Universitat Autònoma de Barcelona

escola d'enginyeria

(2876-1: OnTheBus)

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per
Mònica Esteve Romeu
i dirigit per
Jordi Roig de Zárate
Bellaterra, 12 de Setembre de 2011

Universitat Autònoma de Barcelona

etse

Escola Tècnica Superior d'Enginyeria

El sotasignat, Jordi Roig de Zárate

Professor/a de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en

I per tal que consti firma la present.

Signat: Jordi Roig de Zárate

Bellaterra, 12 de Setembre de 2011

Índex de continguts

1 Introducció

1.1 Motivació

Aquest projecte neix de les creixents necessitats de mobilitat actuals. Cada dia ens movem més i més lluny, per estudiar, treballar, comprar, visites al metge, oci, etc. Aquestes necessitats són també presents en els grans nuclis urbans, on pot ser necessari cobrir grans distàncies per poder arribar a un determinat destí. Moltes persones utilitzen el transport privat com a forma de desplaçament, però cada cop més gent opta per el transport públic.

Si be els mitjans de transport públic eren anys enrere considerats una forma de transport per persones amb pocs recursos econòmics, com estudiants o jubilats, cada dia més gent opta per aquest tipus de transport per diversos motius, que poden ser econòmics o ecològics d'entre d'altres. Els parcs automobilístics de les ciutats no paren de créixer, i es comencen a fer patents problemes derivats de d'aquest augment de vehicles, com, l'augment de la pol·lució ambiental, la dificultat per aparcar en nuclis molt concorreguts o la saturació d'algunes vies. És per això que les autoritats cada cop són més conscients de la necessitat de pacificar el trànsit creant zones peatonals, zones trenta i impulsar el transport públic com a forma preferent de desplaçament, creant noves línies de metro i bus a les ciutats.

Aquest increment en el número de línies d'autobús ha provocat que actualment sigui molt complex determinar quina és la línia d'autobús que s'ha d'agafar per arribar al nostre destí, determinar quins transbords podem fer o quina es la disponibilitat horària de cada línia. Per això, les companyies d'autobús que operen en cada zona, proporcionen tota aquesta informació en una pagina web, o en marquesines especialment ubicades a les parades. Les pagines web però, són de difícil consulta des del carrer, especialment si opera més d'una companyia a la zona, i la informació no es troba unificada en una única pàgina. Les marquesines en canvi, tenen més aviat una funció de validació, ja que podem determinar si a la parada on ens trobem hi circula la línia que volem agafar, encara que aquestes marquesines poden ser vandalitzades. El problema d'accedir a aquesta informació, es pot agreujar més encara en persones que no estan familiaritzades amb la zona o persones discapacitades, així com persones d'edat avançada.

Per sort, en el món actual, la presencia de terminals mòbils amb accés a Internet és tota una realitat. Aquestes plataformes, a més de disposar de connexió a Internet, incorporen tota mena de sensors, com brúixoles, giroscopis, acceleròmetres i GPS, una molt bona combinació per poder desenvolupar aplicacions que solucionin aquestes necessitats creixents.

1.2 Què volem fer, on volem arribar

El que és vol aconseguir, es unificar tota la informació referent a les línies d'autobús de diverses ciutats en una única aplicació, amb una interfície universal de fàcil accés i ús, per tal de poder guiar al usuari fins a la parada d'autobús, donar informació dels temps d'espera, i informar-lo durant el trajecte amb l'autobús, indicant a l'usuari quan ha de baixar de l'autobús per arribar a un determinat destí.

Es vol aconseguir una aplicació intuïtiva, fàcil d'utilitzar, tant per aquelles persones acostumades a fer servir dispositius mòbils, com per aquelles persones que hi tenen algunes dificultats, com les persones grans.

Degut a que els terminals mòbils actuals no disposen de software adaptat per a persones amb discapacitats visuals, volem utilitzar el disseny universal per fer el més accessible possible la nostre aplicació, perquè incloses aquestes persones puguin fer-la servir de forma fàcil, senzilla i pràctica.

No es vol que la aplicació es quedi al nivell d'una prova pilot en una ciutat concreta, volem estendre la possibilitat d'utilitzar l'aplicació en diverses ciutats del món, per presentar una solució realista al problema del guiatge en autobús dins les ciutats.

2 Estudi de viabilitat

2.1 Estat de l'art

En aquesta secció es comentarà l'estat de l'art de les aplicacions de guiatge en transport públic accessible.

En una primera part s'explicarà l'estat actual d'aquest tipus d'aplicacions, per a passar després a detallar l'estat de cadascuna de les vessants..

2.1.1 ESTAT ACTUAL DEL MERCAT

En els últims anys, els sistemes de guiatge personal, han cobrat especial importància, es poden trobar nombrosos desenvolupaments de sistemes de guiatge, tant gratuïts com de pagament, així com exemples de implementacions senzilles de sistemes de seguiment de rutes de codi obert.

No obstant això, aquests sistemes solen estar orientats al desplaçament autònom (sense considerar l'ús de sistemes de transport públic), i no s'han trobat referències a sistemes de guiatge per a persones discapacitades.

Amb l'arribada de dispositius de cost relativament reduït, amb la tecnologia necessària per a implementar aquests sistemes de guiatge, és d'esperar que el nombre d'aplicacions d'aquest àmbit augmentin amb rapidesa, sobre tot tenint el compte els costos reduïts, per al seu desenvolupament, així com l'ampli ventall d'usuaris.

2.1.2 SISTEMES DE GUIATGE

Els sistemes de guiatge, que estan àmpliament difosos en les plataformes, es poden prendre com a referència gran nombre d'aplicacions com GoogleMaps, TomTom o AndNav.

Aquestes aplicacions es basen en la utilització de sistemes de cartografia externs al dispositiu que es descarreguen a través de serveis online utilitzant la connexió a Internet del dispositiu o bé incorporen mapes juntament amb l'aplicació, tant gratuïts com de pagament, així com el sistema civil de GPS (Geo Posicionament per Satèl·lit).

En aquest àmbit a l'hora de utilitzar tant els sistemes cartogràfics, com el sistema de GPS s'han trobat diversos problemes, alguns d'ells àmpliament coneguts i d'altres amb escassa literatura disponible, que han fet que s'hagin hagut de modificar diverses especificacions de

desenvolupament. Totes aquestes problemàtiques seran àmpliament detallades en els respectius punts.

2.1.3 SISTEMES D'AJUDA A L'ÚS DE TRANSPORT PÚBLIC

Els últims dos anys s'ha incrementat la demanda d'ajuda a l'hora de desplaçar-se per la ciutat en transport públic, degut a l'alt volum de tràfic produït pel transport privat i les despeses associades a ell, així com per la poca informació de qualitat proporcionada per les pròpies companyies de transport.

Aquesta mala gestió per part de les companyies de transport, està essent lentament suplida mitjançant aplicacions proporcionades per companyies de desenvolupament de software (UrbanStep), projectes universitaris (ValenciaBus), o aplicacions desenvolupades per la pròpia companyia de transports (TMB Virtual).

Paradoxalment, per tal implementar els sistemes d'ajuda a l'ús del transport públic ha sigut necessari fer mineria de dades amb la informació proporcionada per les companyies de transport públic, ja que la gran majoria de companyies no proporcionen la informació en un format fàcil de manipular informàticament. Per exemple, en alguns casos ha sigut necessari programar un software per tal d'extreure automàticament les dades de les línies i parades d'autobusos a partir de la informació mostrada en la pàgina web de la companya, o ha calgut fer una transformació dels sistemes de coordenades utilitzats per les companyies a un sistema únic per a que totes les dades funcionin correctament a OnTheBus.

2.1.4 SISTEMES D'AJUDA ALS USUARIS AMB PROBLEMES D'ACCESSIBILITAT

Els sistemes d'ajuda al discapacitat en dispositius basats en *Android* es troben en fases molt verdes de la seva evolució. Tot i que vàries funcionalitats que són necessàries per a implementar un sistema d'aquest tipus es troben disponibles mitjançant la pròpia arquitectura proporcionada pel sistema, aquestes manquen de la estabilitat i maduresa necessàries per a fer aplicacions comercials per a col·lectius de discapacitats.

D'altre banda les limitacions no sempre venen donades per a l'arquitectura, sinó que pot ser que sigui el propi dispositiu el que dificulti l'accessibilitat, degut a la utilització de components més assequibles i de baixa qualitat per a abaratir el preu final del dispositius.

2.2 Tecnologia a emprar

En aquest apartat es defineixen les tecnologies requerides per al desenvolupament d'aquest projecte.

2.2.1 El sistema operatiu Android

Android és un sistema operatiu per a dispositius mòbils en general, com poden ser telèfons, tablets, reproductors MP3, etc, encara que recentment s'està expandint el seu ús per a tot tipus de dispositius, com poden ser televisors o fins i tot microones o rentadores.

Inicialment, la companya *Android Inc.* va començar el seu desenvolupament, i posteriorment va ser comprada per Google a l'any 2005. La seva llicència és *Apache Software Licence,* de codi obert.

Google, a l'any 2007, va fundar la *Open Handset Alliance (OPH)*, que es tracta d'una aliança comercial formada per 78 companyies que desenvolupen estàndards oberts per dispositius mòbils. En formen part, entre d'altres, *HTC*, *Motorola, NVidia* i *Samsung*.

Al novembre de 2007, es va publicar la primera versió de la API d'*Android*, que és una interfície de programació d'aplicacions que conté el conjunt de procediments en forma de llibreria de software que permet utilitzar els components dels terminals *Android* (sensors, pantalla, SO, etc) per desenvolupar aplicacions.

Existeixen altres sistemes operatius per a dispositius mòbils, com per exemple IOS de l'empresa *Apple*, *BlackBerry OS* de *RIM (Research in Motion)*, *Symbian* de *Nokia* o *Windows Phone* de *Microsoft*. Un dels motius de que s'hagi escollit el sistema operatiu *Android* és que ofereix més possibilitats, facilitat de programació i garanties d'èxit que qualsevol dels altres esmentats.

Per exemple, en el cas de *IOS* el *kit* de desenvolupament només està disponible per a ordinadors amb sistema operatiu *MAC OS*, també fabricat per *Apple*, i no tots els integrants del grup de OnTheBus en posseeixen un. *Symbian* era el sistema operatiu que més quota de mercat posseïa en el moment de la decisió, però en els últims anys havia anat perdent usuaris i quedant-se antiquat: fins i tot la companya que el desenvolupa, *Nokia*, havia començat a desenvolupar d'altres sistemes operatius per substituir-lo. *Windows Phone* era un sistema operatiu massa nou i no existia molta literatura per poder aprendre'n ràpidament el llenguatge i sistema de programació.

Però el que va acabar decidint que es faria servir *Android* és el ràpid creixement i acceptació que estava tenint, que feia presagiar que en pocs mesos seria el sistema operatiu més utilitzat. A més a

més, al ser un sistema operatiu ofert a terminals de diferents marques, existeix una àmplia varietat de dispositius de diferents preus, pel que es va pensar que seria molt més fàcil arribar a tots els tipus d'usuaris que es pretenia: d'elevada edat, estudiants, persones amb problemes d'accessibilitat o no.

2.2.1.1 Versions de la API d'Android

Actualment existeixen 13 versions de la API d'*Android*. Cada vegada que apareix una nova versió, incorpora noves funcionalitats i s'actualitza constantment. A la següent *Taula 1* es mostren les diferents versions i el seus noms.

Platform Version	API Level	VERSION_CODE
Android 3.2	13	HONEYCOMB_MR2
Android 3.1.x	12	HONEYCOMB_MR1
Android 3.0.x	11	HONEYCOMB
Android 2.3.4 Android 2.3.3	10	GINGERBREAD_MR1
Android 2.3.2 Android 2.3.1 Android 2.3	9	GINGERBREAD
Android 2.2.x	8	FROYO
Android 2.1.x	7	ECLAIR_MR1
Android 2.0.1	6	ECLAIR_0_1
Android 2.0	5	ECLAIR
Android 1.6	4	DONUT
Android 1.5	3	CUPCAKE
Android 1.1	2	BASE_1_1
Android 1.0	1	BASE

Taula 1 - Versions de la API d'Android

2.3.1.2 *Justificació de la versió de la API utilitzada*

Durant les primeres reunions de grup, es va veure que els seus membres ja havien començat a fer projectes de prova per anar aprenent a programar per *Android*, però no tots havien fet servir la mateixa versió. Es va investigar i es va veure que en principi no hi havia cap problema amb qualsevol de les versions disponibles en aquell moment, ja que, malgrat que a la API s'indicava que algunes funcions no estaven disponibles per les primeres versions, en principi cap d'elles semblava que anés a suposar un inconvenient per al desenvolupament del projecte. Però davant de tantes versions diferents, calia posar-se d'acord en utilitzar la mateixa versió tots els components del grup, per evitar problemes que poguessin sorgir més endavant.

Es van mirar les estadístiques de les versions que incorporaven els terminals a la pàgina web de la API, on s'observà que gairebé el 80% dels terminals tenien una versió igual o superior a *Android* 2.1. A més a més, es va veure que molts fabricants havien anunciat actualitzacions per els terminals que tenien al mercat amb les versions 1.5 i 1.6.

Aleshores, es va prendre la decisió de que la versió mínima de la API que suportaria *OnTheBus* seria *Android* 2.1 en un inici.

A la següent *Figura 1* i *Taula 2*, es poden veure les estadístiques actuals de les versions utilitzades.

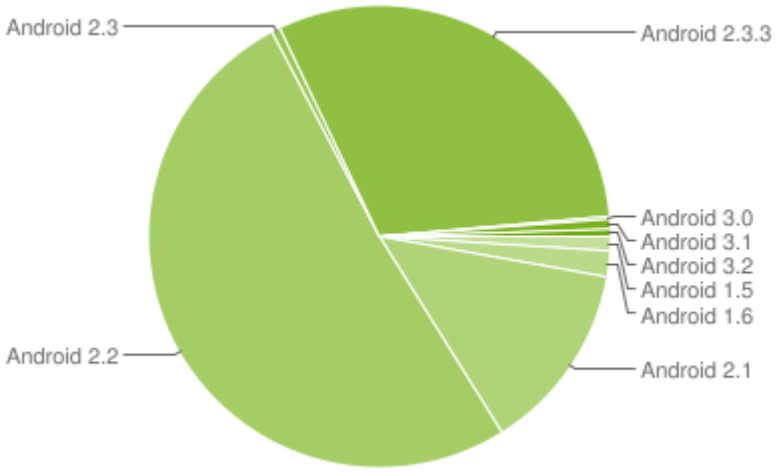


Figura 1- Gràfic circular de les versions d'Android que incorporen els terminals a Setembre de 2011.

Platform	Codename	API Level	Distribution
Android 1.5	Cupcake	3	1.0%
Android 1.6	Donut	4	1.8%
Android 2.1	Eclair	7	13.3%
Android 2.2	Froyo	8	51.2%
Android 2.3 - Android 2.3.2	Gingerbread	9	0.6%
Android 2.3.3 - Android 2.3.4		10	30.7%
Android 3.0	Honeycomb	11	0.2%
Android 3.1		12	0.7%
Android 3.2		13	0.5%

Taula 2 - Resum dels percentatges de terminals de cada versió d'Android a Setembre de 2011

Sumant els percentatges de les versions majors a 2.1, es pot observar que són el 97.2% dels terminals.

Per tant, la decisió inicial va ser encertada, ja que molts terminals s'han anant actualitzant amb el temps i els nous terminals que han anat sortint al mercat ja incorporen les últimes versions. A la següent *Figura 2* es pot veure aquesta tendència, on destaca l'estancament de les versions inferiors a *Android* 2.1 i també el ràpid creixement de la versió 2.3 *Gingerbread*, que al inici del desenvolupament d'aquest projecte, encara no estava disponible.

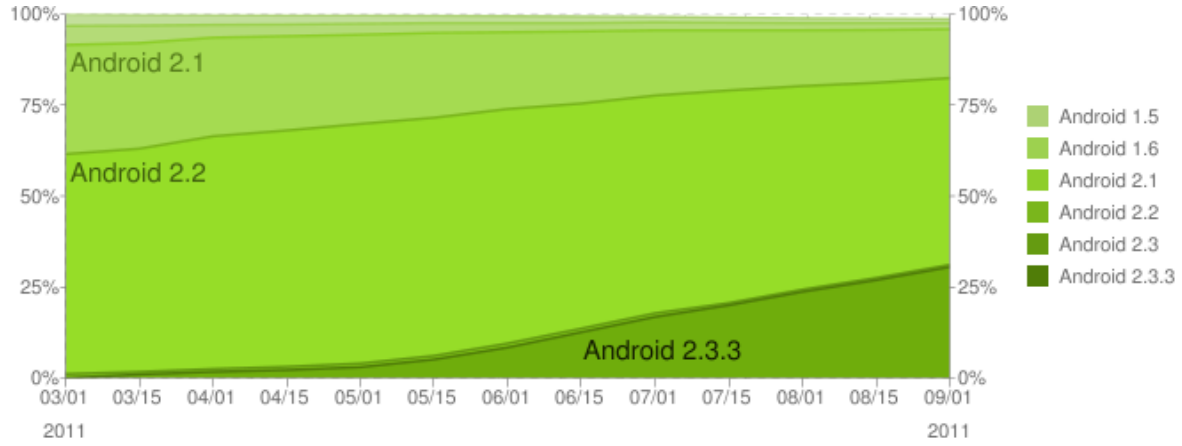


Figura 2 - Evolució de les versions dels terminals entre els mesos de Març i Setembre de 2011

2.2.2 SERVEIS DE CLOUD COMPUTING

Cloud Computing (computació en el núvol o núvol de còmput) és una tecnologia informàtica que presta un servei de computació en servidors remots o centres de dades emmagatzemades a Internet.

L'usuari del servei no ha de tenir coneixements en la matèria del càlcul ni ha de crear i mantenir la infraestructura que requereix: l'únic requisit per utilitzar-lo és tenir connexió a Internet.

En aquest projecte, hi ha tres càlculs que no és possible fer-los per diversos motius.

- Calcular les rutes caminant a partir de dos punts, amb indicacions pas a pas.
- Aconseguir el temps d'espera del bus en temps real (quan l'usuari arriba a la parada) de totes les línies i parades d'autobús de sis ciutats diferents.
- Saber l'adreça completa (país, ciutat, carrer i número) a partir de dues coordenades geogràfiques, i també el pas invers.

Aleshores s'ha decidit que es faran servir serveis de computació en el núvol en aquests tres casos. Per tant, és necessari utilitzar un servei de *Cloud Computing* que calculi la ruta caminant entre dos punts, indicant el sentit de cada gir i la orientació (Nord, Sud, etc) de cada part de la ruta. També és necessari un altre servei que faci les funcions de geocodificació (procés que permet obtenir l'adreça completa a partir d'un parell de coordenades geogràfiques) i geocodificació inversa (obtenir les coordenades exactes a partir del nom del carrer, número i ciutat). Per últim, és necessari consultar a les companyies d'autobusos de les diferents ciutats el temps d'espera en temps real, sempre i quan la companyia ho permeti.

2.2.3 SUBVERSION

Subversion és un sistema de control de versions amb llicència lliure *GNU/GPL* que automatitza la tasca de guardar, recuperar i barrejar versions de fitxers, utilitzant el protocol *SVN*.

Permet que diferents usuaris utilitzin el mateix repositori on s'hi allotja el codi font del programa, amb la possibilitat d'actualitzar les modificacions de qualsevol usuari o restaurar l'aplicació a un estat anterior en cas de que sigui necessari. A més a més, permet crear línies de desenvolupament paral·leles amb noves funcionalitats i més endavant, quan es tingui la seguretat de que funcionen correctament, incorporar-les a la versió principal i estable de l'aplicació.

Hi ha d'altres sistemes de control de versions, com *Mercurial* o *Git*, aquest últim molt utilitzat en l'àmbit del software lliure i en el projecte *AOSP* (*Android Open Source Project*), on col·laboren diferents fabricants de dispositius i processadors per millorar el sistema operatiu *Android*. Però es va decidir utilitzar *Subversion* perquè ja se sabia instal·lar i configurar, i l' utilització d'un altre sistema hagués suposat invertir més temps en aprendre a instal·lar-lo i utilitzar-lo, temps que va ser preferible invertir en aprendre a programar en *Android*.

2.2.4 ENTORN DE DESENVOLUPAMENT

En aquest apartat es descriu el software necessari per poder desenvolupar OnTheBus

Ecllipse IDE i Java Standard Development Kit (SDK): *Eclipse* és un entorn de desenvolupament integrat que ens permet programar en diferents llenguatges de programació, entre ells el llenguatge *Java*, compilar el nostre codi i executar-lo. És necessària també la instal·lació del SDK de *Java*, que és un kit de desenvolupament pel llenguatge *Java*. Hi ha altres entorns per programar en Java perfectament vàlids per aquest projecte, com per exemple *NetBeans*, però *Eclipse* és l'*IDE* recomanat per *Google* i per al que existeixen les últimes versions dels *plugins* que es descriuran seguidament, i és l'*IDE* que els integrants del grup estan acostumats a fer servir i dominen.

Android SDK: instal·lant aquesta eina podrem connectar el nostre terminal Android, ja que ens proporciona els drivers del dispositiu per al nostre sistema operatiu (està disponible per Windows, Linux i MacOS). Si no tenim un dispositiu *Android*, podem simular el sistema operatiu mitjançant un emulador.

ADT Eclipse plugin: com el seu nom indica, és un afegit d'*Eclipse*. És molt complet, ens permet descarregar la versió d'*Android* que desitgem, crear emuladors amb les característiques que necessitem simular (acceleròmetre, *GPS* i altres tipus de sensors, resolució de pantalla, emmagatzemament extern, memòria *RAM*, etc), veure el *Log* d'informació i errors del terminal o emulador que estem executant, dissenyar la nostra interfície gràfica amb un editor gràfic, simular el nostre geoposicionament i fins i tot un recorregut introduint les coordenades geogràfiques de cada punt en un fitxer, entre d'altres coses interessants pel projecte. S'ha fet servir aquest perquè és l'únic que existeix que permet emular el sistema i, per tant, poder provar OnTheBus sense necessitat de tenir un dispositiu físic.

Subclipse: Es tracta d'un *plug-in* per *Eclipse* que permet connectar al repositori *SVN* de l'aplicació i efectuar les operacions modificació de codi i actualització del codi dels altres usuaris dins el mateix entorn de desenvolupament integrat *Eclipse*.

2.3 Planificació del projecte

El projecte s'ha dividit en diferents etapes de dissenys i implementació definint fites a la finalització de cadascuna de les etapes i dependència directe entre la finalització i inici d'algunes d'elles.

1. Anàlisi de requeriments: En aquesta etapa es posa de manifest què es vol desenvolupar i es defineix a alt nivell que es farà.

2. Disseny i implementació objectes comuns: En aquesta etapa es dissenya i implementa els objectes de cada mòdul que seran visibles per la resta.

3. Disseny i implementació dels mòduls: Durant aquesta etapa es desenvolupen les funcionalitats que ha d'oferir cada un dels mòduls.

4. Proves unitàries: Durant aquesta etapa es realitzen proves a nivell modular tant en emulació com en terminal físic.

4. Disseny i implementació de comunicació entre mòduls: Durant aquesta etapa es conjunten tots els mòduls definint la informació a traspassar entre els mateixos i com es traspassa.

5. Proves de l'aplicació: Durant aquesta etapa es realitzen diferents proves de l'aplicació tant en emulació com en terminal físic.

6. Proves de camp (UAT): Durant aquesta prova es realitzen proves de camp fent ús de l'aplicació en escenaris reals. També es realitzen unes proves d'acceptació d'usuari que es duen a terme amb la col·laboració del director del projecte.

La durada de cada una de les etapes així com la seva planificació es pot veure en la següent taula (*Figura 3*):

		Nombre	Duración	Inicio	Terminado
1		Gestió del projecte	1.062 days	22/10/10 16:00	28/07/11 17:00
2		Dinamització	1.062 days	22/10/10 16:00	28/07/11 17:00
9		Gestió general	164 days	10/12/10 8:00	27/07/11 17:30
10		Documentació general	842,667 days	1/11/10 17:00	9/06/11 17:30
16		Definició document de requerimients	277,333 days	25/10/10 8:00	4/01/11 17:30
24		Desenvolupament	442,667 days?	5/01/11 8:00	29/04/11 17:30
25		Mòdul algoritme de cerca	256,633 days	5/01/11 16:03	14/03/11 17:00
26		Disseny / Eleccio de l'algorisme	85,967 days	5/01/11 16:03	27/01/11 17:30
30		Implementació	170,667 days	28/01/11 8:00	14/03/11 17:00
33		Test de l'algorisme	117,633 days	5/01/11 16:03	4/02/11 16:30
43		Mòdul core	111,833 days	15/03/11 8:00	12/04/11 16:45
44		Disseny	26,167 days	15/03/11 8:00	21/03/11 16:15
52		Implementació	53,667 days	21/03/11 16:15	4/04/11 16:45
56		Test del Controlador	32 days	4/04/11 16:45	12/04/11 16:45
60		Mòdul BDD	187,222 days	5/01/11 16:03	23/02/11 16:53
61		Buscar informació del servei d'autobusos	17,037 days	5/01/11 16:03	28/01/11 16:06
62		Aprovisionament dades	18,519 days	28/01/11 16:06	23/02/11 16:53
63		Estudi i Disseny E/R Base dades "Autobús"	2,222 days	5/01/11 16:03	7/01/11 16:23
64		Normalitzar E/R e implementar Base dades "Autobús".	4,444 days	7/01/11 16:23	13/01/11 17:03
65		Omplir Base de Dades "Autobús" amb les dades del ser...	2,667 days	14/01/11 8:00	18/01/11 17:00
66		Implementar (dintre de l'aplicació) la Base de Dades (SQ..	1,481 days	19/01/11 8:00	20/01/11 16:43
67		Generar els mètodes per les diferents connexions, acce..	8,148 days	5/01/11 16:03	17/01/11 16:16
68		Parsejar la informació dels temps que triguen els difere..	11,111 days	5/01/11 16:03	20/01/11 16:13
69		Modul Intericie d'usuari	442,667 days?	5/01/11 8:00	29/04/11 17:30
70		Esbossar un disseny	213,333 days?	5/01/11 8:00	1/03/11 17:00
75		Prototipatge	149,333 days	2/03/11 8:00	8/04/11 17:00
80		Mòdul de só	43,3 days	5/01/11 16:03	17/01/11 17:03
90		Mòdul de texte	21,967 days	5/01/11 16:03	11/01/11 17:03
93		Mòdul Gestures	53,333 days	5/01/11 16:03	19/01/11 16:03
97		Mòdul de Vibració	21,333 days	5/01/11 16:03	11/01/11 16:03
100		Interficie final	80 days	11/04/11 8:00	29/04/11 17:30
105		Mapa	165,967 days	5/01/11 16:03	17/02/11 17:00
106		Estudi de la API de Google Maps	32,633 days	5/01/11 16:03	13/01/11 17:03
107		Control i visualització del Mapa	32 days	14/01/11 8:00	21/01/11 17:00
111		Possicionament en Pantalla	31,667 days	24/01/11 8:00	31/01/11 16:30
114		Rutes al Mapa	69,667 days	31/01/11 16:30	17/02/11 17:00
117		Mòdul Geoposicionament	272 days?	18/02/11 8:00	29/04/11 17:00
118		Geoposicionament GPS	208 days	18/02/11 8:00	13/04/11 17:30
119		Obtenció posició actual	37 days	18/02/11 8:00	28/02/11 16:30
122		Obtenció geoposició d'una adreça	37,667 days	28/02/11 16:30	9/03/11 17:00
127		Guiat rutes a peu - Backend	74,667 days	10/03/11 8:00	29/03/11 17:00
132		Guiat rutes en autobus - Backend	58,667 days	30/03/11 8:00	13/04/11 17:30
136		Obtenció rutes caminades	64 days?	13/04/11 17:00	29/04/11 17:00

Figura 3- Planificació temporal del projecte

		Nombre	Duración	Inicio	Terminado
1		Gestió del projecte	1.062 days	22/10/10 16:00	28/07/11 17:00
2		Dinamització	1.062 days	22/10/10 16:00	28/07/11 17:00
9		Gestió general	164 days	10/12/10 8:00	27/07/11 17:30
10		Documentació general	842,667 days	1/11/10 17:00	9/06/11 17:30
16		Definició document de requerimients	277,333 days	25/10/10 8:00	4/01/11 17:30
24		Desenvolupament	442,667 days?	5/01/11 8:00	29/04/11 17:30
25		Mòdul algoritme de cerca	256,633 days	5/01/11 16:03	14/03/11 17:00
26		Disseny / Eleccio de l'algorisme	85,967 days	5/01/11 16:03	27/01/11 17:30
30		Implementació	170,667 days	28/01/11 8:00	14/03/11 17:00
33		Test de l'algorisme	117,633 days	5/01/11 16:03	4/02/11 16:30
43		Mòdul core	111,833 days	15/03/11 8:00	12/04/11 16:45
44		Disseny	26,167 days	15/03/11 8:00	21/03/11 16:15
52		Implementació	53,667 days	21/03/11 16:15	4/04/11 16:45
56		Test del Controlador	32 days	4/04/11 16:45	12/04/11 16:45
60		Mòdul BDD	187,222 days	5/01/11 16:03	23/02/11 16:53
61		Buscar informació del servei d'autobusos	17,037 days	5/01/11 16:03	28/01/11 16:06
62		Aprovisionament dades	18,519 days	28/01/11 16:06	23/02/11 16:53
63		Estudi i Disseny E/R Base dades "Autobús"	2,222 days	5/01/11 16:03	7/01/11 16:23
64		Normalitzar E/R e implementar Base dades "Autobús".	4,444 days	7/01/11 16:23	13/01/11 17:03
65		Omplir Base de Dades "Autobús" amb les dades del ser...	2,667 days	14/01/11 8:00	18/01/11 17:00
66		Implementar (dintre de l'aplicació) la Base de Dades (SQ..	1,481 days	19/01/11 8:00	20/01/11 16:43
67		Generar els mètodes per les diferents connexions, acce..	8,148 days	5/01/11 16:03	17/01/11 16:16
68		Parsejar la informació dels temps que triguen els difere..	11,111 days	5/01/11 16:03	20/01/11 16:13
69		Modul Intericie d'usuari	442,667 days?	5/01/11 8:00	29/04/11 17:30
70		Esbossar un disseny	213,333 days?	5/01/11 8:00	1/03/11 17:00
75		Prototipatge	149,333 days	2/03/11 8:00	8/04/11 17:00
80		Mòdul de só	43,3 days	5/01/11 16:03	17/01/11 17:03
90		Mòdul de texte	21,967 days	5/01/11 16:03	11/01/11 17:03
93		Mòdul Gestures	53,333 days	5/01/11 16:03	19/01/11 16:03
97		Mòdul de Vibració	21,333 days	5/01/11 16:03	11/01/11 16:03
100		Interficie final	80 days	11/04/11 8:00	29/04/11 17:30
105		Mapa	165,967 days	5/01/11 16:03	17/02/11 17:00
106		Estudi de la API de Google Maps	32,633 days	5/01/11 16:03	13/01/11 17:03
107		Control i visualització del Mapa	32 days	14/01/11 8:00	21/01/11 17:00
111		Possicionament en Pantalla	31,667 days	24/01/11 8:00	31/01/11 16:30
114		Rutes al Mapa	69,667 days	31/01/11 16:30	17/02/11 17:00
117		Mòdul Geoposicionament	272 days?	18/02/11 8:00	29/04/11 17:00
118		Geoposicionament GPS	208 days	18/02/11 8:00	13/04/11 17:30
119		Obtenció posició actual	37 days	18/02/11 8:00	28/02/11 16:30
122		Obtenció geoposició d'una adreça	37,667 days	28/02/11 16:30	9/03/11 17:00
127		Guiat rutes a peu - Backend	74,667 days	10/03/11 8:00	29/03/11 17:00
132		Guiat rutes en autobus - Backend	58,667 days	30/03/11 8:00	13/04/11 17:30
136		Obtenció rutes caminades	64 days?	13/04/11 17:00	29/04/11 17:00

Figura 4- Planificació temporal del projecte

Com es pot observar, la data prevista de finalització es el 28 de juliol de 2011, superant el primer termini d'entrega del projecte, tenint com a marge per a la resolució de problemes el mes d'agost.

S'ha de tenir en compte que s'ha definit que la dedicació de cada un dels recursos es de unes 4 hores dia.

2.3.1 DIAGRAMA DE GANTT

A continuació podem veure un diagrama de Gantt (*Figura 5*) amb el calendari del projecte així com les dependències entre les tasques globals.

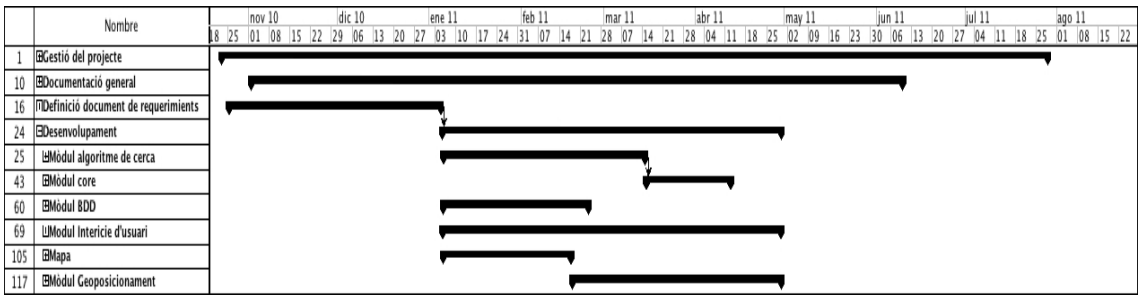


Figura 5 - Diagrama de gantt

2.4 El disseny universal

> ***"The opposite of useful, the opposite of something that is well designed is… useless" – by @JHockenberry***

> ***"No es pot pensar en la discapacitat de forma general, perquè cadascuna té les seves necessitats específiques. A cada problema, una solució" – Per @Carlos Martínez, vicepresident d'ASOCIDECAT, l'Associació de Sord-Cecs de Catalunya***

La idea de disseny universal, va començar a prendre forma als voltants dels anys 90 com a evolució lògica del disseny sense barreres, accessible i de la tecnologia assistida de recolzament, amb la intenció d'aprofitar-se en un rang ampli de disciplines.

El Disseny Universal es refereix a un conjunt de principis acordats per un grup d'arquitectes, dissenyadors de productes, enginyers i investigadors del disseny ambiental que serveixen com a guia per l'obtenció de productes i entorns que siguin el més flexibles, equitatius, intuïtius possibles entre d'altres característiques, per poder ser utilitzats pel nombre màxim possible de persones, sense necessitats d'adaptacions o de disseny especialitzat. No s'ha d'oblidar que hi ha diversos problemes

d’accessibilitat per a un percentatge d’usuaris, els quals normalment es deixen de banda a l’hora de dissenyar *software*.

En essència són 7 principis que a més de servir per futurs desenvolupaments, també tenen funció avaluadora per dissenys actuals.

Al ser d'àmbit general, és possible que segons el tipus de disseny, algun d’aquests principis es pugui obviar o no ser del tot rellevant en el context de l’aplicació.

Els 7 principis, dels quals s’ha basat el disseny de l’aplicació, són:

1. **Ús equitatiu:** El disseny ha de ser fàcil d'usar i adequat per a totes les persones independentment de les seves capacitats i habilitats. Així doncs, cal evitar segregar a qualsevol usuari i proporcionar les mateixes formes d’ús per a tots. Idèntiques quan sigui possible i equivalents quan no hi hagi cap altre alternativa. Per tant, ha de ser utilitzable i a un preu raonable independent de les problemes d’accessibilitat dels usuaris.
2. **Ús Flexible:** El disseny ha de poder adequar-se a un ampli rang de preferències i habilitats individuals. Ha d’oferir opcions per tal que la interfícies es pugui utilitzar de diverses formes, tan si l’usuari es esquerrà com dretà i que sigui el més adaptable possible al ritme de l’usuari.
3. **Ús Simple o intuïtiu**: El disseny ha de ser fàcil d'entendre independentment de l'experiència, els coneixements, les habilitats o el nivell de concentració de l'usuari. Cal dissenyar una interfície sense complexitat innecessària i que sigui molt intuïtiva i guiable per a l’usuari, independentment del seu grau de coneixement amb les tecnologies i la seva edat. Es pot proporcionar d’informació complementaria per tal de que sigui més guia, ja sigui ajudes, o avisos retroalimentats durant la tasca.
4. **Informació fàcil de percebre:** El disseny ha de ser capaç d'intercanviar informació amb l'usuari, independentment de les condicions ambientals o les capacitats sensorials del mateix.
Per tal de que sigui fàcil de percebre, caldria utilitzar diferents mitjans, ja siguin sonors, vibracions, visuals i lumínics, per a que la presentació sigui més completa. La informació ha de ser llegible i els elements s’han de diferenciar entre si per a que es puguin percebre per si sols.
5. **Tolerància a l’error:** El disseny ha de minimitzar les acciones accidentals o fortuïtes que poden tenir conseqüències fatals o no desitjades. Per a que hi hagi aquesta tolerància, s’ha fer un ampli testeig del disseny, cal tenir en compte totes les màximes accions que pot arribat a fer un usuari per tenir en compte que no comportin errors. Així doncs, cal advertir dels errors de forma

clara, que s'entenguin els errors i per a que s'han produït, fet que l'usuari al entendre-ho no repetiria certes accions per tal de no conduir al mateix error.

6. **Mínim esforç físic**: El disseny s'ha de poder utilitzar de forma còmoda, eficaç i amb el mínim esforç possible. Cal que es pugui utilitzar minimitzant certs aspectes, com ara les accions repetitives, amb el mínim d'esforç físic constant, i en una posició neutral. En cas de que calgui un esforç, que la força utilitzada de la operació sigui raonable.
7. **Dimensions apropiades d'ús i aproximació:** Les mides i espais han de ser apropiats per l'abast, manipulació i ús per part de l'usuari independentment de la seva mida, posició i mobilitat. Proporcionar una interfície, on la mida dels elements siguin proporcionals a la mida del terminal. A més a més, que sigui lo suficientment grans com per a que siguin llegibles i de fàcil accés independentment de la mida dels dits dels usuaris o de la seva mobilitat.

L'objectiu principal d'aquesta filosofia és aconseguir una societat on totes les persones puguin participar, independentment de la seva condició, incloses les persones amb discapacitats (ja siguin limitació funcional visual, auditiva, cognitiva o de mobilitat), gent gran, de diferents grups de poblacions, etc...

Una manera de millorar el disseny universal és treballar directament amb un grup d'usuaris amb problemes d'accessibilitat, els quals et poden explicar quins problemes troben en els dissenys d'aplicacions i en l'avaluació de prototips realitzats.

D'altre banda, també cal sensibilitzar als dissenyadors. Per exemple el propi dissenyador pot simular certs problemes d'accessibilitat per veure en primera persona, quins problemes es pot trobar. Una manera és simular les limitacions de mobilitat, limitant els moviments amb objectes, embenatges, barres de ferro per immobilitzar articulacions, etc...

D'aquesta forma canviaria la manera de veure les conseqüències que poden tenir un mal disseny per als grups d'usuaris amb problemes d'accessibilitat.

En el propi context particular, s'ha trobat dificultats en el fet de que actualment, la gran majoria de telèfons mòbils amb *S.O*. *Android* instal·lat pateixen fragmentació, és a dir, hi ha molta diversitat de models**,** amb dispositius diferents, i cada companyia ha decidit els dissenys del seus aparells, amb o sense teclat, amb o sense botons i ni tan sols s'han posat d'acord en el nombre estàndard de botons a utilitzar.

Aquests detalls han portat a haver de cercar informació de tots o gairebé tots els dispositius mòbils amb *Android* per tal de poder concretar uns criteris perquè el disseny de la interfície fós el més universal possible.

A l'actualitat, s'ha pogut establir un nombre mínim de botons, que són:

- Els botons de Home, per minimitzar la *App*.
- El botó de Menú, per obtenir més opcions.
- El de *Back*, per tornar a la pantalla anterior.

Un altre factor important, ha sigut el fet de realitzar una aplicació de geo-localització. Aquest fet ha condicionat totalment en el moment de prendre les decisions més importants, en base a la realització de la interfície de guiatge en temps real i la selecció de destinació. El disposar d'un mapa i de la capacitat de representar informació sobre ell, juntament amb altres factors estètics, ha portat a dissenyar dues interfícies paral·leles i equivalents (principi [1]) per tal de poder treure el màxim rendiment a cada perfil.

La primera d'elles està destinada a persones que no necessiten cap tipus d'accessibilitat especial. Basant-se en el principi [2], es dóna a l'usuari la opció d'activar/desactivar la forma en que rebrà la informació, ja sigui per veu o per text i la possibilitat de seleccionar el seu mode d'entrada de dades per defecte, encara que un cop a la pantalla corresponent es pot triar un altre.

Aquesta interfície utilitza *layouts* (fitxer que conté la distribució dels elements en una pantalla) amb botons virtuals, taules per sintetitzar la informació de les rutes, texts aclaridors i funcionalitats bàsiques amb els botons del dispositiu. D'aquesta manera s'aconsegueix acomplir el principi [3].

La segona interfície està més adaptada per tal de poder proporcionar la millor experiència a l'usuari que tingui una necessitat d'accessibilitat, solucionant el problema de la localització dels botons i de la informació, substituint-los per una barra lateral (per dretans o esquerrans, segons la configuració) que s'encarregarà de fer saber a l'usuari a on es troba en cada moment.

S'ha afegit ajudes vibratòries, visuals i sonores per informar a l'usuari sobre l'estat del guiatge i que no es trobi desorientat [1].

Per tal de poder intercanviar informació de manera eficient (principi [4]), els integrants del mòdul d'interfície gràfica han trobat formes apropiades per transmetre informació, com s'ha comentat abans i per poder rebre informació per part de l'usuari, s'han fet servir mètodes de reconeixement de veu, teclat virtual, *gestures*, historial, etc...

Els mètodes emprats en les dues interfícies, s'han hagut de testejar per poder minimitzar les seves fallades, fent l'aplicació més robusta, basant-nos en el principi [5]. També s'ha hagut de tenir en compte les mides dels botons i dels texts [7] perquè siguin entenedors i no cal dir que pel fet de ser una aplicació mòbil, no cal fer cap esforç físic [6].

Per poder arribar al nombre màxim d'usuaris, s'ha implementat l'aplicació de manera que sigui multillenguatge, aprofitant les facilitats que ens dóna el SDK *d'Android* i pel moment podem oferir l'aplicació en 3 idiomes: anglès, castellà i català. No obstant, està preparada per incorporar altres llenguatges en un futur.

La gran demanda de telèfons amb aquest S.O. ha portat a les companyies a dissenyar aparells per tal de poder arribar al nombre màxim d'usuaris i això ha portat a fer una classificació general de 3 tipus en quant als preus:

Gamma baixa: ZTE Blade, Huawei Ivy, Huawei Sonic, etc…

Gamma mitjana: Samsung Galaxy SCL, HTC Desire, etc…

Gamma alta: Samsung Galaxy S2, LG Optimus 2x, HTC Sensation, HTC Desire HD, etc…

Aquesta diferenciació de gammes, també ha suposat una diferenciació en els components dels aparells i això ha portat per exemple, a haver de realitzar una interfície preparada pel màxim nombre de resolucions i densitats de píxels (*Figura 6*). Per aquesta raó s'ha fet servir com a unitats els dp (*density pixels*) per les mides dels layouts, i sp per les mides de les tipografies.

	Low density (120), *ldpi*	Medium density (160), *mdpi*	High density (240), *hdpi*
Small screen	• QVGA (240x320), 2.6"-3.0" diagonal		
Normal screen	• WQVGA (240x400), 3.2"-3.5" diagonal • FWQVGA (240x432), 3.5"-3.8" diagonal	• HVGA (320x480), 3.0"-3.5" diagonal	• WVGA (480x800), 3.3"-4.0" diagonal • FWVGA (480x854), 3.5"-4.0" diagonal
Large screen		• WVGA (480x800), 4.8"-5.5" diagonal • FWVGA (480x854), 5.0"-5.8" diagonal	

Figura 6 - Resolucions de pantalla i densitat de píxels

2.5 Anàlisi de terminals

2.5.1 Botons físics en dispositius Android

Els terminals mòbils han evolucionat molt durant els últims anys. Els primers terminals tenien botons físics, els quals es podien localitzar amb facilitat sense mirar, cosa que feia que fossin fàcilment accessibles pels usuaris amb problemes d'accessibilitat visuals. Avui en dia la majoria d'aquests botons han passat a ser tàctils i per tant ja no els podem diferenciar com abans.

Per tal de veure aquesta clara tendència a la desaparició dels botons físics per botons tàctils, s'ha realitzat un petit estudi dels diversos terminals que hi ha per al sistema operatiu *Android*.

El primer terminal va aparèixer els 2008, l'anomenat *HTC Dream*. Aquest terminal compta amb un teclat *qwerty* físic amagat sota la pantalla, a més a més, té una bola de navegació (*trackball*) amb la tecla enter per seleccionar els elements de la pantalla situada a la part central inferior del terminal. Conté 5 botons físics, dos a cada costat del *trackball* i un a sobre d'ell. Aquests botons, ordenats d'esquerra a dreta són el trucada, *home*, menú, *back* i penjar trucada/encendre i apagar el terminal. A part d'aquests botons visibles en la cara de la pantalla, en els laterals s'hi troba les tecles de volum i el disparador de la càmera. Així doncs, es pot veure que inicialment els terminals anaven carregats de botons físics.

Com s'ha comentat en l'apartat anterior, el tema de la fragmentació s'ha tingut ben en compte en el procés de disseny de la interfície. S'han estudiat nombrosos models de telèfons mòbils i cercat documentació al respecte i s'ha pogut observar que actualment, només hi han 3 botons que predominen a tots els telèfons amb S.O. *Android*. Ni tan sols el botó de la càmera, o cercar han estat respectats en alguns models, dificultant les interfícies i les implementacions associades.

2.5.2 Diferenciació entre botons físics i tàctils

S'ha realitzat un anàlisi sobre dispositius de diferents companyies i s'ha seleccionat uns models que destaquen sobre la resta en lo referent al seu disseny a nivell de botons fent la classificació en botons físics, tàctils i digitals creats per *S.O*.

2.5.3 Disposició dels botons

En els dispositius analitzats tot i que porten botons semblants en els dispositius que porten botons semblants, s'ha pogut veure que la disposició d'ells és diferent, augmentant la dificultat en la implementació i en les decisions de disseny. Per exemple, fent servir la interfície 2, s'ha vist que en molts dispositius, la barra de desplaçament coincidia amb la posició del botó *Back*, fent inestable el

seu funcionament en cas d'un desplaçament erroni fora de la pantalla. Per aquesta raó es va decidir bloquejar el botó per *software*. De la mateixa manera, es va mirar de bloquejar la resta de botons que habitualment es situen sota la pantalla i es va trobar informació en la que es parlava de la impossibilitat de bloquejar el botó de *Home* en les versions actuals del S.O. per no posar a l'usuari en una situació en la que no pogués avortar una aplicació, encara que en possibles versions, potser no serà així. Al ser impossible el poder bloquejar totalment els botons, es va realitzar una doble vibració als extrems de la barra per tal de mantenir sempre orientat a l'usuari.

2.5.4 ANÀLISI

A continuació es pot veure una petita, però significativa, mostra de diferents tipus de dispositius mòbils amb *Android*. S'ha triat dispositius de les principals marques per ser lo més representatius possibles. Es posa especial èmfasi en la disposició dels botons físics i tàctils. En l'*Annex A* si pot trobar una taula amb una extensa mostra de dispositius mòbils, els quals han servit per a realitzar l'anàlisi complet.

2.6.2.1 Motorola

MILESTONE


Teclat	Analògic extern lliscant
Trackpad/Trackball	Trackpad analògic òptic
Botons tàctils	Back, Menú, Home, Cerca, Càmera
Botons físics	Volum
Extres	No

Taula 3 - Especificacions Motorla Milestone

2.6.2.2 HTC

MAGIC

Teclat	No
Trackpad/Trackball	Trackball
Botons tàctils	No
Botons físics	Back, Menú, Home, Cerca, Trucar, Penjar, Volum
Extres	No

Taula 4 - Especificacions HTM MAGIC

DESIRE Z

Teclat	Analògic extern lliscant
Trackpad/Trackball	Trackpad analògic òptic
Botons tàctils	Back, Menú, Home, Cerca
Botons físics	Volum, On/Off, Càmera
Extres	No

Taula 5 - Especificacions HTC DESIRE Z

CHACHACHA

Teclat	Analògic extern lliscant
Trackpad/Trackball	No
Botons tàctils	Menú, Home, Back, Cerca
Botons físics	Back, Facebook, volum
Extres	No

Taula 6 - Especificacions HTC CHACHACHA

2.6.2.3 SONY ERICSSON

XPERIA PLAY

Teclat	Analògic extern lliscant
Trackpad/Trackball	TouchPad analògic capacitiu
Botons tàctils	No
Botons físics	Back, Menú, Home, Cerca, Volum
Extres	Control Jocs

Taula 7 - Especificacions SONY XPERIA PLAY

2.6.2.4 SAMSUNG

GALAXY SII i9100

Teclat	No
Trackpad/Trackball	No
Botons tàctils	Back, Menú
Botons físics	On/Off, Volum, Home
Extres	No

Taula 8 - Especificacions SAMSUNG GALAXY SII

GALAXY SII amb teclat físic (variant SGH-I927)

Teclat	Analogic extern lliscant
Trackpad/Trackball	No
Botons tàctils	Back, Menú, Home, Cerca
Botons físics	On/Off, Volum
Extres	No

Taula 9 - Especificacions SMASUNG GALAXY SII (TF)

2.5.5 PROPERES TENDÈNCIES

S'ha cercat informació sobre futurs models de telèfons mòbils amb aquest *S.O*. i fins i tot es pot trobar algun futur model que no tindrà cap botó físic. Per part d' *Android*, també s'ha vist la mateixa tendència. En properes versions del Sistema Operatiu, la totalitat dels botons físics seran opcionals, deixant la decisió en mans dels fabricants. Possiblement aquesta decisió s'ha degut a l'objectiu de fusionar les versions 2.3 per *Smartphones* i *Honeycomb*, per *tablets*.

S'hauria d'estudiar si aquesta tendència continuarà en augment o no, per tal de decidir si caldria afegir com a futures millores la remodelació de la interfície.

Un altre punt important, conseqüència de la desaparició dels botons físics, és la inutilització de l'actual revisor de pantalla d'*Android*, el *Talkback*, el qual el seu funcionament és donar una descripció dels elements de la pantalla per els quals es van seleccionant. Aquesta acció es pot a dur a terme, si el terminal mòbil conté un *Trackball* o *Trackpad*. Com es pot veure, si la tendència dels terminals és que no continguin aquests elements, deixa inaccessible el terminal per part d'aquest grup d'usuaris i per tant, aquests hauran de buscar terminals on hi hagin el màxim nombre de botons físics.

2.6 Conclusions estudi de viabilitat

Un cop realitzat l'estudi de viabilitat del projecte es pot decidir la seva viabilitat. Si es decideix que no és viable s'haurà de buscar alternatives i realitzar un nou estudi des de l'inici d'altra banda si és viable es durà a terme.

En primer lloc el que s'ha realitzat és un estudi de les aplicacions de guiatge i d'ús dels transport públics que es poden trobar en el mercat i s'ha determinat que la aplicació cobreix certes mancances front a les aplicacions existents. Les grans aportacions de l'aplicació és l'accessibilitat que ofereix amb diferents tipus d'entrada de dades i un alt nivell de personalització de l'aplicació sense influir en la seva senzillesa.

D’altra banda s’ha realitzat un estudi dels costos que suposa el desenvolupament del projecte i es conclou que són assumibles donat que els costos materials son mínims i el cost més elevat és el cost per persona/hora en el desenvolupament de l’aplicació.

La planificació que s’ha realitzat dóna un calendari raonable encara que el marge per a possibles endarreriments és força ajustat.

Finalment i tenint en compte tots els estudis realitzat es decideix que el projecte és viable i es procedeix a la seva realització.

3 Mòdul de la Interfície d'Usuari

Tal i com s'ha indicat en l'apartat *Estudi de viabilitat*, el *projecte s'ha dividit, entre els recursos disponibles, en 4 subgrups. Així doncs, a partir d'aquest capítol es tractarà sobre el Mòdul de la Interfície d'Usuari, el qual està format pel subgrup 2 format per Eric Lara i Monica Esteve.*

Amb aquest capítol, es pretén fer una petita introducció al mòdul indicant quines són les eines amb les quals es pot treballar amb Android per a desenvolupar la interfície. També es vol informar de quins objectes es fan servir dins d'aquest mòdul, per tal que es tingui una visió global de l'entorn del mòdul.

En la resta de capítols, seguit d'aquest, es reflectirà la feina duta a terme per cada integrant del mòdul.

3.1 Introducció del Mòdul

Per poder elaborar amb èxit una aplicació mòbil d'aquestes característiques, cal donar importància al disseny i a la implementació de la interfície d'usuari. L'objectiu que s'ha marcat és realitzar un programa per a tothom, sent el més polivalent possible. Per aquesta raó, en primer lloc és indispensable detectar quins són els nostres potencials usuaris, per tenir en compte els seus trets diferenciadors. Com que l'aplicació és d'un caire molt obert en aquest sentit, es té un rang molt ampli d'edats d'usuaris, i això ens porta cap a la implementació d'una interfície basada en les directrius del disseny universal(en l'apartat *2.4 El disseny universal*).

3.2 Components de la GUI d'Android

En aquesta secció, es farà un breu repàs als components més utilitzats en l'aplicació i es definiran els seus atributs principals.

Per tal de poder realitzar correctament totes les funcions de l'aplicació mòbil, ha calgut dedicar bona part del temps a conèixer i dominar els diferents tipus d'elements que composen la interfície d'usuari de les aplicacions *Android*.

Com es veurà més endavant, la unitat bàsica d'una aplicació per aquesta plataforma és *l'Activity*. Es podria dir que és l'equivalent a una pantalla d'usuari en una aplicació convencional, a on es pot posar els diferents tipus de *widgets* (elements de la *UI* que componen la pantalla) per interactuar amb l'usuari.

Cada *Activity* té un *layout* (esquema que defineix la ordenació dels elements en pantalla) associat, que conté *ViewGroups* i *Views*. Hi ha dues maneres per definir aquestes interfícies, de forma estàtica, mitjançant arxius XML o dinàmica, mitjançant codi Java dins *l'Activity*.

Les *Views* utilitzen els recursos, com ara cadenes de text, colors, estils, que estan compilats en format binari. La classe *R.java* es genera automàticament per *Android* i és la referència a recursos i actua com a pont entre el codi que sol·licita el recurs i les referències binaries.

En el *Diagrama 1* es pot veure la relació entre aquests tres elements.

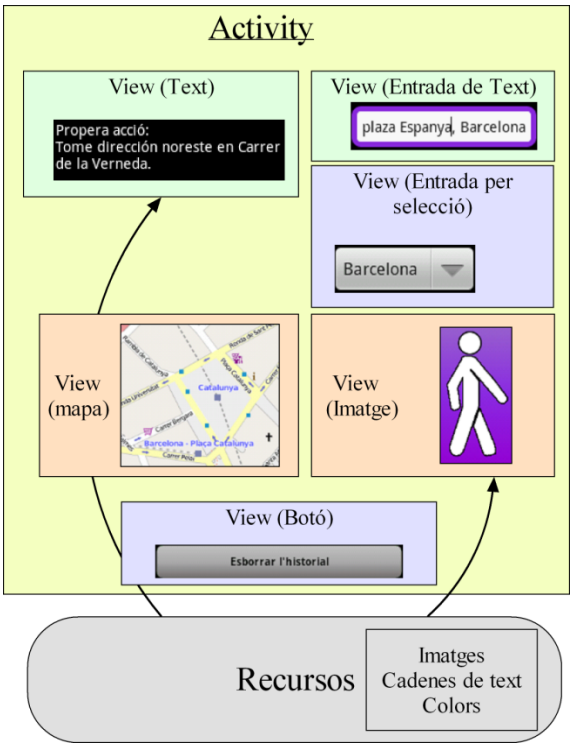


Diagrama 1 - Relació entre Activity - view - recursos

A continuació es farà un senzill repàs a les principals *Views* y *ViewGroups* utilitzades per la realització d'aquest projecte.

3.2.1 VIEWGROUPS I VIEWS

Les *Views* són uns tipus de *widgets* de la interfície gràfica d'una aplicació d'*Android*, ja que les activitats les content. A més a més les *Views* representen elements de la pantalla i s'encarreguen d'interactuar amb l'usuari a través dels events que proporciona el propi *Android*.

Una *Viewgroup* és una agrupació de *Views*, i en si mateixa, també es considera una altra *View*.

Així doncs, les *Views* són un punt molt important per aquest mòdul, mitjançant les

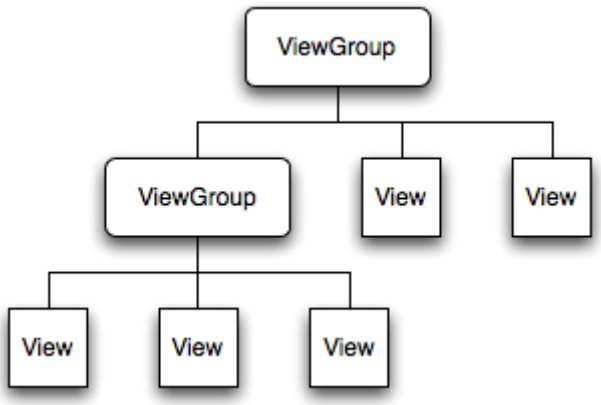


Diagrama 2 - Jerarquia de les View

quals s'ha intentat dissenyar la interfície de l'aplicació seguint el disseny universal.

Cada *Activity* conté un arbre jeràrquic d'elements *View*, el qual els seus elements adopten diferents formes i mides.

En el *Diagrama 2* es pot veure un diagrama amb la jerarquia de *Views*, les quals s'agrupen per *ViewGroups*.

En el següent exemple (*Figura 7* i *Codi 1*), es pot veure el mètode d'inicialització de les *Activities*, a través del mètode *setContentView*, s'associa l'arxiu *main.xml* que es troba dins el directori */res/layout/* del mòdul de la interfície gràfica.

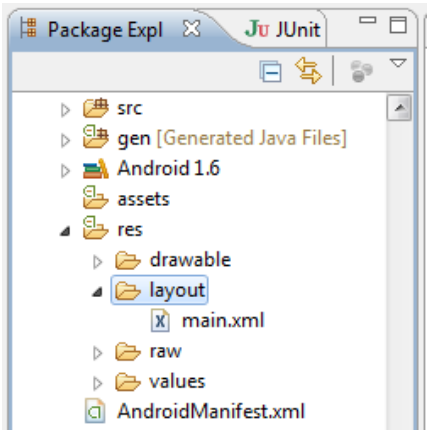


Figura 7 – Sistema de Carpetes per als Layouts

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

Codi 1 - Associació del layout amb l'activity

Tant *Views* com *ViewGroups* tenen atributs comuns que són de molta importància, com el *id* (a l'XML es defineix amb android: *id = "@+id/nom_id"*), que serveix per identificar i modificar els seus valors de manera dinàmica, el *layout_height* i el *layout_width* que com els seus noms indiquen, permeten modificar les seves mides.

Un altre atribut de gran importància és el *visibility*, que ens permet mostrar, ocultar o fer desaparèixer (sense ocupar res) una vista.

Per tal de realitzar les tasques del mòdul de la Interfície d'usuari, s'han utilitzat els dos mètodes d'incorporar les *View* a l'activitat de manera conjunta. O sigui, s'ha realitzat el *layout.xml* i s'ha associat amb l'activitat. Seguidament s'han realitzat canvis sobre els elements de l'activitat dinàmicament en el codi. Aquesta acció s'ha dut a terme així, degut a que depenent de la interacció de l'usuari amb l'aplicació requeria realitzar canvis de colors, mida,... de forma dinàmica.

Un *layout* està estructurat per una sèrie de *tags*(etiquetes), els quals indiquen que és un element *View* de la pantalla. D'aquest element se li incorporen una sèrie de propietats per a definir-lo, com per exemple, la posició, el color, la mida, si és clicable, etc... En el *Codi 2* es pot observar el format que té, on el *LinearLayout*, *TextView*, *Button* són elements *View* incorporats de forma jeràrquica.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hola, sóc TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hola, sóc Button" />
</LinearLayout>
```

Codi 2 - Exemple de format d'un Layout

Per tal de tenir una visió més detallada dels elements que component les *Views* i *ViewGroups*, s'ha adjuntat en l'*Annex C* un apartat amb descripció dels components més utilitzats en la *GUI* de l'aplicació.

A més de tots els elements que s'han comentat en l'annexa, també són necessaris altre tipus d'elements com són els menús, les notificacions, etc... però per qüestió d'espai, no s'ampliarà en aquesta ocasió.

3.2.2 ACTIVITY

Tal i com s'ha introduït, una activitat representa una de les pantalles d'una aplicació. Aquesta té un cicle de vida des de que s'inicia fins que es tanca l'aplicació, determinat per el sistema *Android*, per tal de poder garantir flexibilitat amb els processos del terminal.

El cicle de vida d'una activitat són funcions que s'executen dins el codi de l'activitat, les quals representant diversos estats en que pot estar la pantalla. En el cas d'*Android*, en fa servir 7 estats. Per tal de veure quins són, a continuació es mostren els estats indicats i la seva funcionalitat i l'estat d'interacció amb l'usuari així com el seu *Diagrama 3*.

- ***onCreate():*** Una vegada s'inicia l'aplicació, s'executa aquesta funció, la qual contindrà el codi per tal d'inicialitzar la pantalla amb les *View* corresponents. També dóna accés a qualsevol estat emmagatzemat prèviament en forma de *Bundle*.
- ***onStart():*** S'invoca quan l'activitat està visible en pantalla per a l'usuari.
- ***onPause():*** Aquesta funció s'utilitza al parar l'activitat al passar en un segon pla. La funció guarda l'estat de l'activitat per quan es vulgui continuar amb l'activitat en el estat en que es va parar.
- ***onResume():*** Funció la qual es fa servir sempre, ja que s'invoca quan l'activitat comença a interactuar amb l'usuari, tan si ve del *onCreate*, com si ve del *onRestart*.
- ***onStop():*** S'activa quan l'activitat que esta en segon pla, fa molta estona que ho esta i es vol alliberar els recursos que utilitza o bé quan es vol passar a parar l'activitat. En aquest cas, l'activitat passa a un estat d'invisibilitat per a anar als següents cicles de vida.

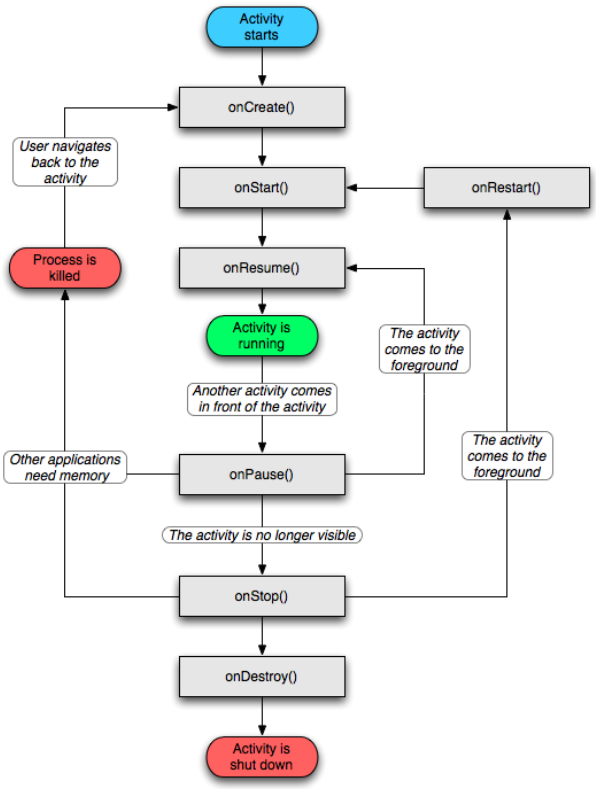


Diagrama 3 - Diagrama del cicle de vida d'una Activity

- ***onRestart():*** Serveix per a tornar reiniciar l'activitat i no haver de inicialitzar-la de nou. Això es possible si l'activitat encara està en la pila d'activitats.
- ***onDestroy():*** Funció per a eliminar l'activitat en el cas que se li hagi indicat o bé si el sistema decideix parar-la per alliberar recursos.

Així doncs, a l'hora d'implementar el codi de totes les pantalles de l'aplicació, s'ha de tenir en compte que per a una correcta implementació, s'hauria de posar les funcions més importants del cicle de vida depenent de la funcionalitat de l'*Activity*.

3.3 POJOs (Plain Old Java Objects)

Definició i descripció dels POJOs fets servir a la interfície gràfica.

Aquesta denominació va aparèixer cap al Setembre de l'any 2000, durant la preparació d'una conferència on es parlaven dels beneficis de la codificació de la lògica de negocis en objectes senzills regulars, en comptes de fer servir *beans* (dependents de *frameworks* especials que segueixen els estàndards EJB anteriors al 3.0).

No va ser pas una nova tecnologia sinó un nom nou per revaloritzar la típica programació orientada a objectes (Rebecca Parsons, MacKenzie Josh i Martin Fowler).

S'ha fet ús d'aquest tipus d'objectes per tal de poder enviar, rebre informació, o d'emmagatzemar utilitats i constants de manera adient. A continuació es podrà veure una breu explicació dels més importants que s'han fet servir al mòdul de la interfície gràfica, però més endavant s'explicaran al seu context particular.

3.3.1 PREFERENCE

Al seu interior s'hi pot trobar totes les preferències de l'usuari per la utilització de l'aplicació, com poden ser: el seu *theme* (tema, aparença) utilitzat, si necessita accessibilitat o no, si es dretà o esquerrà, si vol missatges d'informació addicionals, ... En el *Diagrama 4* es poden veure els seus atributs.

pI : Preference			
boolean vibrate;	boolean dynamicButtonBackground;	int ButtonTextRed;	int ScreenBackgroundRed;
boolean volumne;	boolean dynamicButtonText;	int ButtonTextGreen;	int ScreenBackgroundGreen;
boolean screenInitial;	boolean dynamicText;	int ButtonTextBlue;	int ScreenBackgroundBlue;
int frequency;	boolean dynamicImageButtonBackground;	int TextRed;	
int transfers;	boolean DynamicScreenBackground;	int TextGreen;	float speechRate;
int profile;		int TextBlue;	boolean VisualAdvice;
int typeResponse;	int ButtonBackgroundRed;	int ImageButtonBackgroundRed;	int verbalizationLevel;
boolean accessibility;	int ButtonBackgroundGreen;	int ImageButtonBackgroundGreen;	boolean stopNotifications;
int theme;	int ButtonBackgroundBlue;	int ImageButtonBackgroundBlue;	boolean rightHand;

Diagrama 4 - Atributs del objecte Preference

3.3.2 ROW

Aquesta classe inicialment es va crear per tal de poder començar les *Activities* d'informació i selecció de mapa quan els mòduls que havien de donar la informació encara estaven implementant la solució.

Afortunadament, quan aquesta solució va estar implementada, es va poder re-aprofitar, per tal de mostrar la informació desglossada dels temps de viatge, adaptada a l'usuari, de manera més entenedora. En el *Diagrama 5* es poden veure els seus atributs.

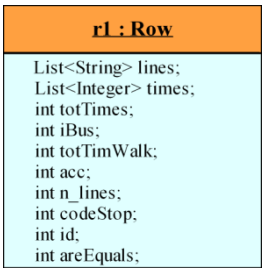


Diagrama 5 - Atributs del objecte Row

3.3.3 MENUROWPRINCIPAL

Per tal de poder mostrar les *ListView* amb un contingut més complex per cada fila, que no pas una simple cadena de text, cal tenir una llista d'objectes. Aquesta llista d'objectes contenen la informació de cada fila del *ListView*. Per tal de poder personalitzar la llista del menú principal, s'ha utilitzat aquest objecte el qual conté una imatge i una cadena de text. En el *Diagrama 6* es poden veure els seus atributs.

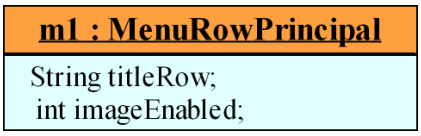


Diagrama 6 - Atributs del objecte MenuRowPrincipal

3.3.4 GUIDANCE

Per poder rebre tota la informació referent al guiatge, es va haver de dissenyar unes classes, lo més senzilles possibles. Aquesta classe en particular, només conté un identificador de tipus *int*, que serà heretat pels objectes de les seves classes derivades. S'ha implementat el seu *Parcelable (i a les classes derivades també)* degut a que *Android* ho demanava per tal de poder ser enviat dins dels paquets als *Intents*.

Classes derivades:

Depenent del tipus de guiatge que es necessiti en cada moment, es va decidir dissenyar 3 tipus de *Guidance* ben diferenciats i un altre tipus, per poder representar una brúixola. En el *Diagrama 7* es poden veure els seus atributs.

3.3.5 GUIDANCE_WLK

Aquesta classe porta tota la informació necessària per orientar a l'usuari en el moment del guiatge a peu, com pot ser informació sobre els metres que falten per girar, indicacions de la acció actual i de la propera, etc...

3.3.6 GUIDANCE_WAIT

Els objectes d'aquesta classe porten la informació referent al moment en que l'usuari es troba a la parada de l'autobús esperant que arribi el proper. Conté els minuts que falten per arribar, el codi de la parada, la línia, les característiques del bus que arribarà, etc...

3.3.7 GUIDANCE_BUS

Conté la informació important per el moment en el que l'usuari es troba a l'interior de l'autobús, com el nom de la parada, si és la darrera o no, etc...

3.3.8 GUIDANCE_COMPASS

Aquesta classe ha de permetre orientar a l'usuari, tant en la interfície *blind* com en la no accessible visualment, per aquesta raó, conté els graus a girar, direcció, sentit, etc...

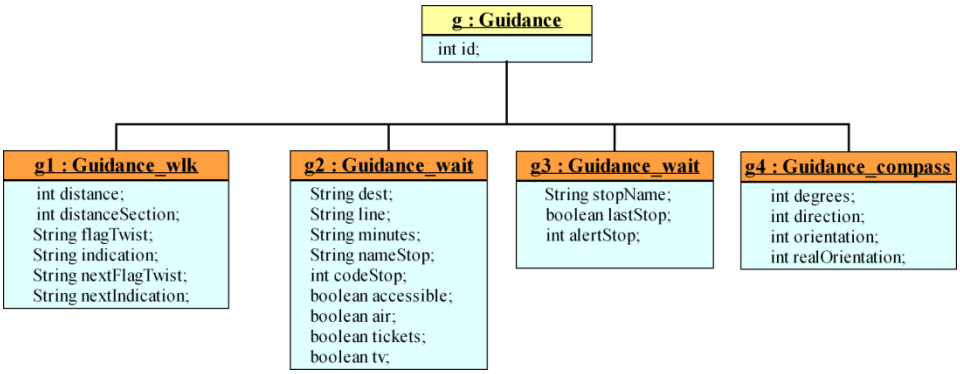


Diagrama 7 - Atributs dels objectes Guidances

3.3.9 FAVORITS

Per tal de poder guardar la informació d'una localització del recorregut en la base de dades de Favorits, s'ha creat un objecte que conté la informació la qual es requereix per aquesta funcionalitat. En el *Diagrama 8* es poden veure els seus atributs

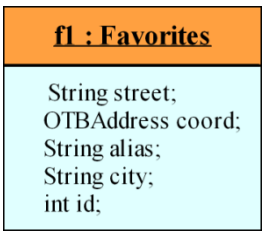


Diagrama 8 - Atributs dels objecte Favorit

3.3.10 CONTACTES

Una altre manera de guardar la informació d'una localització del recorregut és fer servir l'agenda. En aquest cas, la informació que es pot incorporar en ella es molt gran i per això s'ha dissenyat un objecte el qual conté la informació més rellevant a guardar i recuperar. En el *Diagrama 9* es poden veure els seus atributs.

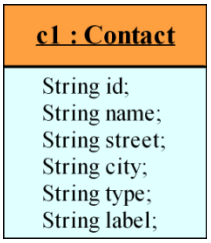


Diagrama 9 - Atributs dels objecte Contact

4 Disseny de l'aplicació OnTheBus

El primer pas a realitzar abans de programar és fer el disseny de la aplicació que volem desenvolupar. Així doncs, en aquest apartat es podran veure els diversos dissenys fets per l'aplicació OnTheBus, així com el resultat final del disseny un cop implementat en Android.

4.1 Dissenys Inicials

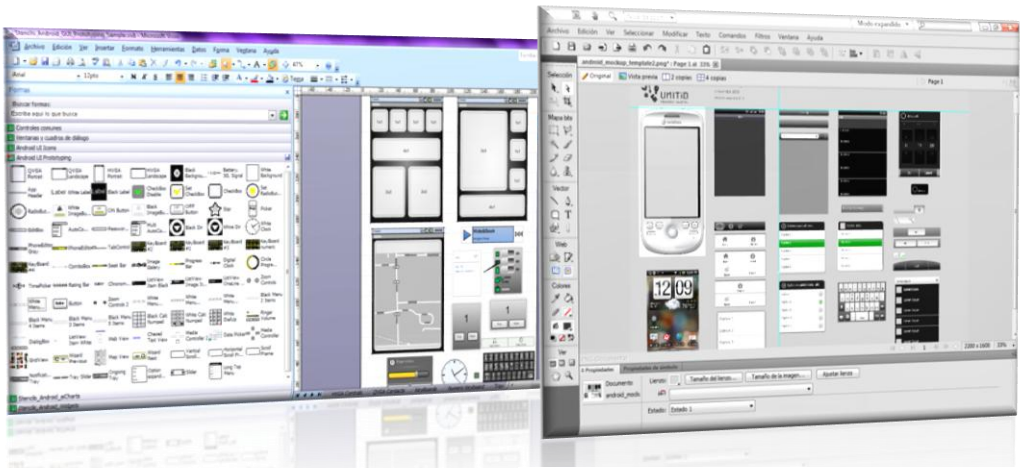
Per tal de realitzar el disseny cal tenir un mínim de requeriments i la idea de com es vol fer l'aplicació.

En el nostre cas, sabem que volem fer una aplicació per a guiar a un usuari des d'una localització inicial a una de final. Per tant els punt claus seran: la localització de l'usuari, l'entrada de dades per part de l'usuari d'un destí, diverses rutes calculades a seleccionar per l'usuari i un guiatge en mapa. Basant-nos en aquests punts, podem començar a dibuixar quina forma tindrà l'aplicació, tan la interfície gràfica com la lògica del programa.

Per tal de fer un disseny inicial, ens cal un software de disseny amb plantilles d'*Android*, per tal de que el disseny de l'aplicació sigui més real i equivalent amb elements de les aplicacions d'*Android*.

Tenim dos software per tal de fer el disseny:

- Adobe Firewoks CS5 amb plantilles (*Imatge 1*)
- Office Visio amb AndroidGUI Prototyping-v1.4 (*Imatge 2*)



Imatge 2 – Office Visio amb AndroidGUI Prototyping-v1.4

Imatge 1 – Adobe Firewoks CS5 amb plantilles

Cadascú té les seves plantilles d'*Android* i les seves funcionalitats, per tant, a l'hora de fer el disseny, només cal elegir el que més còmodes ens sentim i el que sapiguem fer servir més. Per tant, cal fer un parell de dissenys, un amb cada software, per tal de decidir.

El primer disseny de la aplicació *OnTheBus* es pot veure en *l'Annex D* amb el nom de *Prototip 1*. Inicialment partim de que volem que el guiatge es faci en varies ciutats les quals s'ha de decidir si és l'usuari qui elegeix que es troba en una de les ciutat on oferim el servei, o bé si som nosaltres qui ho ha de localitzar automàticament. Això comporta decidir també, si és l'usuari qui introdueix el nom del carrer d'origen de la ruta o bé si el localitzem nosaltres.

En aquest disseny, es té en compte el disseny universal, i per tant, els dissenys dels elements són grans. També es té en compte les diverses maneres d'entrar les dades a través de la interfície amb el teclat, veu o *gestures*. En quant al guiatge, suposem que tindrem els controls bàsics per a manipular el mapa, a més a més que el recorregut a fer estigui indicat mitjançant una línia pintada en el mapa.

Aquest primer prototip és molt bàsic i ens serveix per introduir quins requeriments falten per estudiar i decidir, així com posar-se d'acord en quines funcionalitats es podrien arribar a programar.

El segon disseny es pot veure en *l'Annex D* amb el nom de *Prototip 3*. Aquest nou recau en que hi ha un altre disseny de l'aplicació feta per un company de l'equip de treball. En aquest nou disseny es plantegen uns altres dubtes de sobre com es vol orientar l'aplicació. Es parteix que la localització de la ciutat i per tant la del carrer la farà automàticament l'aplicació. Aquesta idea és la que més atrau, tot i el marge d'error que pugui tenir el GPS. També s'hi recalca que cal un disseny universal i per tant les icones han de ser grans i tenir uns colors prou visibles per a totes els usuaris inclosos el que tinguin una visió reduïda i pels que pateixin daltonisme. En quan als usuaris cecs, veiem que caldrà donar tota la informació mitjançant veu.

Per tal de no discriminar a cap usuari i fer una única interfície, es proposa que les pantalles continguin 2 botons grans a les parts inferiors, les quals siguin prou accessibles. A la part superior estarien els elements menys accessibles, com ara l'entrada per teclat, per tal que aquests usuaris no en facin us degut a la complexitat que els requeriria. En quan al mapa, aquest continua tenint la mateixa tendència de disseny que l'anterior.

Entre els tres prototips inicials, els quals es poden veure en la *Imatge 5*, *Imatge 3* i *Imatge 4*, s'elegeix continuar un disseny a partir del segon, per tant, a partir del disseny del company d'equip.

El motiu de l'elecció és degut a l'estat avançat i més realista en que es trobava. En els prototips 1 i 3 s'intenta plantejar quins requeriments falten, i com es podria encarar el problema de disseny, mentre que el prototip 2 està més clar com ha de ser.

Imatge 5 – Prototip 1

Imatge 3 – Prototip 2

Imatge 4 – Prototip 3

A partir del segon prototip, el qual se sembla poc al disseny final, es varen realitzar les millores discutides entre els companys i professors. D'aquest disseny en surt el quart prototip, el qual es varen concretar la majoria de requisits envers el usuari.

De cara a dissenyar aquest quart prototip, es va separar la part visual de la lògica, la qual resulta complicat decidir l'ordre de les pantalles, la quantitat d'informació a mostrar, els moviments de les dades internament, etc..

En *l'Annex D* es pot veure el prototip amb el nom de *Prototip 4* i seguidament els dissenys lògics de l'aplicació en *l'Annex E*.

En aquest prototip, el qual apareix una versió més fidel del que es voldrà programar, es defineix que hi haurà un apartat de configuració on es podrà manipular diverses opcions de l'aplicació com ara els colors, els controls, les notificacions,...

Tenint en compte que l'aplicació busca ser el més universal possible per tal d'arribar al màxim d'usuaris possibles, partim de la idea que cal que la interfície contingui perfils per ajustar-se a la necessitats de tothom i no perdre l'atractiu visual d'un entorn típic d'*Android*.

Es va proposar la creació de perfils per cada usuari que volgués fer servir l'aplicació per a que es guardessin unes preferències depenen de les seves necessitats.

Aquesta idea no va agradar, degut a que no es creu que hi hagi més d'un usuari en un terminal. Així doncs de cara al producte final, s'ha mantingut la creació d'un únic perfil per un únic usuari, el qual podrà canviar-lo depenent de les seves necessitats.

La proposta és que hi hagi 5 perfils, un per a l'usuari sense limitacions funcionals i els altres 4 per a diferents limitacions funcionals. Aquesta proposta pretén activar uns modes d'entrada d'informació o uns altres depenen de les necessitats de l'usuari. Això vol dir, que si tenim a un usuari amb limitació funcional visual, s'activarà la síntesis de veu per tal de guiar-lo per a totes les pantalles. Així doncs es pretén facilitar al màxim l'accés a l'aplicació depenent de les necessitats de cada usuari.

4.2 Disseny Definitiu

El disseny finalment realitzat conté dues parts de codi diferenciades per tal de poder cobrir totes les necessitats dels usuaris.

Per una banda hi ha la Interfície 1 per a usuaris sense limitacions funcionals visuals i per l'altre la Interfície 2 per a usuaris amb limitacions funcionals visuals. Aquesta separació d'interfícies pretén aplicar el primer principi del disseny universal, el qual diu que cal proporcionar les mateixes formes d'ús per a tothom, el qual en cas de no poder ser idèntiques es faci de manera equivalent.

Després de provar únicament la interfície 1, es va concloure que no era prou apta als usuaris amb limitacions funcionals visuals, degut a que no tots els terminals tenen el botó *TrackBall*. En el cas que l'usuari no tingui un terminal amb el TrackBall, la interfície 1 es queda inaccessible per a l'usuari, degut a que no es pot explorar la pantalla sense clicar als elements. Això és degut a que el funcionament per defecte d'*Android* és d'accedir a les funcionalitats d'un element mitjançant un sol clic.

Així doncs, per tal de cobrir totalment l'accessibilitat d'*Android* es va proposar de fer una segona interfície, la qual permet moure's per l'aplicació amb un senzill funcionament.

La interfície 2 està feta integrament de menús els quals es poden explorar sense la necessitat de botons físics. Això és degut que s'ha incorporat una barra lateral (ja sigui a la dreta o a l'esquerra) que serveix per moure's pel llistat d'elements del menú. Mentre l'usuari es mou per aquesta barra es van passant pels diversos elements del menú els quals es van seleccionant. A continuació l'usuari pot fer un doble clic en la pantalla per tal d'elegir la opció seleccionada i passar al següent menú.

Sobre l'entrada d'informació a la interfície, es pot fer mitjançant *Gestures*, el qual s'ha desenvolupat a partir de la llibreria d'accessibilitat d'*Android* o mitjançant el reconeixement de veu.

En quant a la interfície 1, conté diversos elements distribuïts segons la pantalla en la que estigui, per tant, poden ser *botons*, *EditTexts*, *RadioGroups*,*mapes*,etc. Hi pot haver la veu activada i per tant, els elements del *layout* contenen un camp de descripció accessible amb el *Trackball*. Per tant, un usuari amb una limitació funcional visual, també podria fer ús d'aquesta interfície amb un terminal adequat. De totes maneres, i veient la tendència dels terminals es quedin sense botons, la millor solució és que utilitzin la interfície 2.

En la *Imatge 6* i la *Imatge 7* es pot veure la diferencia entre ambdues interfícies, en la que una garanteix la plena accessibilitat amb una metodologia senzilla i en l'altre garanteix el màxim potencia que ofereix *Android* amb la seva diversitat d'elements. Com es pot observar, ambdues corresponen a la mateixa activitat, les quals tenen com a mateix objectiu guardar les configuracions de notificacions. D'altre banda tenen una forma diferent en quant a la funcionalitat d'interacció entre l'usuari i la interfície.

Imatge 7 – Interfície 1 basada en tots els elements d'Android

Imatge 6 – Interfície 2 basada en pantalles de menús.

5 Accessibilitat en Android

Android incorpora una API d'accessibilitat per a que els programadors en puguin fer ús i garantir la màxima accessibilitat a les aplicacions. Les eines que inclou són els revisors de pantalla, síntesis de veu, vibració i sons de notificacions.

Els revisors de pantalla són unes aplicacions d'accessibilitat per a usuaris cecs, els quals a través de la bola de navegació *trackball*, es poden moure pels diversos elements de la pantalla.

Els usuaris es poden moure d'element en element amb el *trackball*, fent que l'element quedi seleccionat però no pressionat. Llavors, si l'element conté una descripció, el revisor fent uns de la síntesis de veu dirà quin element és i per a que serveix. La descripció de l'element dependrà del programador de l'aplicació, i pot ser que no en tingui o bé que no sigui prou adequat. Com l'element no es pressionat, quan l'usuari vulgui accedir a un d'ells tan sols haurà de clicar el *trackball* si fa la funció d'enter. Llavors l'estat de l'element passarà a pressionant i realitzarà l'acció que tingui programada.

Tal com s'ha dit, els programadors han de incloure les descripcions a totes elements de les pantalles de l'aplicació que hagin desenvolupat. Per tal de poder incloure aquestes descripcions hi ha dues maneres de fer-ho:

- En els atributs dels *layouts* XML:
    - `android:contentDescription="Comentari"`
- En el codi:
    - `varTextView.setContentDescription("Comentari");`

Actualment hi ha dos revisors per a android que són:

- *TalkBack* – Revisor de pantalla oficial per a *Android*
- *Spiel*

Entre els dos revisors de pantalla, són força semblants. Els dos tenen la mateixa funcionalitat i objectius a dur a terme, identificar els elements amb el *TrackBall* i notificar quins són.

Un cop provats, la diferencia trobada és que en el *TalkBack* si es troba amb una paraula escrita tota en lletres en majúscules, no la pronuncia com a paraula sinó que la lletreja. En canvi el *Spiel* pronuncia la paraula correctament. Per posar un exemple:

- El *TalkBack* quan es troba amb Demo COLLAPSE diu "Demo ce o ele ele a pe ese e" en canvi el Spiel diu "Demo COLLAPSE".
- El *TalkBack* quan es troba amb Demo UNO diu "Demo u ene o" en canvi el Spiel diu "Demo UNO".

Tot i aquesta petita diferenciació, els revisors només ajuden a traduir el contingut de la pantalla a veu, el problema més important que tenen és a l'hora de introduir . Només tenen dues maneres, o bé es servir el reconeixement de veu o bé pel teclat. El problema rau en que el teclat sol ser tàctil i per tant els usuaris cecs no en poden fer ús. Llavors, estan obligats a buscar un terminal amb teclat físic si volen emprar aquest tipus d'entrada de dades.

El problema del reconeixement de veu es troba en el soroll ambiental. Si s'està per un ambient de poc soroll el marge d'errors és baix i l'entrada de les dades seria correcte, en canvi, si s'està per un ambient sorollós com ara el carrer, el percentatge d'errors augmenta. A part d'això, hi ha poca compatibilitat entre el revisor de pantalla i el reconeixement de veu degut a que el revisor pot estar dient alguna per la síntesis de veu quan el reconeixement de veu esta en mode gravació. En les proves fetes, el revisor diu "Hablar ahora" mentre el reconeixement de veu ja esta en funcionament i per tant detecta el que diu el revisor i ho grava i no et dona la possibilitat de que parlis.

En aquest cas, en l'aplicació es podria preveure aquest problema i començar la gravació després que el revisor digui "hablar ahora" esperant uns segons abans d'iniciar la gravació.

També ens hauríem d'assegurar que l'usuari pot navegar per tots els controls de l'aplicació fent servir el *trackball* (però no tots els terminals el porten). Per últim, assegurar-nos que els controls de navegació de la nostra aplicació són retornats en ordre transversal amb significat coherent.

Existeix també l'aplicació *Blind*, però aquesta aplicació es un entorn tancat accessible. O sigui, que la diferencia que té amb el revisor de pantalla es que no dóna accés al sistema operatiu ni als controls visuals de la interfície. I el problema està en que la pantalla es torna negre i per tant si es vol demanar ajuda a algú que no sigui cec, no podria veure res de la pantalla.

Hi ha més elements d'accessibilitat derivats del projecte *TalkBack*. Aquestes són dues aplicacions per a complementar-lo:

- *SoundBack*
- *KickBack*

El *SoundBack* proporciona sons cada cop que es va passant pels diversos elements de la pantalla i quan els selecciones. D'aquesta manera, notifica a l'usuari que s'està movent i seleccionant els elements.

En canvi el *KickBack* el que fa és emetre vibracions quan vas passant pels elements de la pantalla i quan els selecciones. Per tant, té la mateixa finalitat que el *SoundBack* de notificar els moviments, però sobre un altre canal de comunicació.

Es poden activar a l'hora el revisor de pantalla amb el SoundBack i el KickBack, tot i que cal tenir en compte que hi haurà moltes notificacions a l'hora i pot desorientar o atabalar a l'usuari.

Per activar i desactivar els programes de revisor de pantalla, els sons i la vibració cal anar al *Menú* → *Ajustes* → *Accesibilidad* en el sistema operatiu *Android*. Primer cal seleccionar si es vol activar la accessibilitat, així els programes de revisors, sons i vibracions s'habiliten. Després es poden seleccionar els programes que es desitgin.

Imatge 8 – Aplicació Mobile Accessibility

Cada cop hi ha més empreses que intenten fer programes accessibles. Un exemple seria *Codefactory*, el qual ha trobat una solució senzilla d'accessibilitat per a *Android*. Aquesta solució es basa en l'aplicació *Mobile Accessibility* (*Imatge 8*), la qual dóna accés a les funciones mes importants del terminal. Navegar per Internet, llegir i escriure correus electrònics i missatges SMS, consultar dades del GPS, gestionar les trucades, accedir al calendari i a l'agenda de contactes. També permet accedir a algunes configuracions del terminal.

Per tal de que l'usuari es pugui moure per la interfície, no li cal el *trackball* degut a que els elements es poden seleccionar amb un sol clic i els hi cal un doble clic per a accedir a ells. Així doncs, l'usuari pot tocar la pantalla tàctil sense por a clicar els elements mentre es mou per les pantalles.

Aquesta aplicació és una bona solució, ja que no requereix de tecles físiques per a moure's per l'entorn tancat. Tot i així, és una interfície simple i per tant que no permet anar més allà d'aquestes funcionalitats descrites i es queda en ser un entorn tancat i limitant el sistema. Així doncs si s'accedeixen a aplicacions fora d'aquest entorn poden deixar de ser accessibles. De totes maneres la mateixa aplicació ja avisa en quin moment s'està apunt de sortir de l'entorn tancat.

Actualment, i veien la tendència a la desaparició dels botons físics dels terminals, ens trobem amb un greu problema per als usuaris cecs. Els revisors de pantalla són funcionals si els terminals incorporen els *trackball*, si aquests desapareixen, els revisors de pantalla deixen de ser útils degut a que els elements de les pantalles tàctils es pressionen amb un sol click.

Així doncs, els usuaris que intentin moure's per les pantalles amb el revisor veuran com descriu els continguts, però que els elements no es queden seleccionats com el *trackball*, sinó clicats. Per tant, aquests usuaris es veuen obligats a buscar terminals amb teclats físics i amb un *trackball*. El nombre de terminals actuals amb aquestes condicions en són pocs, si ens fixem amb l'alta gamma de terminals existents. Així doncs, aquests usuaris han de pagar més per un terminal, degut a que els terminals de gamma baixa solen prescindir dels botons físics per abaratir costos.

6 Procés d'Implementació del Codi

En aquest apartat es podrà veure el procés d'implementació de diversos apartats de l'aplicació. La divisió de l'apartat bé donada per les diverses funcionalitats que s'hi han incorporat per tal de garantir els requisits de l'aplicació, així com els requisits definits en la fase de disseny. Aquestes funcionalitats són les següents:

- *Preferències*
- *Personalització*
- *Idiomes*
- *Favorits*
- *Contactes*

6.1 Preferències

Android posa a disposició la classe *PreferenceActivity*, juntament amb el fitxer *preference.xml*, per tal de facilitar la creació d'una activitat per a guardar i recuperar les preferències. La utilització d'aquesta classe és molt senzilla, cal que l'activitat hereti de *PreferenceActivity* i a continuació, en el mètode *OnCreate* es faci l'associació amb el *xml* de *preference* que es troba en la carpeta res.

Després, cal omplir el fitxer xml amb els atributs adequats de preferències, llistes, *checkbox*, *textview*, *edittext*, subpantalles, etc. En cada atribut cal posar bé les "*android:key*" que serà el nom de la preferència de l'element. Aquesta "*key*" ens caldrà per tal de poder accedir a la preferència en un moment donat.

A més a més ens caldrà un objecte anomenat *SharedPreferences*, el qual ajudarà amb els seus mètodes a recuperar les preferències a partir de la "*key*".

Al tenir-ho amb aquest sistema, el programador no s'ha de preocupar d'on cal guardar la informació ja que el mateix *Android* ho gestiona, només s'ha de preocupar de recuperar els valors quan en faci falta.

Així doncs, amb aquesta classe, *Android* permet crear de forma estàndard les preferències. El contingut de les preferències està marcat pels atributs del *xml*, així doncs té una limitació en quant a personalització del contingut que hi voldrem posar i del format. Aquesta limitació, la qual es va trobar al desenvolupar el fitxer *preference.xml*, és la raó que ha dut a seguir provant si hi havia

alguna altre manera d'afegir les preferències i poder crear activitats pròpies amb els propis *layouts* personalitzats.

Per tal de tenir les preferències personalitzades amb els requisits de la aplicació, s'ha trobat una segona manera de programar-ho. En aquest es tracta de fer servir l'objecte *SharedPreferences* per tal de gestionar de forma pròpia a on guardar les preferències i com recuperar-les.

Per tal d'aplicar-ho, s'ha d'organitzar diverses activitats amb els seus *layouts*, per tant cal dissenyar un apartat de configuració propi. En les activitats, en el moment que calgui guardar atributs de preferències s'ha d'obrir el *xml* on es vulgui guardar els atributs. En cas que al obrir el *xml* no existeixi, *Android* el crearà.

Llavors amb el editor de *SharedPreferences* i el seus mètodes d'escriptura segons el tipus de variables, es guarda l'atribut. En aquest cas, també es fan servir les "*key*" per tal de poder guardar d'una forma predeterminada el fitxer *xml* i posteriorment poder-ho recuperar. A continuació es pot veure el procediment d'escriptura (*Codi 3*) i el de lectura(*Codi 4*).

```
settings = getSharedPreferences("preferenceOTB",MODE_PRIVATE);
SharedPreferences.Editor editor = settings.edit();
editor.putBoolean("key", true);
editor.commit();
```

Codi 3 – Escriptura d'un atribut de preferències

```
settings = getSharedPreferences("preferenceOTB",MODE_PRIVATE);
preference.setVibrate(settings.getBoolean("key ", false));
```

Codi 4 – Lectura d'un atribut de preferències

Com es pot veure en el codi, el nom del *xml* per a guardar les preferències s'hi pot posar el que es desitgi. *Android* guardarà el *xml* en la carpeta de dades de l'aplicació *OTB* instal·lada. Crearà una carpeta especifica per a les preferències, en el qual es pot veure que

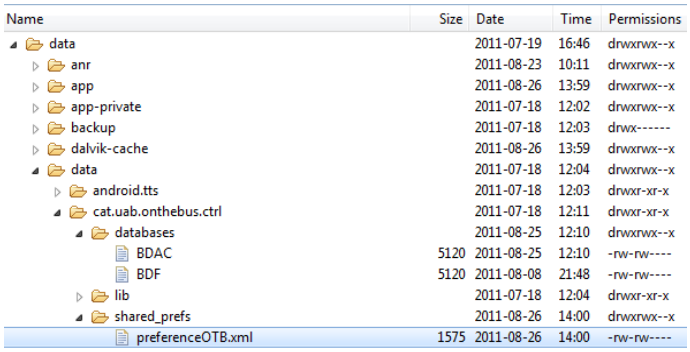
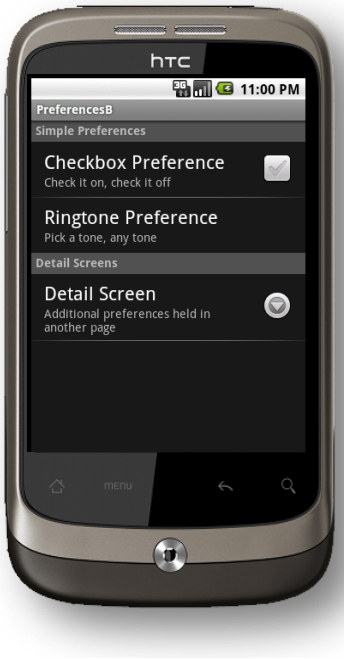


Figura 8 – Sistema de fitxers en el terminal mòbil. Seleccionat és pot veure el fitxer preferenceOTB.xml.

es podrien afegir més xml i després gestionar-los segons les necessitats, tal i com es pot veure en la *Figura 8*.

Aquesta segona forma, permet recuperar els atributs de preferència quan es necessiti, sense tenir que dependre d'un *layout* predeterminat *d'Android,* i així poder personalitzar-ho amb les necessitats i restriccions que hi hagin.

En la *Imatge 10* es pot veure la diferència entre una activitat amb "*PreferenceActivity*" amb una implementació amb del fitxer *preference.xml,* on no hi ha estils aplicats, ni *seekbar*. En canvi a la *Imatge 9* es pot visualitzar una activitat amb "*Activity*" amb un *layout* propi, barrejant atributs de *buttons*, *seekbar*, etc. amb personalització de colors i distribució dels elements en el *layout* de la forma desitjada.



Imatge 10 – Activitat amb PreferenceActivity

Imatge 9 – Activitat amb Activity i SharedPreference

Una vegada decidit quina metodologia seguir per implementar les preferències, queda veure quina és la millor forma de treballar amb elles internament en l'aplicació.

Degut a la quantitat de activitats que hi ha en l'aplicació, cal mirar quina és la millor manera de fer ús de les preferències de manera conjunta. Hi ha alguns mòduls, com el de cerca, que els hi caldrà accedir a certs atributs de les preferències per a fer restriccions de cerca en els busos. També

el guiatge variarà segons les preferències de l'usuari sobre la quantitat i la manera de notificacions que vulgui.

En un primer moment, al seguir la metodologia del *SharedPreferences*, es va provar en carregar els atributs de preferències que calguessin en cada activitat, en variables a utilitzar en aquella activitat en el moment oportú. Fer-ho d'aquesta manera genera que tots els mòduls implicats que vulguin utilitzar les preferències necessitin saber els noms de les "*key*" per tal de poder accedir-hi correctament i podria haver un mal ús sobre elles.

Enfront d'aquest problema, la millor solució és crear un objecte on es carreguin les preferències del fitxer *xml* cada cop que s'inicií l'aplicació. D'aquesta manera el objecte tindrà els seus mètodes de consulta i escriptura mentre s'utilitza l'aplicació sense necessitat de saber el nom exacte de les "*key*", excepte quan s'estiguin manipulant les dades en l'apartat de configuració. Així doncs, amb un sol objecte, qualsevol mòdul de l'aplicació pot consultar les dades de les preferències en un moment donat.

Un cop resolt aquest problema en sorgeix un altre. Els objectes i variables d'una activitat són pròpies d'aquesta activitat. Això vol dir que cal fer servir uns mètodes d'una classe proporcionats per *Android* per tal de passar les dades entre activitats.

La classe Intent és la que s'encarrega de la xarxa de comunicació en les aplicacions *Android*. Té moltes funcionalitats, però en aquest cas, només ens centrarem en el mode de passar informació entre activitats. Per tant, ens cal fer servir els seus mètodes dels *Extres* que són les dades addicionals que es passen al "Intent" per a que els enviï a la nova activitat que iniciarà.

Android aporta els mètodes per tal de passar variables simples i objectes genèrics com llistes de *strings*. Així doncs, els mètodes simples no es poden fer servir degut a que es té un objecte propi format de múltiples tipus simples (*int, bool, strings*). En aquest cas s'ha de mirar de fer servir un mètode més enrevessat de la Classe *Parcelable*.

El funcionament del *Parcelable* consisteix en agafar el objecte per *mapejar-lo* en una cadena de strings amb el mètode "*writeToParcel*". Aquesta cadena de *strings*, que és més senzilla de maneja, es passa a la següent activitat. Una cop en la següent activitat, es recupera el format del objecte recomponent els strings, en el mateix ordre que en el que s'han *mapejat* a l'escriptura, amb el mètode "*readFromParcel*".

En els següents trossos de codi (*Codi 5* i *Codi 6*) es pot veure el format d'enviament:

```
Bundle intent = getIntent().getExtras();
preference = intent.getParcelable("preference");
public static Preference readFromParcel(Parcel in) {
      boolean [] arrayBoolean = new boolean[1];
      in.readBooleanArray(arrayBoolean);

      preferencia_1= arrayBoolean [0];
      preferencia_2= in.readInt();
}
```

Codi 5 – Enviament de l'objecte preferències d'una activitat

```
intent.putParcelable("preference", preference);
public void writeToParcel(Parcel pref, int flags) {
      boolean [] arrayBoolean= new boolean[1];
      arrayBoolean [0] = preferencia_1;
      pref.writeBooleanArray(arrayBoolean);
      pref.writeInt(preferencia_2);
}
```

Codi 6 – Rebuda de l'objecte preferències en la següent activitat

No està recomanat utilitzar la classe *Parcelable* per enviar objectes propis entre activitats, degut a que és un procés lent i costós. Tot i que no es recomani, si no hi ha cap altre forma de solucionar el problema, llavors no hi ha cap més remei que utilitzar-lo.

El funcionament d'aquesta metodologia és correcte, i les preferències es poden passar a totes les activitats cada cop que s'inicialitza una. Tot i ser un procediment costós, al ser un objecte senzill compost dels tipus *int, float, boolean* i *string*, no es nota que en ull humà que trigui més en inicialitzar les activitats.

Tot i el seu bon funcionament, va sorgir un problema derivat de les comunicacions entre mòduls. En algunes activitats on el maneig de dades és alt, com ara l'activitat de seleccionar una de les ruta calculades d'un llistat de rutes, tenir l'enviament l'objecte dificultava la comunicació entre aquest mòdul i el del Nucli de l'Aplicació.

Degut a aquest problema, es replanteja quina era la millor manera de resoldre'l. En aquest cas, es va recorre a buscar algun patró de disseny que servis per tenir l'objecte accessible per a tothom en qualsevol moment sense necessitat d'enviar-lo entre activitats.

El *Singleton* respon a aquestes característiques. És un patró de disseny per a restringir la creació d'objectes a una classe a una única instancia i proporcionar un punt d'accés global a l'objecte.

Per tal de passar a l'objecte sigui d'aquest tipus de patró, s'han de treure els mètodes del *Parcelable* que hi havia en anterioritat "*writeToParcel*" i "*readFromParcel*" i incloure el mètode *getInstance()* (*Codi 7*)

```
public static synchronized Preference getInstance(){
      if(instance == null){
            instance = new Preference();
      }
      return instance;
}
```

Codi 7 – Mètode del patró de disseny Singleton

Per tal de poder accedir des d'una activitat a l'objecte de preferències o des de qualsevol punt d'altres mòduls, només cal cridar al mètode *getInstance()* per a obtenir una instancia única amb les preferències actualitzades.

Aquesta última metodologia d'utilització de les preferències és la més adequada. D'aquesta forma, tothom hi pot accedir i no hi ha problemes en les comunicacions entre aquest mòdul i el Nucli de l'Aplicació.

6.2 Personalització del OnTheBus

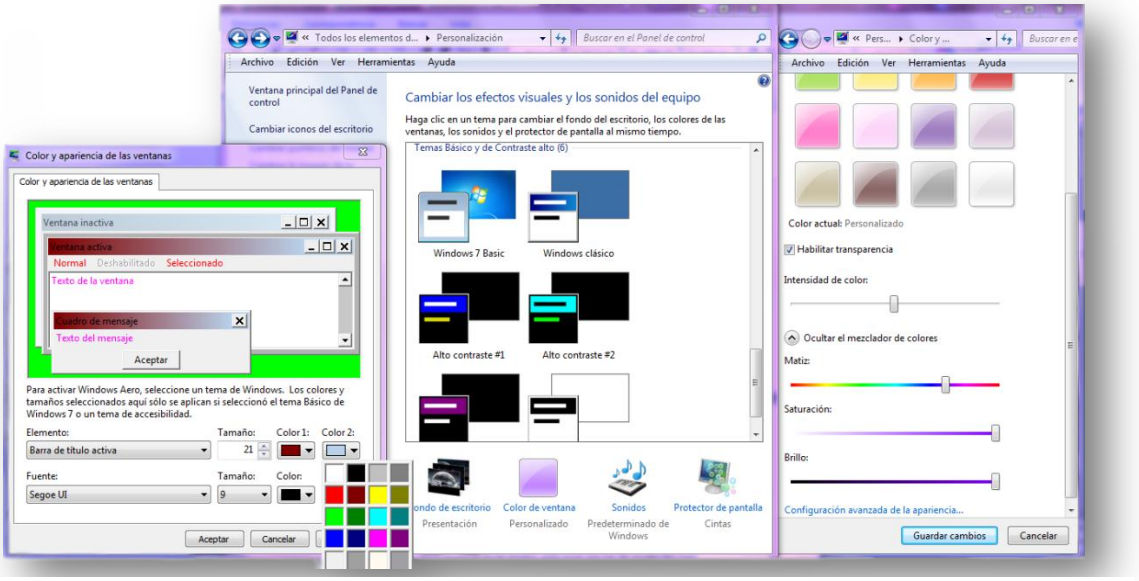
Avui en dia, hi ha una alta demanda per a poder personalitzar blogs, webs i aplicacions. Es pot veure per exemple, com el *Blogger* de *Gmail*, ha passat per diverses fases de disseny. Inicialment tenia les plantilles *Html*, on buscant per Internet se'n trobaven moltes per poder-les posar i ha evolucionat a tenir un propi dissenyador de plantilles per a que els usuaris no hagin de saber *Html* i ho puguin dissenyar amb un editor senzill. I per a un futur, ampliaran les funcionalitats d'aquest dissenyador, per tal que els usuaris en puguin gaudir més de l'ús.

Si ens fixem amb un servei de pàgines basades en *Drupal*, com ara el de la *UAB*, ens trobem que hi ha diverses plantilles per a elles, les quals no són gaire personalitzables. Els usuaris, en aquest cas, opten per demanar el màxim de personalització per a la pàgina. Colors de fons i de les lletres, tipus de lletra, funcionalitats noves, etc...

Com a exemples d'aplicacions, es pot veure l'aplicació de gravació de *CD-DVD Nero* (*Imatge 11*), on té un botó exprés per a canviar els colors de la carcassa entre múltiples temes predeterminats que s'han introduït.

Imatge 11 – Diverses carcasses de l'aplicació Nero 7

I per finalitzar, com a últim exemple, tenim el mateix sistema operatiu *Windows*, en aquest cas el *Windows 7*, el qual té diversos temes de contrasts de colors, es poden personalitzar les vores



Imatge 12 – Finestres de personalització de Windows 7

de les finestres, el menú inici i la barra de tasques com a una configuració estàndard, i també hi ha una configuració avançada per a poder canviar els colors i la mida de la lletra a diversos elements de les finestres (*Imatge 12*).

Així doncs, avui en dia i cada cop més, els usuaris volen poder personalitzar les aplicacions que utilitzen, ja sigui per a donar la seva imatge a la resta de usuaris, o ja sigui per a poder sentir-se més còmodes amb l'entorn a utilitzar depenen de les necessitats de cadascú, independentment de les discapacitats que pugui tenir.

Tenint en compte aquestes necessitats de part de l'usuari, s'han estudiat quines funcionalitats s'hi podien incloure en l'aplicació per tal de poder-la fer més personalitzable i accessible visualment.

Per tal de poder dur a terme aquesta tasca, s'han hagut d'estudiar els diversos elements de l'aplicació per tal de saber fins a quin punt era permès modificar-los de forma estàtica i dinàmica.

6.2.1 PERSONALITZACIÓ DINÀMICA

La personalització dinàmica es correspon a que l'usuari té el control de canviar els colors dels elements després d'haver compilat els fitxers *xml*. Per tal de dur a terme aquesta funcionalitat, cal que els canvis de colors en els elements de l'aplicació, siguin mitjançant funcions en el codi.

Els elements més fàcils de canviar-li els colors dinàmicament en són quatre: els botons, els botons amb imatges, el fons de l'*activity* i el text.

Android, ens proporciona diversos mètodes per a canviar aquests elements. En aquests casos, s'ha de fer servir les funcions:

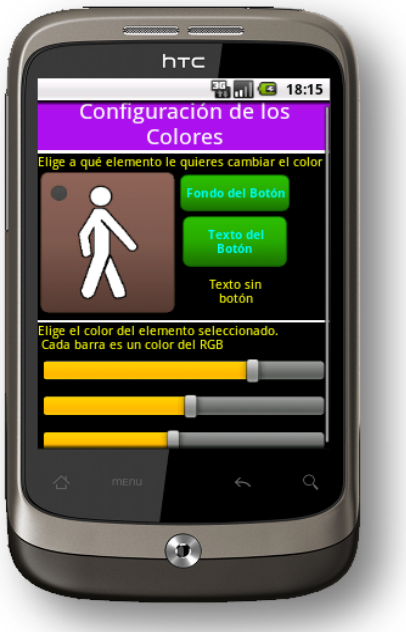
- *setColorFilter(int Color, Mode):* Per canviar el colors dels botons amb o sense imatge.
- *setTextColor(int Color):* Per canviar el color del text.
- *setBackgroundColor(int Color):* Per canviar el color del fons de l'activitat, directament amb el codi de color.
- *setBackgroundResource(int Color):* Per canviar el color del fons de l'activitat, amb el codi de color extret del fitxer de colors predeterminats.

Per tal d'aplicar els canvis de color a tota la col·lecció d'elements, s'ha desenvolupat una classe "*ChangeColor.java*" amb tots els mètodes possibles que fan falta per a modificar-los.

La idea és tenir un únic mètode "*changeElement(view,preference*)", en què a partir de la vista principal de l'activitat i de l'objecte de preferències on hi ha guardat els colors dels elements, es puguin canviar els colors d'aquests elements de la mateixa forma a totes les activitats de l'aplicació.

La funcionalitat del mètode és la següent:

1. Crida del mètode *changeElement()* en l'estat de *onCreate()*, un cop s'ha fet la relació entre el *layout* i l'activitat.
2. En el mètode, gràcies a un bucle sobre la *View* principal, es van recorrent tots els fills que té, els quals seran tots els elements del *layout*.
3. En cas de trobar un dels elements personalitzables dinàmicament, se li aplica la funció de canvi de color adequat.



Imatge 13 – Activitat Configuració dels colors.

Imatge 14 – Activitat Configuració dels fons de pantalla.

L'usuari pot trobar les funcionalitats per a canviar els colors dels elements en dues activitats diferents. En l'activitat de "Configuració dels colors" (*Imatge 13*) està dividida en dos funcionalitats.

1. La part superior serveix per a seleccionar sobre quin element, dels quatre que hi ha, se li volen canviar els colors.
2. Una vegada s'ha seleccionat l'element, a la part inferior hi ha 3 barres que representen els colors del *RGB*, per tant, hi ha la barra de tonalitats de vermell, la del verd i la del blau. L'usuari pot combinar els tres colors movent les barres i veurà com automàticament canvia el color de l'element seleccionat.

En l'activitat de "Configuració del fons de pantalla" (*Imatge 14*), tenim altre cop 3 barres que representen els

colors del *RGB*. Per tant, l'usuari ha combinar els tres colors movent les barres i veurà com automàticament canvia el color del fons de pantalla.

Cal tenir en compte, que l'usuari pot ocasionar una inconsistència de colors, si selecciona el mateix color per a diversos elements, fent que aquests no siguin visibles. En aquests casos, l'usuari podrà restablir els valors de fàbrica per a recuperar els colors per defecte.

6.2.2 PERSONALITZACIÓ ESTÀTICA

La personalització estàtica en l'aplicació es correspon a que l'usuari té diversos temes a poder elegir. Aquests temes s'han dissenyat dins dels fitxers *xml* i dels *layouts*, i per tant són previs a la compilació.

L'elecció de fer diversos temes amb diversos colors als elements sorgeix de lo costós que és fer una personalització dinàmica per a tots els elements. De forma estàtica, la possibilitat de fer els canvis en els elements creix exponencialment, ja que es poden personalitzar tots els elements.

Amb la creació de temes d'alt contrast, s'ajuda als usuaris amb limitacions visuals com ara persones grans o bé als usuaris amb daltonisme, a poder escollir uns colors que puguin distingir correctament. Per tal de poder garantir que els temes siguin d'alt contrast, aquests estan basats en els principals temes d'alt contrast del sistema operatiu Windows 7. S'han creat els següents 5 temes:

1. **Tema per defecte:** Amb una combinació de negre, blanc, gris i taronja.
2. **Tema alt contrast 1:** Amb una combinació de negre, groc ,gris i verd.
3. **Tema alt contrast 2:** Amb una combinació de negre, verd, gris i blau.
4. **Tema negre en alt contrast :** Amb una combinació de negre, blanc,gris i lila.
5. **Tema blanc en alt contrast** : Amb una combinació de blanc, negre i gris.

Android aporta un sistema d'estils, que combinats correctament amb els temes es poden aplicar amb facilitat a tots els elements de la *View*.

A la carpeta *values* del projecte, cal incloure diversos fitxers: *attrs.xml, theme.xml* i *styles.xml.*

El fitxer *attrs.xml* conté un llistat de tags que són els atributs propis que es fan servir com estils. Per tant, quan s'estigui definint un element en el *layout* de l'activitat i se li vol posar un estil propi, se li ha de dir que apliqui l'estil del "tag x" del fitxer *attrs.xml*.

El fitxer *theme.xml* conté la definició de tots els temes. Per cada tema inclou el llistat de tags definits *a attrs.xml* el qual els associa amb l'estil de l'element del fitxer *styles.xml.*

El fitxer *styles.xml* conté la definició de l'estil d'un element, per tant defineix, el color, la mida,… tots els atributs permesos en el *layout*. En el cas de voler definir varis colors d'un atribut, i per tant, definir els estats d'un element, cal fer ús de fitxers *xml* en la carpeta *drawable*. Aquests *xml* definiran quins són els colors per cada estat de l'element.

A mode de resum, es pot veure el *Diagrama 10*, el qual representa un diagrama de seqüències dels fitxers indicats en un exemple de l'estil del botó en l'estat seleccionat.

Per tal que una *Activity* faci servir un tema o un altre, cal que abans de fer l'associació entre *Activity* i el *layout* (*setContentView(layout.xml*) , es carregui el tema. Això es fa d'aquesta manera, degut a que el *layout* té el tag *style*, el qual com es pot observar en el *Diagrama 10*, segons el tema que es vulgui incorporar el l'*Activity*, carregarà uns atributs o uns altres en les *Views*.

Així doncs, es fa servir un mètode propi *(onActivityCreateSetTheme (Activity , theme))* on segons el valor de la variable *theme* que està guardada a preferències, es cridarà a la funció d'Android *setTheme(Theme)* fent que es carregui el tema adequat. .

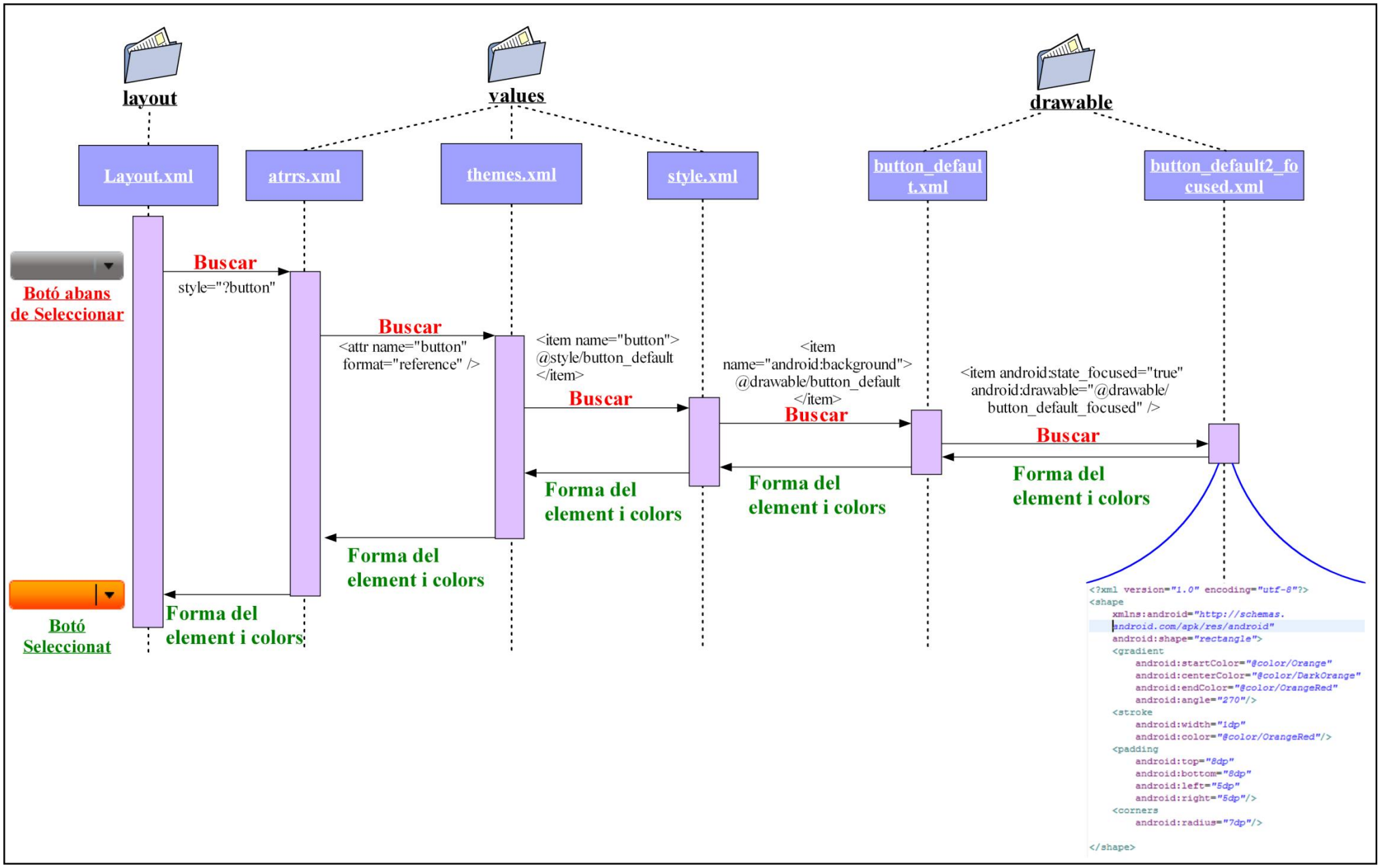


Diagrama 10 – Diagrama de seqüències dels temes i estils en els elements dels layouts.

6.2.3 PERSONALITZACIÓ DE LES VIEWS PER A RESOLDRE PROBLEMES

Android proporciona una sèrie d'eines bàsiques, funcions i atributs per tal de personalitzar les *Views*. Aquestes eines són funcionals fins a un cert punt, degut a que amb les *Views* es poden manejar dintre de les possibilitats bàsiques. En cas de fer un disseny i adaptar-lo al funcionament d'*Android* pot arribar en un punt en que les eines bàsiques no serveixen i cal implementar nous mètodes. Un dels exemples més visibles són els *dialog* (diàlegs, missatges emergents), el quals no se'ls hi pot aplicar un estil propi si s'utilitza el *layout* per defecte i les funcions proporcionades per la classe.

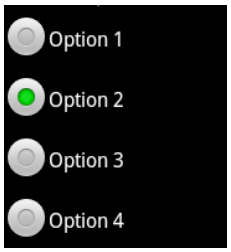
En aquest cas, s'ha optat per implementar una nova classe que estén del *dialog* (*MyCustomDialog*). Així s'implementa un *layout* propi per al *dialog*, per tal de posar colors, imatges, botons i distribuir el *layout* amb el disseny que es vulgui.

Imatge 15 - Botó amb error de visualització

Un altre problema que ve donat per *Android* i en les pantalles petites i de poca densitat, és amb el dibuixat dels botons. En aquest cas, la imatge es distorsiona lleugerament, creant que la línia de contorn del botó no sigui uniforma i faci algun escalat com es pot veure en la *Imatge 15.* Es va detectar que els botons dibuixats personalment pel programador en els *xmls*, no contenien aquest problema. Així doncs, per tal treure aquest error, tots els botons s'han dissenyat amb xmls, fins i tot el tema *default*, el qual en un principi es pensava deixar amb les Views dibuixades pròpiament per Android. En la *Imatge 16* es pot veure un botó dibuixat correctament a partir d'un fitxer xml.

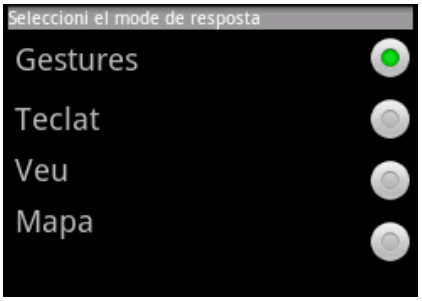
Imatge 16 - Botó amb una correcte visualització

En les pantalles de configuració, es va pensar d'introduir *radiogruops* i *checkboxs* per a seleccionar diversos estats d'una propietat de preferències. Els *radiogruops* que venen per defecte amb *Android* situen els elements de selecció a l'esquerra i el text a la dreta, fent que no hi hagués altre manera d'intercanviar les posicions. En la *Imatge 17* es pot veure com quedaria alineat un *radiogroup* per defecte.



Imatge 17 - Radiogruop per defecte

Per tal de poder seguir amb el disseny proposat, es va dissenyar el *layout* amb *textviews* independents per posar el text i amb *radiogruops* sense omplir el camp de text. Per tal de que quedés ben organitzat el *ViewGruop*, es va anar jugant amb els *LinearLayout*, els *RelativeLayout* i les diverses formes de posicionament, *paddings*, *margins*... El resultat d'aquesta primera implementació es pot observar en la *Imatge 18*, en el qual s'hi pot apreciar que el conjunt de 4 *radiogroup* és més alt que el conjunt de les 4 cadenes de text. A més a més, tenia un problema afegit, ja que al seleccionar al text no permetia seleccionar l'opció. Només es podia seleccionar al *radiogruop* per seleccionar l'opció, fet que feia molt poc funcional el mètode emprat per a la primera implementació.



Imatge 18 - Radiogroup, primera implementació

Imatge 19 - Implementació de ListViews amb radiogroups

Per tal de solucionar aquest petit problema, es va passar a mirar de crear *ListViews*, els quals continguessin els *radiogrups* i es poguessin posar a la dreta. Tan sols cal associar el *layout* adequat amb el disseny dels elements que van a cada fila del *ListView*. Aquest *layout*, pot ser propi o el que ve per defecte d'*Android* i s'associa al crear el *array* del adaptador. En el cas de l'aplicació, s'ha fet una copia del *layout* per defecte d'*Android*, i se n'han fet petites modificacions per ajustar-ho al disseny. Aquesta solució, permet visualitzar les files de forma que les *Views* estan perfectament centrades, així com clicar en qualsevol punt de la fila per a seleccionar l'opció. Tal i com es pot observar en la *Imatge 19,* també es pot aplicar els estils definits en els temes.

Un dels últims problemes trobats al voler aplicar el disseny de l'aplicació, ha estat a l'hora d'incloure més d'un *ListView* en una sola *Activity*. Segons els desenvolupadors del sistema *Android*, és una disseny que no es pot fer, degut a que un *ListView* està preparat per a una sola *Activity*. Davant d'aquest problema, i la negativa per part dels desenvolupadors de trobar una solució generalitzada per a tothom, s'ha buscat altres mitjans per a solucionar-ho.

El problema rau en que un *ListView* té un *ScrollView* associat per tal de poder moure's pel llistat, el qual és més gran que l'alçada de la pantalla. En cas de tenir dos *ListViews* en una pantalla, no se sabria fer bé el *ScrollView* que conté cada llistat. A més a més, no s'expandeix bé els llistats i no queda ni funcional ni estètic com es pot comprovar en la *Imatge 20*.

La solució més senzilla seria que hi hagués un sol *ScrollView* per les *Listview* que hi hagin en una *Activity*, per fer això, caldria expandir els *ListViews* a la seva màxima alçada per a que fossin visibles. A més a més, caldria posar cada llista una sota l'altre i saber-ne quina altura tenen totes juntes per poder-ho associar amb el *ScrollView*. Per tal de dur a terme aquesta solució, s'ha implementat la funció *setListViewHeight(ListView)*. Aquesta funció recorre cada element (fila) d'un ListView agafant la seva alçada i sumant-les per a obtenir el total. Un cop sap l'alçada total carrega amb la funció *setLayoutParams(params)* la nova alçada del *ListView*. Així doncs, s'aconsegueix expandir el *Listview* a la seva màxima alçada i només resta posar un *ScrollView* en el *Layout* de l'*Activity* per tal de poder recórrer les llistes existents de la pantalla si la suma d'elles supera l'alçada de la pantalla.

Imatge 20 - Visualització incorrecte de dos ListViews

Aquesta solució té un petit problema d'alt cost de computació a l'haver de recórrer cada element de la llista. Així doncs, es recomana no fer-la servir per a llistes llargues per tal de no sobrecarregar amb lentitud l'aplicació.

Tot i això, en l'aplicació s'ha fet servir aquesta funció, degut a que les llistes no tenen més de 5 files i per tant, expandir 2 llistes que no tenen més de 10 files es realitza ràpidament i no sobrecarrega l'aplicació. En la *Imatge 21* es pot veure, aquesta solució és la millor manera de tenir petites llistes en una sola *Activity* i no haver de realitzar el triple d'*Activities* per a realitzar la configuració d'un apartat, com ara el de so.

Imatge 21 - Activity amb 3 Listviews

6.2.4 DIBUIXAR ELEMENTS

Hi han diverses maneres de dibuixar els elements del *layout* en la carpeta *drawable*. Es pot fer mitjançant dibuixar-lo en el xml, i per tant, mitjançant codi, o bé es pot fer mitjançant imatges amb el format ".9.png"

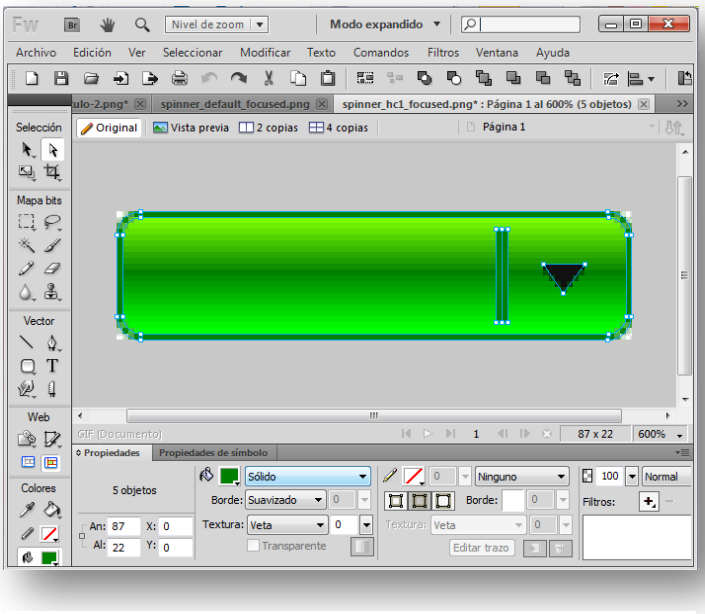
Per tal de dibuixar els elements per *xml*, cal fer servir els tags de *shape*, *gradient*, *stroke*, *padding* i *corners*.

El tag *shape* serveix per indicar quin tipus de polígon es vol dibuixar entre els 4 possibles: ["rectangle" | "oval" | "line" | "ring"]. El gradient serveix per omplir el polígon amb un color o amb un degradat de colors. El *stroke* és la línia de contorn del polígon, se li pot indicar el color i el grossor. El *padding* per a l'espai de text de l'element. I els corners, són les cantonades del rectangle, per tal de fer-los més arrodonits.

Amb aquests tags podem dibuixar mitjançant programació els botons, línies, cercles, etc. Tan sols amb la combinació correcte es poden crear els elements que es vulguin.

Per tal d'indicar els codis hexadecimals dels colors als gradient, la millor manera és fer referència als colors predefinits d'*Android* o bé crear un fitxer anomenat *colors.xml* en la carpeta de *values*. En aquest fitxer es poden llistar tots els colors que es volen, indicant un nom i el seu valor en hexadecimal. D'aquesta manera, es pot fer servir directament tots els colors que es vulguin, mitjançant la comanda *"@color/Nomcolor*", definint un únic cop el seu codi hexadecimal.

Una altre manera es crea els elements mitjançant imatges editades en programes com el *Gimp*, el *Fireworks* o el *Photoshop*. El procediment consisteix en trobar plantilles d'elements d'*Android*, per exemple el *spinner*. En aquest cas, s'ha fet servir el *Fireworks*, on es pot trobar que té una sèrie de plantilles anomenades *Màscares de flex*. Mitjançant aquesta aplicació, s'edita la plantilla, se li dóna la forma i els colors desitjats i es guarda la imatge en format ".png" com es pot veure en la *Imatge 22.*



Imatge 22 – Disseny d'un spinner amb el programa Fireworks CS5.

Una vegada es té la imatge creada, cal fer servir l'eina *Draw 9-patch* que permet crear la imatge en format ".9.png". utilitzant un editor *WYSIWYG*.

Aquest tipus de format, especial per *Android*, permet indicar a la imatge com s'ha de manipular per tal de que no es deformi i comprovar si la imatge es valida per a utilitzar. Per tal d'indicar a la imatge com ha de ser manipulada, cal introduir unes línies en el contorn de la imatge.

Les línies introduïdes al costat esquerra i al costat superior, indica quina és l'àrea que *Android* pot escalar en cas de necessitar més espai per a l'element, com per exemple que hi càpiga el text del botó dins la imatge.

En la *Figura 9*, es pot veure com una imatge de 48x48 píxels amb les indicacions correctes es pot escalar al doble sense deformar-se.

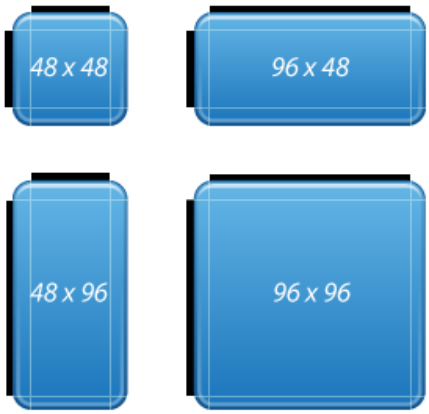


Figura 9 – Escalat d'una imatge 9-patch

També cal tenir en compte que les imatges sempre s'escalen d'una mida petita cap a la gran. Així doncs, les imatges a dibuixar han de ser petites, així es podran escalar correctament i tenir-ne de petites i de grans.

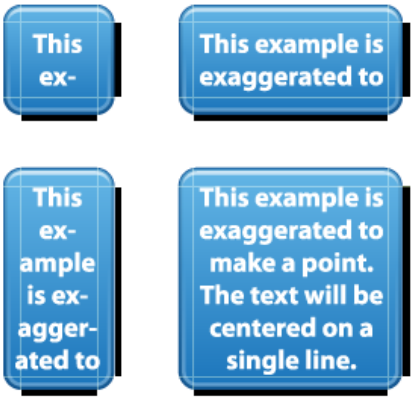


Figura 10 – Àrea de contingut de Text de l'element

En canvi, les línies introduïdes al costat dret i al costat inferior, indica quina és l'àrea en què es pot posar el text dins de la imatge per tal de que quadri adequadament.

En la *Figura 10*, es pot veure com una imatge de 48x48 píxels amb les indicacions de l'àrea de contingut on el text s'ajusta a ell fent els salts de línia corresponents.

Aquest tipus de format d'imatges, no és únicament per a botons, pot servir per a més ocasions, com ara fons de pantalla, i per quan es necessiti fer qualsevol imatge per a una aplicació *Android*.

L'eina *Draw 9-patch*, és un editor molt senzill, del qual es té la imatge com a element principal per editar. Tan sols cal passar el cursor per indicar les línies als costats de les imatges i en tot moment es

Imatge 23 – Eina Draw 9-patch amb les indicacions de format de la imatge

pot veure una previsualització al costat dret. Aquesta previsualització serveix per veure si la imatge s'escalarà correctament. En la *Imatge 23*, es pot veure una vista de l'eina que s'indica.

Una vegada guardada la imatge, es pot posar en la carpeta *drawable* i utilitzar-la sense problemes. Per tal d'utilitzar-la, cal fer-ne la referència en el *xml* on es vulgui fer servir (*@drawable/imatge*).

6.2.5 PERSONALITZACIÓ ESTÀTICA-DINÀMICA

La personalització estàtica-dinàmica és un entremig del que s'ha estat veient. En aquest cas, es refereix als colors de selecció de les llistes de la interfície 2 (*Imatge 24*).

Aquestes llistes són uns menús completament dinàmics, formats per un *Tablelayout* i per *Textviews*. Com no tenen el comportament de botons i és un menú creat i personalitzat per a l'aplicació, la selecció dels elements s'ha fet dinàmica.

Aquest comportament és així degut a que a partir del progrés de la barra lateral es selecciona una posició del menú.

Per tal de personalitzar degudament aquests perfils, tenint en compte els 5 temes creats en la interfície 1, s'ha hagut de crear dos mètodes comuns a totes les *Activities*.

Quan es carrega el menú, es fa servir el mètode *ChangeColor.setTextTable(textview, preference),* que consisteix en indicar de quin color és el text del menú depenent de quin tema hi hagi guardat a les preferències de l'aplicació.

Imatge 24 – Interfície 2 de Onthebus

En quan l'usuari interactua amb la barra lateral, hi ha un mètode per tal de comprovar en quina posició esta situat el cursor de la barra. Dins d'aquest mètode, es fa servir el mètode *ChangeColor.setTableText(Tablelayout, preference, posició_anterior, posició _actual).* Seguint amb la metodologia anterior, a partir del tema que hi hagi guardat a les preferències de l'aplicació, en la posició actual del menú se li aplica un color de fons i un de text a l'element *TextView* i se li indica que està seleccionat per tal de poder-li aplicar el moviment horitzontal al text.

Tenint en compte que l'exigència de la personalització d'una aplicació, avui dia pot ser equivalent a la de tenir un programa funcional, es creu que amb la personalització estàtica i la dinàmica compleixen els requisits que els usuaris puguin tenir en front a aquest tema. Poden elegir diversos temes i si amb aquests no se'n té prou, es poden personalitzar alguns elements.

6.2.6 DISSENY EQUITATIU ENTRE INTERFÍCIES

Tal i com s'ha pogut veure en el disseny, es tenen dues interfícies per tal que l'aplicació sigui accessible al 100%. Per tal que les dues interfícies siguin equitatives, s'han hagut de dissenyar funcionalitats equitatives d'algunes *Views* d'*Android*.

Com s'ha explicat en l'apartat 4.2 Disseny Definitiu, la interfície 2 està constituïda amb una sèrie de menús accessibles per una barra lateral. Com que la barra es desplaça segons l'alçada del terminal, s'ha agafat el terminal més petit i s'ha estudiat quantes files hi poden cabre en la pantalla per a que sigui fàcilment utilitzable i còmode per passar entre elles. S'ha arribat a la conclusió que unes set files serien suficients per tal de complir aquest requisit.

Aquesta limitació, la qual amb els menús d'opcions n'hi ha prou, no és suficient en el cas de voler una llista llarga, com ara el llistat de contactes, els favorits, els llistats de ciutats, el llistat de companyies,etc...

Per tal de solucionar aquesta limitació, s'ha hagut de dissenyar les llistes en funció del màxim indicat. Per tal de poder treballar amb comoditat s'ha elegit que el màxim de files per el llistat en el menú serien sis files, per tal de treballar amb nombres parells. D'altre banda, s'ha de tenir en compte que la llista pot contenir diversos estats, o sigui, pot ser més petita de sis files, pot tenir entre sis i dotze o bé pot ser més gran de dotze. La idea és partir els llistat en mini llistats de sis files, els quals en direm submenús. Això implica que hi haurien tres tipus de submenús en la llista depenent si cal moure's endavant o enrere en ella. El diagrama pretén il·lustrar la idea que s'ha tingut per a resoldré aquest problema de disseny.

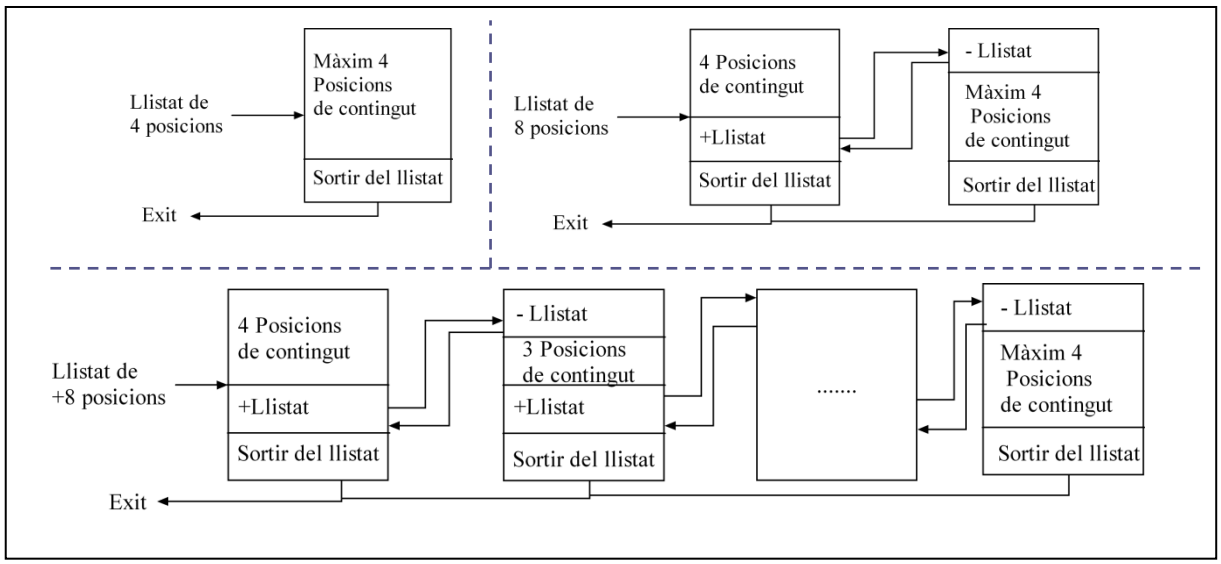


Diagrama 11 - Diagrames dels possibles submenús segons el nombre de elements del llistat

Amb aquesta idea s'ha implementant una funció *loadList()* la qual amb una sèrie de condicionals mira en quin dels estats s'ajusta la llista. Llavors es construeix una nova llista on s'inclouen les opcions d'avançar, retrocedir i sortir del llistat, la qual servirà per carrega en el menú. D'aquesta manera es treballarà en una única llista que ja està muntada i serà fàcil d'anar carregant en la mateixa pantalla.

L'usuari s'anirà movent pels elements i al seleccionar una opció es comprova si es tracta de sortir, d'avançar i retrocedir. Aquesta comparació, la qual pot semblar una pràctica poc funcional, es fa amb *strings*, degut a que el llistat completament dinàmic i no es pot saber amb anterioritat quan ocuparà. Però, cal tenir en compte que sempre es fa amb la cadena de *strings* obtinguda directament

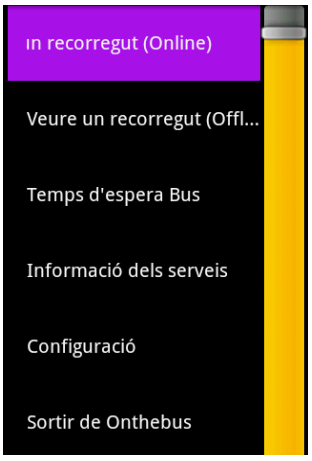
de recursos, tant la que s'afegeix al llistat com amb la que pretenem comparar. Per tant, les cadenes són estàtiques i se'n pot fer la comprovació tranquil·lament, ja que si coincideixen s'entrarà en l'opció.

A continuació es pot comprovar com és el disseny de la llista en les dues interfícies, en la *Imatge 25*.

Imatge 25 - Llistat en cada interfície

D'altra banda, hi ha un problema afegit de visualització del text en una fila, degut a que no podem fer salts de línia en el text fent que les files siguin equitatives. En aquest cas, s'ha optat per un text d'una sola línia encara que excedeixi de l'amplada de la pantalla. Així doncs no es veuria tot el contingut de la fila i per tant quedaria poc accessible si algun usuari en vol llegir el contingut en comptes d'escoltar-lo.

Per tal de resoldre aquest problema de visualització s'ha buscat la forma més òptima per inserir dues funcionalitats a la *View* de *TextView*, els quals consisteix en truncar el text que surti de la pantalla amb la funció *setEllipsize(TruncateAt.END).* D'aquesta manera apareix el text i els punts suspensius donant a entendre que hi ha més text. Per tal de complementar aquesta funcionalitat, se n'ha afegit una segona de manera que quan es selecciona una opció del menú el text faci un *scroll*



Imatge 26 - Menú principal amb el MARQUEE

horitzontal del seu contingut per a que sigui llegit juntament fent servir les funcions *setEllipsize(TruncateAt.MARQUEE)* i *setHorizontallyScrolling(true).* En la *Imatge 26* es pot veure com la primera opció s'ha mogut ja que es veu el final de la frase, i com la segona opció té el text truncat amb els punts suspensius.

6.3 Idiomes

Avui en dia tenir una aplicació en un sol idioma no n'hi ha prou. Cal pensar que un usuari pot ser de qualsevol país i per tant, tenir l'aplicació en un sol idioma pot suposar una barrera.

Es podria fer l'aplicació en Anglès, que és el més estès i en el que solen estar la majoria de programes, però això suposaria un disseny poc universal, degut a que potser hi ha usuaris que no el saben i no es defensen prou bé amb ell.

Una aplicació, si està amb el màxim d'idiomes possibles, ofereix un mercat més ampli i una millor acceptació per part dels usuaris, els quals no acabarien dient el clàssic "No està en Català?".

Android ofereix un bon potencial per a que l'aplicació pugui ser multi idioma amb molta facilitat. Per tal de canviar l'idioma del terminal, tan sols cal anar a "*ajustes->Idioma y teclado*". Depenent de la versió d'*Android* i de la companyia telefònica on s'hagi adquirit el terminal o del fabricant, els idiomes que el terminal puguin contenir.

Per exemple, es pot trobar el terminal *Huawey Sonic,* el qual només conté els idiomes Anglès i castellà, tot i tenir la versió 2.3.3 de *Android*, la qual conté una llarga llista d'idiomes. En canvi, el LG-P500 amb la versió d'*Android 2.2*, conté l'Anglès, el Català, el Castellà, Euskera, Gallec, Francès, Alemany, Italià i Portuguès Així doncs, tot dependrà de quina compilació del sistema operatiu s'ha instal·lat en el terminal.

Per tal de poder traduir una aplicació de forma paral·lela, cal que totes les cadenes de text estiguin en el fitxer *strings.xml*. Aquest fitxer conté un llistat de tots els textos utilitzats per a l'aplicació, evitant així, posar les cadenes de text directament al codi. En cas de posar el text directament al codi, es podrien tenir problemes per a canviar-lo degut a que s'hauria de revisar les línies de codi fins a trobar-lo i tenir molts problemes per a traduir el sistema de l'aplicació complet.

El fitxer *strings.xml* es troba en la carpeta */res/values/.* També es pot fer servir el fitxer *arrays.xml* en cas de necessitar *arrays* de text en l'aplicació, el qual també es troba en la carpeta *values*.

El fitxer *strings* conté una estructura "camp-valor" per cada cadena de text, la qual es pot veure en el *Codi 8*.

```
<string name="NomDelString">OnTheBus</string>
```

Codi 8 - Estructura de la cadena de text en strings.xml

En canvi, el fitxer *arrays* conté una estructura "camp-llistat de valors" per cada *array* de cadenes de text, la qual es pot veure en el *Codi 9.*

```
<string-array name="NomDelArray">
      <item>Iniciar un recorrido (En línea)</item>
      <item>Ver un recorrido (Sin conexión)</item>
</string-array>
```

Codi 9 - Estructura dels arrays de cadenes de text en arrays.xml

Aquests fitxers externs, es pot accedir amb facilitat des del codi amb els mètodes proporcionats per la classe *Resource*. O bé amb l'atribut text en el element corresponent del *Layout* de la *Activity*. En el Codi 10 que hi ha a continuació se'n pot veure un exemple.

```
getResources().getString(R.string.NomDelString));
getResources().getStringArray(R.array.NomDelArray);
android:text="@string/NomDelString"
```

Codi 10 - Mètodes per accedir als recursos del fitxer strings.xml

D'aquesta manera, tindrem totes les cadenes de text en la carpeta */res/values/* i en un sol idioma. Un cop arribats a aquest punt, caldrà fer l'aplicació multi-idioma amb un senzill pas i tenint en compte els codis de representació dels noms dels idiomes.

Si es vol un nou idioma, tan sols cal agregar una nova carpeta */res/values/* afegint en el nom de la carpeta */res/values/,* el nom de la representació de l'idioma. Per exemple, si es vol el Francès,

caldrà afegir la carpeta */res/values-fr/* i dins de la carpeta, cal que hi contingui els fitxers *strings* i *arrays* amb els valors de les cadenes traduïdes en el idioma corresponent. Es pot veure un exemple del sistema de carpetes en la *Figura 11*.

Respecte al canvi d'idioma, aquest el fa automàticament el sistema *Android*. *Android* consulta quin és el idioma del terminal i seguidament mira en el sistema de carpetes de l'aplicació si hi ha una carpeta */res/values-idioma/* expressament per a l'idioma per tal d'anar a buscar el valor de les cadenes de text en ell. En cas de que no trobi l'idioma indicat, es farà servir el que hi hagi per defecte */res/values/.*

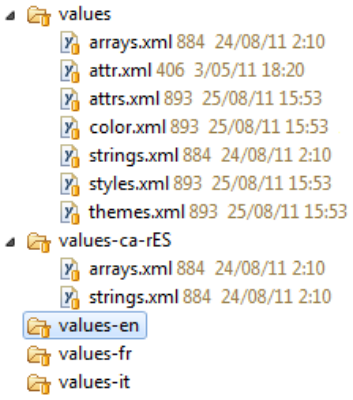


Figura 11 – Sistema de carpetes d'Android per als idiomes

Per tal de traduir el fitxer *strings.xml* del Castellà/Català a l'Anglès i a l'Italià es va proposar que fós accessible per qualsevol usuari que no tingués coneixements sobre els *xmls*. En una primera instancia, es va debatre si s'hauria de fer un *parser* (eina per manipular els fixers xml*)* exprés pels fitxers *strings.xml* i *arrays.xml*, del qual es va arribar a la conclusió de fer servir el *kXml 2* de *java* per a implementar un petit *parser*.

Com no cal inventar la roda, si ja està feta, es varen buscar si ja existien *parsers* o aplicacions que fossin capaços de fer la traducció del *xml* a taules. La decisió de fer aquesta recerca ve donada per l'increment d'usuaris i programadors en *Android*, els quals segurament es deurien trobar amb el mateix problema.

Es va trobar una primera solució ben fàcil, la qual presentava un petit problema. Els fitxers *xmls* es poden obrir en el Excel, el qual fa la traducció del que troba en ells a una taula que representa tota la informació. Els dos fitxers que tenim es poden obrir sense problemes, però, només el fitxer *strings.xml* és el que es pot modificar i guardar conservant el format original.

Així doncs, es va seguir buscant la manera de traduir el fitxer *arrays.xml* fins a trobar una solució que s'ha utilitzat en l'aplicació. En els fitxers *xml* es poden fer referències a altres fitxers per tal d'obtenir un recurs en concret, de la mateixa manera que podem fer servir un *string* en concret en el *layout* d'una pantalla. Per tant, s'ha decidit passar tots els *strings* que contenen els *arrays* al fitxer *strings.xml* i seguidament posar la referència del *string* en el fitxer *arrays*. En el *Codi 11* es pot

```
<!-- Principal.java -->
<string name="MenuPrincipal1">Iniciar un recorregut (En línea)</string>
<string name="MenuPrincipal2">Veure un recorregut (Sense connexió)</string>
<string name="MenuPrincipal3">Temps d\'espera Bus </string>
<string name="MenuPrincipal4">Informació dels serveis </string>
```

Codi 11 - Strings del fitxer strings.xml

Aquesta solució, la qual és la més adequada, permet tenir totes les cadenes de text en el mateix fitxer, poder obrir el *xml* en el *Excel* per tal de canviar el valor de les cadenes a l'idioma que es desitgi i tornar-lo a guardar en el format que demana *Android*. Com es pot utilitzar el Excel, és la millor interfície per tal de que un usuari que no sàpiga programació pugui traduir les cadenes de text d'un fitxer *xml* de *strings* d'*Android* sense problemes .

```
<!-- Principal.java -->
<string-array name="MenuPrincipal">
      <item>@string/MenuPrincipal1</item>
      <item>@string/MenuPrincipal2</item>
      <item>@string/MenuPrincipal3</item>
      <item>@string/MenuPrincipal4</item>
</string-array>
```

Codi 12 - Referències dels strings en el fitxer arrays.xml

6.4 Favorits i Contactes

Avui en dia, quantes més funcionalitats tingui una aplicació relacionades amb les característiques del programa, fa que en sigui més atractiva la seva utilització. En el cas de l'aplicació *OnTheBus,* la qual fa servir direccions postals, es pot veure clarament que una bona funcionalitat a incorporar és poder guardar aquestes direccions en algun registre propi i posteriorment poder-los recuperar en futures rutes com a possibles destins.

Vist que calia guardar aquestes dades d'alguna manera es varen fer dos propostes, una de les quals requeria fer servir una petita base de dades on es guardes una adreça localitzada amb un alies, el qual l'usuari hauria d'introduir. La segona proposta era fer servir l'agenda de contactes que ofereix el terminal *Android*, la qual permetria utilitzar les adreces de coneguts, amic, familiars,... de l'usuari com a destí d'una ruta o bé, en cas de no tenir-ne la direcció, incloure-la després en el contacte existent.

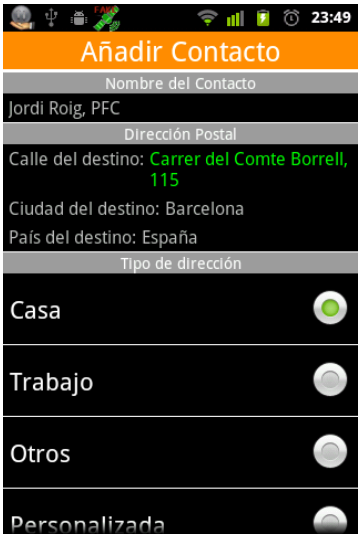
Les propostes realitzades es varen anar definint fins al resultat obtingut en l'aplicació, el qual hi ha implementades aquestes dues funcionalitats.

Imatge 27 - Llistat de contactes de l'agenda

L'usuari en entrar en el mode de guiatge pot seleccionar una adreça de les que existeixin en la seva agenda. L'agenda de contactes pot contenir molts contactes, els quals no tinguin adreça, degut a que és l'usuari qui les ha d'introduir a l'hora de crear-lo. Com que aquests contactes no fan cap servei en l'aplicació, no es mostren en el llistat de contactes, del llistat que arriba de l'aplicació *Android* s'ha fet un filtratge, el qual només es deixa que es mostrin aquells contactes que tenen una direcció postal en la ciutat on es vol fer el guiatge. El motiu el qual cal un filtratge segons la ciutat, és per tal de garantir que es trobi un recorregut vàlid amb bus en les ciutats en les quals s'ofereix el servei. L'usuari pot seleccionar el contacte i prosseguir a realitzar la cerca d'un recorregut. Es pot veure en la *Imatge 27*, com l'usuari visualitza els seus contactes.

També existeix la inversa a l'acció acabada de descriure. Durant el guiatge i a l'acabar el guiatge, s'ofereix la possibilitat de guardar una adreça en un dels registres de l'agenda.

En aquest cas no es fa cap tipus de filtrat en el llistat de contactes, degut a que es pot inserir l'adreça en un contacte que no en tingui o bé inserir l'adreça com a segona adreça en un contacte que en tingui. En el segon cas, *Android* permet tenir forces adreces relacionades en un contacte, i per tant poden tenir una etiqueta per tal de localitzar quina és quina. En el cas de l'aplicació s'ha dissenyat per a que també es pugui indicar de quin tipus de direcció postal es tracta, ja sigui *Home, Work, Other* o personalitzada per l'usuari i guardar-ho. Per tal de il·lustrar quines dades es guardaran en el registre del contacte, es pot veure la *Imatge 28*.



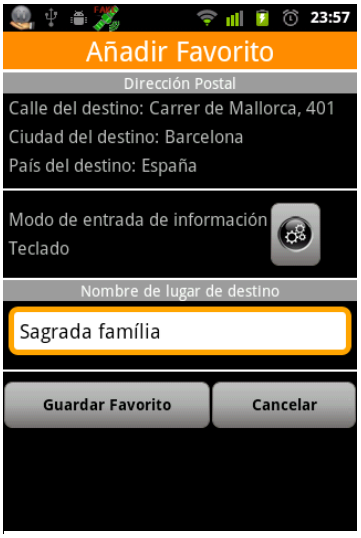
Imatge 28 - Afegir un contacte en l'Agenda

Per una altre banda, en cas de que l'usuari no vulgui fer servir l'agenda, degut a que la direcció que vulgui guardar no concorda amb un contacte sinó amb un lloc conegut com ara un monument, o una empresa, o una botiga,... llavors s'ofereix la funcionalitat de guardar-ho en una base de dades

pròpia de l'aplicació. En aquest cas, el disseny consisteix en mostrar també el llistat de favorits, en el cas d'introduir una adreça de destí, i de permetre guardar una localització durant el guiatge.

Per tal de que l'usuari sàpiga el motiu per el qual ha guardat un favorit, aquest es guarda sota el nom d'un alies que ell mateix podrà introduir quan el vulgui guardar.

L'usuari pot introduir malament el nom del alies, o bé pot voler esborrar un registre en concret o tots. Com existeix aquest marge d'error i és elemental poder modificar les dades que el propi usuari introdueix, s'ha afegit un apartat en la configuració de l'aplicació per tal de que ho pugui modificar o bé eliminar. En la *Imatge 29* es pot veure la interfície per aquesta funcionalitat.



Imatge 29 - Afegint un favorit d'un monument emblemàtic

Amb aquestes dues funcionalitats queda coberta la necessitat de guardar direccions en qualsevol moment del recorregut i fer-les servir en qualsevol moment, al gust que l'usuari. És molt pràctic, degut a que s'estalvia el temps d'introduir el nom complert del carrer cada cop que es vol fer una ruta.

7 Aprofundint en l'aplicació

Funcionalment, l'aplicació es divideix en tres parts corresponents al guiatge, a la configuració de l'aplicació i a la informació sobre els serveis.

En aquest apartat s'explorarà la part de la configuració i la part de la informació sobre els serveis, les quals tenen el mateix funcionament en les dues interfícies. L'únic punt de diferenciació és el nivell de profunditat dels menús per a poder accedir-hi.

També es podrà veure el funcionament del guiatge per a la Interfície 2 i com s'ha adaptat la informació del mapa al sistema de menús.

7.1 Configuració

L'apartat de configuració té com a objectiu guardar les preferències de l'usuari enfront l'aplicació. Aquestes preferències s'han desenvolupat seguint el procediment explicat en l'apartat *6.1 "Preferències",* i es basen en un sèrie de menús els qual les categoritzen. A part de guardar les preferències, també serveix per tenir accés per a modificar els registres de favorits i a l'historial.

En el *Diagrama 12* es pot veure com l'apartat de configuració es divideix en sis categories. La divisió de la configuració en aquestes categories, ve donada per facilitar a l'usuari quines preferències podrà modificar en un sol cop d'ull.

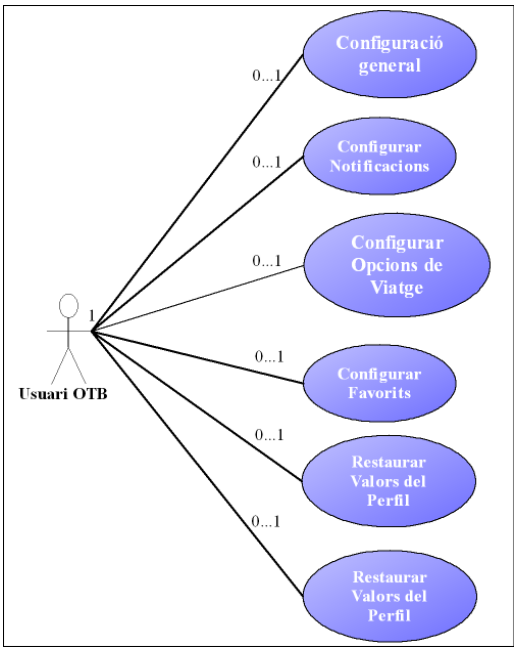


Diagrama 12 – Menú de Configuració

En la categoria de configuració general s'abarca el tema de personalització de l'aplicació, tal i com es pot apreciar en el *Diagrama 14*, amb tota classe d'opcions per a canviar els colors dels elements de l'aplicació, tal i com s'ha explicat en l'apartat *6.2 "Personalització del OnTheBus"*.

També s'ha inclòs una configuració avançada per tal d'eliminar les dades que genera l'aplicació sobre els destins de les rutes. Creiem, que en algun moment aquests registres poden ser molt elevats si els usuaris utilitzen l'aplicació per als desplaçaments quotidians, o bé professionals. Per tant, els

registres passarien a ocupar una part petita de la memòria del terminal, la qual es possible que els

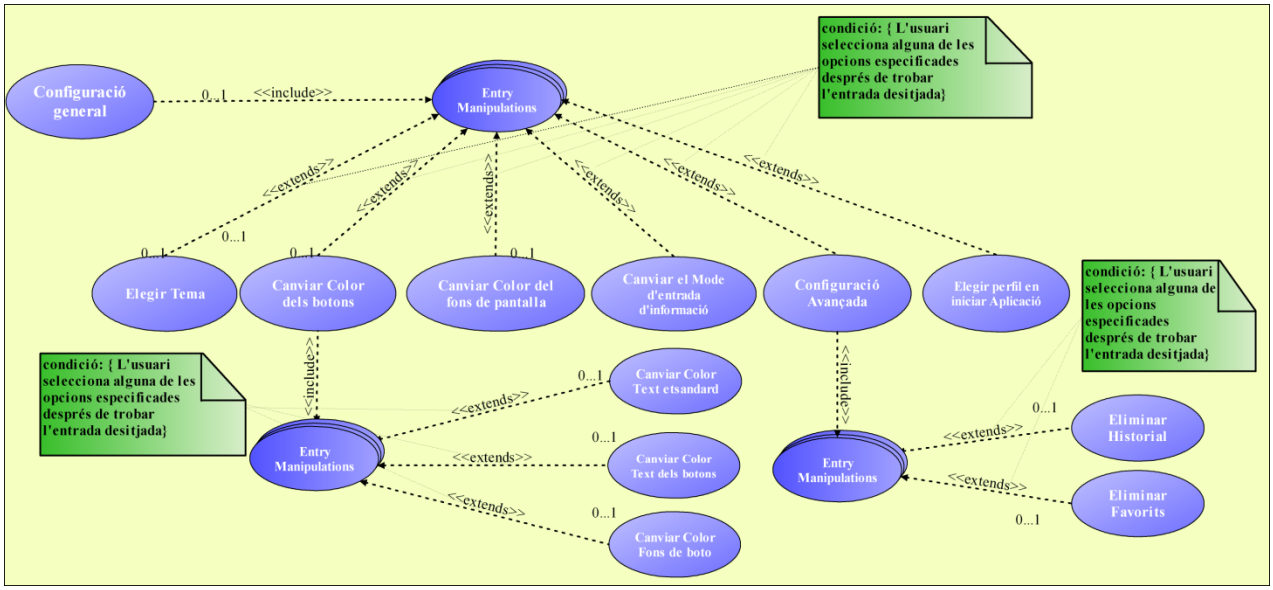


Diagrama 14 – Menú de Configuració General

usuaris prefereixin anar esborrant per a buidar-la.

En la categoria de configurar notificacions s'abarca el tema d'accessibilitat de l'aplicació, tal i com es pot apreciar en el *Diagrama 13*, amb tota classe d'opcions per a poder activar la veu, modificar el volum, incorporar vibracions i indicar si l'usuari ja té un nivell de novell o avançat del funcionament de l'aplicació.

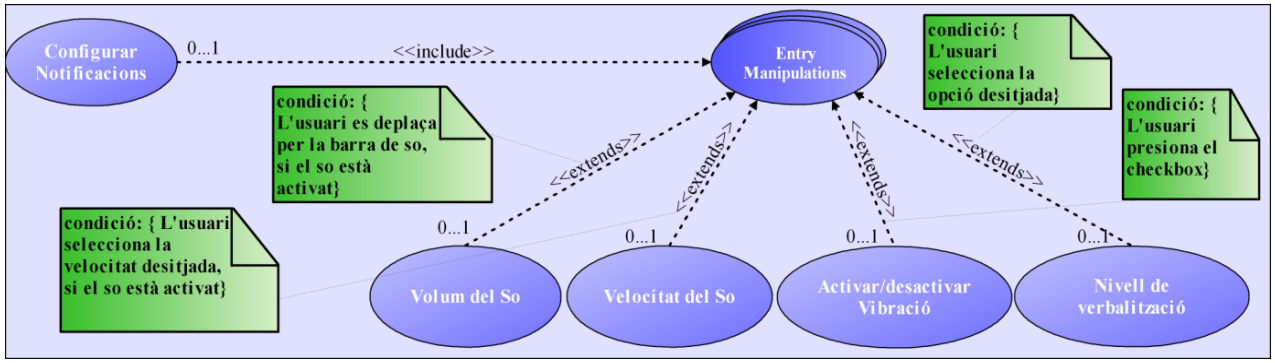


Diagrama 13 – Menú de configuració de les notificacions

Aquesta categoria és important degut a que sinó es pot activar el so o les vibracions, l'usuari es pot distreure del guiatge del recorregut i no assabentar-se de les indicacions. També és important per tal de que l'aplicació sigui al màxim d'accessible per als usuaris amb limitacions funcionals visuals

i auditives, degut a que els hi cal aquests tipus d'avís per a poder desenvolupar-se en l'ambient del programa sense dificultats.

En la categoria de configurar les opcions de viatge s'abarca el tema del guiatge en el recorregut de l'aplicació, tal i com es pot apreciar en el *Diagrama 15*, amb tota classe d'opcions per tal d'indicar com ha de cercar l'aplicació la ruta i com es vol que sigui el seu guiatge.

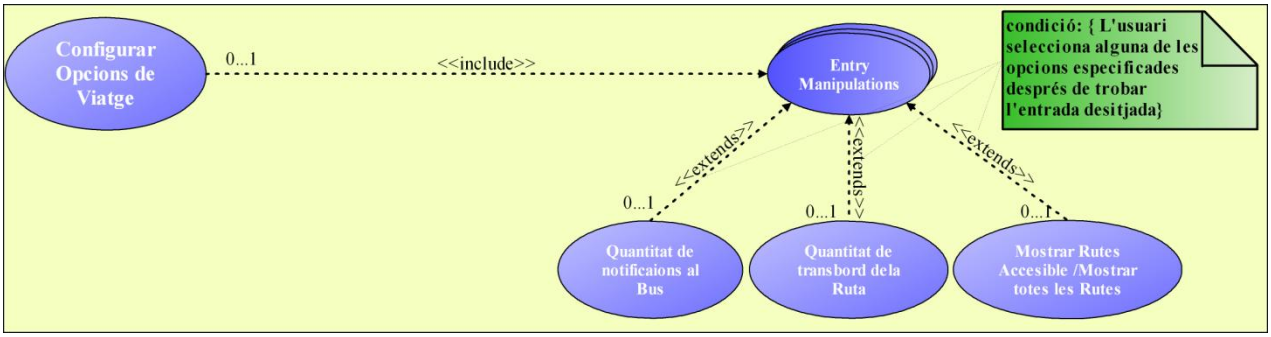


Diagrama 15 – Menú de configuració de les opcions de viatge

Per tal de cercar una ruta, la qual es podria fer agafant un bus o bé múltiples busos, cal saber quina és aquesta prioritat per a l'usuari. Tenint en compte la quantitat de línies de bus existents, la majoria de recorreguts es poden fer amb un sol bus i com a molt amb dos busos i per tant, amb un sol transbordament.

En la categoria de configurar favorits es pot visualitzar una llista de tots els registres de favorits, els quals es podrà fer dues accions, modificar i esborrar, per cada registre de favorits, tal i com es pot apreciar en el *Diagrama 16*.

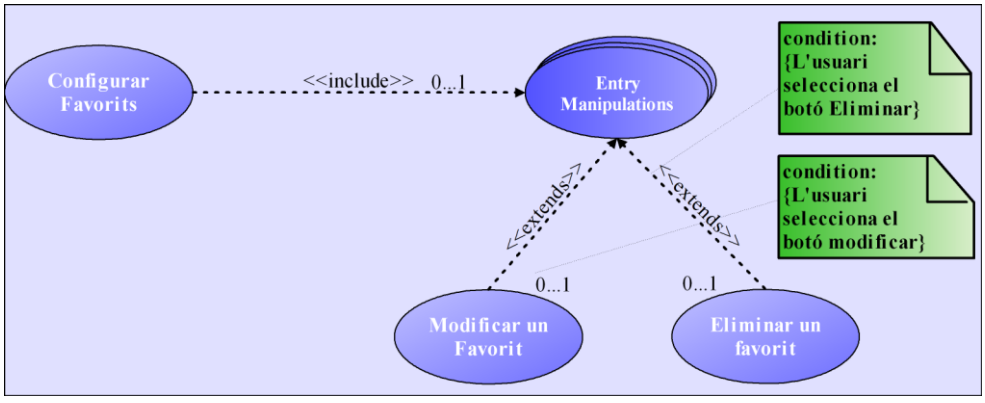


Diagrama 16 – Menú de configuració dels favorits

Prenent com a punt d'inici, que els favorits es guarden durant i a l'acabar el guiatge, se suposa que en la inserció d'ells i poden haver un percentatge d'errors. Aquests errors poden ser deguts a inserir malament el nom del favorit, o per haver-ho fet per equivocació. És per això, que s'ha inclòs aquesta funcionalitat, la qual permet modificar el nom del favorit o bé esborrar-lo del llistat.

Per acabar amb l'apartat de configuració, s'ha inclòs les categories per a restaurar els valors del perfil o bé de l'aplicació. Així doncs, en el cas que l'usuari hagi modificat les preferències de tal manera que no li vagin bé, sempre es podrà recorre a aquestes opcions per a desfer els seus canvis i tenir els valors de l'aplicació com si estigués recent instal·lada.

7.2 Informació sobre els serveis

L'apartat sobre la informació dels serveis, té com a objectiu informar sobre tres serveis. Primerament, es vol fer saber a l'usuari quin és el llistat de ciutats en les quals es permet fer un recorregut amb guiatge, ja que tenim la informació sobre les parades i el temps d'espera del bus en la parada. D'altre banda, sobre cada ciutat es vol informar sobre quines companyies de bus hi ha conjuntament amb les seves dades d'informació de situació i contacte. Aquest segon servei, està encarat per si l'usuari te algun dubte sobre la companyia de serveis i si vol posar en contacte.

Per acabar, es vol oferir un accés al temps d'espera del bus en una parada sense que hi hagi un recorregut i s'hagi de fer el guiatge. Aquest servei, seria per als usuaris que es troben en una ciutat la qual se saben moure sense necessitat de guiatge, i l'únic que volen saber és el temps d'espera que li falta al bus, el qual estarà en un radi determinat.

Per tal de veure com s'han dissenyat les activitats d'aquests serveis, es pot veure el diagrama d'ús de l'apartat del llistat de ciutats i la informació de les companyies de bus en el *Diagrama 17*.

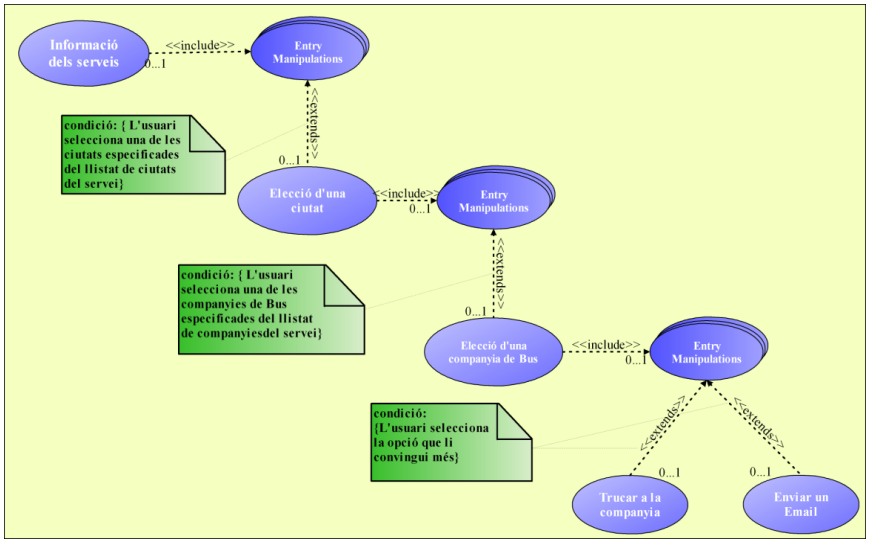


Diagrama 17 – Informació dels serveis

En la interfície 2, l'únic punt que varia és la opció d'enviar un Email. S'ha optat per no posar la opció degut a que la funcionalitat no és pròpia de l'aplicació, sinó que depèn de quina aplicació es fa servir per defecte en el terminal adquirit. Com és una aplicació externa i no podem garantir que hi hagi accessibilitat per a usuaris amb limitacions funcionals visuals, s'ha optat per treure-la i deixar únicament la trucada, la qual serà més senzilla de realitzar per part de l'usuari i més ràpida, que no pas intentar escriure un Email.

7.3 Temps d'espera del bus

L'apartat sobre el temps d'espera del bus en la parada, consisteix en proporcionar a l'usuari directament la consulta del temps d'espera requerit per a que passi el bus d'una línia en una parada propera al punt de localització en la que es troba l'usuari.

Aquesta funcionalitat consisteix en oferir a l'usuari diverses opcions per a utilitzar el servei, basats en la localització automàtica d'una parada segons el GPS, o bé que sigui l'usuari, sota el seu criteri i saben els codis de parada i línia, el que introdueixi manualment les dades i com a ultima opció, se li permet a l'usuari seleccionar les dades de parada i línia a traves d'un mapa, on l'usuari

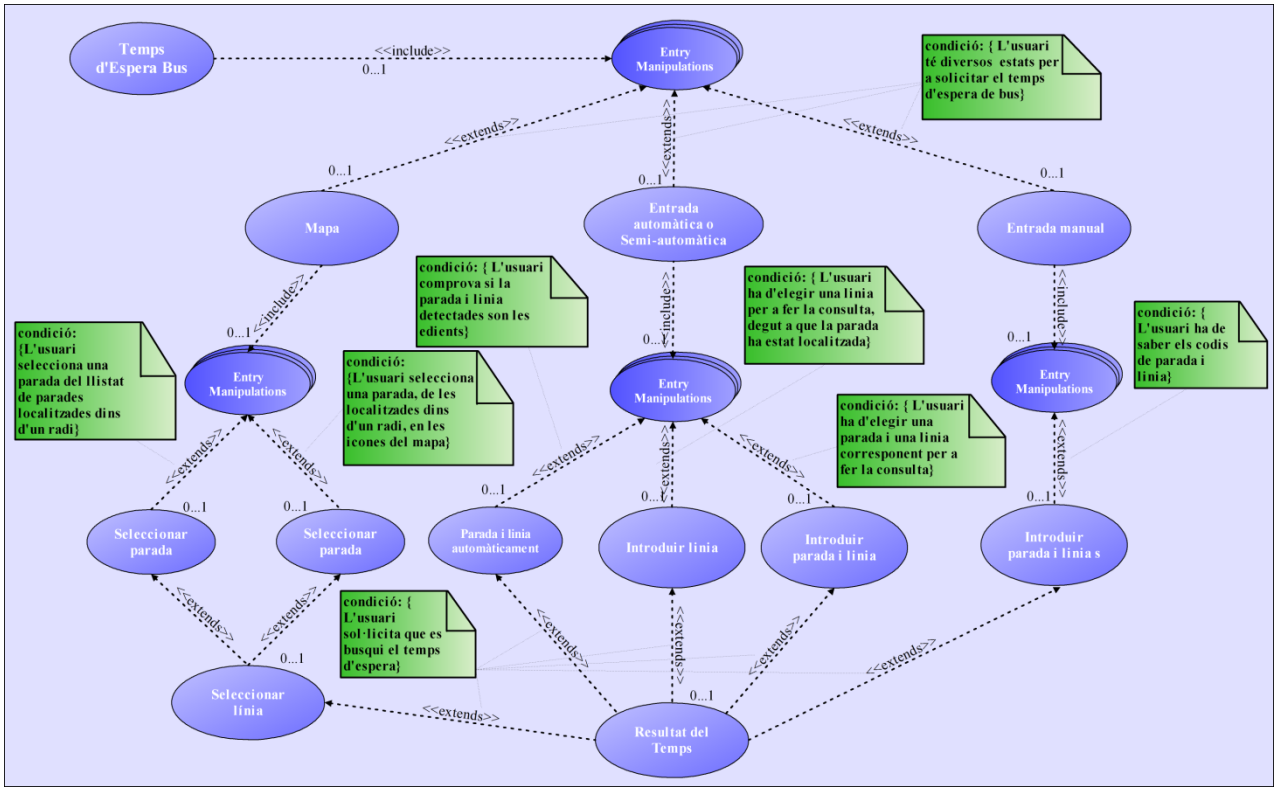


Diagrama 18 - Informació del Temps d'espera Bus

podrà veure també el típic punt de “vostè està aquí” rodejat d’icones de parades de bus.

El motiu per el qual s’ha inclòs la inserció dels codis manualment, és degut al marge d’error que pot tenir el GPS. En la situació on el GPS té plena cobertura i per tant, tingui una localització prou exacte, la cerca de parades més properes a aquesta localització serà correcte. En canvi, en el cas que el GPS tingui un marge d’error més gran, degut a aquesta cobertura, llavors la localització no serà gaire exacte i per tant, la cerca de parades més properes podria ser incorrecte i no ajustar-se a les peticions de l’usuari. En aquest cas, es va pensar en proporcionar el màxim d’opcions per a poder ajustar la funcionalitat al marge d’errors del GPS i a les peticions dels usuaris.

Així doncs, per tal de veure les múltiples opcions o camins per arribar a l’objectiu de cercar el temps d’espera del bus, es pot consultar el *Diagrama 18*, el qual com es pot veure, és una mica més extens i complex que els anteriors diagrames.

En la interfície 2, l’únic punt que varia és el mapa. Tal i com es pot entendre, un usuari amb una limitació funcional visual, no podria seleccionar una icona d’un mapa, sent el mapa gens accessible a l’exploració dels elements.

En quan a la resta d’opcions, es proporcionen de la mateixa manera, ja que funcionament l’aplicació localitza les parades properes per a que l’usuari en seleccioni el codi, o bé proporciona a l’usuari la opció d’introduir el mateix els codis mitjançant l’entrada de dades per Gestures. Per tal de apreciar el resultat d’aquestes pantalles de servei, es poden veure en la *Imatge 30*.

Imatge 30 – A l’esquerra de tot, informació sobre les companyies de bus. A l’esquerra - centre, informació sobre el temps d’espera bus, en el qual s’ha localitzat una parada i una línia. A la dreta central, mapa amb les icones de les parades properes al punt de localització. I a la dreta del tot equivalència de la informació del temps d’espera bus per a la Interfície 2.

7.4 Guiatge per a usuaris amb limitacions visuals

L'*Activity* de guiatge és la més important per a l'aplicació desenvolupada, la que mou més informació i amb la que hi ha més comunicacions. Aquesta *Activity* a nivell d'implementació té una gran diferencia entre la interfície 1 i la 2, degut a que la interfície 1 és molt gràfica i hi ha un mapa amb el recorregut a seguir pintat de color.

Per tal de poder mostrar les mateixes dades en les dues interfícies, s'han posat interpretacions de la informació gràfica del mapa, en el sistema de menús de la interfície per a usuaris amb limitacions funcionals visuals. Així com també s'ha dissenyat una sèrie de submenús per a cada guiatge existent en un recorregut, els quals segons l'estat del recorregut s'aniran intercanviant en la mateixa *Activity*.

En aquesta *Activity,* hi ha cinc submenús, els quals tres són els importants del guiatge i dos són funcionalitats extres per a l'usuari.

- **Guiatge a peu:** Submenú amb les indicacions a realitzar i l'accés a la brúixola per tal que l'usuari es pugui posicionar correctament quan se li indica un punt cardinal.
- **Guiatge esperant el Bus:** Submenú amb la informació de la parada i el temps que falta per a que passi el Bus.
- **Guiatge en el bus:** Submenú amb la informació de la parada que s'ha pujat, la següent a la que va el bus i en la que l'usuari s'ha de baixar.
- **Opcions Extres:** Submenú amb les opcions de poder guardar favorits o contactes, així com sol·licitar a quin carrer es troba l'usuari i demanar informació de la companyia de bus que agafarà en el recorregut.
- **Brúixola:** Submenú que segons la indicació de caminar que ha de realitzar l'usuari, el sistema li indica quants graus ha de girar cap a la dreta o a l'esquerra, fins a orientar-se correctament.

Tal i com s'ha indicat aquests submenús s'intercanvien segons l'estat de guiatge i segons l'estat de sol·licitud dels extres. L'estat del guiatge canvia depenent de la informació que es rep a través de les comunicacions amb el Nucli de l'Aplicació. Hi ha una sèrie de condicionals, els quals cada comunicació que es rep, es comprova de quin tipus de guiatge és i depenent d'això es carrega la nova informació que es rep en el submenú de guiatge corresponent. En canvi, l'estat dels submenús extres, depenen completament del usuari, el qual es canvia si aquest selecciona l'opció d'informació

extra o el de brúixola. Per tal de veure amb més senzillesa el que s'acaba d'explicar, en el *Diagrama 19* és pot veure els estats de l'*Activity*, juntament amb les accions que es duen a terme per canviar els submenús.

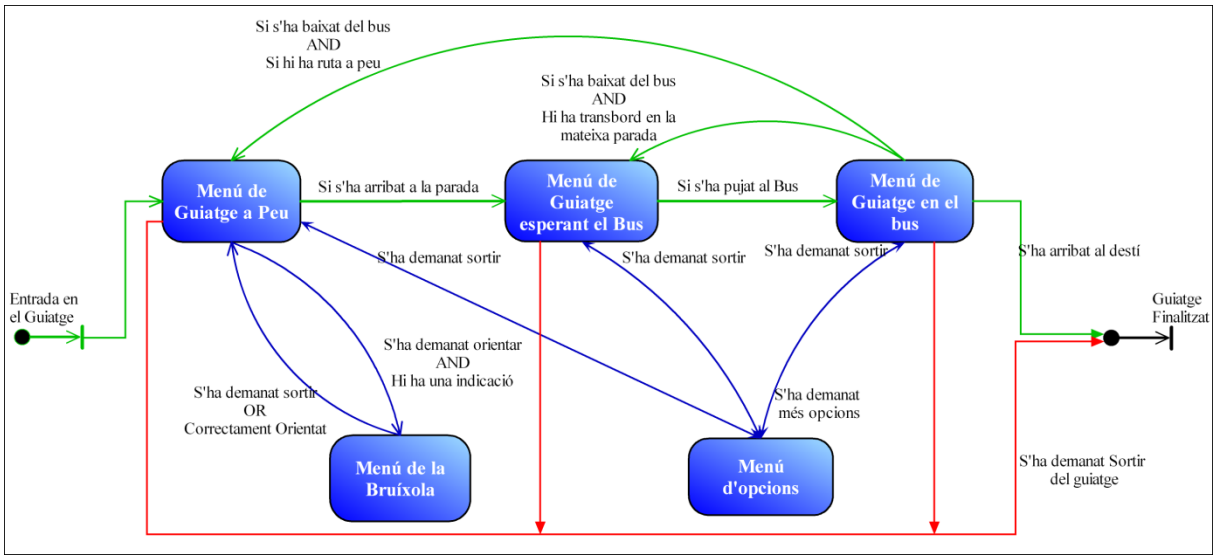


Diagrama 19 - Diagrama d'estats de l'Activity de Guiatge

Per tal de fer una interpretació de la informació del mapa, s'ha hagut de agafar les dades i passar-les a text, en algunes ocasions. La informació que s'ha hagut d'interpretar és la següent:

- **Fletxa informativa de l'acció**: Si el mapa apareix una fletxa de girar a l'esquerra i sota la fletxa, els metres que falten per a fer aquesta acció, en la interfície 2, s'ha incorporat una funció, la qual segons quina fletxa és i els metres que falten, es retorni una frase correcte.

- **Orientació:** La indicació pot tenir referències a punts cardinals, com ara "continuï recte cap al sud-est" Aquesta indicació per a un usuari amb limitacions visuals, no la pot interpretar degut a que no es podrà orientar. Per això s'ha inclòs una brúixola que proporciona 4 camps d'informació, la qual se n'ha interpretat la informació del que seria una brúixola gràfica. Es rep un nombre de graus i un flag de dreta i esquerra. Així com dos nombres de graus que indiquen la orientació de la indicació caminant per una banda i la orientació en que es troba el mòbil.

La informació de graus i del flag, es converteix en una frase que indica “Giri 135 graus cap a la seva dreta”. Aquesta frase es va actualitzant cada 45 graus que es giri l’usuari amb el terminal, fent que a l’arribar a 0 graus s’indiqui que s’està ben orientat. Per altre banda, els graus sobre la orientació del terminal i sobre la orientació de la indicació, es passa per una funció la qual retornarà un text com ara “Orientació de la indicació de guiatge: Nord” i “Orientació en la que es troba vostè: sud-oest”. Amb tota aquesta informació, l’usuari és capaç de orientar-se sense problemes. En el *Figura 12* se’n pot veure la idea de la interpretació de la brúixola per els exemples de text indicats.

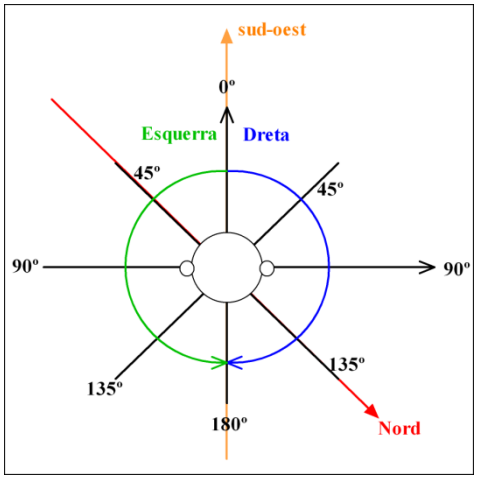


Figura 12 - Esquema de la brúixola en un cas concret

- **Informació de la Parada de Bus***:* En la interfície del mapa, apareixen les parades de Bus pintades en la seva localització. Així doncs, quan s’està en la parada, se sap a quina s’està ja que la podem veure en el mapa pintada juntament amb el punt GPS en el que ens trobem. Que passa amb els usuaris amb limitacions Visuals? Aquest pot arribar a la parada a través del guiatge, però si se’n troba dues de juntes pot ser que no sàpiga quina agafar. És per això que s’han inclòs tres informacions essencials el codi de la parada, el nom i el codi de la línia a agafar. En cas de que vulgui preguntar si està en la parada adequada, en podrà dir el codi o el nom sense cap problema.

- **Recorregut en Bus:** Un cop més, el mapa conté les parades del recorregut en el bus, pintat en el mapa, així com la llista de totes les parades de la línia amb les que ha de recórrer de color verd, les que ha recorregut en blau i la final de la ruta marcada per que sàpiga quina ha de baixar. Aquesta informació s’ha adaptat en la interfície de submenús fent que es digui el codi i el nom de la parada en la que s’ha pujat, indica quina és la següent parada a la que el bus es dirigeix i indica el nom de la parada en la que ha de baixar. A més a més es va actualitzant el nombre de parades que falten per baixar i s’avisa sonorament i amb vibració quan ha de demanar de baixar de la parada. Amb tota aquesta informació, l’usuari pot baixar a la parada a la que es dirigeix.

Amb totes aquestes interpretacions gràfiques, aquesta activitat queda plenament accessible per a l'usuari amb limitacions visuals, fent que pugui realitzar el recorregut amb tota la informació disponible per veu. En el *Diagrama 20* es pot veure el funcionament de l'activitat i de totes les opcions que l'usuari en pot fer ús.

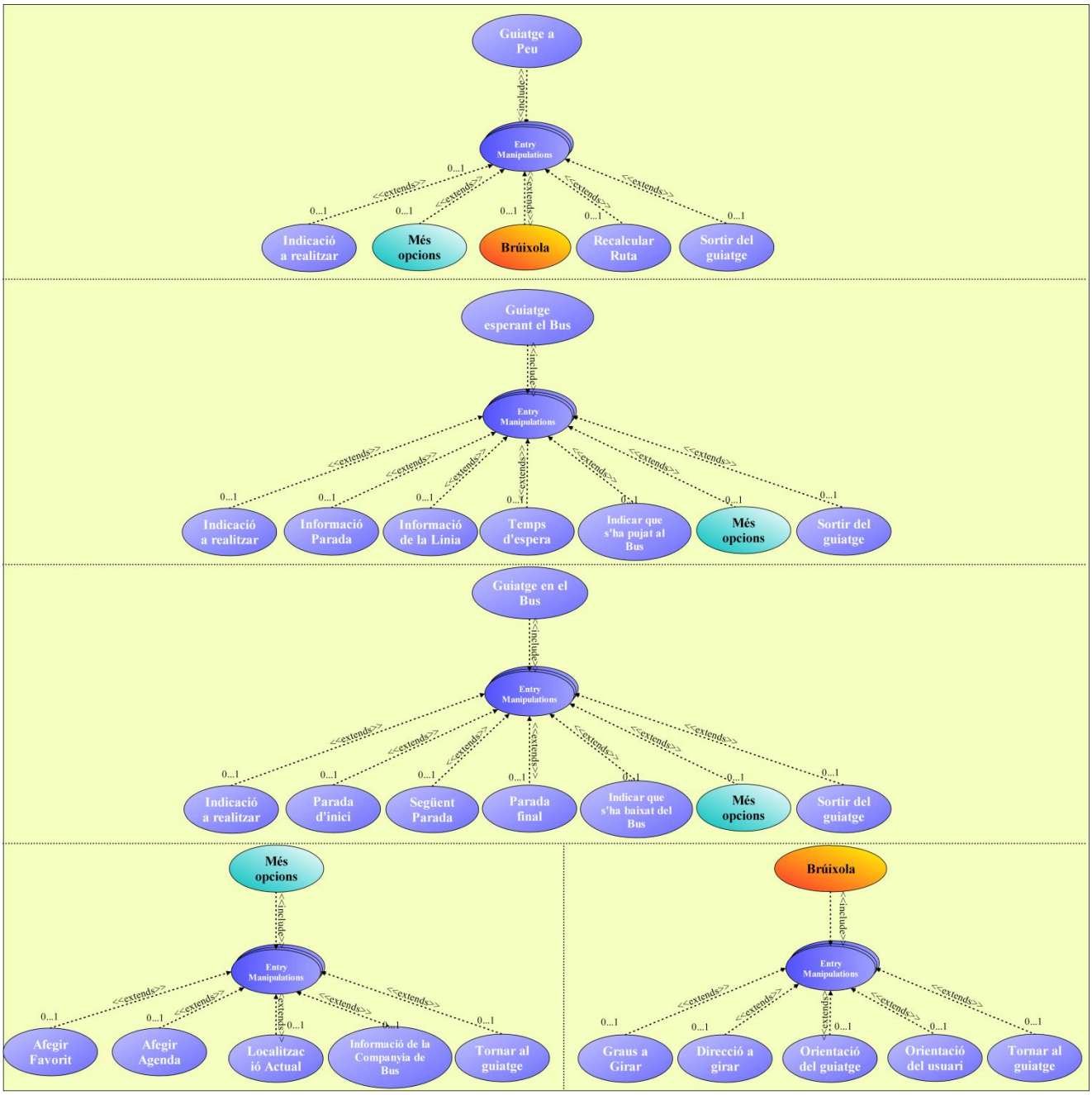


Diagrama 20 - Guiatge per a la Interfície 2

8 Proves i resultants

Durant tot el projecte s'han anat fent múltiples proves per tal de poder testejar el correcte funcionament de l'aplicació. Per la realització de les proves s'ha utilitzat l'emulador que Android incorpora en el seu SDK (permet emular dispositius mòbils amb el SO d'Android incorporat sense necessitat d'utilitzar un dispositiu físic) i els diferents telèfons mòbils disponibles pels membres del projecte.

A continuació es detallen algunes de les proves més significatives realitzades i els resultats obtinguts. Es divideix en dues parts, primerament les proves unitàries(o de mòdul en cas que hi hagin) i en el segon punt les proves i resultats de camp a la ciutat de *Barcelona*.

8.1 Proves i resultats de camp

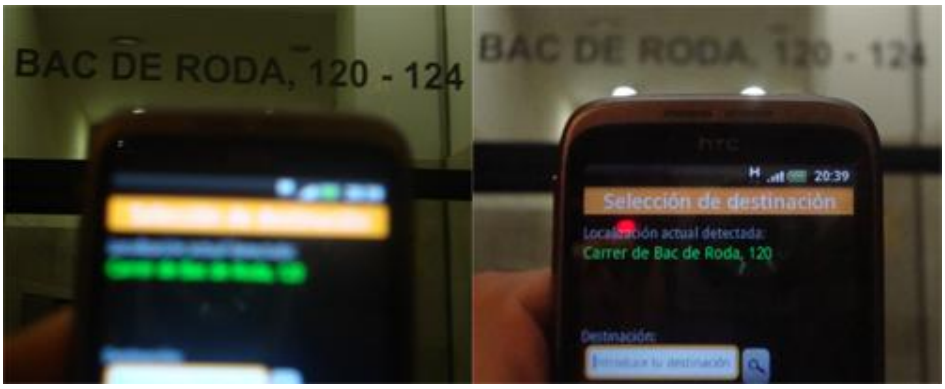
Proves realitzades a Barcelona. Per la resta de mòduls (ciutats) consultar l'*Annex B*. Les proves fan referència al perfil sense discapacitat. El sistema de guiatge és el mateix per tots els perfils, només varia la interfície, per tant, s'obtindran els mateixos resultats.

8.1.1 Proves i Resultats de Localització

En aquest apartat es mostraran els diferents resultats obtinguts al obtenir la localització actual de l'usuari:

- Localització Precisa:

Il·lustració 1 - Localització amb perfecte precisió.



Il·lustració 2 - Localització amb perfecte precisió

- Errors de localització:

Il·lustració 3 - Localització errònia

Es aquest cas es pot comprovar que hi ha un imprecisió. L'usuari està al carrer *Gran Via*, número *1038*, però la localització marca el número *1042* del mateix carrer. Això és degut a que el *Geocoder* no mostra tots els números d'un carrer, sinó que només mostra uns quants en un segment determinat i per tant treu precisió a l'hora d'obtenir i mostrar l'adreça actual del usuari.

Un error d'imprecisió semblant i derivat del *Geocoder* es troba en la *Il·lustració 4.* En aquest cas es pot veure que l'usuari està una mica lluny de la seva posició actual i a més que el *Geocoder* retorna un conjunt de números en els quals pot estar l'usuari, per tant, la precisió en cas d'estar en el conjunt de números, tampoc serà total.

Il·lustració 4 - Localització errònia, agrupació de números.

8.1.2 PROVES I RESULTATS DE GUIATGE

8.1.2.1 Guiatge WALK

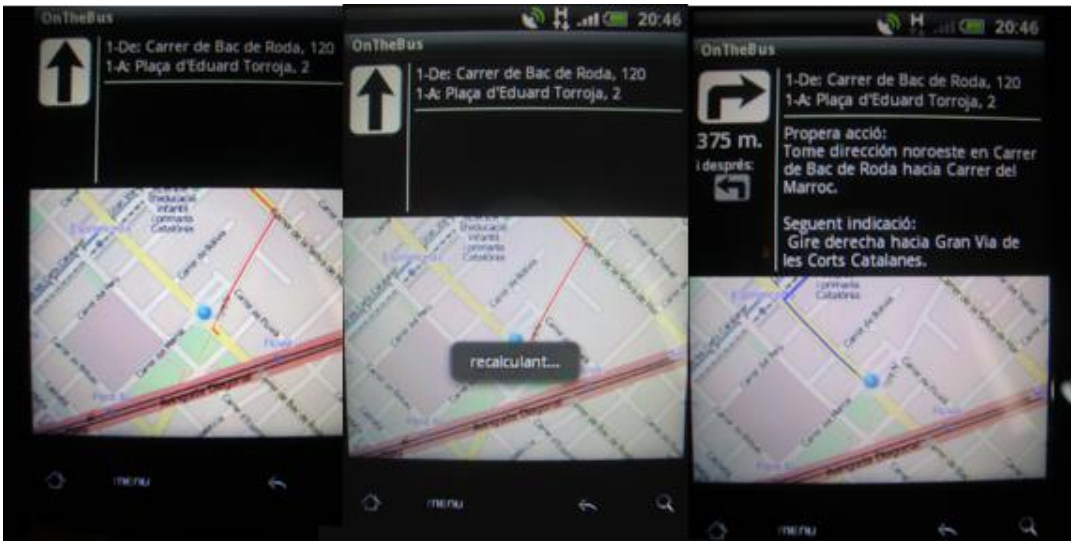
- Guiatges

Il·lustració 5 - Exemple simple de gir.

En la *Il·lustració 5* es pot veure com avança l'usuari pel camí i la indicació marcada. Just en el punt en que l'usuari ha de girar per un carrer, li es mostrada la indicació de gir sense cap tipus d'error. En les següents il·lustracions *Il·lustració 6* mostrarà uns exemples de guiatge amb un bona precisió de girs.

Il·lustració 6 - Guiatge amb un bona precisió de girs

- Recalcular la ruta



Il·lustració 7 - Recalcular Ruta

Es pot apreciar en la primera imatge de la *Il·lustració 7* que l'usuari ha iniciat el guiatge de caminar però que no li arriben indicacions, ja que mentres es calculava la ruta ha mogut la seva posició (es pot apreciar la diferència entre l'inici del camí vermell i el punt de localització). En les dues següents imatges es pot veure com l'aplicació detecta l'error i automàticament recalcula la ruta de caminar fins al proper punt de guiatge o destí de l'usuari.

- Errors
  - GPS

Il·lustració 8 - Guiatge Walk. Error localització GPS.

En la *Il·lustració 8* es pot apreciar que el punt de localització està lluny del camí que hauria de seguir l'usuari (marcat en vermell) i que a més, el punt marca que l'usuari està dintre d'un edifici. Aquest error de localització passa de tant en tant i és molt difícil de controlar, ja que deriva de l'error de GPS que tenen els dispositius mòbils.

  - Errors Mapa

Hi ha vegades que el mapa que utilitzem de *OpenStreetMap* no s'acaba d'aproximar amb l'estat actual dels carrers, i això ens indueix principalment a dos tipus d'error:

    - Cantonades

Il·lustració 9 - Guiatge Walk. Cantonades

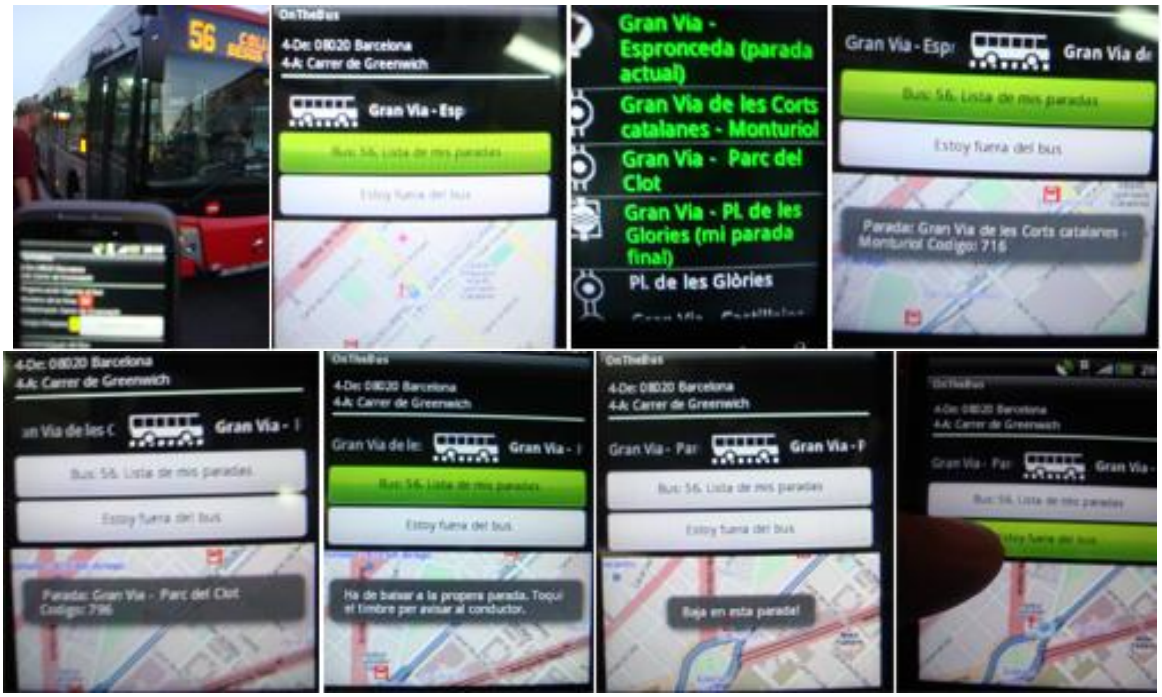
Sovint el mapa mostra cantonades amb girs de 90º (amb forma de "quadrat", com es pot veure a la *Il·lustració 9*) que no s'aproximen amb la realitat, ja que moltes han sigut adaptades en forma de diagonal per aprofitar millor l'espai públic. Aquest fet indueix a imprecisions en el guiatge ja que l'aplicació s'ajusta al mapa proporcionat per poder implementar el servei. Per tal de solucionar-ho s'ha augmentat el radi de la *proximity alert* que avisa que l'usuari ha de girar, per a que aquest ho pugui fer correctament, com es pot veure en l'exemple.

- Voreres

Il·lustració 10 - Guiatge Walk.

És el cas més complicat i de moment impossible de controlar degut a la seva complexitat i dificultat. El servei proporciona una ruta i indicacions que són mostrades a l'usuari en el format indicat. El problema deriva en que moltes vegades quan hi ha un canvi de vorera no es proporciona aquesta informació, i per tant, no es pot indicar al usuari que canviï de vorera quan es fa el gir. En el mapa de la *Il·lustració 10* es pot veure com es realitza un canvi de vorera però no s'informa en cap moment en les indicacions.

8.1.2.2 Guiatge BUS



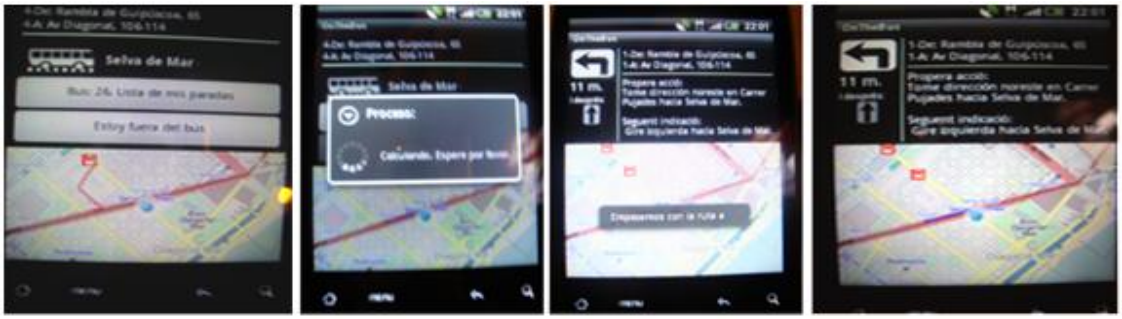
Il·lustració 11 - Guiatge Bus. Etapes de Guiatge

Com es pot veure, un cop l'usuari puja a l'autobús pot consultar les parades que recorrerà, i cada cop que passi amb aquest per una parada rebrà un avís on l'indicarà la parada actual del recorregut. Durant el trajecte entre la penúltima i última parada, es mostra un missatge cada cert

temps alertant a l'usuari de que ha de baixar en la següent parada i finalment al arribar a l'última parada, s'informa a l'usuari de que ha de baixar. Un cop ho hagi fet, l'usuari confirma a l'aplicació que ha deixa't l'autobús per continuar amb el següent guiatge.

- Problema: Passar-se de Parada

En aquest cas, l'aplicació avisa a l'usuari de que no ha baixat en la parada correcta i que cal recalcular la ruta. Un cop l'usuari baixa del autobús, es recalcula automàticament la ruta fins el destí (línia blava de l'última imatge) i comença de nou el guiatge.

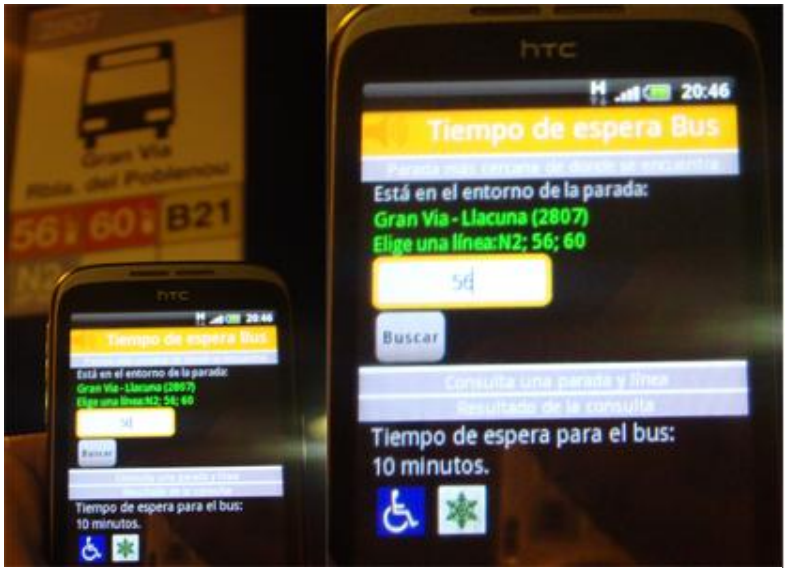


Il·lustració 12 - Guiatge Bus. Exemple de passar-se de parada

8.1.3 PROVES TEMPS ESPERA

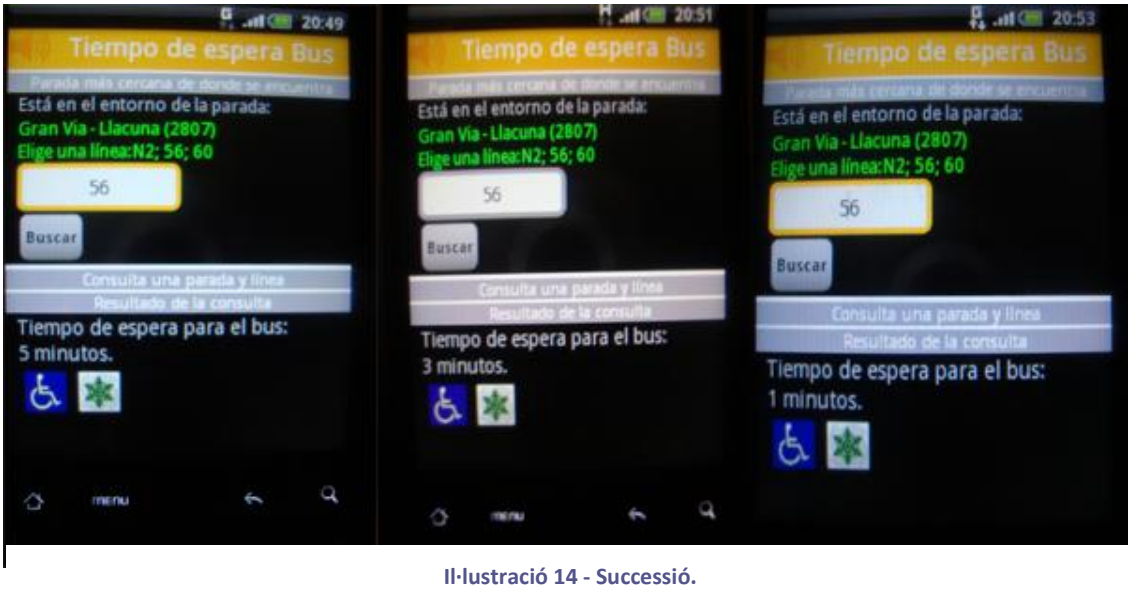
S'ha generat un conjunt de test per veure el nivell de precisió real del temps d'espera a la parada proporcionat per *TMB*.

- Test 1: Precisió Correcte.



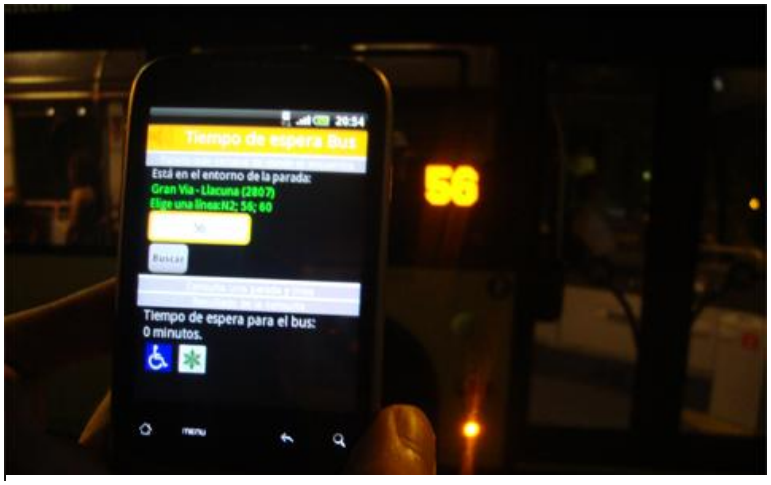
Il·lustració 13 - Precisió correcte.

Com es pot veure en la *Il·lustració 13*, l'aplicació detecta automàticament la parada en la que l'usuari està esperant el bus (codi *2807*). En aquest cas, al haver-hi diferents línies l'usuari ha d'introduir la línia que vol utilitzar.



Il·lustració 14 - Successió.

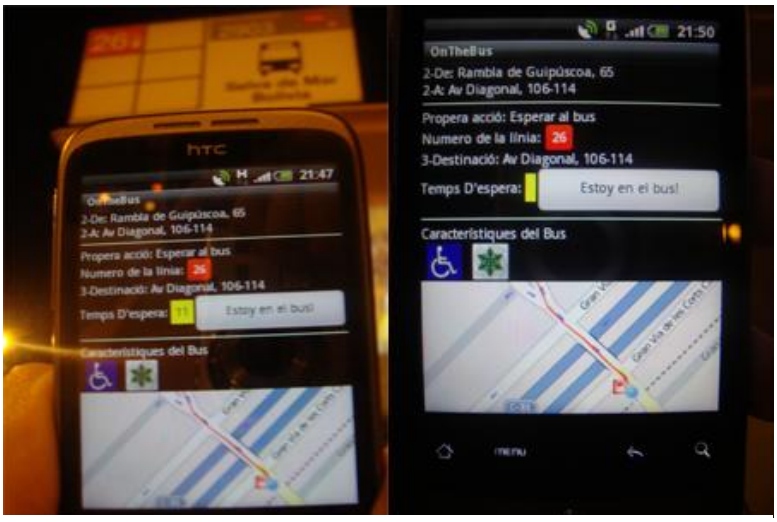
Si s'observa el temps que marca l'aplicació i l'hora que marca el telèfon mòbil, es pot veure que l'autobús sembla que arriba sense cap retard a la parada.



Il·lustració 15 - Precisió correcte.

Finalment, es pot apreciar que l'autobús,ha arribat a l'hora prevista.

- Test 2: Error per falta d'informació.



Il·lustració 16 - Successió Temps Espera errònia. Falta d'informació.

Hi han casos en que la informació del temps d'espera no és disponible ja que falla la consulta a Internet per la connexió o perquè el proveïdor del servei no té les dades operatives, com es pot veure en la *Il·lustració 16* anterior on passem de mostrar els minuts restants a no poder mostrar res.

- Test 3: Retard en l'arribada del autobús.

Il·lustració 17 - Successió Temps Espera. Retard arribada.

A la *Il·lustració 17* es pot observar que l'autobús ha petit un retard en el seu trajecte i trigarà més dels tres minuts que marcava la consulta inicial. Durant tres minuts de rellotge (*20:13 – 20:15*), el temps d'espera marcat per l'aplicació és el mateix, dos minuts. També s'hi produeix una imprecisió quant està a punt d'arribar a la parada, ja que es passa d'un minut restant a vint-i-tres, és a dir, hauria d'haver arribat i ja s'està esperant al següent.

En aquesta sèrie de testos es pot veure que de vegades el temps que falta pel proper autobús potser no concorda del tot amb la seva arribada, tot i tenir una precisió força alta en global. En el test 3 per exemple, es passa d'un minut restant a vint-i-tres, tot i que el bus no ha passat per la parada i pot estar parat en un semàfor o a uns metres. Els minuts que falten s'obtenen a través d'una consulta al servei web de la companyia en qüestió, per tant, depenem exclusivament d'aquest servei i no es pot generar un control en cas de diferències entre la informació que mostrem i l'arribada del bus.

9 Conclusions i possibles ampliacions

9.1 Conclusions personals

En el mòdul de la Interfície d'usuaris, s'ha acomplert l'objectiu principal de dissenyar una interfície universal amb una gran accessibilitat per a terminals mòbils amb tendència a no tenir botons físics. Tot i que el disseny universal indiqui que s'ha de fer una única interfície única, s'ha trobat idoni realitzar-n'hi dues per tal de poder profunditzar amb tots els elements d'*Android*, així com poder oferir la màxima personalització de les *Views* i un disseny molt estètic i visual gràficament amb elements estàndard d'*Android* que s'espera poder trobar en una aplicació del seu entorn.

D'altre banda, l'altre interfície és senzilla, però molt funcional i accessible, plena d'interpretacions dels elements visuals com ara el mapa, la brúixola o bé les fletxes d'indicació. A més a més compta amb la configuració per tal d'ajustar l'aplicació a les necessitats de cada usuari.

També s'han afegit moltes funcionalitats extres al voltant del guiatge, com ara favorits i contactes, i consultes directes del temps d'espera o per trucar a la companyia de bus, que dóna un valor afegit a la funcionalitat principal de l'aplicació, fent que l'aplicació sigui molt completa.

Personalment, s'ha acomplert el meu objectiu d'aprendre el funcionament del Android, les eines bàsiques de programació, diferents maneres de dissenyar pantalles i personalitzar-les. Així com obtenir un terminal propi per a realitzar proves de camp i poder realitzar futures aplicacions de caire personal o professional.

9.2 Conclusions generals

Els objectius principals del projecte s'han assolit. S'ha implementat una solució seguint el disseny universal i que funciona, realitzant correctament el guiatge des del punt inici on es troba l'usuari fins un punt destí fent ús de la xarxa de transport d'autobusos per a diferents ciutats d'arreu del món.

També s'ha aconseguit gestionar diferents grups homogenis en el desenvolupament d'un projecte de dimensions importants creant sinergies entre aquest grups per a assolir l'objectiu principal.

Alguns dels inconvenients que han aparegut han estat relacionats amb la precisió del dispositius GPS que incorporen els mòbils que s'han usat durant les proves de l'aplicació, donant diferents resultats cadascun d'ells. Això ha fet que s'allargués la duració del projecte per sobre del

que s'havia previst inicialment. Aquesta desviació ha suposat aproximadament cinc setmanes de retràs, que han fet que l'entrega es realitzes en setembre.

9.3 Possibles ampliacions per mòdul

En lo respectiu a futures millores de la interfície, es podria ampliar el grau de personalització de l'aplicació. Seria possible canviar els colors de més *Views* de forma dinàmica, com per exemple els *ListView*. A més, l'usuari podria variar la mida de la tipografia de forma dinàmica.

Un altre element que seria interessant personalitzar, seria la barra de progrés "*seekbar*" mitjançant la selecció de *skins* de forma estàtica o bé la selecció del seu color de forma dinàmica.

Altre punt a millorar, seria modificar el *layout* de guiatge fent-lo més atractiu, incorporant un sistema de finestres lliscants per a minimitzar i maximitzar la informació de guiatge.

Una alternativa a la millora anterior seria dissenyar una interfície de guiatge utilitzant conceptes de realitat augmentada.

Per millorar el tema de *Gestures*, es pot incorporar en properes versions, una funcionalitat per a inserir *Gestures* pròpies i associar-los als contactes de l'agenda i a favorits. Una altra millora per als usuaris que no necessiten un perfil d'accessibilitat visual, seria millorar el sistema de detecció de símbols. Per tal de realitzar aquest canvi, a l'hora d'obtenir les diferents puntuacions en el *matching*, es descartarien tots els símbols menys els tres amb millor puntuació i es mostrarien aquestes alternatives a l'usuari com a suggeriment.

En la secció dels Taxis, seria interessant el tema d'enviar la posició actual a la companyia seleccionada en el moment de trucar.

Per aconseguir una experiència d'usuari encara millor, seria interessant explotar més el mòdul de reconeixement de veu i millorar la interfície d'accessibilitat visual, intentant homogeneïtzar les dues interfícies.

Com a darrera millora, no estaria malament aprofitar les eines de *Google Analytics* per tal de trobar els possibles colls d'ampolla del conjunt de menús.

9.4 Possibles ampliacions de l'aplicació

A continuació es presenten algunes de les possibles ampliacions a realitzar en l'aplicació. Es tracta de noves funcionalitats afegides, que resten pendents d'anàlisi i disseny.

Cerca per punts d'interès (POI): Es tractaria d'afegir la geolocalització de diferents punts d'interès per tipologia (comissaries, farmàcies, restaurants, etc.). Encara que queda pendent d'anàlisi aquesta modificació implicaria la modificació de la majoria del mòduls de l'aplicació.

Rutes turístiques: Es tractaria de tenir rutes complertes d'inici a fi amb un o més punts turístics. Es podria enllaçar amb l'ampliació anteriorment esmentada per tal d'emmagatzemar els punts turístics com a punts d'interès.

Recorregut Invers: Es tractaria de poder indicar-li a l'aplicació que emmagatzemi el punt inici i final de ruta per a en un breu espai de temps demanar que es guiï de tornada al punt inicial. És una ampliació interessant per a quan et desplaces puntualment a un destí.

Cercar un pla de negoci: Consistiria en l'addició d'algun tipus de publicitat en l'aplicació.

10 Bibliografia i referències

10.1 Llibres

- Ableson, Frank. Collins, Charlie. Sen, Robi. Android. Guía para desarroladores "Unlocking Android. A developer's Guide". Traducció de José Luis Gómez Celador. 1ª Edició. Ediciones Anaya Multimedia, 2010.

10.2 Documents WEB

- Roberto Calvo Palomino . Desarrollo en Android(v1.0). GSyC/LibreSoft. June 17, 2009.
- Gramlich, Nicolas. Andbook! release.002. Android Programming. Tutorials from the anddev.org-Community.
- Byron, Tutorial: Desarrollo de aplicaciones para Android, viva-moore.blogspot.com, 2010.
- Desarrollo de aplicaciones móviles en Android , www.slashmobility.com, 2010.
- Pedro Valero i Julio Abascal. Accesibilidad. Universitat de Valencia i Universidad del Pais Vasco, 1999.
- Dra. María José Escalona Cuaresma i Dr. José Mariano González Romano, Metodología y Técnicas(Accesibilidad), Curso 2006-07
- Manuals de Serveis dels Terminals Android (HTC, LG, Huawei, Samsung, Sony Ericsson)

10.3 Articles

- Entrevista a Carlos Martínez, vicepresident d'ASOCIDECAT, l'Associació de sord-cecs de Catalunya. Jo també(Revista editada per la Fundació per a Persones amb Discapacitat Illa de Menorca), Edició número 3 (Juny del 2009), Pàg 37 – 39

- Diseñar para los discapacitados. REVISTA DE LA OMPI, Edició número 5 (Octubre del 2009) Pàg 22– 24.

- Guia Móviles - Android. Descubre las novedades del sistema, Personal Computer & Internet, Edició numero 93, Pàg 4 – 25 (2010).

- Samsung Galaxy S, el Android más potente. Personal Computer & Internet, Edició numero 93, Pàg 174 – 177 (2010).

10.4 Llocs WEB

- http://developer.android.com/index.html
  Android Developers.
  Creació al 2007. Escrit en Anglès.
  Pàgina oficial d'*Android*, on apareix tota la API, demos, tutorials i totes les novetats que van incorporants els desenvolupadors d'*Android*.
- http://stackoverflow.com/
  StackOverflow. Escrit en Anglès.
  Pàgina que ofereix una sèrie de serveis gratuïts, la qual és una barreja entre wiki, blogs i forums i Diig/Reddit. Aquesta pàgina ofereix als programadors un lloc on poder realitzar les seves preguntes sobre programació Android, etc... La resta de programadors usuaris de la pàgina, proposen la resolucions fins a trobar-ne una que funcioni.

- http://www.programaraciegas.es/
  Programar a Ciegas. Jonathan Chacón Barbero.
  Creació el 29/12/2006. http://programaraciegas.weblog.discapnet.es. Escrit en castellà.
  És un portal de divulgació d'informació sobre experiències, necessitats i coneixements sobre accessibilitat, usabilitat, disseny per a tots per a Android, Tablets, Iphones,etc...

- http://www.android-spa.com/
  Android-Spa. Escrit en castellà.
  És la comunitat oficial de *Android* en Español, on ofereixen un espai obert a tota la comunitat de desenvolupadors Android, amb tutorials, temes de debat i preguntes, noticies de les noves tecnologia *Android*.

- https://groups.google.com/group/desarrolladores-android
  desarrolladores-android. Creació al 2007. Escrit en castellà.
  Grup d'ajuda per a desenvolupadors d'android, on tenen tutorials i explicacions de resolucions de problemes.

- http://www.elandroidelibre.com/
  El Androide Libre. Escrit en castellà.
  Web de referència Android en Espanyol. Hi ha noticies, tutorials, etc.. Android i smartphones de Google.
- http://www.xatakamovil.com/
  Xataka Móbil. Escrit en castellà.
  Web amb articles i Reviews de terminals mòbils d'Android i altres.

10.5 Blogs Web

- http://www.demalagana.es/
  deMalagana. Escrit en castellà.
  Blog amb diversos articles, entre els quals hi ha 3 tutorials d'Android sobre el multi-idioma, estructura d'Android i els layouts.

- http://www.javielinux.com/index.php
  El blog de Javielinux, Javier Pérez Pacheco. Escrit en castellà.
  Blog personal d'un programador, el qual té petits tutorials d'Android i opinions personals sobre Android.

- http://www.hambonious.com/
  Hambonious. Creació al 2010. Escrit en Anglès.
  Blog personal d'un usuari que té varies entrades sobre Android

- http://www.androidengineer.com/
  Android Engineer. Creació al 2010. Escrit en Anglès.
  Blog personal d'un usuari que té varies entrades sobre Android. Sobretot pel tema de personalització de Views.

- http://blog.androgames.net
  Androgames blog. Creació al 2009. Escrit en Anglès.
  Blog personal d'un usuari que té varies entrades sobre Android. Sobretot pel tema de personalització de Views.

Índex de imatges

Índex de figures

Índex de codis

Índex de diagrames

Índex d'Il·lustracions

Memòria del projecte realitzat per:

Monica Esteve Romeu

Bellaterra, 12 de Setembre de 2011

El projecte OnTheBus pretén ser un sistema de guiatge gratuït, destinat per a la plataforma mòbil Android, útil com a eina d'ajuda pel desplaçament d'un punt d'origen a un punt de destí dins de la ciutat, utilitzant la xarxa de transport metropolità.

Aquesta memòria s'enllaça amb 6 memòries més, dels meus companys amb els qui he realitzat el projecte OnTheBus.

Per poder elaborar amb èxit l'aplicació OntheBus, cal donar importància al disseny i a la implementació de la interfície d'usuari. L'objectiu d'aquesta memòria és explicar com s'ha realitzat i dissenyat l'aplicació perquè sigui accessible per a tothom. Ja que pot tenir un rang molt ampli d'edats d'usuaris, la implementació de la interfície està basada en les directrius del Disseny Universal.

El proyecto OnTheBus pretende ser un sistema de guiado gratuito, destinado a la plataforma móvil Android, útil como herramienta para desplazarse desde un punto de origen a un punto de destino dentro de la ciudad, utilizando la red de transporte metropolitano.

Esta memoria se enlaza con 6 memorias más, de mis compañeros con los cuales he realizado el proyecto OnTheBus.

Para poder elaborar con éxito la aplicación OnTheBus, hay que dar importancia al diseño y a la implementación de la interfaz de usuario. El objetivo de esta memoria es explicar cómo se ha realizado y diseñado la aplicación para que sea accesible para todo el mundo. Como puede tener un rango muy amplio de edades de usuarios, la implementación de la interfaz se ha basado en las directrices del Diseño Universal.

The OnTheBus Project pretends to be a guiding free system, designed for the Android platform, with the target to go from one source point to a destination point in a chosen city, using the metropolitan transport network.

This study links with another six works from my partners who we have been developing this project.

To succeed with OnTheBus app, it is important to give relevance to the design of the application and to the user interface implementation.

The work's objective is to explain how we did the app to be accessible for everybody. Because the OnTheBus program can be used for people of different ages, this causes the interface implementation it is based into the Universal Design guidelines.