RNF1. El *Racó Mobile* ha de facilitar la informació als seus clients de forma gratuïta.
RNF2. El client ha de poder aprendre a usar l'aplicació en un dia.
RNF3. La interfície de l'aplicació ha de ser intuïtiva i fàcil de manejar.
RNF4. El disseny de l'aplicació ha d'estar relacionat amb el de la facultat.
RNF5. Les imatges i icones usades en l'aplicació han de ser de lliure comerç.

Capítol 3

ESPECIFICACIÓ I DISSENY

Aquesta etapa fa referència, en el procés ideal, a la interpretació dels casos d'ús i a la definició dels actors del sistema. Els casos d'ús es defineixen com a una seqüència de fets que ajuden a comprendre el funcionament i requeriments d'aquell punt del sistema en concret. Així doncs, realitzar aquest punt ajuda a descobrir qui usarà el sistema, quins són els objectius de l'usuari i les seves perspectives. En certa manera, és aquí on es "signe" el contracte de, què farà el sistema.

Paral·lelament, poden ser usats per entendre millor els requisits funcionals, que en aquest cas ja s'ha fet prèviament, o bé per especificar-los. Aquest últim punt és el què es detalla a continuació.

Amb la seva aplicació el que es pretén és descriure els requisits funcionals de manera que siguin comprensibles per tots els *stakeholders*. Alhora delimitar la frontera amb el sistema. Comencem doncs, analitzant quins són els actors principals del sistema.

A la resta d'aquest capítol s'explicarà en primer lloc a quins usuaris va dirigit el projecte (veure Secció 3.1). En segon lloc, els diferents casos d'ús que s'han especificat (veure Secció 3.2) . En tercer lloc, es presenta un mapa de navegació de l'aplicació (veure Secció 3.3). Per últim es presenten les tres iteracions que s'han realitzat durant el desenvolupament (veure Seccions 3.3, 3.4, 3.5 respectivament).

3.1. ACTORS

Es defineixen com una entitat externa al sistema que participa en algun escenari, és a dir, en un o més casos d'ús. Un actor pot ser una persona (rols), organitzacions, hardware, altres sistemes software, etc. En l'elaboració del projecte es poden distingir els següents actors:

- Els **estudiants**. Actualment tenen un perfil d'usuari a la facultat i conformen el paper d'actor primari [11] i consumidor de l'aplicació. Aquests seran els que

[11] És l'actor qui inicia la interacció amb el sistema.

principalment interactuaran amb el sistema mitjançant el mòbil. En els casos d'ús estan mencionats com a "estudiants".

- Un altre tipus d'actors, que també es consideren primaris, són els que no són ni estudiants ni pertanyen al sistema, els anomenarem **externs**. Aquests, són els que consulten només la part pública de l'aplicació. Per exemple, futurs estudiants o bé persones externes a la facultat interessades en algun tema, com per exemple, les notícies que es publiquen. En el següent apartat es veurà detalladament quina és la informació que l'aplicació els permetrà consultar. En els casos d'ús estan mencionats com a "externs".
- El **client**. Aquest s'identifica per una necessitat, en aquest cas el projecte a desenvolupar. Aquest seria, el Laboratori de Càlcul de la Facultat d'Informàtica de Barcelona, també conegut com a LCFIB, que és, qui posteriorment gestionarà el projecte i qui subministra la informació necessària.
  En els casos d'ús, només es menciona el "servidor de dades". El rol com a usuari pot està identificat com a un dels altres 2 tipus d'actors.

3.2. DEFINICIÓ DELS CASOS D'ÚS

Com s'ha comentat anteriorment, els casos d'ús s'utilitzen per veure en detall cada funcionalitat del sistema, definir les diferents accions que es poden produir i actuar de manera proactiva. Per la seva definició es podria haver escollit un dels tres possibles mètodes proposats al llibre "*Applying UML and Design Patterns*"[3].

1. Mètode *brief*. Consisteix en realitzar un sumari d'un paràgraf, normalment de l'escenari principal.
2. Mètode *casual*. Consisteix en diversos paràgrafs, que cobreixen diferents escenaris. Aquest serien l'escenari principal i els alternatius que hi puguin haver.
3. Mètode *full dressed*. Aquest és el més complet de tots. Consisteix en descriure els passos i variants descrits amb el màxim detall, a part també es poden definir vàries seccions extres, per tal de definir amb molta més precisió el cas d'ús.

En la realització del projecte, donada la seva complexitat s'ha preferit escollir l'últim model, ja que així es pot oferir un anàlisi molt detallat i estructurat. Aquest, es pot consultar en l'Annex II. En la *Figura 15* es pot observar quina és la base que s'ha seguit per realitzar el mètode *full dressed*. No obstant, per tal de veure a nivell global què fa cada un d'ells i, de manera excepcional, a continuació es realitza una breu descripció seguin el model *brief*.

PROCÉS DE CONSULTA RÀPIDA D'EVENTS

L'usuari en el moment que obri l'aplicació podrà veure una llista d'events ordenats cronològicament. Els possibles events són notícies, correus i avisos de les assignatures. Si l'usuari ha introduït el perfil del Racó podrà tenir accés als dos últims, altrament, només se li presentaran les notícies. Per cada event podrà realitzar una consulta detallada de cada un dels elements mitjançant la seva selecció en la llista.

LOCALITZACIÓ DE LA FACULTAT D'INFORMÀTICA

Accedint a l'apartat FIB, l'usuari podrà visualitzar, mitjançant Google Maps, els principals edificis de la facultat. Aquests són la Biblioteca BRGF, la sala d'actes, l'Omega,

l'edifici Vèrtex i la Plaça de la FIB. També podrà consultar quina és la seva posició actual en el mapa respecte els edificis.

CONSULTAR LES ASSIGNATURES DE LA FACULTAT

Accedint a l'apartat FIB, l'usuari podrà consultar les assignatures del Grau en Informàtica que s'estan impartint a la facultat. També podrà consultar la informació de cada una d'elles. En la consulta particular es mostrarà quins professors la imparteixen, quins són els objectius de l'assignatura i el nombre de crèdits.

CONSULTA DE LES NOTÍCIES DE LA FACULTAT

A l'igual que en la vista inicial l'usuari podrà consultar les notícies de la facultat. No obstant, en aquesta vista ubicada en l'apartat de la FIB, només es mostraran les notícies publicades periòdicament. De nou, podrà consultar la informació particular de cada una.

LOGIN I LOGOUT

L'usuari mitjançant la vista principal o l'apartat Racó, podrà introduir el seu usuari del Racó. Un cop introduïdes les dades podrà consultar la informació del Racó com: assignatures matriculades, correu, agenda, calendari, horari i l'ocupació de les aules.

Dins l'apartat de Racó l'usuari podrà realitzar el Logout, el qual eliminarà les seves dades en l'aplicació.

PREFERÈNCIES

En la vista principal l'usuari podrà accedir al menú, mitjançant el botó menú dels telèfons Android. Aquí podrà activar les notificacions i configurar el nombre d'events (avisos, correus i notícies) que vol que surtin en la llista inicial de l'aplicació.

CONSULTA DE L'AGENDA PERSONAL

L'usuari accedint a l'apartat del Racó podrà consultar els events que hagi introduït a la seva agenda personal del Racó. Aquests estaran ordenats cronològicament de futur a passat amb una senyalització de quin és el dia actual.

OCUPACIÓ DE LES AULES

L'usuari accedint a l'apartat del Racó podrà consultar quin és l'estat de les aules informàtiques A5, B5 i C6. En la vista se l'informarà del nom de l'aula, si hi ha classe o no i quants ordinadors disponibles hi ha. Alhora, accedint a cada element de la llista podrà consultar visualment l'estat de l'aula.

CONSULTA DE LES ASSIGNATURES DE L'ALUMNE

L'usuari accedint a l'apartat del Racó podrà consultar els avisos de les assignatures de les que està matriculat i la informació de l'assignatura. Es mostrarà una llista on es veurà cada element. Podrà realitzar la consulta particular dels elements.

CONSULTA DEL CORREU

L'usuari accedint a l'apartat del Racó podrà consultar els correus que hagi rebut. La informació que es mostrarà serà: el subjecte del correu i qui l'ha enviat.

NOTIFICACIONS

Si l'usuari ha introduït el seu usuari del Racó, podrà activar les notificacions. Aquestes li apareixeran a la barra que està situada a la part superior del telèfon. S'informarà a l'usuari de les publicacions i modificacions dels avisos de les assignatures.

ABOUT

En la vista principal l'usuari podrà accedir al menú, mitjançant el botó menú dels telèfons Android. Un cop dins podrà consultar la informació sobre qui ha fet l'aplicació, per qui ha estat dirigit i codirigit el projecte i, finalment, qui ha realitzat el disseny.

Per acabar aquest apartat a la *Figura 16* es representa de manera gràfica com es van distribuir els requisits en l'aplicació.

En el següent apartat, un cop definits els casos d'ús, es troba representat gràficament la navegació prèvia entre vistes que es va realitzar abans de la implementació. I de manera esquemàtica com representar els elements que la conformen.

Cas d'ús X: Nom
Àmbit
A quin sistema esdevindrà el cas d'ús i en quina part es troba.
Nivell
Els casos d'ús es poden dividir en 2 nivells: • Principal: Escenari on es desenvoluparà el cas d'ús. • Secundari: Escenaris que complementen el cas d'ús principal, o que estaran presents en algun dels seus passos.
Actor/s primari/s
Actor principal que participarà en el cas d'ús.
Stakeholders i altres
Ens determina què haurà de fer el sistema. Es determina la funció i què espera cada stakeholder que el sistema faci. Es determina també la satisfacció del stakeholder.
Precondició
Es descriurà l'estat que sempre s'ha de complir abans que l'escenari principal comenci.
Garanties d'èxit
Es descriurà quin és l'estat final del cas d'ús, el qual, si tot el procés és correcte s'hauria d'assolir.
Escenari principal
Es descriurà l'escenari ideal del cas d'ús. És a dir, quina seria la seva evolució si no es produís cap anomalia.
Extensions
Escenaris alternatius en que el cas d'ús podria esdevenir. Comprèn casos d'ús secundaris, o bé d'errors que es puguin produir. Són branques derivades de l'escenari principal.
Requisits especials
Es descriuran els requisits no funcionals, atributs qualitatius que ha de complir el cas d'ús. Es poden incloure els requisits que hi ha a la *Figura 14*.
Requisit/s assolit
Es descriurà quin o quins requisits descrits anteriorment queden assolits, mitjançant la nomenclatura: • RF_X - Requisit funcional on X és el número del requisit.
Freqüència
Probabilitat de que es produeixi el cas d'ús.

Figura 15. Plantilla base de la descripció del casos d'ús

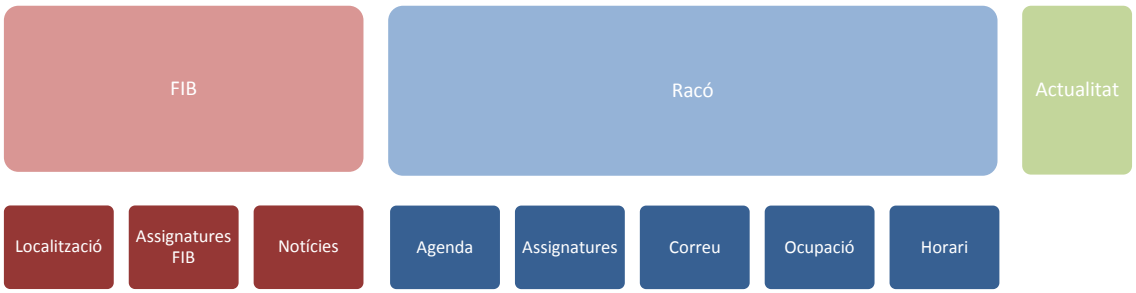

Figura 16. L'estructuració de l'aplicació a nivell visual

3.3. MAPA DE NAVEGACIÓ

En la *Figura 17* es pot veure les diferents pantalles que es van dissenyar abans del inici de la implementació de l'aplicació. Com es veurà més endavant, en l'aplicació final, la navegació entre pantalles no ha canviat. Això indica que la idea inicial que es tenia de com havia de ser l'aplicació era bona. A part, la seva elaboració ha contribuït positivament en l'avenç del projecte. Les diferents persones que participaven en ell (projectista, director i codirector) han estat situats, en tot moment, en el punt desenvolupament que s'estava. També va contribuir a preveure i aplicar millores alhora de navegar per les diferents pantalles que hi ha. Un exemple podria ser: la decisió de navegar mitjançant menús. Com es pot veure en el resultat final no es va portar a terme ja que es va considerar que eren millors les pestanyes que actualment es poden apreciar.

Destacar que els elements que es mostren en el diagrama, com poden ser, mapes, llistes, entrades de text, botons, etc. també van servir alhora de planificar la durada del projecte. Depenen del nombre d'elements i complexitat de cada un d'ells, i depenen del nombre de vistes en què podien aparèixer, el cost temporal del projecte podia variar.

En el següent apartat, es mostren les diferents iteracions que hi ha hagut en el desenvolupament de l'aplicació final. En la iteració 0 es mostra perquè s'ha escollit aquest disseny de navegació entre pantalles i quin aspectes bàsics l'aplicació havia de contemplar. En la primera es mostra el plantejament inicial de les classes que serien necessàries per a la implementació. I en la última els diagrames de classes finals de l'aplicació.

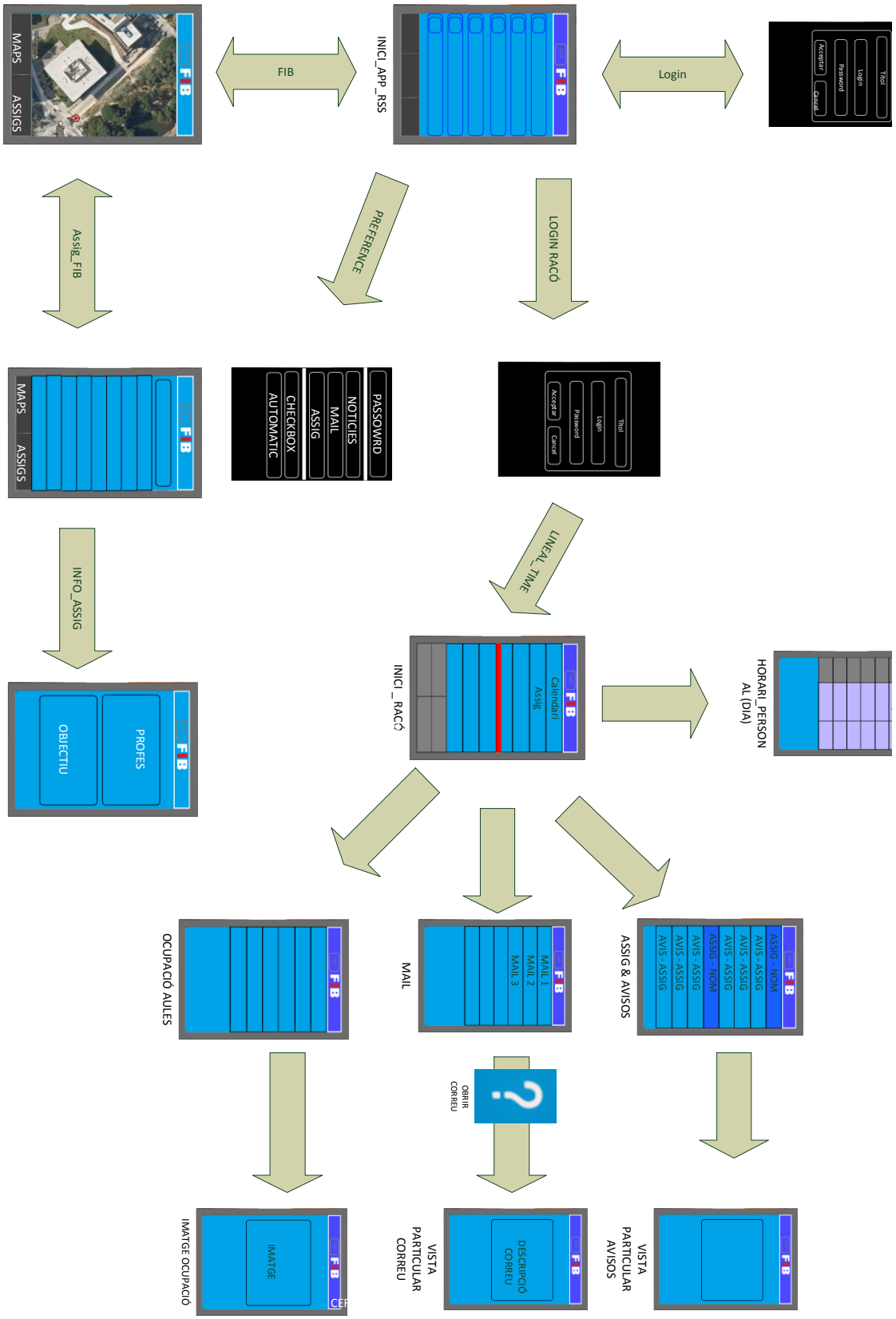

Figura 17. Mapa navegacional de l'aplicació.

3.3. ITERACIÓ 0

Es va dur a terme des del febrer de 2011 fins a principis de juny del mateix any.

Es van destinar dues setmanes a l'estudi i aprenentatge de programació d'aplicacions en Android. L'objectiu era escollir la millor manera de dissenyar l'aplicació per tal de que fos el màxim de simple i comprensible pels usuaris. Un cop es van assolir els coneixements mínims per poder començar una aplicació d'aquesta complexitat, es va acordar amb el director de projecte la realització de vàries versions amb diferents navegabilitats i formats per tal de poder escollir la més adient. Els principals aspectes que es va considerar que s'havien de treballar són els que es troben descrits a la *Figura 18*.

Fins el mes de maig es van realitzar 3 tipus de versions de l'aplicació. D'aquesta manera, es podia veure i valorar les diferents opcions que cadascuna presentava. En les *Figures 19, 22 i 24* es representa la valoració de les versions i en les *Figures 20, 21, 23, 25 i 26* les imatges més destacades de cada una.

Característica	Descripció
Navegabilitat	Les navegacions han de ser fàcils i ràpides entre les diferents funcionalitats.
Intuïtiva	S'ha de precisar clarament en quina funcionalitat estem accedint.
Persistència	No sempre es té accés a Internet, així que, una opció és guardar les dades al mòbil.
Connectivitat i parseig	L'usuari ha de poder obtenir les dades disponibles en pantalla per ser consultades el més aviat possible.
Completa	Ha de tenir totes les funcionalitats pactades amb el client.

Figura 18. Aspectes destacats a l'hora de dissenyar l'aplicació

Versió 1 – Aplicació	
Característica	**Descripció**
Navegabilitat	Es realitza mitjançant el botó menú del telèfon. Es mostren les opcions possibles segons la jerarquia descrita en el punt anterior.
Intuïtiva	Poc intuïtiva ja que hi ha poques aplicacions Android que la navegabilitat es realitzi mitjançant el menú.
Persistència	Mitjançant base de dades i *SharedPreferences*[12].
Connectivitat i parseig	Es bloqueja la pantalla amb un quadre de diàleg fins que s'ha obtingut i parsejat les dades corresponents. Aquesta acció es produeix per cada pantalla que necessita connexió al servidor.
Completa	Té totes les funcionalitats que el client ha demanat.

Figura 19. Aspectes destacats de la versió 1

[12] Tipus d'emmagatzament de dades que ofereix Android per guardar informació {clau-valor}.

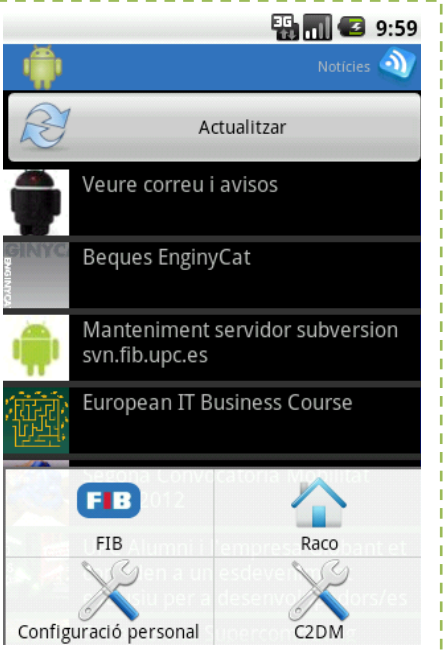

Figura 20. Navegació des del botó menú

Figura 21. Pantalla bloquejada mentre es carrega la informació

Versió 2 – Aplicació Mòbil a partir d'un Webview[13]	
Característica	**Descripció**
Navegabilitat	Es realitza mitjançant el menú on cada funcionalitat és un enllaç a una pàgina web.
Intuïtiva	Poc intuïtiva ja que hi ha poques aplicacions Android que la navegabilitat es realitzi mitjançant menú d'aquest tipus.
Persistència	No hi ha base de dades ja que les connexions es realitzen en temps real. Si no es disposa d'Internet no es podran mostrar les dades.
Connectivitat i parseig	Es bloqueja la pantalla amb un quadre de diàleg fins que s'ha obtingut i parsejat les dades corresponents. Aquesta acció es produeix per cada pantalla que necessita connexió al servidor, sempre i quan no s'hagi visitat prèviament.
Completa	Té totes les funcionalitats que el client ha demanat.

Figura 22. Aspectes destacats de la versió 2

[13] Vista que mostra una pàgina web.

Figura 23. Vista de l'aplicació integrada en un Webview.

Versió 3 – Aplicació	
Característica	**Descripció**
Navegabilitat	Es realitza mitjançant el "*Widget Tabs*[14]" que l'*SDK* d'Android proporciona.
Intuïtiva	Molt intuïtiva ja que l'usuari veu en tot moment on es troba i les opcions de les que disposa.
Persistència	Mitjançant una base de dades i *SharedPreferences*.
Connectivitat i parseig	Es bloqueja la pantalla amb un quadre de diàleg fins que s'ha obtingut i parsejat les dades corresponents. Aquesta acció es produeix per cada pantalla que necessita connexió al servidor, sempre i quan no s'hagi visitat prèviament, on el contingut a mostrar serà el guardat a la base de dades
Completa	Té totes les funcionalitats que el client ha demanat.

Figura 24. Aspectes destacats de la versió 3

[14] Contenidor per a una vista amb pestanyes. Aquest objecte conté un conjunt d'etiquetes que l'usuari fa clic per seleccionar una fitxa específica, i un objecte FrameLayout que mostra el contingut d'aquesta pàgina.

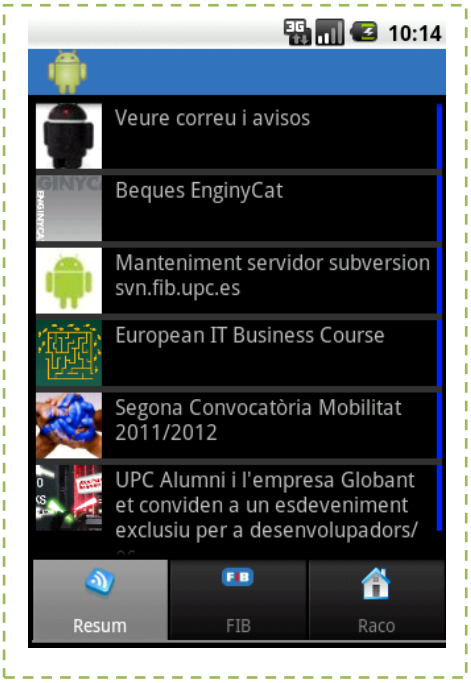

Figura 25. Navegació des de Tabs

Figura 26. Pantalla bloquejada mentre es carrega la informació

Un cop realitzades les tres versions es va procedir a analitzar quina podria ser la millor solució pel *Racó Mobile*. Les conclusions que se'n van extreure van ser:

- L'aplicació no podia mostrar les dades directament d'una pàgina web, és a dir, ser un *WebView*. S'havia de poder implementar un sistema de persistència on no fos necessària la connexió a Internet per poder navegar per l'aplicació i poder consultar les dades ja obtingudes prèviament. Per tant, la versió 2 va quedar descartada.
- La diferència més important entre les versions 1 i 3 era la navegabilitat. Així doncs, donat que la majoria d'aplicacions en Android per a realitzar la navegació i mostrar el contingut utilitzen *Tabs*[15] es va optar per la versió 3. El menú s'utilitza per a funcionalitats secundàries com podria ser: actualitzar, preferències, about, etc.
- Mentre es realitzava la càrrega de dades no es podia bloquejar totes les vistes amb un quadre de diàleg. Si hi havia dades prèviament al telèfon s'havien de poder mostrar el més aviat possible. Paral·lelament s'havia de poder realitzar la petició al servidor. Tot i que cap dels tres dissenys en aquell moment ho implementava, es va tenir en compte de cara al disseny final.

Un cop establertes les principals característiques de l'aplicació i quina de les versions era l'escollida es va començar a desenvolupar. A continuació, en la iteració 1, es presenta la primera etapa del desenvolupament de l'aplicació final.

3.4. ITERACIÓ 1

Aquesta iteració va durar tot el mes de juny. En aquest període es va decidir com seria la implementació i disseny de l'aplicació.

[15] http://developer.android.com/reference/android/widget/TabWidget.html

Es van estudiar quines serien les millors opcions per realitzar un disseny intuïtiu, accessible ràpidament a les funcionalitats i que el flux d'informació fos fluït. Les opcions que es van valorar van ser:

I. A**rquitectura tres capes [2] [4] [19].**

El patró de disseny (o estil) en capes té com a gran avantatge que facilita la canviabilitat (que vol dir: extensibilitat, portabilitat, mantenibilitat i reestructurabilitat) i la prova. Com a inconvenient, fa difícil d'obtenir l'eficiència òptima en afegir feina innecessària o redundant.
Les crides (i retorns) en un patró en capes han de complir:

- Els components (objectes/classes) s'agrupen en capes.
- Tal i com s'observa a la *Figura 27* la comunicació (relacions entre classes, missatges entre objectes) sols es produeix entre elements de la mateixa capa o de capes contigües.

La *capa de presentació* sap com presentar les dades a l'usuari, però ignora quines transformacions cal fer per donar resposta a les peticions de l'usuari.

a. Es relaciona amb els usuaris: rebent-ne esdeveniments i presentant-los respostes i resultats.
b. Es relaciona amb la capa de domini: passant-li els esdeveniments externs (crides a accions) i consultes, i rebent-ne les respostes i resultats.
c. S'ocupa de:
   - Assabentar-se de les peticions dels usuaris.
   - Ordenar l'execució d'accions.
   - Comunicar els resultats de les accions als usuaris.
   - Tractar les vistes.

La *capa de domini* sap com satisfer les peticions de l'usuari, però ignora on es guarden les dades i com es presenten a l'usuari.

a. Es relaciona amb la capa de presentació: passant-li les respostes i resultats, i rebent-ne els esdeveniments externs (crides a accions) i consultes.
b. Es relaciona amb la capa de gestió de dades: passant-li les operacions de consulta i modificacions de dades, i rebent-ne les respostes i resultats.
c. S'ocupa de:
   - Assabentar-se dels esdeveniments.
   - Controlar-ne la validesa.
   - Canviar l'estat del domini.
   - Executar les accions encomanades.
   - Assabentar-se de les consultes.
   - Obtenir-ne el resultat.
   - Comunicar la resposta.

La *capa de gestió de dades* sap on i com estan emmagatzemades les dades, però ignora com tractar-les.

a. Es relaciona amb la capa de domini: passant-li les respostes i resultats, i rebent-ne les operacions de consulta i modificació de dades.
b. Es relaciona amb el sistema de gestió de bases de dades o fitxers: passant-li les operacions de consulta i modificacions de dades en el format i llenguatge adequats. Rep les respostes i resultats.
c. S'ocupa de:
   - Permetre-li al domini d'ignorar on són les dades.
   - Permetre que determinats objectes del domini siguin persistents.

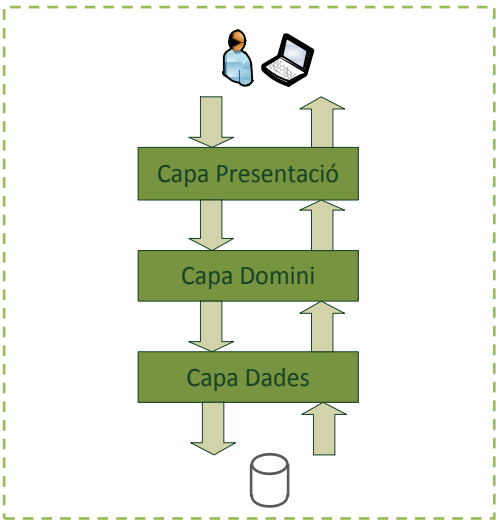

Figura 27. L'arquitectura tres capes

Amb aquesta estructura s'aconsegueix:

- Que un canvi en la representació persistent de les dades (per exemple, un canvi en el sistema gestor de bases de dades o fitxers), normalment, només afecti a la capa de gestió de dades (diem normalment perquè l'estructura adequada per alguna solució amb determinats gestors de bases de dades pot afectar la capa de domini).
- Que un canvi en la interfície del programa (per exemple, un canvi en el sistema de finestres o en els perifèrics usats per a la comunicació amb l'usuari) afecti només la capa de presentació.
- Que la capa de domini sigui independent dels canvis de plataforma, sistema operatiu, etc.

Finalment, després de l'arquitectura en tres cap i el Model-Vista-Controlador (veure Secció 1.3.1) es va decidir que el segon era ideal per desenvolupar l'aplicació perquè ens aportava el següent:

a. La base de dades estaria unificada en un únic arxiu (veure Secció 4.6.2).
b. El projecte un cop entregat el gestionaria l'empresa client (LCFIB) de manera que es facilita el fet de que en cas de voler realitzar canvis, aquests siguin mínims. Normalment, aquests estaran en les vistes, modificant la informació tal i com es mostra o mostrant-ne de noves. El model és complet en relació a la informació que la facultat ara mateix pot aportar. D'aquest fet, afegir que l'aplicació guanya en flexibilitat (adaptació a millores) i reusablitat (poder reutilitzar el model i les vistes).
c. Relacionat amb el punt anterior destacar que la lògica d'interfície d'usuari canvia amb més freqüència que les dades i la lògica de negoci. Si es realitza un disseny que enllaci components de la interfície i del model, llavors la conseqüència serà que, quan es vulgui canviar la interfície, s'hauran de modificar els components del model.
d. En Android, com ja s'ha comentat anteriorment, les vistes són fitxers *XML* en un directori separat dels controladors. El *MVC* permet crear una relació 1 a 1 entre les vistes i els controladors.
e. L'últim punt a destacar, és que els controladors omplen les llistes a mostrar amb la informació corresponent. En Android, per tal de que cada llista es pugui mostrar amb les característiques desitjades s'han de crear els adaptadors de llistes. Aquests, permeten mostrar a l'usuari la informació tal i com es desitgi (veure Secció 4.3). Els adaptadors en el disseny final s'han definit al model.

Un altra fet que també compleix amb la relació Model-Vista-Controlador és que des del model és on es realitzen les connexions per obtenir les dades. Quan la informació és retornada i parsejada aquests s'ocupen d'actualitzar la informació a les vistes per mitjà dels controladors.

Un cop es va escollir amb quin disseny arquitectònic es programaria l'aplicació es va procedir a realitzar els diagrames del model i dels controladors per tal de tenir una aproximació de l'estructura de l'aplicació.

3.4.1. MODEL CONCEPTUAL

Quan es realitza el disseny basat en objectes es divideix l'aspecte estàtic i l'aspecte dinàmic (veure Secció 3.4.3). L'aspecte estàtic, és la representació visual i conceptual de les classes o objectes de la realitat. Il·lustra el conjunt d'objectes mitjançant un diagrama de classes, on no hi ha representades les operacions (mètodes). Proveeix d'una perspectiva conceptual. L'aspecte estàtic ha de representar:

- Objectes del domini.
- Associacions entre les classes.
- Atributs de les classes.

El model, explicat en el la Secció 1.3.1, representa la informació que contindrà el sistema. Haurà de contenir els objectes que l'aplicació requereix per mostrar, com per exemple, assignatures, avisos, correu, classes, etc. amb les corresponents associacions entre ells.

Una part important del model és que ha de poder ser exportable a qualsevol altre projecte. S'ha procurat crear un model el màxim genèric possible, el qual es pot observar a la *Figura 28*. A continuació es descriuen les classes més destacades:

- La classe *GestióConnexió*: és la que tracta totes les peticions a servidor dels controladors, retornant la informació parsejada i tractada.
- La classe *Usuari*: conté un atribut que fa referència a les dades guardades en les *SharedPreferences*, i que mitjançant aquest atribut podríem accedir a les seves preferències particulars, on hi haurà el nom d'usuari i contrasenya, entre d'altres.
- La classe *ParserAndUrl*: permet realitzar les crides asíncrones. Se li passa un vector d'adreces per connectar-nos al servidor i saber el parsejador que s'ha d'invocar un cop s'hagin rebut les dades.
- La classe *LlistesItems*: s'utilitza per poder retornar la informació al controlador que ha sol·licitat aquella informació.

El model conceptual representat a la *Figura 28* no permet, gràficament, expressar vàries restriccions que cal remarcar. Així doncs, mitjançant les restriccions textuals, es poden detallar explícitament.

1. Restriccions de clau externa:
    a. No poden existir dos *notícies* amb el mateix id.
    b. No poden existir dos *avisos* amb el mateix id.
    c. No poden existir dos *correus* amb el mateix id.
    d. No poden existir dos *aules* amb el mateix nom.
    e. Només hi haurà un *username* al sistema.
    f. No poden existir dos *events* amb el mateix títol i descripció.

2. Restriccions que no són de clau externa:
    a. Les dates de publicació dels *itemGeneric* i *events* seran iguales o prèvies a la data actual.

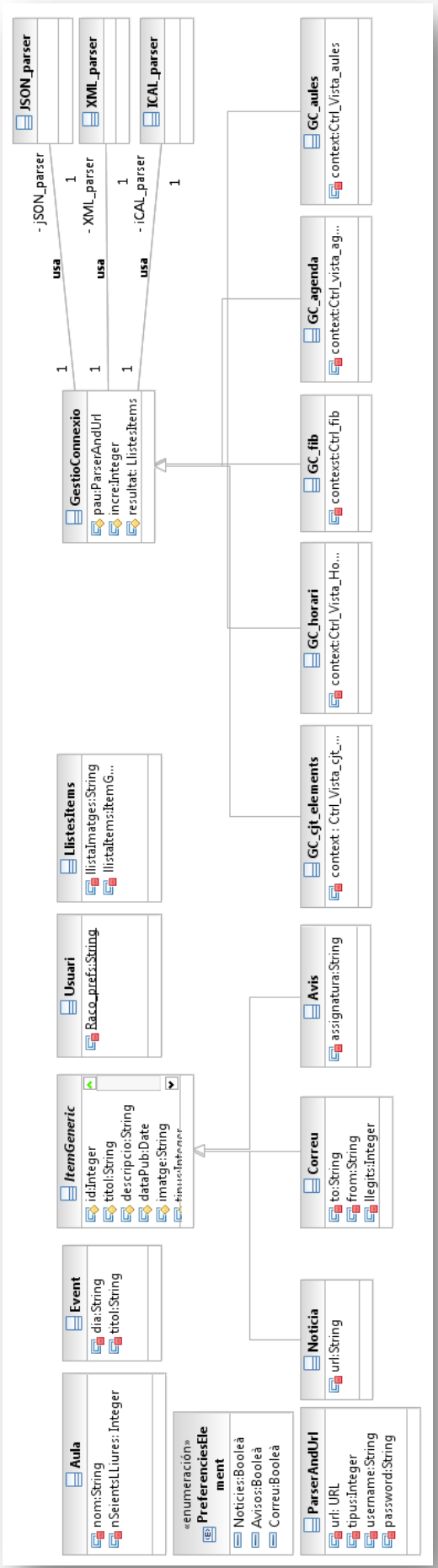

Figura 28. Definició del model de la iteració 1

ESPECIFICACIÓ AMB LLENGUATGE OCL

L'OCL [20] és un llenguatge formal, tipat i d'expressions. No té efectes laterals, és a dir, no altera els objectes del model, i per tant, no altera el sistema. S'usa per a descriure les expressions dels models *UML*. Aquestes expressions solen especificar les condicions invariants que s'han de complir en el sistema que es modela o bé, les consultes que es realitzen sobre els objectes del model.

Un diagrama *UML*, a l'igual que un diagrama de classes, no sol ser prou refinat per proporcionar tots els aspectes rellevants d'una especificació. Hi ha, entre d'altres coses, la necessitat de descriure les limitacions addicionals sobre els objectes en el model. Aquestes sovint es descriuen en llenguatge natural. La pràctica ha demostrat que aquest sempre donarà lloc a ambigüitats. Per tal d'evitar aquestes ambigüitats, es van crear els llenguatges formals. El desavantatge dels tradicionals llenguatges formals és que són molt complexos i enrevessats, en canvi, el llenguatge OCL simplifica el problema.

OCL ha estat desenvolupat per omplir aquest buit. Es tracta d'un llenguatge formal que segueix sent fàcil de llegir i escriure. S'ha desenvolupat com un llenguatge de modelatge de negoci dins de la divisió d'assegurances d'IBM.

OCL és un llenguatge d'especificació pur, per tant, en una expressió OCL es garanteix que que no hi haurà efectes secundaris, per exemple, en els objectes del model. Quan una expressió OCL s'avalua, simplement torna un valor. No es canvia res en el model. Això significa que l'estat del sistema mai canviarà a causa de l'avaluació d'una expressió OCL, tot i que una expressió OCL pot ser utilitzada per especificar un canvi d'estat (per exemple, una post-condició).

OCL no és un llenguatge de programació, per tant, no és possible escriure la lògica del programa o de control de flux. No es poden invocar processos o activar operacions no consultores en les operacions .

OCL és un llenguatge escrit, de manera que cada expressió té un tipus. Per estar ben format, una expressió ha de complir les regles de tipus de conformitat de la llengua, per exemple, no es pot comparar un enter amb una cadena. Cada classificador definit dins d'un model *UML* representa un tipus diferent. A més, OCL inclou un conjunt predefinit de tipus complementaris.

Com en un llenguatge d'especificació, totes les qüestions d'aplicació estan fora d'abast i no poden ser expressats en OCL. L'avaluació d'una expressió és instantània, això significa

que els estats dels objectes en un model no es poden canviar durant l'avaluació.

Com a conclusió d'aquest subapartat es pot dir que: s'usa OCL quan els models gràfics no són suficients per a una especificació precisa i no ambigua. Serveix per especificar invariants (restriccions i regles de derivació) del Model Conceptual. També es pot usar per especificar precondicions, postcondicions i sortides de les operacions. En les *figures 29* i *30* es pot veure la definició de les claus del diagrama "definició del Model" (*Figura 28*).

1	
a	**Context** *Noticia* **inv** *Noticia.allInstances -> forAll(n1,n2 | n1<> n2 implies n1.id<>n2.id)*
b	**Context** *Avis* **inv** *Avis.allInstances -> forAll(a1,a2 | a1<> a2 implies a1.id<>a2.id)*
c	**Context** *Correu* **inv** *Correu.allInstances -> forAll(c1,c2 | c1<> c2 implies c1.nom<>c2.nom)*
d	**Context** *Aula* **inv** *Aula.allInstances -> forAll(a1,a2 | a1<> a2 implies a1.nom<>a2.nom)*
e	**Context** Usuari::nombreUsuaris(): Integer **inv** derive: self.preferenciesUsuari.username -> size() > 1
f	**Context** *EventAgenda* **inv** *EventAgenda.allInstances -> forAll(e1,e2 | e1<> e2 implies e1.nom<>e2.nom and e1.descripcio<>e2.descripcio)*

Figura 29. Restriccions textuals de clau externa

2	
a	*--Idem s'hauria de fer per avisos i correus.* **Context** *Noticia* **inv** *Noticia.allInstances -> forAll(n1,n2 | n1<> n2 implies n1.dataInici<=dataActual)*

Figura 30. Restriccions textuals. No de clau externa

Com s'ha dit en la definició del llenguatge OCL, també es poden expressar operacions consultores que no tenen perquè representar restriccions textuals. A continuació es presenten dos exemples sobre aquesta definició.

- obtenir tots els *itemGenerics* que continguin un títol.

Context** Ctrl_Vista_cjt_elements **inv
LlistaItem= self.ItemGeneric->select(it \| it.titol --> notEmpty())

- En aquest segon exemple podem obtenir totes les aules a mostrar.

Context** Ctrl_Vista_aules **inv
LlistaItem= self.aula

`

Un cop vist el model que s'ha pensat per l'aplicació, es troba descrit a continuació la vista, que com ja s'ha comentat, és la que interactua amb l'usuari.

3.4.2. VISTA

En Android les vistes es representen mitjançant fitxers *XML*, els quals es defineix quins elements contindrà la vista. En la *Figura 31* es pot veure el diagrama de classes d'aquesta capa. Com es pot observar la quantitat de classes és elevada. Això és, perquè en un principi, per tal de poder crear cada pantalla el màxim d'específica possible, es va optar per generar un fitxer *XML* per cada vista que hi havia.

A continuació es mostra una breu descripció dels elements que, en un principi, es va pensar usar:

- *ListView:* Mostra una llista en pantalla.
- *TextView:* Camp de text no editable i que conté informació per l'usuari.
- *EditText:* És on l'usuari entra les dades per a què el sistema realitzi una acció.
- *ImageButton:* Permet mostrar un botó amb una imatge i text.
- *Button:* Mostra un botó amb text dins. (Exemple: Acceptar, Cancel·lar...)
- *TabWidget*: Permet afegir pestanyes a la vista.
- *TabHost*: Són les vistes contingudes en cada una de les pestanyes (en el *TabWidget*).
- *MapActivity*: Aquest, no es mostra en el diagrama ja que està contingut dins d'una pestanya de la classe *Vista_fib*. És la vista del mapa de Google Maps.

Els tres elements que es presenten a continuació, serveixen per definir com s'organitzen els anteriors descrits, dins la vista principal de l'aplicació.

- *LinearLayout*: Consisteix en situar un element seguit de l'altra. No permet sobreposar elements. Per exemple, si li diem que per defecte és vertical ens posarà cada element un sota l'altra.
- *RelativeLayout*: És el més flexible de tots, ja que per defecte, alinea tots els elements a la part superior esquerre, tots sobreposats. El disseny de la vista que es mostrarà a l'usuari ho ha de definir el desenvolupador explícitament. Tal i com es realitza en una pàgina web *HTML*.

Un cop descrit la capa de vista en el *MVC*, es presenten els controladors que s'utilitzaran per poder mostrar la informació i relacionar-se amb el model.

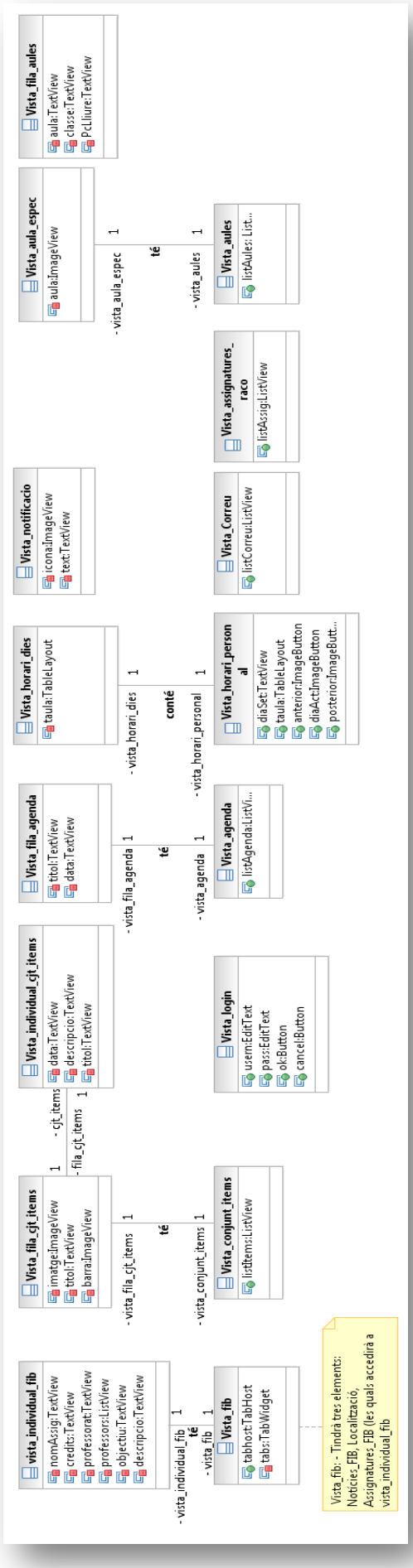

Figura 31. Definició de la vista de la iteració 1

3.4.3. CONTROLADORS

Tal i com s'ha comentat en la Secció 1.3.1, els controladors són els encarregats d'enllaçar les vistes amb el model i processar els events rebuts de les vistes com a conseqüència d'events d'usuari.

En el *Racó Mobile*, cada vista que l'usuari pot interactuar és assignada a un únic controlador. Aquest és l'encarregat de comunicar-se amb els elements que figuren en la vista i el model. Amb el model, la comunicació més comuna és mitjançant la classe *GestióConnexió* que és la que permet obtenir les dades per ser mostrades posteriorment. Les vistes li proporcionen els elements que necessita per poder mostrar la informació.

Per tal de que quedi més clar quina és la seva funcionalitat s'ha decidit presentar-los mitjançant diferents exemples. En ells es pot veure el model estàtic del cas, la descripció i finalment, s'ha cregut oportú representar-ho mitjançant el model de comportament, el qual, és descrit a continuació.

MODEL DE COMPORTAMENT

El model de comportament representa la part dinàmica del model. Mitjançant diagrames de seqüències simula quin serà el flux de la informació, quines classes participen en el cas i qui és el responsable en cada moment. Els diagrama de seqüència es defineixen com una notació que permet il·lustrar les interaccions dels actors i les operacions que s'inicialitzen amb aquesta. Es representen mitjançant una imatge que mostra per un escenari particular d'un cas d'ús, els events que els actors externs generen, el seu ordre, i la interacció interna del sistema.

Quan s'està dissenyant software, les preguntes més comunes que es presenten són: quins events es produiran al sistema? i Perquè? Realitzar aquesta etapa permet controlar aquests events (del ratolí, teclat, propis del sistema, pantalla tàctil, etc.) i executar les respostes apropiades. Bàsicament, el sistema software reacciona amb les següents tres accions: 1) events externs dels actors (persones o sistema), 2) Events programats i 3) errors o excepcions.

Un cop vista quina és la filosofia del model de comportament, s'han realitzat tres exemples diferents que mostraran en detall quin és el paper que juguen els controladors en el sistema.

CAS1 – MOSTRAR AGENDA DEL RACÓ

En aquest cas, procedim a veure el procés que segueix una de els funcionalitats que realitza l'aplicació. Aquesta està ubicada en la pestanya del Racó, on es mostra la informació que l'usuari té guardada en l'agenda. En la *Figura 32* es mostra el diagrama *UML* amb el que ens basarem per explicar el comportament del controlador.

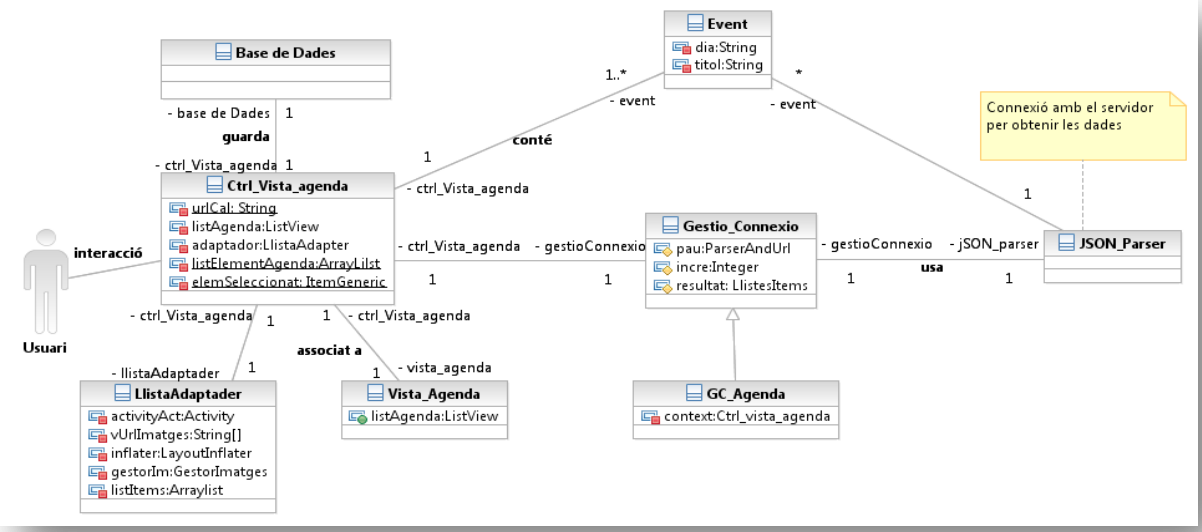

Figura 32. Diagrama UML per a l'obtenció de l'agenda de l'usuari

Per tal de facilitar la lectura es descriuen els elements que es poden observar a la Figura anterior.

- *Ctrl_Vista_Agenda*: és el controlador del cas.
- *LlistaAdapter*: Aquesta classe s'encarrega de mostrar cada element de la llista d'una manera personalitzada, és a dir, tal i com el desenvolupador desitgi.
- *Vista_Agenda*: conté els elements que hi ha a la vista.
- *Event*: element que pertany en el model.
- *Gestió_Connexió*: classe que s'encarrega de gestionar les connexions dels diferents controladors.
- *GC_Agenda*: s'utilitza per retornar el control a l'activitat.
- *JSON_Parser*: la seva funció és basa en obtenir la informació del servidor, parsejar-la i retornar-la a la classe *GestióConnexió*.
- *Base de Dades*: es guarda la informació que arriba del servidor per a futures consultes.

Un cop explicats els diferents elements participants, es veurà quina és la funció del controlador en aquest cas. Per tal de descriure'l, s'ha fet mitjançant la descripció de l'evolució del cas d'ús. S'ha suposat que no hi havia dades a la base de dades, ja que així, es pot veure el procés principal del cas.

1. L'usuari accedeix aquesta funcionalitat.
2. El controlador importa la vista. Carrega la classe *Vista_Agenda* i assigna els elements que hi ha a *Vista_Agenda* a elements de la classe, és a dir, *listAgenda* li assigna la llista que hi ha definida a *Vista_Agenda*.
3. El controlador crida a la classe *Gestió_Connexió*, passant-li com a paràmetres la *URL*.
4. La classe *Gestió_Connexió* rep la petició i gestiona la classe parsejadora que li correspon, ja que, com es pot veure en el diagrama de classes hi ha 2 parsers més (*XML*[21] i *ICAL*[22]). En aquest cas però, serà la de *JSON*[23], la qual se li passa la *URL* perquè processi la informació.

5. La classe *JSON_Parser* rep la petició, es connecta el servidor, parseja la informació que li arriba (se suposa que tot ha estat correcte), i crea els corresponents *Events*. Retorna la llista a la classe *Gestió_Connexió*.
6. Aquesta mitjançant la classe *GC_Agenda*, torna la prioritat al controlador (*Ctrl_Vista_Agenda*).
7. El controlador, obté les dades, les guarda a la Base de Dades i crida a la classe *LlistaAdapter*.
8. Aquesta, és una classe "personalitzada" és a dir, ens permet mostrar les llistes amb el disseny que nosaltres desitgem.
9. Es mostra la llista a l'usuari. Un cop mostrada la llista, el controlador és qui rebrà la petició de l'usuari d'accedir algun element de la llista.

Finalment, per tal de veure en més detall el procés anteriorment descrit, s'ha cregut oportú representar-ho mitjançant el model de comportament, el qual es troba a la *Figura 33*.

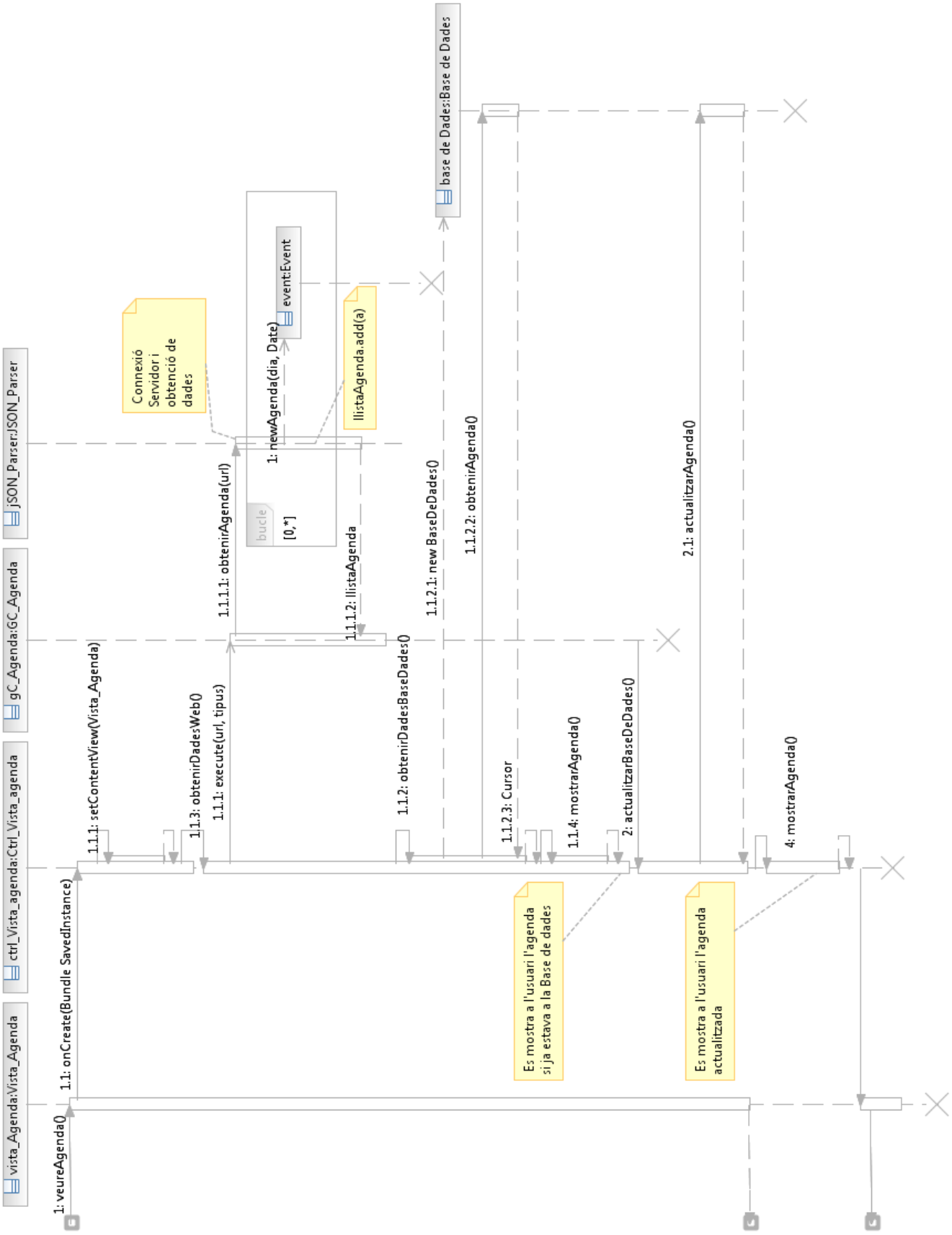

Figura 33. Model de Comportament del cas 1

CAS2- MOSTRAR L'HORARI

Aquest cas, s'ha centrat en el procés de com es mostra l'horari a l'usuari. Val a dir que el procés d'obtenció de les dades del servidor, idealment, només s'hauria de produir un cop al llarg del quadrimestre, ja que, s'obté tot l'horari. El procés de connexió amb el servidor no es mostra degut a que es fa de la mateixa manera que el cas 1. Així doncs, en aquest cas es veurà la segona manera de mostrar la informació: accés a base la de dades. Aquest cas, a l'igual que l'altre està ubicat dins la pestanya Racó. Es pot veure el cas a la *Figura 34*.

Per tal de facilitar la lectura s'han descrit els elements que es poden observar.

- *ControladorVistaHorari*: és el controlador del cas.
- *LlistaHorari*: conté les propietats dels elements que hi hauria dins la vista horari.
- *VistaHorari*: conté els elements que hi ha a la vista.
- *EventHorari*: element del model que conté les propietats d'un dia en una hora.
- *Base de Dades*: es guarda la informació que arriba del servidor per a futures consultes.

Un cop explicats els diferents elements participants, es veurà quina és la funció del controlador en aquest cas. Per tal de descriure'l, s'ha fet mitjançant la descripció de l'evolució del cas d'ús.

1. L'usuari accedeix en aquesta funcionalitat.
2. El controlador importa la vista. Carrega la classe *VistaHorari* i assigna els elements que hi ha a *VistaHorari* a elements de la classe, és a dir, a la llista a *mListHorari*.
3. El controlador crida a la classe *BaseDadesManager* amb la informació del dia actual.
4. La base de dades li retorna una llista d'elements.
5. El controlador transforma els elements de la base de dades a *EventHorari*.
6. El controlador els assigna a la llista d'*EventHorari*.
7. El controlador mostra l'horari a l'usuari.

Finalment, per tal de veure en més detall el procés anteriorment descrit, s'ha cregut oportú representar-ho mitjançant el model de comportament que mostra la *Figura 35*.

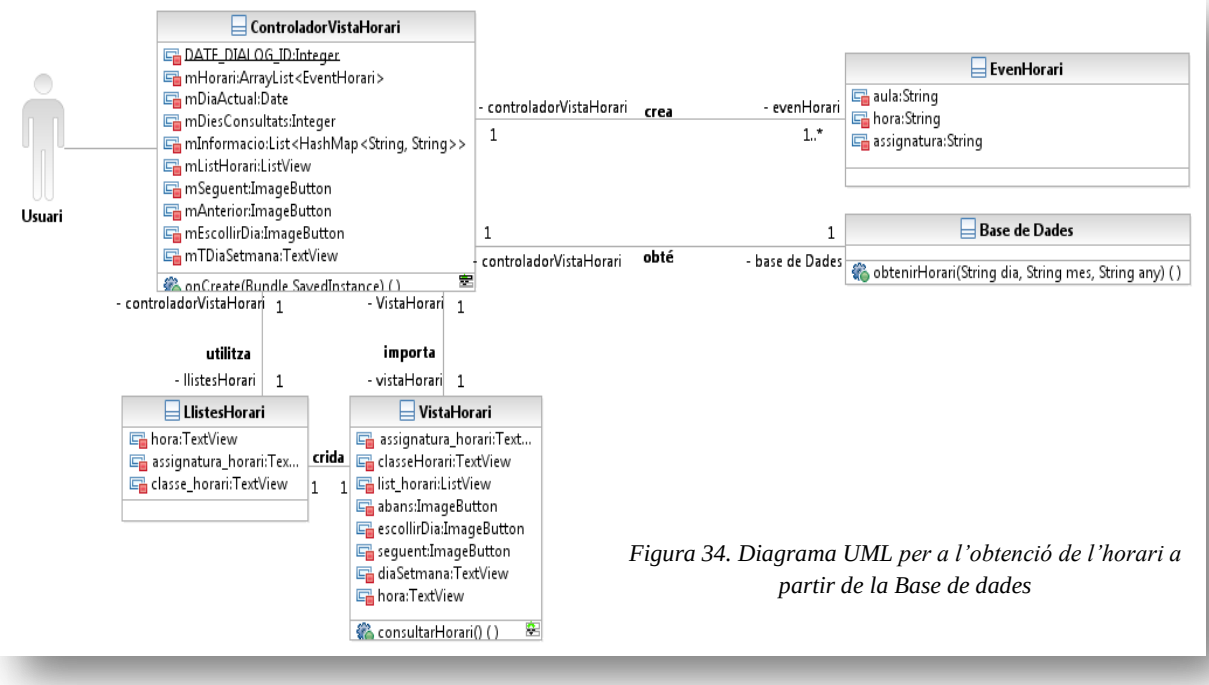

Figura 34. Diagrama UML per a l'obtenció de l'horari a partir de la Base de dades

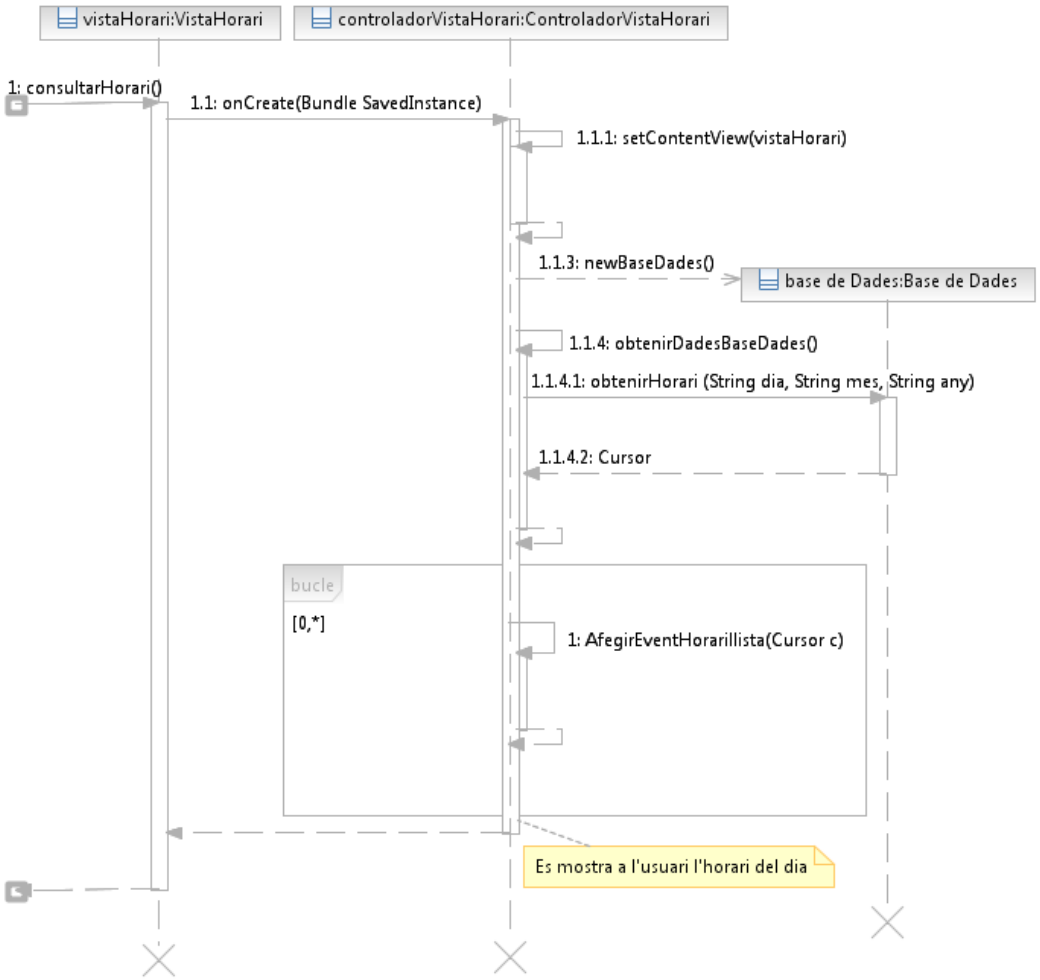

Figura 35. Model de Comportament de cas 2

CAS3- MOSTRAR LA INFORMACIÓ D'UNA ASSIGNATURA

Aquest últim cas, s'ha centrat en el procés de com s'obté, es guarda i es mostra la informació d'una assignatura. Aquest és diferent del primer degut a que la connexió es realitza directament amb la classe que parseja la informació. Aquest fet, és degut a que la crida és bloquejant, és a dir, fins que no s'ha obtingut la informació no es retorna al fil principal. Aquest cas es pot trobar en les assignatures de la pestanya FIB o bé a la zona Racó, quan es consulta la informació de les assignatures que s'està cursant. En la *Figura 36* es pot el veure diagrama *UML*.

Per tal de facilitar la lectura s'han descrit els elements que es poden observar.

- *ControladorVistaInformacióAssignatura*: és el controlador del cas.
- *vista_assignatura_fib_info*: conté els elements que hi ha a la vista.
- *Assignatura*: element del model que conté les propietats de l'objecte que es volen mostrar.
- *Base de Dades*: es guarda la informació que arriba del servidor per a futures consultes.

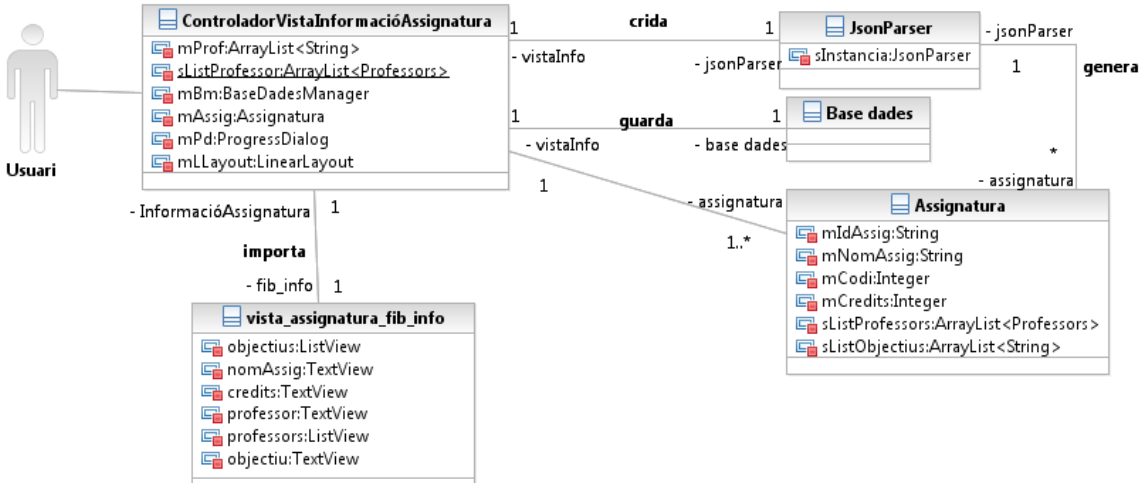

Figura 36. Diagrama UML per a l'obtenció de la informació d'una assignatura

Un cop explicats els diferents elements participants, anem a veure quina és la funció del controlador en aquest cas. Per tal de descriure'l, ho farem mitjançant la descripció de l'evolució del cas d'ús.

1. L'usuari accedeix aquesta funcionalitat.
2. El controlador importa la vista. Carrega la classe *Vista_assignatura_fib_info* i assigna els elements que hi ha elements de la classe.
3. El controlador busca la informació a la Base de Dades. Si no hi ha informació a la base de dades, invoquem la crida al parsejador.
4. El parsejador obté la informació i crea una llista d'assignatures.
5. El controlador actualitza la base de dades amb la nova informació
6. El controlador els assigna als diferents elements de la classe *Vista_assignatura_fib_info*.
7. El controlador mostra la pantalla d'informació de l'assignatura.

Finalment, per tal de veure en més detall el procés anteriorment descrit, s'ha cregut oportú, tal i com es veu a la *Figura 37*, representar-ho mitjançant el model de comportament.

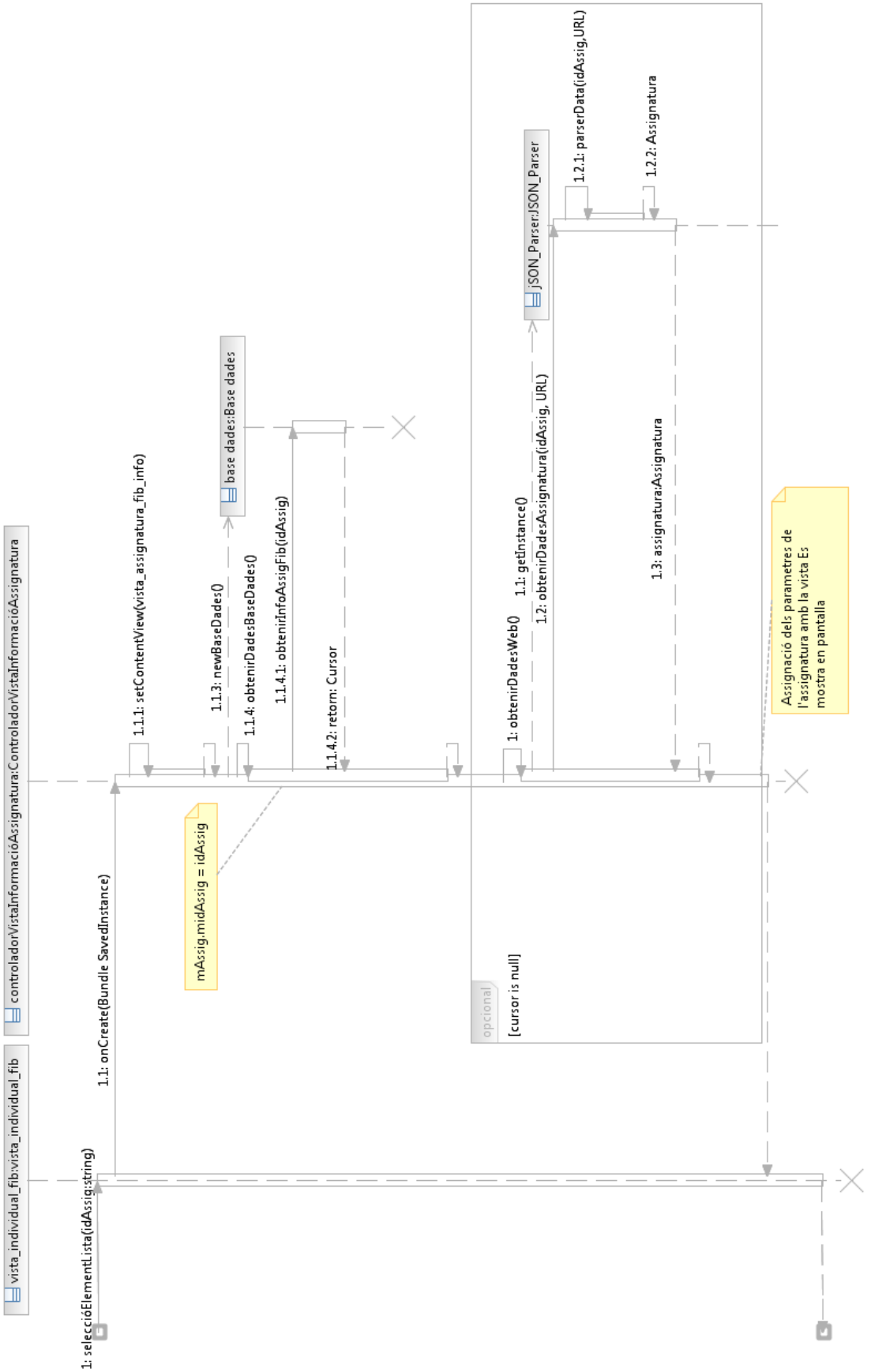

Figura 37. Model de comportament del cas 3

3.4.4. DIAGRAMA DE CLASSES

En la *Figura 38* es pot apreciar una visió global del sistema. En aquesta hi ha representats els tres components del patró de disseny Model-Vista-Controlador. A continuació es mostra una descripció dels trets més rellevants del diagrama.

- A la part superior es poden veure les vistes. Com ja s'havia mencionat, cada vista té el seu propi controlador assignat. Aquest és específic de cada una i realitza la funció concreta que li pertoca. En cada classe que defineix una vista i els elements que aquesta conté.
- Els controladors que estan en la tercera fila, fan d'enllaç entre el model i les vistes. En el diagrama, perquè sigui més fàcil d'entendre, no s'ha representat les relacions entre la classe *Gestió_Connexió* i els controladors. En general, tots els controladors els atributs que contenen són:
    - Elements de la vista: s'utilitzen per poder captar els events que es produeixen.
    - Adaptadors: s'utilitzen per poder mostrar a l'usuari les llistes de manera personalitzada. No en el format per defecte que ofereix Android.
    - Elements de connexió: s'utilitzen per poder realitzar tot el procés d'obtenció de dades. Les classes són: *ParserAndUrl*, *LlistesItems* i *Gestió_Connexió* explicades en el punt 3.5.1. Model Conceptual.
- Finalment tenim el model a la part inferior. En aquest hi ha els elements que el sistema mostrarà a l'usuari:
    - *Event*: representa els elements que conformen l'agenda.
    - *EventHorari*: representa els elements que conformen l'horari.
    - *ItemGeneric:* classe abstracte que conté atributs compartits per les classes *Notícia*, *Correu* i *Avis.*
    - *Aula:* representa els elements que conformen les aules.

  I finalment, els que realitzen el procés d'obtenció de dades i emmagatzament.

3.4.5. DIAGRAMA DE CLASSE

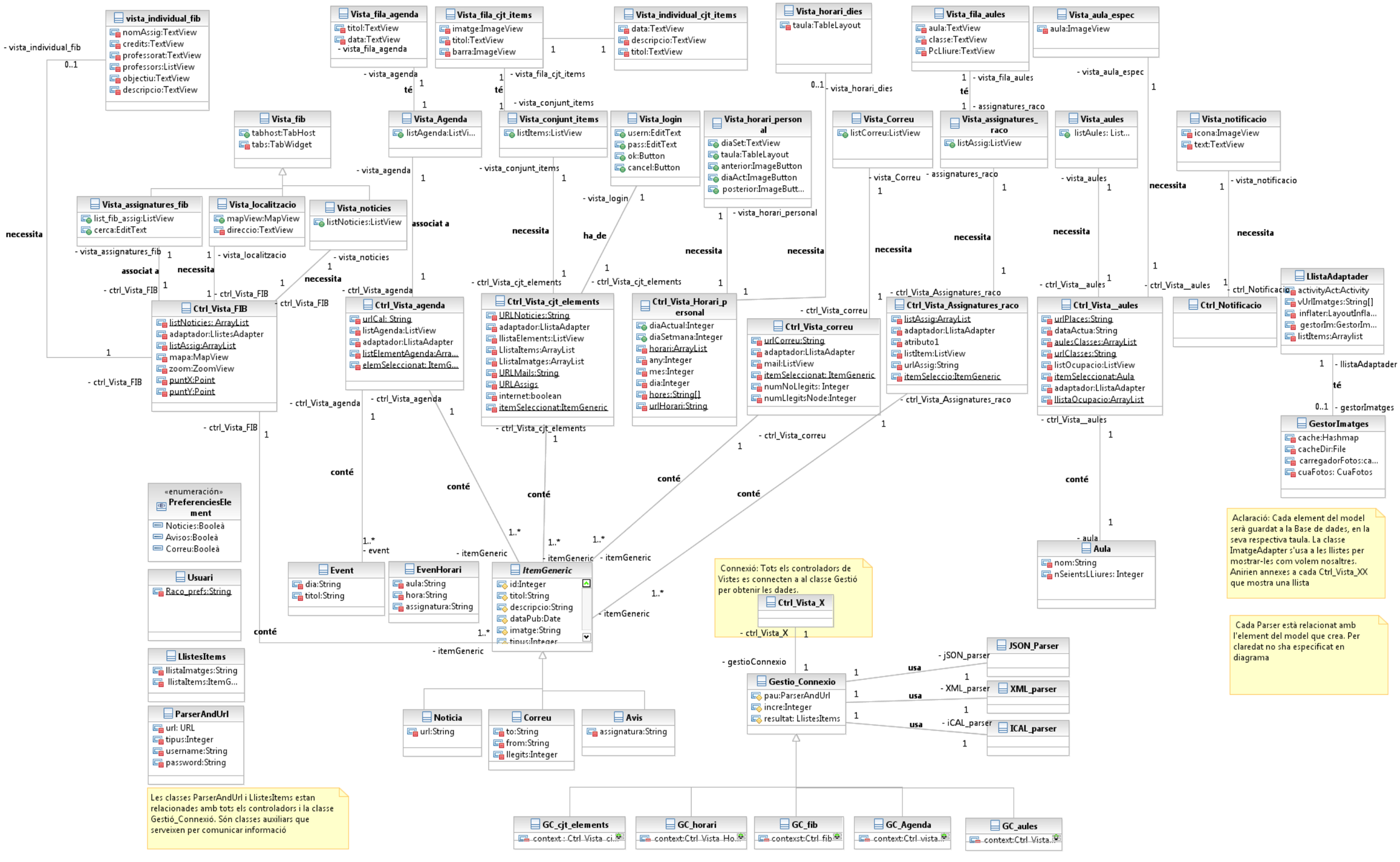

Figura 38. Diagrama de classes de la iteració 1

3.5. ITERACIÓ 2

Aquesta iteració va començar a finals de juny i va durar fins a l'última setmana d'agost.

Va consistir en la fase d'implementació final del *Racó Mobile*. Després d'analitzar els requisits dels usuaris i del sistema, les funcionalitats de l'aplicació, especificar els diferents casos d'ús, escollir la millor forma de mostrar la informació i com tractar-la, i finalment, decidir l'arquitectura de disseny que s'aplicaria, es va procedir a realitzar la implementació del projecte.

En la iteració 1, es van realitzar diferents diagrames de com es pensava que l'aplicació es podria desenvolupar. No obstant, un cop s'estava implementant el codi, es va estar obert a nous canvis que poguessin aparèixer, sempre amb caràcter de millora. Així doncs, en aquesta iteració, s'explica quins són els principals canvis que s'han realitzat respecte les decisions i diagrames de la iteració 1. Per fer-ho s'ha basat de nou, amb l'estructura de *MVC*, ja que així, es pot explicar en més detall cada un d'ells.

Comencem doncs, pel primer diagrama que tal i com s'ha fet en la iteració 1, consisteix en el model conceptual.

3.5.1. MODEL CONCEPTUAL

En la *Figura 39*, es pot apreciar el model conceptual resultant un cop acabada la implementació de l'aplicació. Com es pot veure, falta una classe important. Aquesta és la que gestiona la base de dades de l'aplicació, la qual es troba explicada en detall més endavant.

A continuació es pot veure una breu explicació sobre els canvis que s'han realitzat respecte la iteració 1.

Els canvis que s'han produït són:

Classe AssignaturesAvisos

Les llibreries d'Android ofereixen certes funcions per mostrar les llistes les quals es poden cridar amb tipus d'informació bàsica, com per exemple un *String*. No obstant, amb la creació d'adaptadors de llistes es poden realitzar crides en els nostres propis objectes.

Tan les llibreries, com les pròpies implementacions dels adaptadors, només poden rebre un sol tipus d'informació. Una mateixa crida no pot contenir objectes de diferents tipus, per aquest motiu, s'ha creat aquesta classe. Així doncs, en la vista de la pestanya Racó on es mostren les assignatures i els avisos que l'alumne té matriculat, es va utilitzar aquesta classe per transmetre a l'adaptador les dades que es mostren a l'usuari.

La classe Event no podia ser genèrica

En la iteració 1, es va dissenyar aquesta classe per simbolitzar varis elements del model com podia ser l'agenda o l'aula i l'horari (tot i que es podien realitzar petits canvis per adaptar-ho). Finalment, com es pot observar en el model anterior, es va decidir crear una classe per l'agenda, l'aula i l'horari per separat. Així, la seva gestió en les vistes era més senzill i entenedor, i alhora també quedava més clar en el model.

Nous atributs

Aquest és un canvi que es produeix en tot projecte, ja que per molt que s'intenti afinar el màxim el model inicial la implementació és molt diferent i sempre sorgeixen noves idees i/o canvis per tal de millorar l'estructura i implementació. Així doncs, si es compara els diagrames

resultants de les dues iteracions es pot observar certs canvis, per exemple, l'aparició dels tres nous objectes del model explicats en el punt anterior o l'aparició de la classe assignatura o la relació entre assignatures i avisos. Amb aquesta reestructuració, s'obtenen classes més específiques i es poden gestionar millor les vistes. Alhora de mostrar un objecte no s'ha de realitzar cap tipus de filtratge, com s'havia de fer en l'anterior model per distingir de si es tractava d'una aula o bé l'horari.

Renom de la classe Usuari

En la iteració 1, la classe *Usuari* representava les preferències que un usuari podia tenir en el sistema. En aquesta iteració, es va creure que l'objecte *Usuari* feia referència a una persona i no a les seves preferències. Així doncs, per aquest motiu es va decidir canviar-li el nom.

L'atribut RACÓ_PREFS

Aquest atribut, és el nom de les preferències que hi ha al sistema per a què es pugui accedir-hi des de qualsevol punt. En el mòbil, aquestes es guarden en un fitxer del sistema o bé en un de predefinit per l'aplicació. En aquest cas, s'ha decidit donar-li un nom particular per tal de poder-les distingir de les de sistema.

La classe IcalParser

Aquesta no està relacionada amb la classe *GestióConnexó* com en un principi s'havia dissenyat. Aquest fet, és degut a que només hi ha una classe que necessita les seves dades i que la connexió no es fa asíncrona, és a dir, es realitza de manera que l'aplicació es bloqueja fins que s'han obtingut les dades. Per tant, l'obtenció de dades es realitza directament des del controlador.

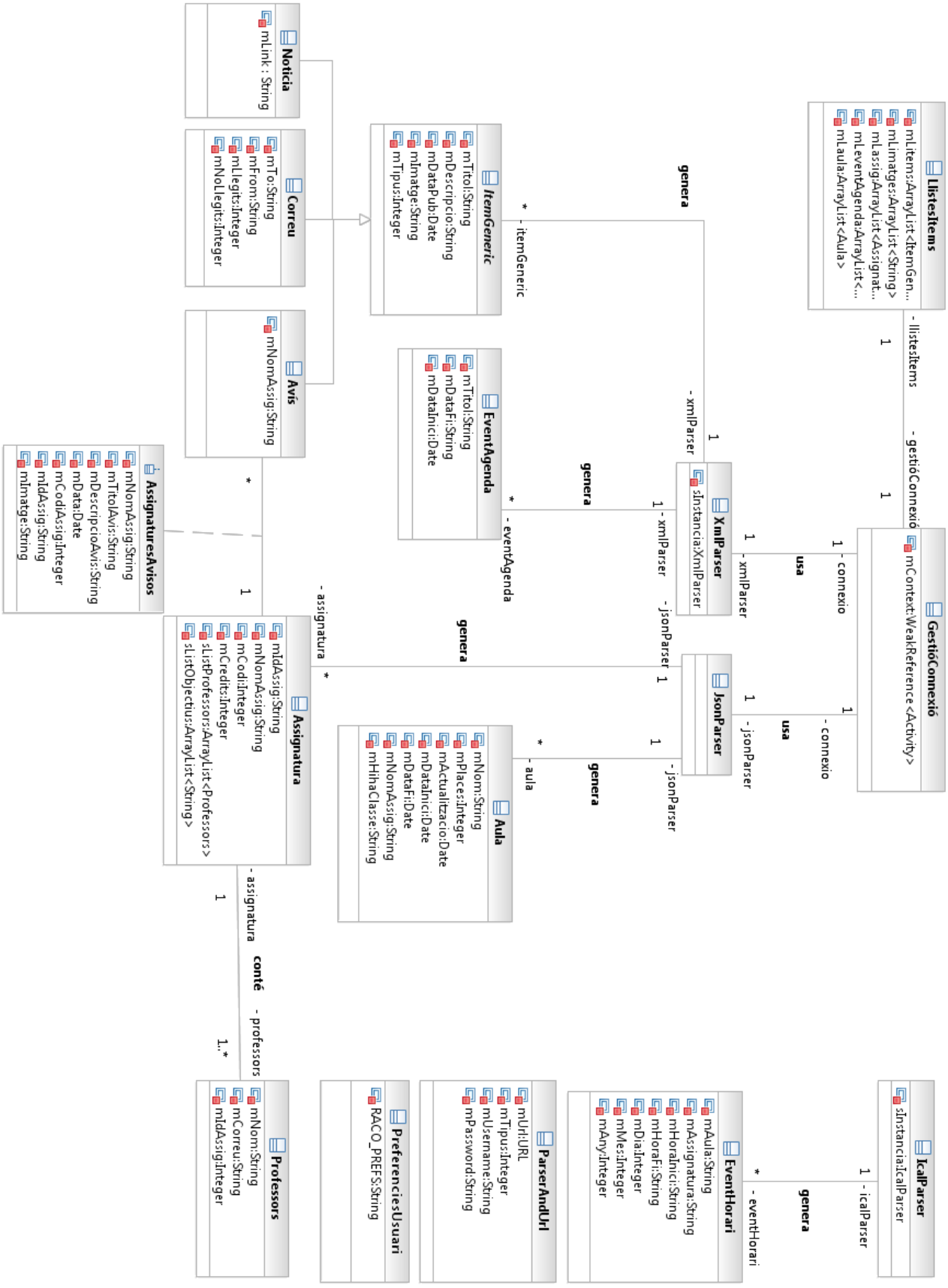

Figura 39. Definició del model de la iteració 2

La classe Base dades Manager

La *Figura 40*, tot i que es mostra a part del model, també en forma part. La decisió de realitzar-ho així ha estat per fer més entenedor el diagrama i la pròpia classe.

S'ha aprofitat també per descriure els atributs que es poden observar. Primer de tot, s'observa que la Base de Dades, conté molts atributs. Aquests fan referència a diferents noms que s'han utilitzat per crear les taules. Són utilitzats en diverses funcions dins la mateixa classe. L'objectiu de crear-los d'aquesta manera fou pensat perquè en cas que s'hagi de realitzar un canvi en una columna o taula sigui el més fàcil i ràpid possible.

Els últims tres atributs fan referència a elements que *SQLite* [26] necessita per poder realitzar les operacions que se li demanen.

Figura 40. Definició de la Base de Dades

3.5.2. VISTA

En la *Figura 41*, es pot apreciar les vistes resultants de l'aplicació. En l'anterior iteració s'ha vist com es representen internament, així que, en aquesta s'ha aprofitat per explicar les principals vistes i les que són més de tipus auxiliar.

Splash

Aquesta vista apareix quan s'inicia l'aplicació. Serveix per mostrar el logotip i el nom de l'aplicació.

Té un temps limitat d'aparèixer, un cop passat aquest, es mostra la següent vista: Inici Aplicació.

Inici Aplicació

Com es pot veure aquesta es pot presentar de dues maneres diferents. Una d'elles és la que es troba quan s'obre l'aplicació per primera vegada (és la vista que està a la part). En aquesta, es mostra un element de configuració de l'usuari del Racó i les notícies que publica la FIB.

L'altra vista que es pot veure és quan ja tenim configurat l'usuari del Racó. En aquest cas es mostren els avisos, els correus i les notícies. Els tres elements es mostren ordenats per data.

Destacar que amb les dues vistes es pot accedir a la zona FIB, per contra, a la zona Racó només es pot accedir-hi amb el *Login* realitzat.

Vista Error

Aquesta apareixerà quan l'usuari demani consultar dades, per exemple, la informació d'una assignatura, i no es tinguin dades per subministrar o bé es produeixi un error a l'obtenir les dades.

Zona Fib

Aquesta zona és pública per qualsevol usuari. Així doncs, es podran visitar les notícies que publica la facultat, quina és la localització dels principals edificis de la facultat i també la informació de les assignatures del Grau que actualment s'imparteixen.

Zona Racó

Aquesta part és privada per a cada estudiant. En ella es podrà consultar, mitjançant les pestanyes que es veuen a la imatge, les diferents funcionalitats.

Destacar que en el calendari es podrà realitzar una consulta particular d'un dia en concret, o bé, desplaçar-se dia a dia. D'altra banda, en l'ocupació de les aules a part de la informació que es mostra també es pot veure l'estat de les aules visualment.

Notificacions

Com es pot veure en la part superior de la *Figura 41* s'ofereix una vista del què serien les notificacions que rebrà l'usuari cada vegada que es publiqui o es modifiqui un avís d'alguna assignatura que l'usuari estigui matriculat. Aquesta opció, però, es pot habilitar i deshabilitar des de les preferències que hi ha en el menú Vista Inicial.

En la iteració 1, s'ha explicat el funcionament de les vistes a partir dels fitxers *XML* que es creen. Així doncs, s'ha introduït la idea de crear una sola vista per pantalla amb la idea de fer el disseny el màxim específic possible per a cada una.

Finalment, per tal de fer un disseny homogeni amb els mateixos colors, formes, tamanys de lletra, etc. s'ha decidit que les vistes serien més genèriques, intentant així aprofitar el màxim cada vista. L'objectiu era facilitar l'aplicació del disseny i que un canvi es pogués veure reflectit a les vistes corresponents i no haver de replicar aquest canvi a diferents documents.

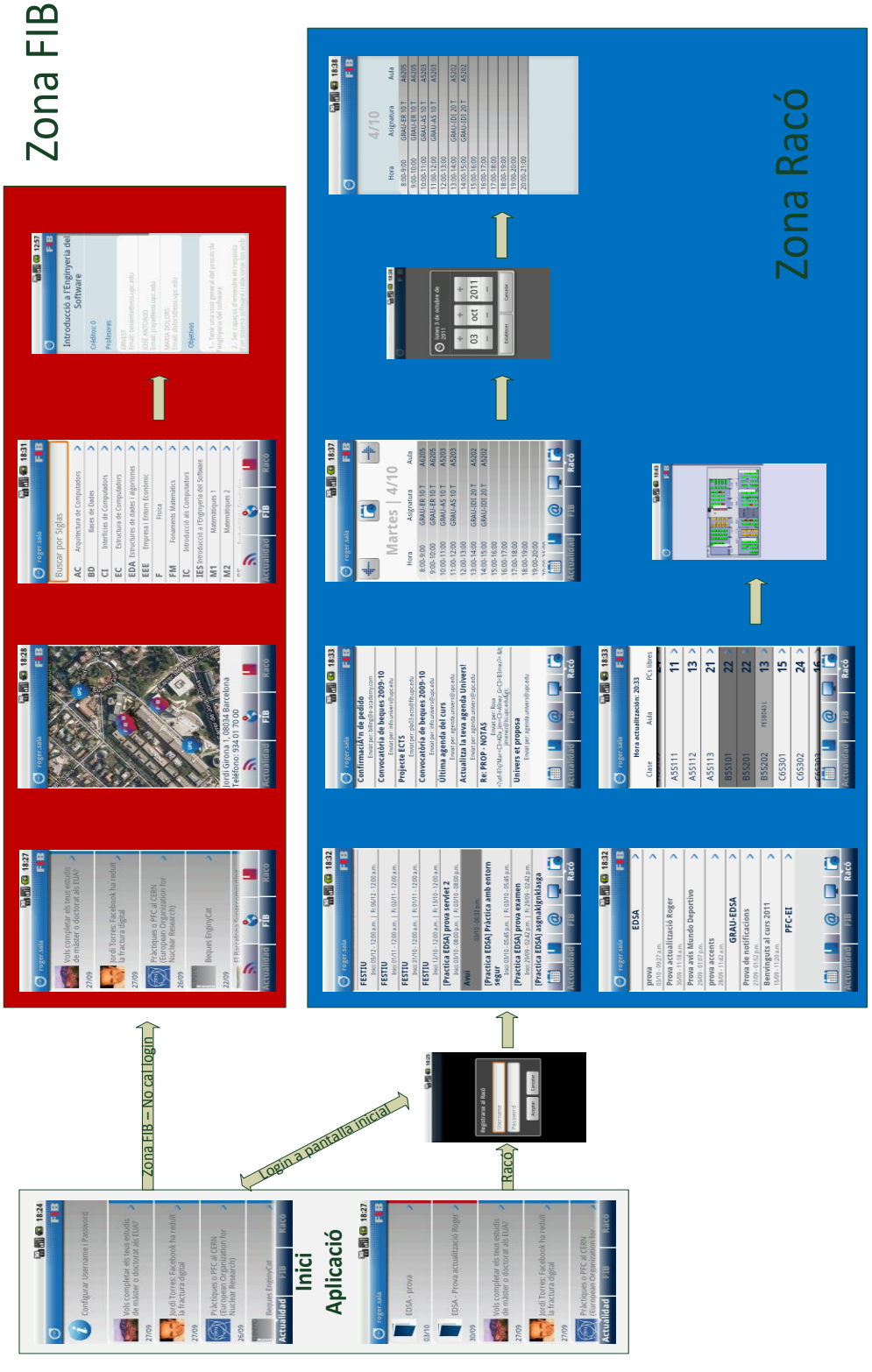

Figura 41. Vistes de l'aplicació final

3.5.3. CONTROLADORS

En la *Figura 42* es pot veure el diagrama final dels controladors de les vistes de l'aplicació. S'ha intentat generalitzar el màxim per tal de que cada classe contingués estrictament els seus elements particulars, intentant evitar així la duplicació d'informació.

A la part superior la classe *ControladorTabGeneric* conté la informació compartida dels tres elements principals. Aquests són els qui contenen les informacions per poder navegar. El *ControladorTabIniApp* és qui crea la vista inferior de pestanyes. Llavors, es generen els altres dos tipus de grups de pestanyes que es troben en l'aplicació. Aquestes aprofiten les mateixes variables de la classe superior. Aquesta acció es pot realitzar degut a que en Android la manera típica de crear grups de pestanyes es realitza de la mateixa manera.

Cada Controlador de pestanya crea les seves particulars vistes. Aquestes són les que s'encarreguen de gestionar la informació que es mostrarà. Es pot veure, observant que cadascuna d'elles conté un vector de la informació que mostrarà i una llista del tipus.

Els adaptadors de les llistes no s'han representat en el diagrama degut a que podria arribar a ser difícil d'entendre. Cadascun d'aquests controladors conté un adaptador de les llistes que és el que gestiona com es mostraran les llistes a l'usuari (veure Secció 4.3).

Les classes que estan representades a la part superior també formen part dels controladors de les vistes. Per exemple, el *Login* és cridat pels controladors *TabIniApp* i *ZonaRacó*, però per una millor comprensió del diagrama no s'ha representat les relacions.

Per últim, val a dir que la gran majoria dels controladors hereten de la classe *GestióActualitzacióLlistesActivity*. Aquest fet és degut a que, en Android, quan es realitzen les peticions de forma asíncrona s'ha de tenir controlat en tot moment quin és l'estat de la vista actual que s'està mostrant a l'usuari. Així doncs, mitjançant aquesta classe, es poden actualitzar les dades a la Base de dades de manera transparent per l'usuari i, en cas que la vista que ha sol·licitat l'actualització de la informació estigui activa, refrescar-la per l'usuari. Tot aquest procés es realitza a partir d'una variable *Context*[16], que es guarda de manera estàtica en aquesta classe.

Pel què fa a les notificacions, no s'han incorporat en aquest diagrama. En la Secció 4.4 es pot apreciar una extensa explicació de què són les notificacions amb Android i com s'han implementat en el Racó a partir del diagrama que obtingut.

A continuació es presenta els aspectes més destacats a nivell de disseny final que es creu que cal remarcar mitjançant el diagrama de classes resultant de l'aplicació.

[16] Interfície a la informació global sobre un entorn d'aplicació. Aquesta és una classe abstracta, la implementació és proporcionada pel sistema Android. Permet l'accés a recursos específics de l'aplicació i les classes, així com les trucades per a les operacions de nivell d'aplicació, com ara activitats de llançament, difusió i recepció dels Intents, etc

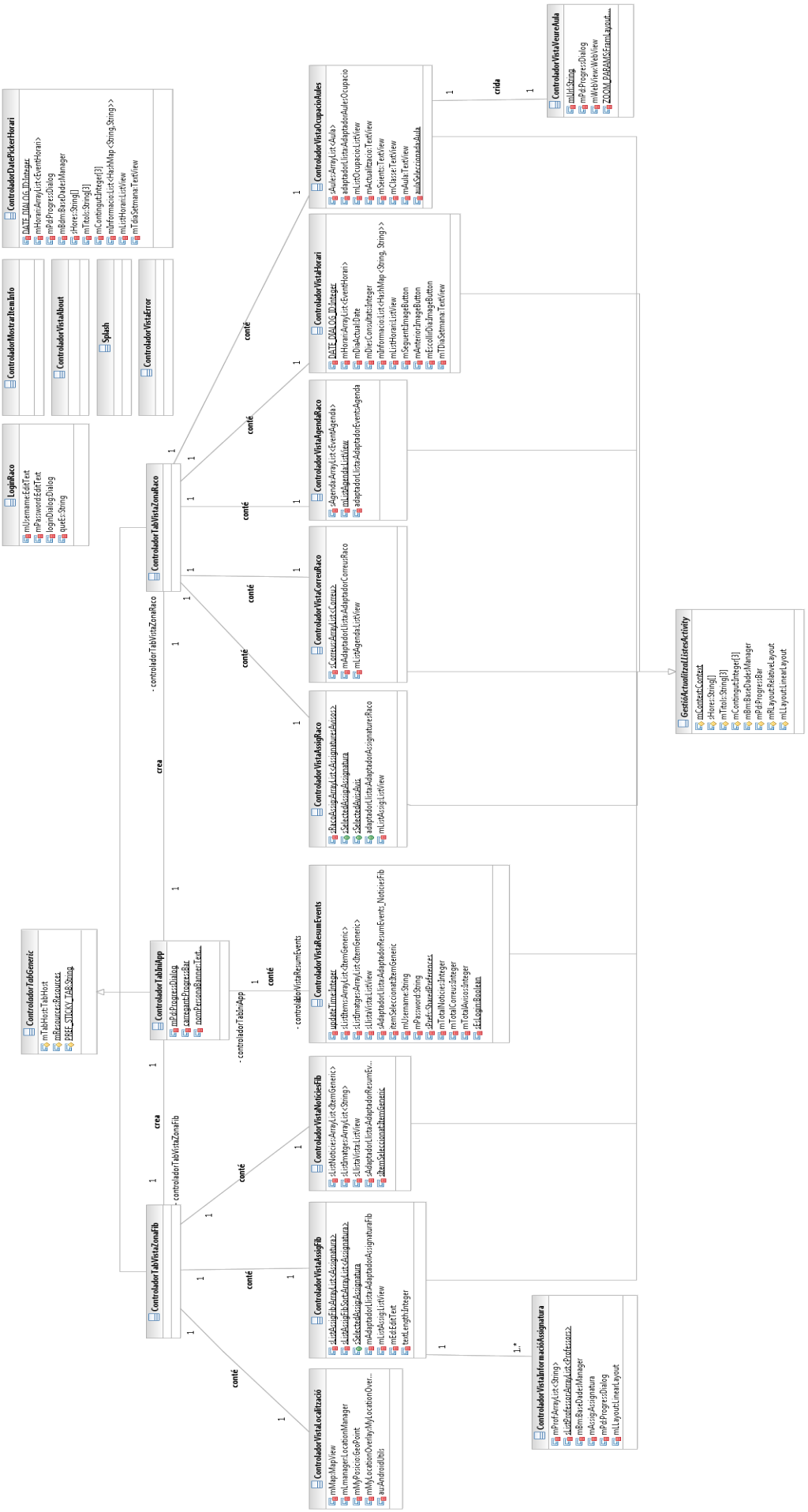

Figura 42. Definició dels controladors de la iteració 2

3.5.4. ASPECTES A DESTACAR DEL DIAGRAMA DE CLASSES

Com es pot observar a la *Figura 45* no es mostren les vistes. S'ha considerat que és més important destacar el procés d'obtenció de les dades i el seu tractament, és a dir, la interacció entre controladors i model.

Tal i com s'ha destacat anteriorment, cada controlador (en la part superior del diagrama) és l'encarregat d'obtenir la informació que mostrarà i processar-la per tal de mostrar-la a l'usuari. Aquest procés, es pot dividir en dos parts: Obtenció de dades (controlador a model) i processament de les dades (model a controlador). Primer de tot, expliquem el procés d'obtenció de dades.

Per tal de realitzar aquest procés, es menciona de nou, la classe *ParserAndUrl*. La seva funció és facilitar la transmissió de dades entre els controladors i la classe *GestióConnexió*. Així doncs, es creen instàncies de la classe on s'indica el parser que es necessita per aquella funcionalitat i la *URL* per connectar-se al servidor. La classe *GestióConnexió* pot administrar l'obtenció de les dades correctament.

Les dades es poden obtenir mitjançant connexions asíncrones o síncrones. La primera d'elles té l'avantatge de poder realitzar una petició i paral·lelament mostrar les dades que es tenen guardades a la base de dades per a què l'usuari pugui consultar-les, és a dir, no es bloqueja la vista a l'usuari. L'esquema que es mostra a la *Figura 43* mostra el procés que segueix cada controlador que utilitza aquest mètode.

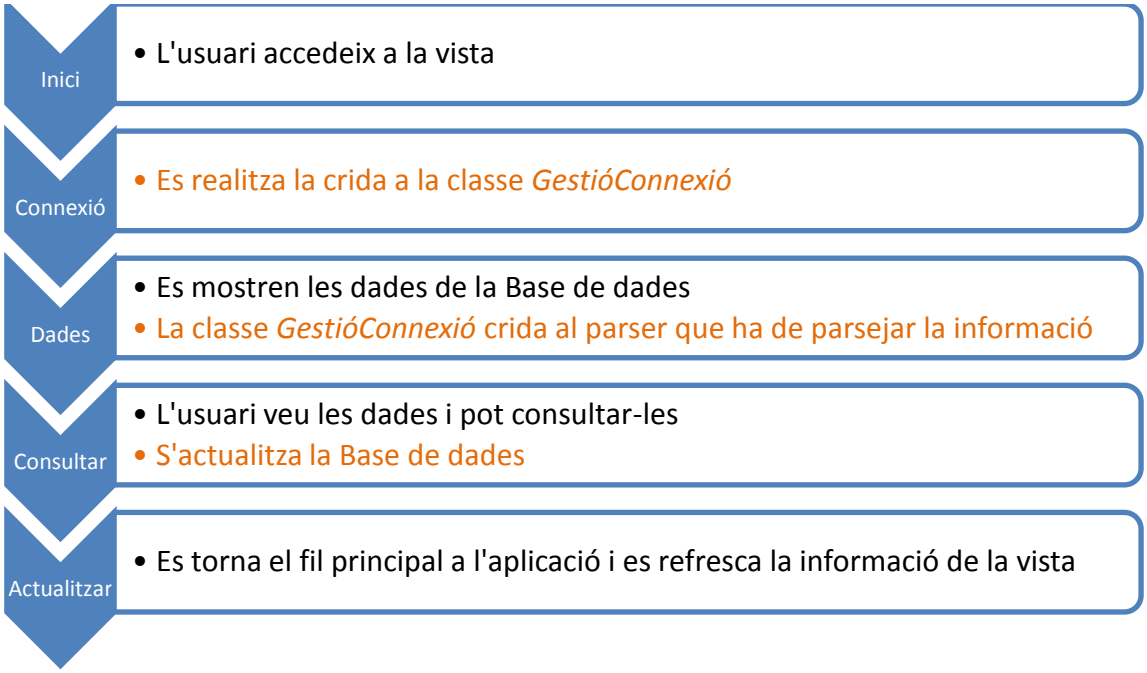

Figura 43. Diagrama d'una crida asíncrona

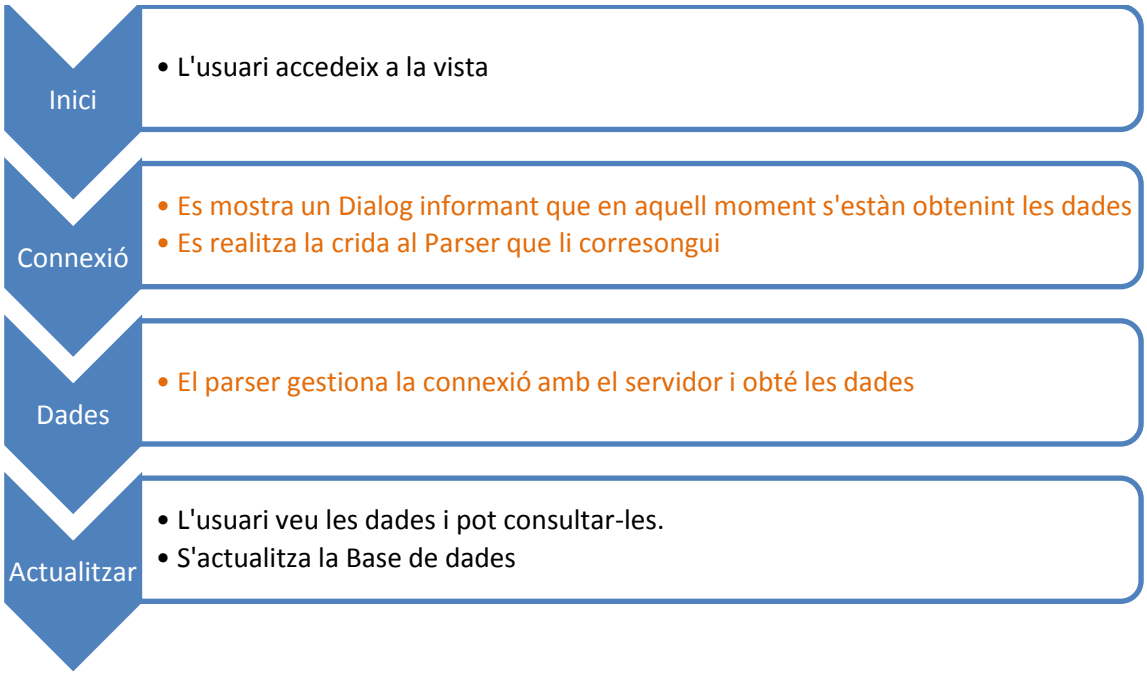

Figura 44. Diagrama d'una crida síncrona

Per contra, les crides síncrones, bloquegen el procés principal de la vista en particular durant el període d'obtenció de dades. Per tal de poder informar a l'usuari de l'estat del procés i evitar que pugui interactuar amb el telèfon mentre es realitza la crida i processen les dades es poden mostrar *Dialogs*[17].

Aquest procés el realitzen les classes que tenen una connexió directe amb els parsers, en aquest cas l'*IcalParser* i *JsonParser*. Per tal de veure-ho amb més exactitud es pot observar la *Figura 44* següent, on es mostra una descripció del procés que segueix.

Un cop vist com és el procés d'obtenció de les dades, és a dir, la relació dels controladors cap al model, anem a veure la relació inversa: el processament de les dades.

La classe anomenada *LlistesItems*, serveix per retornar la informació que es necessita per mostrar a cada vista. Aquesta conté les diferents llistes d'elements que en algun moment s'hauran de processar. La classe *GestióConnexió* crea una instància i omple les llistes indicades en el procés anteriorment descrit. Així doncs, instanciant aquesta classe es pot arribar a obtenir la informació que teníem pendent per rebre. La classe *GestióConnexió* només és instanciada en les connexions asíncrones ja que, és aquí on es necessita saber quina informació se'ns retorna. En les crides síncrones, per contra, la informació ja és retornada al moment i no és necessari crear cap classe auxiliar. Seguidament els controladors poden instanciar *LlistesItems* i actualitzar les dades a la Base de Dades.

[17] Un diàleg és en general una petita finestra que apareix al capdavant de l'activitat actual.

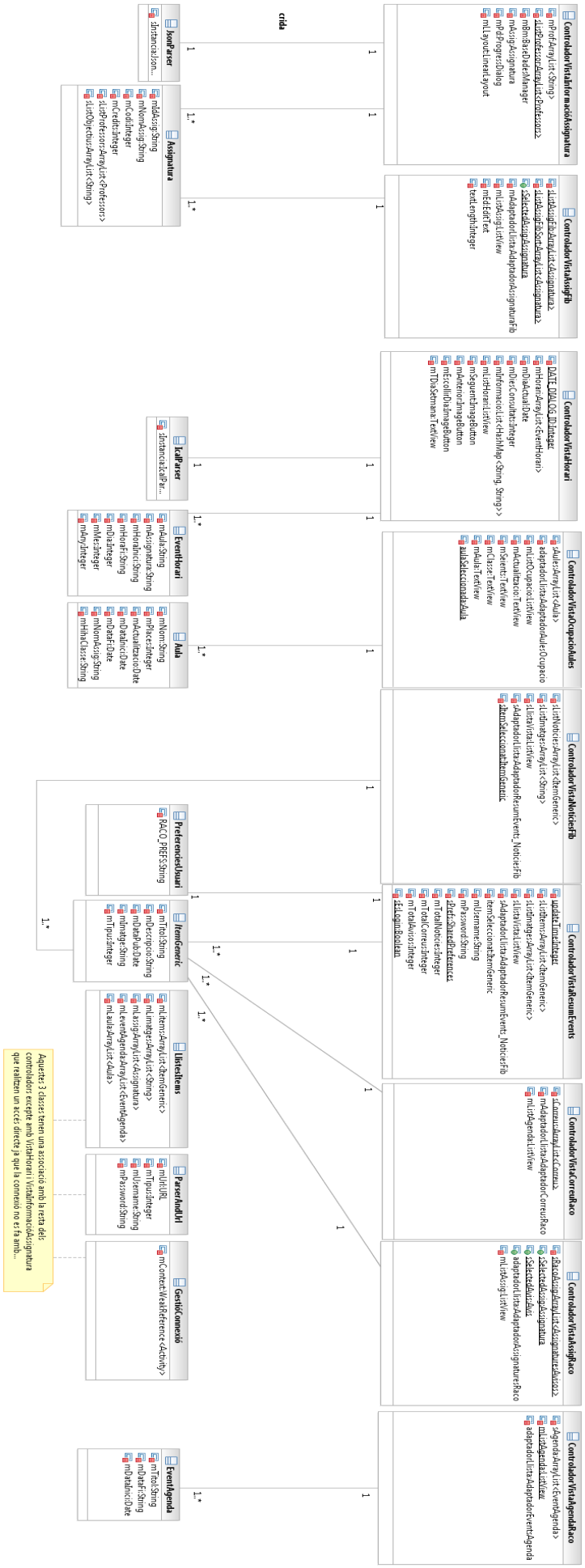

Figura 45. Aspectes destacats del diagrama de classes final

3.6. APUNT FINAL

Arribats aquest punt s'ha pogut veure en detall el procés que s'ha seguit per tal de desenvolupar l'aplicació des del mes de febrer de 2011 fins al setembre/octubre del mateix any. S'ha vist detalladament les tres iteracions que la implementació del projecte ha tingut i el resultat final obtingut.

Tot realitzant la implementació de l'aplicació, ens vam adonar de certs detalls que es va creure que era important incloure en aquest document, ja que, són fets no trivials i que val la pena explicar en detall. Així doncs, es va incloure el següent punt en el document, on es troben explicats.

Capítol 4

DETALLS SOBRE LA IMPLEMENTACIÓ

En els apartats anteriors, s'ha estat explicant en detall quin ha estat el procés d'implementació de l'aplicació i els diferents motius que ens portaven a realitzar les decisions que s'han pres.

Aquest punt, s'ha cregut convenient destinar-lo a explicar els principals aspectes que s'han de tenir en compte alhora de començar un projecte Android. La programació per a dispositius mòbils a nivell històric està poc detallada degut a que és un fet recent. Mentre es realitzava el projecte s'han anat anotant diferents inconvenients que s'han trobat.

Al llarg d'aquest capítol es presenta en primer lloc una descripció de com està estructurat un projecte en Android (veure Secció 4.1). En segon lloc, la definició de les vistes usant el format *XML* (veure Secció 4.2). En tercer lloc, què és un *Base Adapter* i perquè s'utilitza (veure Secció 4.3). En quart lloc, s'explica com implementar un sistema de notificacions (veure Secció 4.4). En cinquè lloc, què és el *Proguard* i perquè s'utilitza en Android (veure Secció 4.5). Per acabar es detallen diferents aspectes que cal destacar en la implementació del projecte (veure Secció 4.6).

4.1. ESTRUCTURA D'UN PROJECTE ANDROID

Quan es comença un projecte en Android, el primer de tot és preparar l'entorn. Un cop es té el projecte creat, aquest té diferents directoris. En un principi pot arribar a desconcertar quina és la seva funcionalitat i/o per a què s'haurien de fer servir.

En la *Figura 46*, es pot observar una captura de pantalla de com s'estructura un projecte inicialment. En primer lloc, es pot veure que de l'arrel del projecte es diferencien 4 directoris: *src, gen, assets i res*. Cadascun d'ells realitza una funció específica que descrivim a continuació:

- *src*: Aquest directori és on hi ha els fitxers Java els quals s'implementa l'aplicació. En ell està permès crear tots els *packages* que siguin necessaris. Android per defecte crea el *package* que es defineix al crear el projecte.

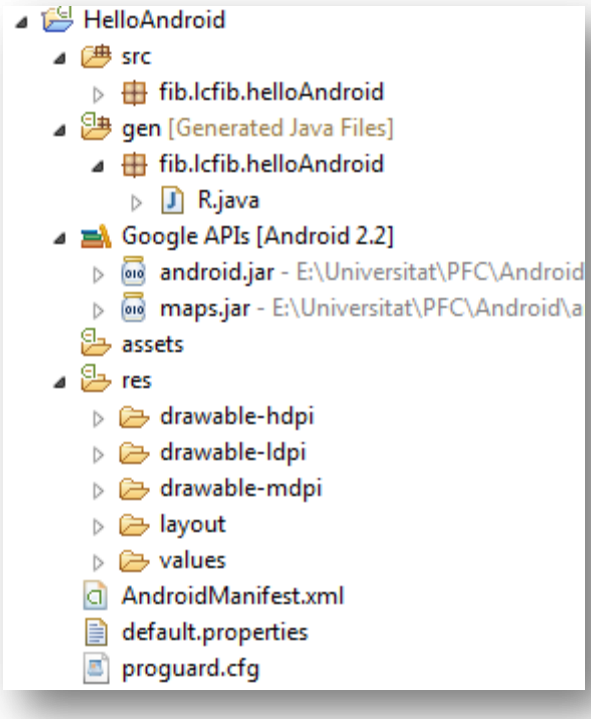

Figura 46. Estructura inicial d'un projecte Android

- *gen*: Aquest directori no és editable per l'usuari. En ell es creen diferents arxius que s'eliminen i es tornen a crear dinàmicament per tal de que es puguin usar elements definits en altres directoris. Un dels fitxers que es creen automàticament i que és important de conèixer el seu funcionament és la classe *R.java*, la qual es pot observar a la *Figura 46*.
    - *R.java:* Aquesta classe fa d'intermediària entre els elements creats en el directori *res* i *src*. Si s'observa el seu contingut, es pot observar que indexa tot el directori *res*. Així, quan des d'un arxiu Java del directori *src* es necessita un element mitjançant una crida similar a: *R.drawable.nom_element* li permet obtenir i treballar amb aquest. En l'exemple anterior, s'ha usat el subdirectori *drawable* però es podria haver usat qualsevol dels altres que hi ha definit a l'interior de *res*. Més endavant es veurà quin és el contingut de cada subdirectori.
- *res:* Aquest directori conté tots els subdirectoris que dóna'n suport a les vistes. Tots els subdirectoris realitzen una funcionalitat específica amb l'objectiu que en l'aplicació es vegin els elements d'interacció, menús, imatges etc. a continuació es pot veure què conté cada subdirectori generat per defecte quan es crea un projecte:
    - *drawable*-hdpi: conté les imatges que hi ha a l'aplicació per a pantalles que tenen una definició: *640dp* x *480dp*.
    - *drawable*-ldpi: conté les imatges que hi ha a l'aplicació per a pantalles que tenen una definició: *426dp* x *320dp*.
    - *drawable*-mdpi: conté les imatges que hi ha a l'aplicació per a pantalles que tenen una definició: *470dp* x *320dp*.
    - *layout*: conté la definició de les vistes, és a dir, conté els diferents fitxers que alhora contenen els elements que es mostraran a les vistes.

- *values*: conté els diferents valors que poden prendre les informacions que hi ha a les vistes. Per exemple, pot haver-hi el fitxer *Strings.xml* que permet definir el text a mostrar i alhora permet que l'aplicació sigui multilingüe. També, el fitxer *Colors.xml* que permet definir els colors de l'aplicació amb constants.
  Hi ha una nomenclatura per realitzar l'aplicació multilingüe la qual consisteix en crear varis directoris *Values* amb el format següent:
  - *values-es* (Castellà), *values-en* (Anglès), i així per tots els idiomes que pot suportar els dispositius Android. Dins d'aquests directoris es crea un nou fitxer *Strings.xml* amb les mateixes etiquetes i la corresponent traducció.

  Si es considera necessari, també es poden crear altres subdirectoris que no estan definits per defecte però que el mateix *SDK* d'Android ofereix. A continuació una breu descripció de quins són i per a què es poden usar:

- *menu:* Quan es vol afegir un menú a la vista s'ha de crear un fitxer *menu.xml* amb el contingut desitjat pel menú.
- *xml:* Quan es vol crear l'opció de preferències en l'aplicació, s'ha d'afegir el fitxer *preferencies.xml* amb el contingut que es vol oferir a l'usuari.

- *assets:* realitza la mateixa funció que el directori *res*. La diferència és que el contingut d'aquest directori no és indexat per la classe *R.java*. Quan es vol obtenir un element d'aquest directori s' haurà d'especificar la ruta relativa i el nom dels fitxer/s.

A part d'aquests 4 directoris hi ha altres elements que també cal destacar:

- *Google APIs:* És el mapa que facilita Android mitjançant Google Maps. Quan es crea el projecte és necessari que s'indiqui la versió que nosaltres desitgem i que inclogui l'opció Google API. El seleccionar-ho s'importen automàticament les llibreries necessàries per incorporar Google Maps en el projecte.
- *AndroidManifest.xml:* Aquest fitxer és el punt cèntric de l'aplicació. És on s'invoquen totes les *Activities* que hi ha en el projecte. És on s'indica quina és l'*Activity* que s'executarà quan l'aplicació sigui llançada. A partir d'aquest, també es pot determinar quins permisos són necessaris per accedir a la informació que requerim, per exemple, accedir a la localització de l'usuari, poder escriure a la tarja *SD*, etc.
- *default.properties:* Aquest fitxer conté informació a nivell general de l'aplicació. Per exemple, quina *target* s'ha fet servir. En el *Racó Mobile*, s'ha usat: *Google Inc.:Google APIs:8.*
- *proguard.cfg:* En aquest fitxer és on es defineix tota la codificació de seguretat que s'aplicarà a l'arxiu *.apk* (executable i instal·lador de l'aplicació). L'objectiu és evitar que una tercera persona que obtingui l'arxiu pugui descodificar i modificar el codi inicial. Més endavant, en la Secció 4.5 es pot trobar una explicació més detallada sobre el sistema *Proguard.*

Arribats aquest punt, ja es té un coneixement avançat d'un projecte Android. Hauríem de ser capaços de poder crear un projecte i saber on situar i com consultar els diferents elements que en l'aplicació pugui haver-hi.

A continuació, es presenta la descripció de com són les vistes en Android. Com es veurà es realitzen mitjançant fitxers *XML*, un fet diferencial respecte als llenguatges de programació que es pot està més habituats a treballar, com podria ser: Java, C, PHP, etc.

4.2. VISTA EN XML

En Android, aquest és un dels aspectes que pot arribar a desconcertar més al desenvolupador. El principal motiu, és que els elements que l'usuari veu en l'aplicació estan definits en fitxers *XML*. Els elements que s'afegeixen als *XML* són buits d'informació, ja que és el controlador qui s'encarrega d'omplir-los d'informació i de capturar els events que en aquella vista puguin succeir. Tots els elements que es poden afegir en una vista són proporcionats per les llibreries d'Android. Afegint a l'inici de cada *XML*, concretament, a l'etiqueta més externa de tots els elements especificats en el disseny, la següent línia de codi:

```
xmlns:android="http://schemas.android.com/apk/res/android"
```

Amb aquesta acció s'està important les eines que Android proporciona per poder crear, *Layouts*, *TextView*, *MapView*, etc. Un altra fet a destacar, és que degut a l'alt nivell d'abstracció que s'arriba la quantitat de vistes a dissenyar en *XML* pot ser mínima. Aquest fet, facilita l'acabament de l'aplicació en referència a aplicació del disseny.

En la *Figura 47* es pot veure un exemple d'una vista. El primer element que s'observa, en aquest cas és un *LinearLayout*. Primer de tot afegeix la línia que importa les llibreries i en segon lloc es defineix els paràmetres de les dimensions que ocuparà en la vista. En el seu interior, es pot veure que hi haurà dos elements principals: un element *ProgressBar* i una *ListView*. El primer servirà per mostrar informació a l'usuari mentre s'està obtenint la informació, i el segon, serà una llista que es mostrarà un cop la informació hagi estat obtinguda.

En la descripció anterior, en cap moment s'ha mencionat cap controlador, aquest fet, és degut a que la vista de la *Figura 47* es pot aplicar a qualsevol vista que requereixi una llista i una imatge de carregant.

En quan a l'aplicació del disseny es refereix, es pot veure certs fragments de codi que mostren que aplicar el disseny pot ser molt simple. Per exemple, el fragment: "android:background=*"@color/llistaInici"* permet, mitjançant un color definit en el directori *values/colors.xml*, canviar el color de fons quan es mostra el *ProgressBar*. Tots els controladors que importin aquesta vista ja tenen definit el seu color de fons amb aquesta acció.

Mitjançant el codi Java també es poden crear els elements que es defineixen en els *XML's*, no obstant, s'està perdent reusabilitat. Tal i com s'ha explicat en els anteriors paràgrafs, definir-los en els *XML's* té l'avantatge de crear vistes abstractes i que puguin ser reaprofitades per varis controladors.

Per acabar aquest apartat, destacar que al llarg del punt s'ha fet èmfasi en la importància d'utilitzar els mateixos components. Ara bé, per tal de modificar el contingut de cada llista, és a dir, què hi ha a l'interior de cada fila s'han de crear els mètodes propis i vistes perquè es mostrin d'una forma més personalitzada. Aquest punt, doncs, és el que es veurà a continuació.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent" android:id="@+id/vistes_generals_raco">
    <RelativeLayout android:id="@+id/layoutCarregantDades"
        android:background="@color/llistaInici" android:layout_height="fill_parent"
        android:layout_width="fill_parent">
        <ProgressBar android:layout_height="wrap_content"
                android:layout_width="wrap_content"
android:layout_centerInParent="true"
                android:id="@+id/carregantDades" android:visibility="gone"
                style="@android:style/Widget.ProgressBar.Inverse"></ProgressBar>
    </RelativeLayout>
    <ListView android:id="@+id/vista_llista_raco "
android:layout_height="wrap_content"
        android:layout_width="fill_parent" android:scrollbars="none"
        android:background="@color/llistaInici" android:dividerHeight="2.0sp"
        android:divider="@color/gris" android:cacheColorHint="@color/llistaInici"/>
```

Figura 47. Vista definida en les llistes de la Zona del Racó

4.3. BASE ADAPTER

És la classe que utilitza Android per mostrar el contingut de les llistes. En un principi, només accepta tipus d'objectes bàsics, com per exemple, *Strings*. No obstant, heretant d'aquesta classe es poden implementar mètodes propis i adaptar-los perquè mostrin el tipus de dades que desitgem. També es pot adaptar cada ítem de la llista perquè tingui un aspecte diferent. A les *figures 48* i *49* es mostra a partir d'una sèrie d'imatges. En la *Figura 48* es mostra la informació mitjançant el *Base Adapter* que les llibreries d'Android proporcionen. En el codi que hi ha a la imatge, el què es fa és declarar un *ArrayAdapter* on el seu contingut serà únicament *Strings*. Dins la creació de l'*ArrayAdapter*, es pot veure com se li indica quina *View* s'aplicarà, que tal i com s'aprecia només és un *TextView*. Llavors se li passa una *ArrayList* de *Strings* que s'anomena dades que és qui conté la informació que es mostrarà.

En canvi, a la *Figura 49*, s'hereta de la classe *Base Adapter* i com es pot veure el disseny de la llista és personalitzat. En el codi es pot veure com es crea un adaptador nou, en aquest cas *AdaptadorAulesOcupacio*. Aquest serveix per mostrar les aules tal i com es poden veure en la imatge. En aquest cas, al crear un nou adaptador, se li passa el context de l'aplicació mitjançant el *this* i l'*Array* amb la informació de les aules. L'*ArrayList* es pot observar que és de tipus *Aula*.

En la *Figura 50* es pot veure una breu descripció de com s'implementa aquesta classe. A l'interior de cada funció es pot veure una descripció amb el què s'ha de fer per poder realitzar una llista personalitzada.

Un cop vista la plantilla anterior, deixarem de banda el tema de les vistes, i ens centrarem en una altra funcionalitat que aprofita el potencial que ofereix Android als seus desenvolupadors i usuaris. Així doncs, passem detallar la gestió de notificacions al *Racó Mobile.*

```
List.setAdapter(new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1,
dades));
```

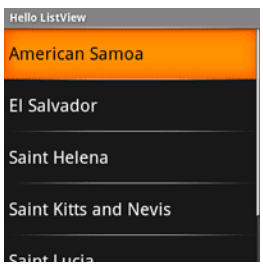

Figura 48. Insertar informació en una llista (ListView) amb tipus bàsics.

```
mListOcupacio.setAdapter(new AdaptadorAulesOcupacio(this, sAules));
```

Figura 49. Insertar informació en una llista (ListView) amb tipus propis.

```
public class CustomAdapter extends BaseAdapter {

	private Activity mActivity;
	private LayoutInflater mInflater;
	private ArrayList<?> mLitems;

	public CustomAdapter(Activity a, ArrayList<?> it) {
		mActivity = a;
		mLitems = it;
		mInflater = (LayoutInflater) mActivity
			.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
	}

	public static class VistaH {
	//TODO Crear els elements que contindrà la vista
	}

	@Override
	public int getCount() {
		// TODO retornar el tamany de la llista mLitems
		return 0;
	}

	@Override
	public Object getItem(int position) {
		// TODO retornar al posicó de l'Item
		return null;
	}

	@Override
	public long getItemId(int position) {
		// TODO  retornar al posicó de l'Item
		return 0;
	}

	@Override
	public View getView(int position, View convertView, ViewGroup
	parent) {
		// TODO Assignar la vista i insertar la informació
		return null;
	}
}
```

Figura 50. Plantilla per realizar un propi adaptador de llistes

4.4. SISTEMA DE NOTIFICACIONS

El sistema de notificacions permet informar als usuaris mitjançant el mòbil que s'ha produït una acció externa a ells. En Android, les notificacions es poden generar de dues formes diferents:

1. Les pot generar el propi telèfon, per exemple, indicar que una aplicació està activa.
2. Les pot generar un element extern per informar, per exemple, que té un nou avís del Racó per consultar.

En el *Racó Mobile* com es veurà més endavant s'han treballat els dos mètodes.

Abans d'entrar en detall de com s'ha implementat en l'aplicació, s'ha cregut convenient realitzar una descripció del procés de les notificacions provinents de l'exterior del telèfon. Així, s'adquirirà una visió de com funciona tot aquest sistema i quan s'entri en detalls més tècnics es facilitarà el seu seguiment.

Comencem primer de tot per veure una visió global de com es realitza el procés de registrar l'aplicació i el procés d'enviar una notificació al telèfon. En les *Figures 51* i *52* es pot veure una descripció gràfica del procés complet.

En tot moment, els actors principals són tres: el telèfon, la part del servidor client i el sistema que ofereix Google, el *C2dm*[18] [24].

El *C2dm* és el servei que ofereix Google als desenvolupadors per tal de poder realitzar d'una manera fàcil i homogènia el sistema de notificacions. Actualment, aquest servei està en fase beta.

Les principals característiques del *C2dm* són:

- Permet a altres servidors d'aplicacions enviar breus missatges a les seves aplicacions Android. El servei de missatgeria no està dissenyat per enviar una gran quantitat de contingut a través dels missatges. El seu ús més freqüent és utilitzar-lo per indicar a l'aplicació que hi ha noves dades al servidor, de manera que l'aplicació pugui notificar-ho a l'usuari i si és el cas mostrar-ho.
- El *C2dm* no ofereix cap garantia sobre el correcte enviament o l'ordre dels missatges. Així, per exemple, es pot utilitzar aquest sistema per a una aplicació de missatgeria instantània per notificar a l'usuari que té missatges nous, però no és recomanable utilitzar-ho pels missatges que s'envien constantment.
- Una aplicació en un dispositiu Android no necessita estar en execució per rebre els missatges. El sistema desperta l'aplicació a través de *Intent Broadcast* quan el missatge arriba. Sempre i quan, l'aplicació està configurada amb el receptor del *Intent* que arriba i amb els permisos de recepció apropiats.
- No es proporciona cap interfície d'usuari integrada ni gestors per controlar les dades del missatge. *C2dm* simplement passa les dades del missatge rebut directament a l'aplicació, que té el control total de com tractar la situació. Per exemple, l'aplicació pot enviar un avís, mostrar una interfície d'usuari personalitzada, o realitzar una sincronització de dades.

[18] Android Cloud to Device Messaging. http://code.google.com/intl/ca/android/c2dm/

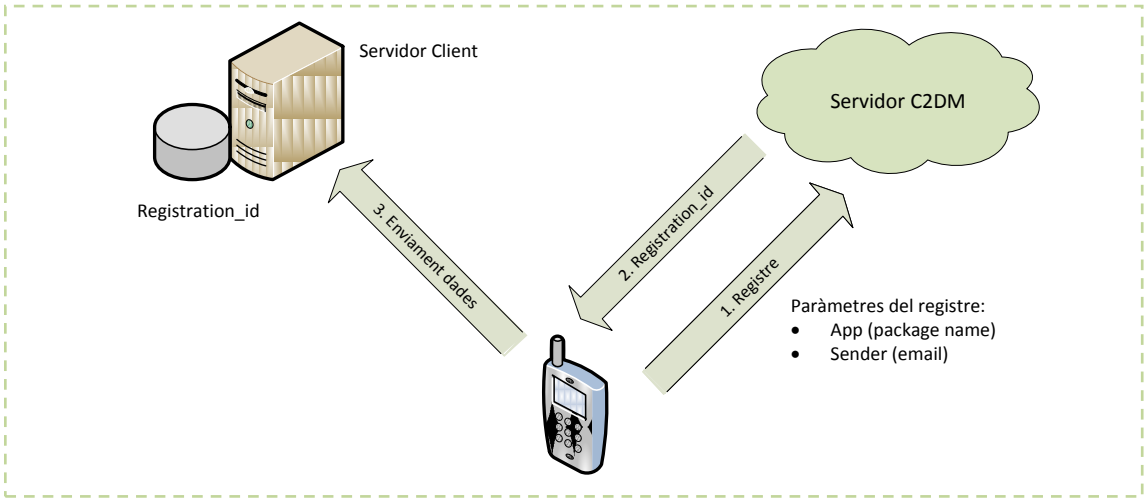

Figura 51. Registre al C2DM i servidor

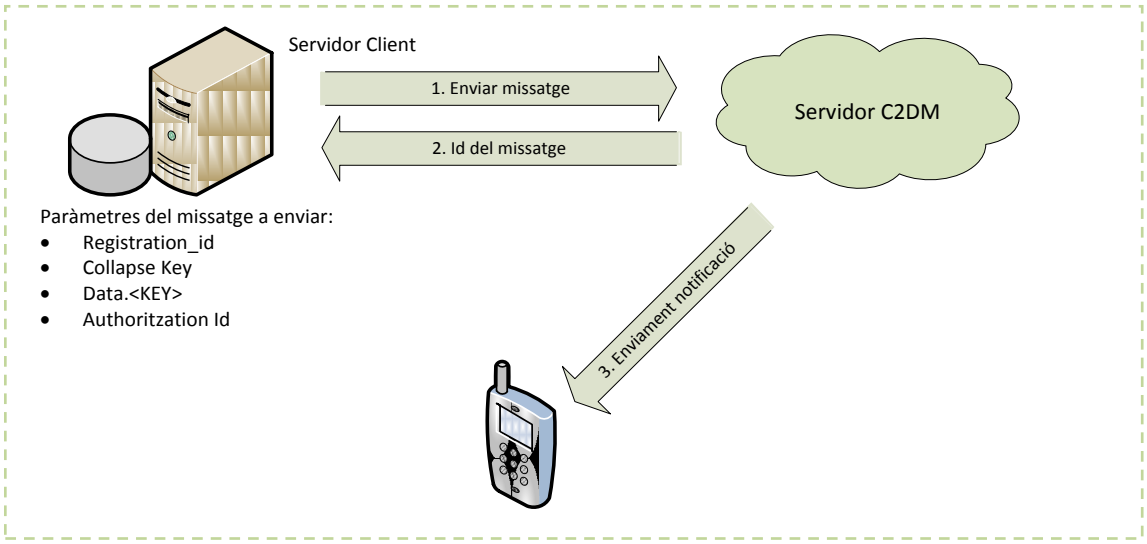

Figura 52. Enviament d'una notificació

- Es requereix que els dispositius tinguin Android 2.2 o superior i que també han de tenir l'aplicació *Android Market* instal·lada. Per usar el servei no és necessari tenir l'aplicació publicada al *Market*.
- Utilitza una connexió existent per als serveis de Google. Amb lo qual requereix que els usuaris tinguin configurat el seu compte de Google en els seus dispositius mòbils.

En quan al servidor client, és propi dels desenvolupadors de l'aplicació. El servidor ha de ser un servidor que funcioni per HTTPS que compleixi amb els següents criteris:

- Capaç de comunicar-se amb el seu client.
- Capaç de llançar les peticions HTTP al servidor *C2dm*.
- Capaç de manejar les sol·licituds i les dades que hi ha en cua, segons sigui necessari.
- Capaç d'emmagatzemar l'autenticació *ClientLogin*, és a dir, el *token* i *l'ID* de registre de clients. L'autenticació de *ClientLogin* s'inclou en la capçalera de les peticions POST que envien missatges.
  Per tal d'utilitzar el servidor *C2dm*, prèviament s'ha hagut de donar d'alta el servidor al servei. Per tal de portar-ho a terme, només s'ha de crear una compte de Google i omplir

un formulari que des de la mateixa *pàgina web*[19] on es consulta el funcionament del *C2dm* es pot accedir.

Un cop descrits els principals actors, passem a descriure cada una de les imatges anteriors i explicant quina és la successió de fets que hi ha en cada una d'elles.

- Comencem pel **procés de registre**:

Quan es realitza l'aplicació s'escull el moment en què comença el procés. Com a cas genèric, aquest començarà en el moment en què l'usuari executi l'aplicació per primera vegada.

1. En la *Figura 51* es pot veure com el primer de tot és comunicar-se amb el servei de Google. Per tal de realitzar-ho s'envia el *package* de l'aplicació, el qual és adquirit via el fitxer *Manifest.xml* que hi ha en el projecte. També s'adjunta el correu amb què s'ha registrat la part del servidor client.
2. Un cop rebudes les dades el servidor es crea un *registration_id*[20]. Quan la tercera part vulgui enviar una notificació al servidor, aquest sabrà l'aplicació i qui s'ha registrat per enviar-li. El *registration_id* és retornat al dispositiu i guardat.
3. Un cop guardat, s'envia al servidor client, que emmagatzemarà el *registration_id* en la seva base de dades per quan es vulgui enviar una nova notificació.

- Procés d'**enviament de notificació**:

1. Com es pot veure a la *Figura 52* el servidor client comença el procés. Primer de tot, s'estableix connexió amb el *C2dm*. Aquest comunica quin és el seu *Authoritzation_id*[21]. Un cop rebuda l'autenticació, el servidor client envia una petició adjuntant les següents dades:
   - *Registration_id*: per saber a qui va dirigida la notificació.
   - *Collapse Key*: s'utilitza per gestionar un grup de missatges, com quan el dispositiu està fora de línia, de manera que només l'últim missatge s'envia al client.
   - *Data.<KEY>*: les dades que volem enviar al dispositiu. No hi ha límit en el nombre de parells de clau / valor, encara que hi ha un límit en la mida total del missatge.
   - *Authoritzation_Id*: l'identificador que acabem de rebre.
2. El servei de Google, en el cas en què totes les dades són correctes envia el missatge i li retorna l'identificador de missatge amb què ha estat enviat als usuaris.
3. El *C2dm* enviar el missatge a el/s dispositiu/s que tinguin aquell servei activat.
4. El dispositiu ha d'estar preparat per capturar el missatge i realitzar les accions corresponents.

De forma genèrica, s'acaba d'explicar quins són els dos processos que succeeixen quan es gestionen les notificacions en Android. En el següent apartat, es veurà com s'ha integrat aquesta gestió en el *Racó Mobile*.

[19] *http://code.google.com/intl/ca/android/c2dm/*
[20] Identificador únic que és generat en el moment del registre de l'aplicació.
[21] Identificador únic que és generat en el moment d'establir connexió amb el C2dm.

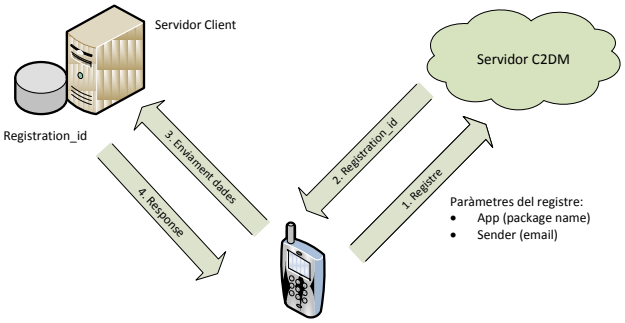

Figura 53. Registre al C2DM i servidor en el Racó Mobile

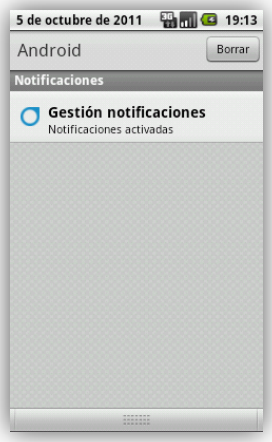

Figura 54. Auto-notificació que es mostra quan es rep resposta del servidor client

4.4.1. ADAPTACIÓ AL *RACÓ MOBILE*

Per tal d'implantar un sistema de notificacions en el projecte, es va partir de la base anteriorment explicada, no obstant, es van aplicar algunes modificacions, que es detallen a continuació.

S'ha dividit en dos parts, la primera d'elles és la gestió de les notificacions en el dispositiu i l'altra és com està realitzat el sistema en el servidor client, en aquest cas, el Racó.

SISTEMA DE NOTIFICACIONS AL DISPOSITIU

Tal i com s'ha comentat a l'inici del cas base, s'ha de decidir quan s'activaran les notificacions en l'aplicació. En aquest cas, s'ha optat per crear en l'apartat de les *Preferències* un *checkbox* el qual permet activar i desactivar el servei.

Pel què fa al procés de registre l'aplicació, la interacció amb el *C2dm* és la mateixa que s'ha explicat en el cas base. Per contra, la notificació del *registration_id* al servidor client s'ha modificat tal i com es pot veure a la *Figura 53*. A continuació trobem la descripció de com es realitza.

A l'igual que el cas base, el dispositiu envia un missatge al servidor client on li notifica el *registration_id*. S'ha afegit una resposta del servidor client on es retorna una resposta amb valor 200 o bé un 500, depenen de si el procés ha estat satisfactori o no respectivament. Un cop s'ha rebut la resposta, el mateix dispositiu genera una auto-notificació (el primer tipus que s'ha mencionat a l'inici de l'apartat) on s'informa de si ha anat correcte o no el procés. En la *Figura 54* es mostra la notificació que es genera.

Pel què fa el procés de d'entrada de les notificacions i les diferents accions a realitzar per mostrar a l'usuari, s'ha optat per les següents característiques:

- Si arriba una notificació es mostra la informació que arriba amb aquesta, és a dir, el contingut de *data.<KEY>*. En el cas en què hi hagi vàries notificacions a consultar es mostrar un text genèric on s'indica el nombre de notificacions pendents per llegir que té l'usuari.
- El to, vibració i volum de les notificacions, són per defecte els que l'usuari té configurats en el dispositiu.

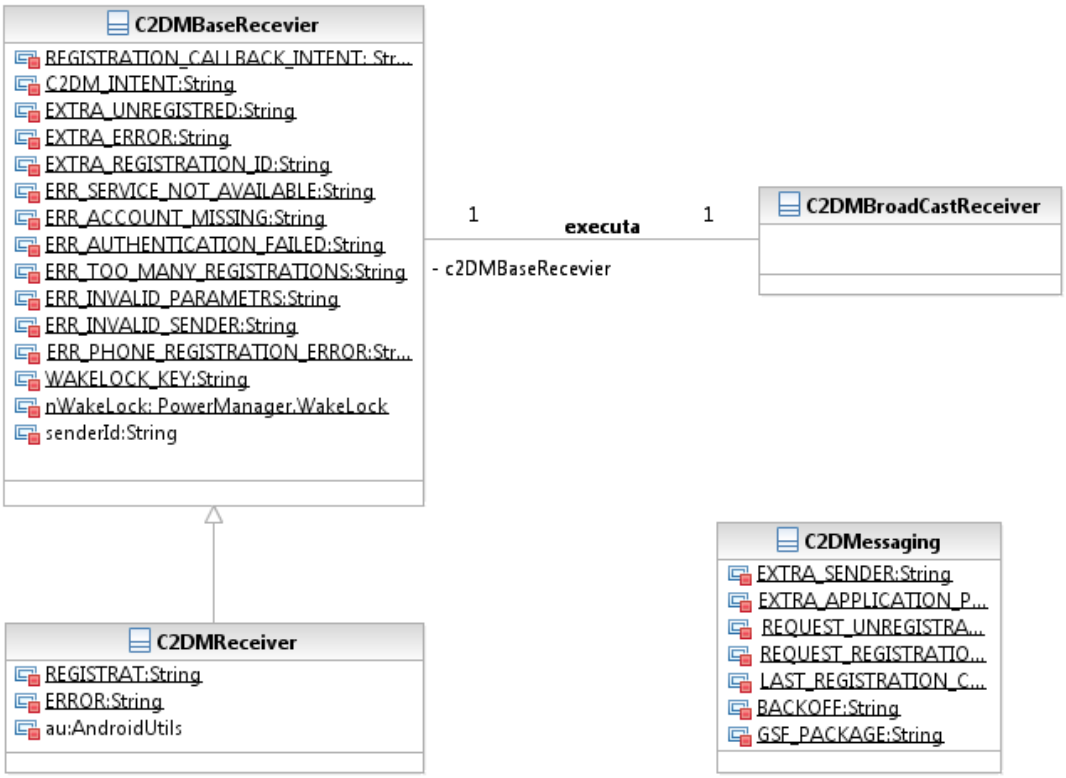

Figura 55. Diagrama sobre la gestió de les notificacions

- Quan l'usuari llegeix la notificació, s'obre l'aplicació en la vista inicial, és a dir, en la zona actualitat, on allà ja li apareixerà la informació de l'avís. Si es tracta d'una notificació de projecte final de carrera, s'obre el navegador del telèfon per tal de que l'usuari pugui consultar via web la informació.

En la *Figura 55* es presenta el diagrama de classes que s'ha obtingut al realitzar el procés de notificació.

S'ha cregut oportú mostrar el diagrama en particular perquè és interessant descriure què fa cadascuna d'aquestes classes en el procés.

- *C2DMBaseReceiver*: Aquesta classe és proporcionada per Google. És una classe abstracte la qual s'encarrega de gestionar l'estat de les notificacions, és a dir, gestiona el tipus d'acció que s'ha de realitzar: desregistrar, registrar o crear la notificació en la barra de notificacions. És en aquesta classe, on es gestiona l'estat de la bateria del telèfon. Quan arriba la notificació el telèfon deixa d'estar en estat ociós, i quan s'ha processat la notificació si s'escau s'ha de tornar a deixar el telèfon en l'estat en què estava. S'ha de dir que aquesta classe es recomana no editar-la i en cas de necessitat crear una altra classe que hereti d'aquesta i ho gestioni.

- *C2DMReceiver*: Aquesta classe realitza la tasca descrita anteriorment. Hereta els mètodes de registrar, desregistrar i d'error de la classe anteriorment descrita. Així es poden realitzar accions que personalitzades per a gestionar les notificacions.

- *C2DMBroadCastReceiver*: Aquesta classe s'activa quan el servidor *C2dm* envia la notificació. Captura la petició i executa el mètode definit en la classe *C2DMBaseReceiver*, que fa que es comenci a processar la notificació.

- *C2DMessaging*: Aquesta classe és cridada per la classe *PreferenciesUsuari*. La qual quan s'activa o es desactiva el *checkbox* de les notificacions, realitza una petició a una de les dues operacions: *register()* o *unregistered()*. És en aquest punt on comença tot el procés d'activació descrit anteriorment.

SISTEMA DE NOTIFICACIONS A LA PART SERVIDOR CLIENT

Per la part del servidor, respecte el cas base no hi ha hagut cap tipus de canvi. S'ha cregut interessant, però, explicar el procés d'enviament de notificacions.

Cada cinc minuts s'executa un *Thread* al servidor del Racó. Comprova les notificacions pendents d'enviar i les envia mitjançant el sistema de notificacions corresponent (segons les preferències de l'usuari). Això permet que l'usuari pugui rebre la mateixa notificació per correu electrònic (si hi ha una prèvia subscripció al servei) i a l'aplicació mòbil a la vegada.

Prèviament a la implementació, per tal de realitzar les proves inicials es va treballar amb un *Script* el qual podem veure a la *Figura 56*.

En el punt **A** de la Figura es realitza una petició al servidor *C2dm* per tal d'obtenir l'autorització de *Login*. Com es pot veure és una petició via *POST*, on es detalla: el tipus de compte amb què està registrat el servidor, el correu amb què hi ha el registre fet, la contrasenya, el tipus de servei que es sol·licita i el nom del *package* de l'aplicació i finalment, l'adreça on realitzar la petició. Amb les línies finals (`- |grep Auth |cut -b6-`) el què es fa és quedar-se únicament amb l'autorització de *Login* que el *C2dm* retorna.

En l'apartat **B** s'assigna de forma manual el *registration_id* que hi ha guardat en la base de dades del servidor client.

Per acabar, en l'apartat **C** es realitza l'enviament de la notificació. Com s'observa, es passa per paràmetre l'autorització de *Login* obtinguda, el *registration_id* de qui ha de rebre la notificació, el *data.payload* que conté el missatge que es mostrarà al rebre la notificació i finalment li passem *collapse_key* a 0 perquè ara mateix només enviï l'últim missatge.

```
A
TOKEN=`wget -q --post-data
"accountType=GOOGLE&Email=XXXXXXX@gmail.com&Passwd=XXXXXX&service=ac2dm&source=fib.lcfi
b.raco" https://www.google.com/accounts/ClientLogin -O - |grep Auth |cut -b6-`

B
echo "Enviant missatge a C2DM..."
KEY=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

C
curl --header "Authorization: GoogleLogin auth=$TOKEN" "https://android.apis.google.com/c2dm/send" -d
registration_id=$KEY -d "data.payload=Acabes de rebre una notificació!!" -d collapse_key=0
echo
```

Figura 56. Script per enviar les notificacions

Al llarg d'aquest punt, s'ha detallat quin és el sistema de notificacions utilitzat en Android. Un altre aspecte important al realitzar aplicacions és tenir en compte la possibilitat de que es pugui obtenir l'arxiu executable (*.apk*) de l'aplicació. A l'igual que moltes aplicacions és possible descompilar-lo i obtenir el codi original, amb lo qual, l'aplicació passa a ser vulnerable a modificacions. En el següent apartat, es menciona aquest fet, i què es pot fer per tal de codificar el codi.

4.5. EL PROGUARD

Una de les accions que es recomana realitzar un cop s'ha finalitzat una aplicació i es vol publicar perquè altres usuaris se la descarregui i la utilitzin, és protegir el codi amb què s'ha creat. En Android, l'arxiu executable que es genera té una extensió *.apk*. Aquest arxiu, és el que instal·larà cada usuari en el seu telèfon.

Tot i que està desenvolupat sobre un sistema *UNIX* els usuaris no disposen de suficients permisos per poder treballar com a usuaris privilegiats, també anomenat, *root*[22]. Aquest fet, provoca que usuaris que tenen un coneixement avançat amb l'àmbit informàtic realitzin el procés de *rooteig* del telèfon. Amb aquesta acció passen a tenir privilegis de *root*, amb lo qual, poden navegar pel sistema de fitxers del telèfon i manipular el directori on hi ha les aplicacions instal·lades. Al accedir-hi es poden obtenir els arxius *.apk* i manipular-los. Una alternativa al procés de *rooteig*, és tenir instal·lades aplicacions que ens permetin navegar pel sistema de fitxers.

El fet és que, si s'obté l'arxiu *.apk* es pot realitzar un procés d'enginyeria inversa i obtenir el codi amb què ha estat programat. Un cop obtingut el codi, es pot modificar i generar de nou un *.apk* manipulat. Per tal d'evitar, o si més no dificultar, aquest procés, Android ofereix als un arxiu anomenat *Proguard.cfg*. Com hem vist en la Secció 4.1 aquest fitxer es genera automàticament en el moment de crear el projecte. A continuació entrem més en detall sobre el *Proguard* [25].

Proguard és una eina que optimitza i ofusca el codi mitjançant l'eliminació de codi no utilitzat i canvia el nom a les classes, els camps i mètodes amb noms semànticament foscos. El resultat és un fitxer *.apk* de menor mida que dificulta la realització d'enginyeria inversa. Cal destacar, que és important utilitzar aquest mètode quan s'utilitzen funcions que són sensibles, en quan a seguretat fa referència, per exemple, quan s'estan utilitzant funcions amb llicència.

El *Proguard* només s'executa quan es crea una aplicació en mode de llançament, de manera que no s'ha de treballar amb el codi ofuscat quan es crea una aplicació en mode de depuració. Utilitzar *Proguard* és completament opcional, però altament recomanable.

Per habilitar el *Proguard* perquè s'executi en el moment de generar l'arxiu *.apk*, cal que s'estableixi la propietat *proguard.config* a l'arxiu *default.properties*. La ruta que s'indica pot ser una ruta absoluta o un camí relatiu a l'arrel del projecte.

Suposem que no s'ha modificat la ubicació de l'arxiu *proguard.cfg* i que es troba en la seva ubicació per defecte (directori arrel del projecte), es pot especificar la ubicació de la següent manera afegint la següent acció a l'arxiu *default.properties*:
"*proguard.config=proguard.cfg"*.

[22]És el nom convencional de la compte d'usuari que posseeix tots els permisos en tots els modes (mono i ulti) usuari.

Pel què fa al *Racó Mobile*, s'ha activat. No obstant, s'ha considerat que donat que és una aplicació consultora de dades, és a dir, no s'escriuen mai dades en el servidor les regles que ja hi ha definides en el fitxer per defecte eren suficients.

En els punts anteriors, s'han estat detallant punts que poden ser comuns a totes les aplicacions Android que es desenvolupin. En el següent apartat, es detallaran punts particulars sobre la implementació del *Racó Mobile*.

4.6. PUNTS A DESTACAR

El llarg d'aquest punt, es detallaran dos punts que s'ha cregut convenient destacar. En el primer d'ells s'explicarà quin és el flux d'informació, des de que l'usuari accedeix a la vista fins que se li mostra la informació. Aquest fet el comparteixen totes les vistes de l'aplicació, excepte la vista que fa referència a *Google Maps*. I en el segon punt, s'explicarà perquè la classe *BaseDeDadesManager* s'ha realitzat en una sola classe i no en vàries.

4.6.1. FLUX D'INFORMACIÓ

Els diferents controladors de l'aplicació tenen un punt en comú, aquest és, la manera en què s'obté la informació de la base de dades, del servidor i com es mostra. Tot aquest procés, a mesura que es van anar implementant les classes es va notar que era repetitiu per cada una d'elles. Fet que va provocar que els noms de les funcions que realitzaven aquestes tasques fossin en cada una d'elles iguales. Finalment, es va optar per crear una classe on tots els controladors l'heretessin. Així, a l'afegir un nou controlador ja es creaven els mètodes bàsics automàticament. En el diagrama de classes que s'ha presentat en els punts anteriors aquesta es s'anomena*: GestioActualitzacionsLlistesActivity*. En la *Figura 57* es veu el flux d'informació i quines funcions són invocades quan un usuari accedeix per primera vegada des de que s'ha llançat aquella vista.

Com es pot veure, el primer mètode que es crida és l'anomenat *onCreate()*. Aquest mètode és natiu d'Android i és l'encarregat d'inicialitzar el controlador. Seguidament s'obtenen les dades de la base de dades i es mostren a l'usuari. Paral·lelament, es realitza una crida al servidor per tal d'obtenir les dades actualitzades. Quan es retorna del procés d'obtenció de dades, la classe *GestióConnexió()* s'encarrega d'actualitzar les base de dades i posteriorment de mostrar les dades a l'usuari actualitzades.

Destacar que controladors com el gestor de l'Agenda, no actualitza sempre les dades, només en cas de que l'usuari ho forci. Aquest fet, és degut a que l'Agenda en principi té poques modificacions. Per tant, el flux anterior el realitzaria la primera vegada. En les futures accions el seu propi flux seria el que es mostra a la *Figura 58*.

Fins ara, s'ha parlat del primer accés des de que s'ha llançat la vista, però un cop llançada els següents accessos que es realitzen no s'accedeix via el mètode *onCreate()*, sinó que és via *onResume()*. En el *Racó Mobile*, s'aprofita aquesta acció per mostrar directament la informació guardada en la base de dades. En la *Figura 59* es pot veure el flux d'informació.

Un cop vist el diagrama de flux, anem a veure ara com s'ha estructurat la Base de dades de l'aplicació, la qual com s'ha vist anteriorment accedim a cada vista per tal de mostrar la informació a l'usuari.

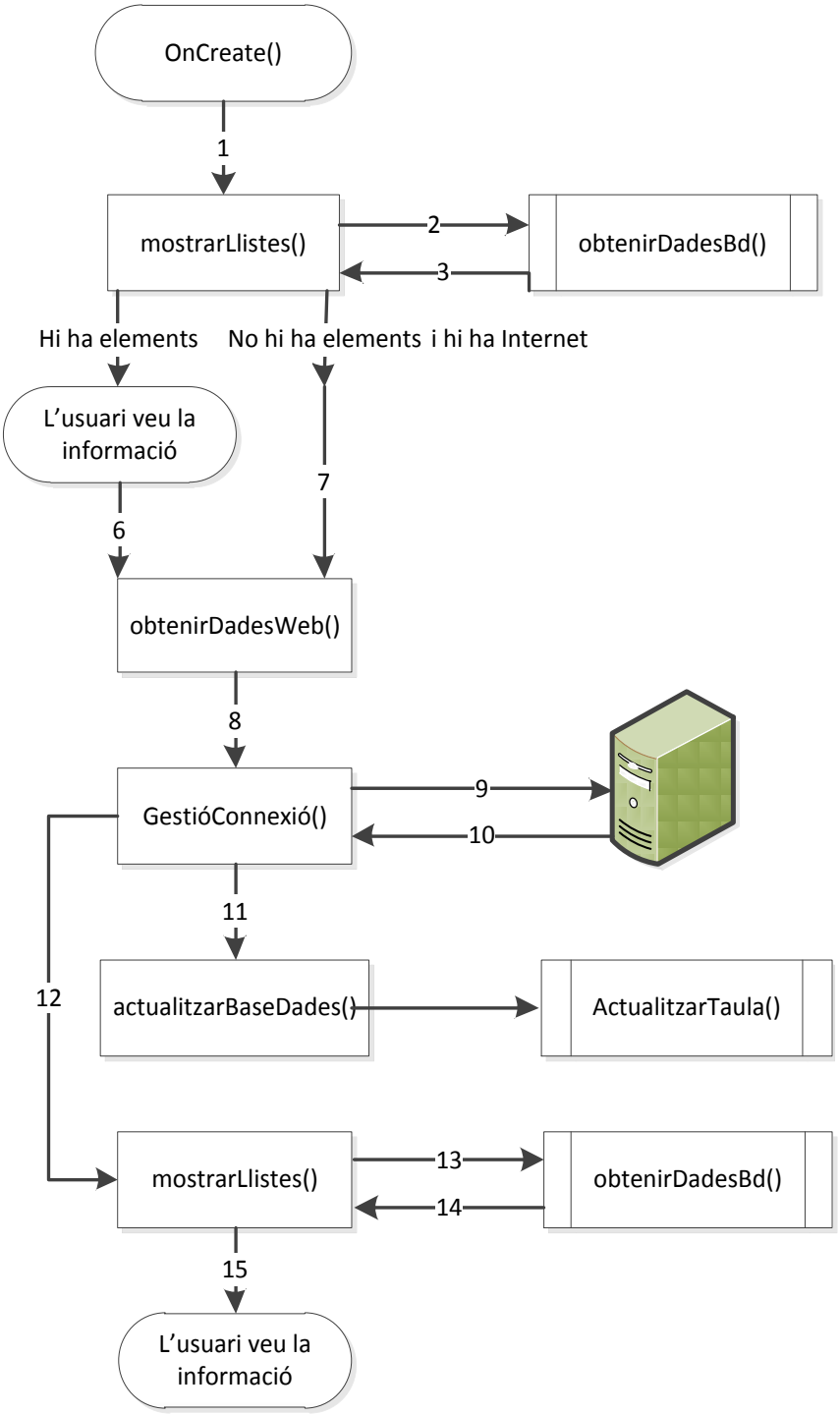

Figura 57. Diagrama de flux al accedir per primera cop a la vista

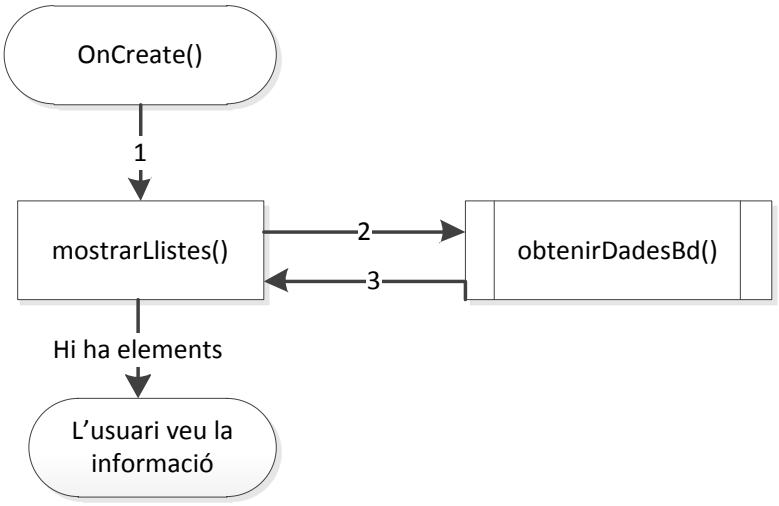

Figura 58. Diagrama de flux al accedir per primera cop a la vista agenda

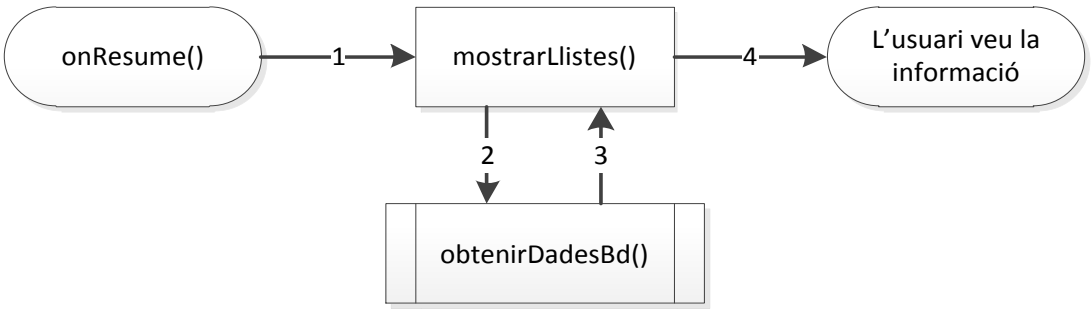

Figura 59. Diagrama de flux al accedir per segona vegada a la vista

4.6.2. BASE DE DADES

Per tal de realitzar la persistència en l'aplicació es va optar per crear una base de dades mitjançant *SQLite*. Android proveeix de varis sistemes alhora de crear la persistència de dades, per exemple, per emmagatzemar dades amb *clau-valor* es recomana usar *SharedPreferences*, el qual, ja s'ha explicat anteriorment. Alhora d'usar persistència privada d'una gran quantitat de dades, recomana usar *SQLite*.

SQLite és una base de dades de codi obert que suporta les característiques estàndards de base de dades relacional, com la sintaxi SQL, transaccions i declaracions. A més, requereix de poca memòria en temps d'execució (aprox. *250 KByte*).

SQLite en Android no requereix cap configuració o administració de bases de dades. S'especifica l'*SQL* per treballar amb la base de dades i aquesta s'administra automàticament.

En Android, quan es crea base de dades es guarda en el telèfon. Concretament, en el directori *"data/dades/APP_NAME/bases_de_dades/FILENAME"*.

- *data*: és el camí que *Environment.getDataDirectory()* retorna.
- *APP_NAME*: és el nom de l'aplicació.
- *FILENAME*: és el nom que se li assigna a la base de dades durant la seva creació.

SQLiteOpenHelper és la classe la qual s'hereta per poder gestionar i accedir la base de dades. Per exemple, ofereix els mètodes *getReadableDatabase()* que permet obtenir la base de

dades només de lectura. El mètode *getWriteableDatabase()* per poder llegir i escriure la base de dades.

Per a la clau principal de la base de dades sempre s'ha d'utilitzar l'identificador *_id*. Algunes de les funcions d'Android es basen en aquesta norma. El *SQLiteDatabase* ofereix mètodes com, *insert(), update(), delete()* i el mètode *execSQL()* que permet executar directament *SQL*. L'objecte *ContentValues* permet definir claus/valors d'inserció i actualització. La clau és el nom de la columna i el valor és el valor d'aquesta. Es pot aprofitar *ContentValues* per tal de realitzar la inserció d'elements.

Les consultes es poden crear a través del mètode *rawQuery()* que accepta *SQL* o mitjançant el mètode *consulta()* que proporciona una interfície per especificar les dades dinàmiques o mitjançant el *SQLiteQueryBuilder*. El *SQLiteBuilder* és similar a la interfície d'un proveïdor de continguts per la qual cosa se sol utilitzar en *ContentProviders*. Una consulta retorna sempre un *Cursor*. Per obtenir el nombre d'elements podem utilitzar el mètode *getCount()*. Per moure'ns entre les files de dades individualment, podem utilitzar el mètode *MoveToFirst()* i *MoveToNext()*. A través del mètode *isAfterLast()* es pot comprovar si encara hi ha algunes dades.

El fet de que en l'aplicació s'hagi escollit realitzar persistència de dades és perquè s'ha cregut que la principal diferència entra una aplicació web i una aplicació nativa és el fet de la persistència de dades i la no necessitat de disposar de connexió a Internet per poder realitzar consultes.

Alhora d'implementar-ho es va tenir un problema. La idea inicial era crear una classe per a cada taula que es necessités, és a dir, tenir un apartat en el projecte anomenat Base de Dades i en el seu interior posar-hi totes les classes. El problema va aparèixer quan ens vam adonar que el *DataBaseHelper* i *SQLiteDatabase* havia de ser sempre invocat cap a la mateixa classe i que en el seu interior hi havia d'haver-hi tots els mètodes gestors de la base de dades. Es va intentar, també tenir varis fitxers amb el mateix nom de la base de dades, no obstant tampoc funcionava, ja que l'accés produïda errors constantment. D'aquest fet, s'extreu l'explicació de perquè s'ha implementat tota la base de dades en la mateixa classe.

Capítol 5

TEST DE L'APLICACIÓ

L'aplicació al llarg de la primera setmana de setembre ja estava en fase beta. Es va instal·lar a cinc estudiants perquè durant un mes i mig realitzessin proves de funcionament i usabilitat de l'aplicació. Al llarg d'aquest temps, es van anar rebent diferents peticions i opinions, els quals han ajudat a millorar l'aplicació i fer-la més robusta, fiable i sobretot, usable. Els estudiants són treballadors de Laboratori de Càlcul amb lo qual, el nivell d'exigència era més elevat.

Durant el tram final se'ls va realitzar una enquesta per tal de veure si s'havien assolit els objectius inicials del projecte. A continuació es presenten les conclusions finals de l'enquesta. Per veure més informació sobre l'enquesta es pot consultar l'Annex II.

EFICIÈNCIA

La navegació entre pantalles, sobretot els primers cops que s'obre l'aplicació pot ser lenta. Aquest fet és degut a que en cada vista es consulta informació al servidor. La majoria de les dades que s'obtenen del servidor són fitxers *JSON* i *XML*. Aquest tipus de fitxers fan parseig de la informació més àgil, per contra l'*ICAL* és més costós. No obstant, només s'utilitza en l'agenda i l'horari i com s'ha vist s'obtenen de la base de dades, un cop s'han obtingut la dades per primera vegada. Pels resultats de les enquestes es pot veure que les dades obtingudes del servidor són bastant ràpides, ja que les satisfacció és òbvia per part dels 5 enquestats.

La càrrega de la informació des de la base de dades és ràpida ja que 4 dels 5 enquestats han atorgat la màxima nota. Com s'ha pogut veure anteriorment, l'accés a la base de dades en temps d'execució ocupa màxim 250 *KBytes* amb lo qual és una bona opció fer-ne ús.

Analitzats els resultats anteriors, es pot donar per assolit el requisit de l'eficiència en l'aplicació.

FIABILITAT

Com es pot veure en els resultats de les enquestes l'aplicació durant un mes i mig ha tingut un màxim de 4 parades forçades. Aquestes foren a l'inici de la prova ja que era quan més *bugs* es van reportar, per la falta de testeig previ.

L'aplicació al llarg d'aquest mes i mig ha experimentat grans canvis a nivell de fiabilitat. Pràcticament cada setmana s'anaven corregint errors reportats. Així doncs, gràcies a tots els problemes que s'han trobat, ara mateix, l'aplicació ja és fiable. Afegir que passat el mes i mig l'aplicació continua funcionant i no s'ha rebut cap notificació de parada forçada.

Val a dir, que al mercat hi ha molts dispositius Android diferents amb lo qual, pot ser que l'aplicació en algun moment s'hagi de veure sotmesa a certs canvis segons el grau d'adaptació.

ACTUALITAT

A nivell d'aspecte s'ha intentat emular el màxim el disseny de la facultat, per exemple en els colors. També, es pot veure que en el desenvolupament de l'aplicació s'han intentat aprofitar el màxim els elements que Android proporciona ja que així, s'assegura el seu correcte funcionament en qualsevol versió. En els resultats de l'enquesta es pot veure que l'usuari està satisfet amb l'aplicació.

La valoració global és molt positiva ja que l'usuari es mostra satisfet amb el disseny i els serveis oferts que estan basats amb els que ofereix Android. El *Banner*, per exemple, intenta aprofitar elements del Racó, així com les llistes els diferents colors que en el seu interior ofereix.

TESTABILITAT

En la *Figura 77* de l'Annex II, es pot veure que la participació ha estat activa per part dels usuaris. A part de *bugs* també es van reportar millores que han ajudat a que l'aplicació acabi tenint una alta estabilitat i fiabilitat.

El fet d'evitar a l'usuari haver de posar-se en contacte amb el desenvolupador per comunicar-li errors, és positiu. L'automatització d'aquest procés és un avantatge, deixant de banda, tota la informació que subministra, la qual es detalla després a l'apartat final d'aquest punt en la *Figura 108*.

Aquesta funcionalitat s'ha assolit amb èxit ja que els usuaris que provaven l'aplicació van aportar noves idees, com per exemple, l'ordre de les pestanyes en què es mostrava la informació a la Zona Racó, o la idea d'enviar un correu als professors mitjançant la vista d'informació de les assignatures.

Un altra factor molt important, fou el nombre d'errors que es van trobar i la seva importància, com per exemple, la càrrega de dades quan no hi ha Internet i sense haver-hi informació a guardada. Un de molt important que es va reportar fou, que la vista de *Google Maps* consumia fins a un 93% de la bateria en ús del sistema, fet que el mateix dia ja va quedar solucionat.

USABILITAT I SATISFACCIÓ

Com es pot veure en les dues primeres figures d'aquest apartat en l'Annex II, l'ús de l'aplicació ha estat elevat. Es garanteix que el nombre d'errors reportats ha estat fiable. Es Considera que usar l'aplicació prop de 3 vegades al dia és un nombre elevat, ja que al estar a la FIB mateix, la disponibilitat d'estar davant d'un ordinador és elevada amb la implicació de tenir el Racó en pantalla.

En les *Figures 81* i *82*, es pot observar un aspecte important de l'aplicació. El fet és que les imatges que hi ha en les pestanyes i la lletra és intuïtiva. A les llistes també s'ha incorporat una imatge amb una fletxa per indicar que era accessible.

En les *Figures 83, 84* i *85* es pot veure, la confirmació de que la nostra aplicació és usable i que satisfà els requisits que s'havien marcat a l'inici. Si es consulta la descripció dels casos d'ús es pot observar que com a requisits no especials es destaca sempre que la lletra s'ha de poder veure a distància i que els botons i pestanyes han de ser lo suficientment grans perquè l'usuari pugui accedir-hi sense problemes.

A continuació es presenten tres preguntes que es van realitzar als cinc usuaris per tal de que després d'aquest mes de proves donessin la seva visió funcional i de futur sobre l'aplicació. Aquí tenim les respostes:

1. **Quines 3 funcionalitats has utilitzat més?**
   - Ocupació de les aules, les notificacions i mirar la informació de les assignatures.
   - Veure el meu horari, veure els avisos de les assignatures i pestanya actualitat
   - Veure el meu horari, pestanya actualitat i veure la ocupació de les aules.
   - Assignatures matriculades, l'ocupació de les aules i el calendari.
   - Veure el meu horari, els avisos d'assignatura, el meu horari.

Com es pot veure les funcionalitats més visitades són: el meu horari, l'ocupació de les aules i els avisos de les assignatures matriculades. És obvi que hagin sortit aquestes degut a que es va instal·lar a l'inici del curs quan els estudiants encara no tenen del tot clar a quines aules han d'assistir. Una de les funcions principals de l'aplicació és la de consultar els avisos, amb lo qual s'esperava també la seva aparició. L'ocupació de les aules sí que és un fet curiós, no obstant, val a dir, que és molt pràctic poder veure a quina aula es pot anar en un moment determinat.

2. **Afegiries/Eliminaries alguna funcionalitat?**
   - Buscar si hi ha algun company connectat alguna aula.
   - Poder descarregar els arxius adjunts dels avisos.
   - Poder veure una llista amb les últimes notificacions que han arribat.
   - Treure el correu de la FIB .

Les tres primeres respostes es poden incloure a la llista de futures ampliacions del projecte. Pel què fa a l'últim punt, la informació que es pot obtenir és tota la que es mostra. Potser en un futur es podria valorar opcions per obtenir més informació.

3. **Què t'agradaria que l'aplicació incorporés en un futur?**
   - Un *widget* per a l'escriptori amb les últimes notificacions.
   - Un altre *widget* que et digui quina classe et toca en aquest moment (assignatura i aula),
   - Una versió del fòrum reduïda (encara que sigui només per a llegir els posts).

- Llegir els adjunts directament des de l'aplicació i no via web.
- La gestió de projectes finals de carrera.
- La possibilitat de descarregar els fitxers adjunts d'un avís des de l'aplicació i la gestió de pràctiques (descarregar lliuraments i similars).

Les respostes de la pregunta anterior es poden considerar una futura ampliació del projecte. Les respostes són funcionalitats que mentre realitzaven les proves els van mancar. És d'important rellevància tenir aquestes idees en compte de cara a futures ampliacions.

Per acabar aquest apartat destacar una llibreria que s'ha fet servir i que actualment està incorporada en l'aplicació. Aquesta, s'anomena *BugSense* [27].

El seu ús ha servit per evitar que l'usuari cada vegada que l'aplicació se li tanqués inesperadament s'hagués de posar en contacte amb el desenvolupador. *BugSense* automàticament i totalment transparent a ell envia un correu informant de l'error. Aquesta ha estat molt útil ja que la informació subministrada en el correu informa exactament de: quin ha estat l'error, en quina classe i com a informació addicional: la marca del telèfon i la versió Android que porta instal·lada, entre d'altres. A continuació presentem una imatge d'exemple de sortida d'error.

ERROR MESSAGE

java.lang.RuntimeException: Unable to start activity ComponentInfo{fib.lcfib.raco/fib.lcfib.raco.Controladors.TabIniApp}: java.lang.RuntimeException: Unable to start activity ComponentInfo{fib.lcfib.r...

WHERE

ActivityThread.java:278

SHORT STACKTRACE

java.lang.RuntimeException: Unable to start activity ComponentInfo{fib.lcfib.raco/fib.lcfib.raco.Controladors.TabIniApp}: java.lang.RuntimeException: Unable to start activity ComponentInfo{fib.lcfib.raco/fib.lcfib.raco.Controladors.ControladorVistaResumEvents}: android.database.sqlite.SQLiteException: no such column: llegits: , while compiling: SELECT _id, titol, descripcio, imatge, dataPub, tipus, llegits, no_llegits FROM correus 1 at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:2787) 2 at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:2803) 3 at android.app.ActivityThread.access$2300(ActivityThread.java:135) 4 at android.app.ActivityThread$H.handleMessage(ActivityThread.java:2136) 5 at android.os.Handler.dispatchMessage(Handler.java:99)

Affected App Versions

1.0

Affected SDKs

2.2

Details (last occurence of error)

OS	Android 2.2
App Version	1.0
Device	HTC Desire
Country	ES
wifi_on	true
appname	fib.lcfib.raco
mobile_net_on	false
gps_on	not available [permissions]
screen_dpi(x:y)	254.0:254.0
screen:width	480
screen:orientation	normal
screen:height	800

Figura 60. Missatge d'error enviat per la llibreria BugSense

Capítol 6

CONCLUSIONS I TREBALL FUTUR

Un cop l'aplicació ja ha estat finalitzada, testejada i aprovada per les diferents parts (director del projecte i el Laboratori de Càlcul), s'han extret les conclusions finals del projecte.

Per tal de descriure-les en més detall s'ha cregut convenient realitzar-ho mitjançant diferents punts. Els punts amb què s'ha estructurat han estat, en primer lloc els objectius acadèmics i del projecte (veure Secció 6.1 i 6.2 respectivament). En segon lloc, es detallen futures ampliacions o un possible camí per continuar el projecte (veure Secció 6.3). En tercer lloc, es detalla la conclusió a nivell personal (veure Secció 6.4). Seguidament s'ha afegit un últim punt que s'ha cregut convenient destacar com annex a les conclusions. El cost del projecte (veure Secció 6.5) .

Comencem doncs amb el primer punt, on s'ha fet referència a les conclusions arribades a nivell acadèmic.

6.1. OBJECTIUS ACADÈMICS

En la introducció del projecte s'han proposat varis objectius a nivell acadèmic. Com per exemple, aplicar les metodologies que s'han ensenyat a diferents assignatures de la universitat. Un cop finalitzada l'aplicació i la documentació es pot afirmar que tots els objectius plantejats han estat assolits perfectament.

Tal i com s'ha comentat a l'inici, durant la carrera, en les diferents assignatures que s'han realitzat s'han anat adquirint diferents coneixements. El projecte, ha estat una primera prova, i tal i com es pot comprovar al llarg de la memòria i aplicació s'han pogut aplicar amb èxit. Així doncs, s'ha pogut negociar una aplicació amb el client, analitzar, dissenyar, implementar i documentar. Cada fase, ha requerit de certa cerca d'informació extra, ja que, en alguns casos ha estat necessari refrescar el coneixement i en d'altres s'ha tingut la necessitat d'ampliar-lo. A part, al llarg de la implementació, s'han pogut aplicar diferents patrons de disseny (veure Secció 1.3). Aquests han estat explicat a assignatures de la carrera on la seva aplicació havia estat a nivell de pseudocodi o teòrica.

Des d'un punt de vista personal, es valora tot l'esforç acadèmic i de sacrifici que s'ha hagut de fer per arribar a realitzar el projecte de final de carrera. Aquest ha servit per adonar-se de tots els coneixements i informació que la carrera li ha aportat.

No obstant, per ser crítics respecte amb treball que s'ha realitzat, destacar que la planificació que es va preveure del projecte en un primer moment fou molt optimista. Aquest fet ha comportat que s'ha hagut d'anar adaptant a noves reestructuració. El motiu principal d'aquest fet, fou que el tractament de *bugs* de l'aplicació no es va tenir suficientment en compte alhora de valorar la durada del tram final del projecte. Amb lo qual, l'últim mes s'ha hagut de treballar paral·lel, realitzant documentació i aplicació.

Passem ara a detallar els objectius del projecte, els quals aniran guiats, sobretot, pels requisits que en l'anàlisi s'han detallat.

6.2. OBJECTIUS DEL PROJECTE

L'avaluació dels objectius del projecte, s'ha basat en dos punts principals: com ha estat finalitzada l'aplicació i quin és el grau de satisfacció del client.

En la Secció 3.2 s'ha especificat cada cas d'ús i quin requisit assolia. Donat que s'han implementat i complert tots els casos d'ús que s'havien pactat amb el client, es pot donar aquest objectiu per assolit. Tal i com s'ha detallat a la *Figura 64*.

Pel què fa a la satisfacció del client, també és un objectiu assolit, ja que, l'aplicació serà distribuïda mitjançant l'*Android Market*. Si el client vol penjar l'aplicació perquè estigui disponible per a tots els usuaris és perquè es mostra satisfet. Un dels fets que ha pogut facilitar la seva satisfacció, és degut, al contacte constant que s'ha tingut. Ha pogut estar en tot moment, al corrent de quin era l'estat de l'aplicació, tan visual com de desenvolupament. Aquest fet ha facilitat molt el desenvolupament ja que el prototipatge es redueix en nombre d'iteracions.

Un cop descrits els principals objectius que han pogut repercutir en el resultat final de projecte, passem a detallar possibles camins que el projecte pot tenir en un futur.

#Cas dús	Cas d'ús	Implementat	No Implementat
1	Notícies de la FIB		
2	Situació de la FIB		
3	Assignatures del Grau		
4	Login		
5	Events conjunts		
6	Preferències		
7	Agenda		
8	Horari personal acadèmic		
9	Ocupació de les aules		
10	Assignatures cursant		
11	Correu		
12	Notificacions		
13	About		

Figura 61. Quadre resum dels casos d'ús assolits

6.3. POSSIBLES AMPLIACIONS

Les ampliacions que s'han definit a continuació, són possibles idees que han sorgit mentre es realitzava la implementació del projecte, o observant les necessitats de les persones que han estat realitzant el testeig de l'aplicació.

Abans de començar a detallar-les val la pena fer referència al Racó i totes les seves funcionalitats. El Racó, està preparat per suportar funcionalitats per a professors, estudiants, personal PAS, operadors i tota l'estructura de govern (secretaria, cap d'estudis, etc.) amb lo qual, si ens basem en aquest fet les ampliacions poden abastar una previsió molt gran. Amb lo què, s'ha decidit presentar unes ampliacions basades en l'aplicació que s'ha realitzat i que en poc temps puguin ser factibles.

Un apunt inicial que es vol destacar és que tal i com es pot veure en les imatges de l'aplicació, aquesta ha estat implementada mitjançant pestanyes. Aquest fet, fa que alhora d'ampliar segons quins àmbits, l'aplicació passi a ser poc usable per a l'usuari. Per exemple, si ampliem la zona FIB no hi ha problema perquè encara es poden afegir més pestanyes. Per contra, si la que volem ampliar és la zona Racó, un mateix recomana reemplaçar les pestanyes actuals per una llista on es mostrin les funcionalitats actuals més les que es volen afegir, o també, es podria fer mitjançant una graella de botons.

Dit això, una possible ampliació que es podria realitzar seria incloure una vista on es mostressin els diferents fils del fòrum del Racó. Aquest fet es podria realitzar, amb tres vistes addicionals. La primera d'elles podria mostrar una llista o botons dels diferents fòrums que hi ha. Per exemple, el general, taulell d'anuncis, etc. i un cop s'accedeixi a un element de la llista mostrar els diferents temes que hi ha oberts. Posteriorment, en l'última vista mostrar tot el fil d'aquell tema.

Una altra possible ampliació podria ser el fet de poder consultar qui està ocupant cada ordinador de les aules. És a dir, en la vista on es mostra el mapa de l'ocupació de les aules afegir la informació: PC – alumne , situada a la part inferior de la imatge.

Recordar també les ampliacions que han proposat els estudiants que provaven l'aplicació. Aquestes estaven definides en el Capítol 5.

Finalment com a última ampliació, la migració de l'aplicació a dispositius *Tablets*. Aquesta s'ha detallat a continuació.

6.3.1. EL PAS DE SMARTPHONES A TABLETS

En Android, la migració d'una aplicació nativa per a *Smartphones* a *Tablets*, es fa mitjançant un classe anomenada *Fragment*. Un *Fragment*[23] representa un comportament o una part de la interfície d'usuari en una *Activity*. Es poden combinar múltiples fragments en una sola *Activity* per construir una interfície d'usuari de diversos panells o reutilitzar un fragment en múltiples *Activities*. Un fragment pot ser equivalent a una secció modular d'una *Activity*, que realitza el seu cicle de vida, rep els seus propis esdeveniments d'entrada, i que es pot afegir o treure mentre l'*Activity* està en marxa. En la *Figura 62* s'ha presentat un esquema comparatiu entre una llista usada en Smartphones i una llista en *Tablet*.

En un *Tablet* s'unifica tot en una mateixa vista, i en un *Smartphone*, degut al seu tamany es realitza en dos. L'aplicació podria tenir les mateixes funcionalitats que disposa actualment. L'únic que canviaria seria l'estructura visual. La pantalla és més gran, amb lo qual, el contingut a mostrar no pot ser només una llista perquè no s'estaria aprofitant tot l'espai de que es disposa. Tal i com es pot veure a la *Figura 66* la navegació podria continuar essent l'actual, és a dir, mitjançant pestanyes. Aquest fet seria possible si a la part superior o inferior s'implementés una barra de navegació amb els continguts que té l'aplicació per a Smartphones.

Per tal de fer més visual aquest procés i no tan abstracte s'ha implementat una possible vista inicial. Com es pot veure a la *Figura 66* s'aprofita tota la pantalla per mostrar-hi tota la informació de que es disposa. En aquest cas, es pot distingir les notícies de la FIB a la part esquerre de la imatge i a la dreta la descripció de la notícia que s'està consultant. Així doncs, s'ha aconseguit en una sola vista el que actualment s'ha realitzat en dues.

[23] Podem consultar més información a: *http://developer.android.com/guide/topics/fundamentals/fragments.html*

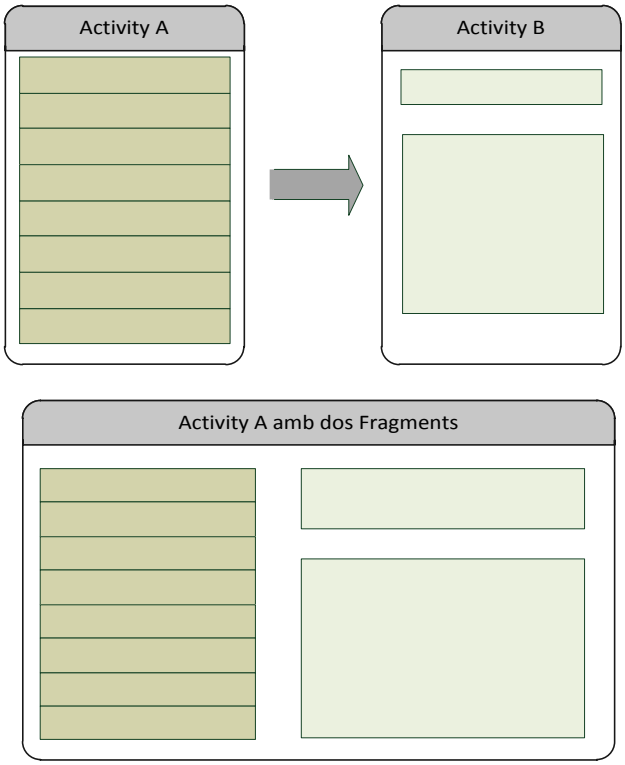

Figura 62. Part superior – Smartphone. Part inferior – Tablet

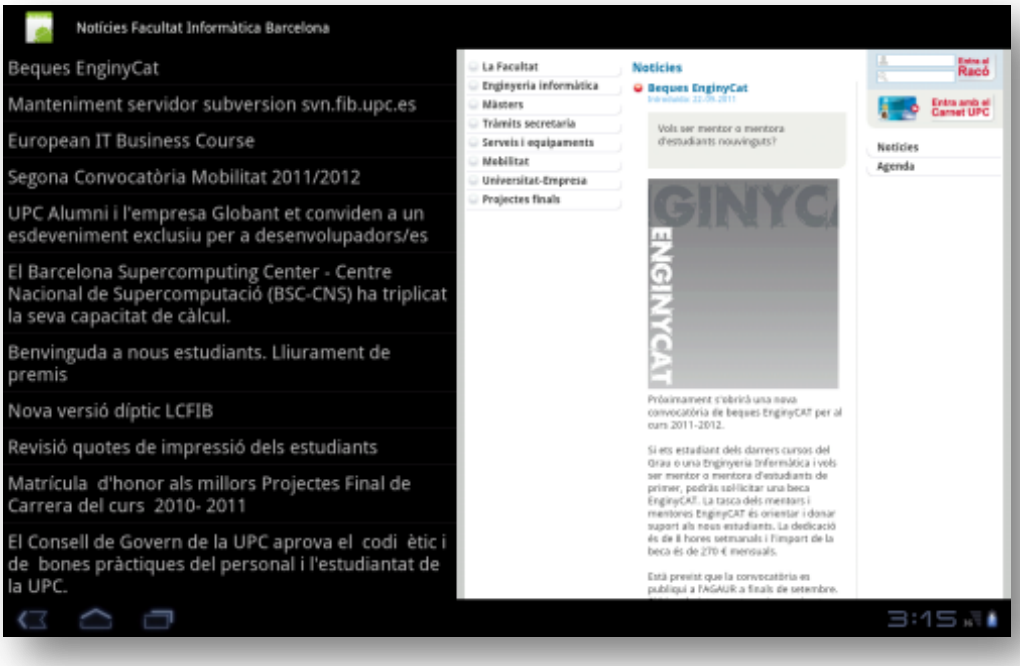

Figura 63. Vista incial des d'un Tablet

6.4. CONCLUSIÓ PERSONAL

Deixant de banda, els objectius i les ampliacions, a nivell personal ha estat una gran experiència, tan positiva com negativa. Un mateix es mostra satisfet del resultat final que s'ha obtingut. Portar a terme un projecte d'aquest abast des del seu inici fins al final ha estat una

prova per veure de les capacitats adquirides. S'ha après a nivell organitzatiu, didàctic i de programació a aplicar molts mètodes que al llarg de la carrera s'havien explicat.

Poder realitzar un projecte per la universitat, juntament amb el LCFIB també és una gran satisfacció. Ajudar a evolucionar la universitat la qual t'ha donat una formació és, per un mateix, gratificant.

A nivell de planificació d'un projecte, ha estat un bon exemple per veure que tot el que es mencionava era cert. Per exemple, quan es diu que és molt complicat entregar en les dates planificades perquè sempre surt algun imprevist, es volen afegir extres al projecte, etc. es pot afirmar que és totalment encertat. Un dels pròxims objectius serà que amb el temps i a mesura que es vagin realitzant més projectes es puguin anar afinant més els períodes de realització del projecte.

Per acabar, afegir que aprendre a desenvolupar per a un sistema operatiu com Android que està entrant fort en el mercat, és un gran pas per a nivell personal. Gràcies a la realització del projecte s'ha aconseguit entrar en el món laboral i treballar del que a un mateix li agrada.

Juntament amb la planificació s'adhereix un cost de producció i desenvolupament. El *Racó Mobile* tot hi haver estat realitzat com a projecte final de carrera, podria haver estat desenvolupat per alguna empresa externa i per tant, hagués tingut de ser remunerat. Així doncs, en el següent apartat es presenta el cost del projecte *Racó Mobile*.

6.5. Cost del Projecte

Com tot projecte, s'han de valorar els costos de la feina realitzada. En aquest cas, al tractar-se del projecte final de carrera les dades que es presenten a continuació no s'han executat. No obstant, creiem que és important tenir present el cost de la feina realitzada al llarg d'aquests nou mesos.

El projecte ha estat portat a terme per un analista i un programador. Pel què fa a l'analista es poden veure les diferents tasques que ha realitzat en la *Figura 64.* En resum la seva participació ha estat realitzar l'anàlisi de requisits, l'especificació, la planificació i la documentació final del projecte. Pel què fa al programador la seva tasca, la qual es pot consultar en la *Figura 65*, ha estat implementar l'aplicació i realitzar el testeig d'aquesta.

El *Racó Mobile* serà publicat a l'*Android Market*, fet que afegeix un cost al projecte de 27€. Finalment, un cop vist les diferents tasques realitzades per les dues persones implicades en el projecte. En la *Figura 66* presentem la valoració global del projecte i podem observar com el projecte té un cost total de **36.228€.**

Tasca	Nombre d'hores
Primera reunió: Anàlisi de requisits	16h
Previ a iteració 1: disseny casos d'ús	80h
Modelatge Iteració 1	16h
Segona reunió: Concretar requisits	16h
Modelatge dels casos d'ús (versió 2)	40h
Acabament dels casos d'ús	40h
Documentació final	80h

Figura 64. Treball realitzat per l'analista

Tasca	Nombre d'hores
Desenvolupament versions iteració 0	**512h**
Aprenentatge d'Android	40h
Versió amb menús	324h
Versió webview	24h
Versió amb pestanyes	124h
Desenvolupament versió final	**320h**
Realització de les funcionalitats	280h
Aplicació del disseny	40h

Figura 65. Treball realitzat pel programador

Perfil	Nombre d'hores	Preu/hora (€)	Total (€)
Analista	288h	45	12.960€
Programador	831h	28	23.268€
Cost extra	-	-	27€
TOTAL			**36.228€**

Figura 66. Resum del cost total del projecte

En la *Figura 67,* es pot veure la planificació del projecte que fou presentada en el document previ que es va entregar el 29 de juliol de 2011. Com es pot apreciar, la finalització del projecte estava prevista pel dia 31 d'octubre de l'any actual. La demora ha estat de dos mesos. Cal remarcar que l'aplicació es va acabar en el temps preestablert. No obstant, les diferents revisions tant de la documentació com de l'aplicació que s'han anat realitzant han contribuït a aquesta demora.

Al llarg del projecte i les conclusions s'ha parlat del fet de pujar l'aplicació a l'*Android Market.* En la secció que segueix, s'explica quins són els passos i què s'ha de fer per pujar l'aplicació i que estigui disponible per als usuaris.

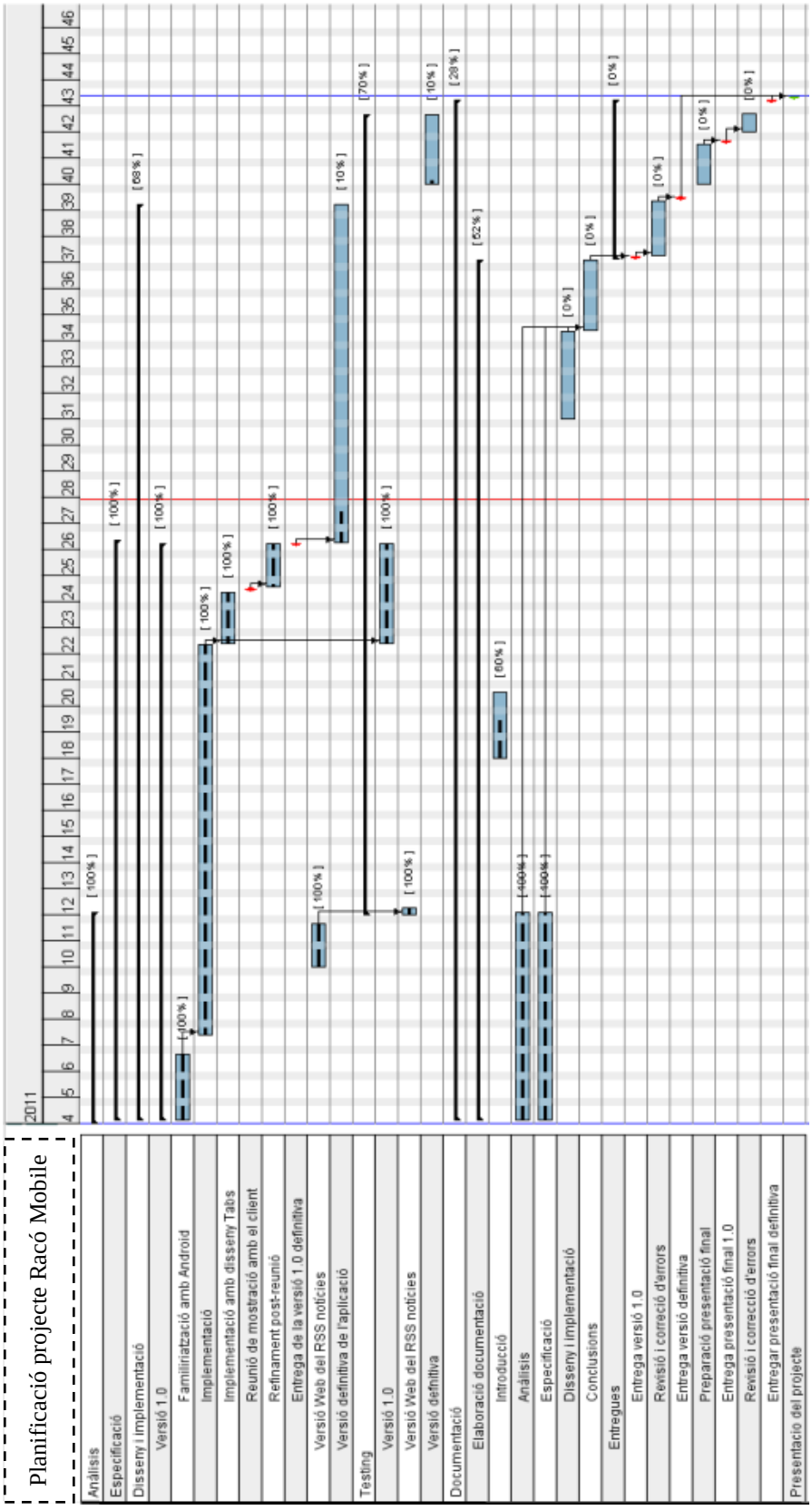

Figura 67. Planificació inicial del projecte

BIBLIOGRAFIA

LLIBRES

[1] – *Android in Practice*.
Autor: Charlie Collins, Michael Galpin, Matthias Käppler
Edició: 2012

[2] – *Hello, Android: Introducing Google's Mobile Development Platform*.
Autors: Susannah Davidson Pfalzer, Steve Peter.
Edició: Tercera de 2010

[3] – *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*.
Autor: Craig Larman
Edició: Tercera 2004

[4] – *Enginyeria del software. Especificació: Especificació de sistemes orientats a objectes amb la notació UML*.
Autor: Dolors Costal, Xavier Franch, M.Ribera Sancho i Ernest Teniente
Edició: Tercera de 2005

ONLINE

[5] Web oficial d'Android developers.
http://developer.android.com/

[6] Apple Inc. iPhone.
http://www.apple.com/iphone

[7] Blackberry. App World
http://es.blackberry.com/services/appworld

[8] Microsoft. Windows Mobile.
http://www.microsoft.com/windowsmobile/en-us/

[9] Wikipedia, Android History
http://en.wikipedia.org/wiki/Android_(operating_system)

[10] Android Developers, Platform Versions
http://developer.android.com/sdk/android-version_number.html

[11] Android Developers, Relative number of active Android devices
http://developer.android.com/resources/dashboard/platform-versions.html

[12] Open Handset Alliance. Open Handset Alliance.
http://www.openhandsetalliance.com.

[13] Canonical Ltd. Ubuntu Linux.
http://www.ubuntu.com

[14] Java
http://java.com

[15] Eclipse
http://www.eclipse.org/

[16] Android Developers, Emulator
http://developer.android.com/guide/developing/tools/emulator.html

[17] Fabrice Bellard. QEMU.
http://bellard.org/qemu/

[18] Google Maps API
http://code.google.com/intl/ca/apis/maps/signup.html

[19] Arquitectura 3 capas
http://es.wikipedia.org/wiki/Programación_por_capas

[20] OCL
http://www.lri.fr/~wolff/teach-material/2008-09/IFIPS-VnV/UML2.0OCL-specification.pdf

[21] W3C R. Extensible Markup Language (XML).
http://www.w3.org/XML/

[22] iCAL, Calendar
http://en.wikipedia.org/wiki/ICalendar

[23] JacksonTreeModel – Constructing Trees from JSON content
http://wiki.fasterxml.com/JacksonTreeModel

[24] Push Notifications – C2dm
http://code.google.com/intl/ca/android/c2dm/
http://code.google.com/intl/ca/android/c2dm/signup.html

[25] Proguard in Android
http://developer.android.com/guide/developing/tools/proguard.html

[26] SQLiteDatabase
http://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html

[27] BugSense, Tracked Applications
http://www.bugsense.com/dashboard/project/6fad7199

[28] Android Market - Publish
https://market.android.com/
http://developer.android.com/guide/publishing/publishing.html

ANNEX I: ESPECIFICACIÓ DELS CASOS D'ÚS

Cas d'ús 1: Procés de consulta ràpida d'events
Àmbit
Aplicació per Smartphones que usen el sistema operatiu Android 2.2 o superior. Pantalla inicial de l'aplicació.
Nivell
Principal: Procés de visualització de la llista. Secundari: Veure la informació particular de cada ítem.
Actor/s primari
Estudiants. Externs.
Stakeholders i altres
- *Estudiants*: Vol realitzar una consulta de les notícies, correu o dels avisos de les assignatures de les qual està matriculat. Aquests ítems s'han de guardar al telèfon per poder-los consultar en cas que no es disposi de connexió Internet. - *Externs*: Vol realitzar una consulta de les notícies de la facultat. Aquest ítem s'ha de guardar al telèfon per poder-lo consultar en cas que no es disposi de connexió Internet. - *Servidor de dades:* Vol rebre una petició de les dades, mitjançant una URL vàlida i un protocol adequat. Vol servir les dades correctament i amb la màxima eficiència possible.
Precondició
Els actors estudiants poden estar registrats a l'aplicació de tal manera que se'ls mostrarà els avisos i els correus. Altrament , tindran els mateixos privilegis que un actor extern.
Garanties d'èxit
Les notícies, els avisos i el correu són mostrats per l'aplicació (referent a l'usuari en particular) i l'usuari les pots consultar. Les dades es guarden al dispositiu.
Escenari principal
1. L'usuari posa en funcionament l'aplicació. 2. L'aplicació treballa amb "background". 2.1. Es connecta al servidor amb les URL's per tal d'obtenir la informació. 3. L'aplicació es connecta a la base de dades del mòbil. 4. Es mostra la llista amb les dades (segon l'usuari) que hi havia guardades. 5. Les dades del punt 2 arriben, s'ordenen per data, i es mostren a l'usuari. 6. S'actualitza la base de dades amb nova informació. 7. L'usuari surt de l'aplicació o continua navegant per aquesta.
Extensions
a. No es poden obtenir les dades del servidor o no es disposa de connexió: 1. L'usuari posa en funcionament l'aplicació.

2. La connexió al servidor falla, es notifica a l'usuari que hi ha hagut un error al actualitzar les dades.
3. L'usuari pot intentar actualitzar la informació o consultar les dades que hi ha guardades a la base de dades.

b. Base de dades no creada: (inici aplicació per primera vegada)
1. Es demana la informació al servidor.
2. La base de dades no està creada.
3. Es crea la base de dades.
4. S'actualitza amb la informació que s'ha obtingut del servidor i es mostra al usuari.
5. L'usuari posa en funcionament l'aplicació.

c. L'aplicació falla inesperadament:
1. L'usuari posa en funcionament l'aplicació.
2. L'aplicació s'atura i mostra un missatge d'error.
3. L'usuari pot reiniciar de nou l'aplicació.

d. L'usuari vol que se li mostrin els avisos i correu:
1. L'usuari posa en funcionament l'aplicació.
2. EL primer ítem que es mostra és el de configuració d'avisos i correu.
3. L'usuari accedeix i comença el cas d'ús de Login.

e. L'usuari consulta informació d'algun ítem.
1. L'usuari veu la llista d'ítems.
2. Prem sobre un ítem.
3. Es mostra una nova vista amb la informació particular del ítem.
4. L'usuari prem el botó enrere (del telèfon) per tornar a la llista anterior o abandona l'aplicació.

f. L'usuari actualitza la informació.
1. L'usuari veu la llista d'ítems.
2. Prem sobre "menú"
3. Escull la opció "actualitzar"
4. El sistema, es connecta de nou al servidor.
5. Obté les dades necessàries.
6. Actualitza les base de dades
7. Mostra la nova informació a l'usuari

Requisits especials

- La llista ha de ser visible a 30-40cm de distància del telèfon.

- Els botons i pestanyes han de tenir una mida adequada perquè el seu ús no sigui dificultós.

- El servidor respon en menys de 8 segons.

- L'Idioma de l'aplicació és el que l'usuari té configurat en el telèfon.

- Cal connexió a Internet.

Requisit/s assolit

RF_5 – Events conjunts.

RF_1 – Notícies de la FIB.

RF_10 – Avisos de les assignatures que està cursant.

RF_11 – Correu.

Freqüència

Sempre que l'usuari obri l'aplicació.

Cas d'ús 2: Localització de la Facultat Informàtica
Àmbit Aplicació per Smartphones que usen el sistema operatiu Android 2.2 o superior. Pantalla situada a l'apartat FIB.
Nivell Principal: L'usuari pot veure mitjançant "Google Maps" la situació de la facultat i la informació de contacte.
Actor/s primari Estudiants i futurs estudiants. Externs.
Stakeholders i altres - *Nous estudiants i externs*: Vol consultar on és la facultat o posar-se en contacte telefònicament amb aquesta.
Precondició Cal connexió a Internet.
Garanties d'èxit L'usuari podrà veure, mitjançant un Pin[24] on és la Facultat d'Informàtica.
Escenari principal 1. L'usuari posa en funcionament l'aplicació. 2. Accedeix a l'opció del menú inferior anomenada "FIB". 3. Accedeix a l'opció de localització. 4. El mapa es mostra en pantalla i també la informació de la FIB (telèfon i direcció). 5. L'usuari pot navegar pel mapa. 6. L'usuari surt de l'aplicació o continua navegant per aquesta.
Extensions 2. No es pot mostrar el mapa o no es té connexió a Internet: 1. L'usuari posa en funcionament l'aplicació. 2. La connexió al servidor falla o no es detecta la connexió. Es notifica a l'usuari que hi ha hagut un error al actualitzar les dades. 3. L'aplicació falla inesperadament: 1. L'usuari posa en funcionament l'aplicació. 2. L'aplicació s'atura i mostra un missatge d'error. 3. L'usuari pot reiniciar de nou l'aplicació.
Requisits especials - El Pin ha de ser suficientment gran i visible a 30-40cm de distància. - Els botons i pestanyes han de tenir una mida adequada perquè el seu ús no sigui dificultós. - El telèfon i la direcció han de ser visibles a 30-40cm. - Cal connexió a Internet. - L'Idioma de l'aplicació és el que l'usuari té configurat en el telèfon.
Requisit/s assolit RF_2 – Situació.
Tecnologia i llista de variació de dades

[24] Imatge situada en un punt en el mapa per tal de facilitar a l'usuari la localització del lloc desitjat.

- Les visualització del mapa depèn de les dades subministrades pel servidor de Google Maps.
Freqüència
Poc freqüent durant l'any, excepte els mesos de matriculació de nous estudiants, o events que s'organitzin a la FIB.

Cas d'ús 3: Consultar les assignatures de la facultat
Àmbit
Aplicació per Smartphones que usen el sistema operatiu Android 2.2 o superior. Pantalla situada a l'apartat FIB.
Nivell
Principal: Procés de visualització que permet als estudiants veure la informació de les assignatures que s'imparteixen a la facultat. Secundari: Veure la informació particular de cada assignatura.
Actor/s primari
Estudiants i futurs estudiants.
Stakeholders i altres
- *Estudiants i nous estudiants*: Volen realitzar una consulta de les assignatures que s'imparteixen a la facultat. Aquestes es guardaran al telèfon per poder-les consultar en cas que no es disposi de connexió a Internet.
Precondició
En aquest cas no hi ha cap precondició.
Garanties d'èxit
Les assignatures es mostren a l'usuari i pot ser consultada la seva informació. Es pot realitzar una cerca per ID de l'assignatura. Les dades són guardades al dispositiu.
Escenari principal
1. L'usuari posa en funcionament l'aplicació. 2. Accedeix a l'opció del menú inferior anomenada "FIB". 3. Accedeix a l'opció de localització. 4. L'aplicació es connecta a la base de dades del telèfon. 5. Es mostra la llista amb les dades que hi havia guardades. 6. Les dades del punt 4 arriben i es mostren a l'usuari. 7. S'actualitza la base de dades amb la nova informació. 8. L'usuari pot realitzar una cerca mitjançant el ID de l'assignatura o realitzar "scroll" per la mateixa pantalla i buscar l'assignatura. 9. L'usuari surt de l'aplicació o continua navegant per aquesta.
Extensions
a. No es poden obtenir les dades del servidor o no es disposa de connexió: 1. L'usuari posa en funcionament l'aplicació. 2. La connexió al servidor falla, es notifica a l'usuari que hi ha hagut un error al actualitzar les dades. 3. L'usuari pot consultar les dades que hi ha guardades a la base de dades. b. Base de dades no creada: (a l'iniciar l'aplicació per primera vegada) 1. Es demana la informació al servidor.

2. La base de dades no està creada.
3. Es crea la base de dades.
4. S'actualitza amb la informació que s'ha obtingut del servidor i es mostra al usuari.
5. L'usuari posa en funcionament l'aplicació.

c. L'aplicació falla inesperadament:
1. L'usuari posa en funcionament l'aplicació.
2. L'aplicació s'atura i mostra un missatge d'error.
3. L'usuari pot reiniciar de nou l'aplicació.

d. L'usuari consulta informació d'alguna assignatura.
1. L'usuari veu la llista d'assignatures.
2. Prem sobre un ítem.
3. Es mostra una nova vista amb la informació particular del ítem.
4. L'usuari prem el botó enrere (del telèfon) per tornar a la llista anterior o abandona l'aplicació.

Requisits especials

- La llista ha de ser visible a 30-40cm de distància del telèfon.
- Els botons i pestanyes han de tenir una mida adequada perquè el seu ús no sigui dificultós.
- El servidor respon en menys de 8 segons de temps.
- L'idioma de l'aplicació és el que l'usuari té configurat en el telèfon.
- Cal connexió a Internet.
- L'idioma del teclat per interactuar amb la cerca serà el que l'usuari tingui configurat en el telèfon.

Requisit/s assolit

RF_3 – Assignatures de la FIB.

Freqüència

Poc freqüent, excepte els inicis de quadrimestre (consultes ràpides de les informacions) o en períodes de matrícula.

Cas d'ús 4: Consultar les notícies de la facultat

Àmbit

Aplicació per Smartphones que usen el sistema operatiu Android 2.2 o superior.
Pantalla situada a l'apartat FIB.

Nivell

Principal: Procés de visualització que permet als estudiants, externs i nous estudiants, veure informació de les notícies que publica la facultat.
Secundari: Veure la informació particular de cada notícia.

Actor/s primari

Estudiants i futurs estudiants.
Externs.

Stakeholders i altres

- *Estudiants, nous estudiants i externs*: Volen realitzar una consulta de les notícies que es publiquen a la facultat. Aquestes es guardaran al telèfon per poder-les consultar en cas que no

es disposi de connexió a Internet.

Precondició

En aquest cas no hi ha cap precondició.

Garanties d'èxit

Les notícies es mostren a l'usuari i pot ser consultada la seva informació.

Les dades són guardades al dispositiu.

Escenari principal

1. L'usuari posa en funcionament l'aplicació.
2. Accedeix a l'opció del menú inferior anomenada "FIB".
3. Accedeix a l'opció de notícies.
4. L'aplicació es connecta a la base de dades del mòbil.
5. Es mostra la llista amb les dades que hi havia guardades.
6. Les dades del pas4 arriben i es mostren a l'usuari.
7. S'actualitza la base de dades amb la nova informació i es mostra a l'usuari.
8. L'usuari surt de l'aplicació o continua navegant per aquesta.

Extensions

a. No es poden obtenir les dades del servidor o no es disposa de connexió:
    1. L'usuari posa en funcionament l'aplicació.
    2. La connexió al servidor falla, es notifica a l'usuari que hi ha hagut un error al actualitzar les dades.
    3. L'usuari pot consultar les dades que hi ha guardades a la base de dades.

b. Base de dades no creada: (inici aplicació per primera vegada)
    1. Es demana la informació al servidor.
    2. La base de dades no està creada.
    3. Es crea la base de dades.
    4. S'actualitza amb la informació que s'ha obtingut del servidor i es mostra al usuari.
    5. L'usuari posa en funcionament l'aplicació.

c. L'aplicació falla inesperadament:
    1. L'usuari posa en funcionament l'aplicació.
    2. L'aplicació s'atura i mostra un missatge d'error.
    3. L'usuari pot reiniciar de nou l'aplicació.

d. L'usuari consulta informació d'alguna notícia.
    1. L'usuari veu la llista de les notícies.
    2. Prem sobre un ítem.
    3. Es mostra una nova vista amb la informació particular del ítem.
    4. L'usuari prem el botó enrere (del telèfon) per tornar a la llista anterior o abandona l'aplicació.

Requisits especials

- La llista ha de ser visible a 30-40cm de distància del telèfon.

- El servidor respon en menys de 8 segons de temps.

- L'idioma de l'aplicació és el que l'usuari té configurat en el telèfon.

- Cal connexió a Internet.

Requisit/s assolit

RF_1 – Notícies de la FIB.

Freqüència

Poc freqüent.

Cas d'ús 5: Login
Àmbit Aplicació per Smartphones que usen el sistema operatiu Android 2.2 o superior. Pantalla situada a la pantalla principal (botó configuració de la llista) o a l'entrar al Racó.
Nivell Principal: Procés que permet realitzar la personalització de l'aplicació mitjançant la introducció de username i password a l'aplicació.
Actor/s primari Estudiants.
Stakeholders i altres - *Estudiants*: Usuari introduirà els seu username i password al sistema. Aquestes es guardaran al telèfon per poder-les consultar en cas que siguin necessàries en events futurs.
Precondició L'estudiant té un usuari assignat per la Facultat d'Informàtica.
Garanties d'èxit Les dades són guardades al dispositiu.
Escenari principal 1. L'usuari posa en funcionament l'aplicació. 2. Té 2 opcions per tal d'introduir les seves dades: 2.1. Mitjançant el botó de configuració que hi ha a la llista inicial d'events. 2.2. Mitjançant l'opció del Racó i el mateix accés ja demana el login. 3. L'usuari introdueix els seu username i password. 4. El sistema verifica si són correctes. 5. Si són correctes mostra la pantalla següent (llista d'events inicials o al Racó). 6. Les dades són guardades el telèfon.
Extensions a. L'aplicació falla inesperadament: 1. L'usuari posa en funcionament l'aplicació. 2. L'aplicació s'atura i mostra un missatge d'error. 3. L'usuari pot reiniciar de nou l'aplicació. b. L'usuari no introdueix les dades correctament 1. L'usuari entra les dades i accepta. 2. En el procés de verificació es detecta un error. 3. Es notifica a l'usuari i la pantalla de login es neteja perquè tornin a ser introduïdes de nou.
Requisits especials - La lletra de les dades és prou gran perquè sigui visible a 30-40cm. - Els botons i pestanyes han de tenir una mida adequada perquè el seu ús no sigui dificultós. - Els caràcters del password no són visibles. - El servidor respon en menys de 8 segons . - L'idioma de l'aplicació és el que l'usuari té configurat en el telèfon. - Cal connexió a Internet.
Requisit/s assolit

RF_4 – Login.
Freqüència
Molt freqüent.

Cas d'ús 6: Preferències
Àmbit
Aplicació per Smartphones que usen el sistema operatiu Android 2.2 o superior. Pantalla situada al botó menú de l'apartat inicial.
Nivell
Principal: Visualització d'una llista on es podran seleccionar: • Número d'ítems que es vol que es mostrin a la llista inicial. • Activació/desactivació de l'arribada de les notificacions. Secundaria: • L'usuari pot seleccionar el nombre d'ítems (a partir d'un "radio button").
Actor/s primari
Estudiants.
Stakeholders i altres
- *Estudiants*: Vol configurar el nombre d'elements de la llista a mostrar o configurar les notificacions.
Precondició
L'estudiant ha realitzat el cas d'ús de Login.
Garanties d'èxit
La configuració del nombre d'elements i/o notificacions es realitza satisfactòriament. Les dades són guardades al dispositiu.
Escenari principal
1. L'usuari posa en funcionament l'aplicació. 2. Accedeix al botó menú en l'apartat inicial. 3. L'usuari pot: 3.1. Modificar les opcions a mostrar en la llista conjunta d'events. 3.2. Activar/desactivar les notificacions. 4. Les dades es guarden al telèfon.
Extensions
a. No es poden obtenir les dades del servidor o no es disposa de connexió: 1. L'usuari posa en funcionament l'aplicació. 2. No es poden obtenir les dades prèviament guardades. 3. L'usuari pot realitzar una restauració de les dades inicials. b. L'aplicació falla inesperadament: 1. L'usuari posa en funcionament l'aplicació. 2. L'aplicació s'atura i mostra un missatge d'error. 3. L'usuari pot reiniciar de nou l'aplicació.
Requisits especials
- Les llistes han de ser visibles a 30-40cm de distància del telèfon. - L'idioma de l'aplicació és el que l'usuari té configurat en el telèfon.

Requisit/s assolit
RF_6 – Preferències.
Freqüència
Freqüent. Al inici de l'aplicació.

Cas d'ús 7: Procés de consulta de l'agenda personal
Àmbit
Aplicació per Smartphones que usen el sistema operatiu Android 2.2 o superior. Pantalla situada a l'apartat Racó
Nivell
Principal: Procés de visualització de la llista.
Actor/s primari
Estudiants.
Stakeholders i altres
- *Estudiants*: Vol realitzar una consulta sobre l'estat del seu calendari. En ell es mostren els events que l'estudiant ha insertat manualment i els que s'afegeixen automàticament al Racó al matricular-se d'una assignatura.
Precondició
L'estudiant ha realitzat el cas d'ús de Login.
Garanties d'èxit
Els events de l'agenda es mostren en forma de llista en la pantalla del mòbil. Les dades són guardades al dispositiu.
Escenari principal
2. L'usuari posa en funcionament l'aplicació. 2. Accedeix a l'apartat del Racó. 3. Escull l'opció de veure l'agenda. 4. Es mostren les dades guardades a la base de dades del telèfon 5. S'envia la petició al servidor per obtenir les dades. 6. Les noves dades arriben i s'actualitza la llista i es guarden els noves dades. 7. L'usuari pot decidir si continua navegant o tanca l'aplicació.
Extensions
c. No es poden obtenir les dades del servidor o no es disposa de connexió: 1. L'usuari posa en funcionament l'aplicació. 2. La connexió al servidor falla, es notifica a l'usuari que hi ha hagut un error al actualitzar les dades. 3. L'usuari pot intentar actualitzar la informació o consultar les dades que hi ha guardades a la base de dades. d. Base de dades no creada: (inici aplicació per primera vegada) 1. Es demana la informació al servidor. 2. La base de dades no està creada. 3. Es crea la base de dades. 4. S'actualitza amb la informació que s'ha obtingut del servidor i es mostra al usuari. 5. L'usuari posa en funcionament l'aplicació. e. L'aplicació falla inesperadament:

1. L'usuari posa en funcionament l'aplicació.
2. L'aplicació s'atura i mostra un missatge d'error.
3. L'usuari pot reiniciar de nou l'aplicació.

f. L'usuari actualitza la informació.

1. L'usuari veu la llista d'ítems.
2. Prem sobre "menú".
3. Escull la opció "actualitzar".
4. El sistema, es connecta de nou al servidor.
5. Obté les dades necessàries.
6. Actualitza les base de dades.
7. Mostra la nova informació a l'usuari.

Requisits especials

- La llista ha de ser visible a 30-40cm de distància del telèfon.
- Els botons i pestanyes han de tenir una mida adequada perquè el seu ús no sigui dificultós.
- El servidor respon en menys de 8 segons.
- L'idioma de l'aplicació és el que l'usuari té configurat en el telèfon.
- Cal connexió a Internet.

Requisit/s assolit

RF_7 – Agenda.

Freqüència

Freqüent en èpoques d'examens parcials i finals.

Cas d'ús 8: Consultar l'horari

Àmbit

Aplicació per Smartphones que usen el sistema operatiu Android 2.2 o superior.

Pantalla situada a l'apartat Racó

Nivell

Principal: Procés de visualització de la llista.

Secundari: Veure la informació particular de cada ítem.

Actor/s primari

Estudiants.

Stakeholders i altres

- *Estudiants*: Vol realitzar una consulta sobre quin és el seu horari personal del dia actual.

Precondició

L'estudiant ha realitzat el cas d'ús de Login.

Garanties d'èxit

Es mostra la informació del dia actual en què es troba. Una taula amb les hores, assignatura i aula.

Hi ha un accés a una consulta d'un dia en concret.

L'usuari pot també realitzar consultes per dies (escollit en preferències).

Escenari principal

1. L'usuari posa en funcionament l'aplicació.

2. Accedeix a l'apartat del Racó.
3. Escull l'opció de veure horari.
4. Es realitza una petició al servidor per obtenir l'horari del dia actual.
5. Es mostra l'horari.
6. L'usuari pot decidir si continua navegant o tanca l'aplicació.

Extensions

a. No es poden obtenir les dades del servidor:
    1. L'usuari posa en funcionament l'aplicació.
    2. La connexió al servidor falla, es notifica a l'usuari que hi ha hagut un error al actualitzar les dades.
b. L'aplicació falla inesperadament:
    1. L'usuari posa en funcionament l'aplicació.
    2. L'aplicació s'atura i mostra un missatge d'error.
    3. L'usuari pot reiniciar de nou l'aplicació.
c. L'usuari selecciona realitza la consulta del dia següent / anterior:
    1. L'usuari prem una de les 2 opcions que hi ha sota la taula.
    2. L'aplicació mostra la informació d'aquell dia prèviament ja sol·licitada.
    3. L'usuari pot continuar navegant o sortir de l'aplicació.
d. L'usuari selecciona realitza la consulta d'un dia en concret:
    1. L'usuari prem l'opció de consultar un dia.
    2. Selecciona la data que vol consultar.
    3. L'aplicació es connecta al servidor i obté les dades.
    4. Es mostren les dades per pantalla, mitjançat una quadrícula.
    5. L'usuari pot continuar navegant o sortir de l'aplicació.

Requisits especials

- La taula ha de ser visible a 30-40cm de distància del telèfon.
- Els botons i pestanyes han de tenir una mida adequada perquè el seu ús no sigui dificultós.
- El servidor respon en menys de 20 segons de temps.
- L'idioma de l'aplicació és el que l'usuari té configurat en el telèfon.
- Cal connexió a Internet.

Requisit/s assolit

RF_8 – Horari acadèmic.

Freqüència

Freqüent, sobretot l'inici del curs.

Cas d'ús 9: Ocupació de les aules

Àmbit

Aplicació per Smartphones que usen el sistema operatiu Android 2.2 o superior.

Pantalla situada a l'apartat Racó

Nivell

Principal: Procés de visualització de la llista.

Secundari: Visualització del mapa de l'aula.

Actor/s primari

Estudiants.

Stakeholders i altres

-*Estudiants*: L'estudiant vol realitzar una consulta sobre l'estat de les aules o veure el mapa de d' una aula en concret.

Precondició

L'estudiant ha realitzat el cas d'ús de Login.

Garanties d'èxit

L'estudiant pot veure el nombre de màquines lliures de cada aula, si en aquell moment hi ha classe o no, i addicionalment pot consultar el mapa d'ocupació visual.

Escenari principal

1. L'usuari posa en funcionament l'aplicació.
2. Accedeix a l'apartat del Racó.
3. Escull l'opció de veure l'ocupació.
4. Es mostren les dades guardades a la base de dades del telèfon
5. S'envia la petició al servidor per obtenir les dades.
6. Les noves dades arriben i s'actualitza la llista i es guarden els noves dades.
7. L'usuari pot decidir si continua navegant o tanca l'aplicació.

Extensions

a. No es poden obtenir les dades del servidor o bé no es disposa de connexió:
    1. L'usuari posa en funcionament l'aplicació.
    2. La connexió al servidor falla, es notifica a l'usuari que hi ha hagut un error al actualitzar les dades.
    3. L'usuari pot intentar actualitzar la informació o consultar les dades que hi ha guardades a la base de dades.
b. Base de dades no creada: (inici aplicació per primera vegada)
    1. Es demana la informació al servidor.
    2. La base de dades no està creada.
    3. Es crea la base de dades.
    4. S'actualitza amb la informació que s'ha obtingut del servidor i es mostra al usuari.
    5. L'usuari posa en funcionament l'aplicació.
c. L'aplicació falla inesperadament:
    1. L'usuari posa en funcionament l'aplicació.
    2. L'aplicació s'atura i mostra un missatge d'error.
    3. L'usuari pot reiniciar de nou l'aplicació.
d. L'usuari actualitza la informació.
    1. L'usuari veu la llista d'ítems.
    2. Prem sobre "menú".
    3. Escull la opció "actualitzar".
    4. El sistema, es connecta de nou al servidor.
    5. Obté les dades necessàries.
    6. Actualitza les base de dades.
    7. Mostra la nova informació a l'usuari.
e. L'usuari vol consultar el mapa visual de les aules.
    1. L'usuari selecciona una aula de la llista.
    2. En funció de si és A5,B5 o C6 s'envia una petició al servidor.
    3. El servidor retorna la imatge on es pot veure visualment la ocupació d'aquell edifici.
    4. L'usuari prem el botó enrere (del telèfon) per tornar a la llista anterior o abandona l'aplicació.

Requisits especials

- La llista ha de ser visible a 30-40cm de distància del telèfon.

- Els botons i pestanyes han de tenir una mida adequada perquè el seu ús no sigui dificultós.

- El servidor respon en menys de 8 segons de temps.

- L'idioma de l'aplicació és el que l'usuari té configurat en el telèfon. - Cal connexió a Internet. - La imatge és lo suficientment gran com perquè es pugui apreciar des de 30-40cm del telèfon. Apart es disposa de zoom en cas de necessitat.
Requisit/s assolit RF_9 – Ocupació de les aules.
Freqüència Freqüent, ja que durant el curs es consulta habitualment l'ocupació.

Cas d'ús 10: Consultar assignatures de l'alumne
Àmbit Aplicació per Smartphones que usen el sistema operatiu Android 2.2 o superior. Pantalla situada a l'apartat Racó
Nivell Principal: Procés de visualització de la llista d'avisos i assignatures que està cursant. Secundaria: Consultar la informació d'un ítem de la llista (pot ser assignatura o avís).
Actor/s primari Estudiants.
Stakeholders i altres - *Estudiants*: Vol realitzar una consulta sobre les assignatures que està cursant., ja sigui, veure la informació de l'assignatura o els avisos.
Precondició L'estudiant ha realitzat el cas d'ús de Login. L'estudiant ha de tenir una assignatura matriculada.
Garanties d'èxit Les assignatures amb els seus corresponents avisos es mostren al dispositiu. Les dades són guardades al dispositiu.
Escenari principal 1. L'usuari posa en funcionament l'aplicació. 2. Accedeix a l'apartat del Racó. 3. Escull l'opció de veure assignatures. 4. Es mostren les dades guardades a la base de dades del telèfon 5. S'envia la petició al servidor per obtenir les dades. 6. Les noves dades arriben i s'actualitza la llista i es guarden els noves dades. 7. L'usuari pot decidir si continua navegant o tanca l'aplicació.
Extensions a. No es poden obtenir les dades del servidor o no es disposa de connexió: 1. L'usuari posa en funcionament l'aplicació. 2. La connexió al servidor falla, es notifica a l'usuari que hi ha hagut un error al actualitzar les dades. 3. L'usuari pot intentar actualitzar la informació o consultar les dades que hi ha

guardades a la base de dades.

b. Base de dades no creada: (inici aplicació per primera vegada)
    1. Es demana la informació al servidor.
    2. La base de dades no està creada.
    3. Es crea la base de dades.
    4. S'actualitza amb la informació que s'ha obtingut del servidor i es mostra al usuari.
    5. L'usuari posa en funcionament l'aplicació.

c. L'aplicació falla inesperadament:
    1. L'usuari posa en funcionament l'aplicació.
    2. L'aplicació s'atura i mostra un missatge d'error.
    3. L'usuari pot reiniciar de nou l'aplicació.

d. L'usuari actualitza la informació.
    1. L'usuari veu la llista d'ítems.
    2. Prem sobre "menú".
    3. Escull la opció "actualitzar".
    4. El sistema, es connecta de nou al servidor.
    5. Obté les dades necessàries.
    6. Actualitza les base de dades.
    7. Mostra la nova informació a l'usuari.

e. L'usuari consulta informació d'alguna assignatura.
    1. L'usuari veu la llista d'assignatures i avisos
    2. Prem sobre un ítem.
    3. Es mostra una nova vista amb la informació particular del ítem.
    8. L'usuari prem el botó enrere (del telèfon) per tornar a la llista anterior o abandona l'aplicació.

Requisits especials

- La llista ha de ser visible a 30-40cm de distància del telèfon.
- Els botons i pestanyes han de tenir una mida adequada perquè el seu ús no sigui dificultós.
- El servidor respon en menys de 8 segons de temps.
- L'idioma de l'aplicació és el que l'usuari té configurat en el telèfon.
- Cal connexió a Internet.

Requisit/s assolit

RF_10 – Assignatures matriculades.

Freqüència

Molt freqüent.

Cas d'ús 11: Consultar el correu

Àmbit

Aplicació per Smartphones que usen el sistema operatiu Android 2.2 o superior.

Pantalla situada a l'apartat del menú al Racó

Nivell

Principal: Procés de visualització de la llista de correus que l'usuari disposa i els correus que té pendents de llegir.

Secundaria: Consultar la informació d'un correu en particular.

Actor/s primari

Estudiants.

Stakeholders i altres

-*Estudiants*: Vol realitzar una consulta sobre els correus que hi ha a la seva bústia d'entrada i els correus que té pendents de llegir.

Precondició

L'estudiant ha realitzat el cas d'ús de Login.

Garanties d'èxit

Les assignatures amb els seus corresponents avisos es mostren al dispositiu.

Les dades són guardades al dispositiu.

Escenari principal

1. L'usuari posa en funcionament l'aplicació.
2. Accedeix a l'apartat del Racó.
3. Escull l'opció de veure correu.
4. Es mostren les dades guardades a la base de dades del telèfon
5. S'envia la petició al servidor per obtenir les dades.
6. Les noves dades arriben i s'actualitza la llista i es guarden els noves dades.
7. L'usuari pot decidir si continua navegant o tanca l'aplicació.

Extensions

a. No es poden obtenir les dades del servidor o no es disposa de connexió:
    1. L'usuari posa en funcionament l'aplicació.
    2. La connexió al servidor falla, es notifica a l'usuari que hi ha hagut un error al actualitzar les dades.
    3. L'usuari pot intentar actualitzar la informació o consultar les dades que hi ha guardades a la base de dades.

b. Base de dades no creada: (inici aplicació per primera vegada)
    1. Es demana la informació al servidor.
    2. La base de dades no està creada.
    3. Es crea la base de dades.
    4. S'actualitza amb la informació que s'ha obtingut del servidor i es mostra al usuari.
    5. L'usuari posa en funcionament l'aplicació.

c. L'aplicació falla inesperadament:
    1. L'usuari posa en funcionament l'aplicació.
    2. L'aplicació s'atura i mostra un missatge d'error.
    3. L'usuari pot reiniciar de nou l'aplicació.

d. L'usuari actualitza la informació.
    1. L'usuari veu la llista d'ítems.
    2. Prem sobre "menú".
    3. Escull la opció "actualitzar".
    4. El sistema, es connecta de nou al servidor.
    5. Obté les dades necessàries.
    6. Actualitza les base de dades.
    7. Mostra la nova informació a l'usuari.

f. L'usuari consulta informació d'algun correu.
    1. L'usuari veu la llista de correus.
    2. Prem sobre un ítem.
    3. Es mostra una nova vista amb la informació particular del ítem.
    8. L'usuari prem el botó enrere (del telèfon) per tornar a la llista anterior o abandona l'aplicació.

Requisits especials

-La llista ha de ser visible a 30-40cm de distància del telèfon.

- Els botons i pestanyes han de tenir una mida adequada perquè el seu ús no sigui dificultós.

-El servidor respon en menys de 8 segons de temps. -L'idioma de l'aplicació és el que l'usuari té configurat en el telèfon. - Cal connexió a Internet.
Requisit/s assolit
RF_11 – Correu.
Freqüència
Freqüent sinó té el correu redireccionat. Altrament, no serà útil.

Cas d'ús 12: Notificacions
Àmbit
Aplicació per Smartphones que usen el sistema operatiu Android 2.2 o superior. Barra superior del telèfon.
Nivell
Principal: L'usuari podrà desplegar la barra i veure de quin tipus és la notificació.
Actor/s primari
Estudiants
Stakeholders i altres
-*Estudiants*: Se li comunica que hi ha un nou avís al racó d'alguna assignatura de les quals està matriculat.
Precondició
L'estudiant ha realitzat el cas d'ús de Login. L'usuari ha d'estar subscrit als avisos en el Racó.
Garanties d'èxit
L'estudiant pot veure una nova notificació a la seva barra de notificacions
Escenari principal
1. L'usuari rep una notificació al telèfon 2. Consulta la notificació. 3. L'aplicació s'obre i li mostra el contingut de l'avís. 4. S'actualitza la llista i es guarden els noves dades. 5. L'usuari pot decidir si continua navegant o tanca l'aplicació.
Extensions
a. No es poden obtenir les dades del servidor o no es disposa de connexió: 1. L'usuari posa en funcionament l'aplicació. 2. La connexió al servidor falla, es notifica a l'usuari que hi ha hagut un error al actualitzar les dades. 3. L'usuari pot intentar actualitzar la informació o consultar les dades que hi ha guardades a la base de dades. b. L'aplicació falla inesperadament: 1. L'usuari posa en funcionament l'aplicació. 2. L'aplicació s'atura i mostra un missatge d'error. 3. L'usuari pot reiniciar de nou l'aplicació. c. L'usuari actualitza la informació. 1. L'usuari veu la llista d'ítems. 2. Prem sobre "menú". 3. Escull la opció "actualitzar". 4. El sistema, es connecta de nou al servidor.

5. Obté les dades necessàries. 6. Actualitza les base de dades. 7. Mostra la nova informació a l'usuari.
Requisits especials -El so i vibració de la notificació és per defecte la que el dispositiu té configurada. -L'idioma de l'aplicació és el que l'usuari té configurat en el telèfon. - Cal connexió a Internet.
Requisit/s assolit RF_12 – Notificacions.
Freqüència Molt freqüent ja que el llarg del quadrimestre es publiquen avisos constantment.

Cas d'ús 13: Logout
Àmbit Aplicació per Smartphones que usen el sistema operatiu Android 2.2 o superior. Pantalla situada a l'apartat Racó
Nivell Principal: Es permetrà a l'usuari sortir de l'opció del Racó per tal de tornar-se a loguinejar més endavant.
Actor/s primari Estudiants.
Stakeholders i altres -*Estudiants*: Vol sortir del Racó sense guardar les seves dades al telèfon.
Precondició L'estudiant ha realitzat el cas d'ús de Login.
Garanties d'èxit L'usuari surt de l'opció del Racó. L'usuari i el password són eliminats de la base de dades.
Escenari principal 1. L'usuari posa en funcionament l'aplicació. 2. Accedeix a l'apartat del Racó. 3. Escull l'opció de veure correu. 4. Es mostren les dades guardades a la base de dades del telèfon 5. S'envia la petició al servidor per obtenir les dades. 6. Les noves dades arriben i s'actualitza la llista i es guarden els noves dades. 7. L'usuari pot decidir si continua navegant o tanca l'aplicació.
Extensions a. L'aplicació falla inesperadament: 1. L'usuari posa en funcionament l'aplicació.

2. L'aplicació s'atura i mostra un missatge d'error. 3. L'usuari pot reiniciar de nou l'aplicació.
Requisit/s assolit
RF_4 – Login.
Freqüència
Poc freqüent ja que l'objectiu de l'aplicació és mantenir l'usuari informat del seu perfil a la Intranet.

Cas d'ús 14: About
Àmbit
Aplicació per Smartphones que usen el sistema operatiu Android 2.2 o superior. Pantalla situada a l'apartat del menú de les tres opcions inicials que hi haurà.
Nivell
Principal: Procés de visualització, de la informació de qui és l'autor de l'aplicació i correu de contacte.
Actor/s primari
Estudiants Externs
Stakeholders i altres
-*Estudiants*: Vol veure qui és el propietari de l'aplicació i qui la gestiona.
Garanties d'èxit
L'usuari pot veure qui ha creat i està gestionant l'aplicació.
Escenari principal
1. L'usuari posa en funcionament l'aplicació. 2. Accedeix al botó menú. 3. Escull l'opció de veure "About". 4. Es mostren les dades d'informació de contacte que hi ha guardades al telèfon.
Extensions
a. L'aplicació falla inesperadament: 1. L'usuari posa en funcionament l'aplicació. 2. L'aplicació s'atura i mostra un missatge d'error. 3. L'usuari pot reiniciar de nou l'aplicació. 4. Mostra la nova informació a l'usuari.
Requisits especials
-La lletra del text que es mostra ha de ser visible a 30-40cm de distància del telèfon. -L'idioma de l'aplicació és el que l'usuari té configurat en el telèfon.
Requisit/s assolit
RF_13– About.
Freqüència
Freqüent. Les persones es voldran assegurar que l'aplicació és fiable abans d'usar-la.

ANNEX II: RESULTATS DE L'ENQUESTA DE TESTEIG

A continuació, en la següent taula tenim les respostes que van donar els cinc estudiants que van està provant l'aplicació durant un mes i mig. Destacar que el grau de satisfacció es valora de 1 a 5, essent 1 equivalent a molt poc satisfactori i 5 el màxim nivell de satisfacció.

Enquesta Racó Mobile	**Respostes**	**Usuari 1**	**Usuari 2**	**Usuari 3**	**Usuari 4**	**Usuari 5**
1.- Creus que l'aplicació navega ràpid entre pantalles?	1 2 3 4 5	3	4	4	4	4
2.- Creus que les dades que es carreguen del servidor és ràpid, amb 3G? I amb Guifi?	1 2 3 4 5 / 1 2 3 4 5	4 / 4	4 / 4	5 / 5	4 / 4	5 / 5
3.- Creus que l'aplicació carrega les dades ràpid de la base de dades, és a dir, mentre s'està obtenint la nova informació?	1 2 3 4 5	5	3	5	3	5
4.- Quants cops l'aplicació ha forçat una parada?	0 0-2 2-4 4-6 >6	0	0 - 2	0	0 – 2	0
5.- L'aspecte de l'aplicació et sembla innovador?	1 2 3 4 5	4	3	4	4	3
6.- Creus que s'aprofiten correctament els elements utilitzats? DatePicker, Pestanyes…	1 2 3 4 5	5	4	4	5	5
7.- Quants Bugs aproximadament has reportat?	0 0-3 3-6 >6	0 - 3	3 - 6	0	0 – 3	0 – 3
8.- Creus que la llibreria BugSense és una bona eina de cara a reportar missatges?	1 2 3 4 5	5	5	5	5	4
9.- Has utilitzat molt l'aplicació?	Gens Bastant Normal Molt	Normal	Bastant	Normal	Normal	Normal
10.- Quants cops el dia la utilitzes?	0 0-3 3-5 5-7 >7	0 – 3	3 - 5	3 – 5	3 – 5	0 – 3
11.- T'ha sembla intuïtiva la navegació?	1 2 3 4 5	5	4	4	5	5
12.- Ha entès les funcionalitats o has necessitat ajuda?	1 2 3 4 5	5	4	5	3	5
13.- Consideres que les pestanyes tenen un tamany adequat?	1 2 3 4 5	3	4	4	4	4
14.-Consideres la lletra que es mostra en l'aplicació llegible(suficientment gran)?	1 2 3 4 5	5	5	5	5	5
15.-Consideres la lletra que els botons en l'aplicació són suficientment grans?	1 2 3 4 5	5	5	5	5	5

Figura 68. Respostes de les preguntes de l'enquesta per part dels provadors de l'aplicació.

Les tres preguntes que implicaven una resposta subjectiva per part dels enquestats, no s'han adjuntat aquí donat que ja s'han comentat en el corresponent apartat.

En la següent taula s'adjunta la informació dels telèfons mòbils dels que disposaven cadascun dels provadors:

	Marca del telèfon	Versió sistema operatiu
Usuari 1	HTC Wildfire	2.2
Usuari 2	Nexus One	2.3.4
Usuari 3	HTC Tattoo	2.3.7
Usuari 4	HTC Wildfire	2.2
Usuari 5	HTC Wildfire	2.2

Figura 69. Informació sobre els dispositius i versió instal·lats

A continuació es presenta les diferents gràfiques han sorgit de la realització de l'enquesta. El grau de satisfacció es valora de 1 a 5, essent 1 equivalent a molt poc satisfactori i 5 el màxim nivell de satisfacció.

EFICIÈNCIA

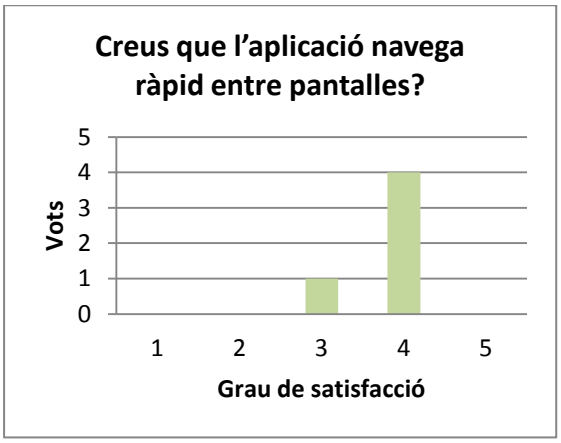

FIGURA 70

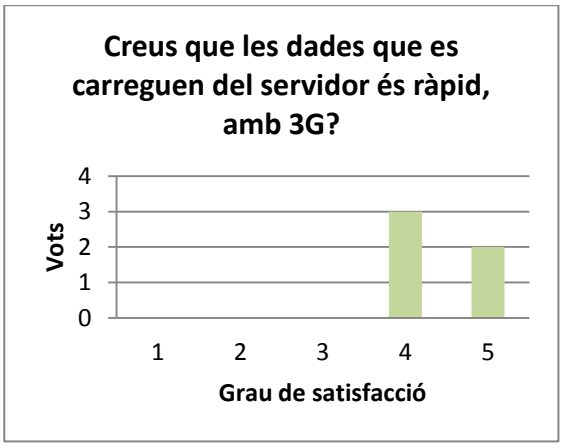

FIGURA 71

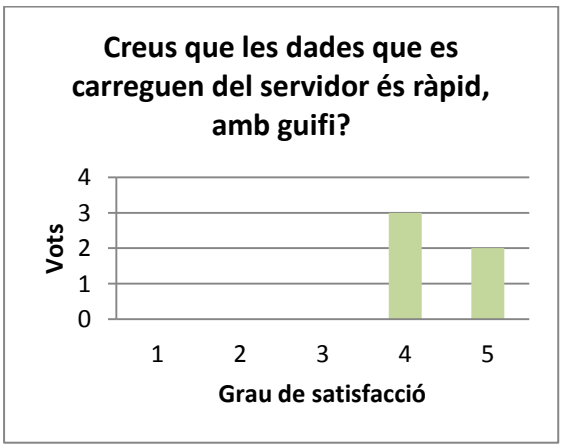

FIGURA 72

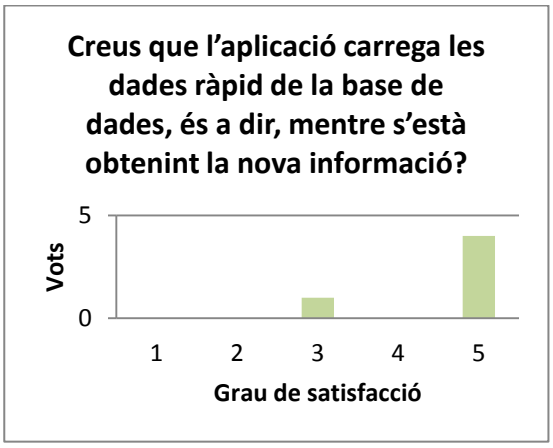

FIGURA 73

FIABILITAT

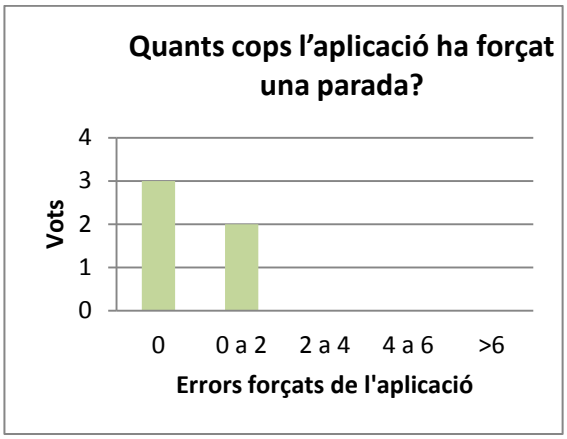

FIGURA 74

ACTUALITAT

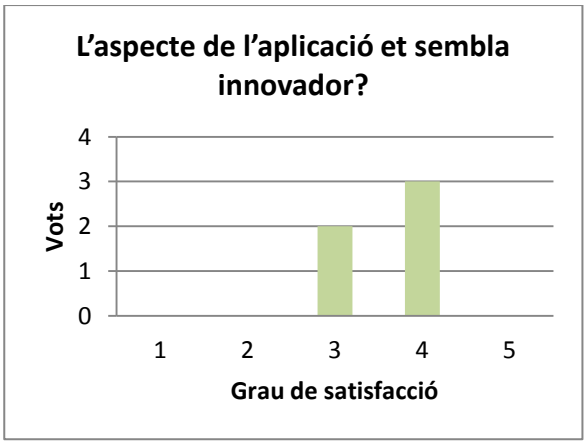

FIGURA 75

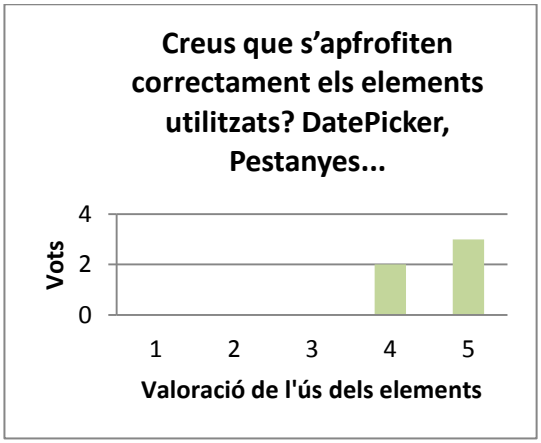

FIGURA 76

TESTABILITAT

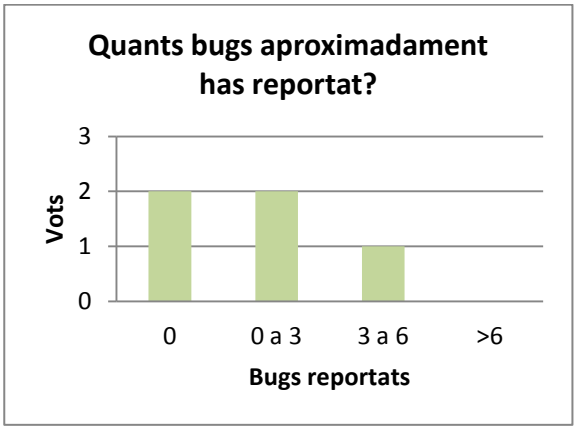

FIGURA 77

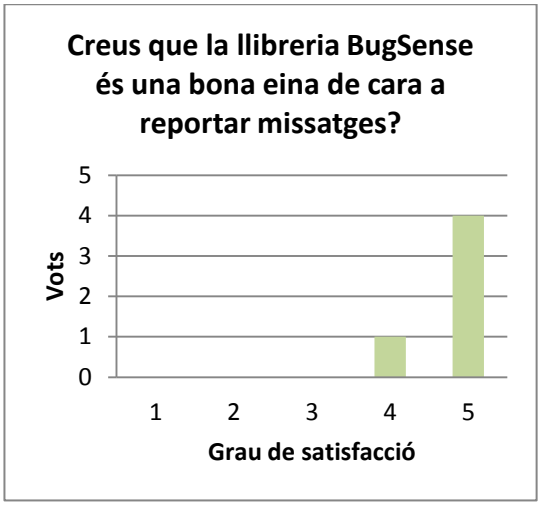

FIGURA 78

USABILITAT I SATISFACCIÓ

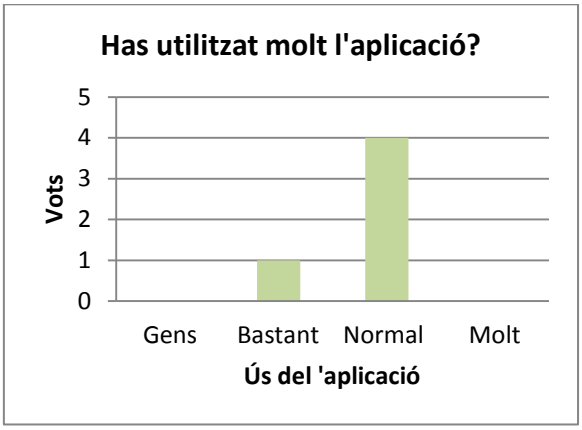

FIGURA 79

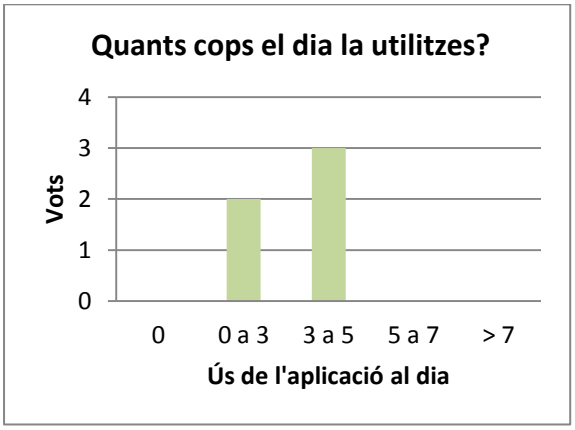

FIGURA 80

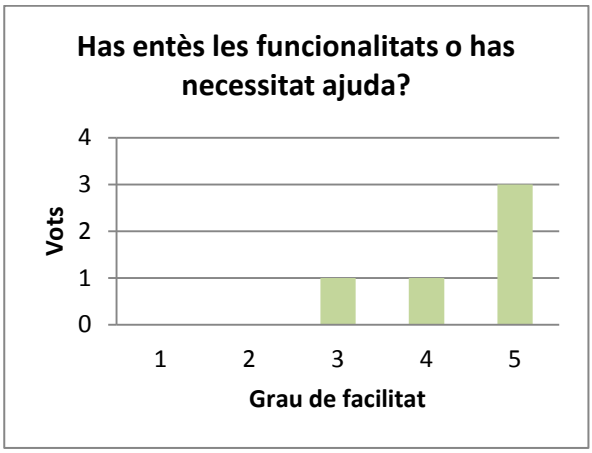

FIGURA 81

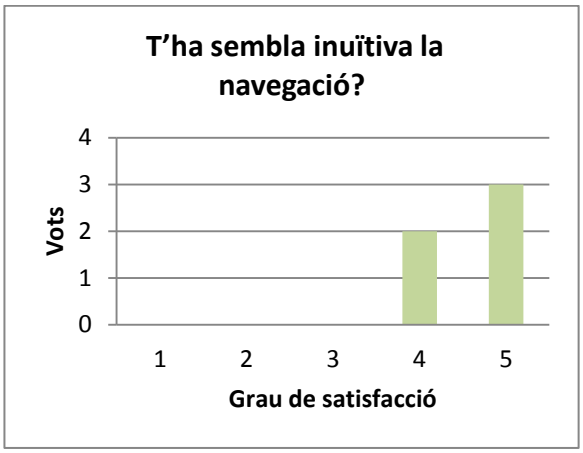

FIGURA 82

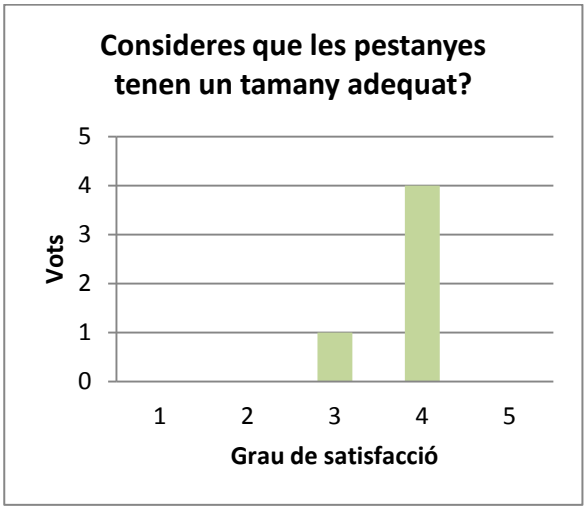

FIGURA 83

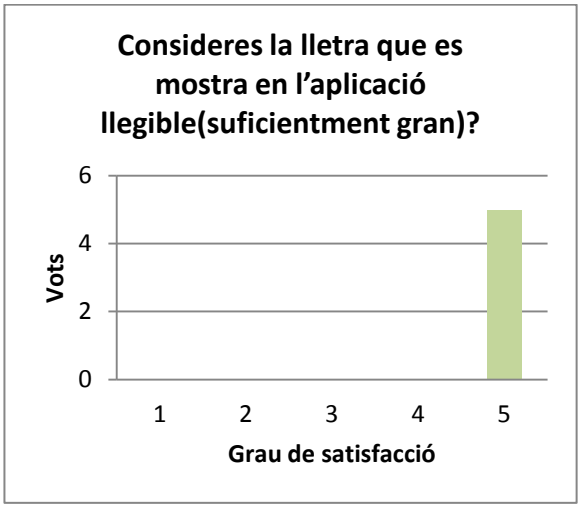

FIGURA 84

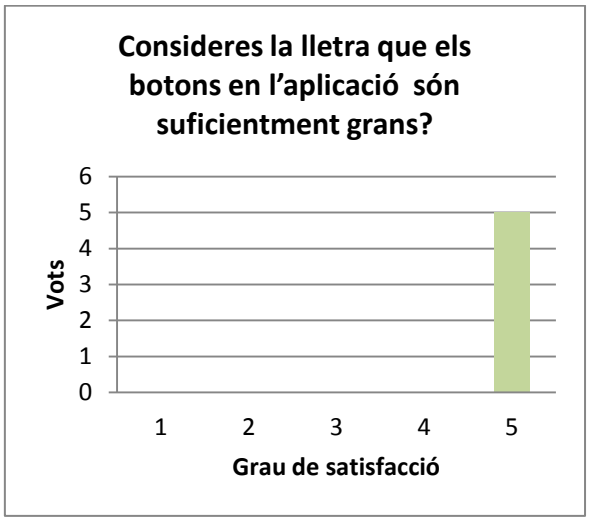
Consideres la lletra que els botons en l'aplicació són suficientment grans?
Vots
6
5
4
3
2
1
0
1
2
3
4
5
Grau de satisfacció

FIGURA 85

ANNEX III: PUBLICACIÓ A L'*ANDROID MARKET*

L'aplicació *Racó Mobile* serà distribuïda mitjançant l'*Android Market*. Recordar que és la distribuïdora oficial d'aplicacions per als usuaris d'Android. Amb lo qual, s'ha cregut convenient realitzar aquest apartat.

A continuació es detallen els diferents passos que s'han de seguir per tal de poder arribar a pujar una aplicació al Market:

1. Provar l'aplicació en gran mesura en un dispositiu real.
2. Considerar l'addició d'un Acord de Llicència de l'aplicació.
3. Pensar en la possibilitat d'afegir suport de llicències.
4. Especificar una icona i l'etiqueta en l'arxiu *manifest* de l'aplicació.
5. Desactivar el registre, la depuració i realitzar una neteja de dades /arxius .
6. La versió de la seva aplicació.
7. Obtenir una clau de xifrat adequat.
8. Registrar-se per obtenir una clau *API* de Google Maps, si l'aplicació està utilitzant els elements *MapView*.
9. Signatura de la sol·licitud.
10. Provar l'aplicació compilada.

Pel què fa als punt de l'1 al 8, es consideren fàcilment realitzables ja que fins al moment, ja s'han portat a terme. A continuació detallem, com fer el pas 9 i la publicació final al *Market*.

Els punts en què s'ha estructurat l'annex han estat, per una banda, el procés previ i de signatura de l'aplicació. I d'altra banda, el procés per a realitzar la publicació a l'*Android Market*.

SIGNATURA DE L'APLICACIÓ

El sistema Android requereix que totes les aplicacions instal·lades han de ser signades digitalment amb un certificat de clau privada. Aquesta acció es porta a terme pel desenvolupador de l'aplicació. El sistema Android utilitza el certificat com un mitjà per identificar l'autor de la sol·licitud i l'establiment de relacions de confiança entre les aplicacions. El certificat no s'utilitza per controlar les aplicacions que l'usuari pot instal·lar. El certificat no ha de ser signat per una autoritat de certificació: és perfectament admissible, i típic, per a aplicacions d'Android usar certificats amb signatura personal.

Els punts importants per entendre la signatura d'aplicacions són els següents:

- Totes les sol·licituds han d'estar signades. El sistema no instal·la una aplicació que no està signada.
- Per signar les aplicacions es pot usar certificats amb signatura. No hi ha autoritat de certificació necessària.

- Quan l'aplicació estigui llesta per ser publicada als usuaris finals, ha de ser signada amb una clau privada adequada. No es pot publicar una aplicació que està signada amb la clau de depuració generada per les eines de l'*SDK*.
- Si el certificat de l'aplicació expira després de que l'aplicació sigui instal·lada, l'aplicació seguirà funcionant amb normalitat.
- Es pot utilitzar les eines estàndards com *Keytool* i *Jarsigner* per crear les claus i la signatura de sol·licitud per a l'arxiu *APK* .
- Una vegada que s'hagi signat la sol·licitud, es pot utilitzar l'eina *zipalign* per optimitzar el paquet *APK* final.

Un cop s'hagin realitzat totes les accions descrites anteriorment, l'aplicació ja està preparada per ser publicada. A continuació, la descripció de les accions que s'haurien de realitzar per publicar l'aplicació.

PUBLICACIÓ AL *MARKET*

Per publicar l'aplicació a l'*Android Market*, primer hem de registrar al servei. Es pot fer mitjançant un compte de Google i acceptant els termes del servei. Un cop registrat, es podrà pujar l'aplicació al servei en qualsevol moment, actualitzar tantes vegades com es vulgui, i publicar-la quan estigui llesta. Un cop publicat, els usuaris poden veure l'aplicació, descarregar-la i valorar-la.

Mitjançant l'adreça *http://market.android.com/publish* es pot realitzar el registrar com a desenvolupador i publicar l'aplicació.

El publicar l'aplicació s'ha d'assegurar que compleix els requisits enumerats a continuació, que són imposats pels servidors de l'*Android Market* quan es carrega l'aplicació:

- La sol·licitud ha d'estar signada amb una clau criptogràfica privada i que el seu període de validesa acabi després del 22 d'octubre 2033.
- L'aplicació ha de definir els camps: *versionCode* i *versionName.* Aquest atributs es defineixen en l'element <*manifest*> de l'arxiu *Manifest* de l'aplicació. El servidor utilitza el camp *versionCode* com a base per a la identificació interna i la gestió dels canvis de l'aplicació. El camp *versionName* es mostra als usuaris i ens indica quina és la versió de l'aplicació.
- L'aplicació ha de definir una icona (*icon*) i un atribut (*label*) en l'element <*application*> del seu arxiu de *Manifest*. La icona i l'atribut és la imatge i el nom de l'aplicació que veuran els usuaris en els moments previs a llançar l'aplicació des del dispositiu.

Un cop assegurats tots els passos anteriors l'aplicació ja pot ser publicada a l'*Android Market*. Una última acció, totalment opcional per a l'usuari distribuïdor, és publicar un enllaç a la pàgina web el qual dóna als usuaris un accés directe a la descàrrega de l'aplicació en l'*Android Market*. En aquest cas seria a la web de la FIB o bé en el Racó mateix. Per portar-ho a terme simplement s'ha d'afegir el següent enllaç:

https://market.android.com/details?id=<package name>