



On aparcar?

Memòria de Projecte Final de Grau

Grau de Multimèdia

Desenvolupament d'aplicacions interactives

Autor: Albert Juncà Tatjé

Consultor: Kenneth Capseta Nieto

Professor: Carlos Casado Martinez

24/06/2014

Document i aplicació amb llicència Creative Commons Reconeixement – NoComercial - SenseObraDerivada 4.0 Internacional. No se'n permet l'ús comercial ni la creació d'obres derivades.

Albert Juncà Tatjé



Aplicació desenvolupada amb les *Android Developer Tools*:

<http://developer.android.com/sdk/index.html>

Per al desenvolupament del projecte, s'han utilitzat les següents *Application programming interface* (API):

- Google Directions API:
<https://developers.google.com/maps/documentation/directions/>
- Google Geocoding API:
<https://developers.google.com/maps/documentation/geocoding/>
- Google Maps Android API:
<http://developer.android.com/google/play-services/maps.html>

Codi de tercers utilitzat o adaptats:

- **JSONParser.java** de Ravi Tamada, amb llicència Creative Commons Reconeixement 3.0 No adaptada (CC BY).
- **Decoding Polylines from Google Maps Direction API** de Jeffrey Sambells.
- **Decoding Google Maps Direction API JSON and draw directions on map** de George Mathew.

Imatges de tercers utilitzades o adaptades:

- **Parking lot at Houston airport.** de Addison Berry amb llicència Creative Commons Reconeixement, No comercial i Compartir Igual 2.0 Generic.
(<https://www.flickr.com/photos/add1sun/3466383400/>)

Als meus pares, Joan i Balbina,
al meu germà David
i tota la meva família,
per tot el suport rebut al llarg del grau.

Abstract

El projecte que es desenvoluparà al llarg d'aquesta memòria, consisteix en intentar utilitzar les tendències a la participació i col·laboració, que han aparegut en les xarxes socials, per tal que amb l'ajuda d'altres usuaris poder realitzar una tasca tan quotidiana com trobar lloc per aparcar el cotxe.

Per diversos motius, a vegades és necessari utilitzar el cotxe, enlloc del transport públic, per a desplaçar-nos. El problema d'utilitzar-lo en una ciutat més o menys gran és on aparcar-lo, provocant, en molts cassos, una pèrdua de temps. És en aquest punt on s'ha pensat que, vistos els avenços tecnològics, es podria buscar una manera de dinamitzar aquesta tasca.

“On aparcar?” pretén utilitzar la idea de les xarxes socials, enfocant el seu ús per tal de que els usuaris es puguin ajudar entre ells, en aquest cas, per trobar un lloc on aparcar. Per tal d'ajudar-se entre usuaris, aquests podran marcar una ubicació en el mapa on hi hagin llocs lliures per aparcar, d'aquesta manera un usuari que busqui aparcament en una zona determinada, podrà consultar la informació i utilitzar-la.

Paraules clau: Memòria, Treball de Fi de Grau, aparcament, puntuació, Android, Smartphone, Java, PHP, MySQL, Google Maps, GPS, geolocalització.

Notacions i Convencions

A continuació es llisten les diferents tipografies i estils, que s'utilitzaran al llarg del document per a diferenciar els diferents continguts.

- **Títols de capítols:** Arial, negreta, 20 punts i color blau.

- **Títols de capítols**

- **Subtítols de capítols:** Arial, negreta, 13 punts i color blau.

- **Subtítols de capítols**

- **Text normal:** Arial a 11 punts.

- Text normal

- **Peus d'imatge o taula:** Arial a 8 punts.

- Peus d'imatge o taula

- **Nota peu de pàgina:** Arial a 9 punts.

- Nota peu de pàgina

- **Línies de codi:** Courier New a 9 punts.

- Línies de codi

Índex

1. Introducció.....	10
2. Descripció.....	11
3. Objectius	12
3.1 Personals	12
3.2 Professionals.....	12
4. Marc teòric.....	13
5. Continguts	14
6. Metodologia.....	16
7. Arquitectura de l'aplicació.....	18
7.1 Client.....	18
7.2 Bases de dades.....	18
7.3 Servidor Web.....	18
8. Plataforma de desenvolupament	19
8.1 Programari.....	19
8.2 Maquinari	20
9. Planificació	21
9.1 Dates clau	21
9.2 Fites	21
9.3 Diagrama de Gantt.....	22
10. Procés de treball	23
10.1 PAC 2 – Disseny del projecte.....	23
10.2 PAC 3 – Desenvolupament del projecte i prototip.....	27
10.3 Entrega final – Proves, desenvolupament final i lliurament de l'aplicació	32
11. APIs utilitzades	36
12. Diagrames UML	37
13. Prototips	38
13.1 Wireframes	38
13.2 Hi-Fi	41
14. Perfils d'usuari	44
15. Usabilitat.....	45
15.1 Formes d'interacció.....	45
15.2 Navegació	46
16. Seguretat.....	47
16.1 Base de dades.....	47
16.2 Formularis	47
17. Tests	48
18. Versions de l'aplicació/servei.....	49
18.1 Alpha.....	49
18.2 Beta	49
18.3 Versió 1.0	49
19. Requisits d'instal·lació.....	50
19.1 Client.....	50
19.2 Servidor.....	50
20. Instruccions d'instal·lació/implantació.....	51
20.1 Dispositiu real	51
20.2 Emulador Genymotion.....	52
21. Instruccions d'ús	53
21.1 Inici	53
21.2 Menú	54
21.3 Pantalla Mapa.....	55
21.4 Perfil usuari.....	58
22. Bugs.....	60
23. Projecció a futur.....	61
24. Pressupost.....	63
25. Anàlisi de mercat	64
25.1 Audiència potencial.....	64
25.2 Competència	64
26. Conclusions.....	65
Annex 1. Lliurables del projecte.....	66
Annex 2. Codi font (extractes)	68
Annex 2.1 Manifest.xml.....	68
Annex 2.2 LoginActivity.....	69
Annex 2.3 Carregar Google Maps	71

Annex 2.3 Carregar Google Maps	71
Annex 2.4 Canvi dígit dades GPS.....	72
Annex 2.5 Afegir marcadors diferents.....	72
Annex 2.6 Consultes SQL.....	73
Annex 3. Codi extern utilitzat.....	74
Annex 3.1 JSONParser.....	74
Annex 3.2 Decoding Google Maps Direction API JSON	76
Annex 3.3 Decoding Polylines from Google Maps Direction	78
Annex 4. Captures de pantalla	79
Annex 5. Glossari	83
Annex 6. Bibliografia.....	84

Figures i taules

Índex de figures

Figura 1: Representació concepte “On aparcar?” – Albert Juncà i Addison Berry (CC BY-NC-SA 2.0)	10
Figura 2: Esquema metodologia en cascada.....	16
Figura 3: Esquema metodologia en cascada retroalimentada	16
Figura 4: Estructura base de dades	18
Figura 5: Diagrama de Gantt	22
Figura 6: Android SDK Manager.....	24
Figura 7: Projecte Android “On aparcar?”	24
Figura 8: Prova registre dades aplicació i MySQL	25
Figura 9: Importació llibreria Google Play Services	26
Figura 10: <i>Actionbar</i> funcionant.....	28
Figura 11: Mostra icona i marcadors mapa	32
Figura 12: Diagrama UML	37
Figura 13: Pantalla principal – Identificació	38
Figura 14: Pantalla registrar nou usuari	38
Figura 15: Pantalla menú	39
Figura 16: Pantalla Mapa	39
Figura 17: Pantalla definir destí	39
Figura 18: Pantalla afegir aparcament	39
Figura 19: Pantalla ocupar aparcament	40
Figura 20: Pantalla perfil altres usuaris	40
Figura 21: Pantalla perfil propi.....	40
Figura 22: Avís als conductors	41
Figura 23: Pantalla principal.....	41
Figura 24: Pantalla registrar	41
Figura 25: Menú	41
Figura 26: Mapa	42
Figura 27: Finestra “Definir destí”	42
Figura 28: Finestra “Marcar aparcament”	42
Figura 29: Finestra “Anar / Ocupar aparcament”	42
Figura 30: Finestra “Perfil usuari”	43
Figura 31: Finestra “Informació”	43
Figura 32: Esquema navegació	46
Figura 33: Aplicació copiada al dispositiu.....	51
Figura 34: Procediment instal·lació aplicació.....	51
Figura 35: Pantalla principal Genymotion.....	52
Figura 36: Pantalla identificació	53
Figura 37: Pantalla registre	53
Figura 38: Pantalla canviar contrasenya	54
Figura 39: Pantalla menú	54
Figura 40: Pantalla mapa	55
Figura 41: Exemple guardar posició cotxe	55

Figura 42: Tornar al cotxe	56
Figura 43: Marcar aparcament	56
Figura 44: <i>InfoWindow</i> del marcador	57
Figura 45: Anar a l'aparcament	57
Figura 46: Ocupar aparcament.....	57
Figura 47: Llista resultats	58
Figura 48: Definir destí.....	58
Figura 49: Accedir al perfil propi	58
Figura 50: Accedir perfil altres usuaris	59

Índex de taules

Taula 1: Tipus de marcadors.....	56
Taula 2: Pressupost.....	63
Taula 3: Pressupost total.....	63

1. Introducció

Tot i la feina realitzada per les administracions, per tal de millorar i incrementar el tipus de cobertura de mitjans de transport públics, en moltes ciutats aquesta resulta insuficient per a determinades circumstàncies. Ja sigui per temes d'horaris, cobertura reduïda o comoditat, molta gent continua desplaçant-se utilitzant cotxe propi. El problema dins de ciutat arriba quan toca aparcar el cotxe, ja que molts cops es converteix en una odissea i pèrdua de temps degut a la quantitat de cotxes en ciutat.

Tenint en compte que a la societat actual, on ja s'observa que el present i futur de les relacions entre persones passa per a l'ús de les xarxes socials, en gran part gràcies als nous dispositius mòbils, permetent estar constantment connectats, s'ha pensat en aprofitar el potencial que aquestes tenen i enfocar la seva utilització en la col·laboració entre els usuaris.

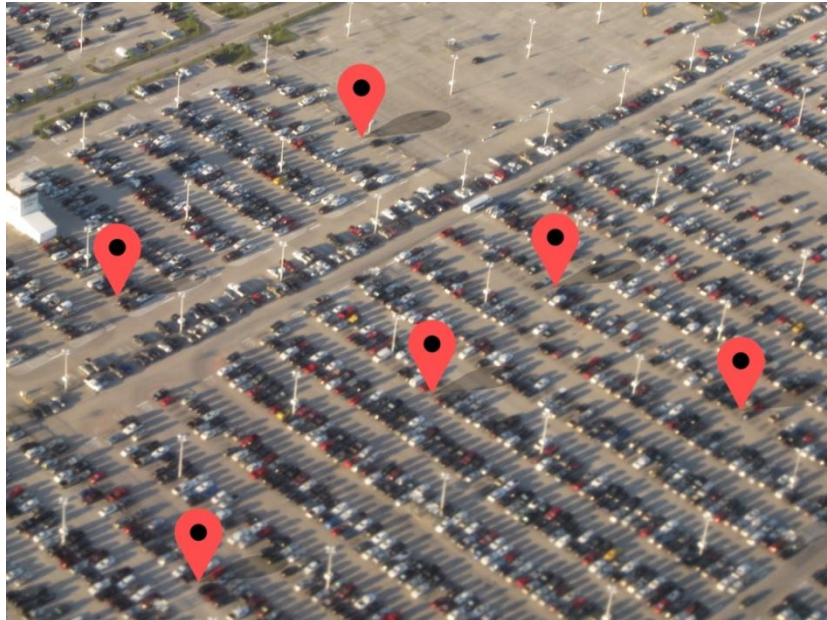


Figura 1: Representació concepte "On aparcar?" – Albert Juncà i Addison Berry (CC BY-NC-SA 2.0)

Així doncs, tenint un problema, un nou sistema de comunicació en augment i els dispositius mòbils, es podria aprofitar la combinació d'aquests elements per tal de que entre tots puguem informar dels llocs disponibles per aparcar, ja sigui indicant el lloc disponible que deixem o algun lloc que trobem pel carrer.

L'objectiu del projecte és aprofitar les noves tecnologies per oferir ajuda entre usuaris.

No és objectiu d'aquest treball / aplicació, incitar a utilitzar el cotxe enlloc del transport públic, quan aquest està disponible, com tampoc provocar un ús indegut del mòbil durant la conducció, però sí que pretén facilitar la cerca d'un lloc per aparcar.

2. Descripció

El TFG consisteix en dissenyar i desenvolupar una aplicació per a mòbils anomenada “On aparcar?”, la qual ens faciliti la cerca d’un lloc on aparcar el cotxe, gràcies a la informació proporcionada per altres usuaris.

L’aplicació “On aparcar?” es desenvoluparà per a dispositius Android, ja que actualment és el sistema amb més quota de mercat i per al bon funcionament de l’aplicació serà necessari un nombre elevat d’usuaris. Números a banda, també cal tenir en compte altres factors, com la facilitat de publicar aplicacions en Android, en comparació amb iOS.

Tenint decidit el sistema Android, el més indicat és programar l’aplicació amb un llenguatge de programació nadiu d’aquest sistema com és el cas de Java. Java és un llenguatge de programació orientada a objectes, el qual, utilitzant llenguatge PHP (PHP: Hipertext Preprocessor) com a intermediari, interactuarà amb una base de dades MySQL (My Structured Query Language), on es guardaran les diferents dades necessàries per a l’aplicació, com són les dades dels usuari o les coordenades dels punts on aparcar.

Les funcions que l’usuari podrà dur a terme són:

- Enregistrar un lloc disponible, especificant les característiques d’aquest (zona blava, zona verda, etc.) i característiques del carrer (cèntric o apartat) per tal de determinar quant de temps es mantindrà activat el punt.
- Un cop aparcat el cotxe, es valorarà positivament l’usuari que n’ha informat. Per a solucionar els possibles problemes de precisió del GPS (Global Positioning System), es tindrà en compte el numero dels edificis.
- Apart de valorar, s’activarà l’opció de recordar el punt d’aparcament, per tal de tornar a l’aparcament.

La visualització de les dades serà utilitzant la API (Application Programming Interface) de Google Maps, utilitzant el punters de diferent colors, depenent del tipus d’aparcament. Aquest punts s’aniran mostrant a mesura que el cotxe avanci, mostrant els del carrer actual i els pròxims.

3. Objectius

3.1 Personals

- Posar a prova els coneixements adquirits al llarg del grau.
- Aprofundir en el desenvolupament d'aplicacions per a telèfons intel·ligents.
- Aprendre més del llenguatge de programació Java, amb el qual no he treballat gaire.
- Millorar aspectes de programació en llenguatges que ja conec i he utilitzat (PHP i MySQL).

3.2 Professionals

3.2.1 Principals

- **Aconseguir una aplicació completament funcional** que permeti trobar aparcament amb l'ajuda de les aportacions de la resta d'usuaris.
- **Estudiar context:** Existeixen varies aplicacions amb finalitats semblants. Un dels principals objectius és estudiar aquestes aplicacions i les funcions que ofereixen per tal d'aconseguir desenvolupar-ne una de competent amb la resta.
- **Disseny de l'aplicació:** Dur a terme un disseny lògic de l'aplicació, intentant que aquesta tingui un bon rendiment i una usabilitat senzilla, per tal que pugui ser utilitzada per a qualsevol usuari.
- **Desenvolupar l'aplicació:** Per tal d'obtenir un bon rendiment, es treballarà amb llenguatge Java, ja que és nadiu d'Android. treballant sobre una base de dades MySQL per tal de mostrar i gestionar la informació de geolocalització amb Google Maps.

3.2.2 Secundaris

- Diferenciar-se de la resta d'aplicacions similars a partir de noves funcions.
- Complir amb la GUI (Graphical User Interface) definida per a Android.
- Afegir funcionalitat de tornar al cotxe.

4. Marc teòric

Trobar un lloc on aparcar el cotxe és una acció que tots, com a conductors, hem de realitzar i que en algunes situacions, com hores puntes o llocs molt cèntrics, pot comportar una gran quantitat de temps. Tenint en compte que avui en dia resulta fàcil estar connectat a la xarxa, pràcticament en qualsevol punt, gràcies als telèfons intel·ligents, i l'augment constant de la utilització de les xarxes socials en la nostre societat, pot resultar beneficiós utilitzar aquest medi per a solucionar els problemes d'aparcament.

Algunes aplicacions similars serien:

- **Apparcar**¹: La qual indica punts on està disponible un aparcament.
- **Waze**²: Aplicació GPS la qual proporciona informació sobre el trànsit. La informació del trànsit és aportada pels propis usuaris.

Es podria dir que “On aparcar?” es basa en principis d'aquestes aplicacions, ja que la informació d'aparcament prové dels propis usuaris.

¹ Apparcar: <http://www.apparcar.com/>

² Waze: <https://play.google.com/store/apps/details?id=com.waze>

5. Continguts

Per accedir als continguts de l'aplicació, serà necessari que l'usuari estigui registrat dins el sistema. Un cop identificat, podrà visualitzar sobre el mapa de Google Maps la informació que es troba dins la base de dades. En resum, es podrien dividir els continguts entre les diferents activitat o pantalles de l'aplicació:

Pantalla inicial: Tot just arrencar el programa, salta una alerta, amb l'objectiu de recordar a l'usuari que no interactuï amb l'aplicació mentre estigui conduint, sinó que s'agafi el seu temps per a configurar-la abans de començar a conduir. Un cop acceptat l'avís, es podrà identificar per tal de tenir accés complet al contingut.

Menú principal: Aquí es proporcionaran dos maneres d'accedir als continguts.

- Aparcaments propers: Per si no es vol especificar un destí concret, l'usuari podrà navegar pel mapa, cercant de manera manual aparcament en la zona del seu interès. Aquesta seria la opció principal per a utilitzar "On aparcar?", buscar un aparcament dins la pròpia ciutat.
- Definir destí: L'usuari pot especificar una direcció concreta i l'aplicació cercarà l'aparcament més pròxim en un radi de 500 m. S'aconsella aquesta opció en trajectes curts, ja que la informació d'aparcament és molt volàtil i el programa no ha estat pensat com a GPS de grans trajectes.

Pantalla mapa: Aquí es mostraran els continguts principals de l'aplicació i on l'usuari durà a terme gran part de la interacció. Sobre el mapa es visualitzarà la informació, allotjada en la base de dades, que ha estat aportada per la resta d'usuaris. Per altra banda, el propi usuari també podrà afegir informació al mapa, indicant nous punts on aparcar, o eliminant-ne al ocupar aparcaments.

Els continguts d'aquest apartat son:

- Punters sobre el mapa, corresponents a les dades aportades pels usuaris. L'estil visual d'aquest vindrà donat per les característiques de l'aparcament, aconseguint així diferenciar els diferents tipus d'aparcament d'una manera ràpida, evitant que l'usuari hagi d'interactuar per obtenir aquest informació. Paral·lelament, aquest punters seran clicables per tal de visualitzar-ne les dades més complertes i accedir a l'opció d'ocupar l'aparcament.
- Les diferents opcions situades al *Actionbar*, que permetran realitzar la interacció:

- “On tinc el cotxe?” Permet guardar la posició d'on s'ha aparcat, per tal de tornar-hi fàcilment. Aquesta informació només serà accessible per al propi usuari.
- “Marcar aparcament” Afegirà nou contingut a la base de dades, amb les coordenades i tipus d'aparcament. Aquest contingut serà visible per a tothom i s'eliminarà un cop un altre usuari el marqui com a ocupat o passat un temps.
- “Definir destí” correspon a la mateixa opció disponible a la pantalla de menú.

Utilitzant el boto “menú” del dispositiu o el boto “*overflow*” de la barra d'acció, depenent de la versió del sistema *Android*, en aquesta pantalla, es podrà accedir al perfil propi d'usuari i a l'ajuda.

Pantalla perfils usuari: L'accés a aquesta secció serà a partir de clicar al nom d'usuari que apareix junt a la informació dels aparcaments. En aquesta pantalla es podrà consultar la informació referent a cada usuari: Nom d'usuari, ciutat, puntuació i aparcament correctes.

Pantalla perfils propi: Accessible a partir del boto “menú” del dispositiu. Com en el cas anterior, permet veure les nostres dades amb la opció de canviar la imatge que ens identifica.

6. Metodologia

La metodologia que s'aplicarà durant el desenvolupament del projecte serà la de cascada o clàssica, en la que cada fase ens durà a la següent:

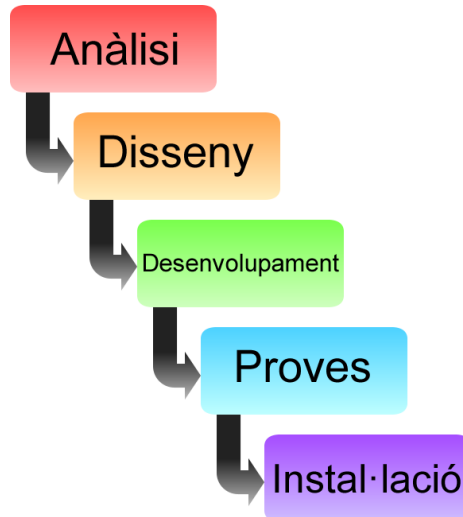


Figura 2: Esquema metodologia en cascada

La teoria d'aquesta metodologia preveu que quan s'ha superat una fase, s'avança cap a la següent sense poder retornar a les etapes anteriors. En la pràctica resulta difícil complir estrictament aquest criteri, ja que és molt possible que en les etapes d'anàlisi o disseny quedi aspectes per resoldre, o que un cop s'ha començat el desenvolupament del projecte i realitzar les primeres proves, s'observa que determinats aspectes s'han de canviar.

Per això es preveu utilitzar la variant de cascada retroalimentada:

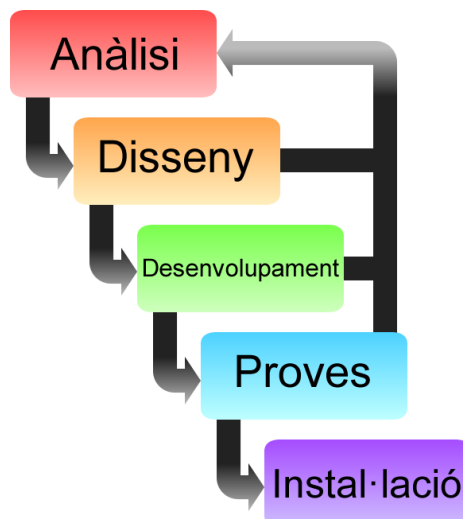


Figura 3: Esquema metodologia en cascada retroalimentada

Aquesta variant permetria tornar a fases anteriors per tal de solucionar problemes d'enfocament o de programació.

A aquesta metodologia s'hi aplicarà el un diagrama de Gantt per tal de planificar un calendari de tasques i entregues de les diferents etapes del projecte. Aquest diagrama, junt amb la planificació es pot trobar en el punt 9.

7. Arquitectura de l'aplicació

Tenint clarament definida l'aplicació, el següent pas és descriure l'arquitectura en la qual funcionarà, la qual consistirà en la típica estructura de client, servidor i bases de dades.

7.1 Client

Com s'ha comentat, es desenvoluparà una aplicació per a mòbils amb sistema Android, compatible a partir de la versió 2.3 *Gingerbread*, la qual es programarà en llenguatge Java. Utilitzant un llenguatge nadiu del sistema ens permetrà aconseguir una millor estabilitat i rendiment, apart de poder aprofitar totes les característiques dels sistemes Android sense problemes.

Per tal d'utilitzar l'aplicació serà necessari que el dispositiu mòbil es pugui connectar a internet, per tal de poder obtenir la informació de la base de dades i enviar-ne, i per altra banda utilitzar la funció de GPS.

7.2 Bases de dades

Per a la base de dades s'utilitzarà el gestor MySQL, amb les següents taules i relacions:

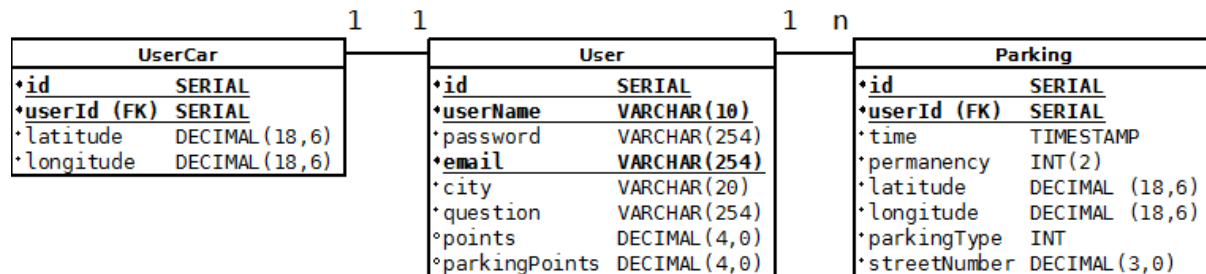


Figura 4: Estructura base de dades

7.3 Servidor Web

Tot i que es pot realitzar una connexió directa entre aplicacions en Java i la base de dades, com per exemple utilitzant l'API JDBC (Java Database Connectivity), per a motius de millor compatibilitat, rendiment i seguretat s'utilitzarà un servidor web com a intermediari entre el client i la base de dades.

En el servidor s'allotjaran els *scripts* PHP que permetran accedir a la base de dades i dur a termes les funcions de registre i obtenció de dades, mitjançant fitxers JSON (JavaScript Object Notation).

8. Plataforma de desenvolupament

8.1 Programari

Els programes utilitzats en el desenvolupament del projecte són:

Adobe Illustrator CS 6: Utilitzat per a la creació dels *wireframes*.

Adobe Fireworks CS 6: Utilitzat per a la creació del disseny gràfic.

Dia³: Editor de diagrames, utilitzat per al disseny dels esquemes UML i base de dades.

notepad++⁴: Editor de text, utilitzat per a desenvolupar els *scripts* en PHP.

Android SDK (software development kit) + Eclipse⁵: Per a desenvolupar aplicacions per Android es necessiten una sèrie de programes, com les llibreries API d'Android, les *Android Developer Tools (ADT)* i un entorn de desenvolupament integrat (IDE), com per exemple Eclipse, per a desenvolupar aplicacions software.

Per tal de facilitar la creació d'aplicacions, Android ofereix un paquet de programari amb tot els elements anomenats, ja configurats, per al desenvolupament d'aquestes.

Genymotion⁶: Tot i que l'ADT ja proporciona un emulador d'Android per realitzar les proves de l'aplicació durant el desenvolupament, per a les proves més avançades s'ha utilitzat l'emulador Genymotion (gratuït per a ús no comercial). Els motius han sigut la major rapidesa, facilitat per a instal·lar aplicacions amb sistema *drag & drop* i la major estabilitat.

Per la banda del **servidor**, serà necessari que disposi de les llibreries i serveis necessaris per a l'execució de pàgines en PHP i del sistema de bases de dades MySQL amb el gestor phpMyAdmin. Com que de moment no s'ha previst publicar l'aplicació, s'utilitzen els servidors per a desenvolupament proporcionats per la UOC, els quals compleixen amb els requisits esmentats:

- Servidor Web Apache 2.0
- Llibreries PHP 5.3.10
- Base de dades MySQL 5.5.37 amb gestor phpMyAdmin 3.4.10.

³ Dia – Diagram editor: <http://live.gnome.org/Dia>

⁴ Notepad++: <http://notepad-plus-plus.org/>

⁵ Android SDK: <http://developer.android.com/sdk/index.html>

⁶ Genymotion emulator: <http://www.genymotion.com/>

8.2 Maquinari

Ordinador: On es durà a terme el desenvolupament del projecte mitjançant els diferents programes esmentats en el punt anterior.

Sony Xperia U: Dispositiu mòbil amb sistema operatiu Android v2.3.7, per tal de poder instal·lar l'aplicació i realitzar proves sobre dispositiu real.

Servidor web i base de dades: En aquest cas s'utilitzarà el servidor per a desenvolupament proporcionat per la Universitat Oberta de Catalunya, per tal d'allotjar les dades que formaran l'aplicació i les pàgines en PHP.

9. Planificació

El desenvolupament del projecte ve marcat per les entregues parcials, les quals serviran com a referència per a diferenciar les diferents etapes del projecte. A continuació es detallen els diferents aspectes de la planificació.

9.1 Dates clau

26/02/2014 – 11/03/2014: PAC 1 – Anàlisi del projecte i planificació.

12/03/2014 – 06/04/2014: PAC 2 – Disseny del projecte.

07/04/2014 – 11/05/2014: PAC 3 – Desenvolupament del projecte i prototip.

12/05/2014 – 24/06/2014: Entrega final – Final desenvolupament, proves i lliurament de l'aplicació.

9.2 Fites

26/02/2014 – 11/03/2014: PAC 1 – Anàlisi del projecte i planificació.

- Anàlisi del projecte
- Definició
- Planificació
- Esborrany memòria TFG

12/03/2014 – 06/04/2014: PAC 2 – Disseny del projecte.

- Disseny wireframes
- Disseny UML
- Disseny base de dades
- Inici desenvolupament aplicació
- Esborrany memòria TFG

07/04/2014 – 11/05/2014: PAC 3 – Desenvolupament del projecte i prototip.

- Desenvolupament aplicació
- Esborrany memòria TFG

12/05/2014 – 24/06/2014: Entrega final – Proves, desenvolupament final i lliurament de l'aplicació.

- Realització proves en simulador.
- Realització proves en dispositius reals.
- Disseny gràfic aplicació

- Reparació errors disseny
- Reparació errors codi
- Lliurament memòria TFG i aplicació

9.3 Diagrama de Gantt

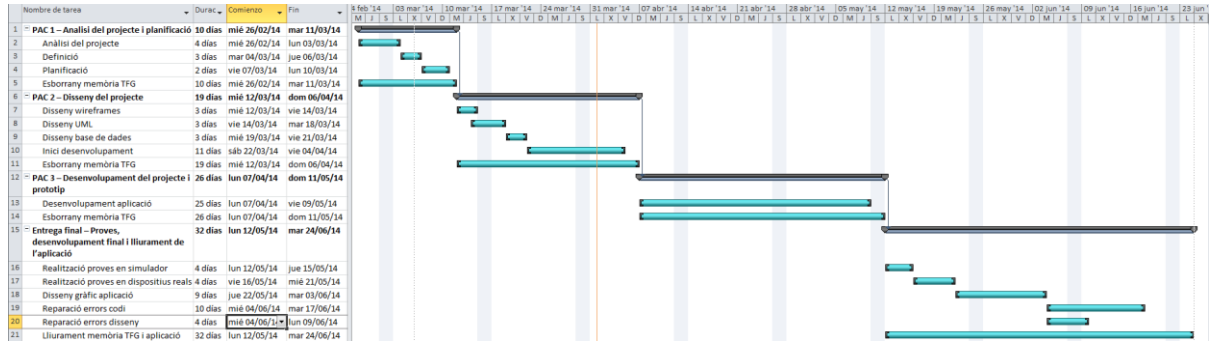


Figura 5: Diagrama de Gantt

10. Procés de treball

El procés de treball s'estructurarà per lliuraments, per tal d'oferir una visió cronològica del treball.

10.1 PAC 2 – Disseny del projecte.

El procés de desenvolupament comença amb la segona PAC. En aquest període s'inicien les tasques de disseny dins les quals es realitzen:

Wireframes: Per tal d'organitzar i mostrar els elements en pantalla. A partir d'aquests resultats es durà a terme el primer disseny per als prototips de l'aplicació.

Llenguatge Unificat de Modelat (UML): On es representa de forma gràfica i esquemàtica l'estructura de l'aplicació.

Disseny Base de dades: Estructurar les diferents taules i els camps de cada una, on es guardaran les dades necessàries per al projecte.

Amb el procés de disseny finalitzat, s'inicia el desenvolupament de l'aplicació per Android. Dins aquest període es duen a terme les següents tasques:

Instal·lació SDK: El primer punt del desenvolupament consisteix en obtenir les eines necessàries, en aquest cas l'SDK d'Android, tal com es comenta en l'apartat 8. Android proporciona un paquet per a desenvolupadors amb tot el programari necessari. Així doncs, es realitza la instal·lació d'aquest i la configuració de l'emulador per a proves.

Creació projecte: Abans d'iniciar el projecte, és necessari escollir per a quin rang de versions del sistema Android es vol que aquest sigui compatible i en quina versió es treballarà concretament, ja que pot ser necessari instal·lar les API de versions antigues.

Actualment les versions més utilitzades⁷ són la 4.1, seguida de la 2.3.3 – 2.3.7. Com que els dispositius reals als quals es realitzaran les futures proves corresponen a la versió 2.3.7, el

7

<http://androidandme.com/2014/01/news/google-posts-fresh-android-distribution-stats-jelly-bean-and-kitkat-usage-on-the-rise/>

projecte es centrarà en aquesta versió. Tot i així cal tenir en compte que seria perfectament possible instal·lar l'aplicació en versions superiors.

Així doncs, el primer pas és assegurar que es té el SDK d'Android 2.3.3 instal·lat amb l'ajuda del SDK Manager:

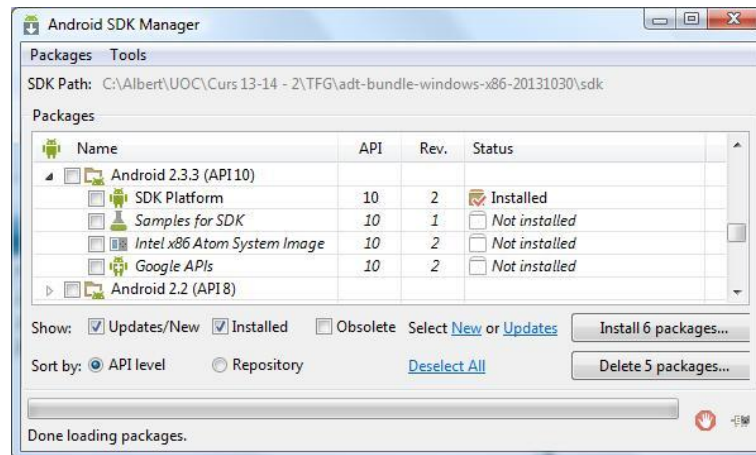


Figura 6: Android SDK Manager.

I la creació del projecte amb els següents paràmetres:

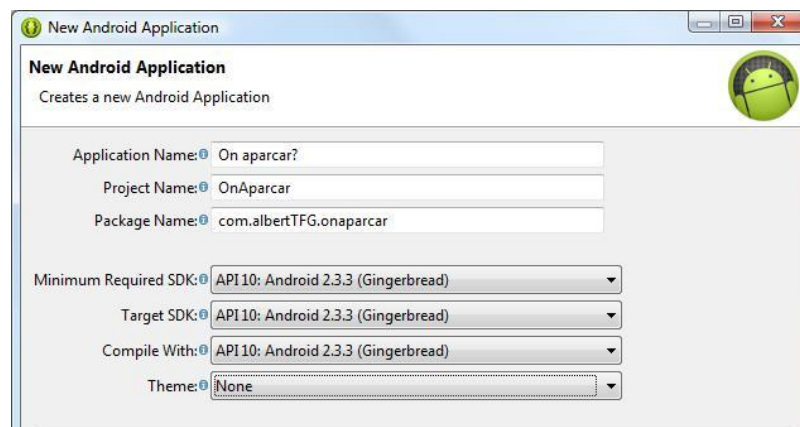


Figura 7: Projecte Android "On aparcar?"

Amb aquesta configuració s'especifica que:

- Minimum Required SDK: La versió mínima on es podrà instal·lar l'aplicació és la 2.3.3.
- Target SDK: Correspon a la versió amb la qual s'han realitzat tests i es pot confirmar que l'aplicació funciona sense problemes.
- Compile SDK: Versió utilitzada per a la compilació. Recomanat utilitzar la mateixa opció que el *target SDK*.

Creació activities: Seguint el disseny UML i els wireframes es generen les tres pantalles principals: LoginActivity, RegisterActivity i AparcarActivity.

Connexió aplicació – Base de dades: Per a la connexió amb la base de dades s'utilitzarà un script en PHP. Aquesta opció ofereix més compatibilitat amb bases de dades MySQL i millor rendiment que altres sistemes de connexió directa.

Les dades entre aplicació i el servidor MySQL es transmetran en format JSON. Per a realitzar el tractament de les dades, s'utilitza la llibreria **JSONParser.java** de Ravi Tamada⁸, i s'adaptarà part del codi d'interacció.

Al llarg de l'aplicació s'utilitzarà de forma recurrent la llibreria **JSONParser.java**, per tal d'interpretar les dades retornades per les API de Google.

Un cop enllestit el codi es realitzaran proves per a comprovar el registre de dades des de l'emulador d'Android:

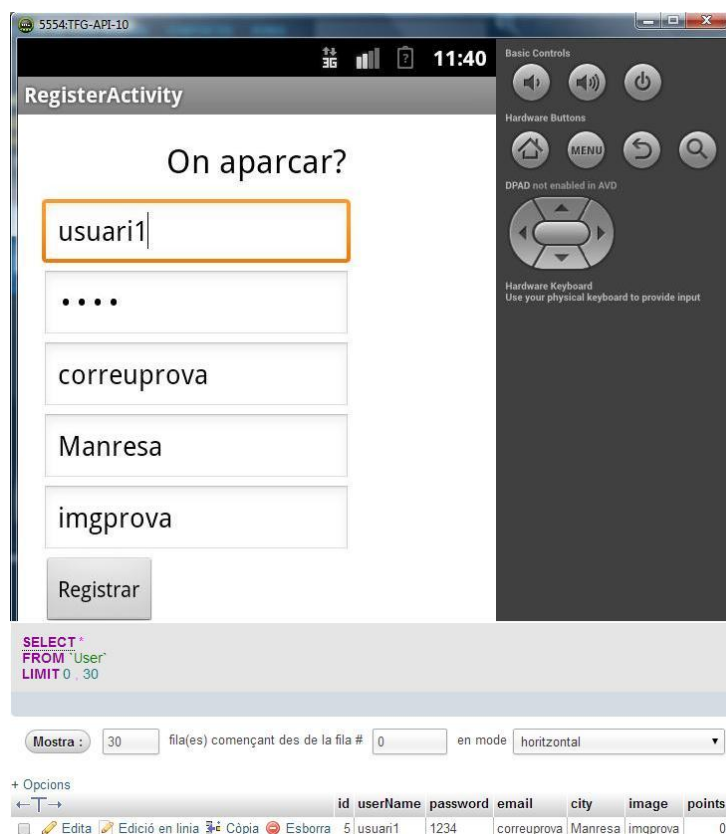


Figura 8: Prova registre dades aplicació i MySQL

⁸ <http://www.androidhive.info/2012/05/how-to-connect-android-with-php-mysql/>

Afegir les Google Maps Android API: Google Maps és una part fonamental de l'aplicació. Amb la intenció de tenir temps de solucionar possibles problemes, serà el següent pas a realitzar.

Google Maps Android API forma part del paquet **Google Play Services** i per aquest motiu serà necessari instal·lar-lo mitjançant l'Android SDK Manager. Un cop instal·lat, es necessari importar-lo com a projecte d'Eclipse, d'aquesta manera es pot utilitzar com a llibreria del projecte.

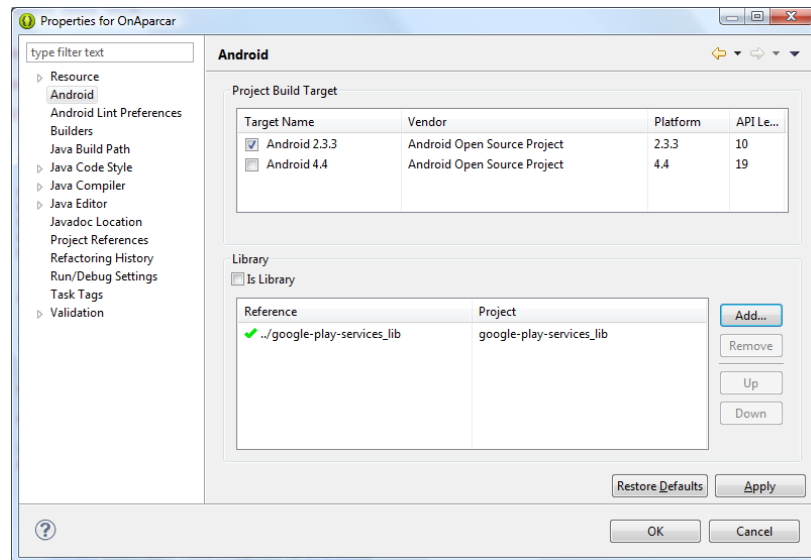


Figura 9: Importació llibreria Google Play Services

El següent pas per afegir Google Maps a l'aplicació és aconseguir una API key de Google Maps. El procediment consisteix en:

- 1) Aconseguir el SHA1 fingerprint del projecte. (Eclipse: Window / Preferences / Android / Build).
- 2) Accedir al API Console⁹ de Google i crear un nou projecte amb la API de Google Maps.
- 3) Dins el nou projecte es genera un nou API Access utilitzant el SHA1 fingerprint de l'aplicació i el nom del projecte: SHA1fingerprint;com.albertTFG.onaparcar

S'obtindrà l'API key per a treballar amb Google Maps, que s'haurà d'afegir dins el AndroidManifest.xml:

```
<meta-data
    android:name="com.google.android.maps.v2.API_KEY"
    android:value="API_KEY"/>
```

⁹ Google API Console: <https://code.google.com/apis/console>

Un cop realitzada l'activitat amb el codi per carregar el mapa i afegits els permisos necessaris dins el `AndroidManifest.xml`, l'aplicació ja permet interactuar amb Google Maps.

Per tal que Google Maps funcioni, el dispositiu ha de disposar de Google Play Services i Google Play Store, fet que no suposa cap problema en dispositius reals. En l'emulador ha estat necessari instal·lar-los abans de realitzar les proves.

10.2 PAC 3 – Desenvolupament del projecte i prototip

Durant el termini corresponent a la PAC3 es continua duent a terme el desenvolupament de l'aplicació, amb la finalitat d'aconseguir un prototip funcional en finalitzar aquest període.

Preparació nou emulador Genymotion: Com s'ha comentat a l'apartat "8.1 Programari", per a realitzar les proves amb Google Maps s'utilitza l'emulador Genymotion, el qual permet generar dispositius virtuals amb la mateixa versió d'Android amb la qual es realitzaran les proves reals, la 2.3.7. En aquest emulador s'hi instal·laran les *Google Apps*, *Google Play Store* i *Google Play Services*.

Obtenir localització: Amb el mapa funcionant, es realitzarà el procés per a obtenir la localització actual. L'*Activity* on es carregarà el mapa passarà a implementar la interfície *LocationListener* la qual proporciona tots els procediments necessaris per obtenir la localització actual del dispositiu i genera els intervals per a actualitzar la posició.

Es crea el *LocationManager* per tal de configurar el sistema de localització:

```
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 3000, 10, this);
```

D'aquesta manera s'especifica que s'utilitzarà el GPS amb un interval de 3 segons (3000 mil·lisegons) i que la distància mínima per actualitzar seran 10 metres.

Per poder utilitzar la interfície *LocationListener*, s'afegeixen els següents mètodes obligatoris:

- **onLocationChanged(Location location):** Permet especificar accions a cada canvi de localització.
- **onProviderDisabled(String provider):** Permet dur a terme accions en cas que el proveïdor del localitzador, GPS en aquest cas, sigui desactivat per l'usuari.
- **onProviderEnabled(String provider):** Permet dur a terme accions en cas que el proveïdor del localitzador, GPS en aquest cas, sigui activat per l'usuari.

- **onStatusChanged(String provider, int status, Bundle extras):** Conté les accions que es realitzarien en cas de que el proveïdor de localització sofreixi algun canvi no realitzat per l'usuari.

Creació de sessions amb Shared Preferences: Per tal de mantenir l'usuari identificat durant totes les parts del programa i tenir a l'abast la informació necessària per realitzar consultes d'aquest usuari, s'utilitzaran les *Shared Preferences*. D'aquesta manera es facilita la realització de determinades consultes a la base de dades referents a l'usuari, ja que disposarem del seu ID per a identificar-lo. Amb aquesta funció podem carregar el perfil d'usuari o saber si l'usuari ha guardat la ubicació del seu cotxe, sense repetir la mateixa consulta per a cada tasca.

Android també disposa d'una base de dades interna per a emmagatzemar informació, bases de dades SQLite, però per a les poques dades que es guardaran en aquesta aplicació és millor gestionar-ho amb les *Shared Preferences*, que obligar al sistema a mantenir tot una base de dades només per a tres camps.

Actionbar: En la pantalla del Mapa, que és on es duran a terme gran part de les accions, s'activarà l'*Actionbar*, posant a disposició de l'usuari les tres accions més importants: Guardar posició del cotxe / tornar al cotxe, registrar aparcament i buscar una direcció com a destí. Les dos accions restants, consultar perfil i ajuda, seran accessibles a partir del boto *Menu* del dispositiu, o del botó *overflow* en aquells dispositius que no tenen botons de hardware.

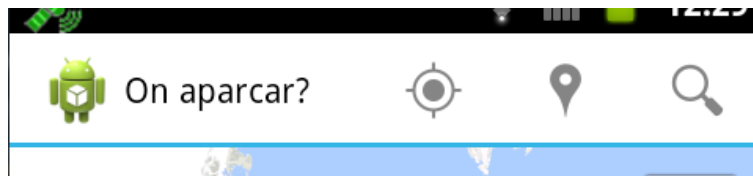


Figura 10: *Actionbar* funcionant

Com en el cas de Google Maps, per tal de poder utilitzar l'*actionbar* serà necessari importar una llibreria com a projecte d'Eclipse i utilitzar-lo dins el projecte. En aquest cas es tracta de la llibreria *Android-Support-v7-Appcompat*.

L'*Actionbar* es dissenya com la resta de pantalles i s'especifica el nom de la plantilla dins el codi encarregat de generar el menú.

```
@Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.aparcar_actions, menu);
        return true;
    }
```

Funcionalitat marcar aparcament: Aquesta funcionalitat s'activa a través de l'*actionbar* i realitza el següent procediment:

- 1) L'aplicació realitza una consulta a la pàgina **addParking.php** enviant els paràmetres:
 - **ID usuari:** Per tal de poder tenir una relació de cada usuari amb els aparcaments proporcionats i rebre el punt corresponent.
 - **Temps:** En funció del nivell de trànsit del carrer, el marcador de l'aparcament serà visible durant un temps determinat.
 - **Altitud i Longitud:** Corresponent a la posició actual de l'usuari i per tant, on l'usuari deixarà l'aparcament disponible.
 - **Tipus aparcament:** Per especificar si és un aparcament normal, zona blava o zona verda.
 - **Número carrer:** Per tal de que l'usuari que ocupi l'aparcament verifiqui que es tracta del mateix que s'està oferint.
- 2) **addParking.php** retorna un JSON amb els resultats del registre, en cas de ser favorables, s'actualitzen els marcadors del mapa.

Funcionalitat guardar posició cotxe / anar cap al cotxe: De nou s'activa a través de l'*actionbar*. Aquesta boto realitza dos funcions depenent de si ja s'ha guardat la posició del cotxe o no. En cas d'utilitzar el boto tenint un registre de posició, es carregarà una ruta al mapa per tal de tornar al cotxe

Guardar posició:

- 1) Quan l'usuari s'identifica a l'aplicació, es comprova si té registres de posició del cotxe i es guarda la informació al *Shared Preferences*.
- 2) En cas de que no existeixi registre, es realitza una crida a la pàgina **addCar.php**, enviant els paràmetres:
 - **ID usuari:** Per tal de poder tenir una relació de cada usuari amb la ubicació del seu cotxe.
 - **Altitud i Longitud:** Corresponent a la posició actual de l'usuari i per tant, on ha aparcat el cotxe.
- 3) **addCar.php** executa la sentència SQL per realitzar el registre dins la taula "UserCar" de la base de dades i retorna una resposta a l'aplicació en format JSON.
- 4) L'aplicació interpreta el JSON i si el registre a tingut èxit, s'afegeix un marcador nou al mapa, només visible per al propi usuari, indicant la ubicació del cotxe i guarda un registre a les *Shared Preferences*.

Anar al cotxe:

- 1) Si la funció detecta que ja existeix un registre a la base de dades, cridarà a la funció **getDirections**, passant com a paràmetres les coordenades del cotxe, obtingudes del marcador del mapa, i un *String* per indicar el mètode de desplaçament, en aquest cas “walking”.
- 2) **getDirections** realitza una consulta a Google Directions API, per tal d’obtenir un JSON amb les indicacions per arribar fins al punt de destí, en aquest cas el cotxe de l’usuari i dibuixar el recorregut en el mapa.
getDirections és un mètode adaptat a partir d’un exemple de **George Mathew**¹⁰, apart d’utilitzar un mètode de **Jeffrey Sambells**¹¹ per a descodificar els *poly lines* de la API.

Funció definir destí: Accessible des de la pantalla de menú o des de l’*actionbar* a la pantalla del mapa. Permet especificar un destí i que al arribar-hi, l’aplicació cerqui l’aparcament més proper.

- 1) L’usuari especifica una direcció formada per: carrer, número, ciutat. Aquesta cadena de text s’envia a la funció **getGeoCoding**.
- 2) **getGeoCoding** realitza una consulta a Google Geocoding API, per tal d’obtenir resultats de l’adreça i coordenades.
- 3) Es presenta una llista de resultats a l’usuari per tal de que esculli el resultat desitjat.
- 4) Un cop escollit un dels resultats, es torna a fer ús del mètode **getDirections** per obtenir les indicacions al mapa.

Carregar marcadors al mapa: Al iniciar la pantalla del mapa es carregaran per primer cop els marcadors. Posteriorment, cada cop que es realitzi alguna modificació que afecti els marcadors, aquests es recarregaran per tal de reflectir els canvis.

Per a la carrega, es realitza una consulta al pàgina **getMarkers.php**, passant-li com a paràmetre la ID d’usuari. Aquest *script* executa dos sentències SQL: una per obtenir tot els aparcaments i l’altra per comprovar si l’usuari ha guardat la posició del cotxe, amb el paràmetre ID.

Aquest codi PHP retorna un JSON amb el llistat d’aparcaments amb les seves dades. Així doncs es recorre tota la llista col·locant cada marcador a les coordenades específiques i amb

¹⁰ <http://wptrafficanalyzer.in/blog/driving-route-from-my-location-to-destination-in-google-maps-android-api-v2/>

¹¹ <http://jeffreysambells.com/2010/05/27/decoding-polylines-from-google-maps-direction-api-with-java>

l'ajuda d'un *switch* es carrega l'element gràfic per a cada tipus d'aparcament (zona blava, etc) i tipus de carrer (nivell trànsit).

Anar a / Ocupar aparcaments: En un primer moment s'havia previst que les accions dels aparcaments es realitzessin per mitjà de botons dins els *InfoWindow* (globus amb informació) de cada marcador. Finalment s'ha alterat aquest punt, ja que el contingut dels *InfoWindow* es carrega com si es tractes d'una imatge, i per tant, no deixa aplicar acció per a diferents botons.

Per solucionar aquest inconvenient, dins el globus de cada marcador s'hi carregarà la informació del carrer, obtinguda del mapa, i serà en pulsar sobre l'*infowindow* que s'activarà la finestra per ocupar l'aparcament.

En aquesta finestra es presenten dos opcions:

- **Anar a l'aparcament:** En escollir aquesta opció, s'obtidran les coordenades del marcador i gràcies a la funció **getDirections** es mostrarà la ruta cap a l'aparcament des de la nostra posició actual.
- **Ocupar l'aparcament:** Aquesta opció envia les coordenades del marcador, el número de carrer introduït i la ID de l'usuari que està ocupant, a la pàgina **removeMarkers.php**. En aquest *script* es realitza una recerca de l'aparcament, a partir de les coordenades, dins la taula **Parking**. Un cop localitzat l'aparcament es comparen les ID de l'usuari ocupant i de qui ha publicat l'aparcament, per tal d'evitar que un mateix usuari vagi ocupant els seus llocs només per augmentar la puntuació.

Un cop comprovat que els usuaris són diferents, es comprova si els números de carrer coincideixen. En cas afirmatiu, l'usuari publicador rep un punt extra de puntuació i un punt d'aparcament correcte. Finalment s'elimina el marcador de la base de dades.

Veure perfil: Per a visualitzar els perfils, s'ha creat una funció **viewProfile(int)**, on el valor **int** correspon a la ID de l'usuari a consultar el perfil. Aquest mètode realitza una crida a la pàgina **getInfo.php** on es cerca l'usuari per ID i es retornen els camps: ciutat, punts i punts d'aparcament.

Proves: Al finalitzar cada etapa s'han realitzat proves per comprovar el bon funcionament de cada nova funcionalitat. Abans de l'entrega de la PAC3 s'han dut a terme proves generals de l'aplicació.

10.3 Entrega final – Proves, desenvolupament final i lliurament de l'aplicació

A continuació s'expliquen les tasques realitzades en l'última etapa del projecte.

Test del prototip: Amb el prototip funcional obtingut al final de l'etapa anterior, es realitzen diferents proves per a comprovar el correcte funcionament de l'aplicació i detectar errors o punts per acabar.

Les proves es realitzen amb un grup de 4 usuaris amb edats entre els 20 i 60 anys, tant en emulador com en dispositius reals i amb diferents grau d'experiència en telèfons intel·ligents.

Disseny gràfic: Es realitzen els dissenys de la icona de l'aplicació, els diferents marcadors i s'acaba de donar format a les diferents pantalles del programa.

Per proporcionar informació ràpida a l'usuari, i evitar distraccions en la conducció, els marcadors proporcionen informació de l'aparcament:

- **Color intern:** Representa el tipus aparcament
 - Blau: Zona blava.
 - Verd: Zona verda.
 - Blanc: Aparcament normal.
- **Color extern o contorn:** disponibilitat de l'aparcament
 - Vermell: Carrer molt transitat, aparcament de poca disponibilitat.
 - Taronja: Carrer amb trànsit regular, disponibilitat mitjana.
 - Negre: Carrer poc transitat, molta disponibilitat.

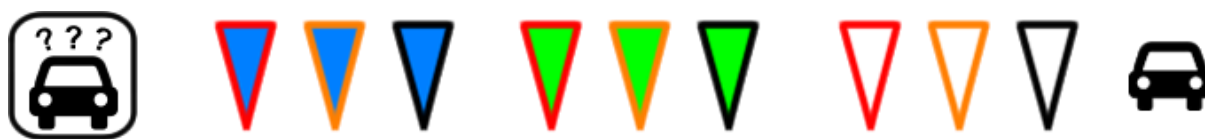


Figura 11: Mostra icona i marcadors mapa

10.3.1 Correccions i millores

A continuació es comentaran les millores aplicades o correccions de problemes detectats en les proves.

Comprovació camps dels formularis: Un problema comú en formularis és la possibilitat d'introduir dades no vàlides en determinats camps, com per exemple lletres en un camp numèric. Aquest cas es soluciona fàcilment afegint `android:inputType="number"` a les característiques del camp.

Un cas més complex és el de validar el correu electrònic. Tot i això l'API d'Android incorpora una funció per a detectar si el format d'un correu es correspon al de `xxxx@xxx.xxx`, aquesta és la següent `android.util.Patterns.EMAIL_ADDRESS.matcher(email_a_validar).matches()`. La funció retornarà un valor booleà en funció de si es tracta d'un format correcte o no.

Evitar SQL Injection: S'utilitza la funció MySQL `mysql_real_escape_string(String)`¹² dins les consultes SQL dels fitxers PHP. Aquesta funció afegirà el símbol `/` davant dels símbols especials, convertint així les cadenes de text en segures i evitant que puguin executar accions a la base de dades.

Cal tenir en compte que aquesta funció queda obsoleta a partir de la versió 5.5 de PHP, així doncs, en cas d'actualitzar les llibreries del servidor, serà necessari utilitzar la extensió MySQLi.

Canviar contrasenya: Per a solucionar problemes de pèrdua de contrasenya, s'ofereix la possibilitat de canviar la contrasenya. Com que en aquest projecte no es compta amb servidor de correus, per a validar l'usuari es buscarà una alternativa.

Per a canviar la contrasenya s'afegirà el camp de pregunta secreta, ja que el nom d'usuari i correu electrònic són dades que poden conèixer altres usuaris. Així doncs la pregunta secreta ha de ser de caràcter personal i que difícilment algú pugui conèixer, en aquest cas s'ha optat per "Nom del primer professor d'EGB?", que tot i no ser un mètode segur 100%, redueix bastant les possibilitats de suplantació d'usuaris.

Així doncs, s'afegirà el camp "*question*" a la taula "*User*" i al formulari de registre. A continuació es crea la nova pantalla amb el formulari de canvi de contrasenya, on serà necessari introduir correctament el nom d'usuari, el correu electrònic i la resposta secreta. Paral·lelament, es crearà una nova pàgina PHP amb la consulta necessària per a comprovar les credencials i aplicar el canvi si s'escau.

Encriptació contrasenya i pregunta secreta: Com a últim retoc per a millorar la seguretat, s'aplicarà un mètode per a encriptar les contrasenyes i respostes secretes, utilitzant algun dels algorismes suportats per Java, com és el "SHA-256".

¹² http://es1.php.net/mysql_real_escape_string

Pantalla informació: S'afegeix la típica pantalla d'informació del programa. En aquest cas mostra l'autor, consultor, professor i llicència.

Mostrar només aparcaments actius: Cada aparcament serà vàlid durant un temps determinat que varia en funció de la intensitat del trànsit del carrer.

- Nivell alt: 2 minuts
- Nivell mitjà: 15 minuts
- Nivell baix: 1 hora

No interessa mostrar un aparcament d'intensitat alta al cap de 20 minuts, ja que molt probablement en 2 minuts o menys ja estarà ocupat. Per mostrar els aparcament actius es modifica la sentència SQL del fitxer **getMarkers.php**.

La taula d'aparcaments registra l'hora en que s'ha registrat i un camp de permanència amb els minuts de validesa comentats. Aprofitant aquestes dades, es realitzarà una diferencia entre la hora de registre i la hora de consulta. Si el resultat és superior al temps de permanència, es mostrarà l'aparcament; en cas contrari no es mostrarà.

Cerca aparcament ens els 500 m pròxims a destí: Com que l'objectiu de l'aplicació és trobar un aparcament, es modifica l'opció de "Especificar destí" amb la característica que ara buscarà aparcament en un radi de 500 metres del destí. Només si hi ha resultats, s'obtidran les rutes al mapa.

Per aconseguir aquestes dades es genera un nou fitxer PHP, **nearParking.php**, amb una consulta que calcula la distancia entre la coordenada de destí i les coordenades dels aparcament registrats i no caducats.

Esborrar línies de ruta: Durant les proves amb usuaris, es detecta que les línies de ruta no s'eliminen al ocupar un aparcament o esborrar la posició del cotxe. S'aplica el procediment per tal que al realitzar alguna d'aquestes dos opcions, s'esborri la ruta del mapa.

Error al iniciar aplicació sense connexió o GPS: Durant els tests van aparèixer errors per no tenir el GPS activat o connexió a Internet, els quals tancaven completament l'aplicació i sense proporcionar cap tipus d'informació a l'usuari. Per a solucionar aquest inconvenient, al iniciar el programa es realitza una comprovació de connexions, avisant que s'han d'activar en cas de no estar-ho.

Enllaços a registre o recuperació de contrasenya: Alguns usuaris han trobat que els enllaços per a registrar-se o recuperar les credencials són massa petits. Com a millora s'aplica una petita modificació respecte als *wireframes*, augmentant el text i la ubicació.

Enllaç a perfil usuari: Junt a la informació de l'aparcament, apareix el nom de l'usuari que l'ha compartit, permetent visitar el seu perfil. Per reforçar que es tracta d'un enllaç, se l'hi aplica un subratllat i el text amb color blau, similar als enllaços d'una pàgina web.

Creacio document ajuda: Amb les funcions ja programades i aplicades les correccions necessaris, per a finalitzar el desenvolupament es crea el document d'ajuda de l'aplicació. Aquest es generarà en HTML, ja que permet obtenir un text més ben formatat i una millor visualització que utilitzant programació. Aquesta ajuda s'allotjarà al servidor web i serà accessible des del menú / *overflow*.

11. APIs utilitzades

Android 2.3.3 APIs: Per a desenvolupar aplicacions per Android es fa totalment imprescindible utilitzar les API d'aquest sistema, ja que proporcionen totes les llibreries necessàries i assegura que es puguin aprofitar totes les característiques que permet aquest sistema operatiu. L'elecció de la versió a utilitzar depèn principalment de la versió instal·lada en els dispositius on es podran realitzar les proves.

Google Maps Android API: Per les característiques de l'aplicació i l'ús que se n'ha previst, està clar que la millor manera de proporcionar la informació es mitjançant un mapa. Google Maps resulta una de les millors opcions, ja que tan aquest com el sistema operatiu Android pertanyen a Google i això ens assegura una bona compatibilitat. Per altra banda, la API de Google Maps permet una gran adaptació i personalització del mapa, com és la possibilitat d'afegir diferents tipus de marcadors o de registrar les coordenades de punts d'interès en una base de dades i així poder-los recuperar, punts clau en la utilitat de l'aplicació.

El seu ús gratuït està limitat a 2500 peticions/dia

Google Geocoding API: Aquesta API resulta un bon complement de Google Maps, ja que la seva funció és convertir una adreça en coordenades. D'aquesta manera es pot generar un cercador de direccions amb l'objectiu d'establir punts de destí en el GPS.

Les consultes es realitzen a partir d'una URL, acompanyada d'una sèrie de paràmetres. Els resultats de la consulta es retornen en format XML o JSON.

El seu ús gratuït està limitat a 2500 peticions/dia

Google Directions API: La funcionalitat de Google Directions és calcular la ruta entre dos punts específics, permetent aplicar les línies de ruta sobre el mapa de Google Maps. La consulta es pot realitzar tant especificant les direccions amb el format clàssic o mitjançant les coordenades.

Com en el cas anterior, la consulta es realitza mitjançant URL amb els paràmetres especificats en aquesta, obtenint el resultat en format XML o JSON

El seu ús gratuït està limitat a 2500 peticions/dia

12. Diagrames UML

Diagrama UML de les diferents pantalles (*Activity*) que componen l'aplicació, amb els mètodes corresponents.

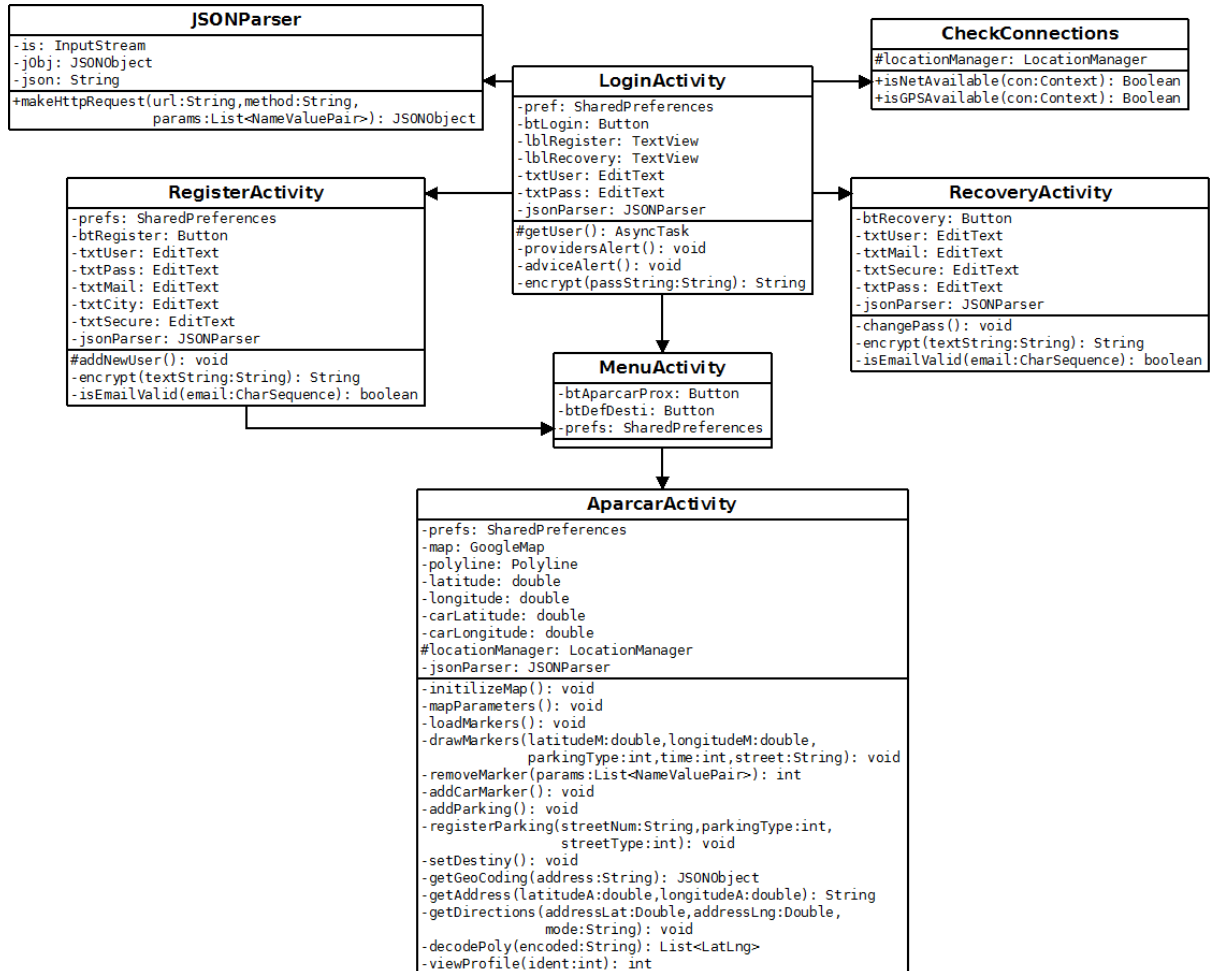


Figura 12: Diagrama UML

13. Prototips

13.1 Wireframes

Per al disseny de l'aplicació s'han seguit les pautes establertes per Android, l'*Android Design*¹³, on s'especifiquen entre altres: estructura bàsica, marges, alçada mínima d'elements clicables, etc.

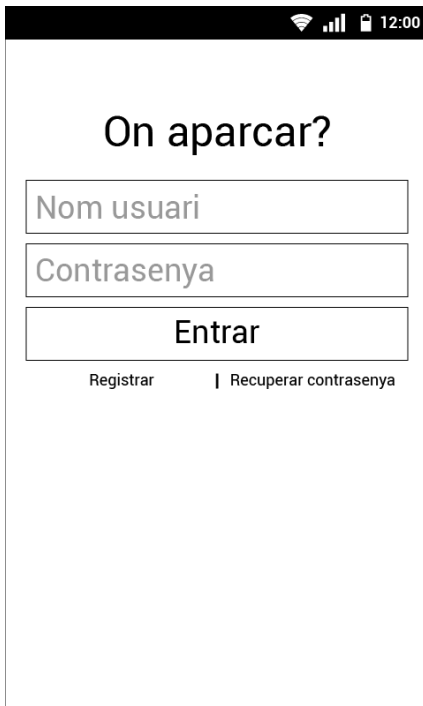


Figura 13: Pantalla principal – Identificació

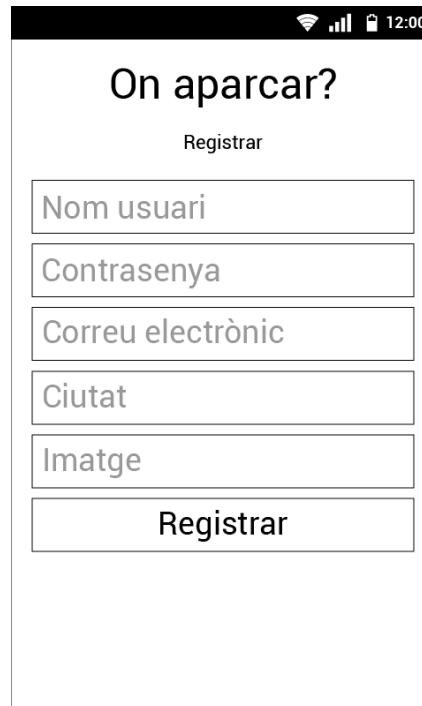


Figura 14: Pantalla registrar nou usuari

¹³ Android design: <http://developer.android.com/design/index.html>



Figura 15: Pantalla menú

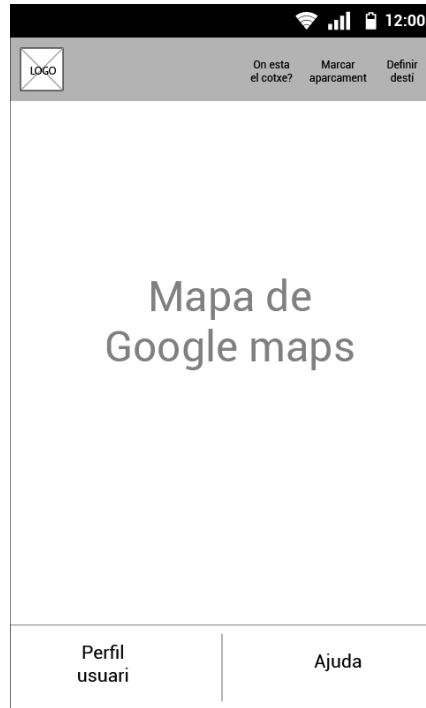


Figura 16: Pantalla Mapa

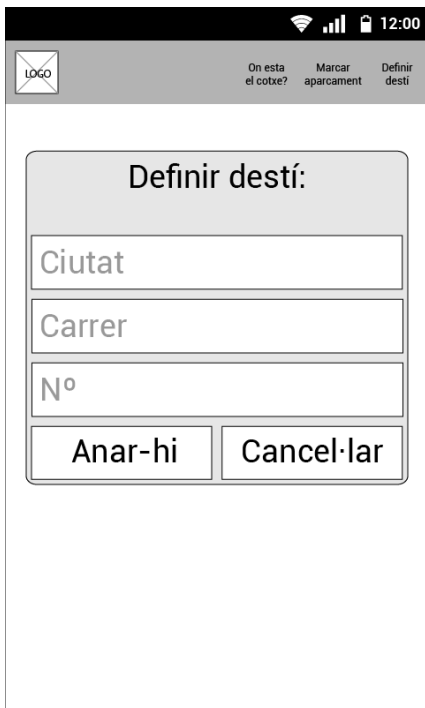


Figura 17: Pantalla definir destí

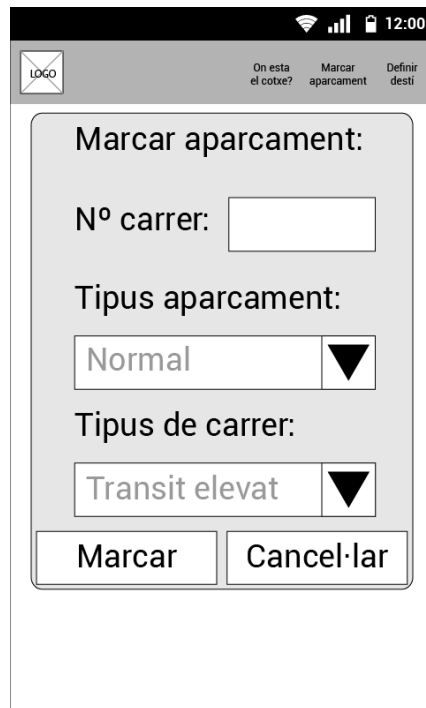


Figura 18: Pantalla afegir aparcament

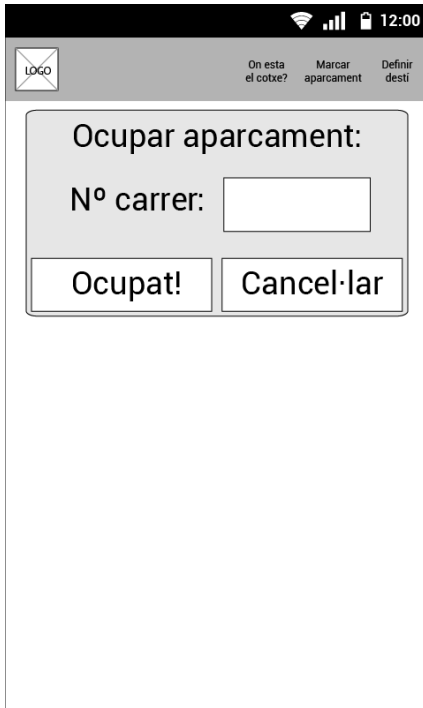


Figura 19: Pantalla ocupar aparcament

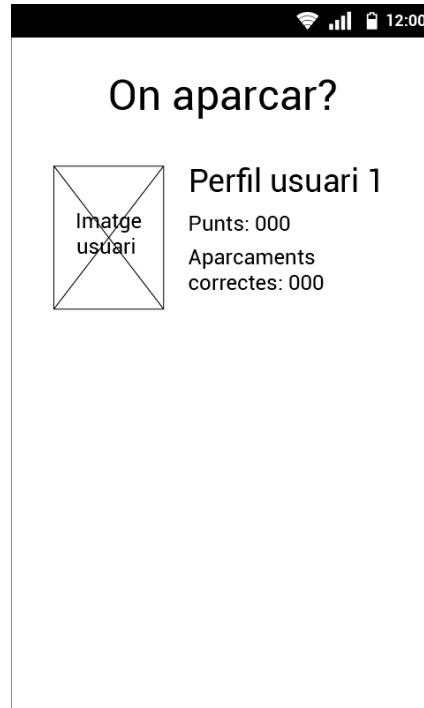


Figura 20: Pantalla perfil altres usuaris

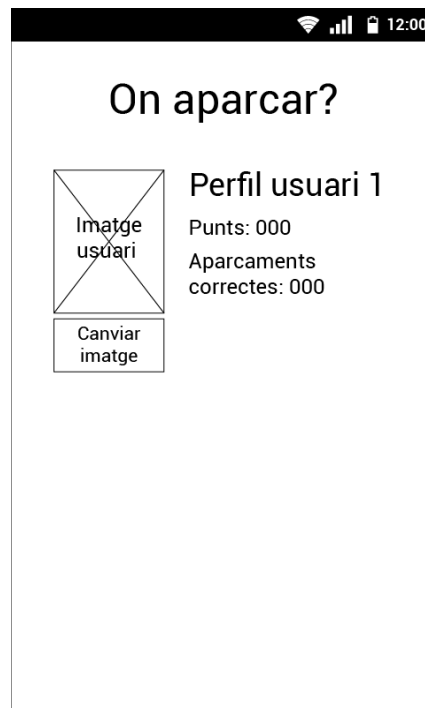


Figura 21: Pantalla perfil propi

13.2 Hi-Fi

A continuació s'exposen les pantalles finals, amb el disseny gràfic finalitzat i els retocs aplicats tant per qüestions tècniques o per resultat de les proves amb usuaris.

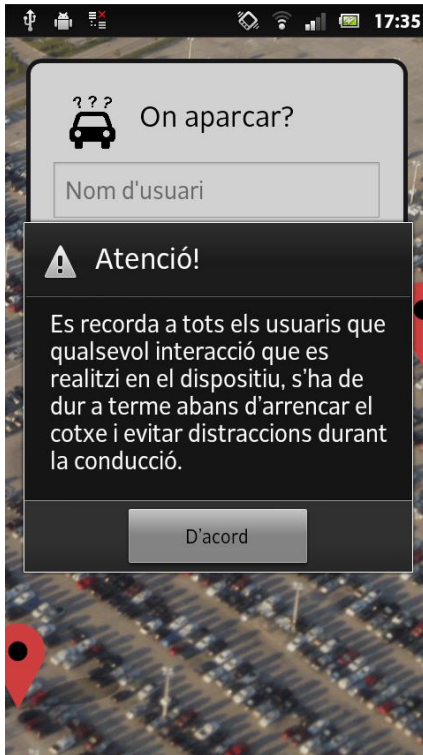


Figura 22: Avis als conductors

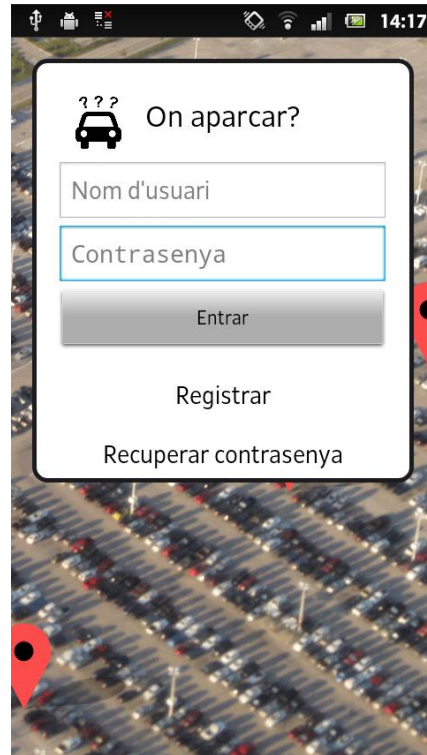


Figura 23: Pantalla principal

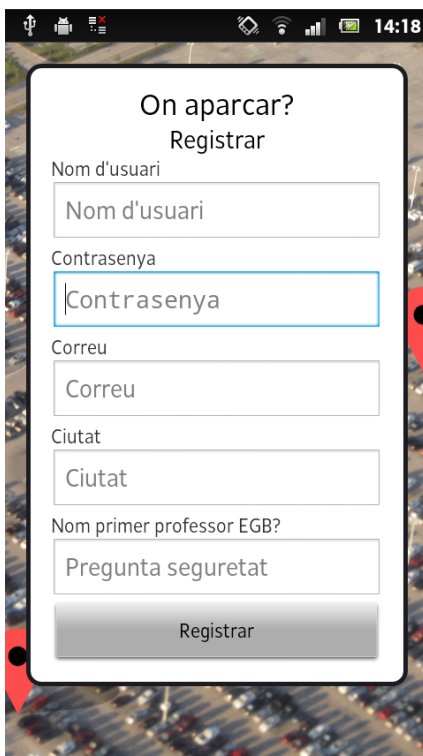


Figura 24: Pantalla registrar

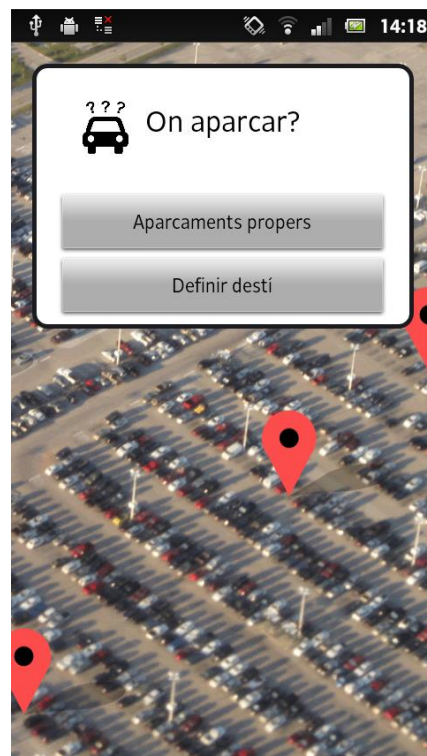


Figura 25: Menú



Figura 26: Mapa

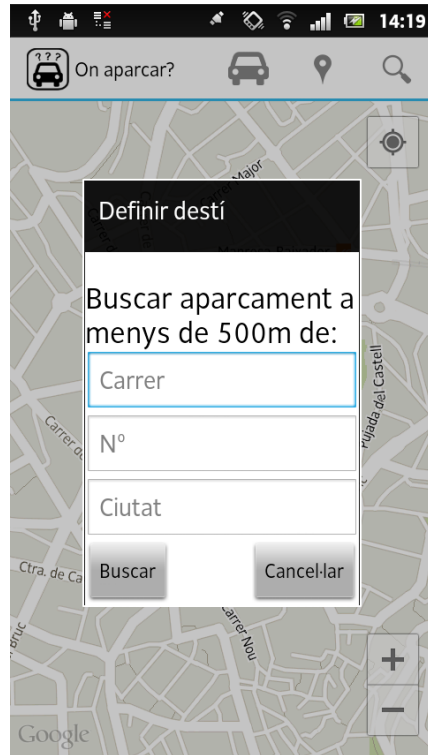


Figura 27: Finestra "Definir destí"

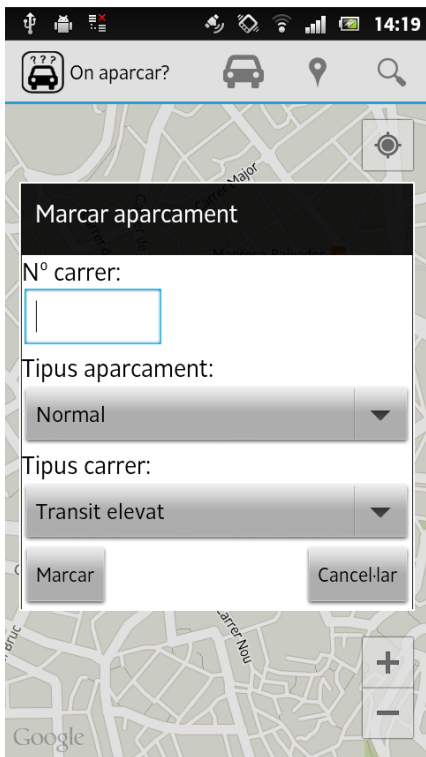


Figura 28: Finestra "Marcar aparcament"



Figura 29: Finestra "Anar / Ocupar aparcament"

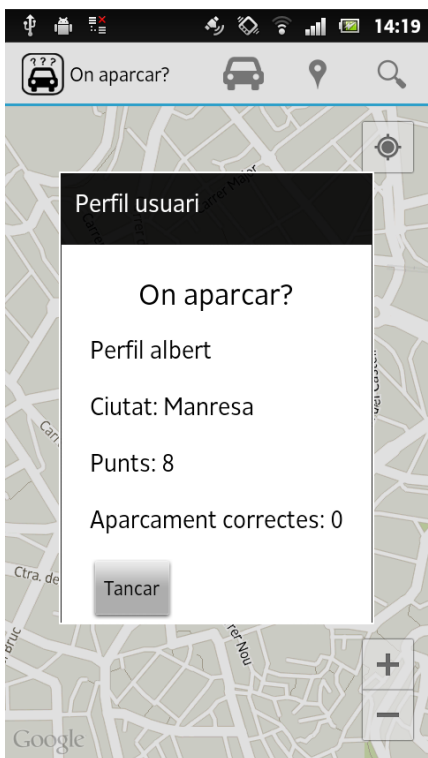


Figura 30: Finestra "Perfil usuari"



Figura 31: Finestra "Informació"

14. Perfils d'usuari

L'aplicació "On aparcar?" està pensada per als següents perfils d'usuari:

- Persones que disposin de mòbils intel·ligents amb sistema Android i que coneguin mínimament les funcionalitats d'aquest i com s'estructuren els apartats en Android.
- Persones que realitzen un ús elevat del cotxe.
- Usuaris que encara que no tinguin permís per a conduir, puguin ajudar al conductor utilitzant l'aplicació.
- Residents en ciutats on el trànsit resulta elevat.
- Usuaris disposats a col·laborar, oferint informació d'aparcaments.

15. Usabilitat

En el disseny de l'aplicació s'han tingut en compte els següents principis d'usabilitat:

- **Consistència:** Tot i que l'aplicació compta amb varies pantalles i apartats, pràcticament només s'utilitzen dues plantilles diferents. Aquesta constància de l'entorn ajuda a l'usuari a trobar fàcilment les funcions tot i canviar de pantalla.
- **Control de l'usuari:** L'aplicació ha de ser intuïtiva i les accions han de ser clares. Quan s'interactua amb l'aplicació, aquesta ha de respondre tal com espera l'usuari. Les accions han de donar com a resultat allò que diu que s'obtindrà, sense donar sorpreses o enviar l'usuari fora de l'aplicació.
- **Prevenió d'errors:** Aquest principi s'aplica directament als formularis, on:
 - No s'han de permetre caràcters en un camp numèric.
 - Els camps amb valors predeterminats es presentaran com a llista desplegable.
 - Els camps de text ofereixen ajuda del valor a d'introduir.
- **Interfície explorable:** S'ofereix flexibilitat en la navegació de l'aplicació.
- **Metàfores:** En l'apartat principal de l'aplicació, algunes accions estaran representades en metàfores, per tal d'oferir una comprensió ràpida.
- **Missatges d'error:** En el cas de produir-se algun tipus d'error, procurar que aquest executi una finestra de diàleg, evitant tecnicismes, per tal que l'usuari pugui comprendre que ha passat.
- **Temps de resposta:** Quan s'executin accions que requereixen un temps per al procés, es mostrarà una finestra de diàleg amb l'objectiu de demostrar que l'aplicació no s'ha bloquejat, sinó que està treballant, però que es requereix un temps d'espera.

15.1 Formes d'interacció

Per interactuar amb el programa, s'utilitzen les accions bàsiques per interactuar amb qualsevol dispositiu tàctil. Les accions són:

- **Tap:** Tocar breument la pantalla. Acció simple per activar la majoria de botons.
- **Scroll, swipe, pinch:** Accions de desplaçar el dit, tant horitzontalment com verticalment o de utilitzar dos dits, com si es volgués pessigar. Aquestes accions serviran principalment per al mapa de Google Maps i les pantalles on es pugui desplaçar pel contingut.

15.2 Navegació

L'aplicació ha estat pensada per tal de que sigui simple d'utilitzar i que la majoria d'accions es puguin realitzar de manera ràpida. Per aquest motiu s'ofereixen varies opcions, però amb molt poca profunditat, de tal manera que la majoria d'accions o apartats estiguin sempre a la vista.

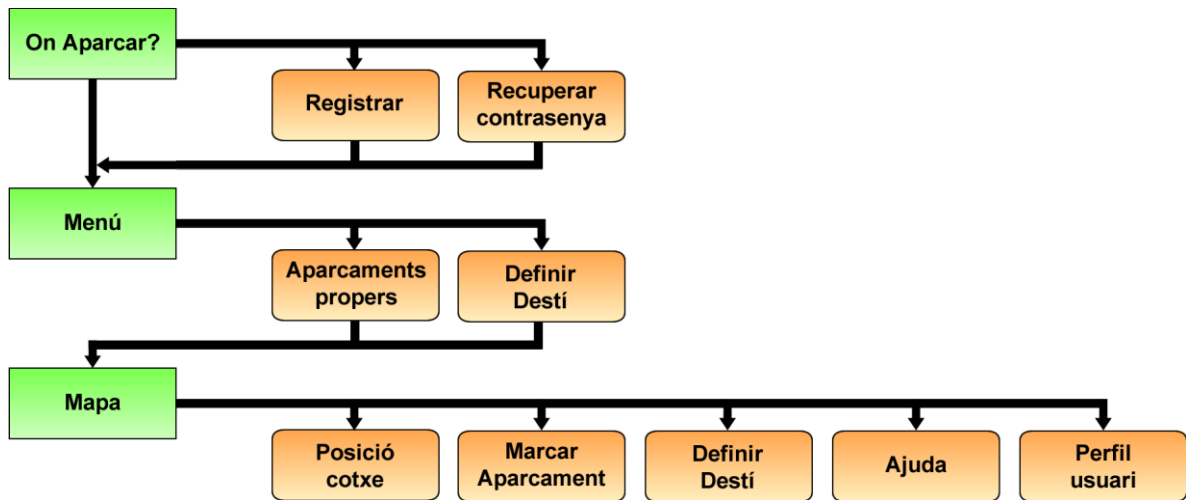


Figura 32: Esquema navegació

16. Seguretat

16.1 Base de dades

Les bases de dades és un dels punts vulnerables de l'aplicació, ja que aquesta depèn totalment de la informació que s'hi allotja. Per a una bona gestió de les dades es duran a terme les següents tasques:

- Només l'administrador tindrà un accés total a les bases de dades. Per augmentar-ne la seguretat, la contrasenya d'aquest usuari es canviarà en períodes de 3 mesos.
- Els usuaris degudament identificats podran editar les taules i camps necessaris per aportar informació al programa i sempre a través de l'aplicació, mai directament sobre la base de dades.
- Realització de còpies de seguretat regulars de tota la base de dades, per tal de poder restaurar la informació en cas de pèrdua total o parcial.

Per altra banda, la connexió a la base de dades es realitzarà utilitzant un script en PHP. Utilitzant aquest sistema, s'aconsegueix una nova capa de seguretat, ja que connectant l'aplicació directament a la base de dades, amb l'API JDBC per exemple, ens obliga a tenir les credencials de la base de dades al mig del codi. D'aquesta manera només caldria aconseguir l'.APK del programa i descompilar-lo per obtenir les dades.

16.2 Formularis

Bona part de l'aplicació està formada per formularis, permetent així la interacció amb l'usuari, com el registre de nous usuaris o afegir nous aparcaments. Per altra banda, també suposen un punt d'accés a la base de dades, motiu pel qual serà necessari afegir certs controls per tal d'evitar possibles atacs **SQL Injection**, que podrien malmetre la informació enregistrada.

També serà necessari aplicar protocols d'encryptació en les contrasenyes dels usuaris, assegurant així que les dades es transmetin de forma segura i sigui difícil de poder aprofitar si cauen en mans de terceres persones.

Finalment quedaria cobrir el problema que poden suposar els *Bots*. Amb la finalitat d'evitar que s'infiltrin dins el sistema, generant una gran quantitat de dades falses i consumint ample de banda, es podria estudiar la possibilitat d'afegir un *Captcha* dins el formulari de registre o per altra banda bloquejar als usuaris que generin una gran quantitat d'informació en pocs minuts.

17. Tests

Els tests necessaris que s'han de realitzar per tal d'assegurar un bon funcionament del programa són:

Control formularis: Aquestes proves consistiran en comprovar la correcta validació de els dades en els formularis per part de l'aplicació. Per a dur a terme les proves s'introduiran elements erronis dins als formularis, amb l'objectiu de comprovar que no es puguin introduir sentències SQL (SQL Injection) o que s'introdueixin les dades requerides (per exemple una estructura xxx@xxx.xxx en el camp correu electrònic).

Correcte funcionament dels punters: Comprovar que la localització GPS sigui la correcta i el punter es posicioni en les coordenades correctes. També serà necessari supervisar que els punters apareguin amb l'estil gràfic determinat per les característiques de l'aparcament.

Usabilitat: Un cop superats els problemes de caràcter més tècnic, es donarà pas a les proves d'usabilitat. En aquestes s'escolliran un grup d'usuaris, amb diferents nivells d'experiència sobre l'ús de noves tecnologies, per tal que realitzin algunes de les accions que permet l'aplicació. L'objectiu és avaluar si el disseny i funcionament de l'aplicació resulten prou clars, per tal que un usuari amb pocs coneixements en aquests tipus d'aplicacions, pugui realitzar les tasques bàsiques i poder aprofitar el programa.

18. Versions de l'aplicació/servei

A continuació es llistaran les funcionalitat que s'han previst per a cada versió.

18.1 Alpha

- Connexió aplicació base de dades.
- Registre d'usuaris.
- Integració de la API de Google Maps.
- Identificació d'usuaris.

18.2 Beta

- Gestió aparcaments.
- Guardar posició del cotxe.
- Visibilitat perfils.

18.3 Versió 1.0

- Disseny final.
- Correcció bugs versions anteriors.
- Contrasenyes encriptades.
- Recuperació contrasenya.

19. Requisites d'instal·lació

19.1 Client

Al tractar-se d'una aplicació desenvolupada per Android i en llenguatge Java, serà necessari que el smartphone on es vulgui realitzar la instal·lació del programa "On aparcar?" disposi d'aquest sistema operatiu.

Els requisits tècnics general serien:

- Smartphone amb sistema Android 2.3, 2.3.3 o 2.3.7.
- Connexió a internet, per tal de carregar el mapa i els continguts de la base de dades.
- És necessari que el dispositiu incorpori GPS, per tal d'obtenir la ubicació de l'usuari.

Tot i que la base de l'aplicació és Android 2.3.X, hauria de ser compatible amb versions superiors. Tot i així, en la versió actual de l'aplicació, aquesta resulta incompatible en versions Android 4.X, ja que s'han d'aplicar unes polítiques de programació més estrictes (com per exemple controlar que els procediments llargs s'executin en segon terme, evitant un aparent col·lapse del dispositiu).

19.2 Servidor

El programa es connecta a una base de dades, on es troben allotjades les dades dels usuaris i aparcament, mitjançant pàgines programades en PHP. Així doncs, serà necessari que el servidor escollit disposi de les llibreries necessàries per a la interpretació i execució de codi PHP , i per altra banda, que permeti l'ús d'una base de dades on registrar tota la informació.

Tenint en compte el servidor on s'han realitza les proves, el requisits de servidor serien:

- Servidor Web Apache 2.0
- Llibreries PHP 5
- Base de dades MySQL 5.5

20. Instruccions d'instal·lació/implantació

Per assegurar el bon funcionament de l'aplicació es recomana utilitzar un dispositiu Android 2.3.X o en el cas de realitzar proves a l'ordinador utilitzar l'emulador Genymotion.

20.1 Dispositiu real

1) Connectar el dispositiu a l'ordinador i copiar el fitxer **OnAparcar.apk** dins algun directori del dispositiu per a posteriorment realitzar la instal·lació, per exemple a la carpeta *Download*:

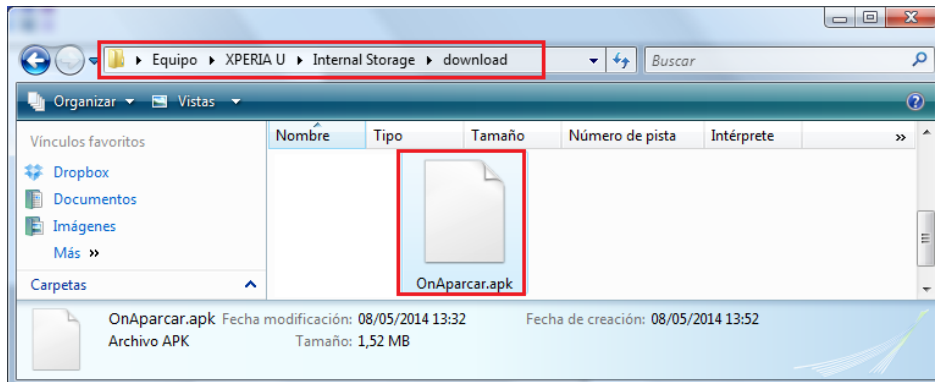


Figura 33: Aplicació copiada al dispositiu.

Ja des del dispositiu, utilitzar algun gestor de fitxers, com per exemple *Astro*¹⁴, per instal·lar l'aplicació.

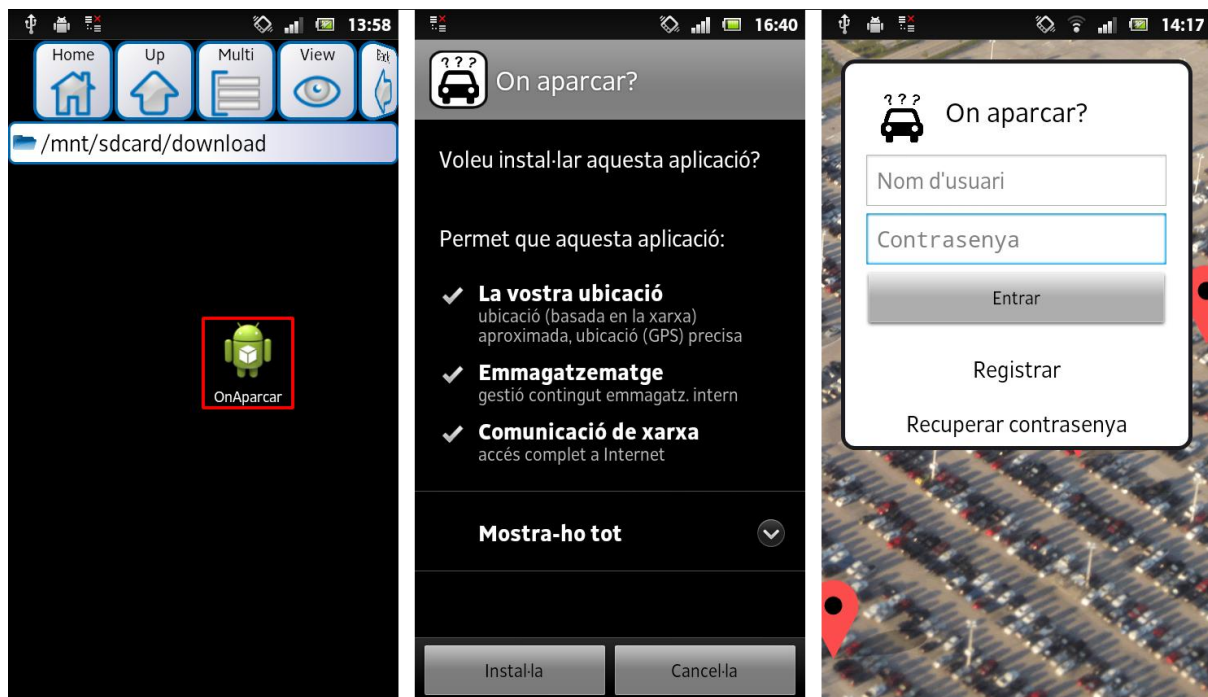


Figura 34: Procediment instal·lació aplicació.

¹⁴ Astro file manager: <https://play.google.com/store/apps/details?id=com.metago.astro>

20.2 Emulador Genymotion

En cas de provar l'aplicació a l'ordinador, es recomana utilitzar l'emulador Genymotion. L'inconvenient recau en el fet que requereix registrar-se per a poder-lo utilitzar, però es compensa amb la rapidesa i facilitat d'us en comparació amb l'emulador de les ADT.

Un cop instal·lat permet crear una maquina virtual d'Android 2.3.7, que esdevé completament compatible amb l'aplicació:

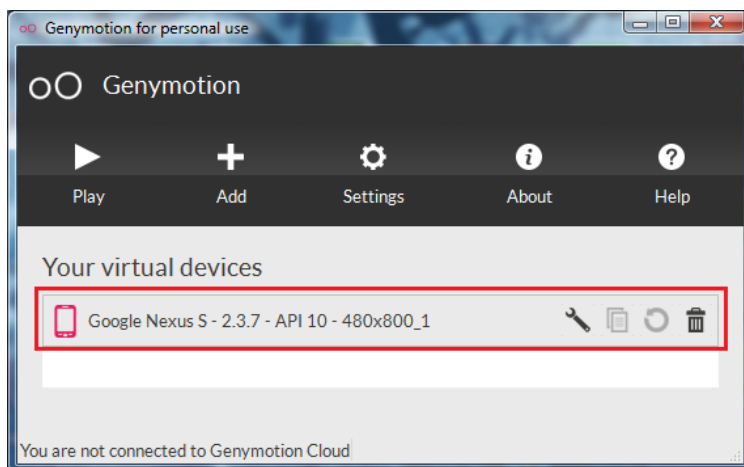


Figura 35: Pantalla principal Genymotion

Un cop encès, per instal·lar l'aplicació només cal arrossegar el fitxer sobre la pantalla de l'emulador, realitzant una acció de *drag & drop*.

Google Maps v2 for Android requereix que el dispositiu tingui instal·lat els Google Play services, per aquest motiu serà necessari instal·lar els següents paquets:

- 1) **Google apps:** gapps-gb-20110828-signed.zip
- 2) **Google Play Store:** com.android.vending-4.6.16.apk
- 3) **Google Play Services:** com.google.android.gms.apk

Com que l'emulació de GPS no acaba de funcionar correctament en cap emulador, la millor opció serà instal·lar alguna aplicació com **GPS Fake Location**, ja sigui utilitzant algun fitxer .apk o utilitzant la pròpia Google Play Store.

21. Instruccions d'ús

Abans de començar amb les explicacions, es recorda a tots els usuaris que qualsevol interacció que es realitzi en el dispositiu, s'ha de dur a terme abans d'arrencar el cotxe i evitar distraccions durant la conducció.

És necessari activar el GPS i l'accés a internet per a poder utilitzar l'aplicació.

21.1 Inici

Identificació

La pantalla inicial de l'aplicació "On aparcar?" correspon a la identificació d'usuaris per tal d'accedir a les funcionalitats d'aquesta.

Per identificar-se és necessari escriure el nom d'usuari i contrasenya en els camps corresponents.

En cas de nous usuaris serà necessari registrar-se, a partir de l'enllaç d'aquesta mateixa pantalla.

En cas d'estar registrat, però no recordar la contrasenya, s'haurà d'accedir a l'apartat "Recuperar contrasenya".

Registre

Els nous usuaris hauran de dur a terme el registre.

Per accedir a l'aplicació serà necessari completar els següents punts:

- **Nom usuari:** nom per identificar-se en l'aplicació.
- **Contrasenya:** contrasenya per accedir.
- **Correu:** correu electrònic de contacte
- **Ciutat:** ciutat on viu l'usuari.
- **Pregunta secreta:** en cas de necessitar canviar la contrasenya, serà un dels camps per a validar la identitat l'usuari.

Un cop realitzat el registre, s'accedeix al menú de l'aplicació.

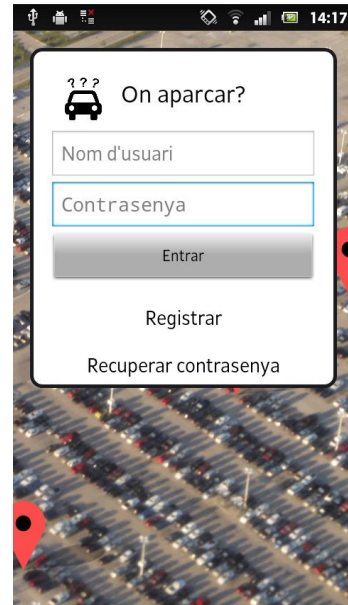


Figura 36: Pantalla identificació

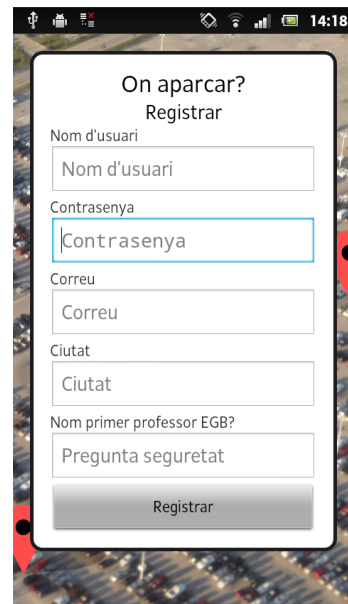


Figura 37: Pantalla registre

Recuperar contrasenya

En cas de no recordar la contrasenya, se'n pot establir una de nova omplint el formulari amb les dades d'autenticació, amb les mateixes dades del procés de registre.

- **Nom usuari:** usuari registrat que vol canviar la contrasenya.
- **Correu:** camp per a validar identitat.
- **Pregunta secreta:** segon camp per a validar identitat.
- **Nova contrasenya:** la contrasenya que s'utilitzarà a partir d'ara.

En cas de poder confirmar la identitat, s'efectuarà el canvi de contrasenya i l'usuari es podrà identificar amb la nova contrasenya.

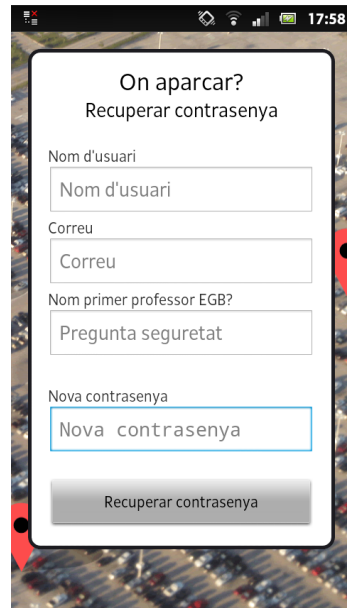


Figura 38: Pantalla canviar contrasenya

21.2 Menú

El menú de l'aplicació presenta dues opcions:

- **Aparcaments propers:** L'objectiu d'aquesta opció és accedir al mapa per visualitzar els aparcament disponibles, sense destí definit.
- **Definir destí:** Opció adequada en cas de realitzar un viatge llarg i voler utilitzar l'aplicació com a GPS. Un cop a destí, es seleccionarà automàticament l'aparcament més pròxim.

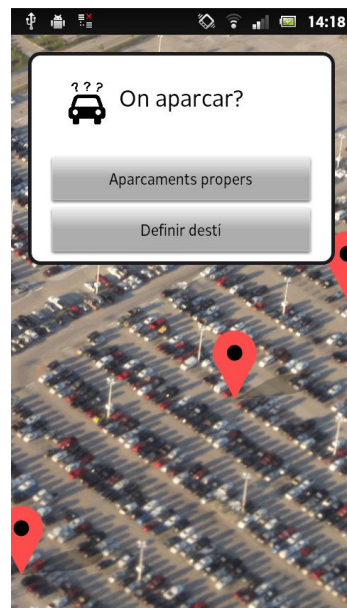





Figura 39: Pantalla menú

21.3 Pantalla Mapa

A la pantalla del mapa es poden realitzar diverses opcions:

Actionbar:

-  Guardar posició del cotxe / anar al cotxe.
-  Marcar un aparcament.
-  Definir un destí.

Menú  / *Overflow* :

- Perfil usuari.
- Ajuda.
- Informació.

Informació dels punters:

- Ocupar aparcaments.

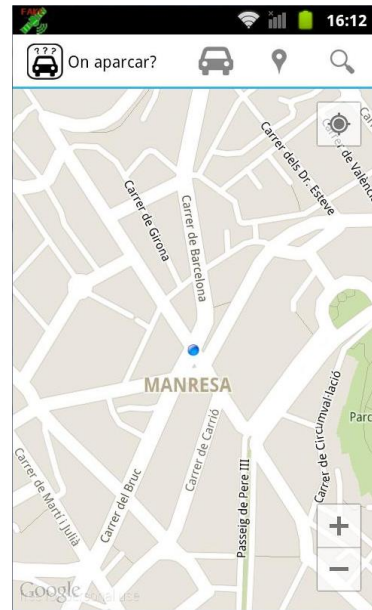




Figura 40: Pantalla mapa

Guardar cotxe

Per registrar la ubicació on s'ha aparcad el cotxe, per després tornar-hi sense problemes, s'ha de polsar el boto: 

El programa agafarà la posició actual del dispositiu, així doncs, per registrar la ubicació correcta, s'ha de realitzar l'acció dins o prop del cotxe aparcad.

La posició del cotxe quedarà marcada amb el marcador: 

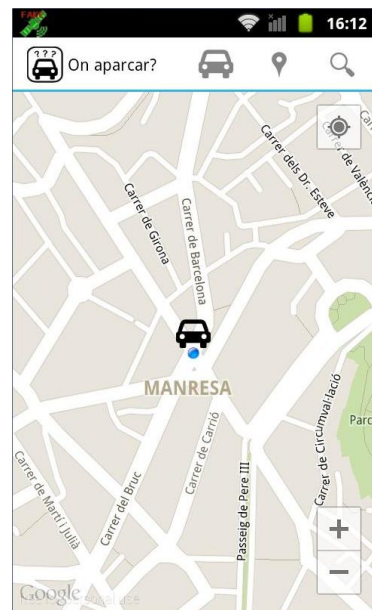



Figura 41: Exemple guardar posició cotxe

Tornar al cotxe

Quan es vulgui tornar al cotxe, serà suficient polsant de nou el boto  i en el mapa apareixerà la millor ruta per arribar-hi.

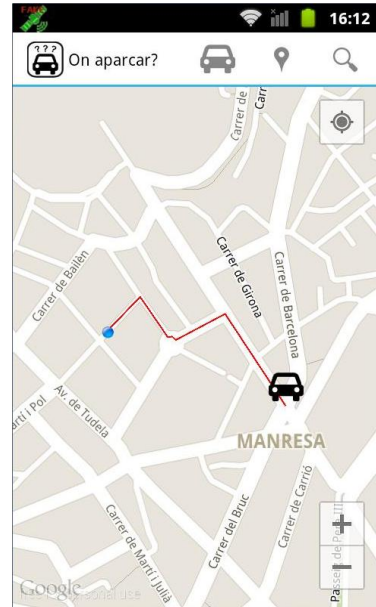



Figura 42: Tornar al cotxe

Marcar aparcament

En cas de voler marcar un aparcament disponible, serà necessari realitzar l'acció abans d'abandonar l'aparcament o estar prop de l'aparcament, amb la finalitat d'aconseguir el màxim de precisió.

L'acció s'inicia amb el boto , el qual carregarà una finestra de diàleg que s'haurà d'omplir amb la següent informació:

- **Número de carrer:** Per tal de tenir un referent.
- **Tipus d'aparcament:** Normal, zona blava o zona verda.
- **Tipus de carrer:** Per determinar la validesa de l'aparcament.

A continuació apareixerà un marcador indicant l'aparcament. Aquest marcador tindrà unes característiques diferents en funció del tipus d'aparcament:

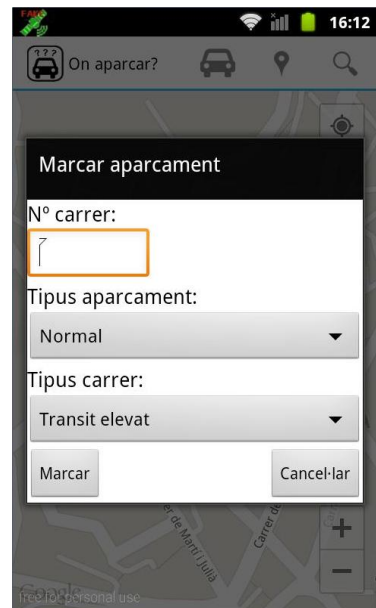











Figura 43: Marcar aparcament

	Trànsit baix	Trànsit mitjà	Trànsit elevat
Aparcament normal			
Zona blava			
Zona verda			

Taula 1: Tipus de marcadors

Anar cap a l'aparcament

En cas d'optar per la cerca manual d'aparcaments, un cop escollit l'aparcament desitjat, hi ha la possibilitat d'obtenir les direccions per arribar fins aquest.

Serà necessari polsar sobre el marcador i sobre el globus d'informació d'aquest.



Figura 44: InfoWindow del marcador

A continuació apareixerà una finestra amb les possibles opcions i informació del carrer, on serà necessari prémer el botó "Anar-hi". A continuació s'obtindran les indicacions.



Figura 45: Anar a l'aparcament

Ocupar aparcament

Un cop s'ha arribat a l'aparcament, és necessari marcar-lo com a ocupat. El procediment consisteix en:

- 1) Polsar sobre el marcador d'aparcament.
- 2) Polsar sobre el globus d'informació del marcador.
- 3) En la finestra que apareix només cal indicar el numero de carrer i polsar "Ocupat!"

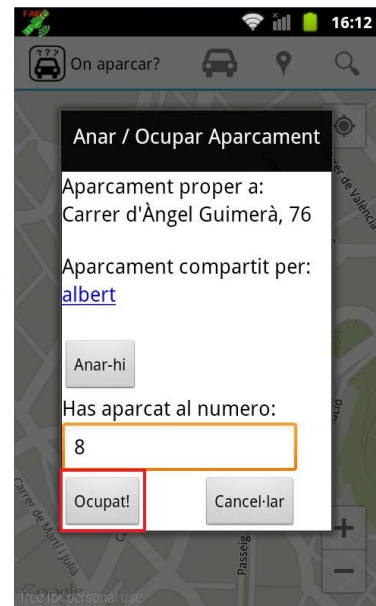



Figura 46: Ocupar aparcament

Definir destí

En cas de realitzar un trajecte llarg i aprofitar el GPS de l'aplicació, s'ha d'utilitzar la opció  de la barra d'acció o escollir "Definir destí" a la pantalla de menú.

A continuació apareixerà una finestra on s'ha d'escriure el nom del carrer, numero i ciutat.

Després de prémer el botó "Buscar", apareixerà un desplegable amb possibles resultats.

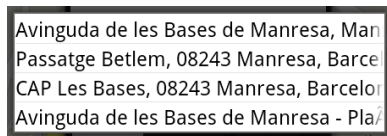


Figura 47: Llista resultats

S'escull el desitjat i a continuació el botó "Anar-hi" i automàticament apareixerà la ruta marcada.

Prop de la destinació, l'aplicació cercarà un aparcament pròxim i canviarà la ruta cap aquest.

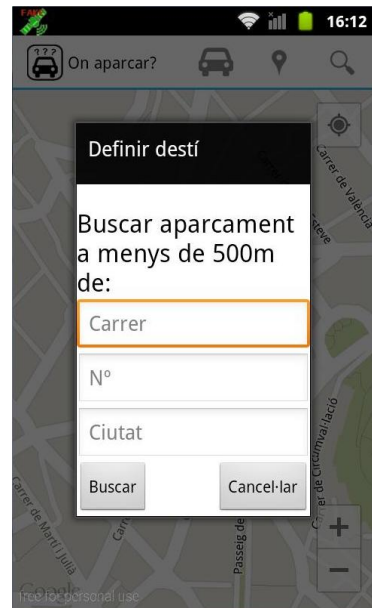


Figura 48: Definir destí

21.4 Perfil usuari

En els perfils s'hi pot consultar la informació de l'usuari i la seva puntuació. Els punts s'obtenen al compartir aparcaments i quan aquests són ocupats i el número de carrer coincideix.

Per a consultar el perfil propi depèn del tipus de dispositiu:



- Dispositiu amb boto de menú : Polsant aquest boto apareixerà l'opció per accedir al perfil.
- Dispositius sense boto menú: A partir del boto *Overflow*  de l'*actionbar*.



Figura 49: Accedir al perfil propi

Per accedir a altres perfils, només serà possible a partir de la informació d'un aparcament que estigui marcat. Entrant a la finestra d'opcions de l'aparcament i polsant sobre el nom de l'usuari que l'ha publicat.



Figura 50: Accedir perfil altres usuaris

22. Bugs

Alguns dels errors o problemes detectats són:

- Com s'esperava, en provar l'aplicació en versions d'Android superiors a 2.3.X sorgeixen problemes d'incompatibilitat en algunes parts del codi, especialment en la carrega del mapa.
- Els emuladors no incorporen tots els programes que formen els Google Play Services, és possible que aquest llenci errors. **Aquests errors no afecten a l'aplicació.**
- En dispositius amb pantalla petita (mdpi¹⁵) hi han problemes de visualització.
- En alguns cassos, al accedir al mapa, la càmera no es situa en al posició d'usuari. No s'ha pogut solucionar ja que el codi es correcte i la majoria de cops funciona. Una solució provisional es utilitzar el boto de google maps per trobar la ubicació d'usuari.

La resta d'errors que s'han pogut solucionar:

- Durant les proves en dispositius, la detecció GPS només funcionava correctament en dispositius reals mentre que en l'emulador no. Aquest fet és degut a que els emuladors de sistemes Android 2.3.X no emulen correctament el posicionament GPS. La solució ha estat utilitzar aplicacions de simulació GPS com *Fake GPS location*¹⁶.
- En les proves amb telèfons intel·ligents, s'ha detectat que les dades d'aparcaments no es rebien correctament i, per tant, feia impossible la interacció amb aquests. El problema era degut a que en l'emulador d'Android, s'utilitzava l'anglès com a llengua del sistema i aquesta separa les coordenades amb un punt, mentre que en dispositius reals s'utilitza com a llengua el català o castellà, utilitzant la coma per al sistema numèric. El fet de registrar les coordenades utilitzant punt i llegir-les utilitzant coma provocava que no s'obtinguessin resultats de la base de dades. Com a solució, en les pàgines de PHP afectades, es realitzarà una substitució de “,” per “.” amb la funció PHP: `str_replace(',', '.', $latitude)`
- Errors típics de programació o en la creació de consultes SQL. Tots corregits durant les proves.

¹⁵ Devices and Displays: <http://developer.android.com/design/style/devices-displays.html>

¹⁶ Fake GPS location: <https://play.google.com/store/apps/details?id=com.lexa.fakegps>

23. Projecció a futur

Tot i que l'aplicació actual compleix amb els objectius proposats i funcionalitats plantejades en l'anàlisi del projecte, possiblement farien falta algunes millores per tal d'arribar a ser una aplicació competent i poder publicar-la.

Algunes d'aquestes millores serien:

- Implementar el sistema d'avatars d'usuari, que tot i estar plantejat, finalment no s'ha implementat, ja que es tractava d'una funcionalitat de baixa prioritat i ha estat necessari prioritzar altres característiques.
- Millorar el sistema de validació d'aparcament. Utilitzar el numero de carrer per a comprovar que es tracta del mateix aparcament és una bona referencia, ja que n'hi ha poques més, però també resulta un camp ambigu, ja que en algunes circumstancies pot ser confús.

Per altra banda, utilitzar la posició GPS tampoc resulta 100% fiable, ja que la precisió varia a cada dispositiu.

- Aplicar un sistema d'enciptació de dades més complex.
- Millorar l'ús del GPS en ruta, oferint indicacions i col·locant la càmera en un punt més idoni per aquesta funció.
- Adequar el codi per a versions Android 3 – 4, més estrictes amb temes de seguretat.
- Millorar el sistema de puntuació, per tal d'obtenir un sistema més atractiu de cara als usuaris i que convidi a participar i col·laborar.
- Utilitzar servidors web i bases de dades amb la capacitat suficient per a suportar el gran nombre de consultes que suposaria la publicació de l'aplicació.
- Implementar en el servidor tasques programades, per exemple amb CRON, per tal de netejar la bases de dades d'aparcament caducats.
- Repassar el codi per tal d'intentar optimitzar-lo en els punts on sigui possible.

- Aplicar millores de disseny gràfic en els menús.
- Adaptar l'aplicació a totes les mides de pantalla, ja sigui adaptant els disseny gràfic o aplicant sistemes de desplaçament vertical en les pantalles.
- Implementar algun sistema per evitar el registre de bots i que omplin la base de dades amb dades falses o generació de brossa.

24. Pressupost

El pressupost es presentarà dividit en les diferents fases de la creació del projecte. Tot i que per a la realització d'aquest projecte no està prevista la publicació de l'aplicació, i que per a les proves s'està utilitzant un servidor proporcionat per la UOC, en el pressupost s'han inclòs les despeses de publicació a Google Play i el cost aproximat d'un servidor.

Tasques	Dies	Cost
Anàlisi		
<ul style="list-style-type: none"> Anàlisi del projecte Definició Planificació 	9	300€
Disseny		
<ul style="list-style-type: none"> Disseny Wireframes Disseny UML Disseny base de dades Disseny gràfic 	18	650 €
Desenvolupament		
<ul style="list-style-type: none"> Desenvolupament aplicació Desenvolupament script PHP Implementació base de dades Realització de proves i tests Correcció codi i disseny 	30	1750€
Altres despeses		
<ul style="list-style-type: none"> Certificat Google Play Lloguer servidor web i BDD 		18,12€ 50€/mes

Taula 2: Pressupost

TOTAL	
Desenvolupament aplicació	2718,12€
Lloguer servidor	50€ / mes

Taula 3: Pressupost total

25. Anàlisi de mercat

25.1 Audiència potencial

Referent als usuaris potencials per a l'aplicació, correspondrien amb els perfils del punt 14:

- Persones que disposin de mòbils intel·ligents amb sistema Android i que coneguin mínimament les funcionalitats d'aquest i com s'estructuren els apartats en Android.
- Persones que realitzen un ús elevat del cotxe.
- Usuaris que encara que no tinguin permís per a conduir, puguin ajudar al conductor utilitzant l'aplicació.
- Residents en ciutats on el trànsit resulta elevat.
- Usuaris disposats a col·laborar, oferint informació d'aparcaments.

25.2 Competència

Existeixen aplicacions similars a "on aparcar?":

- **Aparcar:** Aplicació per trobar aparcament, reservar-lo i pagar. Es basa en un sistema en el qual s'utilitzen una places d'aparcament específiques per al usuari de l'aplicació. Aquest reserven la plaça mitjançant l'aplicació, amb la qual també paguen. Actualment l'aplicació està en proves i només funciona a Oviedo.
- **Aparkarma**¹⁷: Potser l'aplicació més similar. Aquí els usuaris ofereixen les places d'aparcament a canvi de punts. El conductor que ofereix la plaça a d'esperar que arribi l'usuari assignat i aquest últim l'hi donarà un punt a canvi. Com més punts es tenen, més prioritat per obtenir un lloc on aparcar.
- **Aplicacions per recordar aparcament:** "¿Dónde está mi coche?"¹⁸, "MyCar Locator Free"¹⁹ o "Find My Car"²⁰, són alguns exemples d'aplicacions que permeten enregistrar la ubicació d'on s'ha aparcat el cotxe. No representen una competència directa, ja que l'objectiu és diferent, però s'han de tenir en compte, ja que és una funció que també està disponible a "On aparcar?".
- **Waze:** Aplicació GPS la qual proporciona informació sobre el trànsit. La informació del trànsit és aportada pels propis usuaris. Tot i que no és una aplicació dedicada als aparcaments, s'utilitza la mateixa filosofia de que els usuaris proporcionin la informació.

¹⁷ Aparkarma: <http://www.aparkarma.com/>

¹⁸ ¿Dónde está mi coche?: <http://tinyurl.com/kqxfkrg>

¹⁹ MyCar Locator Free: <http://tinyurl.com/ao7ggnr>

²⁰ Find my car: <http://tinyurl.com/beg5baq>

26. Conclusions

Des de l'inici del grau Multimèdia, tenia clar que un dels aspectes en els quals volia aprofundir era en el camp de les aplicacions interactives. Primer va ser amb les pàgines web 2.0, que era un terreny que ja coneixia d'estudis previs, però amb l'augment dels telèfons intel·ligents i el gran nombre de tot tipus d'aplicacions que apareixien per aquests dispositius, vaig tenir clar que per al treball de final de grau havia de desenvolupar una aplicació per a smartphones.

La realització d'aquest projecte m'ha servit, per una banda, per posar en practica els coneixements adquirits al llarg del grau, com són el disseny d'interfícies multimèdia, gestionar la informació a proporcionar a l'usuari, utilitzant els conceptes d'arquitectura de la informació, o realitzar una bona planificació de les tasques. També han estat útils assignatures de caràcter més tècnic per tal d'elaborar una base de dades adequada al tipus d'informació a emmagatzemar i les assignatures de programació, en especial en PHP i de programació orientada a objectes.

Per altra banda, he experimentat el procés de creació des de zero d'una aplicació de mòbil, passant per les diferents fases de desenvolupament, talment com es faria en un projecte professional. Durant aquestes fases de desenvolupament, també he comprovat que la creació d'aplicacions per a mòbil és una tasca realment laboriosa i més difícil del que pot semblar, degut a la gran quantitat d'aplicacions que apareixen diàriament.

En general he quedat molt satisfet dels resultats obtinguts, ja que en primer lloc he complert amb els principals objectius de realitzar una aplicació mòbil i que sigui funcional amb sistemes Android 2.3. En segon lloc, per tot el que he après, tenint en compte que he partit d'un nivell més aviat baix pel que fa a programar en Java, i de total inexperiència en programar directament per Android.

Finalment, com a perspectives de futur, crec que l'aplicació pot resultar prou interessant com per a ser publicada, però caldrien aplicar moltes de les millores que es comenten al punt 23 "Projecció de futur", per tal d'aconseguir una aplicació apta per a tots els sistemes Android i que l'aplicació sigui competent en un mercat en evolució constant.

Annex 1. Lliurables del projecte

Documentació

- **PAC_FINAL_mem_JuncàTatjé_Albert.pdf:** Memòria del projecte.
- **Autoinforme_d'avaluació_JuncàTatjé_Albert.pdf:** Autoinforme avaluant l'ús de les competències transversals.

Presentacions

- **PAC_FINAL_vid_JuncàTatjé_Albert.flv:** Presentació en vídeo del projecte.
- **PAC_FINAL_prs_JuncàTatjé_Albert.pdf:** Presentació escrita-visual del projecte.

Projecte

- **PAC_FINAL_prj_JuncàTatjé_Albert.zip:** Fitxer comprimit amb els fitxers del projecte, codi font i altres arxius de treball o “màster”.

L'estructura és la següent:

- **OnAparcar.apk:** Paquet instal·lable per a Android del projecte.
- **Manual_TFG_Albert_Junca.pdf:** Instruccions d'instal·lació i d'us de l'aplicació.
- **Codi font:**
 - **BBDD:** Fitxer SQL ajuncat.sql amb l'estructura de les taules de la Base de dades.
 - **PHP:** Fitxers que s'allotjaran al servidor web (PHP, HTML, CSS i Manual PDF).
 - **Projecte Eclipse:** Conte el directori corresponent al projecte per a Eclipse.
- **Disseny gràfic:** Arxius “màster” del disseny gràfic i interfície del projecte.
- **Esquemes Dia:** Arxius “màster” dels esquemes UML i de la base de dades, creats amb Dia.

- **Recursos emulador:** En cas d'utilitzar algun emulador per a provar l'aplicació, es possible que vingui sense els programes necessaris per a fer funcionar l'aplicació:
 - **gapps-gb-20110828-signed.zip:** Últimes Google apps per a dispositiu 2.3.X
 - **com.android.vending-4.6.16.apk:** Google Play Store.
 - **com.google.android.gms.apk:** Google Play Services.
 - **com.lexa.fakegps.apk:** Aplicació Fake GPS Location, per a simular diferents ubicacions GPS.

- **Wireframes:** Arxius "màster" dels wireframes dissenyats per al projecte.

Annex 2. Codi font (extractes)

Annex 2.1 Manifest.xml

Fitxer fonamental de l'aplicació. Especifica versions compatibles del sistema, permisos necessaris, etc.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.albertTFG.onaparcar"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="10"
        android:targetSdkVersion="10" />

    <!-- List of permissions for application -->
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="com.google.android.providers.
        gsf.permission.READ_GSERVICES" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <!-- Detect current location -->
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

    <!-- Google Maps V2 needs OpenGL ES 2.0. -->
    <uses-feature
        android:glEsVersion="0x00020000"
        android:required="true" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/icon"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.albertTFG.onaparcar.LoginActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.
                    category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name="com.albertTFG.onaparcar.RegisterActivity"
            android:label="@string/title_activity_register" >
        </activity>
        <activity
            android:name="com.albertTFG.onaparcar.AparcarActivity"
            android:label="@string/title_activity_aparcar"
            android:theme="@style/Theme.AppCompat.Light" >
        </activity>

        <!-- Google Services library -->
        <meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />
    </application>
</manifest>
```

```

<!-- API key for Google Maps V2 -->
<meta-data
    android:name="com.google.android.maps.v2.API_KEY"
    android:value="YOUR_API_KEY" />

<activity
    android:name="com.albertTFG.onaparcar.MenuActivity"
    android:label="@string/title_activity_menu" >
</activity>
<activity
    android:name="com.albertTFG.onaparcar.RecoveryActivity"
    android:label="@string/title_activity_recovery" >
</activity>
</application>

</manifest>

```

Annex 2.2 LoginActivity

Pantalla inicial, on es pot observar diferents mecanismes que es repeteixen al llarg de l'aplicació.

Inici – definició variables:

```

//Creating shared preferences, to save login information
private SharedPreferences prefs;

// Variable for Button Login
private Button btLogin;
// Variables for Labels
private TextView lblRegister;
private TextView lblRecovery;
// Variables for EditText fields
private EditText txtUser;
private EditText txtPass;

// Creating JSON Parser object
JSONParser jsonParser = new JSONParser();

// URL to addUser PHP script
private static String url_get_user =
"http://multimedia.uoc.edu/~ajuncat/android/getUser.php";

```

Constructor:

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //Remove title bar
    this.requestWindowFeature(Window.FEATURE_NO_TITLE);
    setContentView(R.layout.activity_login);

    // Assign visual elements Button to their variables and add ClickListener
    txtUser = (EditText) findViewById(R.id.userText);
    txtPass = (EditText) findViewById(R.id.passText);
    btLogin = (Button) findViewById(R.id.btn_login);
}

```

Primer exemple d'element clicable per al boto "Entrar":

```

btLogin.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // onClick execute getUser method:
        // Get fields values
        String userString = txtUser.getText().toString();
        String passString = txtPass.getText().toString();
    }
});

```

Primer exemple de connexió a base de dades amb JSON:

```

// Building parameters for the login request
List<NameValuePair> params = new ArrayList<NameValuePair>();
params.add(new BasicNameValuePair("user", userString));
params.add(new BasicNameValuePair("passw", passString));

// Getting JSON Object with results of the HttpRequest
// Specifying the URL, method and request parameters
JSONObject json = jsonParser.makeHttpRequest(url_get_user,
    "POST", params);

// Check for success tag from JSONObject
try{
    int success = json.getInt("success");
    if (success == 1){

```

Primer exemple d'ús de les SharedPreferences:

```

// If successfully added user:
// Save shared preferences with logged user data
SharedPreferences prefs = getSharedPreferences
    ("AparcarPref", Context.MODE_PRIVATE);
SharedPreferences.Editor editor = prefs.edit();

editor.putBoolean("login", true);
editor.putInt("userId", json.getInt("id"));
editor.putBoolean("userCar", false);
editor.commit();

```

Canvi de pantalla:

```

// And go to the app menu activity
Intent i = new Intent(getApplicationContext(),
    MenuActivity.class);
startActivity(i);

// Closing this screen
finish();
}
//else if (success == 0){
else{
    // "success" == 0: No results from database with that
    parameters
    // Show a message
    Toast.makeText(getApplicationContext(), "Usuari o
        contrasenya incorrectes!!!",
        Toast.LENGTH_LONG).show();
}
}
catch (JSONException e){
    e.printStackTrace();
}
});

```

Annex 2.3 Carregar Google Maps

Funció `initilizeMap()` encarregada de comprovar que el mapa estigui carregat i carregar-lo en cas contrari. Un cop carregat es procedeix a configurar el mapa `mapParameters()` i obtenir els marcadors de la base de dades `loadMarkers()`.

```
private void initilizeMap() {
    // Check if Map is initialized
    if (map == null) {
        map = ((SupportMapFragment) getSupportFragmentManager()
            .findFragmentById(R.id.map)).getMap();

        // check if map is created successfully or not
        if (map == null) {
            Toast.makeText(getApplicationContext(), "Impossible carregar el mapa!",
                Toast.LENGTH_SHORT).show();
        }
        else {
            // If map is created, go set parameters
            // Configure the map
            mapParameters();

            // Load Markers (Parkings) from database
            loadMarkers();
        }
    }
}
```

Annex 2.3 Carregar Google Maps

Un cop el mapa es funciona, s'obté la posició del dispositiu i es configura el mapa per tal que es mostri la ubicació amb la pantalla centrada:

```
// Configure the map
private void mapParameters() {
    // Get LocationManager from System Service LOCATION_SERVICE
    locationManager = (LocationManager)
        getSystemService(Context.LOCATION_SERVICE);

    // Request location parameters: use GPS,
    // check location every 3 seconds (3000 milliseconds), min distance for update
    // 10 meters
    locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
        3000, 10, this);

    LatLng latLng = new LatLng(latitude, longitude);

    // Enabling MyLocation icon of GoogleMap
    map.setMyLocationEnabled(true);

    // Move camera to user location
    map.moveCamera(CameraUpdateFactory.newLatLng(latLng));

    // Zoom in the user location
    map.animateCamera(CameraUpdateFactory.zoomTo(100));
}
```

Annex 2.4 Canvi díigits dades GPS

Una de les maneres per obtenir la informació dels aparcaments és comparant les coordenades del programa amb les de la base de dades. El problema es que en registrar una ubicació proporcionada pel GPS, aquesta només conte 6 decimals (##.#####) i la dada proporcionada pels marcadors de Google Maps està formada per 12 decimals (##.#####) fent impossible la comparació per SQL. La solució consisteix en transformar les coordenades a 6 decimals i poder comparar amb la taula sense problemes:

```
// Building parameters for the request
// GPS Location returns a coordinate format ##.#####
// Google Maps Marker returns a coordinate format ##.##### making the SQL
search impossible
// We use DecimalFormat("##.#####") to change the Marker coordinate format
List<NameValuePair> params = new ArrayList<NameValuePair>();
params.add(new BasicNameValuePair("latitude", new
    DecimalFormat("##.#####").format(marker.getPosition().latitude)));
params.add(new BasicNameValuePair("longitudo", new
    DecimalFormat("##.#####").format(marker.getPosition().longitudo)));
```

Annex 2.5 Afegir marcadors diferents

Per a determinar si posar un marcador verd o blau, s'utilitza el sistema `switch...case` amb el tipus d'aparcament:

```
private void drawMarkers(double latitudeM, double longitudeM, int parkingType, int time,
String street){
    // Depending on parkingType, set different color for marker
    switch (parkingType) {
        case 0:
            String iconWhite="marker_normal_"+time;
            int iconWhiteID = getResources().getIdentifier(iconWhite,
                "drawable", getPackageName());

            map.addMarker(new MarkerOptions()
                .position(new LatLng(latitudeM, longitudeM))
                .icon(BitmapDescriptorFactory.fromResource(iconWhiteID))
                .title(street));
            break;
        case 1:
            String iconBlue="marker_blue_"+time;
            int iconBlueID = getResources().getIdentifier(iconBlue,
                "drawable", getPackageName());

            map.addMarker(new MarkerOptions()
                .position(new LatLng(latitudeM, longitudeM))
                .icon(BitmapDescriptorFactory.fromResource(iconBlueID))
                .title(street));
            break;
        case 2:
            String iconGreen="marker_green_"+time;
            int iconGreenID = getResources().getIdentifier(iconGreen,
                "drawable", getPackageName());

            map.addMarker(new MarkerOptions()
                .position(new LatLng(latitudeM, longitudeM))
```



```

        .icon(BitmapDescriptorFactory.fromResource(iconGreenID))
        .title(street));
        break;
    case 3:
        // Case 3 if for usercar. If put car marker, modify shared
        // preferences
        // Modify shared preferences to avoid add more car markers
        SharedPreferences.Editor editor = prefs.edit();
        editor.putBoolean("userCar", true);
        editor.commit();
        //set car coordinates, if user want to search route
        carLatitude=latitudeM;
        carLongitude=longitudeM;

        map.addMarker(new MarkerOptions()
            .position(new LatLng(latitudeM, longitudeM))
            .icon(BitmapDescriptorFactory
                .fromResource(R.drawable.marker_car))
            .title("El meu cotxe"));
        break;
    }
}

```

Annex 2.6 Consultes SQL

1) Consulta per obtenir només els aparcament que no han caducat.

```
SELECT * FROM Parking WHERE TIMESTAMPDIF(MINUTE, time, NOW()) < permanency
```

2) Consulta, utilitzada per Google Maps²¹, per cercar elements pròxims a una coordenada concreta, en milles. Adaptada per obtenir resultats no caducats i a una distància inferior a 500 metres (0.310686 milles):

```

SELECT latitude,longitude,time,permanency,
( 3959 * acos( cos( radians('".$latitude.$') ) *
cos( radians( latitude ) ) *
cos( radians( longitude ) -
radians('".$longitude.$') ) +
sin( radians('".$latitude.$') ) *
sin( radians( latitude ) ) ) )
AS distance FROM Parking HAVING distance < '0.310686'
AND TIMESTAMPDIF(MINUTE, time, NOW()) < permanency
ORDER BY distance ASC LIMIT 0, 1

```

²¹ Google Maps API Samples

https://code.google.com/p/gmaps-samples/source/browse/trunk/articles-phpsqlsearch/phpsqlsearch_genxml.php?r=1560

Annex 3. Codi extern utilitzat

Annex 3.1 JSONParser

Codi de Ravi Tamada (CC BY). Aquesta classe permet que l'aplicació envii i rebi dades en format JSON utilitzant una URL i els mètodes GET o POST. S'utilitza per la transferència de dades amb la base de dades.

```
public class JSONParser {

    static InputStream is = null;
    static JSONObject jsonObj = null;
    static String json = "";

    // constructor
    public JSONParser() {

    }

    // function get json from url
    // by making HTTP POST or GET mehtod
    public JSONObject makeHttpRequest(String url, String method,
        List<NameValuePair> params) {

        // Making HTTP request
        try {

            // check for request method
            if(method == "POST"){
                // request method is POST
                // defaultHttpClient
                DefaultHttpClient httpClient = new
                    DefaultHttpClient();
                HttpPost httpPost = new HttpPost(url);
                httpPost.setEntity(new
                    UrlEncodedFormEntity(params));

                HttpResponse httpResponse =
                    httpClient.execute(httpPost);
                HttpEntity httpEntity = httpResponse.getEntity();
                is = httpEntity.getContent();

            }else if(method == "GET"){
                // request method is GET
                DefaultHttpClient httpClient = new
                    DefaultHttpClient();
                String paramString = URLEncodedUtils.format(params,
                    "utf-8");
                url += "?" + paramString;
                HttpGet httpGet = new HttpGet(url);

                HttpResponse httpResponse =
                    httpClient.execute(httpGet);
                HttpEntity httpEntity = httpResponse.getEntity();
                is = httpEntity.getContent();

            }

        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        } catch (ClientProtocolException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```

    }

    try {
        BufferedReader reader = new BufferedReader(new InputStreamReader(
            is, "iso-8859-1"), 8);
        StringBuilder sb = new StringBuilder();
        String line = null;
        while ((line = reader.readLine()) != null) {
            sb.append(line + "\n");
        }
        is.close();
        json = sb.toString();
    } catch (Exception e) {
        Log.e("Buffer Error", "Error converting result " + e.toString());
    }

    // try parse the string to a JSON object
    try {
        jsonObj = new JSONObject(json);
    } catch (JSONException e) {
        Log.e("JSON Parser", "Error parsing data " + e.toString());
    }

    // return JSON String
    return jsonObj;
}
}

```

A continuació es troba la part corresponent al registre de dades, adaptada al registre d'usuaris:

```

/**
 * Adapted from
 * http://www.androidhive.info/2012/05/how-to-connect-android-with-php-mysql/ by Ravi
 * Tamada CC BY
 * Background Async Task to Create new product
 */
class addNewUser extends AsyncTask<String, String, String> {

    /**
     * Before starting background thread Show Progress Dialog
     */
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        progressDialog = new ProgressDialog(RegisterActivity.this);
        // Add text to Progress Dialog
        progressDialog.setMessage("Registrant usuari..");
        progressDialog.setIndeterminate(false);
        progressDialog.setCancelable(true);
        progressDialog.show();
    }

    /**
     * Add new user
     */
    protected String doInBackground(String... args) {
        //Get fields values
        String userString = txtUser.getText().toString();
        String passString = txtPass.getText().toString();
        String mailString = txtMail.getText().toString();
        String cityString = txtCity.getText().toString();
        String imageString = txtImage.getText().toString();

        // Building Parameters
    }
}

```

```

List<NameValuePair> params = new ArrayList<NameValuePair>();
params.add(new BasicNameValuePair("user", userString));
params.add(new BasicNameValuePair("passw", passString));
params.add(new BasicNameValuePair("mail", mailString));
params.add(new BasicNameValuePair("city", cityString));
params.add(new BasicNameValuePair("image", imageString));

// getting JSON Object
// Note that add user url accepts POST method
JSONObject json = jsonParser.makeHttpRequest(url_create_product,
        "POST", params);

// check log cat fro response
Log.d("Create Response", json.toString());

// check for success tag
try {
    int success = json.getInt(TAG_SUCCESS);

    if (success == 1) {
        // If successfully added user, go to main app activity
        Intent i = new Intent(getApplicationContext(),
                AparcarActivity.class);
        startActivity(i);

        // closing this screen
        finish();
    } else {
        // failed to add user
        // TODO Add fail dialog
    }
} catch (JSONException e) {
    e.printStackTrace();
}

return null;
}

```

Annex 3.2 Decoding Google Maps Direction API JSON

Mètode per a obtenir tota la informació de les rutes, obtinguda de l'API Google Direction.

Funció de George Mathew.

```

// Decode the information from JSON and draw directions on map
// Code adapted from the original code of George Mathew
//http://wptrafficanalyzer.in/blog/driving-route-from-my-location-to-destination-in-
-google-maps-android-api-v2/
List<List<HashMap<String, String>>> routes = new
ArrayList<List<HashMap<String, String>>>();
JSONArray routesJSON = null;
JSONArray legsJSON = null;
JSONArray stepsJSON = null;

routesJSON = jsonDirections.getJSONArray("routes");

//Traversing all routes
for(int x=0;x<routesJSON.length();x++){
    legsJSON = (JSONObject)routesJSON.get(x).getJSONArray("legs");
    List pathRoute = new ArrayList<HashMap<String, String>>();

    //Traversing all legs
    for(int c=0;c<legsJSON.length();c++){
        stepsJSON = (JSONObject)legsJSON.get(c).getJSONArray("steps");
    }
}

```

```

//Traversing all steps
for(int y=0;y<stepsJSON.length();y++){
    String polyline = "";
    polyline = (String)((JSONObject)((JSONObject)stepsJSON.get(y))
        .get("polyline")).get("points");
    List<LatLng> list = decodePoly(polyline);

    //Traversing all points
    for(int l=0;l<list.size();l++){
        HashMap<String, String> pointMap =
            new HashMap<String, String>();
        pointMap.put("lat",
            Double.toString(((LatLng)list.get(l)).latitude));
        pointMap.put("lng",
            Double.toString(((LatLng)list.get(l)).longitude));
        pathRoute.add(pointMap);
    }
    routes.add(pathRoute);
}

ArrayList<LatLng> pointsRoute = null;
PolylineOptions lineOptions = null;

// Traversing through all the routes
for(int il=0;il<routes.size();il++){
    pointsRoute = new ArrayList<LatLng>();
    lineOptions = new PolylineOptions();

    // Fetching i-th route
    List<HashMap<String, String>> pathList = routes.get(il);

    // Fetching all the points in i-th route
    for(int y=0;y<pathList.size();y++){
        HashMap<String,String> point = pathList.get(y);

        double lat = Double.parseDouble(point.get("lat"));
        double lng = Double.parseDouble(point.get("lng"));
        LatLng position = new LatLng(lat, lng);

        pointsRoute.add(position);
    }

    // Adding all the points in the route to LineOptions
    lineOptions.addAll(pointsRoute);
    lineOptions.width(2);
    lineOptions.color(Color.RED);
}

// Drawing polyline in the Google Map for the i-th route
map.addPolyline(lineOptions);

```

Annex 3.3 Decoding Polylines from Google Maps Direction

L'API de Google Directions retorna els punts de ruta codificats. A l'aplicació s'utilitza el següent mètode de Jeffrey Sambells, per tal d'obtenir les coordenades dels punts de ruta.

El mètode rep un String amb els punts codificat i retorna una llista de coordenades:

```
/**
 * Jeffrey Sambells's Method to decode polyline points
 *
 * http://jeffreysambells.com/2010/05/27/decoding-polylines-from-google-maps-direction-api-with-java
 * */
private List<LatLng> decodePoly(String encoded) {

    List<LatLng> poly = new ArrayList<LatLng>();
    int index = 0, len = encoded.length();
    int lat = 0, lng = 0;

    while (index < len) {
        int b, shift = 0, result = 0;
        do {
            b = encoded.charAt(index++) - 63;
            result |= (b & 0x1f) << shift;
            shift += 5;
        } while (b >= 0x20);
        int dlat = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
        lat += dlat;

        shift = 0;
        result = 0;
        do {
            b = encoded.charAt(index++) - 63;
            result |= (b & 0x1f) << shift;
            shift += 5;
        } while (b >= 0x20);

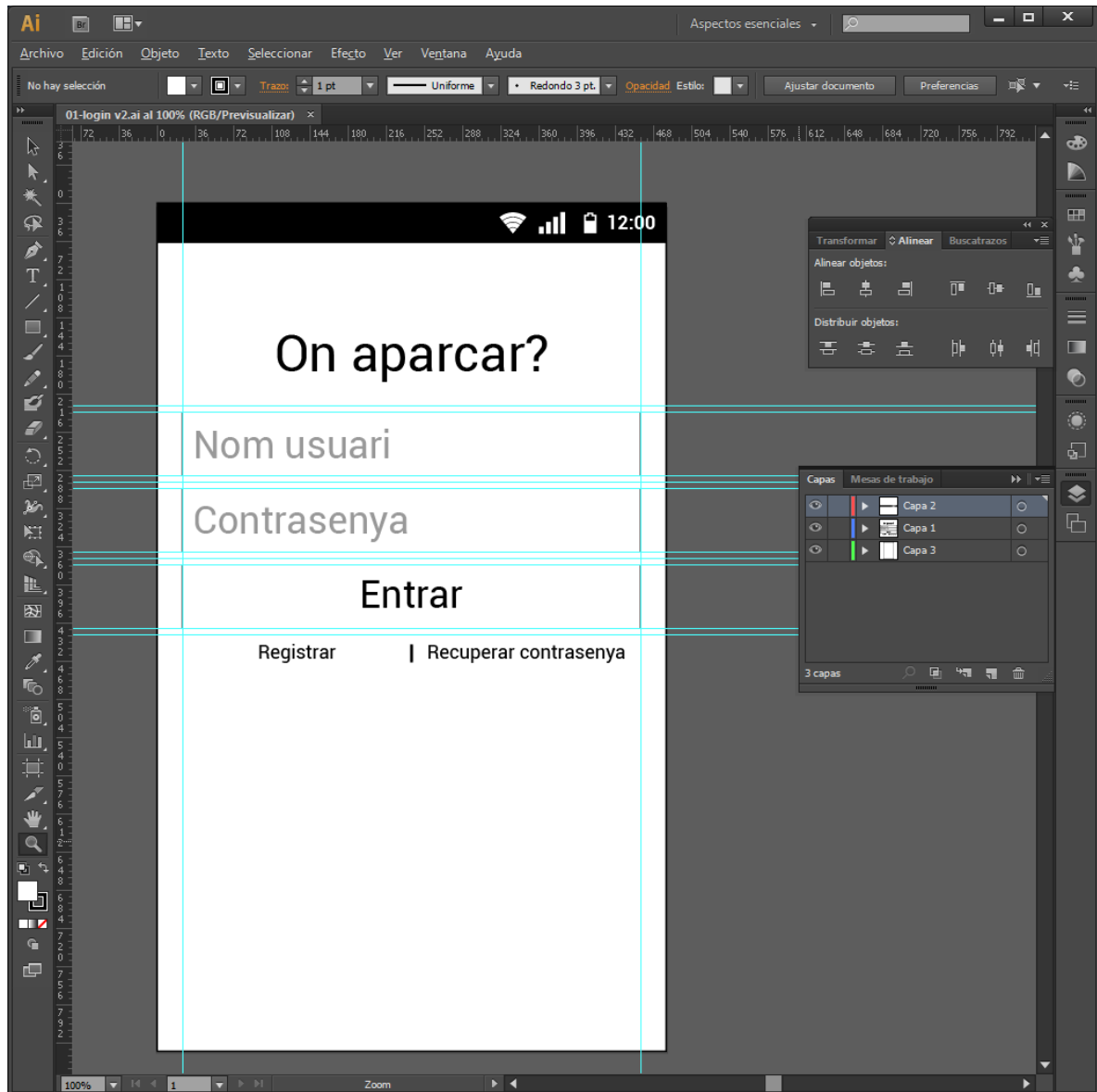
        int dlng = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
        lng += dlng;

        LatLng p = new LatLng((((double) lat / 1E5)),
                               (((double) lng / 1E5)));
        poly.add(p);
    }

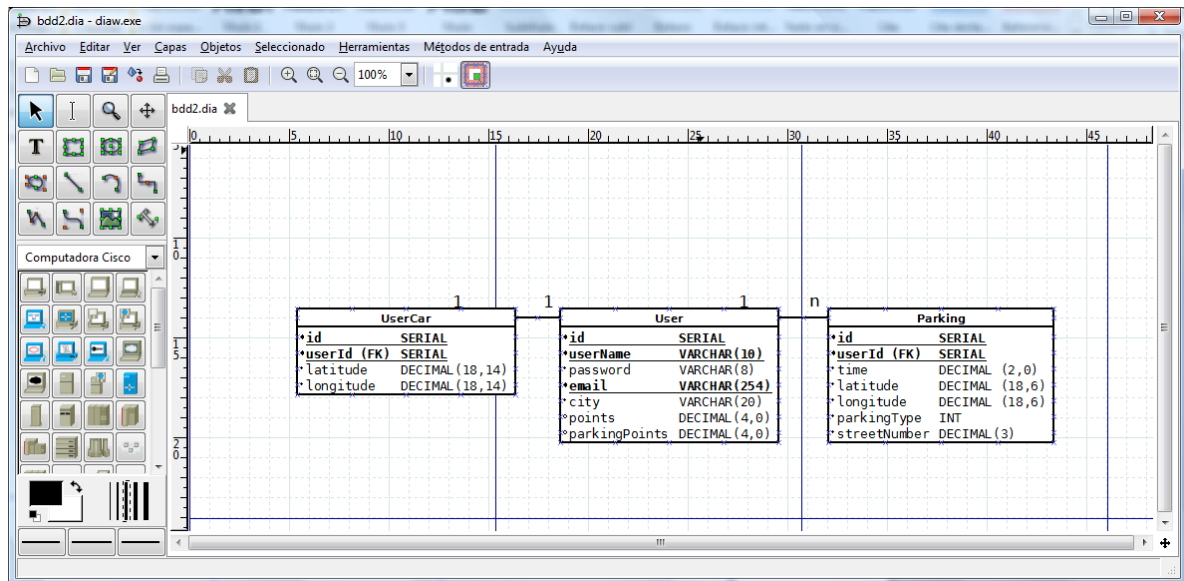
    return poly;
}
```

Annex 4. Captures de pantalla

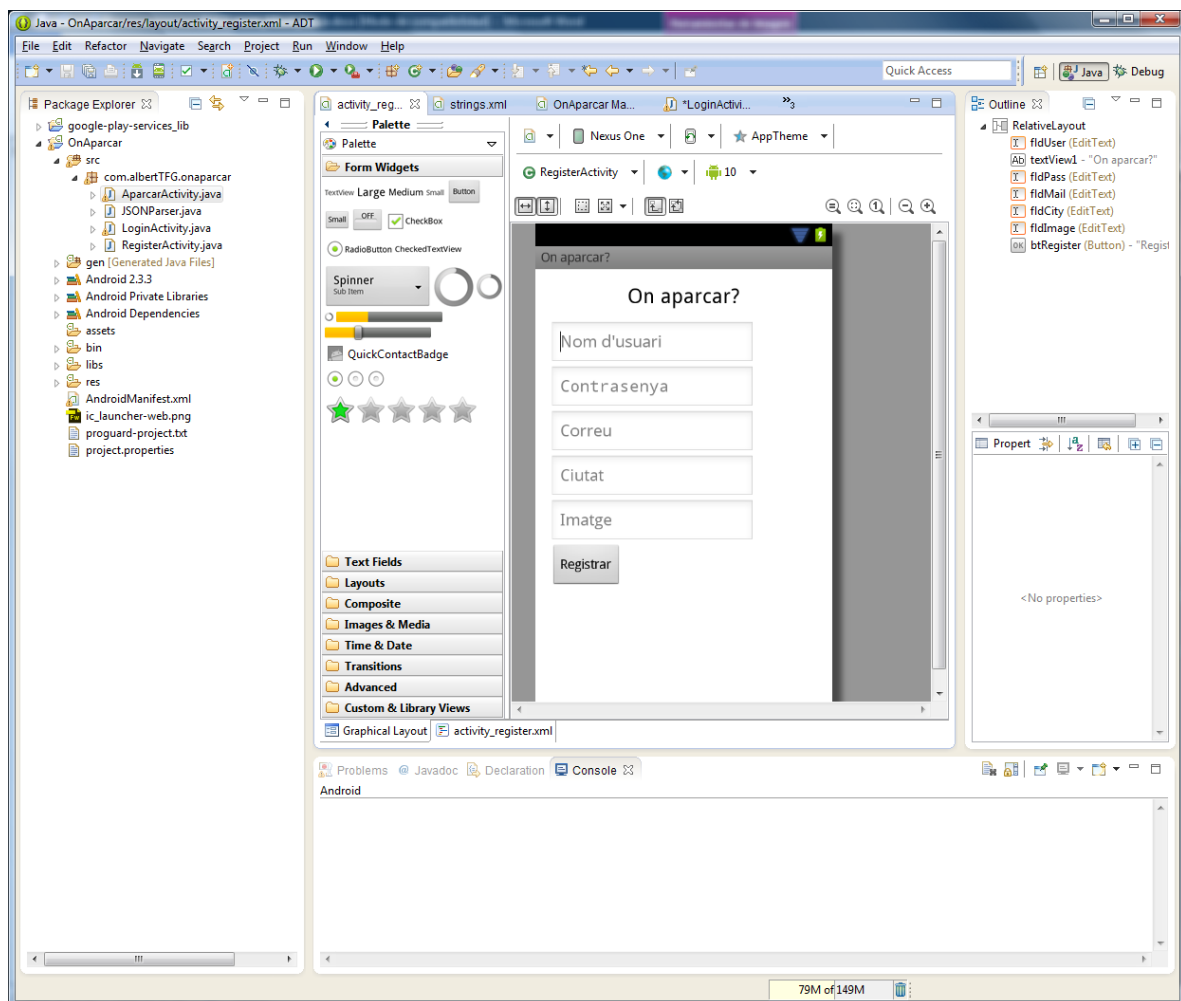
Disseny wireframes amb Adobre Illustrator CS6:



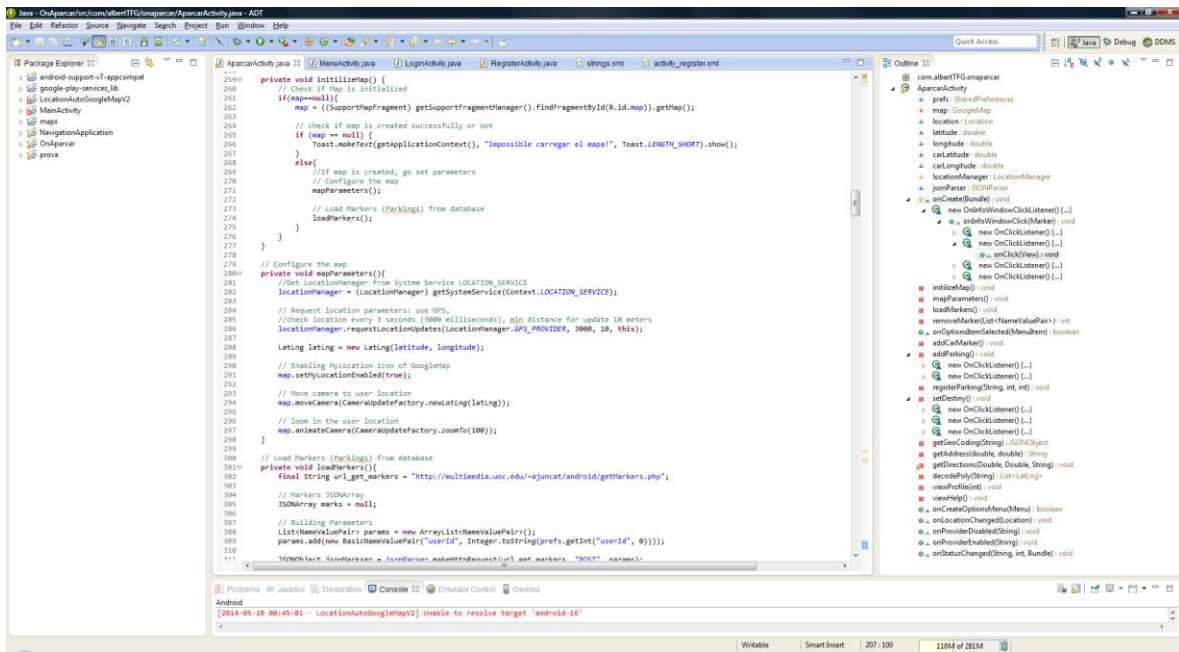
Disseny esquema base de dades amb Dia:



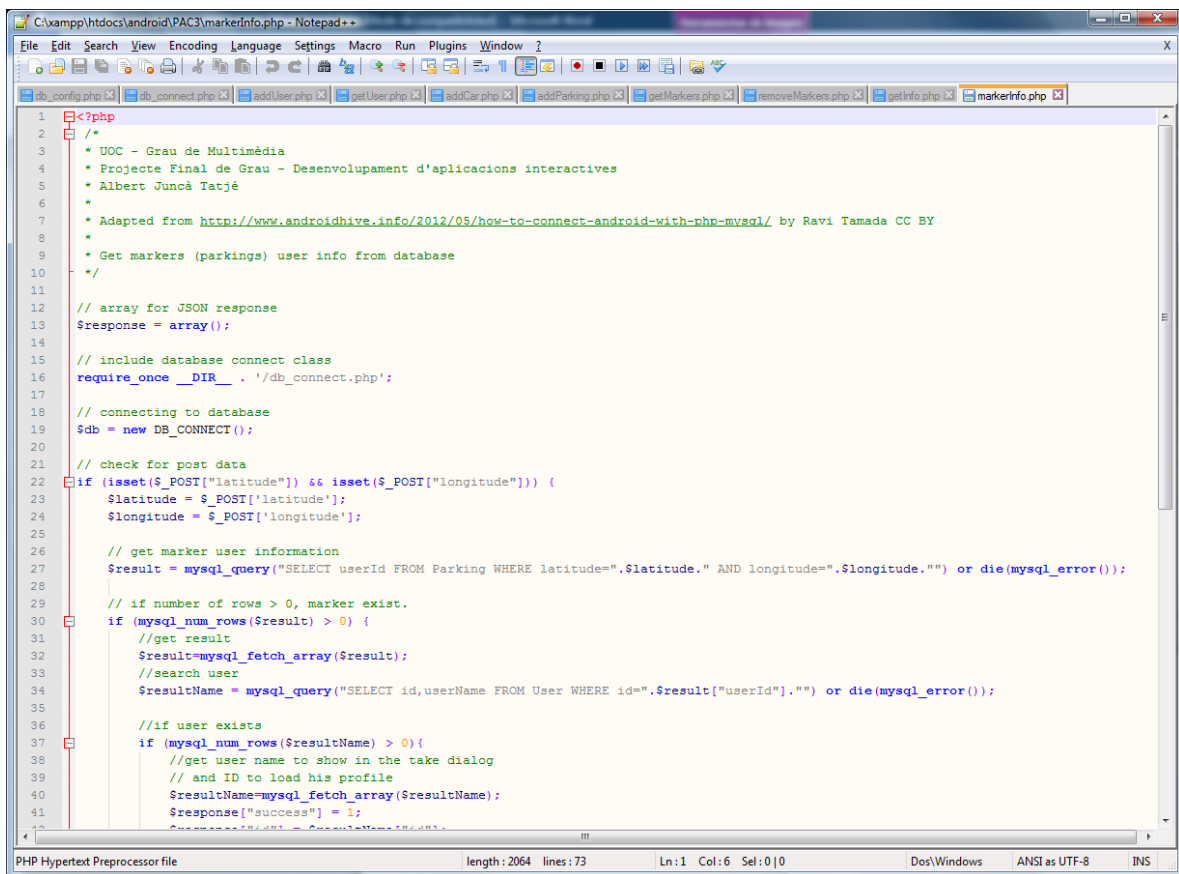
Creació pantalles de l'aplicació amb Eclipse:



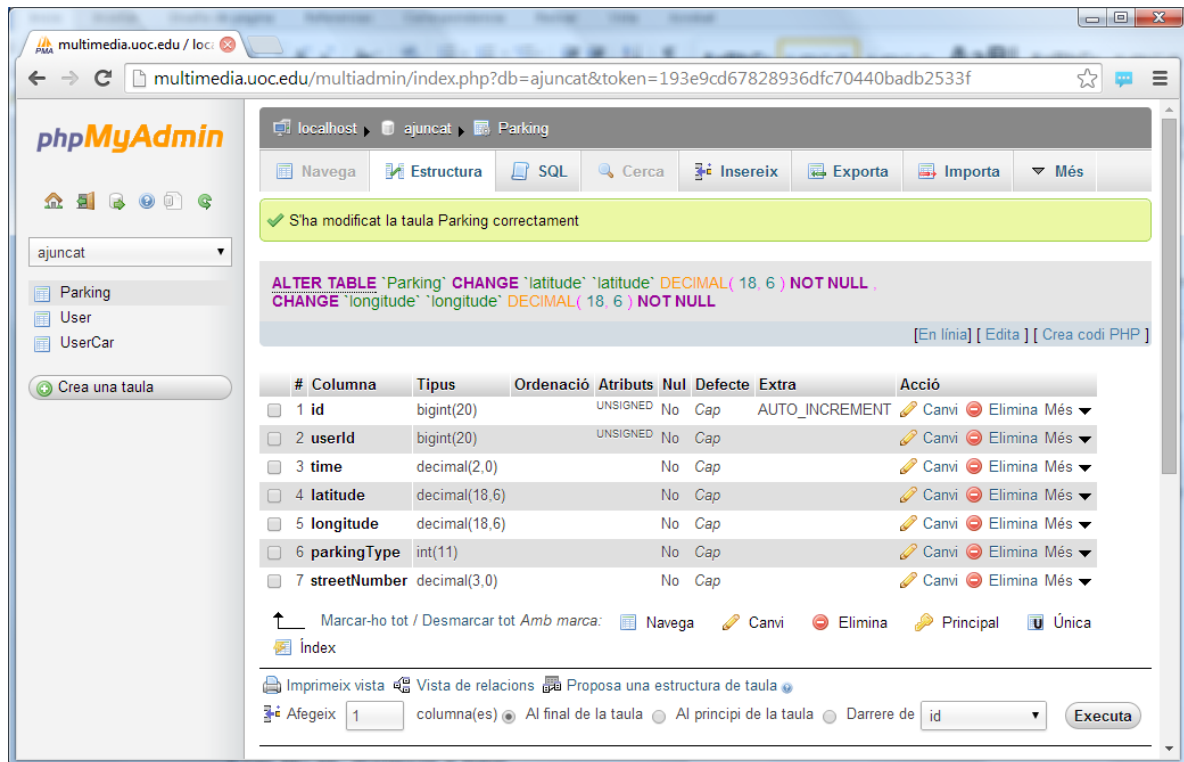
Programació amb Eclipse:



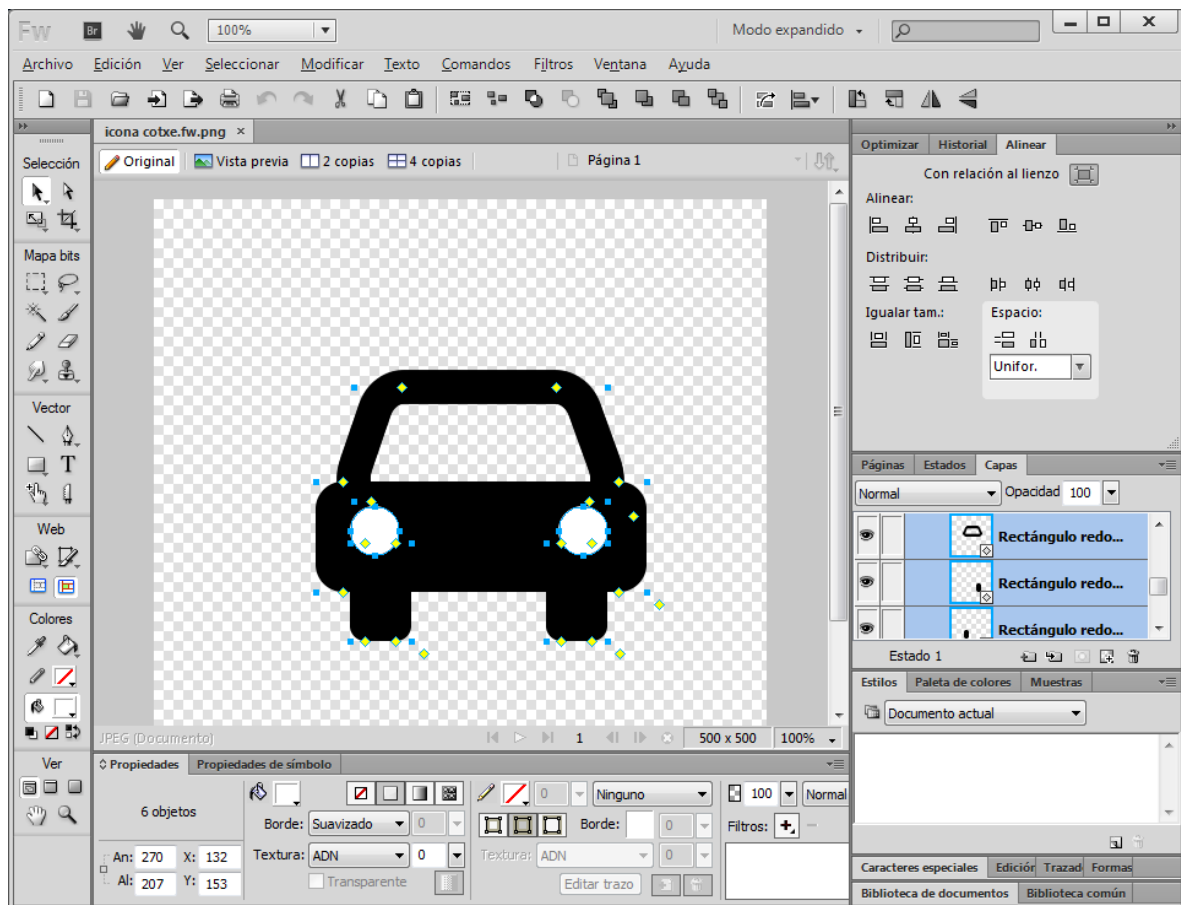
Creació scripts PHP amb notepad++:



Gestió base de dades MySQL amb phpMyAdmin:



Creació d'elements gràfics amb Adobe Fireworks CS6:



Annex 5. Glossari

- **Android:** Sistema operatiu per a telèfons mòbils.
- **Android development tools (ADT):** Extensió per a Eclipse que permet la creació d'aplicacions Android amb aquest programa.
- **Application programming interface (API):** Conjunt de mètodes que ofereix una llibreria.
- **Creative Commons:** Organització sense ànim de lucre dedicada a reduir les barreres legals per a compartir treballs creatius.
- **Graphical user interface (GUI):** Interfície d'usuari composta per elements gràfics, buscant una interacció intuïtiva.
- **Integrated development environment (IDE):** Eines informàtiques per a facilitar i agilitzar el desenvolupament de programes.
- **Java:** Llenguatge de programació orientat a l'objecte.
- **JavaScript object notation (JSON):** Format lleuger que s'utilitza en l'intercanvi de dades estructurades entre diferents plataformes.
- **My structured query language (MySQL):** Sistema de gestió de bases de dades relacional.
- **PHP Hipertext preprocessor (PHP):** Llenguatge de programació d'alt nivell utilitzat per a generar pàgines web amb contingut dinàmic.
- **Script:** Conjunt d'instruccions d'un llenguatge de programació.
- **Smartphone:** Telèfon intel·ligent el qual permet instal·lar sistema operatiu i aplicacions.
- **Software development kit (SDK):** Conjunt d'eines per a la creació de programes per a un sistema concret.
- **Wireframe:** Representació esquemàtica de la interfície d'usuari. S'utilitza per a organitzar la distribució d'elements i contingut.
- **Unified modeling language (UML):** Representació gràfica de les classes que formen un programa.

Annex 6. Bibliografia

Abelló Gamazo, Albert; Cabré i Segarra, Blai; Costal Costa, Dolors; Rodríguez González, M. Elena; Segret i Sala, Ramon; Sistac i Planas, Jaume; Urpí Tubella, Toni (2010). Ús de bases de dades. Barcelona: FUOC.

Arnedo Moreno, Joan; Riera i Terrén Professorat, Daniel (2010). Disseny i programació orientada a objectes. Barcelona: FUOC.

Costal Costa, Dolors. Disseny de bases de dades. Barcelona: FUOC.

de Fuenmayor López, Daniel; González Sancho, Marcos (2012). Aplicacions Rich Media. Barcelona: FUOC.

Mathew , George (10 de 5 de 2013). Driving route from my location to destination in Google Maps Android API V2 (Step 11).

Obtingut de

<http://wptrafficanalyzer.in/blog/driving-route-from-my-location-to-destination-in-google-maps-android-api-v2/>

Mariné Jové, Pere; Rodríguez, José Ramón (2010). Gestió de projectes. Barcelona: FUOC.

Monjo Palau, Tona (2011). Disseny d'interfícies multimèdia. Barcelona: FUOC.

Rosenfeld, L.; Morville, P. (2002). Arquitectura de la informació per al World Wide Web, Febrer 2010 UOC

Tamada, Ravi (2 de 5 de 2012). How to connect Android with PHP, MySQL.

Obtingut de <http://www.androidhive.info/2012/05/how-to-connect-android-with-php-mysql/>

Wagner, Alex (12 de 1 de 2014). Google posts fresh Android distribution stats, Jelly Bean and KitKat usage on the rise.

Obtingut de

<http://androidandme.com/2014/01/news/google-posts-fresh-android-distribution-stats-jelly-bean-and-kitkat-usage-on-the-rise/>

Sambells, Jeffrey (27 de 5 de 2010). Decoding Polylines from Google Maps Direction API with Java.

Obtingut de

<http://jeffreysambells.com/2010/05/27/decoding-polylines-from-google-maps-direction-api-with-java>

Android Design

Obtingut de <http://developer.android.com/design/index.html>

Android Developer Tools

Obtingut de <http://developer.android.com/tools/index.html>

Aparkarma

Obtingut de <http://www.aparkarma.com/>

Apparcar

Obtingut de <https://play.google.com/store/apps/details?id=com.YRH.ParkingApp>

Fake GPS Location

Obtingut de <https://play.google.com/store/apps/details?id=com.lexa.fakegps>

Find My Car

Obtingut de

https://play.google.com/store/apps/details?id=com.elibera.android.findmycar&feature=search_result

Google Code

Obtingut de <https://code.google.com/>

Google Directions API

Obtingut de <https://developers.google.com/maps/documentation/directions/>

Google Geocoding API

Obtingut de <https://developers.google.com/maps/documentation/geocoding/>

Google Maps Android API

Obtingut de <http://developer.android.com/google/play-services/maps.html>

MyCar Locator Free

Obtingut de <https://play.google.com/store/apps/details?id=com.nomadrobot.mycarlocatorfree>

MySQL

Obtingut de <http://www.mysql.com>

PHP

Obtingut de <http://www.php.net>

Waze

Obtingut de <https://play.google.com/store/apps/details?id=com.waze>

¿Dónde está mi coche?

Obtingut de

<https://play.google.com/store/apps/details?id=com.latarce.location.dondeestamicoche>