Projecte Android App

Memòria

Ricard Sales López
Grau Eng.Informàtica
Tutoritzat per:
Dr.Francesc Guim Bernat
25 de Juny de 2014

INDEX DE CONTINGUT

INDEX DE FIGURES

1. INTRODUCCIÓ

Als darrers anys hem assistit a una evolució en les prestacions dels dispositius mòbils molt gran. **Apple** va crear una nova manera de relacionar-se amb aquests dispositius i una plataforma eficaç, però tancada, al voltant del seu propi *hardware*. L'evolució d'aquests esdeveniments va ser resposta de dues maneres diferents: per una banda els qui es van voler diferenciar del model "Apple" (com el cas de l'empresa **Blackberry**[1]) i per una altra banda els qui van abraçar el paradigma, copiant les bondats gràfiques del sistema operatiu *iOS*[2] (que és el que utilitza Apple amb els seus dispositius).

El desenvolupament d'un sistema operatiu sumat al disseny del *hardware* suposava uns costos enormes per a competir amb l'exitosa solució dels de Cupertino[3]. Cada marca va intentar, en primera instància, i amb els seus propis recursos, crear un sistema operatiu per als seus dispositius (per exemple, **Nokia** va desenvolupar el sistema operatiu *Symbian OS*[4]).

Google, atenta a tota aquesta situació, va veure l'oportunitat de situar-se estratègicament en una posició dominant si pogués oferir un sistema operatiu per a dispositius d'infinitat de marques. L'estratègia emprada ens recorda el cas de **Microsoft** i el seu sistema operatiu *Windows*, respecte a la mateixa Apple i el seu sistema operatiu *Mac OS*[5].

Realment la història s'ha repetit amb Google, a l'aconseguir, de manera semblant a Microsoft, crear un sistema operatiu que suportés un ventall ampli de *hardware*. Aquest fet va desencadenar la proliferació de noves marques de dispositius mòbils, que creaven terminals clònics amb components dissenyats per tercers (aprofitant l'impuls que la nova plataforma de Google – anomenada *Android*[6] – els brindava).

Android, com a sistema operatiu, té molt que agrair a *Linux* en la seva arquitectura (sense menystenir la màquina virtual de *Java*) però la filosofia de posicionament avantatjós de Google no té res a veure amb les llicències *Opensourse*[7]. De fet l'*iOS* és bàsicament un sistema operatiu UNIX amb una plataforma de desenvolupament amb programació *ObjectiveC*[8] (finalment totes dues plataformes giren entorn de *UNIX*). De manera semblant al que va fer en el seu moment Microsoft amb l'*MS-DOS*[9], Google no cobra als fabricants per utilitzar el seu sistema operatiu, però es reserva la propietat intel·lectual del mateix, així com l'explotació de la tenda virtual.

En definitiva, en el transcurs dels darrers quatre anys, el panorama de fabricants de terminals mòbils ha trobat un filó que han aprofitat, assumint el risc que Google tenia una posició estratègica incontestable.

A dia d'avui *Android* ha permès anar un pas més enllà, incloent suport per a *tablets*, *mini-pc*, dispositius per a *SmartTV*[10] i tota classe de sistemes encastats. *Android* pot córrer o executar-se en sistemes de *hardware* relativament senzills, d'igual manera que Línux, però atorga unes interfases gràfiques molt intuïtives i una plataforma única de descàrregues, anomenada *Google Play*, que sens dubte l'ha catapultat cap a un èxit des del punt de vista d'experiència d'usuari. No era difícil la recepta de l'èxit, tenint en compte l'estela deixada per Apple Store (centre de descàrregues que utilitza la companyia de la poma mossegada).

La situació actual ens convida a pensar que *Android* ha nascut per a quedar-se amb nosaltres unes dècades més. El negoci que ha creat al seu voltant és lo suficientment important com per a destinar recursos, tant pel que fa al desenvolupament d'aplicacions com pel que fa a l'estudi de la seva arquitectura. En aquest projecte invertiré temps i recursos per a ambdues coses i posaré en pràctica tot allò relacionat amb la programació que he pogut anar treballant gràcies a les diverses assignatures troncals així com el coneixement que he adquirit en la branca d'arquitectura de computadors i sistemes operatius.

2. MOTIVACIONS DEL PROJECTE

El projecte que plantejo respon a motivacions que puc resumir en tres nivells.

Des d'un punt de vista acadèmic, aquest projecte resulta especialment atractiu, donat que, a diferència dels ordinadors tradicionals, els dispositius mòbils disposen d'un ampli ventall de *chipsets* o dispositius integrats com GPS[11], quatre tipus diferents de connexions RF (2G/3G/4G[12], Wifi, Bluetooth, NFC[13], etc..), acceleròmetres, càmeres, giroscopis, brúixoles, detectors de llum ambiental, pantalles tàctils...que conviden a aprofundir en l'estudi del seu funcionament.

A nivell emocional, en una primera aproximació al potencial de les *API*[14] i els resultats gràfics possibles, resulta especialment motivador plantejar el desenvolupament d'una aplicació que pugui exprimir algunes d'aquestes funcionalitats integrades que he esmentat abans. Visualment parlant, és possible, gràcies al recull de classes d'objectes que ha fet Google, obtenir resultats molt atractius amb relativament poques línies de codi.

Pel que fa a la vessant econòmica, tal i com hem vist en l'apartat anterior, *Android*, com a plataforma, ofereix sens dubte oportunitats de negoci clares. En l'exercici de la meva activitat professional com a freelance, la programació amb *Android* m'obre un nou camp empresarial, compatible a nivell d'horari, amb la meva activitat quotidiana. El fet de poder treballar a qualsevol hora i des de casa em permetria diversificar els àmbits en què treballo i minimitzar la meva dependència respecte un únic model de negoci o mercat.

En conclusió, aprofitar les sinèrgies d'aquest projecte redundarà en un clar benefici a nivell formatiu, que de retruc m'obrirà noves portes a nivell professional.

3. DEFINICIÓ DEL PROJECTE

Amb aquest projecte pretenc donar a conèixer el procés de creació d'una aplicació per a *Android* al voltant de la funcionalitat de geo-posicionament i la consulta a Web Services[15], per tal d'oferir valor afegit a la recopilació de posicions.

L'aplicació ha de buscar en un moment concret a petició de l'usuari les dades meteorològiques en la posició actual. Un cop tornada aquesta informació ha de donar la possibilitat de guardar les dades en un registre que també emmagatzemarà la informació meteorològica de la posició que determinem com a "Home". Aquesta posició "Home" ha de poder-se configurar de forma automàtica prenent les dades del GPS.

Dintre del registre de mesuraments, s'han de poder revisar en detall les dades guardades de "Home" i de la posició guardada juntament amb la seva temperatura i el temps que hi feia. I en cada mesurament hem de poder veure una representació sobre un mapa de la posició actual, la posició on es va guardar la dada i home. Hem de poder veure la distància del recorregut total fet del primer mesurament fins l'últim a més de la representació gràfica.

A nivell de registre de mesuraments també volem una representació gràfica de tots ells i que sigui possible navegar a través dels mateixos per a revisar la posició de les mesures.

És obligat que els registres es puguin esborrar un a un o bé buidar totes les dades per tal de poder reiniciar les nostres mesures o eliminar aquelles que ja no ens siguin útils.

La voluntat inicial és poder presentar aquest coneixement d'una forma atractiva a nivell gràfic i poder guardar el registre en un arxiu de manera permanent.

3.1. OBJECTIUS PER ASSOLIR

Els objectius a assolir amb aquest treball són els següents:

- Familiaritzar-me amb l'arquitectura d'*Android* i l'entorn de desenvolupament *Android Studio*.
- Aprendre a utilitzar l'*API* per a dispositius específics com el mòdul GPS i la part de representació gràfica.
- Estudiar i explotar Web Services tant de *Google Maps*[16] com serveis per a consultar dades meteorològiques.
- Agafar fluïdesa en la creació de la capa de presentació gràfica de l'aplicació.
- Recordar part dels estudis relacionats amb la relació del *software* amb l'usuari, per tal de fer-ne fàcil i atractiu l'ús.
- Aprendre de l'experiència de planificació i afinar les meves prediccions de temps en projectes de programació.

Finalment, el repte més important és posar en pràctica tots els coneixements assolits durant els estudis. Una de les coses mes importants que he après en aquests anys ha estat acostumar-me a adquirir un ritme d'aprenentatge ràpid. Sens dubte, aquest projecte posarà al límit aquesta capacitat així com la de gestionar el temps, que de per sí serà molt afitat.

4. PLANIFICACIÓ TEMPORAL

4.1. PLANIFICACIÓ D'ENTREGUES

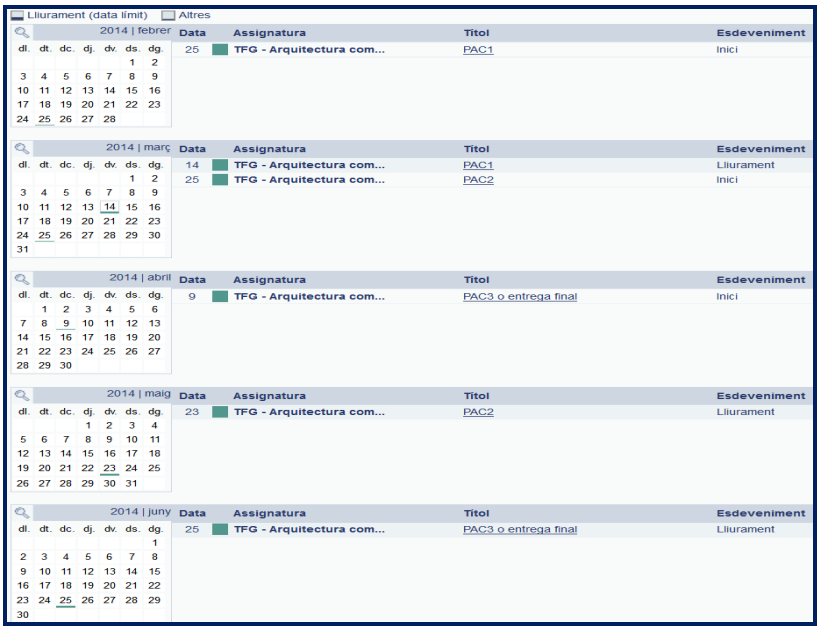

Figura 1: Planificació d'entregues

4.2. PLANIFICACIÓ DETALLADA

Nombre de tarea	T	Comença	Fi
TFC Arquitectura de Computadors i SS.OO. - Aplicació d'Android	**121 d**	**25/02/14**	**25/06/14**
01 - Definició i Planificació (PAC1)	**18 d**	**25/02/14**	**14/03/14**
01.01 - Redactat Motivació, Objectius i Descripció	15 d	25/02/14	11/03/14
01.02 - Elaboració del Pla de Treball i Valoració Econòmica	2 d	12/03/14	13/03/14
01.03 - Revisió i Entrega Definició i Pla de Treball (PAC1)	1 d	14/03/14	14/03/14
02 - Desenvolupament Tècnic (PAC2)	**70 d**	**15/03/14**	**23/05/14**
02.01 - Investigació, Aprenentatge i Documentació	20 d	15/03/14	03/04/14
02.02 - Documentar la Selecció de Tecnologies Escollides	2 d	04/04/14	05/04/14
02.03 - Disseny del Programari	4 d	06/04/14	09/04/14
02.04 - Preparació Entorn de Treball	2 d	10/04/14	11/04/14
02.05 - Implementació Codi	40 d	12/04/14	21/05/14
02.06 - Valoració de Desviacions i Recull de Dificultats	1 d	22/05/14	22/05/14
02.07 - Revisió i Entrega Desenvolupament Tècnic (PAC2)	1 d	23/05/14	23/05/14
03 - Disseny i Implementació (PAC3/Entrega Final)	**33 d**	**24/05/14**	**25/06/14**
03.01 - Correcció de Desviacions	7 d	24/05/14	30/05/14
03.02 - Filmar i Fotografiar Demostracions	2 d	31/05/14	01/06/14
03.03 - Finalitzar la Memòria que Inclourà la Demo	7 d	02/06/14	08/06/14
03.04 - Realització de PowerPoint de la Memòria	7 d	09/06/14	15/06/14
03.05 - Realització d'un Video de Presentació i Demo	9 d	16/06/14	24/06/14
03.06 - Revisió Entrega Final (PAC3)	1 d	25/06/14	25/06/14

Figura 2: Planificació detallada

4.3. DESCRIPCIÓ DE LES DIVISIONS EN ACTIVITATS

4.3.1. REDACTAT MOTIVACIÓ, OBJECTIUS I DESCRIPCIÓ

L'inici del procés es òbviament una primera aproximació al projecte. Durant aquest primer període, he escollit, entre diferents alternatives, quina seria la tasca a desenvolupar.

Un cop decidits els detalls del que es vol projectar arriba el moment de redactar el perquè de la decisió presa i argumentar-ne les raons.

En acabat, ja estaré en disposició de descriure en detall què és el que ha de fer l'aplicació que vull desenvolupar i els objectius personals que vull assolir.

4.3.2. ELABORACIÓ DEL PLA DE TREBALL I VALORACIÓ ECONÒMICA

L'arribada d'aquesta fita ens obliga a mirar el calendari per tal d'encabir-hi tota la feina a fer en les dates d'entregues, que són força justes. S'ha de fer un exercici de càlcul humil per tal de dotar de temps suficient a cada tasca.

Un cop fet aquest càlcul temporal i coneixent què ens caldrà per tal de portar endavant el desenvolupament i les proves de l'aplicació, ja estic en disposició de poder fer una valoració econòmica del que em suposarà tot plegat.

La presentació d'aquests càlculs la realitzaré mitjançant un diagrama de Gannt i un recull de costos a partir d'una valoració del temps i dels costos del *hardware*.

4.3.3. REVISIÓ I ENTREGA DEFINICIÓ I PLA DE TREBALL (PAC1)

Com a regla bàsica de treball, em vull imposar la norma de dedicar un dia a fer una relectura de l'entrega per tal de polir els aspectes de redacció i formals del document a entregar. És per aquest motiu que introdueixo aquesta fita a la programació del pla de treball.

4.3.4. INVESTIGACIÓ, APRENENTATGE I DOCUMENTACIÓ

Aquest període és probablement el més important de tots, ja que en depenen les decisions de disseny i l'auto-formació. Poder assolir els objectius personals i el desenvolupament funcional del projecte està relacionat directament amb la intensitat amb la que treballi durant aquest temps.

4.3.5. DOCUMENTAR LA SELECCIÓ DE TECNOLOGIES ESCOLLIDES

He de fer una petita parada en aquest punt per a posar per escrit les tecnologies que faré servir i el perquè de la seva elecció. En certa manera ha de ser un petit informe del temps d'aprenentatge (comentat al punt anterior) que he passat i del que he tret en clar per a aplicar al projecte.

4.3.6. DISSENY DEL PROGRAMARI

Arribats a aquest punt estaré en disposició de fer el disseny de l'aplicació a nivell teòric. És possible que durant la implementació el disseny pugui patir petits ajustos a causa de la meva falta d'experiència. Malgrat tot espero afinar al màxim possible a la primera per tal d'evitar marxes enrere, que sens dubte tindrien un efecte molt negatiu en l'aplicació.

4.3.7. PREPARACIÓ ENTORN DE TREBALL

Hi ha un període necessari per tal d'adequar l'entorn de treball. A causa de les necessitats de computació una mica fortes (donat que s'han d'emular dispositius *Android* actuals) he de fer una inversió en *hardware*, que per una altra banda també hauré de muntar. Tanmateix hauré de configurar el programari de desenvolupament i proves.

4.3.8. IMPLEMENTACIÓ CODI

El gruix principal del volum de feina s'anirà en implementar el codi de l'Aplicació, doncs es posarà a prova la feina feta abans d'implementar-la. Per tots és conegut que el temps que es destina a escriure el codi és també un temps en què es cometen errors i, per tant, un moment en que es va tirant endavant i enrere fins a obtenir un codi operatiu. Aquest procés també el considero un moment en el què s'aprèn de l'experiència.

El final del període és per a documentar el codi escrit i millorar el seu funcionament intentant simplificar-ho. Menys instruccions per a fer el mateix suposa sempre un estalvi de processador i memòria al dispositiu final en el que ha de córrer l'aplicació. Òbviament, les primeres proves en l'emulador es faran durant la programació, especialment cap el final de la mateixa.

4.3.9. VALORACIÓ DE DESVIACIONS I RECULL DE DIFICULTATS

És força probable que durant el procés ens trobem amb dificultats que o bé no hem calculat abans o bé són imponderables. Per tant, és l'ocasió per a fer un recull de l'experiència viscuda fins el moment i valorar si s'han produït desviacions en l'ús del temps o en les funcionalitats de l'aplicació que es van determinar al principi.

4.3.10. REVISIÓ I ENTREGA DESENVOLUPAMENT TÈCNIC (PAC2)

Aquesta fita té ha veure amb la voluntat de fer una revisió formal del text a entregar.

4.3.11. CORRECCIÓ DE DESVIACIONS

Dins de la programació he deixat aquest temps per si fos possible redreçar alguna de les desviacions (en cas de ser petites). Aquest és un temps de maniobra, passada l'entrega de la PAC2, que crec necessari planificar, tot i que desitjo que no em faci falta utilitzar-la a fons.

4.3.12. FILMAR I FOTOGRAFIAR DEMOSTRACIONS

Desitjo poder fer vídeos i fotografies de les demostracions de funcionament de l'aplicació per tal d'utilitzar-ho més tard en els documents finals a entregar. Es tracta de poder generar continguts per a enriquir la presentació del projecte.

4.3.13. FINALITZAR LA MEMÒRIA QUE INCLOURÀ LA DEMOSTRACIÓ

El document principal és la memòria, motiu pel qual s'ha de finalitzar i tancar el document final a lliurar en aquest període. Afinar-lo a nivell formal és importantíssim a fi que resulti atractiu per al lector.

4.3.14. REALITZACIÓ DE POWERPOINT DE LA MEMÒRIA

Un cop acabada la memòria i utilitzant com a base la mateixa, prepararé una presentació amb PowerPoint. La idea és crear un document que m'ajudi per a fer una presentació d'avant d'un tribunal amb una sala amb projector.

4.3.15. REALITZACIÓ D'UN VÍDEO DE PRESENTACIÓ I DEMOSTRACIÓ

Aprofitant les idees de la presentació voldria crear un vídeo que serveixi tant com a presentació del projecte com per a fer una demostració de l'aplicació. Part dels continguts els hauria generat en la fita 4.3.12 i la resta vindrien de la presentació.

4.3.16. REVISIÓ ENTREGA FINAL (PAC3)

Encara que durant la redacció dels diferents documents aniré afinant-ho tot, crec que abans de l'entrega final caldrà revisar tota la documentació. Faré especial incidència en la coherència a nivell estètic del tres documents a entregar per tal que la imatge dels mateixos sigui adequada per a una presentació professional.

4.3.17. PERÍODE DE CONSULTES DEL TRIBUNAL

La darrera tasca roman fora de la planificació pròpiament dita. Com a part final del TFG, s'establirà un període per tal de poder respondre les preguntes que es poden generar un cop el tribunal analitzi en detall el projecte. En aquest cas l'objectiu és que no siguin precisos aclariments sobre aspectes que no estiguin a la memòria. Malgrat tot, és també necessari enfrontar les preguntes del tribunal per a estar segurs que s'han assolit els coneixements tècnics i formals exigibles.

4.4. DIAGRAMA DE GANTT

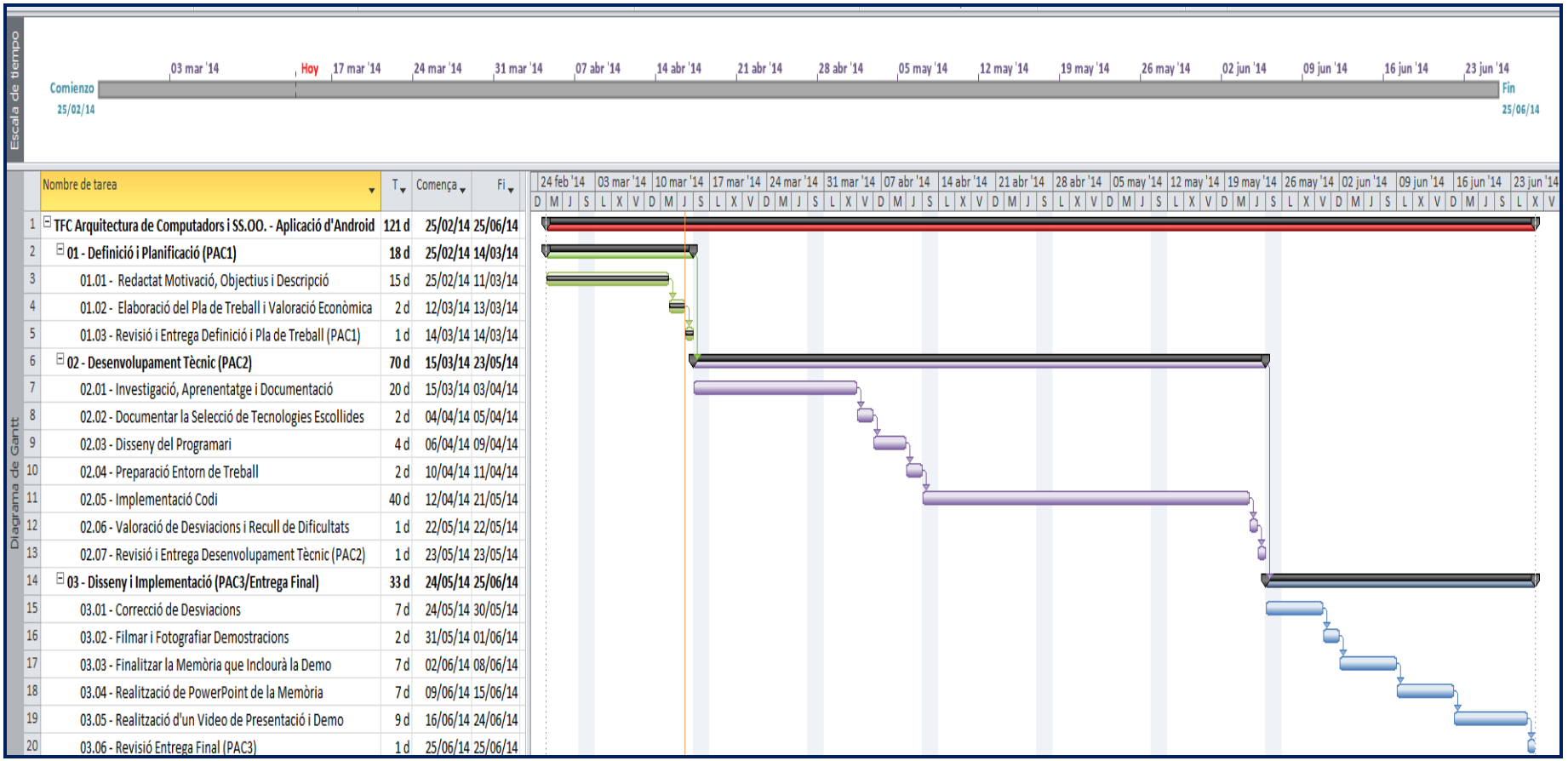

Figura 3: Diagrama de Gantt

5. VALORACIÓ ECONÒMICA

A continuació es presenta la **valoració econòmica** del temps a invertir i de l'ampliació de *hardware* necessària per a l'emulador de dispositius *Android*. Es compta de mitjana dues hores diàries de dedicació, amb el preu/hora que ofereixo habitualment per a projectes curts.

Valoració Econòmica en Temps	Dies	Hores de Treball	Cost a 22€/h
01 - Definició i Planificació (PAC1)	**18**	**36**	**792,00 €**
01.01 - Redactat Motivació, Objectius i Descripció	15	30	660,00 €
01.02 - Elaboració del Pla de Treball i Valoració Econòmica	2	4	88,00 €
01.03 - Revisió i Entrega Definició i Pla de Treball (PAC1)	1	2	44,00 €
02 - Desenvolupament Tècnic (PAC2)	**70**	**140**	**3.080,00 €**
02.01 - Investigació, Aprenentatge i Documentació	20	40	880,00 €
02.02 - Documentar la Selecció de Tecnologies Escollides	2	4	88,00 €
02.03 - Disseny del Programari	4	8	176,00 €
02.04 - Preparació Entorn de Treball	2	4	88,00 €
02.05 - Implementació Codi	40	80	1.760,00 €
02.06 - Valoració de Desviacions i Recull de Dificultats	1	2	44,00 €
02.07 - Revisió i Entrega Desenvolupament Tècnic (PAC2)	1	2	44,00 €
03 - Disseny i Implementació (PAC3/Entrega Final)	**33**	**66**	**1.452,00 €**
03.01 - Correcció de Desviacions	7	14	308,00 €
03.02 - Filmar i Fotografiar Demostracions	2	4	88,00 €
03.03 - Finalitzar la Memòria que Inclourà la Demo	7	14	308,00 €
03.04 - Realització de PowerPoint de la Memòria	7	14	308,00 €
03.05 - Realització d'un Video de Presentació i Demo	9	18	396,00 €
03.06 - Revisió Entrega Final (PAC3)	1	2	44,00 €
TFC Arquitectura de Computadors i SS.OO. - Aplicació d'Android COST TOTAL DE LA INVERSIÓ DE TEMPS (1)	**121**	**242**	**5.324,00 €**

Valoració Econòmica Hardware	
CPU Intel Core i7 4771	258,11 €
ASUS Z87-K	96,49 €
Kingston HyperX blu 16GB (8x2)	121,92 €
Google Nexus 4 Android 4.4	Sense Cost
Woxter Tablet PC Nimbus 101 Q (10,1") Android 4.2	Sense Cost
TFC Arquitectura de Computadors i SS.OO. - Aplicació d'Android COST TOTAL DE LA INVERSIÓ EN HARDWARE (2)	**476,52 €**

Valoració Econòmica Totals	
Total Inversió en Temps (1)	5.324,00 €
Total Inversió en Hardware (2)	476,52 €
TOTAL PROJECTE TFC Arquitectura de Computadors i SS.OO. - Aplicació d'Android	**5.800,52 €**

Figura 4: Valoració econòmica

6. APRENENTATGE I DOCUMENTACIÓ

El gruix fonamental d'aquest projecte ha estat l'aprenentatge de les *API* d'*Android*, *Google Maps* i *OpenWeatherMap.org*. Les decisions sobre les *API*s (a banda de la imprescindible d' *Android*) les justificaré en el següent punt, anomenat "Tecnologies i Eines Escollides". Les eines fonamentals han estat les webs de documentacions d'*API*s i exemples oficials següents:

- Pàgina oficial per a desenvolupadors d'Android[17]
- API de Google Maps[18]
- API d'Openwheathermap[19]

Malgrat que tota la ingent documentació i exemples està molt ordenada, he acabat tenint problemes per a poder entendre com programar per *Android* només amb aquest ajut. No podem desmerèixer la meravellosa feina feta, però només coneixent *Java* – que és el meu punt de partida personal - no és suficient per a entendre com funciona i com es programa un dispositiu *Android*. He hagut de buscar formació a internet i tres eines bàsiques m'han pogut procurar el nivell que necessitava per a programar l'aplicació:

- **Curs d'Android de FranchoLab**[20]
- **Curs OnLine d'*Android* Avançat del Col·legi d'Enginyers Tècnics d'Informàtica del Principat d'Asturies**[21]: Vaig poder-me matricular a un preu baix gràcies a la col·laboració amb el col·legi català COETIC del que sóc afiliat.
- **GitHub**[22]: És una plataforma de col·laboració on molts programadors pengen de forma gratuïta el seu codi i on he pogut trobar exemples del que necessitava programar. S'ha de dir, malgrat tot, que hi ha molt codi antic i que no es pot implementar a l'ultima *API*.

La planificació d'aquesta fase del projecte ha quedat molt curta, doncs és senzillament impossible saber la quantitat d'hores d'aprenentatge real que he necessitat. Puc aproximar que han estat entre 1,5 i 2 mesos a raó de 40 hores setmanals, durant les quals he intercalat temps d'aprenentatge i d'implementació de codi. Tot i que he realitzat desenes de programes durant l'aprenentatge, quan m'he trobat davant d'un repte pel que fa al codi, he hagut de buscar de nou o estudiar més mòduls que pensava que no em calien en una primera estimació. Aquesta ha estat sense dubte l'etapa més dura, però també la que em donarà un rèdit major a nivell d'aprenentatge.

Crec sincerament que el camp és enorme, molt mes gran del que tenia previst en un principi. Aquest ha estat sens dubte "un pecat d'ignorància" però he pogut avançar molt en la programació d'*Android* i em sento amb capacitat de desenvolupar petites *Apps* a nivell professional.

7. TECNOLOGÍES I EINES ESCOLLIDES

7.1. ENTORN DESENVOLUPAMENT

Per tal de fer una primera aproximació, vaig indagar quines eines utilitzaven habitualment els desenvolupadors d'*Android*. I vaig comprovar que, fonamentalment, hi ha dues opcions majoritàries com a eina de desenvolupament o IDE[23]:

- ***Eclipse***: és una bona opció per a més d'un llenguatge, però és necessari configurar mòduls per tal de poder-hi treballar (i això no sempre resulta fàcil de configurar).

- ***Android Studio***: és l'IDE desenvolupat per Google, basat en *IntelliJ*, i que integra tot el necessari per a programar i provar amb emulador de diferents models de telèfons *Android*.

L'opció més segura, especialment per a un nou vingut, és *Android Studio* (*Studio* en endavant). S'ha de dir que al poc temps de treballar amb ell t'hi acostumes i es torna una eina molt potent. Per una altra banda integra en l'aplicació principal una *Interface* per a veure els logs[24] a nivell de *Java* i de dispositiu *Android*, navegar pels diferents fitxers i programar en *Java*/*Android*.

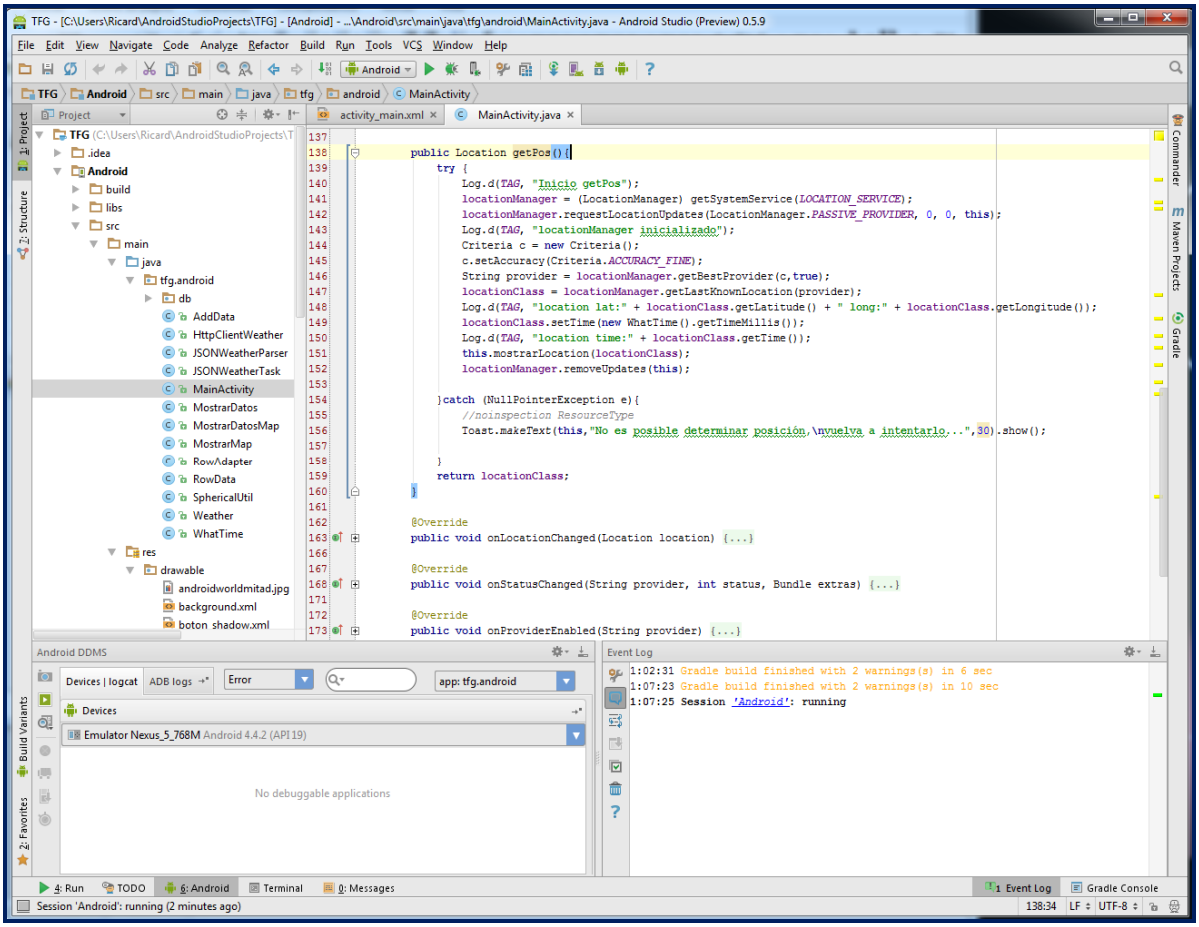

Figura 5: *Android Studio*

Per la forma de treballar d'*Android*, es programen per un costat la lògica de l'aplicació i per un altre les representacions gràfiques i menús. És per aquest motiu que l'*Studio* ens dóna una opció de programació gràfica i per *XML*, a banda de per codi, per facilitar les tasques:

- **L'opció gràfica** ens permet arrossegar nous elements o configurar els que tenim amb el ratolí directament sobre la pantalla d'un terminal a escollir.

- **L'*XML*** es pot editar veient el resultat en temps real a la representació del resultat.

En les captures següents es pot veure com funcionen en un Nexus 4:

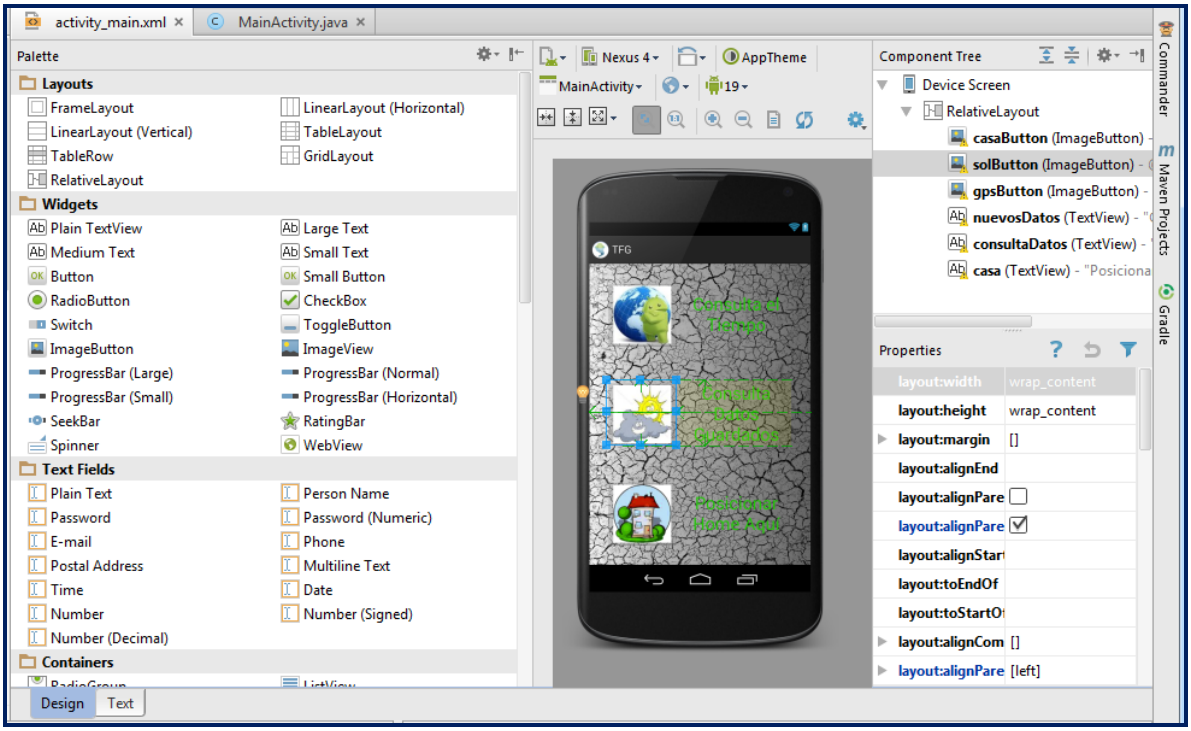

Figura 6: Captura opció gràfica

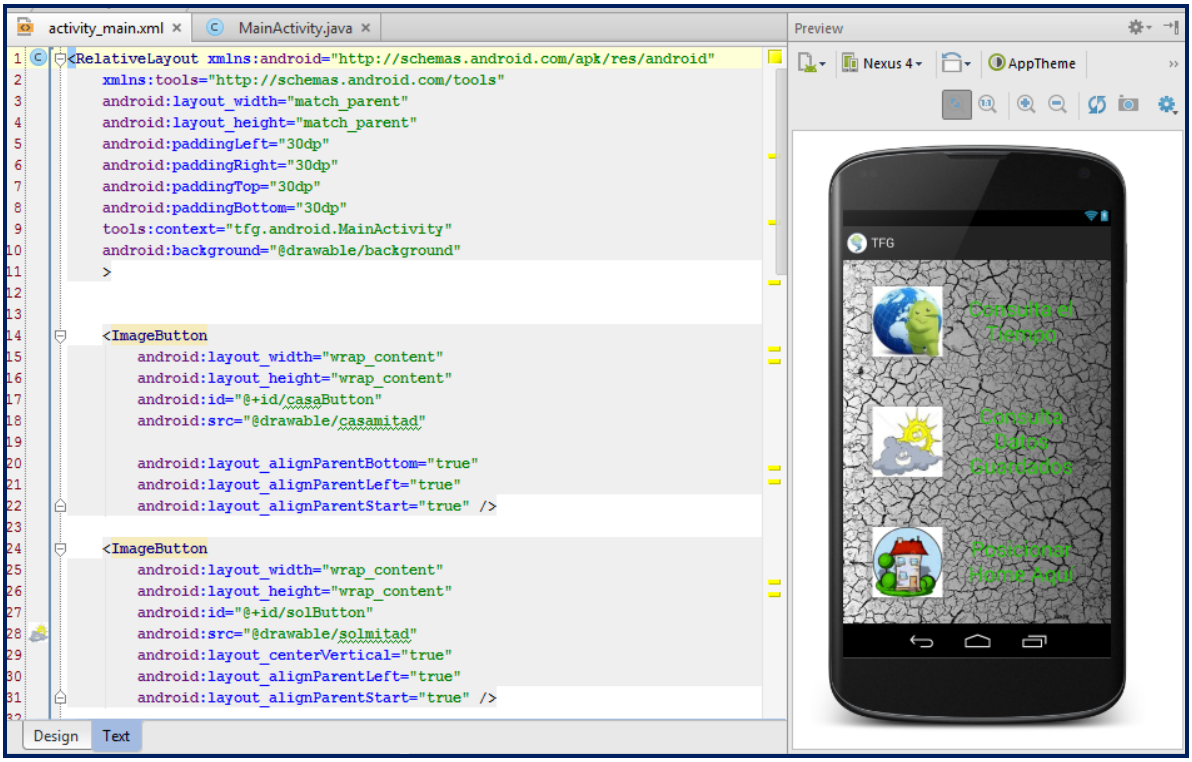

Figura 7: Captura opció XML

Es pot intuir la potència d'aquests editors fixant-nos en les "*Palette*" i "*Properties*" de la captura de l'entorn gràfic. Per una altra banda, veure l'efecte de l'edició del *XML* a la part dreta del dispositiu que escollim és una eina molt útil per no haver d'anar compilant per a veure quins resultats donaran les nostres modificacions.

Com a eines integrades en vull destacar dues que son primordials en el dia a dia. Per una banda l'***Android* Virtual Device Manager (AVD)** que serveix per a configurar i arrancar terminals simulats per a provar les aplicacions. Per una altra banda l'***Android* Debug Monitor (DDMS)** que em permet configurar els paràmetres i monitoritzar el rendiment i *log* del meu dispositiu real (connectat al meu PC) o emulat[25]. La captura mostra totes dues eines durant l'execució de l'aplicació. Com es pot veure, hem utilitzat el Monitor per a indicar al nostra terminal emulat (un Nexus 5 tal com es pot veure a la finestra de l'AVD) que la posició actual GPS es Latitud= 37,422006 i Longitud= -122,084095. Aquesta posició coincideix amb la seu de Google a Mountain View, California, USA (la posició es pot veure al següent link: https://www.google.com/maps/place/37%C2%B025%2719.2%22N+122%C2%B005%2702.7%22W/@37.422006,-122.084095).

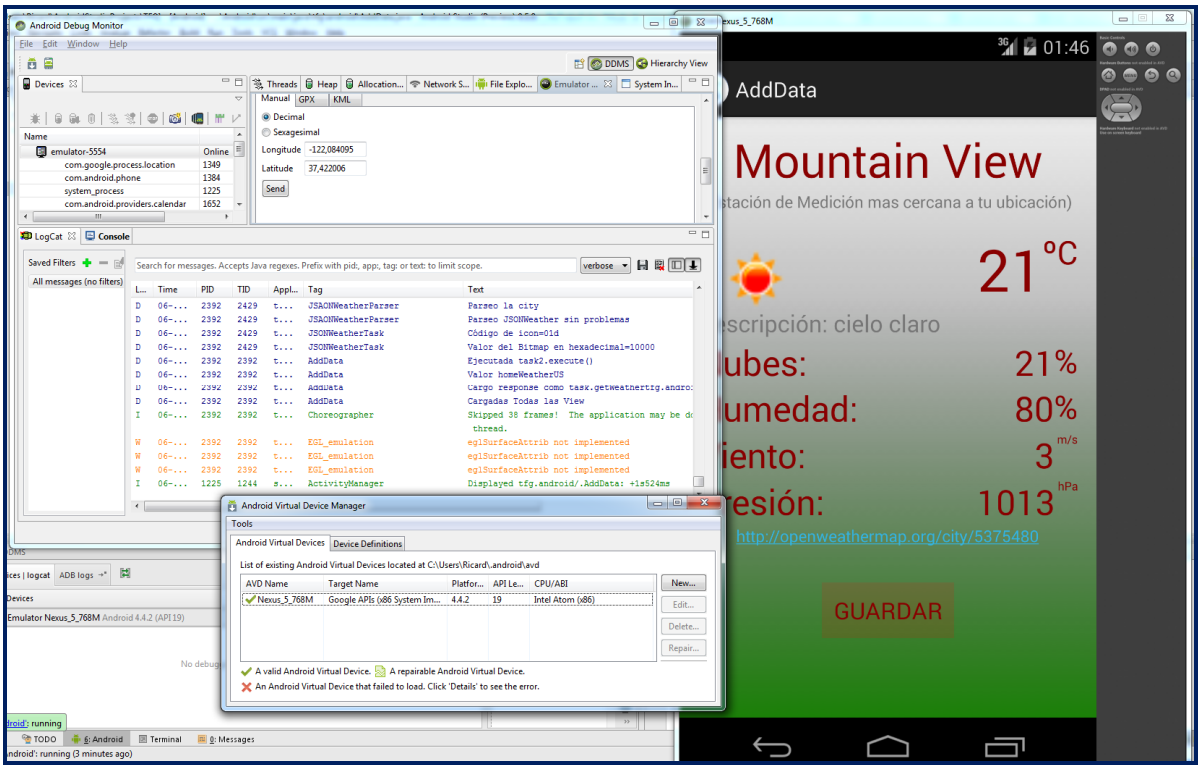

Figura 8: *Android* Debug Monitor

L'únic aspecte negatiu d'aquest fantàstic entorn de programació és l'enorme consum de recursos que provoca. En l'apartat econòmic he hagut de calcular la compra del *hardware* necessari per tal d'assumir la potència que precisa. En aquest sentit, he comprat una nova *CPU* (i conseqüentment una placa base nova) amb 4 cores i 8 fils d'execució amb 8GB de *RAM DDR3* de 1600Mhz i un disc dur *SSD*. Tot l'entorn crea quasi una desena de processos que amb una *CPU* de 8 fils i amb 3,9 Mhz en posició turbo pot assumir. Aquests processos tenen associats una gran quantitat de *RAM*. Amb 8GB, tenint en compte que tinc oberts documents i vídeos de diferents cursos mentre treballo, no és difícil trobar-me que, després de 4 ó 5 hores, tinc el consum a un 75%. S'ha d'indicar que *Windows 7* consumeix un 25% de la memòria només d'arrancar el sistema y que mesurant el consum de memòria només iniciar la primera vegada l'*Android Studio* –sense cap programa a banda- i el terminal virtual el consum passa a ser d'un 54%. La velocitat de la *RAM* és finalment el que em proporciona un rendiment alt, retardant al mínim possible el processador. S'ha de dir que part del desenvolupament l'he fet amb un terminal físic Nexus 4 degut a les limitacions gràfiques que presenta l'emulador i per la major agilitat del Nexus.

L'*SSD* m'ajuda tant en l'arrancada ràpida d'aplicacions com amb l'optimització de l'emulador de terminal. El sistema d'arxius del mateix i qualsevol acció del seu sistema operatiu es fa partint del fragment de disc dur que té assignat la màquina emulada.

En principi he utilitzat l'entorn de l'*Studio* sobre *Windows 7* i no sobre *Windows* per un tema de rendiment de la màquina emulada. Com vaig sobrat de potència amb el nou processador l'opció de *Linux* que em consumeix menys *CPU* no és determinant.

Hi ha un problema amb *Windows 7*, i és que, de moment, no es poden emular màquines amb mes de 768 MB ja que es torna inestable. Això és un handicap que assumeixo perquè a la versió *Linux* no hi ha versió per a emular el terminal sobre un *Atom* x86 amb la versió de Google optimitzada per x86, la qual és molt mes ràpida que treballar sobre una versió estàndard.

Per als requeriments de l'aplicació que volem desenvolupar, a banda del propi terminal, necessitem recursos externs per tal de treballar amb mapes i per fer consultes del temps. La decisió sobre els mapes era òbvia ja que *Google Maps* s'integra perfectament amb *Android* i es poden importar tot de classes directament en la nostra aplicació.

7.2. API GOOGLE MAPS

No es poden explicar totes les classes i mètodes de *Google Maps* però hem de dir que ìmportem "*com.google.android.gms.maps*". Enganxo una mica de codi totalment comentat en vermell per a poder entendre com funcionen els principals mètodes de l'*API*:

```
public class MostrarMap extends FragmentActivity implements LocationSource,        LocationListener,
GooglePlayServicesClient.ConnectionCallbacks,      GooglePlayServicesClient.OnConnectionFailedListener {
   private GoogleMap map = null;
   private LatLng posicion,posHome; […]
      posicion= new LatLng(latitude,longitude);
      posHome =  new LatLng(lathome,lonhome);
//inicialitzem opcions de polyline
      PolylineOptions  options= new PolylineOptions();
      options.add(posHome); //afegim un punt a la polyline a la posición de Home
      options.add(posicion); //afegim un punt a la polyline a la posición del mesurament
//configurem color amb el valor hexadecimal 0xbyte_tranparència+byte_red+byte_green+byte_blue. Hem escollit blau
marí.
      options.color(0xff0000ff);
      setUpMapIfNeeded();//inicialitzo el mapa si es necessari.
      map.addPolyline(options);//afegim la polyline per pintar-la al mapa.
      moveCamera();//coloquem la vista centrada en una localització del tipus Location que en diem locationview
   }
   private void setUpMapIfNeeded() { […]
// obtenim amb getMap() un GoogleMap  d'un SupportMapFragment que aconseguim amb getSupportFragmentManager.
      map = ((SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map)).getMap();
//configurem un tipus de mapa Hybrid on es veu el dibuix i la imatge de satel·lit. Es pot escollir entre diversos tipus de
mapa  MAP_TYPE_NORMAL    per  a  tipus  dibuix,  MAP_TYPE_SATELLITE  per  a  vista  només  per  satel·lit,
MAP_TYPE_TERRAIN per a mostrar el mapa amb información de relleu, MAP_TYPE_NONE sin capas per afegir notes.
      map.setMapType(GoogleMap.MAP_TYPE_HYBRID);
      // Si ja s'ha inicialitzat correctament el mapa fem el setup del mateix.
      if (map != null) {
         setUpMap();
      }
   }
   private void setUpMap() {
      map.setMyLocationEnabled(true); //activo la capa per representar la meva localització
      map.setLocationSource(this); //configuro la meva posició actual al mapa
[...]
   }
   private void setMarker(LatLng position, String titulo, String info,float color) {
      Marker myMaker = map.addMarker(new MarkerOptions()//color del marcador
            .position(position) //posició marcador
            .title(titulo) //títol del marcador
            .snippet(info) //descripció del marcador
            .icon(BitmapDescriptorFactory.defaultMarker(color))); //color del marcador
   }
   private void moveCamera() {
[…] cada vegada que hi ha un canvi fem el següent:
      }
      if(cambio) {// per a posicionar la càmera sobre la posición locationView utilitzem un objecte CameraPosition
         CameraPosition newCameraPosition = new CameraPosition.Builder()
 // configuro el centre a .target(LatLon).
//Calculo el valor amb la clase SphericalUtil.(LatLon, distància a veure, graus_desde_nord). Azimuth val 0º per defecte.
            .target(SphericalUtil.computeOffset(
                  new LatLng(locationView.getLatitude(), locationView.getLongitude()), 100, azimuth))
                  //Configuro el .zoom(float) calculant una vista apropiada
            .zoom((float) (16 + 3 * (Math.abs(tilt) / 90)))
            .bearing((float) azimuth//angle de la càmera partint de 0º al Nord.
            .tilt((float) clamp(tilt)) //angle de la càmera en alçada de l'observador.
            .build(); // torna un constructor CameraPosition.builder i l'utilitzem  per moure la càmera
         map.moveCamera(CameraUpdateFactory.newCameraPosition(newCameraPosition));
         cambio=false;}
      }
```

Figura 9: Codi comentat API GoogleMaps

7.3. API OPENWEATHERMAP.ORG

Respecte a les consultes meteorològiques, he trobat serveis webs sense *API* o només amb lectures a nivell regional o nacional. *OpenWeatherMap* m'oferia una *API* que podia funcionar amb *JSON*[26], *XML*[27] o *HTML*[28] amb icones integrades que expliquen la situació del temps gràficament. El plus que significa per una banda poder utilitzar les icones i per una altra que es tracta d'un servei a escala mundial em va fer decidir. A més, el funcionament de l'*API* és molt senzill, tal i com podem veure partint d'un exemple:

http://api.openweathermap.org/data/2.5/weather?&units=metric&lang=sp&mode=json&q=lat=41.533640&lon=2.036287

Figura 10: Consulta a l'API Openweathermap

En la part de l'adreça fins al "?" estem cridant l'*API*; a partir d'aquí afegim opcions amb **"*&*"**: la primera és **"*units=metric*"** per aconseguir els resultats en sistema mètric internacional i graus centígrads; l'opció **"*lang=sp*"** em dóna el comentari del temps en espanyol; el tipus de resposta que demanem és a **"*mode=*"** i pot tenir valor **"*json*","*xml*"** o **"*html*"** (escollim la primera); la pregunta es farà segons latitud i longitud tal com veiem al final de la consulta amb el text **"*q=lat=41.533640&lon=2.036287*"**. Doncs bé, el primer que ens podem preguntar és perquè *JSON*? *Android* implementa una llibreria molt bona per a objectes d'aquesta classe. Qualsevol cadena de text es pot passar com a argument per a generar un objecte de la classe *JSONObject* i després accedir a cada valor a través del nom de l'etiqueta de la variable. Un cop feta la consulta *http*, la resposta en forma de text s'utilitza per a generar un *JSONObject*. La sortida tipus (per a entendre com és el seu funcionament) és com aquesta:


```
{"coord":{"lon":108.45,"lat":11.94},
   "sys":{"message":0.0022,"country":"Vietnam","sunrise":1402611702,"sunset":1402657855},
"weather":[{"id":804,"main":"Clouds","description":"nubes","icon":"04d"}],
"base":"cmc stations",
"main":{"temp":30.373,
                "temp_min":30.373,
                "temp_max":30.373,
                "pressure":981.95,
                "sea_level":1019.28,
                "grnd_level":981.95,
                "humidity":57
},
"wind":{"speed":2.06,
                "deg":260.002
},
"clouds":{"all":92},
"dt":1402626056,
"id":1584071,
"name":"Thành Phố Đa Lạt",
"cod":200
}
```


Figura 11: Resposta JSON de l' API OpenWeatherMap

Com podem veure és molt fàcil de llegir la informació i realment no utilitzo totes les dades que m'ofereix el servei ja que no aporta més a l'aprenentatge ni a l'aplicació. Un detall que sí hem de comentar a banda fa referència a la icona que necessito carregar. En el meu cas, me la baixo no només per a mostrar-la sinó per a guardar-la com una cadena de bits i estalviar-me la consulta quan reviso el registre de mesures guardades. M'ho puc permetre perquè les icones no ocupen gaires bytes i és preferible guardar-les que gastar bateria al fer la consulta a internet. Un cop tenim l'objecte *JSON* llegim l'etiqueta "***icon***" i obtenim en el cas anterior el valor "***04d***". Per a descarregar l'arxiu i guardar-ho en una matriu de bytes. Fem la següent consulta: http://openweathermap.org/img/w/04d.png

El resultat ho guardem a una matriu com dèiem (amb notació en ***Java byte[]***) i després es fa un ***cast o parse***[29] per tal de transformar els bytes en una imatge.

8. DISSENY DE L'APLICACIÓ

Després de prendre les decisions sobre els serveis, el *hardware* i passar per un llarg procés d'aprenentatge es va fer la següent aproximació conceptual:

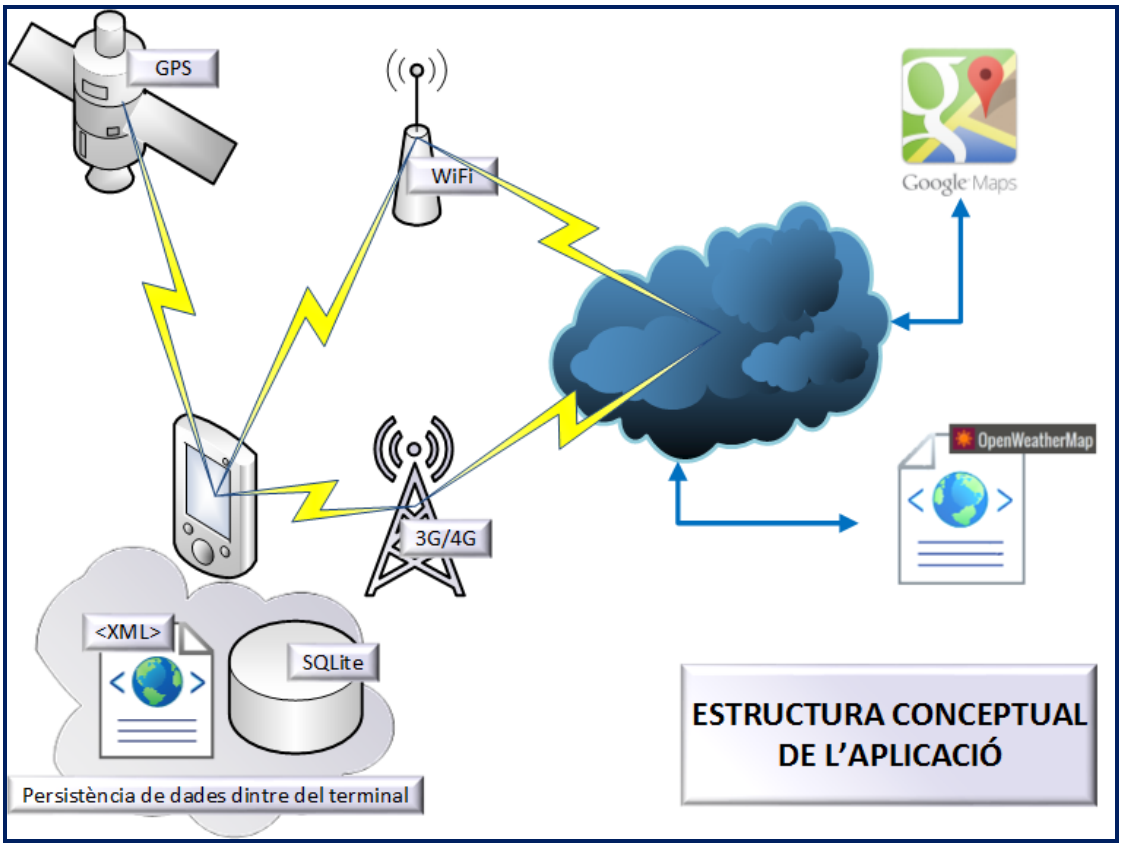

Figura 12: Disseny de l'aplicació

Bàsicament el terminal fa consultes GPS (pot arribar a aconseguir la posició per la *IP* de la *Wifi* o per la triangulació de les antenes *3G/4G*) i pot guardar la posició de *Home* en *XML*.

En el cas que es tracti d'una consulta de les dades meteorològiques utilitzarà la sortida a internet que sigui possible per tal de fer tres consultes a *OpenWeatherMap*. Aquestes seran les dades meteorològiques de:

- La posició *Home* (es llegeix aquesta posició de l'*XML* de la persistència de dades).
- La posició actual.
- La icona del temps en la posició actual.

Un cop recuperades totes les dades fem la persistència sobre una base de dades *SQLite* que integra la plataforma *Android* de forma nativa.

Si l'usuari vol representar sobre un plànol, ja sigui un mesurament o tot el registre, es fa una crida a la llibreria de *GoogleMaps* per tal de representar les marques de les posicions que l'aplicació llegeix de la base de dades. L'objecte creat per la llibreria passa per internet les posicions i els tipus de marques que es volen fer sobre el mapa. El servei de *GoogleMaps* respon des dels seus servidors al terminal amb el mapa que ha de mostrar, havent-hi un continu flux de dades mentre es manipula la sortida gràfica. Es tanca aquesta connexió quan sortim de la representació del mapa.

S'ha d'indicar que és permès per l'aplicació reescriure la posició de *Home* (òbviament no deixar-la buida) i esborrar una o totes les entrades del registre gràcies a sentències *SQL* de la nostre base de dades.

D'aquesta manera queden determinades les relacions entre els diferents serveis i les pròpies capacitats del terminal.

9. DIAGRAMES D'IMPLEMENTACIÓ

El diagrama de classes que he implementat és el següent:

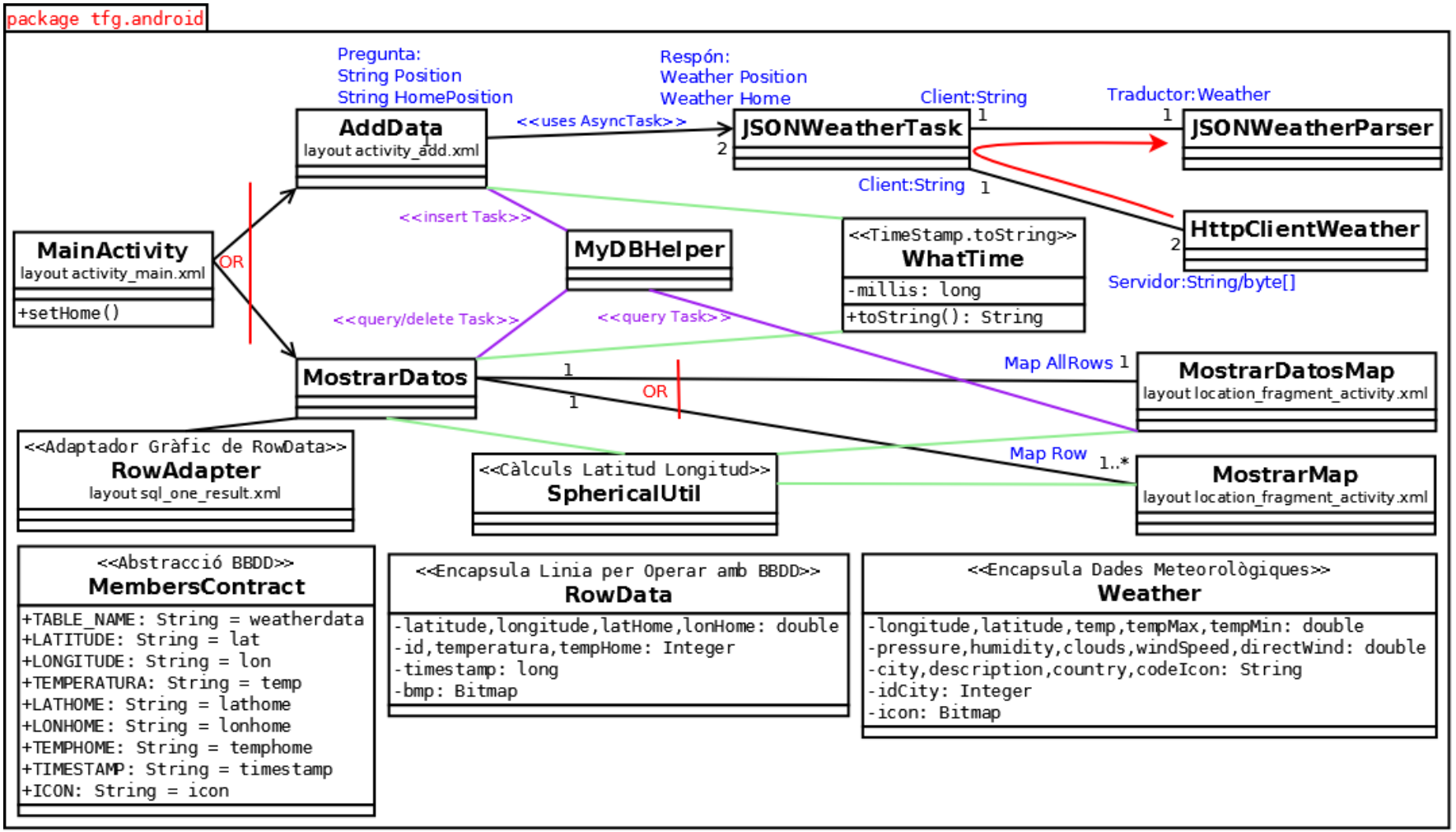

Figura 13: Diagrama de classes

El diagrama de seqüència de l'operació d'afegir és aquest:

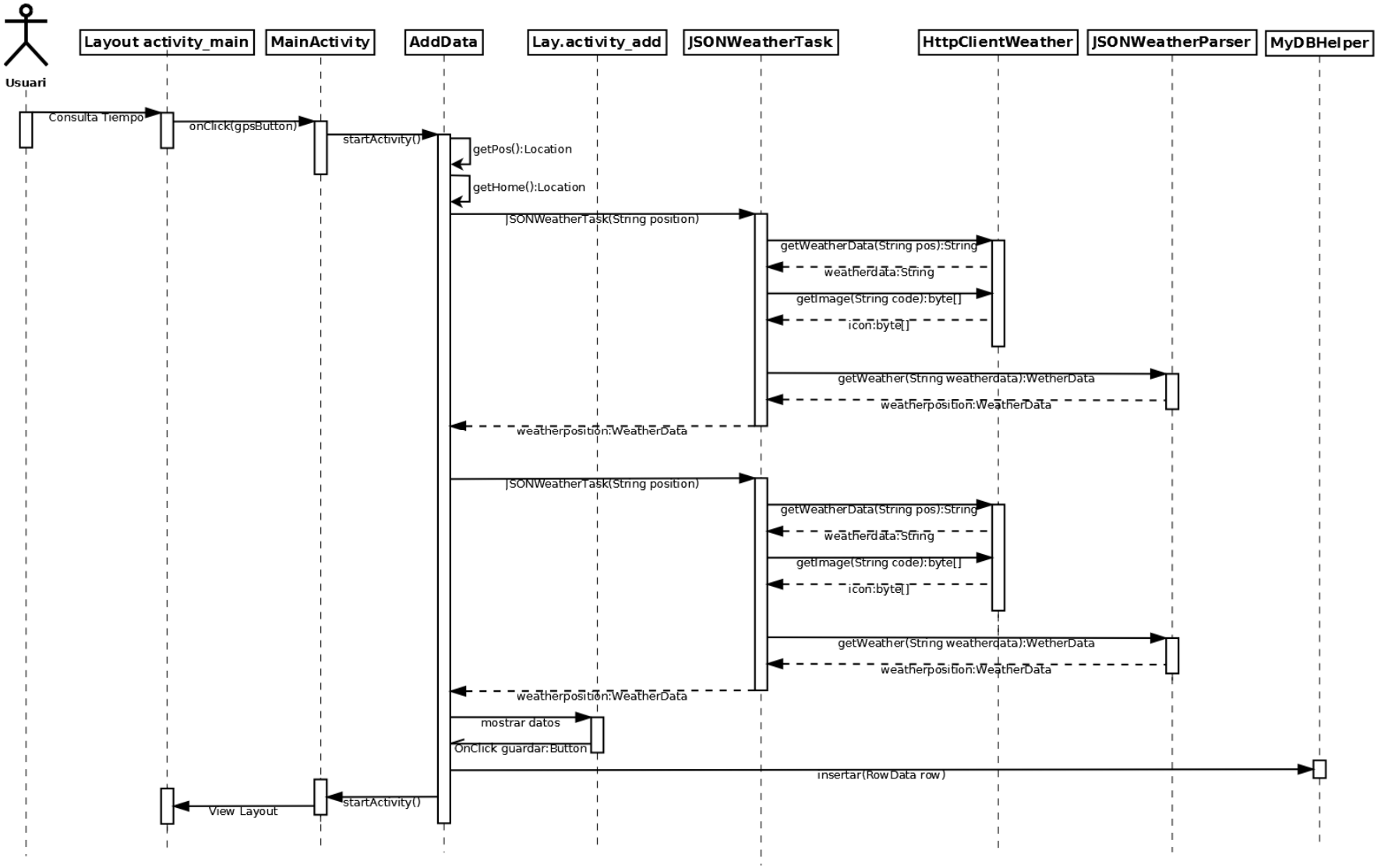

Figura 14: Diagrama de seqüència "Consulta el Tiempo"

El diagrama de seqüència de l'operació "mostrar dades" és aquest:

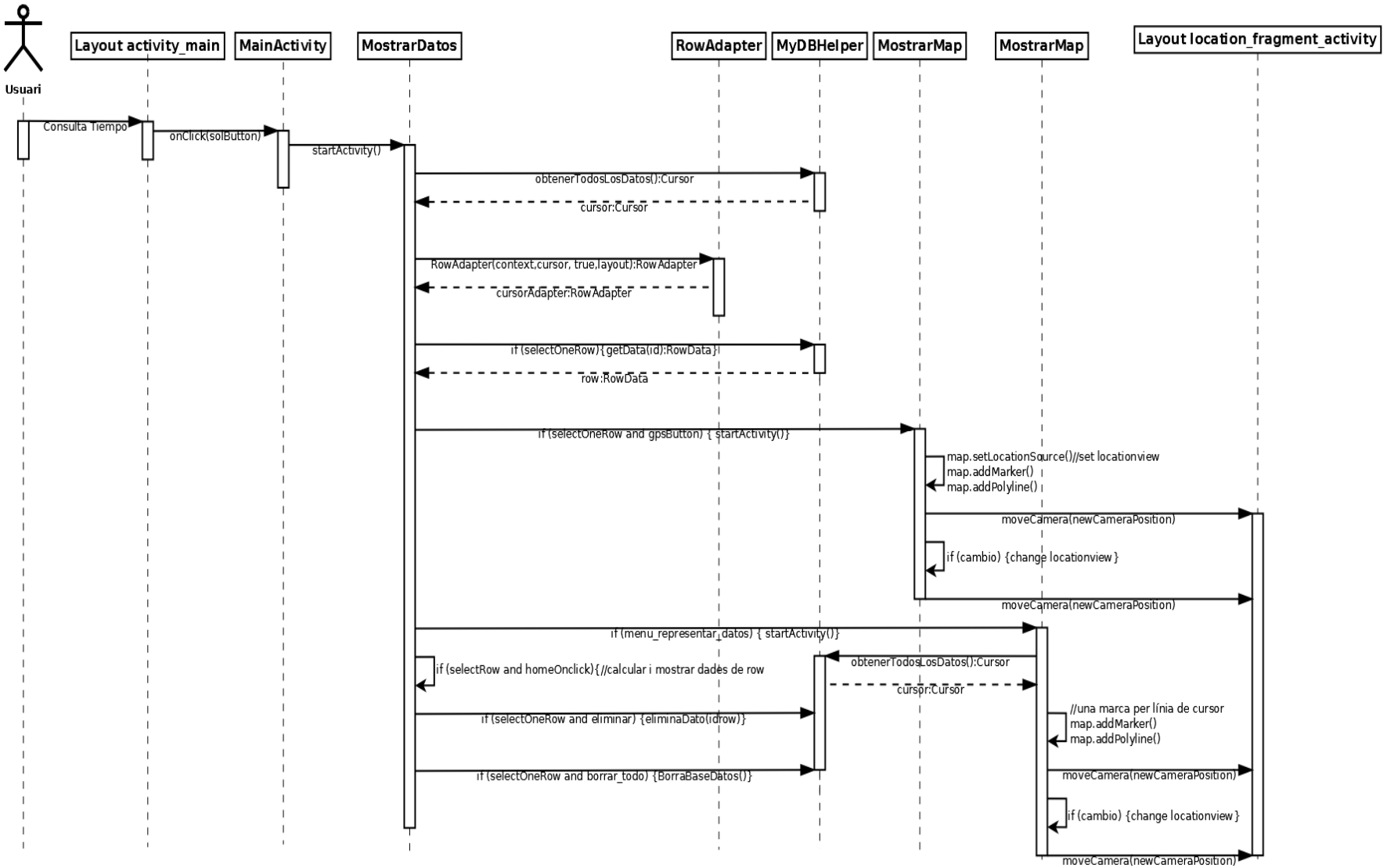

Figura 15: Diagrama de seqüència "Consulta Datos Guardados"

En primer lloc les classes que encapsulen dades per facilitar l'ús de grups de dades entre classes no les incloc en les relacions per facilitar la lectura. Per una altra banda ***MembersContract*** és tant sols una classe per a poder modificar fàcilment els noms de taula o columna sense haver de tocar res a nivell de programació de ***MyDBHelper***.

Les relacions que van cap a ***SphericalUtil*** (una classe proporcionada per Google que he importat al meu projecte) i ***WhatTime*** les he marcat en verd en el diagrama de classes. Tenen un paper secundari però com hi ha càlculs molt repetitius en diferents classes he preferit deixar els mètodes que utilitzo en classes a banda i cridar-les quan han fet falta. La primera d'aquestes classes m'ajuda en càlculs de distàncies entre punts geo-posicionats a partir d'una geometria esfèrica. La segona simplement em transforma l'hora en format de mil·lisegons a una marca de temps en format *String*[30] per a poder imprimir per pantalla.

Respecte a la resta de colors, els rols de cada classe els he marcat en blau. Les relacions entre classes per defecte van marcades en negre així com la cardinalitat entre elles. En verd (com ja he comentat) trobem les relacions de càlcul principalment. El violeta queda per a relacions que tenen a veure amb consultes a ***MyDBHelper***, que com veurem més endavant gestiona la base de dades que utilitzem. Com a últim color tenim el vermell que utilitzo per a indicar els fluxos entre classes i si els fluxos són excloents.

A *Android* cada tasca es pot dividir en ***Activities*** per tal de tractar les aplicacions com si fossin mòduls i poder alliberar memòria de forma dinàmica sense perdre el control de l'aplicació. Com hem vist en la descripció de l'*Studio*, a banda de les Activitats tenim la possibilitat de programar representacions gràfiques anomenades *layout*[31] o menú, les quals, durant les declaracions de les Activitats, es relacionen amb les mateixes. Dintre dels menús o *layouts* es pot determinar què és *clickable*. Quines accions s'han de portar a terme un cop es fa click és quelcom que programem a l'*Activity* relacionada. En el diagrama hem afegit com a comentari de les classes implicades les *layouts* principals.

L'aplicació neix a la classe **Main*Activity*** i permet apagar, o determinar la posició actual com la posició *Home* gràcies al mètode ***setHome()***. Per a consultar el temps o mostrar les dades, l'activitat ***Main*** crida a ***MostrarDatos*** o ***AddData***. En aquests dos casos, l'activitat cridada pren el control de l'aplicació.

En el cas d' ***AddData*** pren la posició actual GPS (o amb menys precisió pot utilitzar posicionament 3G) i llança una tasca asíncrona que mor quan acaba el seu propòsit.

Aquesta tasca és l'activitat ***JSONWeatherTask***, que és cridada dues vegades amb la posició actual i la posició de *Home* en format *String*. En cada cas torna un objecte de la classe ***Weather*** amb les dades meteorològiques corresponents.

JSONWeatherTask aconsegueix això fent un parell de consultes a ***HttpClientWeather***, una la del temps i l'altra la icona corresponent. Aquesta última classe fa les consultes a ***OpenWeatherMap.org*** respectant el format que l'*API* necessita per a tornar la informació. Com es pot veure al diagrama, depenent de si és una consulta del temps o d'icona, es torna un *String* o una matriu de bytes. Un com s'ha rebut la informació en format *String* de *OpenWeatherMap.org* es passa a ***JSONWeatherParser***, que gràcies a les llibreries d'*Android* transforma el text rebut en un ***JSONObject*** i d'aquí omple les dades d'un objecte de la classe *Weather*. Un cop de tornada a *JSONWeatherTask* s'hi afegeix la icona en format Bitmap a l'objecte Weather rebut i es passa com resposta a la classe ***AddData***.

De tornada a *AddData*, es "grafiquen" les dades a la *layout* ***activity_add.xml*** i es dóna la oportunitat per a guardar la dada. Si es vol guardar amb l'objecte ***Weather*** es crea un objecte ***RowData***, es passa a ***MyDBHelper*** i l'*Activity* es tanca i torna el control a la ***MainActivity***.

En el cas d'anar a **MostrarDatos** obtenim tots els mesuraments de la base de dades a través de la classe ***MyDBHelper*** (en cas d'esborrar un mesurament o tots ho demanem a la mateixa classe i tornem a "graficar" la sortida). El primer que crida l'atenció és que no té una *layout* associada i és perquè s'estén d'una activitat especial que implica una representació en forma de llista. Aquesta activitat es diu ***ListActivity***.

El primer que es fa, doncs, és obtenir totes les dades en una mena d' "iterador" anomenat "Cursor". Un cop tenim les dades anem llegint línia a línia de la base de dades i cada cop que llegim una línia o ***Row***, utilitzem ***RowAdapter*** per a donar format gràfic a la mateixa. Aquesta sortida gràfica que té els seus propis botons i funcionalitats es va afegint a la llista a mostrar que està implícita a la classe ***ListActivity*** fins a formar la llista sencera en pantalla.

MostrarDatos calcula i dóna opcions per a navegar-hi. Quan es demana representar un mesurament en concret, es passen a l'activitat **MostrarMap** les dades de posició del mesurament i de *Home* i aquesta última activitat pren el control i "gràfica" els punts amb l'*API* de GoogleMaps. Si el que es vol és representar totes les dades, és MostrarDatosMap qui demana a la base de dades (sempre a través de *MyDBHelper*) mesurament per mesurament i cada posició l'afegeix per a "graficar" igual que la classe anterior.

10. ANÀLISI PROJECTE

10.1. PRIMERS PASSOS

Inicialment, vaig fer una aproximació general força bona del que em caldria i quins serveis utilitzaria. L'estructura conceptual del projecte s'ha mantingut força ferma des del començament però hi ha dues coses que vaig adonar-me aviat que no havia mesurat bé.

- La primera cosa que em va resultar evident en els primers passos va ser la necessitat d'avançar la compra de *hardware* per actualitzar el meu PC de treball. En un inici vaig pensar que durant l'aprenentatge amb exemples senzills no em calia molta potència de computació, però va ser evident que el propi entorn de desenvolupament demandava més capacitat. No depenia del que programés sinó que els requeriments base són alts.

- En segon lloc, al final de la primera setmana d'aprenentatge em vaig començar a adonar de l'enormitat del camp que estava intentant assumir, tot i partint d'un bon nivell de *Java*. Sincerament vaig pensar que, acostumat com estava a programar, no suposaria una corba d'aprenentatge tant lenta. El nivell de les meves pròpies capacitats era semblant a estar amb aigua fins als turmells tenint per endavant l'Ebre i havent-lo de creuar nadant. Per això, després del primer ensurt, vaig posar fil a l'agulla per arribar el mes lluny possible.

10.2. PLANIFICACIÓ

La planificació ha estat mes o menys encertada exceptuant dos punts que han desquadrat tot el que venia a continuació. Això ha estat mes notable aquest desajust, ja que aquests punts eren aparentment dels primers a tractar, a banda de la planificació inicial.

- El primer punt (que és el que ha presentat un major grau d'error) ha estat el format pels 20 dies que vaig comptar per a l'aprenentatge. Estimant 2 hores per dia tenia la intenció que en 40 hores l'aprenentatge estaria fet. Però tenint en compte les hores de re-aprenentatge durant la implementació del codi probablement hauria hagut d'estimar entre 200 i 240 hores. Això ha creat un daltabaix a la programació de temps feta i ho he pogut equilibrar ja pràcticament al final, invertint les hores necessàries en jornades maratonianes.

- El segon punt, que ja he comentat en apartats anteriors, ha estat la necessitat d'avançar en el temps la compra de hardware, a causa de les exigències de l'entorn necessari.

10.3. DIFICULTATS

La dificultat més gran de tot plegat ha estat la ingent quantitat d'hores que s'han d'invertir per a fer un codi convenientment afinat. S'han de preveure les càrregues de *RAM*, el consum de *CPU* i la quantitat de dades que es consulten a internet.

Tot això obliga a prendre decisions sobre el diag*ram*a bàsic de dades i, per tant, arreglar una mala decisió modifica totalment el diagrama i la forma de fer. Probablement amb més experiència es podrien afinar encara més els requeriment però he tractat de fer-ho seguint l'exemple d'experts. Sóc conscient de les meves limitacions, però el projecte m'ha ofert l' oportunitat de fer una base de coneixement suficient per a començar amb petits projectes de manera professional.

Una complicació que no puc deixar d'esmentar ha estat l'obsolescència del codi penjat amb exemples a les pàgines oficials o en cursos i recursos a tot el món. Encara que de tot és possible trobar un exemple relativament ràpid, també és exageradament freqüent que codi penjat relativament fa poc no funcioni en la versió d'*API* actual, la 19 en el meu cas. Això fa que s'aprenguin malament coses directament de fonts oficials i que no sigui fins que ho implementis al teu codi que no vegis que no funciona. A partir d'aquest moment s'ha de buscar per un rosari de fòrums les causes del perquè no funciona i com se soluciona. Aquestes recerques en algun cas m'han portat dies sencers durant la implementació del codi ja que durant l'aprenentatge prenia el tema com a superat amb petits exemples que només a vegades funcionaven. Aquesta falta d'actualització de les fonts oficials a nivell d'exemples d' implementació s'ha de dir que a nivell d' "*API reference*" normalment sí estava actualitzat a nivell de classes noves, encara que no sempre en les classes velles quedava clar que no es podien fer servir. Com a justificació s'ha de dir que la documentació és exageradament gran i no és difícil que passin coses com les comentades.

Respecte a l'accés al *hardware* del dispositiu, moltes vegades és complicat i s'ha de seguir un ordre precís d'instàncies de diferents classes per a finalment demanar, per exemple, la posició actual del nostre terminal. Aquests accessos haurien de millorar en el futur, deixant

unes *API* més transparents en els passos intermedis, deixant les opcions per defecte habituals i obrint la possibilitat d'afinar pel camí llarg si cal.

Malgrat tot, amb molta insistència i gràcies a la publicació de codi i vídeo tutorials penjats desinteressadament per programadors d'arreu s'ha pogut fer front a les dificultats.

10.4. DESVIAMENTS I CORRECCIONS

En referència a les **desviacions d'objectius**, puc dir que pràcticament no n'he tingut cap, exceptuant alguna millora gràfica i control d'errors més personalitzat... però això s'ha traduït en importants **desviacions respecte les previsions de temps**.

La finalització del Codi estava prevista per al 21-05-2014 i en aquell moment tenia greus problemes d'estabilitat a l'aplicació i encara no disposava d'una bona representació de les dades sobre un mapa. També tenia problemes amb el *RowAdapter* i els botons que implementava cada fila. En aquell moment ja hi havia invertit més del doble d'hores esperades i l'angoixa per les dates de presentació estaven minvant les esperances de crear un bon codi.

Arribats a aquest punt haig d'agrair molt l'actitud del meu consultor, el **Dr. Francesc Guim**, qui em va ajudar a reorientar els meus objectius cara a l'entrega final i la memòria, treballant a fons dins el període de correccions en què tenia previst haver pogut finalitzar el projecte de forma fiable i assolint els màxims objectius inicials possibles.

Finalment vaig aconseguir el codi funcional del projecte el 4-6-2014, fent una primera entrega del codi al meu consultor. Després de poder provar el codi l'explicaré en el següent punt en diversos dispositius.

He pogut corregir tots els problemes i el codi es presenta estable. A vegades, la solució ha passat per un diagrama de classes diferents, utilitzant classes asíncrones o especialitzacions de la classe *Activity* com *ListActivity* o *FragmentActivity*. La utilització d'altres classes provocava inestabilitats que només un programador amb experiència pot conèixer.

Gràcies a mantenir un ritme fort de feina i als consells del meu consultor he pogut corregir el que impedia un nivell d'excel·lència suficient per al projecte que havia imaginat en un principi.

10.5. PROVES DE FUNCIONAMENT

Les primeres proves les he realitzat sobre tots els dispositius *Android* que comercialitza Google actualment mes el Nexus 4, el qual ja ha estat substituït pel Nexus 5. També s'ha provat sobre Nexus 7 i 10. A nivell de dispositius físics he pogut provar amb un Nexus 4 amb Andorid 4.4, un Huawei G300 4.0 i modificant una mica el codi per a que llegís la posició únicament de la connexió *Wifi* una tablet Woxter Quimbus 98q amb *Android* 4.2 que no disposa ni de GPS ni de 3G.

En general he obtingut bons resultats en quan a visibilitat i funcionament, però millorables en pantalles de 5" quan es visualitzen horitzontalment; tot i això es poden veure bé i són funcionals.

El Huawei ha estat una mica més problemàtic, doncs la versió que incorpora d'*Android* i la personalització del fabricant fa que no funcioni correctament i es pengi sovint l'aplicació. A més és un terminal que no he pogut portar a casa per a connectar-ho al *PC* i veure el *log*. Com no he tingut aquesta opció i el testimoni de l'usuari habitual del terminal és que li passa de forma habitual amb aplicacions que teòricament funcionen en la seva versió, he donat per bones les proves. Actualment el terminal esmentat no tenia espai lliure a la memòria interna ni a la tarja SD, i no podia demanar a l'usuari que buidés coses per a fer les proves. És una pena no haver pogut provar amb més dispositius, però no n'he tingut més al meu abast.

Malgrat tot, la bateria de proves fetes provant totes les opcions de l'aplicació de forma vertical i horitzontal, crec que és suficient per a una aplicació com la que proposo al projecte i que no és crítica en el seu ús.

10.6. CONCLUSIONS

Un cop finalitzat el projecte és el moment de fer recompte. He de dir que, realment, em sento satisfet del volum de cosses que he pogut aprendre sobre programació *Android*.

Puc dir que, dels objectius esmentats a l'apartat 3.1, els he assolit tots, exceptuant el fet d'afitar-me en els marges temporals, on vaig quedar sobrepassat de bon principi.

- He pogut conèixer a fons l'arquitectura d'*Android* i l'entorn de desenvolupament *Android Studio*.

- El fet d'utilitzar l'*API* per al GPS i la part de representació gràfica ha estat part fonamental del projecte. I encara que de la part gràfica em queda camp per explorar he assolit un nivell acceptable per a la majoria d'aplicacions d'empresa.
- En l'aplicació faig servir *Google Maps* i el servei Web de dades meteorològiques de *OpenWeatherMap.org*. No només he aprés a consultar, sinó a treballar la resposta amb un format que no coneixia com és el *JSON*.
- Ara puc, de forma fluïda, crear *layouts* amb programació gràfica o *XML*. He aprés a relacionar *layouts* i elements de les mateixes amb classes per a reaccionar davant d'accions de clicar o passar per sobre.
- He intentat aplicar els principis de creació d'interfases gràfiques. En quant a l'apartat gràfic estic força satisfet; he optat per un estil desenfadat, proper a les aplicacions personals i amb una interfície que intenta ser divertida i amable. Haig de dir que en aquest aspecte les llibreries de *Android* són una mina i, tot i que encara la mecànica no és massa intuïtiva, l'*API* és molt potent.
- Pel que fa a l'experiència de planificació i afinació, he d'acceptar que en les meves prediccions de temps he fracassat estrepitosament, però fent una lectura positiva he après molt de cara a futurs projectes. Aprendre un llenguatge nou, sí que em donava respecte, però el fet de tractar-se d'una *API* em va fer ser massa optimista. D'ara endavant, hi ha criteris que es poden prendre per a valorar de forma més ajustada el temps d'aprenentatge com són el nombre de classes que té l'*API* o el llarg mitjà en hores de vídeo tutorials sobre el tema multiplicat per 3 (per a poder programar els teus exemples). Realment aquest aspecte l'he aprés "per les dolentes", però sens dubte el que he vist no ho oblidaré mai.

Ha estat molt interessant el poder programar per a un sistema operatiu que treballa sobre dispositius mòbils. L'oportunitat d'accedir tant directament a part del *hardware* i l'obligació de tenir presents les limitacions de *hardware* m'ha obligat a pensar en l'arquitectura en cada tasca que he hagut que fer.

Finalment quedo satisfet del projecte en termes d'aprenentatge i d'aplicació de principis acadèmics apressos durant els estudis a la UOC. Penso que l'experiència és totalment exportable al món laboral i quedo molt agraït per l'oportunitat de treballar en un camp tant pràctic com aquest. Agraeixo l'esforç i el recolzament del meu consultor i quedo a disposició del tribunal per a qualsevol aclariment.

11. NOTES A PEU DE PÀGINA

1: **Blackberry OS**: És un sistema operatiu que basa el seu potencial en un *hardware* amb teclat complet. Respecte al paradigma d'Apple, d'un únic botó per a controlar-ho tot és quasi el seu antagònic. http://es.blackberry.com/software/smartphones/blackberry-7-os.html#tab-1

2: http://www.apple.com/es/ios/

3: La seu d'Apple es troba a la localitat de Cupertino, a l'estat de Califòrnia, USA.

4: És un sistema operatiu que va desenvolupar Nokia i que actualment ha estat substituït per *Windows Phone*, després de la compra par part de Microsoft de part de l'accionariat de Nokia. Encara que actualment està en desús, compta encara amb un grup de seguidors nostàlgics. http://www.wayerless.com/2013/01/hasta-siempre-symbian-2007-2013/

5: La història de la creació de *MacOS* i com es van copiar molts conceptes per part de Microsoft per a crear el seu sistema operatiu *Windows* és el principal tema de la pel·lícula Pirates of Silicon Valley. https://www.youtube.com/watch?v=-m1pqqVbgWQ

6: http://www.android.com/

7: http://opensource.org/

8: http://es.wikipedia.org/wiki/Objective-C

9: MS-DOS és el primer sistema operatiu que va llançar MicroSoft. http://es.wikipedia.org/wiki/MS-DOS

10: http://es.wikipedia.org/wiki/Televisi%C3%B3n_inteligente

11: http://es.wikipedia.org/wiki/Sistema_de_posicionamiento_global

12: 2G, 3G i 4G són els diferents estàndards de connexió mòbil a dades ordenats en el temps. Actualment el 4G està en desplegament en el nostre país.

13: NFC és una tecnologia sense fil semblant al bluetooth però d'un abast de pocs centímetres. http://es.wikipedia.org/wiki/Near_Field_Communication

14: http://es.wikipedia.org/wiki/API

15: Són serveis que ofereixen webs per a utilitzar-se en les pròpies aplicacions cridant als mètodes que ofereixen. http://en.wikipedia.org/wiki/Web_service

16: *Google Maps* ofereix localització sobre un plànol i diferents càlculs a partir de cerca o posicions gps. Te la seva pròpia *API*. https://developers.google.com/maps/?hl=es

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

18: https://developers.google.com/maps/

19: http://www.openweathermap.org/current

20: http://francho.org/tag/curso-unutopia-android/

21: http://formacion.citipa.org/

22: https://github.com/

23: Es un entorn gràfic de programació per a un llenguatge o varis que implementa eines de compilació, depuració i proves. http://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado

24: Els logs o log de un sistema o servei es on es guarda un registre de l'activitat del mateix. Si quelcom no funciona o s'ha realitzat alguna acció això afegeix una línia amb l'hora del esdeveniment a aquest arxiu normalment de text. Això es molt útil quan s'ha de revisar el comportament d'una aplicació o sistema i trobar l'origen d'errades o posar data a accions concretes.

25: L'emulació es el procediment per el qual, a través d'un programari es modela de forma precisa el comportament d'un maquinari o sistema operatiu sobre un maquinari o sistema diferent.

26: *JSON* es un mètode lleuger d'intercanvi de dades basat en *JavaScritp* i que molt sovint es fa servir en comtes de *XML* encara que s'assemblen molt. http://es.wikipedia.org/wiki/Json

27: *XML* es un llenguatge per a guardar dades en mode text. http://es.wikipedia.org/wiki/XML

28: *HTML* es el llenguatge base per a una web tradicional. http://es.wikipedia.org/wiki/HTML

29: Un cast o parse es el mètode que transforma un objecte en un altre si la forma del objecte permet la transformació o existeix una convenció per a la transformació.

30: *String* és una classe *Java* que fa referència a cadenes de caràcters. S'utilitza per a treballar amb text i permet l'emmagatzemament del mateix així com implementa mètodes per a modificar-ho.

31: En el context que utilitzem la paraula *layout*, significa l'arxiu que defineix la capa de sortida gràfica com es pot despendre del significat en anglès de les paraules "lay"/capa i "out"/sortida.

12. BIBLIOGRAFIA I RECURSOS UTILITZATS

El programari utilitzat ha estat el següent:

- Microsoft Project 2010
- Microsoft Office 2010
- Microsoft Visio 2013
- Microsoft Office Picture Manager
- Gimp: per a retoc d'imatges. http://www.gimp.org.es/
- Dia: per a diagrames UML de classes. http://www.dia-installer.de/

Les fonts d'informació han estat les següents:

- Materials PDF UOC: Redacció de textos cientificotècnics
- Pla docent de l'assignatura TFG i comentaris i exemples oferts
- *API Android*: http://developer.android.com/develop/index.html
- *API GoogleMaps*: https://developers.google.com/maps/
- *API OpenWeatherMap*: http://www.openweathermap.org/current
- Curs d'*Android* de FranchoLab: http://francho.org/tag/curso-unutopia-android/
- Curs OnLine d'*Android* Avançat del Col·legi d'Enginyers Tècnics d'Informàtica del Principat d'Asturies: http://formacion.citipa.org/
- GitHub: https://github.com/