Help Desk Manager

Aplicació Android per a la gestió d'incidències

Memòria del projecte
d'Enginyeria Tècnica en
Informàtica de Sistemes
realitzat per
Alberto García Puente
i dirigit per
Òscar Cubillo Alonso

Escola d'Enginyeria
Sabadell, *Setembre* de *2013*

El sotasignat, ***Òscar Cubillo Alonso***,

professor de l'Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el treball al que correspon la present memòria ha estat realitzat sota la seva direcció per ***Alberto García Puente***

I per a que consti firma la present.
Sabadell, ***Setembre*** de ***2013***

Signat: ***Òscar Cubillo Alonso***

FULL DE RESUM – PROJECTE FI DE CARRERA DE L'ESCOLA D'ENGINYERIA

Títol del projecte: Help Desk Manager
Autor[a]: Alberto García Puente **Data:** Setembre de 2013
Tutor[a]/s[es]: Òscar Cubillo Alonso
Titulació: Enginyeria Tècnica en Informàtica de Sistemes
Paraules clau: · Català: Android, Java, SDK. · Castellà: Android, Java, SDK. · Anglès: Android, Java, SDK.
Resum del projecte (extensió màxima 100 paraules) • Català: HDM, Help Desk Manager és una aplicació android que permet crear i gestionar les incidències que es presenten en una empresa. • Castellà: HDM, Help Desk Manager es una aplicación android que permite crear i gestionar las incidencias que se presentan en una empresa. • Anglès: HDM, Help Desk Manager is an application that allows to create and to manage incidents that occur in a company.

Taula de continguts

Taula d'imatges

1. Estudi de viabilitat

1.1. Introducció

El projecte consisteix en un programa que realitzi el control de problemes d'empreses amb una millor comoditat i rapidesa, on l'usuari de l'empresa crearà una incidència y el Help Desk (conjunt de persones que gestiona y soluciona totes les possibles incidències via software) ha de resoldre-les.

1.1.1. Descripció

El projecte es una aplicació Android dissenyada amb eines adaptades a aquesta tecnologia. Tractarà d'una llista que contindrà problemes dels treballadors de les empreses, ja estiguin resolts o pendents de resoldre, entre altres, també contindrà informació de com contactar amb els helpdesk, preguntes freqüents, etc.

S'ha decidit realitzar aquesta aplicació Android a causa de que molts programes (Aplicacions webs o Android) que s'utilitzen actualment només realitzen les tasques més usuals. El que es pretén es realitzar una millora, crear de zero una nova aplicació més completa, on sigui més fàcil l'ús per l'usuari i pel propi help desk i on hi hagi una comunicació més directe entre aquests.

1.1.2. Objectius del projecte

	Crític	Prioritari	Secundari
Millorar comunicació		X	
Millorar informació		X	
Més senzill			X
Veure l'estat de la incidència	X		
Incidències ordenades per tipus			X

Il·lustració 1 – Objectius

1.1.3. Definicions, acrònims i abreviacions

- HD: Help Desk, conjunt de persones que gestiona y soluciona totes les possibles incidències.
- Usuari: Treballadors, Help Desk, clients, que fan ús de la aplicació.
- Incidència: La tasca que el HD ha de corregir.
- App: Abreviació d'*Aplicació*.
- SDK: Software Development Kit.

1.1.4. Parts interessades

Nom	Perfil	Responsabilitat
Administrador	Administrador del sistema	Gestió i control del sistema. Gestió d'usuaris.

		Introduir nous continguts a la App. Anàlisis i disseny. Desenvolupament.
Help Desk	Usuari expert	Recollida i consulta d'informació. La seva funció es resoldre el problema.
Client	Usuari no expert	Consulta i creació d'informació

Il·lustració 2 – Personal de l'aplicació

Nom	Descripció	Responsabilitat
Alberto García	Cap de projecte	Defineix, gestiona, planifica i controla.
Alberto García	Analista	Analitza l'aplicació.
Alberto García	Programador	Programa l'aplicació.
Alberto García	Tècnic de Proves	Proves internes y externes.
Oscar Cubillo	Director de Projecte	Supervisa la feina de l'alumne.

Il·lustració 3 – Personal del projecte

1.1.5. Referències

1. Normativa de projectes d'enginyeria tècnica.

   UAB. (19/10/2010). Normativa de projectes. Normativa de projectes [Online]. http://www.uab.cat/Document/541/595/Normativa_PFCNovembre2010.pdf

2. LOPD:

   (13/12/1999). Llei orgànica de protecció de dades de caràcter personal. Llei Orgànica [Online].

   http://www20.gencat.cat/docs/Adjucat/Documents/ARXIUS/lo15_1999lopdcp.pdf

1.2. Estudi de la situació actual

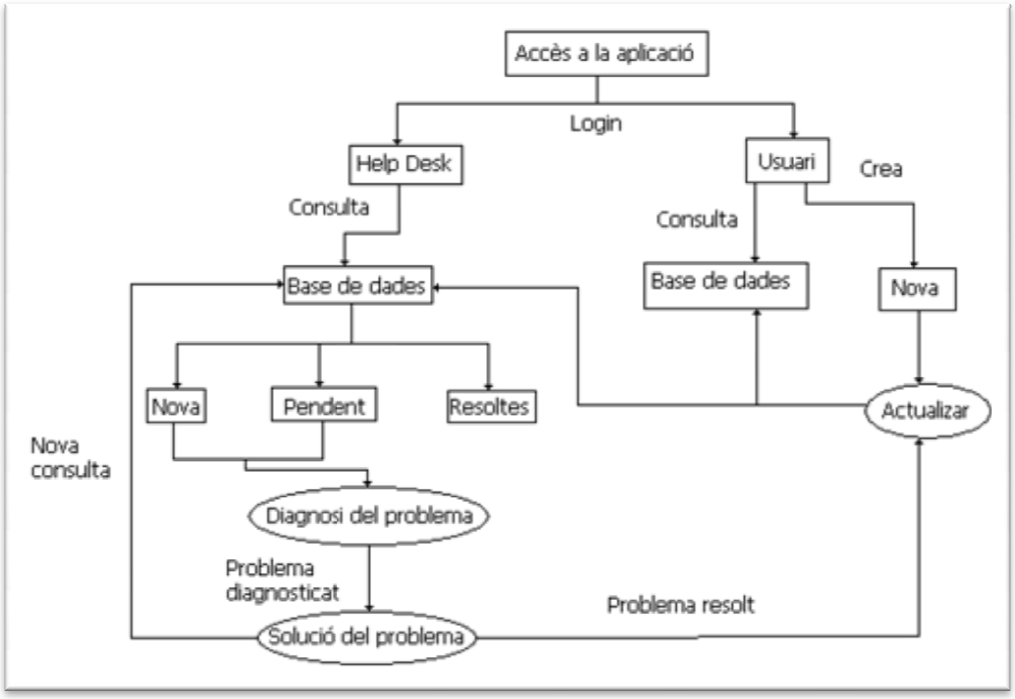
En aquest apartat tracta sobre com serà l'aplicació, a que es dedicarà, quin serà el seu mecanisme, com funcionarà, el que s'espera del projecte, les millores que porta i els seus defectes també. En general, es un estudi de com està actualment el projecte, per veure quines modificacions es fan en aquest.

1.2.1. Context

L'aplicació disposa d'un registre d'incidències. Aquestes apareixeran com resoltes, pendents de resoldre o noves. L'entorn on es desenvoluparà l'aplicació Android serà en el mateix ordinador propi de l'alumne que realitza aquest projecte. Aquesta eina serà desenvolupada a mida i estarà oberta a qualsevol empresa que vulgui provar-la, a mesura que es vagi provant l'aplicació pot ser millorada amb mes característiques que la millorin, per tal de fer una aplicació Android completa. La idea va sorgir quan l'alumne va realitzar unes pràctiques de Help Desk, la motivació principal va ser la de dissenyar una aplicació web que fos millor en tots

els aspectes, però com que encara hi ha poques aplicacions Android es va decidir per fer una aplicació més innovadora.

1.2.2. Lògica del sistema



Il·lustració 4 – Lògica del sistema

Explicació:

L'usuari farà login, depenen si es Help Desk o Usuari realitzarà unes tasques o unes altres.

Cas Help Desk:

Consultarà la base de dades, aquí podrà veure les incidències noves, les pendents de solució i les resoltes. Un cop obre una incidència nova, aquesta se li assignarà, intentarà trobar una solució, si la troba la tanca i s'actualitza la base de dades.

Cas Usuari:

Si te un problema, crearà una nova incidència explicant el seu problema, s'actualitza la base de dades. També te l'opció de consultar les incidències per veure si te un problema semblant ja resolt o no es l'única persona amb el problema.

1.2.3. Descripció física

En la *Il·lustració 5* hi ha les característiques principals de l'ordinador i mòbil que s'utilitzarà per programar l'aplicació.

Ordinador	Mòvil
Memòria ram: 3GB	Memòria ram: 1GB
Processador: Processador intel core	Processador: Procesador Exynos Cortex

2 quad Q8200 2.3 GHz	A9 dual-core 1.2 GHz, GPU Mali-400MP
Disc Dur: 1TB	Disc Dur: 12GB
DVD	
Monitor Samsung	4.3 Pulgades
Targeta Xarxa	
Windows XP-7	

Il·lustració 5 – Descripció

1.2.4. Usuaris i/o personal del sistema

Nom	Descripció	Responsabilitat
Usuari	Gestió	Gestiona altres tasques externes.
HelpDesk	Administració	Administra i soluciona totes les incidències.

Il·lustració 6 – Usuaris

1.2.5. Diagnòstic del sistema

En el diagnòstic del sistema es parla de totes les millores que ha de complir el projecte per tal que aquest tingui èxit, ja que al ser una aplicació que existeix ha de tindre una sèrie de condicions per a que les empreses s'interessin per aquest.

També es parla de les mancances ja que tots els projectes tenen uns pros i uns contres.

- Mancances
  - El sistema depenen del usuari, per tant, en el moment que arribin moltes incidències els Help Desk es podran col·lapsar i arribar a tindre masses incidències per persona.
- Millores
  - Accés eficient a la informació, els usuaris disposen de consultar la llista d'incidències i comprovar si necessiten ajuda externa.
  - Millora de seguretat
  - Millora d'informació respecte les incidències.
  - Millora de comunicació entre l'usuari i el Help Desk, ja que es pretén crear comentaris per incidència.

1.3. Alternatives i selecció de la solució

Aquest apartat parlarem de les possibles alternatives que tindrà una empresa en cas que el projecte no fos rentable, o no tingues èxit. Es presentes dues alternatives, que son dos programes que ja existeixen.

1.3.1. Alternativa 1

Adquirir una aplicació web ja dissenyada com el que ofereix Bankoi. Aquesta aplicació te unes bons accessos a les incidències. La seva interfície fa que els

usuaris s'adaptin bé. Però té un petit problema, que s'aplica a grans empreses i, per tant, és molt costós de mantenir. Un altre problema es que no compleix les expectatives d'aquest projecte ja que aquest està plantejat en ser utilitzat en una plataforma Android.

1.3.2. Alternativa 2

Adquirir una aplicació web ja dissenyada com el que ofereix Tecnocom.

Aquesta aplicació te uns bons accessos a les incidències, organitzades per tema i permet treballar amb molta comoditat. La seva interfície fa que els usuaris s'adaptin bé. Un problema es la comunicació entre usuari i Help Desk, en cas que una incidència es trigui en arreglar s'ha de trucar sempre a l'usuari. També s'aplica a grans empreses i, per tant, és molt costós de mantenir.

Un altre problema es que no compleix les expectatives d'aquest projecte ja que aquest està plantejat en ser utilitzat en una plataforma Android.

1.3.3. Solució proposada

	Costos adquisició	Costos adaptació	Nous recursos	Suport	Integració	Complexitat	Formació
A 1	Si	Desconegut	Si	Negociar	Mitja	Mitjana	Desconegut
A 2	Si	Alts	Si	Negociar	Mitja	Alta	Desconegut
A 3	No	Baixos	Si	Si	Alt	Baixa	Projecte

Il·lustració 7 – Propostes

A la vista de les característiques de aquestes alternatives, potser la millor alternativa seria la tercera encara que fos un projecte el que es pretén es millorar les altres aplicacions existents, a més, la alternativa 3 (solució proposada) no te cap cost d'adquisició i te uns costos d'adaptació baixos i s'adapta a les necessitats dels clients.

1.4. Conclusions

Beneficis:

- Reducció de despeses, al ser part d'un projecte no seria tant car.
- Reducció de costos de personal.
- Millora de seguretat de les incidències.
- Millora la formació dels usuaris, ja que al permetre veure la base de dades de les incidències podran trobar com arreglar el problema sense contactar amb el servei d'ajuda.
- Millora de solució d'incidències.
- Millora de comunicació, ja que creant comentaris, en cas de que una incidència es trigui a solucionar, no s'ha de estar pendent de l'usuari o el Help Desk.

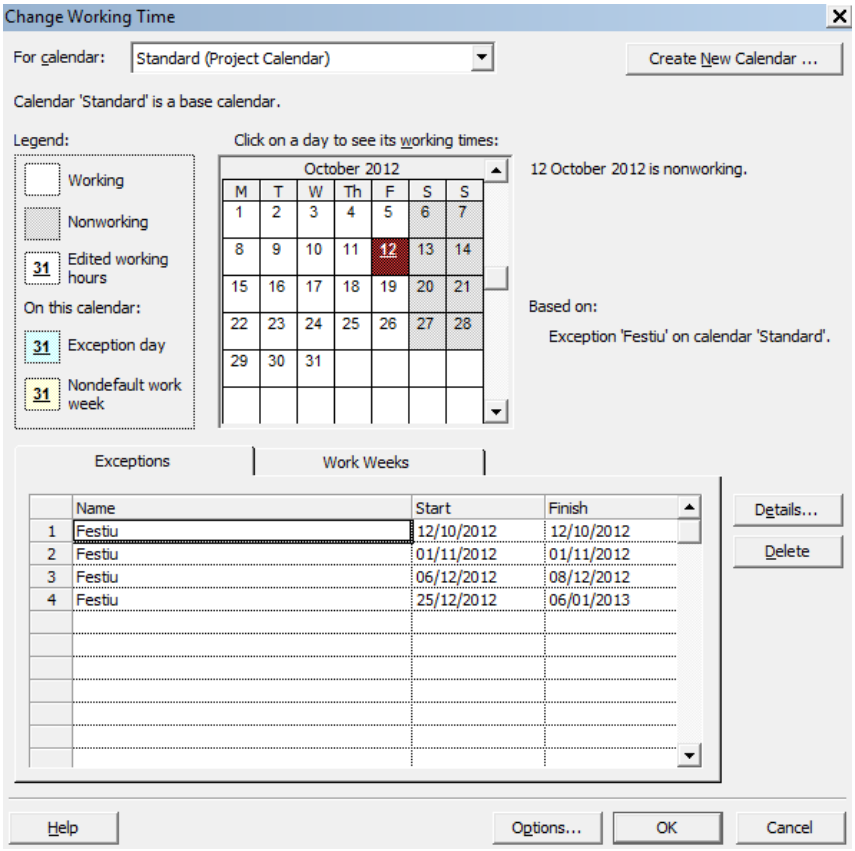
Inconvenients:

- Possible rebuig per part dels usuaris, ja que potser veuen que es semblant a altres aplicacions i al estar acostumats a les aplicacions webs rebutgin una nova aplicació.

1.5. Planificació

El total d'hores que s'ha de dedicar al projecte són aproximadament unes 200 hores, però com que han hagut uns canvis s'han realitzat unes 300 hores aproximadament.

La planificació del projecte és planificada amb l'aplicació Microsoft Project 2007. Primer de tot mirem quins seran els dies els quals no es farà el projecte, per tant s'ha de mirar quins dies seran festius. S'han agafat com a festius aquells dies que son festes generals i no autonòmiques, com per exemple Nadal i altres més. Per modificar-les realitzem un "Change Working Time" i escollim els dies.



Il·lustració 8 – Working time

En aquest cas hem escollit 12 d'Octubre, Festa nacional, l'1 de Novembre, Tots sants, del 6 al 8 de Desembre, Constitució i Inmaculada i per últim des del 25 de Desembre fins a l'1 de Gener, Nadal (Encara que, en cas de anar malament de temps, es pot dedicar unes hores al dia per posar-se al dia del treball).

Un cop estan els dies que es realitzarà el projecte, hem de decidir quins seran les persones que treballaran al projecte i el seu salari, per això haurem de anar a "View → Resource Sheet" i afegir el personal.

	Resource Name	Type	Material Label	Initials	Group	Max. Units	Std. Rate
1	comercial	Work		c		100%	16,00 €/hr
2	director de projecte	Work		d		100%	32,00 €/hr
3	tecnic	Work		t		200%	20,00 €/hr
4	becari	Work		b		150%	9,00 €/hr

Il·lustració 9 – Recursos del projecte

Un cop tenim el calendari i el personal, ens queda la part més important que és la llista de tasques, quines seran les predecessores i qui les portarà a cap. En aquest cas les nostres tasques són les següents:

Nom de les tasques
Sistema Inspecció Automàtica de Cinturons
Diagnòstic
Entrevista amb el client
Document especificació problema
Realització d'un document de diagnòstic
Estudi de viabilitat
Especificació del sistema d'adquisició d'imatges
Especificació del tipus d'il·luminació i càmera
Especificació del tipus de tarja digitalitzadora
Predisseny armari d'inspecció
Estudi detecció defectes
Generació banc de proves
Estudi dels defectes
Document especificació solucions software
Redacció del document de pressupost
Avaluació cost del Hardware
Planificació realització del projecte
Pressupost
Realització del projecte
Disseny armari d'inspecció
Muntatge elements d'inspecció
Codificació detecció defectes
Fase final
Documentació
Instal·lació en línea de producció
Muntatge del sistema
Fase de sintonització
Depuració d'errors

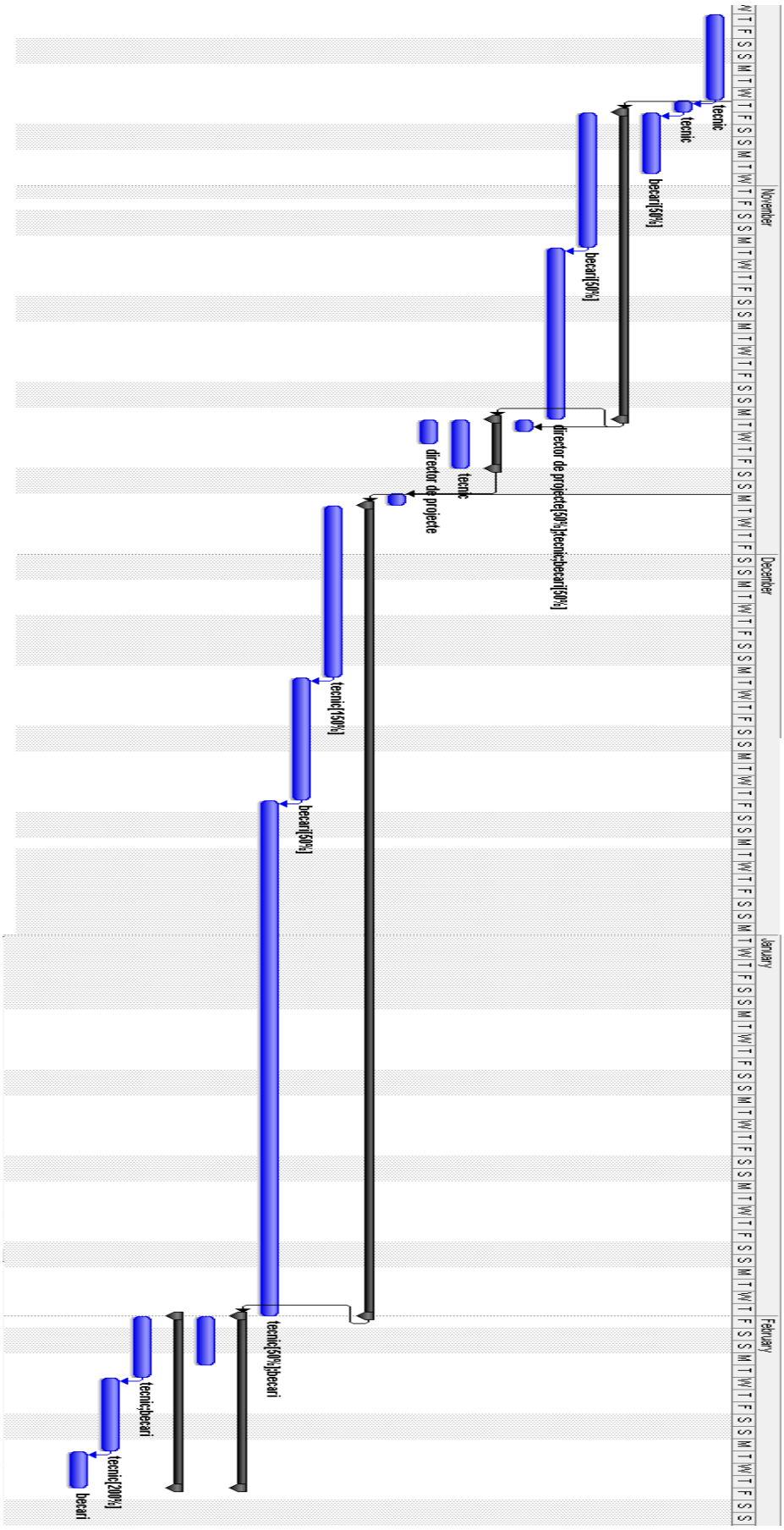
Il·lustració 10 – Tasques del projecte

Un cop les tenim decidim quines seran predecessores d'altres i qui les ha de realitzar:

	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1	**Sistema Inspecció Automàtica de Cintu**	**77 days**	**Mon 15/10/12**	**Thu 14/02/13**		
2	**Diagnòstic**	**3 days**	**Mon 15/10/12**	**Wed 17/10/12**		
3	Entrevista amb el client	1 day	Mon 15/10/12	Mon 15/10/12		comercial;director de
4	Document especificació problem	1 day	Tue 16/10/12	Tue 16/10/12	3	director de projecte
5	Realització d'un document de diagnòst	1 day	Wed 17/10/12	Wed 17/10/12	4	
6	**Estudi de viabilitat**	**27 days**	**Thu 18/10/12**	**Mon 26/11/12**	**2**	
7	**Especificació del sistema d'adqui**	**6 days**	**Thu 18/10/12**	**Thu 25/10/12**		
8	Especificació del tipus d'il·luminaci	5 days	Thu 18/10/12	Wed 24/10/12		tecnic
9	Especificació del tipus de tarja digi	1 day	Thu 25/10/12	Thu 25/10/12	8	tecnic
10	Predisseny armari d'inspecci	3 days	Fri 26/10/12	Tue 30/10/12	9	becari[50%]
11	**Estudi detecció defectes**	**16 days**	**Fri 26/10/12**	**Mon 19/11/12**	**7**	
12	Generació banc de proves	6 days	Fri 26/10/12	Mon 05/11/12		becari[50%]
13	Estudi dels defectes	10 days	Tue 06/11/12	Mon 19/11/12	12	director de projecte[
14	Document especificació solucions sof	1 day	Tue 20/11/12	Tue 20/11/12	11	
15	**Redacció del document de pressi**	**4 days**	**Tue 20/11/12**	**Fri 23/11/12**	**11**	
16	Avaluació cost del Hardwar	4 days	Tue 20/11/12	Fri 23/11/12		tecnic
17	Planificació realització del project	2 days	Tue 20/11/12	Wed 21/11/12		director de projecte
18	Pressupost	1 day	Mon 26/11/12	Mon 26/11/12	15	
19	**Realització del projecte**	**37 days**	**Tue 27/11/12**	**Thu 31/01/13**	**6**	
20	Disseny armari d'inspecció	8 days	Tue 27/11/12	Mon 10/12/12		tecnic[150%]
21	Muntatge elements d'inspecci	8 days	Tue 11/12/12	Thu 20/12/12	20	becari[50%]
22	Codificació detecció defecte	21 days	Fri 21/12/12	Thu 31/01/13	21	tecnic[50%];becari
23	**Fase Final**	**10 days**	**Fri 01/02/13**	**Thu 14/02/13**	**19**	
24	Documentació	2 days	Fri 01/02/13	Mon 04/02/13		
25	**Instal·lació en línea de producció**	**10 days**	**Fri 01/02/13**	**Thu 14/02/13**		
26	Muntatge del sistema	3 days	Fri 01/02/13	Tue 05/02/13		tecnic;becari
27	Fase de sintonització	4 days	Wed 06/02/13	Mon 11/02/13	26	tecnic[200%]
28	Depuració d'errors	3 days	Tue 12/02/13	Thu 14/02/13	27	becari

Il·lustració 11 – Relació Tasca-personal

Com es pot observar tenim el nom de la tasca, la duració que tindrà en dies, el dia que ha de començar la tasca i el dia que s'hauria de tenir finalitzada, si te predecessor o no i qui l'ha de portar a cap. I per acabar comprovem que tot esta correcte amb el diagrama de Gantt ("View → Gantt Chart"):



Il·lustració 12 – Calendari temporal

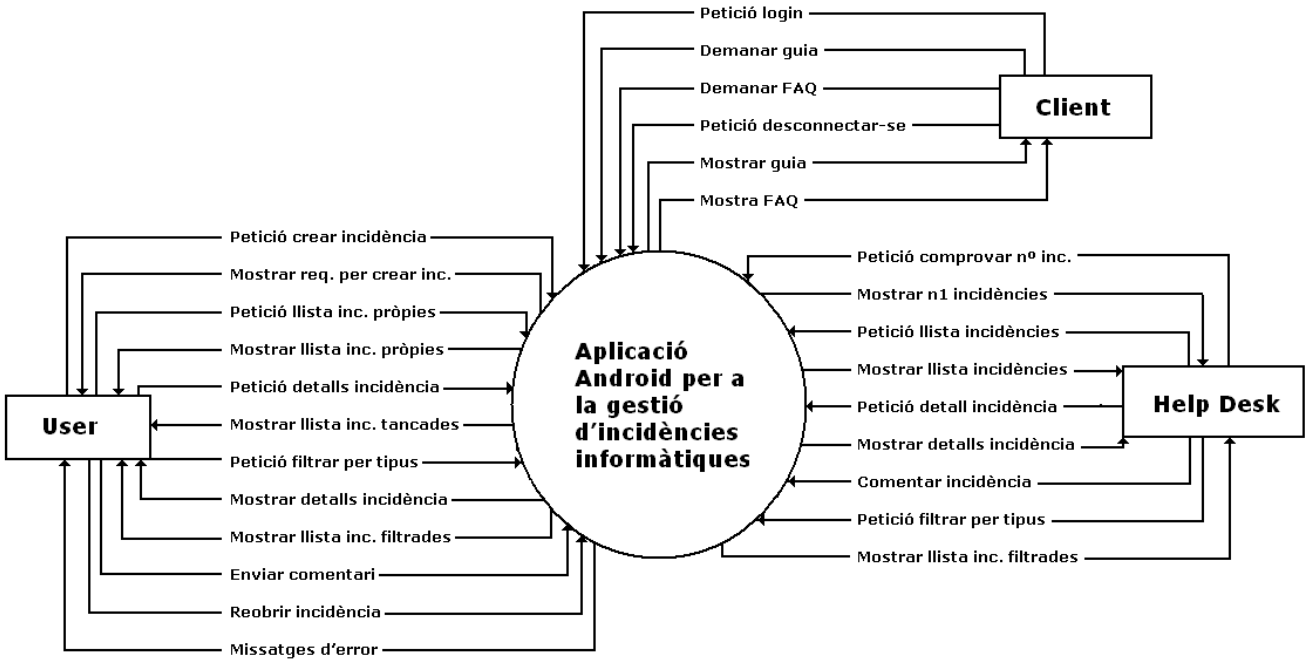
2. Anàlisis Funcional

2.1. Introducció

En aquest capítol es realitza un estudi de la informació prèvia al desenvolupament del projecte. Es durà a terme un diagrama de context i un anàlisis dels casos d'ús.

2.2. Diagrama de context

En el diagrama de context es caracteritzen totes las interaccions que realitza un sistema amb el seu entorn, aquestes poden ser altres sistemes, sectors interns a la organització, o factors externs a aquesta.



Il·lustració 13 – Diagrama de context

2.3. Requisits del sistema

En aquest apartat es parlaran de tots els requisits que ha de complir el projecte per a que tingui èxit, ja siguin funcionals (aquells que el servei ha de proporcionar, com ha reaccionar el sistema i com s'ha de comportar), no funcionals (serveis o funcions ofertes per el sistema) o restriccions del sistema (restriccions que provenen del domini de l'aplicació i que reflexa les seves característiques).

2.3.1. Requisits funcionals

Els requisits funcionals del projecte seran tots aquelles funcions que la aplicació ofereix com per exemple crear una indecència(1), veure el FAQ(2), veure una guia d'ús de l'aplicació(3), consultar incidències pròpies(4), veure detalls de les

incidències pròpies(5), consultar incidències ja resoltes(6), consultar detalls de les incidències resoltes(7), reobrir una incidència pròpia(8), comentar una incidència pròpia(9), veure la llista d'incidències i filtrar-les per tema(10), comprovar el número diincidències(11) i desconnectar de l'aplicació(12).

2.3.2. Requisits no funcionals

Els requisits no funcionals al tindre una base de dades extensa, ha de tindre una seguretat per a que les dades dels usuaris no s'utilitzin fora de l'aplicació(1). Ha de tindre també un control de entrades per evitar usuaris amb males intencions(2). Un altre requeriment seran els recursos ajustats a l'entitat(3), ja que facilitarà l'ús i el treball dels usuaris i com a últim recurs el compliment de LOPD(4).

2.3.3. Restriccions del sistema

Les restriccions del sistema son que l'aplicació estarà desenvolupada a Windows i ha de tindre una base de dades accessible per a tothom.

2.3.4. Catalogació i priorització dels requisits

Taules amb els requisits prioritzats: essencial, condicional, opcional.

Taules que relacionin els requisits amb els objectius del projecte.

	RF 1	RF 2	RF 3	RF 4	RF 5	RF 6	RF 7	RF 8	RF 9	RF1 0	RF1 1	RF1 2
Essencial	X			X	X	X	X	X				X
Condicional		X							X	X		
Opcional			X								X	

Il·lustració 14 – Requisits funcionals

	RNF1	RNF2	RNF3	RNF4
Essencial	X	X	X	
Condicional				X
Opcional				

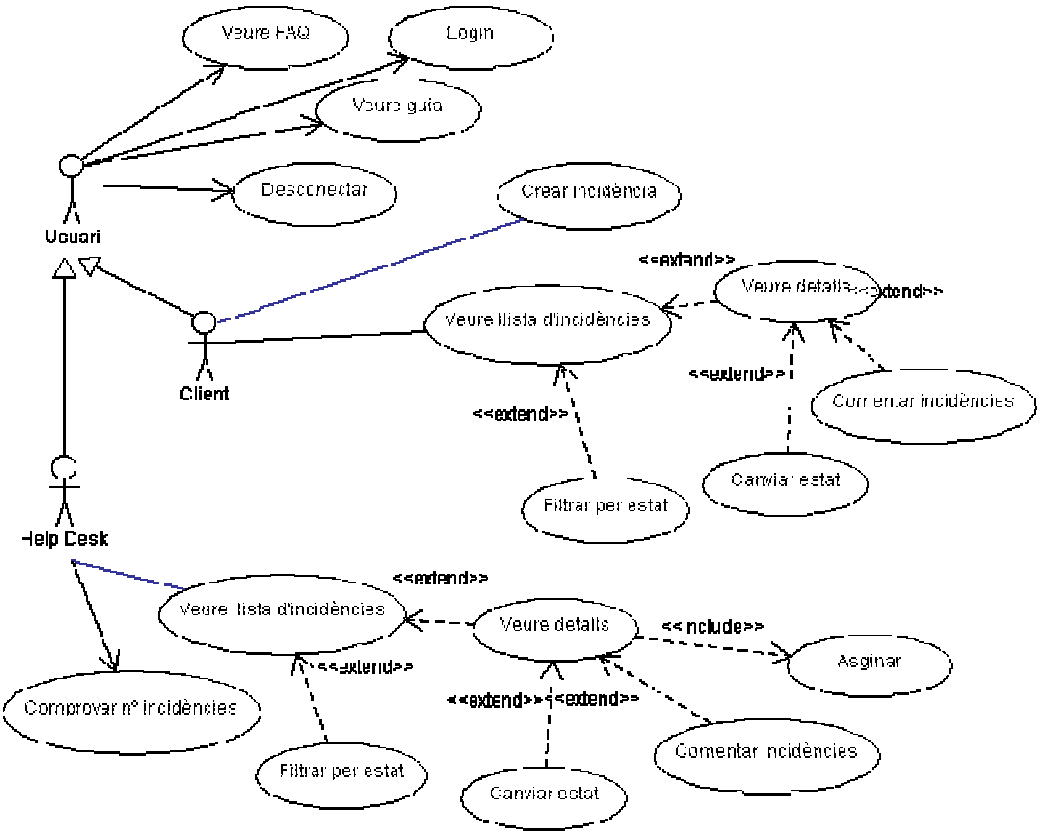
Il·lustració 15 – Requisits no funcionals

Llegenda:

- **Essencial**. Tots els requisits que el projecte ha de tindre obligatòriament.
- **Opcional**. Tots els requisits que farien del projecte un bon ús però que no son completament necessaris.
- **Condicional**. Tots els requisits que millorarien la condició d'ús de l'usuari amb el projecte.

2.4. Casos d'us

A l'aplicació intervenen dos tipus d'usuaris que engloben a usuari que són, el client i el Help Desk. Tots dos tenen funcionalitats en comú, definides en l'actor usuari, i també tenen funcionalitats pròpies.



Il·lustració 16 – Casos d'us

2.4.1. Detall Casos d'us

2.4.1.1. Cas d'us Login

Login	
Descripció	Permet que l'usuari es connecti a l'aplicació.
Actor	Usuari
Pre condicions	Cap
Post condicions	Es mostra per pantalla la confirmació o error.
Flux seqüència normal	1. L'usuari inicia l'aplicació. 2. No hi ha dades d'inici de sessió prèvia. 3. L'aplicació mostra la pantalla de

	login a l'usuari. 4. L'usuari introdueix les seves credencials. 5. Si les credencials són correctes s'inicia l'aplicació. En cas contrari s'executa la seqüència alternativa A.
Flux seqüència alternativa	A. Error en les credencials 1. L'aplicació indica a l'usuari que hi ha un error en les credencials. 2. Es torna a executar el cas d'us.

2.4.1.2. Cas d'us Desconnectar

Desconnectar	
Descripció	Permet que l'usuari es desconnecti a l'aplicació.
Actor	Usuari
Pre condicions	Sessió prèviament iniciada
Post condicions	Es mostra per pantalla la confirmació.
Flux seqüència normal	1. L'usuari indica que vol finalitzar la sessió. 2. L'aplicació finalitza la sessió. 3. Es torna a executar el cas d'us Login.

2.4.1.3. Cas d'us Veure FAQ

Veure FAQ	
Descripció	Permet a l'usuari veure una llista de les preguntes més freqüents.
Actor	Usuari
Pre condicions	Sessió prèviament iniciada
Post condicions	Es mostra per pantalla les preguntes.

Flux seqüència normal	1. L'usuari indica que vol veure el FAQ. 2. L'aplicació mostra la llista de preguntes més freqüents amb la seva resposta.

2.4.1.4. Cas d'us Veure Guia

Veure Guia	
Descripció	Permet a l'usuari veure una guia de l'aplicació.
Actor	Usuari
Pre condicions	Sessió prèviament iniciada.
Post condicions	Es mostra per pantalla la guia.
Flux seqüència normal	1. L'usuari indica que vol veure la guia. 2. L'aplicació mostra la guia de l'aplicació per pantalla.

2.4.1.5. Cas d'us Crear incidència

Crear incidència	
Descripció	Permet al client crear una nova incidència per a que sigui resolta per els Help Desk.
Actor	Client
Pre condicions	Sessió prèviament iniciada. Tenir permisos de Client. Que la incidència sigui omplerta en tots els camps.
Post condicions	Cap
Flux seqüència normal	1. El client indica que vol crear una nova incidència. 2. L'aplicació mostra per pantalla les dades a omplir. 3. El client omple tots els camps(assumpte, tipus del problema i una petita descripció

	del problema). En cas que no s'ompli tots els camps s'executa la seqüència alternativa A. 4. El sistema crea la incidència. En cas de que existeixi un error s'executa la seqüència alternativa B.
Flux seqüència alternativa	A. No s'han omplert tots els camps. 1. L'aplicació avisa que s'han d'omplir tots els camps. B. Ha ocorregut un error en crear la incidència. 1. El sistema avisa a l'usuari.

2.4.1.6. Cas d'us Comentar incidències

Comentar incidències	
Descripció	Permet a l'usuari comentar les incidències.
Actor	Client, Help Desk
Pre condicions	Sessió prèviament iniciada. Accedir prèviament al cas d'us Veure detalls.
Post condicions	
Flux seqüència normal	1. L'usuari indica que vol afegir un comentari. 2. L'aplicació envia el comentari i actualitza la pantalla .

2.4.1.7. Cas d'us Veure detalls

Consultar Veure detalls	
Descripció	Permet a l'usuari veure els detalls de les incidències.
Actor	Client, Help Desk
Pre condicions	Sessió prèviament iniciada.

Post condicions	
Flux seqüència normal	1. L'usuari indica vol consultar els detalls de la incidència. 2. L'aplicació mostra els detalls de la incidència seleccionada, aquest detalls son la id de la incidència, l'assumpte, els detalls basics(creador i tipus), la descripció del problema i els comentaris. En cas de que hagi un error s'executa la seqüència alternativa A.
Flux seqüència alternativa	A. Ha ocorregut un error en crear la incidència. 1. El sistema avisa a l'usuari.

2.4.1.8. Cas d'us canviar estat

Reobrir incidència pròpia	
Descripció	Permet al client i al Help Desk canviar l'estat de la incidència, així el client, per exemple, no ha d'obrir una nova incidència, i el Help Desk la tancarà.
Actor	Client, Help Desk
Pre condicions	Sessió prèviament iniciada. Accedir prèviament al cas d'us Veure detalls.
Post condicions	Cap
Flux seqüència normal	1. L'usuari indica que vol canviar l'estat de la incidència. 2. L'aplicació canvia l'estat de la incidència. En cas de que hagi un error s'executa la seqüència alternativa A.
Flux seqüència alternativa	A. Ha ocorregut un error en canviar l'estat. 1. El sistema avisa a l'usuari.

2.4.1.9. Cas d'us Comprovar Nº d'incidències

Comprovar Nº d'incidències	
Descripció	Permet al Help desk de comprovar el número d'incidències que hi ha, ja siguin noves o pendents de resoldre.
Actor	Help Desk
Pre condicions	Sessió prèviament iniciada. Tenir permisos de Help Desk.
Post condicions	
Flux seqüència normal	1. L'aplicació mostra la llegenda de les incidències noves i pendents de resoldre. En cas de que hagi un error s'executa la seqüència alternativa A.
Flux seqüència alternativa	A. Ha ocorregut un error en crear la incidència. 1. El sistema avisa a l'usuari.

2.4.1.10. Cas d'us Veure llista d'incidències

Veure llista d'incidències	
Descripció	Permet al Help Desk de veure totes les incidències en un llistat, ja estiguin en estat de noves, tancades o pendents.
Actor	Client, Help Desk
Pre condicions	Sessió prèviament iniciada. Tenir permisos de Help Desk.
Post condicions	
Flux seqüència normal	1. L'aplicació mostra totes les incidències. En cas de que hagi un error s'executa la seqüència alternativa A.
Flux seqüència alternativa	A. Ha ocorregut un error en crear la incidència. 1. El sistema avisa a l'usuari.

2.4.1.11. Cas d'us Filtrar per Estat

Filtrar per nova	
Descripció	Permet al Help Desk filtrar la llista d'incidències per noves, pendents o resoltes.
Actor	Client, Help Desk
Pre condicions	Sessió prèviament iniciada. Cas d'us Veure llista d'incidències.
Post condicions	
Flux seqüència normal	1. El Help Desk vol filtrar les incidències per estat. 2. L'aplicació mostrarà només aquelles incidències que l'usuari seleccioni. En cas de que hagi un error s'executa la seqüència alternativa A.
Flux seqüència alternativa	A. Ha ocorregut un error en crear la incidència. 1. El sistema avisa a l'usuari.

2.4.1.12. Cas d'us Asignar

Filtrar per nova	
Descripció	L'aplicació assignarà la incidència oberta al primer Help Desk que la obri, fent que es canviï l'estat de la incidència a pendent.
Actor	Help Desk
Pre condicions	Sessió prèviament iniciada. Cas d'us Veure llista d'incidències.
Post condicions	
Flux seqüència normal	1. El Help Desk veu els detalls d'una incidència oberta. 2. L'aplicació assigna automàticament la incidència al Help Desk.

3. Disseny Tècnic

En aquest capítol es descriu tot el que te a veure amb la interfície de l'usuari, els detalls de l'estructura i l'estructura de la navegació de l'aplicació.

3.1. MVC

Com és ben sabut el patró MVC (Model-Vista-Controlador) és el patró més seguit en aplicacions amb interfície d'usuari, per aquest motiu l'aplicació segueix una de les evolucions que han sorgit al llarg del temps del patró MVC.

3.1.1. Teoria patró MVC

El MVC es un patró d'arquitectura de software que separa les dades i la lògica d'una aplicació de la interfície de l'usuari i el mòdul encarregat de gestionar els esdeveniments i les comunicacions. El MVC proposa la construcció de tres components diferents que son el model, la vista i el controlador, és a dir, es defineixen les components per a la presentació per una banda i la interacció de l'usuari per un altra. Aquest patró es basa en les idees de reutilització de codi i la separació de conceptes, característiques que busquen facilitar les tasques de desenvolupament d'aplicació i el seu manteniment.

3.1.2. Peces del patró MVC

Com hem esmentat anteriorment, el MVC es basa en tres components.

3.1.2.1. Model

Es la representació de la informació amb la qual el sistema opera, per tant, gestiona tots els accessos a aquesta informació, ja siguin consultes, com actualitzacions, implementant també els privilegis d'accés que s'hagin descrit en les especificacions de l'aplicació (lògica del negoci). Envia a la Vista aquella part de la informació que en cada moment es sol·licita per a que sigui mostrada. Les peticions d'accés o la manipulació d'informació arriben al Model a través del Controlador.

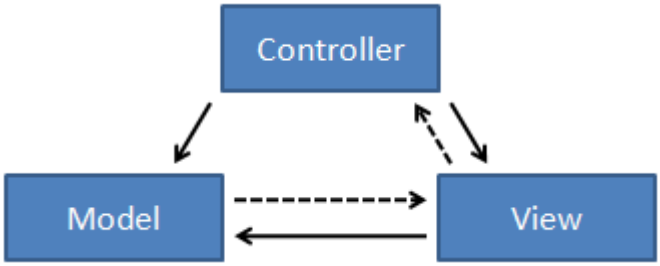
3.1.2.2. Vista

Presenta el Model (informació i lògica del negoci) en un format adequat per interactuar (usualment amb la interfície de l'usuari) per tant, requereix d'aquest "Model" la informació que ha de representar com a sortida.

3.1.2.3. Controlador

Respon a esdeveniments (accions de l'usuari) i invoca peticions al Model"quan es fa alguna sol·licitud sobre la informació. També pot enviar comandes a la seva Vista associada si es sol·licita un canvi en la forma en que es presenta el Model, per tant es podria dir que el Controlador fa d'intermediari entre "Vista" i Model.

3.1.3. Diagrama de classes del patró MVC



Il·lustració 17 – Diagrama de classes simple del patró MVC

Les fletxes contínues representen una associació directe. El component del qual surt la fletxa pot actuar directament sobre el component al que apunta la fletxa, és a dir, actua sobre una o més instàncies que estan contingudes al primer component.

Les fletxes discontínues representen una associació indirecta, és a dir, que el component del qual surt la fletxa no té per què contenir el component a què apunta la fletxa, sinó que hi està associat amb algun mecanisme intermedi.

Associacions directes (línies continues):

- Del controlador al model: El controlador ha de poder accedir al model per veure'n l'estat a fi de prendre decisions i també ha de poder modificar-lo o cridar a mètodes que el modifiquin.

- Del controlador a la vista: El controlador ha de conèixer totes les possibles vistes a fi de seleccionar la vista adient i carregar-la on sigui necessari. El controlador no modifica les vistes, únicament les gestiona i les carrega. Les vistes es modifiquen a si mateixes en funció de l'estat del model.

- De la vista al model: La vista ha d'accedir al model per poder extreure'n dades. És important tenir en compte que les vistes no haurien de poder modificar el model.

Associacions indirectes(línies discontinues):

- Del model a la vista: L'associació és necessària sempre que es vulgui poder notificar a les vistes quan hi ha un canvi en el model. Tot i això, l'associació no ha de ser directa, és a dir, el model no conté les vistes; sinó que les vistes es registren a uns determinats esdeveniments. Quan el model canvia, el model notifica aquest esdeveniment a les vistes, però el model no és conscient que està notificant una vista. Això es pot fer mitjançant el patró Observer.

- De la vista al controlador: La vista generalment conté elements interactius (botons, enllaços, etc.) que en ser activats han de canviar la vista o bé interactuar amb el controlador. Això, però, no implica que la vista hagi de contenir el controlador, sinó que hi ha algun element extern (el sistema operatiu, el navegador, etc.) que escolta aquest tipus d'esdeveniments d'usuari i crida al controlador segons el que s'especifica a la vista.

3.1.4. Interacció de les components

Tot i que hi pot haver diferents implementacions, el flux seqüencial del patró MVC acostuma a ser el següent:

1. L'usuari interactua amb la interfície de l'usuari d'alguna manera.
2. El controlador rep la notificació de l'acció sol·licitada per l'usuari. El controlador gestiona aquest esdeveniment modificant el model en funció de l'estat actual del model i el propi esdeveniment.
3. El controlador accedeix al model, actualitzant, possiblement modificant-lo de forma adequada a l'acció sol·licitada per l'usuari. Els controladors complexes estan sovint estructurats utilitzant un patró de comanda que encapsula les accions i simplifica l'extensió.
4. El controlador delega als objectes de la vista la tasca de desplegar la interfície d'usuari. La vista obté les seves dades del model per a generar la interfície apropiada per a l'usuari on es reflecteixen els canvis en el model.
5. La interfície d'usuari espera noves interaccions de l'usuari, començant el cicle de nou.

S'ha de tenir en compte que depenent de com sigui la interfície d'usuari pot ser que el flux variï en funció del següent:

- Multiplicitat en el nombre de vistes existents. Es refereix al nombre de vistes que hi ha en el patró:
  - Una vista que representa el model. És el cas d'algunes aplicacions Web i d'escriptori.
  - Diverses vistes que representen parts del model. És el cas de les aplicacions Web i de les aplicacions mòbils.
- Multiplicitat en el nombres de vistes mostrades. Es refereix al nombre de vistes que es mostren simultàniament:
  - Una vista: És el cas de la majoria de les aplicacions Web i de pràcticament totes les aplicacions mòbils.
  - Més d'una vista. En el cas d'una gran part de les aplicacions d'escriptori.

En funció d'això s'hauria d'adaptar el patró a l'entorn.

3.1.5. MVC i les bases de dades

Molts sistemes informàtics utilitzen un sistema de gestió de base de dades per a gestionar les dades que ha d'utilitzar l'aplicació, aquesta gestió correspon al component Model del MVC. La unió entre capa de presentació i capa de negoci conegut en el paradigma de la programació per capes representaria la integració enter la Vista i el Controlador d'esdeveniments i accés de dades. El que pretén MVC es separar la capa visual gràfica de la programació i accés a dades, alguna cosa que millora el desenvolupament i manteniment de la Vista i el Controlador.

3.1.6. Ús de MVC en aplicacions Web

Encara que originalment el patró MVC va ser desenvolupat per aplicacions d'escriptori, ha sigut adaptat com a arquitectura per a dissenyar i implementar

aplicacions web en els principals llenguatges de programació. S'han desenvolupat multitud de "frameworks" que implementen aquest patró, aquest "frameworks" es diferencien bàsicament en la interpretació de com les funcions MVC es divideixen entre client i servidor.

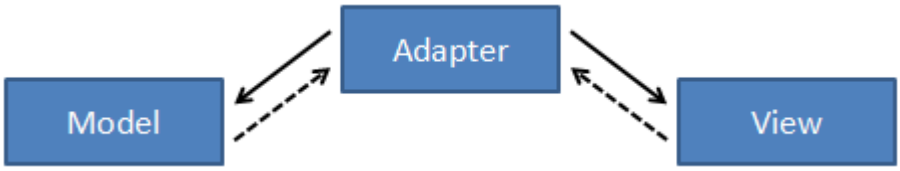
3.2. MVA

Com s'ha esmentat en l'apartat 3.1, l'aplicació segueix una de les evolucions que han sorgit al llarg del temps del patró MVC. Aquest es el patró MVA o Model-Vista-Adaptador.

3.2.1. Teoria patró MVA

MVA és un patró d'arquitectura i arquitectura de diversos nivells, que s'utilitza en enginyeria de programació. En les aplicacions informàtiques complexes que presenten grans quantitats de dades als usuaris, els desenvolupadors sovint volen separar les dades (model) i la interfície d'usuari (vista), per a que els canvis en la interfície d'usuari no afectin el maneig de dades i que les dades es puguin reorganitzar sense canviar la interfície d'usuari. Tant MVA com el tradicional MVC intenten resoldre aquest mateix problema, però amb dos estils diferents de solució. El tradicional MVC organitza el Model, Vista i el Controlador en un triangle, amb el model, vista i controlador com a vèrtexs. El MVA soluciona aquest problema diferent a MVC, disposant d'un controlador de mediació (Adaptador) provocant que no hi hagi connexió directa alguna entre el model i la vista.

3.2.2. Diagrama de classes MVA



Il·lustració 18 – Diagrama de classes MVA

Associacions directes:

- Del Adaptador al Model: L'Adaptador envia els esdeveniments d'acció rebuts de la Vista cap al Model.
- Del Adaptador a la Vista:L'Adaptador envia els esdeveniments de canvi rebuts del Model cap a la Vista.

Associacions indirectes:

- Del Model a l'Adaptador: El Model s'encarrega d'enviar a l'Adaptador els esdeveniments de canvi de propietat.
- De la vista a l'Adaptador: La Vista s'encarrega d'enviar a l'Adaptador els esdeveniments d'acció.

L'adaptador es completament responsable de mantenir el Model i la Vista en sincronització. Tant el Model, com la Vista no saben res l'un de l'altre.

Els avantatges per a l'organització de codi d'aquesta manera són:

- Totes les "parts mòbils" estan centralitzades en un sol lloc, l'Adaptador.
- La separació de les preocupacions entre la vista i l'adaptador. La vista és responsable de maquetació i la presentació visual, mentre que l'adaptador s'encarrega de la sincronització i els aspectes dinàmics de la interfície d'usuari.

- Millor desacoblament entre models i vistes. En concret, la vista no ha de saber res sobre el model.

3.3. Tecnologies a utilitzar

L'aplicació, al ser per plataforma Android, es desenvolupada amb el llenguatge de programació Java i el conjunt d'eines de desenvolupament (SDK), encara que hi ha altres opcions disponibles. Android es un sistema operatiu basat en Linux, dissenyat principalment per a dispositius mòbils amb pantalla tàctil com telèfons intel·ligents o tablets. S'ha escollit programar en Android ja que ha passat a ser una de les preferides pels desenvolupadors per a plataformes mòbils i es recomanable.

3.3.1. Història Android

Va se desenvolupat inicialment per Android Inc, una firma comprada per Google en 2005. Es el principal producte de l'Open Handset Alliance, un conglomerat de fabricants i desenvolupadors de hardware, software i operadors de servici.

Té una gran comunitat de desenvolupador escrivint aplicacions per estendre la funcionalitat de dispositius. Google Play es la tenda d'aplicacions en línia administrada per Google, encara que existeix la possibilitat d'obtenir software externament. Els programes estan escrits en el llenguatge de programació Java. No obstant, no es un sistema operatiu lliure de malware, encara que la majoria es descarregat de tercers.

L'estructura del sistema operatiu Android es composa d'aplicacions que s'executen en un framework Java d'aplicacions orientades a objectes sobre el nucli de les biblioteques de Java en una màquina virtual Dalvik amb compilació en temps d'execució.

3.3.1.1. Adquisició per part de Google

En juliol de 2005 Google va adquirir Android inc., una petita companyia fundada en 2003. Els cofundadors de la Android Inc, que es van anar a treballar a Google, van ser Andy Rubin, Rich Miner, Nick Sears i Chris White.

En Google l'equip liderat per Rubin va desenvolupar una plataforma per a dispositius mòbils basada en el nucli Linux que va ser promocionat a fabricants de dispositius i operadors amb la promesa de promoure un sistema flexible i actualitzable.

3.3.1.2. l'Open Handset Alliance

El 5 de novembre de 2007 la Open Handset Alliance, un consorci de diverses companyies, es va estrenar amb el fi de desenvolupar estàndards oberts per a dispositius mòbils. Aquesta va estrenar el seu primer producte Android construïda sobre la versió 2.6 de Linux.

3.3.2. Actualitzacions Android

Des de l'inici d'Android s'han vist diverses actualitzacions, aquestes actualitzacions al sistema operatiu solen ser per arreglar "bugs" i agregar noves funcions. Cada actualització es desenvolupada sota un nom de codi d'un element relacionat amb postres en ordre alfabètic. Android es considerada un dels elements

promotor de la obsolescència programada a causa de les aparicions de noves versions que, en molts casos, no arriben a funcionar correctament en el hardware dissenyat per a versions prèvies.

3.3.3. Característiques

Les característiques i especificacions de les que disposa Android actualment son:

- Disseny de dispositiu: La plataforma es adaptable a pantalles amb major resolució.
- Emmagatzematge: SQLite es utilitzada per a propòsits de emmagatzematge de dades.
- Conectivitat: Suporta diverses tecnologies de connectivitat (Wi-Fi, HSDPA, HSPA+, etc).
- Missatgeria: SMS i MMS son formes de missatgeria, incloent la Android Cloud to Device Messaging Framework(C2DM).
- Navegador Web.
- Suport de Java: Encara que la majoria de les aplicacions estan escrites en Java, no hi ha una màquina virtual Java en la plataforma. El bytecode Java primer es compila en un executable Dalvik i corre en una màquina virtual.
- Suport multimèdia: Suporta diversos formats multimèdia(3GP, MP4, PNG, etc).
- Suport per streaming: Streaming RTP/RTSP, descàrrega progressiva de HTML. Adobe Flash Streaming es suportat a través de Adobe Flash Player.
- Suport per a hardware addicional: Càmeres de fotos, de vídeo, GPS, sensors, etc.
- Entorn de desenvolupament: Inclou un emulador de dispositius, eines per a la depuració de memòria i anàlisis del rendiment del software.
- Google Play: Es un catàleg d'aplicacions gratuïtes o de pagament, que poden ser descarregades i instal·lades sense la necessitat d'un PC.
- Multi-tàctil: Android té suport natiu per a pantalles capacitives amb suport multi-tàctil.
- Video-trucada: Android suporta videotrucada a través de Google Talk des de la seva versió HoneyComb.
- Multi-tasca: Multi-tasca real d'aplicacions.
- Característiques basades en veu: Està disponible la cerca en Google a través de la veu.
- Tetherin: Android suporta tethering, que permet al telèfon ser usar com un punt d'accés amb filferro o sense fil.

3.3.4. Arquitectura

Les components principals del sistema operatiu Android son:

- Aplicacions: Les aplicacions base inclouen un client de correu electrònic, programa de SMS, calendari, etc. Totes les aplicacions estan escrites en llenguatge de programació Java.

- Marc de treball d'aplicacions: Els desenvolupadors tenen accés complet als mateixos APIs del frameworks utilitzats per les aplicacions base. La arquitectura esta dissenyada per a simplificar la reutilització de components. Aquest mateix mecanisme permet que els components siguin reemplaçats per l'usuari.

- Biblioteques: Android inclou un conjunt de biblioteques de C/C++ utilitzades per diversos components del sistema. Aquestes característiques s'exposen als desenvolupadors a través del marc de treball d'aplicacions.

- Runtime d'Android: Android inclou un set de biblioteques base que proporcionen la major part de les funcions disponibles de les biblioteques base del llenguatge Java. Cada aplicació corre el seu propi procés, amb la seva pròpia instancia de la màquina virtual Dalvik.

- Nucli Linux: Android depèn de Linux per als servicis base del sistema com seguritat, gestió de memòria i de processos, pila de xarxa i model de controladors. També actua com una capa d'abstracció entre el hardware i la resta de la pila de software.

3.3.5. Android SDK

El desenvolupament de programes per Android es fa habitualment amb el llenguatge de programació Java i el conjunt d'eines de desenvolupament SDK.

El SDK (Software Development Kit) d'Android, inclou un conjunt d'eines de desenvolupament. Compren un depurador de codi, biblioteca, un simulador de telèfon basat en QEMU, documentació, exemples de codi i tutorials. Les plataformes de desenvolupament suportades inclouen Linux, Max OS X 10.4.9 o posterior i Windows XP o posterior. La plataforma integral de desenvolupament (IDE, Integrated Development Environment) suportada oficialment es Eclipse juntament amb el complement ADT (Android Development Tools plugin), encara que també es pot utilitzar un editor de text per escriure fitxers Java y Xml i utilitzar comandes en un terminal per a crear i depurar aplicacions.

Les actualitzacions del SDK estan coordinades amb el desenvolupament general d'Android. El SDK suporta també versions antigues d'Android, per si els programadors necessiten instal·lar aplicacions en dispositius ja obsolets o més antics. Les eines de desenvolupament son components que es poden descarregar, així un cop instal·lada l'última versió, poden instal·lar versions anteriors i poden fer proves de compatibilitat.

Una aplicació Android està composta per un conjunt de fitxers empaquetats en format ".apk" i desada en el directori /data/app del sistema operatiu Android. Un paquet APK inclou fitxers ".dex", recursos, etc.

3.4. Guia d'estil

La guia d'estil que segueix l'aplicació no és més que la guia que ha de seguir qualsevol aplicació Android, segons recomana la pròpia pàgina de desenvolupadors d'Android, la qual recomana seguir una sèrie d'objectius i de pautes per a fer una bona aplicació. A continuació parlarem de les pautes i objectius que segueix aquesta aplicació.

3.4.1. Objectius principals

El disseny d'Android està enfocat a tres objectius generals que s'apliquen a les aplicacions bàsiques. El tres objectius son:

1. M'encanta

   Les Apps Android han de ser elegants i estèticament agradables en múltiples nivells. Les transicions han de ser ràpides i clares, i el disseny i la tipografia clars i significatius. Alhora que una eina ben feta, la seva aplicació ha d'esforçar-se per combinar la bellesa, la senzillesa i el propòsit de crear una experiència màgica.

2. Simplificar la vida

   Les Apps Android fan la vida més fàcil i són fàcils d'entendre. Quan la gent utilitza una aplicació per primera vegada ha de captar intuïtivament les característiques més importants. Les tasques simples no han de requerir procediments complexes, i les tasques complexes s'han d'adaptar a la mà i la ment humana. Qualsevol persona, independent de la seva edat i cultura, s'han de sentir fermament en control, i mai sentir-se aclaparats per massa opcions.

3. Fes-me increïble

   Els apartats anterior no són suficients per a fer una aplicació fàcil d'utilitzar. Apps Android dona poder a les persones a provar coses noves i utilitzar aplicacions en noves formes innovadores. Android permet que les persones combinin aplicacions en nous fluxos de treball a través de la multitasca, notificacions i compartir a través d'aplicacions.

3.4.2. Dispositius i pantalles

Com que Android, actualment, s'utilitza en milions de telèfons, tablets i altres dispositius, i que existeix una gran varietat de mides de pantalles i formes, s'ha de prendre avantatge fent que l'aplicació sigui flexible (Estirar i comprimir el disseny per adaptar-se a diferents amplades i llargades), optimitzant el disseny(Aprofitant l'espai extra de la pantalla en els casos dels dispositius més grans) i proporcionar recursos per a diferents densitats de pantalla, per assegurar-se de que l'aplicació es vegi bé en qualsevol dispositiu.

3.4.3. Temes

Els temes són mecanisme d'Android per aplicar un estil coherent amb una aplicació. L'estil del tema especifica les propietats visuals dels elements que composen la interfície de l'usuari, com el color, l'altura, la mida de la font i el contingut. Android ofereix tres temes a l'hora de crear una aplicació:

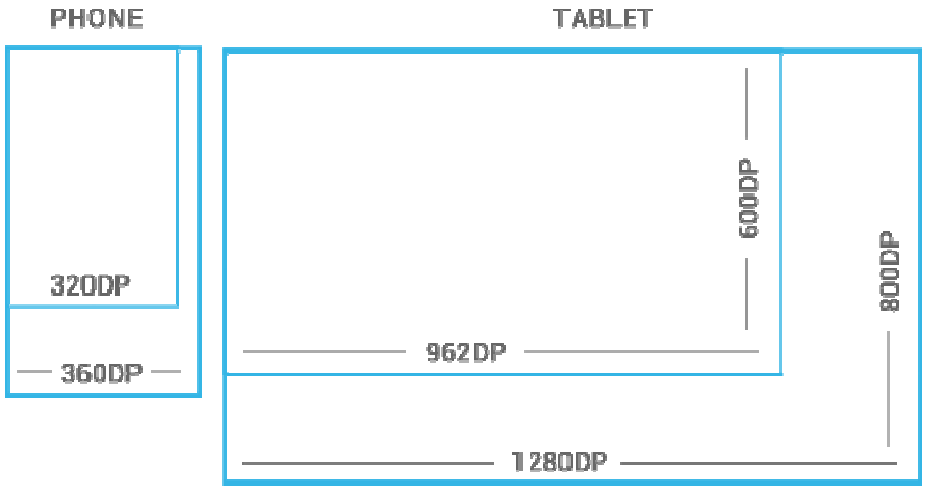
- Holo Light
- Holo Dark
- Holo Light with dark action bars

En aquesta aplicació s'ha escollit l'última de les opcions, "Holo light with dark action bars", ja que d'aquesta forma tenim una visió més clara del contingut i donem a destacar les action bars amb un color fosc per tal de que es vegin amb més claredat.

3.4.4. Mides

Android ens avisa que els dispositius varien, no només en grandària física, sinó també en la densitat de la pantalla. Per simplificar la forma de dissenyar per a múltiples pantalles, s'ha pensar en que cada dispositiu té una mida i una densitat determinada.

Les mides són, per a telèfons, inferiors als 600dp i, per tablets, superiors o iguals als 600dp.



Il·lustració 19 – Mides dels dispositius

S'ha d'optimitzar la interfície d'usuari de l'aplicació mitjançant els dissenys alternatius per a alguns dels diferents grups de mida, i proporcionar imatges de mapa de bits alternatius per als diferents cubs de densitat. Perquè és important dissenyar i implementar els dissenys per a diverses densitats, per això es recomana que les mesures de l'aplicació siguin amb "dp" en lloc de píxels.

L'aplicació està, actualment, dissenyada de forma per a que s'executi perfectament en qualsevol dispositiu i s'aprofiti tot l'espai, i d'aquesta forma l'aplicació es mantingui neta i senzilla.

3.4.5. Escriptura

Android recomana escollir bé les paraules que han de aparèixer a l'aplicació i ens dona a seguir uns petits consells:

- Sigui breu. Sigui concís, senzill i precís.
- Sigui simple. Utilitzeu paraules curtes, verbs actius, i els noms comuns.

- Sigui amigable. Fes servir contraccions. Parli directament.
- Posa primer el més important. Les dues primeres paraules han d'incloure almenys una mostra de la informació més important en la cadena.
- Descriu només el necessari, i no més.
- Evita la repetició.

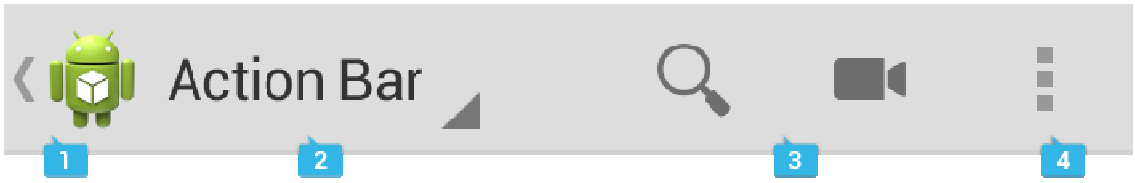
3.4.6. Action Bars

La barra d'accions és una peça posicionada a la part superior de cada pantalla que generalment és persistent al llarg de l'aplicació.

Ofereix diverses funcions clau:

- Realitza accions prominents importants i accessibles d'una manera predictible.
- Suporta navegació consistent i veure el canvi dins de les aplicacions.
- Redueix el desordre, proporcionant un excés d'acció per a les accions que poques vegades s'utilitzen.
- Proporciona un espai dedicat per donar a la seva aplicació una identitat.

Android recomana utilitzar les barres d'accions, ja que és un dels elements de disseny més importants per a posar en pràctica i la forma que ha de tenir s'ha de dividir en quatre zones.



Il·lustració 20 – Action Bar

1. La primera zona és la del logotip de l'aplicació. Com podem observar a l'esquerra del logotip hi ha una fletxa apuntant cap a l'esquerra, aquesta, si és activada, ens permetrà tornar a la pantalla anterior.
2. La segona zona és la del control. Aquesta zona ha mostrar, o bé la part de l'aplicació en la que ens trobem, és a dir, el títol, o bé un control de vista, és a dir, un menú desplegable (Spinners) o un control de pestanyes.
3. La tercera zona és on han de aparèixer les accions més importants, és a dir, les que es vagin a utilitzar més sovint. Si mantenim pressionat una de les accions ens ha d'aparèixer el nom d'aquesta.
4. La quarta i última zona és un desplegable on s'han de trobar les accions menys utilitzades. Aquestes accions les hem de moure nosaltres dins del desplegable.

En aquesta aplicació, al no haver-hi moltes accions a la Action Bar només disposem de les tres primeres zones, on a la primera zona tindrem el logotip de la aplicació amb la fletxa per tornar enrere, a la segona zona tindrem el títol de la

pantalla en la que ens trobem i a la tercera zona tindrem les accions més utilitzades (Crear nova incidència, afegir comentari, etc).

3.4.7. Spinners

Els Spinners proporcionen una manera ràpida per seleccionar un valor d'un conjunt. En l'estat per defecte, un spinner mostra el valor actualment seleccionat. Si premem el spinner mostra un menú desplegable amb tots els altres valors disponibles, des de la qual l'usuari pot seleccionar un nou valor. Es recomanable utilitzar un spinner en els casos que es presenten diverses opcions, així no es carrega massa una pantalla amb totes les opcions disponibles.

3.4.8. Menús

Els menús són un component de la interfície d'usuari normal en molts tipus d'aplicacions. Per proporcionar una experiència d'usuari familiar i consistent, Android recomana utilitzar les API Menú per presentar les accions de l'usuari i altres opcions en les seves activitats.

Per a tots els tipus de menú, Android proporciona un format estàndard XML per definir els elements del menú. En lloc de construir un menú al codi de la nostra activitat, es pot definir un menú i tots els seus articles en un recurs de menú XML.

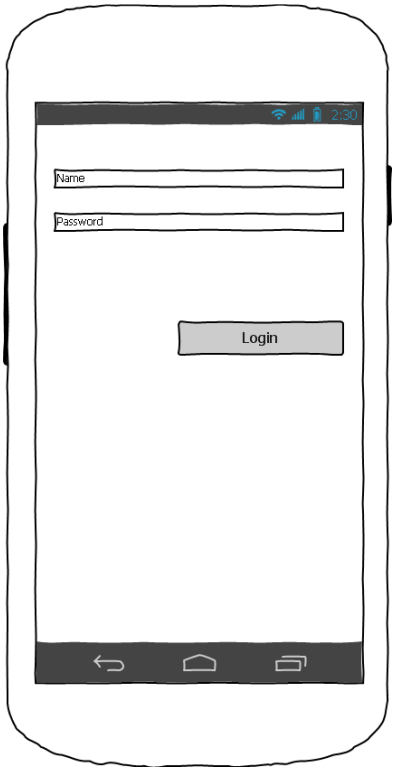
L'ús d'un recurs de menú és una bona pràctica per diverses raons:

- És més fàcil de visualitzar l'estructura de menús en XML.
- Es separa el contingut del menú a partir del codi de conducta de l'aplicació.
- Això permet crear configuracions de menú alternatius per a diferents versions de la plataforma, mides de pantalla i altres configuracions, aprofitant el marc de recursos d'aplicacions.

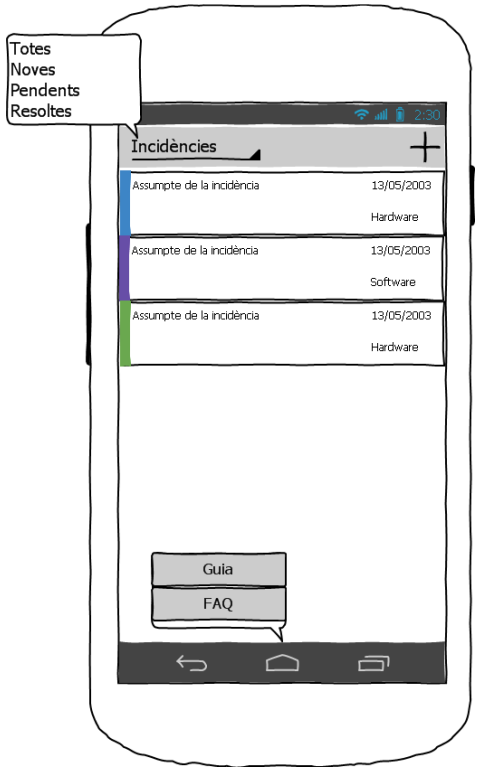
3.4.8.1. Options menú i Action Bar

El menú d'opcions és la col·lecció principal dels elements de menú per a una activitat. És el lloc on s'ha de posar accions que tenen un impacte global en l'aplicació.

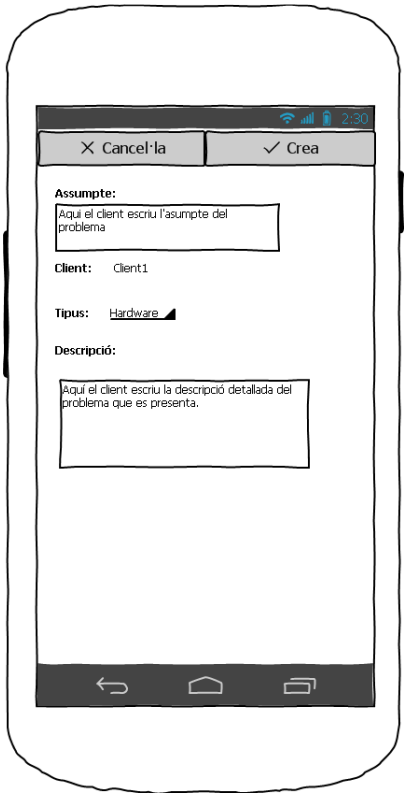
3.5. Vista prèvia



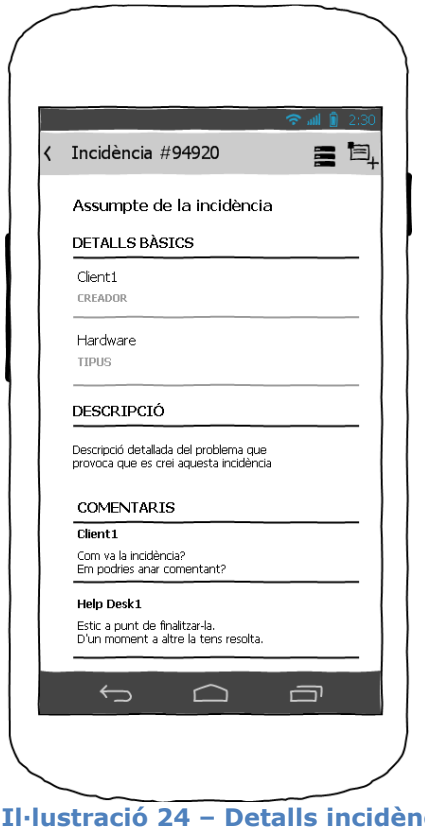
Il·lustració 21 – Login



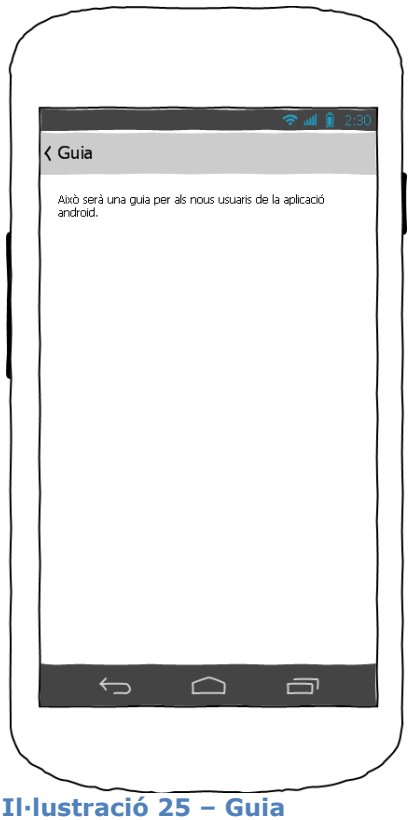
Il·lustració 22 – Incidències



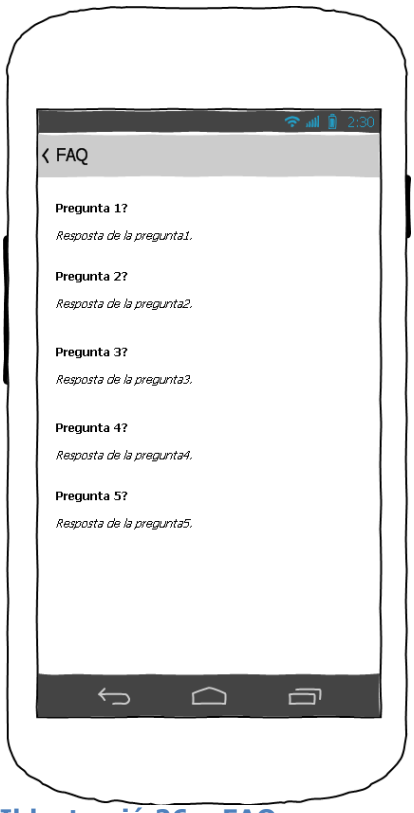
Il·lustració 23 – Crear incidència



Il·lustració 24 – Detalls incidència



Il·lustració 25 – Guia



Il·lustració 26 – FAQ

En la il·lustració 21 veiem la pantalla principal de l'aplicació que tracta del login.

En la il·lustració 22 apareix un cop les credencials siguin correctes, en aquesta trobarem un filtre per a les incidències, un botó "+" que s'utilitzarà per crear la nova incidència i tindrem la llista on el color significa l'estat de la incidència (Blau – nova, morat – pendent, verd – resolta/tancada), l'assumpte, la data de creació i el tipus d'incidència. En cas que polsem el botó de menú sorgiran dos més que seran els botons de guia i FAQ.

La il·lustració 23 apareix en el cas que el botó "+" sigui premut, en aquesta hi han dos botons, un per cancel·lar(en cas que no es vulgui crear i tornar a la il·lustració 22) i un altre per crear y afegir la incidència a la llista de noves. En aquesta haurem d'introduir l'assumpte, el tipus del problema i la descripció. El client és un camp que es posarà automàticament.

La il·lustració 24 apareix en el cas que premem qualsevol incidència i mostrarà tots els detalls de la incidència en qüestió, l'assumpte, el client que ha creat la incidència, el tipus, la descripció i els comentaris que s'han anat afegint. Tindrà un botó "<" posicionat a la part superior esquerrana per tornar a la il·lustració 22 i dos més a la part superior dretana per canviar l'estat de la incidència i per afegir comentari.

La il·lustració 25 apareix en cas que, prèviament, s'hagi polsat el botó menú en la il·lustració 22 i el botó guia. Aquesta tindrà una petita guia de com utilitzar l'aplicació per aquells nous usuaris que encara no han fet ús de l'aplicació. Tindrà un botó "<" posicionat a la part superior esquerrana per tornar a la il·lustració 22.

La il·lustració 26 apareix en cas que, prèviament, s'hagi polsat el botó menú en la il·lustració 22 i el botó FAQ. Aquesta tindrà una sèrie de preguntes freqüents per tal de contestar a l'usuari i no crear una nova incidència. Tindrà un botó "<" posicionat a la part superior esquerrana per tornar a la il·lustració 22.

3.6. Control d'accés

Com tota aplicació actual, on es treballi amb dades empresarials i dades dels treballadors o clients, ha de tenir una seguretat i per tant ha de tenir un control d'accés per evitar un mal ús de la informació que les aplicacions donen.

En aquesta aplicació no hi ha cap apartat de registre, ja que llavors qualsevol usuari es podria registrar i entrar a l'aplicació. El que s'ha fet a l'aplicació (com que és un servidor dummy) és controlar que només existeixin, de moment, dos usuaris, HelpDesk i Client, per tal de poder fer proves amb aquest dos usuaris. Per tal de donar d'alta un nou usuari, hauríem d'anar a l'apartat on es trobar el servidor i afegir unes quantes línies de codi.

Com en aquest cas es tracta d'un servidor dummy, no podem realitzar cap acció de permisos, on el client tingui cert control d'accés a certes parts de l'aplicació, per tant, tots els usuaris creats tindran accés a totes les accions de l'aplicació.

3.7. Disseny dels requeriments

En aquest apartat descriurem el disseny dels requeriments de l'aplicació, és a dir, una explicació de la interfície gràfica de pantalla per pantalla de les mides, accions, vista, etc. Com s'ha esmentat anteriorment està dissenyada per a que l'aplicació aprofiti el màxim d'espai i es vegi correctament en qualsevol dispositiu.

S'ha esmentat anteriorment, en l'apartat 3, que l'aplicació seguia el patró MVA (Model-Vista-Adaptador), a continuació parlarem del que esta format cadascuna de les components.

3.7.1. Adaptador

L'adaptador està format per les classes LoginActivity, LlistaIncidenciaActicity, DetallIncidenciaActivity, NovaIncidenciaActivity, GuiaActivity, FAQActivity, Controller el qual formen el paquet(package) principal i LlistaIncidenciesAdapter que es troba en un altre paquet, aquestes classes són les que s'encarregaran de fer d'intermediari entre el model i la vista.

3.7.1.1. LoginActivity

Es la classe que s'encarrega de controlar l'usuari que entra a la aplicació.

Es comença creant dues variables EditText. Aquestes dues variables son "private" ja que en cap altre lloc es faran utilitzar. El següent pas es inicialitzar la activitat per això es crea els mètodes OnCreate() i OnCreateOptionsMenu(). Dintre del OnCreate() el codi s'encarrega de posar com a títol HelpDesk Manager a la Action Bar, i el setContentView assigna el contingut del layout activity_login i afegeix tots els punts de vista a la activitat.

Es busca un view identificat per les id name i password i s'assignen a les variables nameView i mPasswordView.

A continuació es modifica el text que conté aquestes dues variables per a que un cop s'inicii l'aplicació apareguin l'usuari i el password escrits en els espais corresponents per tal de a l'hora de fer proves amb l'aplicació no es perdi temps introduint les credencials.

Un cop el botó és pres el que controlem és si les credencials són correctes, per això primer de tot es guarda en dues variables de tipus string l'usuari i el password i es fa la connexió amb el ServidorDummy, cridant un mètode on es passen les dues variables. Es crear una altre variable que retornarà un true o un false depenen de si les credencials, agafades en les variables de tipus string anteriors, son correctes o no.

En cas afirmatiu, es crida una funció que guardarà el nom de l'usuari per més tard utilitzar-lo i s'executa la següent activitat, en aquest cas la classe LlistaIncidenciesActivity.

En cas negatiu, apareixerà un AlertDialog, amb títol error, avisant de que les credencials que s'han introduït no son correctes i un botó d'acceptar (ja que el botó de cancel·lar s'ha desactivat) que un cop pres es tancarà el dialog.

El mètode OnCreateOptionsMenu(), el que es fa és assignar el recurs del menú activity_login a la activitat en la que ens trobem. En aquesta activitat, encara que es recomana que totes tinguin un menú, el menú esta desactivat.

3.7.1.2. LlistaIncidenciaActicity

És la classe que ha de permetre veure totes les incidències existents, siguin noves, pendents de resoldre o tancades, i poder crear una de nova. Ha de contenir el menú, que ens ha de permetre anar a la guia de l'usuari, a les preguntes més freqüents i sortir de l'aplicació, i ens ha de permetre entrar en els detalls de la incidència que seleccionem.

Aquesta classe també tindrà els mètodes onCreate() i onCreateOptionsMenu(). El mètode onCreate() es comença assignant el contingut del layout activity_llista_incidencies. En aquest cas, per tal de tindre el codi organitzat i no carregar massa el mètode OnCreate(), cridem un altre mètode nomenat configurarListView(), i li dona un títol a la Action Bar.

El mètode configurarListView és el que s'encarrega de mostrar la llista d'incidències en el listView assignat per la id. Es crea una variable que contindrà la llista de les incidències agafades amb el mètode getIncidenciesDelServidor() i es carrega en el listView assignat. Com que s'han de poder veure els detalls de la incidència seleccionada es crea el setOnItemClickListener, que espera a que una de les incidències sigui premuda, un cop es premi una de les incidències s'agafa la posició. Aquesta posició servirà per saber de quina de les incidències s'han de mostrar els detalls, per tant, es crida un mètode on es passa com a parametre la posició d'aquesta incidència, aquest mètode és obrirDetallDeIncidencia(). Aquest mètode s'encarrega de guardar la posició en una variable, després agafa la id de la incidència saben en la posició que es trobava i es crida l'activitat DetallIncidenciaActivity passant-li la id per saber quins detalls ha de mostrar.

El mètode per agafar les incidències del ServidorDummy conté el mètode obtenirIncidencies() que és un mètode de la classe ServidorDummy que s'encarrega, com ja diu el propi nom, d'obtenir les incidències existents.

Dintre del OnCreateOptionsMenu tenim gairebé el mateix que en la activitat anterior, exceptuant que en aquesta activitat està assignat com a recurs el menú d'activity_llista_incidencies.

Com que aquest menú està visible s'ha de controlar el que fa cadascun en cas de que siguin seleccionats. Esta format per quatre opcions, crear nova incidència, guia, FAQ i sortir, i cadascuna porten a una nova activitat. Per controlar quines de les opcions s'ha seleccionat es realitza un switch. Amb el switch es pretén realitzar un codi més net ja que són diferents opcions les que hi ha i cadascuna crida a un mètode diferent.

S'ha de tenir en compte que la opció de crear la nova incidència apareix en la Action Bar ja que és una opció molt important i que es farà servir sovint. Més endavant s'explica com fer per a afegir els menús en l'Action Bar.

3.7.1.3. LlistaIncidenciaAdapter

Aquesta classe s'encarrega de crear l'estructura que ha de seguir la llista d'incidències, és a dir, la llista que es mostra per pantalla, una incidència sota una altre.

Primer creem les variables que contindran la informació de les incidències que es mostrarà en el llistat. Aquestes són tres variables de tipus TextView (que seran l'assumpte de la incidència, la data de creació i el tipus de la incidència) i una variable de tipus view (que serà l'estat de la incidència).

Creem l'adaptador al que passarem el context i la llista dels objectes.

Cridem el mètode que s'encarrega de fer la resta.

Android te una propietat anomenada Tag (la qual es pot assignar i recuperar a través dels mètodes setTag() y getTag()) que pot contenir qualsevol objecte, per la qual cosa resulta ideal per a guardar el nostre ViewHolder.

La idea és crear i inicialitzar l'objecte ViewHolder el primer cop que inflem l'element de la llista i associar-lo a l'element de forma que posteriorment puguem recuperar-lo. Un cop s'ha inicialitzat comprovem si l'element no existeix, en cas de afirmatiu creem el ViewHolder i l'afegim al ConvertView, en cas que l'element existeixi, el que fem es reutilitzar-lo i obtenim el ViewHolder de dins. L'últim pas seria pintar el ViewHolder i retornar-lo. Per a pintar-lo cridem un mètode per tenir-lo tot ben estructurat, el que fem en aquest mètode no és més que agafar les dades de les incidències i pintar-les al holder.

3.7.1.4. DetallIncidenciaActivity

És la classe que permet veure tots els detalls de la incidència seleccionada a la activitat LlistaIncidenciaActivity a través de la posició i la id. Aquesta classe està composta per diversos mètodes, onCreate(), onCreateOptionsMenu(), obtenirIPintarIncidencia(), pintarIncidencia(), etc, per tal de tenir millor estructurada la classe.

Comencem l'activitat creant les variables de tipus TextView que seran l'assumpte, el client, el tipus i el comentari. Un cop tenim les variables que necessitem creades li segueix el mètode onCreate().

Primer se li assigna el contingut del layout activity_detall_incidencia i es posa un títol a la Action Bar. S'assignen a les variables, creades anteriorment, els views

amb les id corresponents a través del findViewById i es crida el mètode obtenirIPintarIncidencia().

El mètode obtenirIPintarIncidencia principalment s'encarrega de comprovar si la incidència seleccionada existeix, en cas afirmatiu, obtindrà la incidència a través de la id amb el mètode obtenirIncidenciaPerId i la pintarà cridant el mètode pintarIncidencia() passant com a paràmetre la incidència corresponent. En cas negatiu cridarà el mètode mostrarNoEsPotTtrobarIncidencia().

El mètode obtenirIncidenciaPerId() s'encarrega, a través de l'usuari que es va guardar en la classe loginActivity, d'obtenir la incidencia a través de l'usuari i la id.

El mètode pintarIncidencia() s'encarrega d'assignar el que ha de contenir cada variable, el cas dels comentaris esta desactivat ja que l'aplicació necessita millores.

El mètode mostrar NoEsPotTrobarIncidencia() el que farà és avisar amb un alertDialog que la incidència no es troba, veiem com se li assigna un títol, el missatge que mostrarà i que només contindrà el botó d'acceptar que un cop sigui premut es tancarà el dialog.

L'últim pas d'aquesta activitat són els métodes onCreateOptionsMenu() i onOptionsItemSelected().

En el mètode onCreateOptionsMenu(), com a la resta, s'assigna el contingut del menú activity_detall_incidencia. Y a continuació es crea el mètode que controlarà la opció que es seleccioni al menú.

Aquest menú te dues opcions que es troben en la part de l'Action Bar ja que tant canviar l'estat de la incidència, com afegir comentari, seran opcions que s'utilitzaran sovint.

Primer es controla l'element seleccionat, hi ha tres opcions:

- La primera opció és la de l'element de home, es el cas en que es premi el botó del logotip, provocant el retorn a l'activitat anterior.

- La segona opció és la de l'element del canvi d'estat, en aquest cas primer s'han de crear els estats. Per a poder escollir un estat el que es fa a continuació és crear un alertDialog on s'assigna un títol i s'assignen els estats com a elements a escollir.

- La tercera i última opció és la de l'element d'afegir comentari, en aquest cas també es farà a partir d'un alertDialog el qual tindrà assignat el contingut del layout afegir_comentari. Aquest alertDialog està format per un títol, un EditText (on s'escriurà el comentari), un botó d'acceptar i un altre de cancel·lar. Es controlarà quin dels dos botons s'ha premut, si es prem el d'acceptar es recupera l'usuari, per saber qui afegeix el comentari, s'agafa el valor de la variable afeCom, que conté el comentari en format de text, i s'afegeix el comentari al ServidorDummy i es mostrarà per pantalla un missatge afirmant que el comentari s'ha afegit correctament, com que es tracta d'un servidor Dummy no es guardarà el comentari, és aquesta una de les altres millores que es pretén afegir a l'aplicació més endavant. Si pel contrari es el botó de cancel·lar el que s'ha premut mostrarà un missatge per pantalla avisant que s'ha cancel·lat l'acció i es tancarà l'alertDialog.

3.7.1.5. NovaIncidenciaActivity

És la activitat que s'encarrega de crear les noves incidències, demanant l'assumpte, el tipus d'incidència i una petita descripció del problema que es presenta. També conté el nom del client que s'afegeix automàticament i dos botons, acceptar i cancel·lar. Per tal de poder realitzar totes les accions es programa el codi dintre del mètode onCreate().

Primer fora del mètode creem una variable de tipus TextView que utilitzarem per a agafar l'usuari que està creant la nova incidència.

El primer que es fa, dintre del mètode onCreate(), és assignar el contingut del layout activity_nova_incidencia, seguit de la crida del mètode spinnerCreate(). Ocultem l'Action Bar ja que aquí només es posaran els dos botons a la part superior de la pantalla i s'agafa el nom de l'usuari que vol crear la incidència. A continuació es programen els dos botons, tant el botó d'acceptar com el de cancel·lar (al ser un servidor Dummy es poden guardar noves dades però un cop tancada l'aplicació es perdran i, per tant, de moment no es controla que es guardin) faran el mateix excepte pel missatge que mostraran per pantalla. Tots dos botons estaran a l'espera de ser premuts, una vegada es premin el botó cancel·lar mostrarà que la creació s'ha cancel·lat, i el botó acceptar que s'ha creat satisfactoriament.

El que fa el mètode spinnerCreate() és crear el spinner que conté les opcions del tipus d'incidència de l'activitat NovaIncidenciaActivity. El que es fa és assignar a una variable la id del View i es passa com a recursos l'array que conté els tipus, a continuació s'assigna el layout simple_spinner_item per utilitzar-lo quan aparegui la llista d'opcions i s'aplica l'adaptador al spinner.

Per últim tenim el mètode onCreateOptionsMenu(), el que es fa en aquest mètode, és assignar el recurs del menú activity_nova_incidencia a la activitat en la que ens trobem. En aquesta activitat el menú esta desactivat.

3.7.1.6. Controller

Aquesta classe és la que s'encarregarà de guardar el nom d'usuari que ha fet login a l'aplicació, per després recuperar aquest usuari i utilitzar-lo en altres activitats. Consta de dos mètodes:

- GuardarUsuari, és el mètode que s'encarrega a la LoginActivity de guardar l'usuari que ha fet login a l'aplicació.

- RecuperarUsuari és el mètode que s'encarrega d'administrar aquest usuari en les diferents activitats que es cridat.

3.7.1.7. FAQActivity

És la classe que s'encarrega de mostrar les preguntes més freqüents de l'aplicació, està formada pels dos mètodes onCreate() i onCreateOptionsMenu().

Primer creem una variable privada de tipus WebView ja que aquesta activitat el que farà es mostrar un fitxer d'extensió ".html" per pantalla. Després, en el mètode onCreate(), s'assigna el contingut del layout activity_faq i se li afegeix un títol a la Action Bar. Assignem a la variable la id del View a través del findViewById, es permet l'execució de JavaScript i es carrega el fitxer.

L'únic que es fa en el mètode onCreateOptionsMenu(), és assignar el recurs del menú activity_faq a la activitat en la que ens trobem. En aquesta activitat el menú esta desactivat.

Actualment no mostra el fitxer, és una de les opcions que es vol millorar en un futur.

3.7.1.8. GuiaActivity

És la classe que s'encarrega de mostrar les preguntes més freqüents de l'aplicació, està formada pels dos mètodes onCreate() i onCreateOptionsMenu().

Primer creem una variable privada de tipus WebView ja que aquesta activitat el que farà es mostrar un fitxer d'extensió ".html" per pantalla. Després, en el mètode onCreate(), s'assigna el contingut del layout activity_guia i se li afegeix un títol a la Action Bar. Assignem a la variable la id del View a través del findViewById, es permet l'execució de JavaScript i es carrega el fitxer.

L'únic que es fa en el mètode onCreateOptionsMenu(), és assignar el recurs del menú activity_guia a la activitat en la que ens trobem. En aquesta activitat el menú esta desactivat.

Actualment no mostra el fitxer, és una de les opcions que es vol millorar en un futur.

3.7.2. Model

El model està format per les classes Comentari, Estat i Incidència, aquestes contenen informació de com s'ha de veure la vista dels comentaris, l'estat de la incidència i la informació de la incidència, és a dir, conté informació que es enviada al adaptador per a que sigui mostrada.

3.7.2.1. Comentari

Aquesta classe conté la informació de l'estructura del comentari i que és el que conté.

La classe tindrà dues variables, una que contindrà el nom de l'usuari i l'altre el contindrà el comentari, i tres mètodes, Comentari(), que guarda l'usuari i el comentari, getNomUsuari() i getComentari(), que retornen el nom d'usuari i el comentari.

3.7.2.2. Incidència

Aquesta classe conté la informació de l'estructura d'una incidència i que conté, encara que no tota la informació és visible.

La classe tindrà les variables id, estat, assumpte, creador, data, tipus, assignat i comentaris. Té un mètode per assignar l'estructura de la informació de la incidència i un mètode per variable existent per agafar el valor d'aquesta.

3.7.2.3. Estat

Aquesta classe conté la informació de com s'ha de veure l'estat de la incidència.

Conté dos mètodes, un per aplicar la id del color i l'altre per obtenir la id del color.

3.7.3. Vista

La vista està formada per totes aquelles classes que el Android SDK (Software Development Kit) ens ofereix. Aquestes classes que utilitzem en l'aplicació són View, TextView, WebView, ListView, Button, Spinner, EditText i LinearLayout. Aquestes classes s'encarreguen de presentar la informació del model en un format adequat.

3.7.3.1. View

Aquesta classe representa l'element bàsic per als components d'interfície d'usuari. El View ocupa una àrea rectangular a la pantalla, i és responsable del dibuix i la gestió d'esdeveniments. El View és la classe base per als widgets, que s'utilitzen per crear components d'interfície d'usuari interactius (botons, camps de text, etc.) La subclasse ViewGroup és la classe base per als dissenys, que són contenidors invisibles que tenen altres punts del View (o altres ViewGroups) i defineixen les seves propietats de disseny.

3.7.3.2. TextView

Mostra el text per a l'usuari i, opcionalment, els permet editar-lo. TextView és un editor de text complet, però la classe base està configurat per no permetre l'edició.

Per permetre als usuaris copiar part o la totalitat del valor de la TextView i enganxar-lo en un altre lloc, s'ha d'establir l'atribut XML android: textIsSelectable a "true" o s'ha de cridar setTextIsSelectable (true). El flag textIsSelectable permet als usuaris realitzar gestos de selecció en el TextView, que al seu torn provoca que el sistema permeti copiar / enganxar.

3.7.3.3. WebView

Una visió que mostra les pàgines web. Aquesta classe és la base sobre la qual es pot executar sobre el seu propi navegador web o simplement mostrar alguns continguts en línia dins la seva activitat. Utilitza el motor WebKit per mostrar les pàgines web i inclou mètodes per navegar cap endavant i cap enrere a través de l'historial, apropar i allunyar, fer cerques de text i molt més.

3.7.3.4. ListView

És una vista que mostra els elements d'una llista de desplaçament vertical. Els articles provenen dels ListAdapter associats amb aquesta vista.

3.7.3.5. Button

Representa un botó widget. Aquests polsadors es poden prémer o fer clic per l'usuari per realitzar una acció.

3.7.3.6. Spinner

És una visió que mostra un element alhora i permet a l'usuari triar entre diferents elements. Els elements de la Spinner provenen de l'adaptador associat amb aquest punt de vista.

3.7.3.7. EditText

EditText no és més que un TextView configurat per ser editable.

3.7.4. Servidor Dummy

És el servidor que s'encarrega de donar certes dades a l'aplicació, com per exemple les incidències o els usuaris que poden realitzar el login. Està format per dues classes Servidor i ServidorDummy.

La classe Servidor conté els mètodes login(), obtenir Incidencia(), obtenirIncidencies(), afegirComentari(), afegirIncidencia() i tancarIncidencia(). Encara que no totes estan implementades a l'aplicació.

Una part important de la classe ServidorDummy es que implementa els mètodes de la classe Servidor.

La primera part del codi indica que estem creant dos usuaris amb el mateix password, és la forma que tenim amb el servidor dummy per a controlar quins usuaris estan registrats.

Seguit de la creació d'usuaris, tenim el mètode ServidorDummy on es crea l'array d'incidències i afegim manualment les incidències, amb el seu estat, data, assumpte i tipus.

A continuació tenim la resta de mètodes, però explicarem només aquells que encara no estan en procés de desenvolupament. Tenim el mètode login(), que s'encarrega de controlar si les credencials introduïdes són correctes, si ho són retorna un true, sinó retorna un false. Seguit tenim el mètode obtenirIncidencies() que retorna l'array que conté les incidències. A continuació tenim el mètode afegirComentari() que s'encarregarà de carregar la llista de comentaris actuals de la incidència i afegir el nou. I l'últim mètode per explicar és el mètode obtenirIncidenciaPerId() que s'encarrega obtenir la incidència a través d'una id.

3.7.5. Recursos

3.7.5.1. Layouts

Són els continguts assignats a les respectives activitats, en aquest apartat com que són masses línies de codi explicarem com estan tots els layouts estructurats de forma grupal.

Per començar, la gran majoria dels layouts estan formats per linearlayouts amb orientació vertical o horitzontal. La amplada i la alçada, "`android:layout_width`" "`android:layout_height`", estan definides com match_parent o wrap_content i la justificació a través de l'atribut "android:layout_gravity" pot estar alineat a l'esquerra, alineat al centre o alineat a la dreta. A través de les propietats d'amplada i alçada es pot aprofitar el màxim d'espai a tots els dispositius.

Dintre d'aquest linearlayout és on posicionarem totes les classes que ens ofereix el SDK(Software Development Kit) d'Android. Alguns dels atributs utilitzats en aquestes classes són:

- android:id
- android:layout_marginBottom
- android:layout_marginLeft
- android:layout_marginRight
- android:layout_marginTop
- android:orientation
- android:layout_width
- android:layout_height
- android:layout_weight
- android:drawableLeft
- android:text
- android:textAppearance
- android:textColor
- android:textSize
- android:textStyle

3.7.5.2. Menús

Són els menús assignats a cada activitat, com que en aquesta aplicació molts dels menús no són visibles, només esmentarem dos d'ells que són el menú de la llista d'incidències i el menú del detall de la incidència.

- Activity_llista_incidencies
- Activity_detall_incidencia

Com hem esmentat en els apartats anteriors, en alguns casos apareixen els menús a les Action Bars, ho aconseguim amb la línia de codi "`android:showAsAction="ifRoom"`".

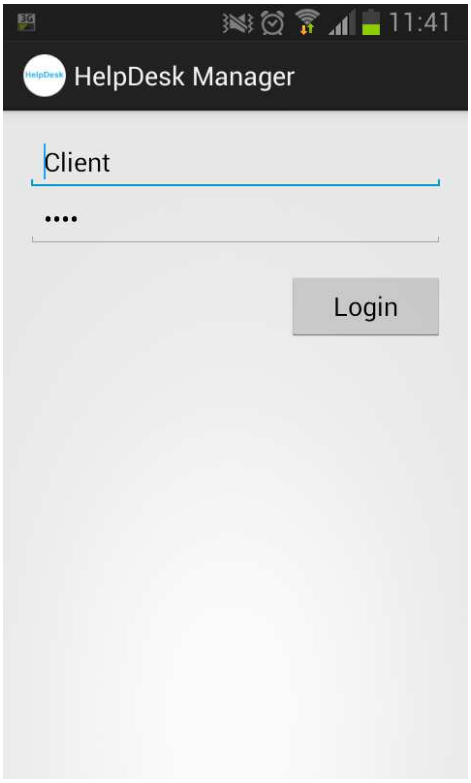
3.7.5.3. Valors

Són els valors que s'utilitzen en les activitats, tenim els següents valors:

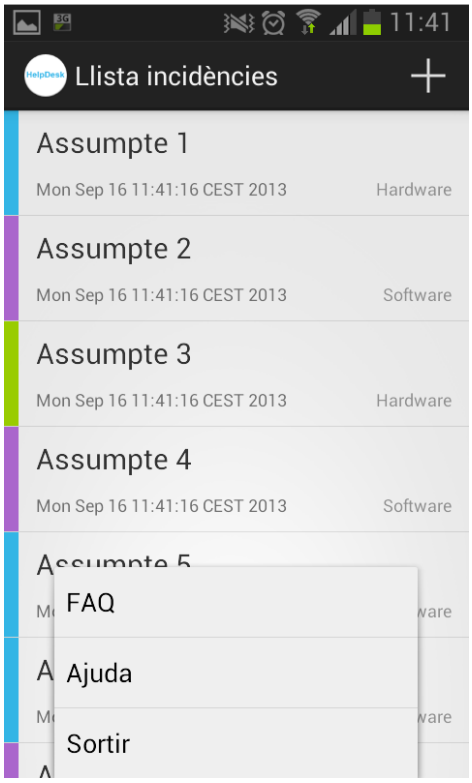
- Spinner
- Color de l'estat: El color Blau representarà el color de nova incidència, el color morat el de pendent de resoldre i el color verd representarà la incidència tancada o resolta.
- Estil Login
- Strings títols
- Login strings

3.7.6. Captures de pantalla

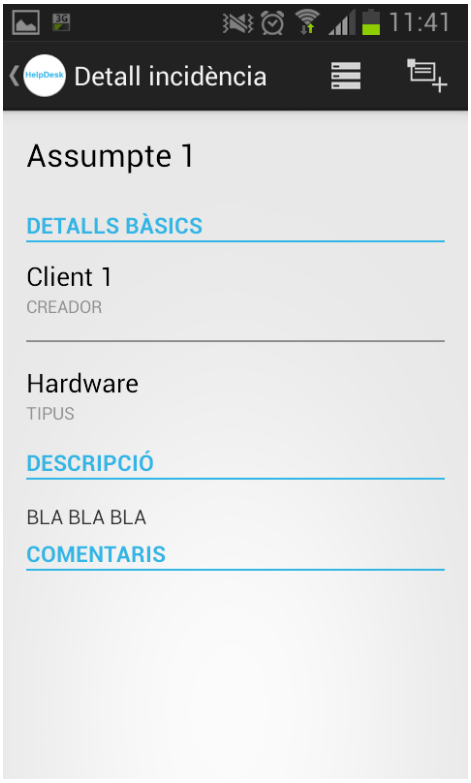
La imatge final de l'aplicació es mostra a continuació amb una imatge de cada activitat, amb els menú i els seus dialogs.



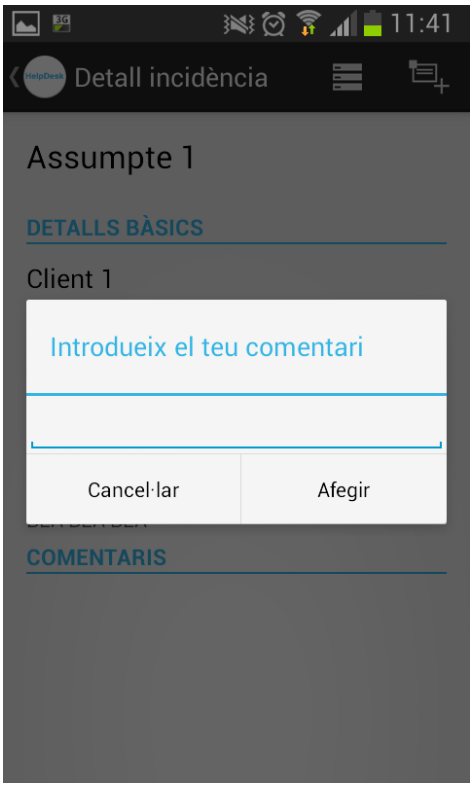
Il·lustració 27 – Activity login



Il·lustració 28 – Activity llista incidències i menú

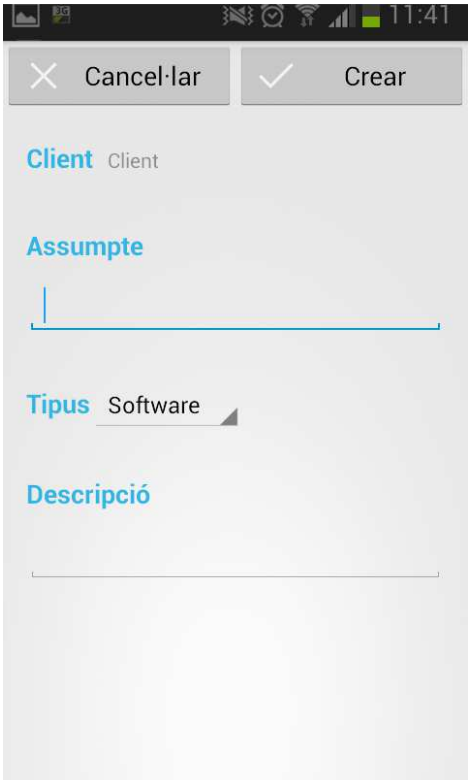


Il·lustració 29 – Activity detalls incidència



Il·lustració 30 – AlertDialog afegir comentari

Il·lustració 31 – AlertDialog escollir estat



Il·lustració 32 – Activity nova incidència

4. Informe de desenvolupament

Aquest capítol és un informe sobre les proves realitzades a l'aplicació per confirmar el seu bon funcionament i evitar el màxim d'errors possibles.

4.1. Pla de proves

El projecte es dona per acabat havent complert la gran majoria objectius previstos en un inici ja que s'ha anat canviant el projecte a mida que s'avançava.

La idea d'aquest apartat es exposar la nostra aplicació Adnroid a tot una sèrie de proves per trobar els errors o manques de requisits que pensàvem solucionar i tenir funcionant al final del nostre projecte.

Cal dir que , a pesar de que un cicle clàssic de desenvolupament de proves es realitza al final del projecte, en el nostre cas les proves s'han anat realitzant a mesura que s'introduïen noves línies de codi a l'aplicació.

S'ha d'afegir que una bona prova seria provar l'aplicació en diversos dispositius, però no a pogut ser possible ja que només disposem d'un sol dispositiu.

4.1.1. Login

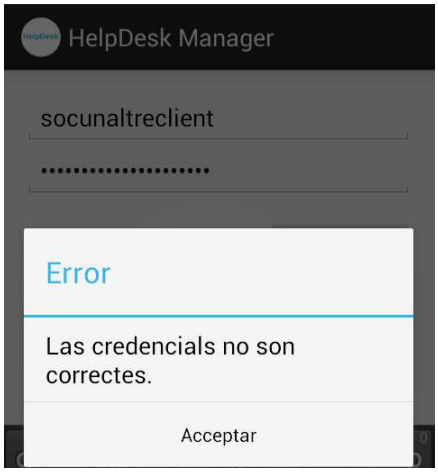
Es comprova que l'aplicació realitzi el login i que rebutgi els usuaris no registrats.

4.1.1.1. Correcte

Es realitza el login amb les credencials que l'aplicació ja posa des d'un inici (Client//1234). Realitza el login correctament i mostra la següent activitat, el llistat de la incidència (Il·lustració 28).

4.1.1.2. Incorrecte

Es realitza el login amb unes credencials inventades que siguin completament diferents de les ja incloses a l'inici de l'aplicació. Realitza el rebuig correctament i mostra el dialog d'error.



Il·lustració 33 – Prova login incorrecte

4.1.2. Llista incidències

Es comprova que l'aplicació mostri els detalls i la llista correctament.

4.1.2.1. Detalls vista prèvia

Es comprova que l'aplicació mostri les incidències amb l'estat (color), assumpte (títol), el tipus i la data ordenades com s'havia programat.

Il·lustració 34 – Prova detalls vista

4.1.2.2. Llistat d'incidències

Comprovem que la part que s'encarrega de llistar totes les incidències amb els detalls principals es col·loquin de manera correcte, és a dir, com una llista, una incidència sota l'altre. I alhora comprovem que es mostren una incidència de cada tipus i de cada estat.

Il·lustració 35 – Prova llistat

4.1.3. Nova incidència

Es comprova que l'aplicació mostri per pantalla l'activitat de la nova incidència, que realitzi correctament la cancel·lació i la creació d'una nova incidència.

4.1.3.1. Cancel·lar

Es procedeix a prémer el botó cancel·lar. L'aplicació mostra el missatge que s'ha cancel·lat la incidència retornant al llistat d'incidències.

Il·lustració 36 – Prova cancel·lació incidència

4.1.3.2. Acceptar

Es procedeix a prémer el botó acceptar. L'aplicació mostra el missatge que s'ha creat la incidència retornant al llistat d'incidències.

Il·lustració 37 – Prova creació incidència

4.1.4. Detall incidència

En aquesta prova es comprova que aparegui correctament el menú dels estats per realitzar el canvi, que es pugui afegir el nou comentari i el retorn a la llista d'incidències.

4.1.4.1. Canviar estat

Es procedeix a prémer el botó canviar estat. L'aplicació mostra el dialog dels estats (Il·lustració 31). Encara que premem un dels botons l'estat no canviarà ja que s'encarregaria el servidor d'aquesta part i es tracta d'un server dummy, com s'esmenta en apartats anteriors.

4.1.4.2. Nou comentari

Es procedeix a prémer el botó afegir comentari. L'aplicació mostra el dialog d'afegir comentari (Il·lustració 30). En cas de prémer el botó cancel·lar retorna al detall de la incidència sense avisar, però en cas d'afegir el comentari mostra el missatge per pantalla.

Il·lustració 38 – Prova afegir comentari

4.1.4.3. Retorn

Es procedeix a prémer el botó de retorn, que conté el logotip de l'aplicació, es realitza correctament el retorn.

4.1.5. FAQ

Es procedeix a prémer el menú FAQ, l'aplicació mostra l'activitat de les preguntes més freqüents però no carrega el webview, ja que un cop es creava el fitxer provocava errors a l'aplicació.

Il·lustració 39 – Prova FAQ

4.1.5.1. Retorn

Es procedeix a prémer el botó de retorn, que conté el logotip de l'aplicació, es realitza correctament el retorn.

4.1.6. Guia

Es procedeix a prémer el menú Guia, l'aplicació mostra l'activitat de les preguntes més freqüents però no carrega el webview, ja que un cop es creava el fitxer provocava errors a l'aplicació.

Il·lustració 40 – Prova Guia

4.1.6.1. Retorn

Es procedeix a prémer el botó de retorn, que conté el logotip de l'aplicació, es realitza correctament el retorn.

4.2. Conclusió del pla de proves

Com s'esmenta en alguns apartats anteriors hi ha proves que no es poden comprovar a causa de tenir un servidor dummy, provocant que no es puguin guardar les noves incidències creades, guardar el canvi de l'estat de la incidència, afegir els nous comentaris, etc.

5. Conclusions de la memòria

En aquest apartat es parla dels objectius que es van enumerar al principi del projecte, les variacions que s'han anat fent a mesura que s'avançava l'aplicació i si existeix alguna possibilitat de millorar o ampliar codi.

5.1. Compliment d'objectius

Es podria afirmar que el producte resultat d'aquest projecte és una solució que encaixa amb els objectius que es van enumerar durant la definició del projecte (Apartat 1.1.2).

L'aplicació desenvolupada permet de manera senzilla accedir a tot tipus d'informació de les incidències, permet veure l'estat de la incidència, de que tracta la incidència, de quin tipus és i la data de creació en un sol cop d'ull, a més, pot profunditzar més la informació prement la incidència.

L'aplicació també ofereix una millora de comunicació a través dels objectius encara que, de moment, no es guarden, és un tema que tractarem en un dels apartats següents inclòs el de l'ordenació d'incidències.

5.2. Variacions en la planificació

La principal variació de la planificació va ser que el projecte inicial consistia en una aplicació web implementada amb una base de dades, aquest tractava del mateix tema, una aplicació per a la gestió d'incidències. Es va decidir canviar per una aplicació Android per tal de crear una nova motivació, provocant que es tornes a començar pràcticament de zero i que es necessités més temps per aconseguir coneixement de programació.

Una altre variació del projectes va ser que l'aplicació contingues un menú principal per accedir a les diferents parts, però resultava una idea massa carregada i es va decidir que un cop es realitzes el login es mostressin les incidències i a partir d'aquí crear la resta de menús i accions corresponents.

També una idea era que les incidències estiguessin ordenades per tipus i de nova a tancada, però es va decidir, per estalviar temps, es faria en cas d'ampliar l'aplicació.

5.3. Líneas d'ampliació futures

Com s'ha comentat anteriorment, han hagut molts canvis al llarg del desenvolupament i objectius que no s'han pogut complir de manera perfecta.

Les ampliacions que es poden dur a terme un cop finalitzat l'aplicació serien:

- Crear una base de dades per poder carregar i guardar dades.
- Guardar les noves incidències.
- Guardar els nous comentaris.
- Ordenar les incidències per tipus i estat.
- Ocultar les incidències tancades a la llista principal i accedir a aquestes incidències amb un spinner a la Action Bar.
- Control de l'usuari, és a dir, diferents permisos per Client i Help Desk.
- Assignar una incidència d'estat nova al Help Desk en el moment en que es vegin els detalls.
- Canviar l'estat de nova a pendent en el moment en que el Help Desk vegi els detalls de la incidència.
- Implementar el fitxer .html per poder carregar-ho amb el webview.
- Afegir llegenda d'incidències.

5.4. Valoració final

Per començar he de dir que el desenvolupament d'aquest projecte ha sigut una experiència enriquidora ja que he pogut profunditzar molt en un llenguatge que, actualment, es molt important i hem pot obrir moltes portes de cara al futur laboral ja que el desenvolupament i programació d'Android estan en auge.

També ha sigut important el fet de realitzar el projecte des de zero, amb tot el que això comporta, com desenvolupar i estructurar de forma clara i senzilla, documentar tots els passos realitzats, problemes que sorgeixen al llarg de la programació, etc.

M'he adonat del que realment costa realitzar un projecte d'aquesta magnitud tenint els coneixements bàsics de programació i de la satisfacció que produeix veure els resultats obtinguts.

Encara que s'han tingut moments en els que hi ha hagut poca motivació, provocant la ralentització del desenvolupament del projecte, la valoració del projecte és molt positiva i me n'alegro d'haver canviat el projecte en el seu moment, encara que el número d'hores hagi sigut molt elevat.

6. Bibliografia

A continuació enumerem les referències bibliogràfiques que s'han utilitzat per a realitzar el projecte, encara que s'han realitzat moltes cerques mostrarem les més utilitzades i profitoses:

- Manual de Programació Android.

  PDF també disponible en versió web. És un manual on explica tot el que s'ha de saber sobre Android dirigit a totes les persones interessades en la programació d'aplicacions mòbils. [Online]

  http://www.sgoliver.net/blog/?page_id=3011

- Aprengui Java com si estigués en primer.

  Aprengui Java com si estigués en primer es un PDF dedicat a la programació Java per als nous usuaris. [Online]

  http://www.tecnun.es/asignaturas/Informat1/AyudaInf/aprendainf/Java/Java2.pdf.

- GitHub

  És una pàgina dedicada a compartir codi amb altres usuaris, on també és possible pujar el teu codi i desar-lo per a no perdre'l, i així compartir idees. [Online]

  https://github.com/.

- Android Developer

  És la pàgina oficial d'Android dissenyada per a tot usuari que vulgui programar una aplicació mòbil. Aquesta ha sigut la més important i més profitosa, ja que a més explica tot amb detall. [Online]

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

- Gantt Chart - Wikipedia

  S'ha obtingut informació sobre el diagrama. [Online]

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

- Història Android - Wikipedia

  S'ha obtingut informació sobre Android. [Online]

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

- Sowftware Development Kit - Wikipedia

  S'ha obtingut informació sobre SDK. [Online]

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

- Model-Vista-Controlador - Wikipedia

S'ha obtingut informació sobre el patró MVC. [Online]

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

- Model-Vista-Adaptador - Wikipedia

  S'ha obtingut informació sobre el patró MVA. [Online]

  http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93adapter

- HTCMania

  És un fòrum on les programadors exposen el seu problema i intenten resoldre-ho. [Online]

  http://www.htcmania.com/forumdisplay.php?f=153

- Stack Overflow

  És un lloc de preguntes i respostes per als programadors professionals i entusiastes. [Online]

  http://stackoverflow.com/

7. Annex

En aquest apartat afegim el codi de totes les activitats de l'aplicació.

7.1. Classe login

```
public class LoginActivity extends Activity {

	// UI references.
	private EditText nameView;
	private EditText mPasswordView;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		getActionBar().setTitle("HelpDesk Manager");
		setContentView(R.layout.activity_login);

		nameView = (EditText) findViewById(R.id.name);
		mPasswordView = (EditText) findViewById(R.id.password);

		// FIXME ESTO ES PARA ENTRAR DIRECTO
		nameView.setText("Client");
		mPasswordView.setText("1234");
		// FIXME BORRAR HASTA AQUI

		findViewById(R.id.sign_in_button).setOnClickListener(new
View.OnClickListener() {
			@Override
			public void onClick(View view) {
				Servidor dummy = new ServidorDummy();
				String nomUsuari =
nameView.getText().toString();
				String contrasenya =
mPasswordView.getText().toString();
				boolean successfulLogin =
dummy.login(nomUsuari, contrasenya);
				if (successfulLogin) {
					//controller
					Controller controller = new
Controller();

	controller.guardarUsuari(LoginActivity.this, nomUsuari);
					startActivity(new
Intent(LoginActivity.this, LlistaIncidenciesActivity.class));
				} else {

					AlertDialog.Builder alertLogin = new
AlertDialog.Builder(LoginActivity.this);
					alertLogin.setTitle("Error")
						.setMessage("Las credencials no
son correctes.")
						.setCancelable(false)
						.setNeutralButton("Acceptar", new
DialogInterface.OnClickListener() {
						public void onClick(DialogInterface
dialog, int id) {
							dialog.cancel();
```

```
                                }
                            });
                                    AlertDialog alert =
alertLogin.create();
                                    alertLogin.show();


                        }

                    }
            });
      }

      @Override
      public boolean onCreateOptionsMenu(Menu menu) {
            super.onCreateOptionsMenu(menu);
            getMenuInflater().inflate(R.menu.activity_login, menu);
            return true;
      }

}
```

7.2. Classe Controller

```
public class Controller {
      public static final String PREFS_NAME = "Users";
      public void guardarUsuari(Context context, String nUsuari) {
            SharedPreferences settings =
context.getSharedPreferences(PREFS_NAME, 0);
            SharedPreferences.Editor editor = settings.edit();
           editor.putString("Usuari", nUsuari);
           editor.commit();
      }
      public String recuperarUsuari(Context context) {
            SharedPreferences settings =
context.getSharedPreferences(PREFS_NAME, 0);
            String nUsuari = settings.getString("Usuari", null);
            return nUsuari;
      }
}
```

Il·lustració 41 – Codi controller

7.3. Classe llista incidències

```
public class LlistaIncidenciesActivity extends Activity {

      @Override
      protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_llista_incidencies);
            configurarListView();
            getActionBar().setTitle("Llista incidències");
      }

      private void configurarListView() {
```

```
            ListView listView = (ListView)
findViewById(R.id.listView1);
            final List<Incidencia> incidencies =
getIncidenciesDelServidor();
            listView.setAdapter(new LlistaIncidenciesAdapter(this,
incidencies));
            listView.setOnItemClickListener(new OnItemClickListener()
{

                  @Override
                  public void onItemClick(AdapterView<?> parent, View
view, int position, long id) {
                        obrirDetallDeIncidencia(position);
                  }

                  private void obrirDetallDeIncidencia(int position) {
                        Incidencia incidencia =
incidencies.get(position);
                        long id = incidencia.getId();
                        Intent intent = new
Intent(LlistaIncidenciesActivity.this,
DetallIncidenciaActivity.class);

      intent.putExtra(DetallIncidenciaActivity.PARAM_INDICENCIA_ID,
id);
                        startActivity(intent);
                  }

            });
      }

      private List<Incidencia> getIncidenciesDelServidor() {
            Servidor dummy = new ServidorDummy();
            List<Incidencia> incidencies = dummy.obtenirIncidencies();
            return incidencies;
      }

      @Override
      public boolean onCreateOptionsMenu(Menu menu) {
            // Inflate the menu; this adds items to the action bar if
it is present.

      getMenuInflater().inflate(R.menu.activity_llista_incidencies,
menu);
            return true;
      }

      @Override
      public boolean onOptionsItemSelected(MenuItem item) {
            // Handle item selection
            switch (item.getItemId()) {
            case R.id.faq:
                  obrirFAQActivity();
                  return true;
            case R.id.guia:
                  obrirGuiaActivity();
                  return true;
            case R.id.nova:
                  obrirNovaIncidenciaActivity();
                  return true;
            case R.id.sortir:
```

```
                  sortirActivity();
                  return true;
            default:
                  return super.onOptionsItemSelected(item);
            }
      }

      private void sortirActivity() {
            LlistaIncidenciesActivity.this.finish();

      }

      private void obrirNovaIncidenciaActivity() {
            Intent intent = new Intent(LlistaIncidenciesActivity.this,
NovaIncidenciaActivity.class);
            startActivity(intent);
      }

      private void obrirGuiaActivity() {
            Intent intent = new Intent(LlistaIncidenciesActivity.this,
GuiaActivity.class);
            startActivity(intent);
      }

      private void obrirFAQActivity() {
            Intent intent = new Intent(LlistaIncidenciesActivity.this,
FAQActivity.class);
            startActivity(intent);
      }

}
```

7.4. Classe detall incidència

```
public class DetallIncidenciaActivity extends Activity {

      public final static String PARAM_INDICENCIA_ID =
"incidencia_id";
      private Incidencia incidencia;
      TextView assumpteValor;
      TextView clientValor;
      TextView tipusValor;
      TextView comentariValor;

      protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_detall_incidencia);
            getActionBar().setTitle("Detall incidència");
            assumpteValor = (TextView) findViewById(R.id.detAssumpte);
            clientValor = (TextView) findViewById(R.id.detClient);
            tipusValor = (TextView) findViewById(R.id.detTipusSpin);
            comentariValor = (TextView)
findViewById(R.id.detDescripcio);

            obtenirIPintarIncidencia();

      }
```

```
	private void obtenirIPintarIncidencia() {
		Bundle extras = this.getIntent().getExtras();
		if (extras != null) {
			long id = extras.getLong(PARAM_INDICENCIA_ID);
			incidencia = obtenirIncidenciaPerId(id);
			pintarIncidencia(incidencia);
		} else {
			mostrarNoEsPotTrobarIncidencia();
		}
	}

	private void pintarIncidencia(Incidencia incidencia) {
		assumpteValor.setText(incidencia.getAssumpte());
		clientValor.setText(incidencia.getCreador());
		tipusValor.setText(incidencia.getTipus());
		// comentariValor.setText(incidencia.getComentaris());

	}

	private Incidencia obtenirIncidenciaPerId(long id) {
		Controller controller = new Controller();
		String nomUsuari = controller.recuperarUsuari(this);

		Servidor servidor = new ServidorDummy();
		Incidencia incidencia =
servidor.obtenirIncidencia(nomUsuari, id);

		return incidencia;
	}

	private void mostrarNoEsPotTrobarIncidencia() {
		//
		AlertDialog.Builder alertInc = new
AlertDialog.Builder(DetallIncidenciaActivity.this);
		alertInc.setTitle("Error").setMessage("No s'ha trobat la
incidència.").setCancelable(false)
				.setNeutralButton("Acceptar", new
DialogInterface.OnClickListener() {
					public void onClick(DialogInterface
dialog, int id) {
						dialog.cancel();
					}
				});
		alertInc.show();
	}

	@Override
	public boolean onCreateOptionsMenu(Menu menu) {
		// Inflate the menu; this adds items to the action bar if
it is present.

	getMenuInflater().inflate(R.menu.activity_detall_incidencia,
menu);
		return true;
	}

	@Override
	public boolean onOptionsItemSelected(MenuItem item) {
		if (item.getItemId() == android.R.id.home) {
			NavUtils.navigateUpFromSameTask(this);
```

```
            } else if (item.getItemId() == R.id.canvi) {
                  CharSequence[] estats = { "Nova", "Pendent",
"Resolta" };
                  AlertDialog.Builder builder = new
AlertDialog.Builder(DetallIncidenciaActivity.this);
                  builder.setTitle("Escull l'estat");
                  builder.setItems(estats, new
DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog,
int which) {
                              // The 'which' argument contains the
index position
                              // of the selected item
                        }
                  });
                  builder.create().show();
            } else if (item.getItemId() == R.id.coment) {
                  final EditText afeCom = new EditText(this);
                  AlertDialog.Builder builder = new
AlertDialog.Builder(DetallIncidenciaActivity.this);
                  builder.setTitle("Introdueix el teu comentari");
                  LayoutInflater inflater = this.getLayoutInflater();

      builder.setView(inflater.inflate(R.layout.afegir_comentari,
null))
                              .setPositiveButton("Afegir", new
DialogInterface.OnClickListener() {
                                    public void
onClick(DialogInterface dialog, int id) {
                                          Servidor dummy = new
ServidorDummy();
                                          Controller controller = new
Controller();
                                          String nomUsuari =
controller.recuperarUsuari(DetallIncidenciaActivity.this);
                                          String comentari =
afeCom.getText().toString();
                                          boolean comentariAfegit =
dummy.afegirComentari(incidencia.getId(), nomUsuari, comentari);
                                          if (comentariAfegit) {

      Toast.makeText(DetallIncidenciaActivity.this, "Comentari afegit
correctament",

      Toast.LENGTH_LONG).show();
                                          }
                                    }
                              }).setNegativeButton("Cancel·lar", new
DialogInterface.OnClickListener() {
                                    public void
onClick(DialogInterface dialog, int id) {
                                          dialog.dismiss();
                                    }
                              });
                  builder.create().show();
            }
            return true;
      }
}
```

7.5. Classe nova incidència

```
public class NovaIncidenciaActivity extends Activity {

	TextView crearValor;
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_nova_incidencia);
		spinnerCreate();
		getActionBar().hide();
		Controller controller = new Controller();
		String nomCreador =
controller.recuperarUsuari(NovaIncidenciaActivity.this);
		crearValor = (TextView) findViewById(R.id.editClient);
		crearValor.setText(nomCreador);
		// Botó CANCEL·LAR
		findViewById(R.id.bCan).setOnClickListener(new
View.OnClickListener() {

			public void onClick(View v) {
				Toast cancel =
Toast.makeText(getApplicationContext(), "Has cancel·lat la creació",
Toast.LENGTH_SHORT);
				cancel.show();
				NovaIncidenciaActivity.this.finish();

			}

		});
		// Botó Acceptar
		findViewById(R.id.bAce).setOnClickListener(new
View.OnClickListener() {

			public void onClick(View v) {
				Toast accept =
Toast.makeText(getApplicationContext(), "S'ha creat correctament",
Toast.LENGTH_SHORT);
				accept.show();
				NovaIncidenciaActivity.this.finish();

			}

		});
	}

	private void spinnerCreate() {
		Spinner spinner = (Spinner) findViewById(R.id.editTipus);
		// Create an ArrayAdapter using the string array and a
default spinner
		// layout
		ArrayAdapter<CharSequence> adapter =
ArrayAdapter.createFromResource(this, R.array.array_tipus,
				android.R.layout.simple_spinner_item);
		// Specify the layout to use when the list of choices
appears

	adapter.setDropDownViewResource(android.R.layout.simple_spinner_
dropdown_item);
		// Apply the adapter to the spinner
		spinner.setAdapter(adapter);
```

```
	}

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

}
```

7.6. Classe FAQ

```
public class FAQActivity extends Activity {
 private WebView mWebView;
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_faq);
		getActionBar().setTitle("FAQ");
		mWebView = (WebView) findViewById(R.id.webFAQ);
		mWebView.getSettings().setJavaScriptEnabled(true);
		mWebView.loadUrl("file:///android_asset/FAQ.html");
	}

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

}
```

7.7. Classe Guia

```
public class GuiaActivity extends Activity {

	private WebView mWebView;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_guia);
		getActionBar().setTitle("Guia");
		mWebView = (WebView) findViewById(R.id.webGuia);
		mWebView.getSettings().setJavaScriptEnabled(true);
		mWebView.loadUrl("file:///android_asset/Guia.html");
	}

	@Override
	public boolean onCreateOptionsMenu(Menu menu) {
```

```
            // Inflate the menu; this adds items to the action bar if
it is present.
            getMenuInflater().inflate(R.menu.activity_guia, menu);
            return true;
      }

}
```

7.8. Classe llista incidències adapter

```
public class LlistaIncidenciesAdapter extends ArrayAdapter<Incidencia>
{

      private static class IncidenciaListViewHolder {
            View estatView;
            TextView assumpteView;
            TextView dataView;
            TextView tipusView;
      }

      public LlistaIncidenciesAdapter(Context context,
List<Incidencia> objects) {
            super(context, 0, objects);
      }

      @Override
      public View getView(int position, View convertView, ViewGroup
parent) {
            // Obtenir ViewHolder
            View elementView = convertView;
            IncidenciaListViewHolder holder = null;

            // Si item no existeix (no estem reutilitzant), creem el
            // ViewHolder i l'afegim a la convertView
            if (elementView == null) {
                  LayoutInflater inflater = (LayoutInflater)
getContext().getSystemService(Context.LAYOUT_INFLATER_SERVICE);
                  elementView =
inflater.inflate(R.layout.element_llista_incidencia, null);

                  holder = new IncidenciaListViewHolder();
                  holder.estatView =
elementView.findViewById(R.id.estat);
                  holder.assumpteView = (TextView)
elementView.findViewById(R.id.assumpte);
                  holder.dataView = (TextView)
elementView.findViewById(R.id.data);
                  holder.tipusView = (TextView)
elementView.findViewById(R.id.tipus);

                  elementView.setTag(holder);
            } else {
                  // Si ja existeix (estem reutilitzant), obtenir el
viewHolder de
                  // dins
                  holder = (IncidenciaListViewHolder)
convertView.getTag();
```

```
            }

            // Pintar ViewHolder
            pintarViewHolder(holder, position);

            return elementView;

      }

      private void pintarViewHolder(IncidenciaListViewHolder holder,
int position) {

            Incidencia incidencia = getItem(position);

            int color =
getContext().getResources().getColor(incidencia.getEstat().getColorId(
));
            String assumpte = incidencia.getAssumpte();
            String data = incidencia.getData().toString();
            String tipus = incidencia.getTipus();

            holder.estatView.setBackgroundColor(color);
            holder.assumpteView.setText(assumpte);
            holder.dataView.setText(data);
            holder.tipusView.setText(tipus);
      }
}
```

7.9. Classe comentari

```
public class Comentari {

      private String nomUsuari;
      private String comentari;

      public Comentari(String nomUsuari, String comentari) {
            super();
            this.nomUsuari = nomUsuari;
            this.comentari = comentari;
      }

      public String getNomUsuari() {
            return nomUsuari;
      }

      public String getComentari() {
            return comentari;
      }

}
```

7.10. Classe estat

```
public enum Estat {
```

```
	NOVA(R.color.estat_nova),
	EN_PROGRES(R.color.estat_en_progres),
	TANCADA(R.color.estat_tancada);

	private int colorId;

	Estat(int colorId) {
		this.colorId = colorId;
	}

	public int getColorId() {
		return colorId;
	};

}
```

7.11. Classe incidència

```
public class Incidencia {

	private long id;
	private Estat estat;
	private String assumpte;
	private Date data;
	private String tipus;
	private String creador;
	private String assignat;
	private List<Comentari> comentaris;

	public Incidencia(long id, String creador, Estat estat, String
assumpte, Date data,
			String tipus) {
		super();
		this.id = id;
		this.creador = creador;
		this.estat = estat;
		this.assumpte = assumpte;
		this.data = data;
		this.tipus = tipus;
		this.comentaris = new ArrayList<Comentari>();
	}

	public long getId() {
		return id;
	}

	public String getCreador() {
		return creador;
	}

	public Estat getEstat() {
		return estat;
	}

	public void setEstat(Estat estat) {
		this.estat = estat;
	}
```

```
	public String getAssumpte() {
		return assumpte;
	}

	public Date getData() {
		return data;
	}

	public String getTipus() {
		return tipus;
	}

	public String getAssignat() {
		return assignat;
	}

	public void setAssignat(String assignat) {
		this.assignat = assignat;
	}

	public List<Comentari> getComentaris() {
		return comentaris;
	}

}
```

7.12. Classe servidor

```
public interface Servidor {

	public boolean login(String nomUsuari, String contrasenya);

	public Incidencia obtenirIncidencia(String nomUsuari, long id);

	public List<Incidencia> obtenirIncidencies();

	public boolean afegirComentari(long idIncidencia, String
nomUsuari, String comentari);

	boolean afegirIncidencia(String creador, String assumpte, String
tipus);

	boolean tancarIncidencia(long idIncidencia);

}
```

7.13. Classe servidor dummy

```
public class ServidorDummy implements Servidor {

	private static final String PASSWORD = "1234";

	private static final String USER_CLIENT = "Client";

	private static final String USER_HELP_DESK = "HelpDesk";
```

```
	private List<Incidencia> incidencies;

	public ServidorDummy() {
		incidencies = new ArrayList<Incidencia>();
		incidencies.add(new Incidencia(1L, "Client 1", Estat.NOVA,
"Assumpte 1", new Date(), "Hardware"));
		incidencies.add(new Incidencia(2L, "Client 2",
Estat.EN_PROGRES, "Assumpte 2", new Date(), "Software"));
		incidencies.add(new Incidencia(3L, "Client 1",
Estat.TANCADA, "Assumpte 3", new Date(), "Hardware"));
		incidencies.add(new Incidencia(4L, "Client 2",
Estat.EN_PROGRES, "Assumpte 4", new Date(), "Software"));
		incidencies.add(new Incidencia(5L, "Client 3", Estat.NOVA,
"Assumpte 5", new Date(), "Hardware"));
		incidencies.add(new Incidencia(6L, "Client 1", Estat.NOVA,
"Assumpte 1", new Date(), "Software"));
		incidencies.add(new Incidencia(7L, "Client 1",
Estat.EN_PROGRES, "Assumpte 2", new Date(), "Hardware"));
		incidencies.add(new Incidencia(8L, "Client 5",
Estat.TANCADA, "Assumpte 3", new Date(), "Software"));
		incidencies.add(new Incidencia(9L, "Client 1",
Estat.EN_PROGRES, "Assumpte 4", new Date(), "Hardware"));
		incidencies.add(new Incidencia(10L, "Client 7",
Estat.NOVA, "Assumpte 5", new Date(), "Software"));
		incidencies.add(new Incidencia(11L, "Client 3",
Estat.NOVA, "Assumpte 1", new Date(), "Hardware"));
		incidencies.add(new Incidencia(12L, "Client 1",
Estat.EN_PROGRES, "Assumpte 2", new Date(), "Software"));
		incidencies.add(new Incidencia(13L, "Client 2",
Estat.TANCADA, "Assumpte 3", new Date(), "Hardware"));
		incidencies.add(new Incidencia(14L, "Client 4",
Estat.EN_PROGRES, "Assumpte 4", new Date(), "Software"));
		incidencies.add(new Incidencia(15L, "Client 8",
Estat.NOVA, "Assumpte 5", new Date(), "Hardware"));

	}

	@Override
	public boolean login(String nomUsuari, String contrasenya) {
		if (nomUsuari.equals(USER_HELP_DESK) &&
(contrasenya.equals(PASSWORD))) {
			return true;
		} else if (nomUsuari.equals(USER_CLIENT) &&
(contrasenya.equals(PASSWORD))) {
			return true;
		}
		return false;
	}

	@Override
	public List<Incidencia> obtenirIncidencies() {
		return incidencies;
	}

	@Override
	public boolean afegirComentari(long idIncidencia, String
nomUsuari, String comentari) {

		Incidencia incidencia =
obtenirIncidenciaPerId(idIncidencia);
```

```
            if (incidencia == null) {
                  return false;
            }
            if (nomUsuari == null) {
                  return false;
            }
            if (comentari == null) {
                  return false;
            }

            Comentari nouComentari = new Comentari(nomUsuari,
comentari);

            incidencia.getComentaris().add(nouComentari);

            return true;
      }

      @Override
      public boolean afegirIncidencia(String creador, String assumpte,
String tipus) {

            long id = incidencies.size();
            Estat estat = Estat.NOVA;
            Date data = new Date();

            Incidencia incidencia = new Incidencia(id, creador, estat,
assumpte, data, tipus);

            incidencies.add(incidencia);

            return true;
      }

      private Incidencia obtenirIncidenciaPerId(long id) {
            Incidencia incidencia = null;

            for (Incidencia incidenciaDeLlista : incidencies) {
                  if (incidenciaDeLlista.getId() == id) {
                        incidencia = incidenciaDeLlista;
                  }
            }
            return incidencia;
      }
}
```

Signat: ***Alberto García Puente***