MY FLIGHT RECORDER

TFC – Desenvolupament d'aplicacions mòbils

Arnau Solsona i Solà

ÍNDEX

1 INTRODUCCIÓ. 2

1.1 MOTIVACIÓ. 3
1.2 DESCRIPCIÓ. 4
1.3 OBJECTIUS. 7
1.4 MÈTODE DE TREBALL. 8
1.5 RECURSOS DE DESENVOLUPAMENT. 9

2 REQUISITS DE L'APLICACIÓ. 10

2.1 USUARIS. 11
2.2 REQUISITS FUNCIONALS. 12
2.3 REQUISITS D'INFORMACIÓ. 13

3 ANÀLISIS DEL SISTEMA. 15

3.1 DIAGRAMES DE CASOS D'ÚS. 15
3.2 DESCRIPCIÓ DE CASOS D'ÚS. 16

4 DISSENY. 22

4.1 ARQUITECTURA DE L'APLICACIÓ. 22
4.2 DIAGRAMA DE SEQÜENCIA. 23
4.3 DISSENY DE LA PERSISTÈNCIA. 24
4.4 PROTOTIPATGE. 26
4.5 CONCLUSIONS DEL PROCÉS DCU 30
4.6 QÜESTIONARIS. 32

5 IMPLEMENTACIÓ. 34

5.1 PROCÉS D'IMPLEMENTACIÓ. 34
5.2 DIAGRAMA DE CLASSES. 13
5.3 DESCRIPCIÓ DE LES CLASSES. 41
5.4 TRACTAMENT DE LES DADES TEMPORALS. 43

6 CONCLUSIONS. 46

7 ANNEXOS. ..... ¡ERROR! MARCADOR NO DEFINIDO.

8 GLOSSARI DE TERMES. 47

9 BIBLIOGRAFIA I RECURSOS. 48

1 Introducció.

Aquest treball de fi de carrera s'emmarca dins l'assignatura "desenvolupament d'aplicacions per a dispositius mòbils".

Els mòbils intel·ligents han tingut un progres impressionant durant aquests últims anys, el 2012 només Samsung i Apple junts han venut més de 350 milions de telèfons intel·ligents. Sense dubte això és indici que aquesta tecnologia no només te un present notable sinó un gran futur. D'aquí l'interès per a desenvolupar aplicacions mòbils .

Dins les possibilitats s'ha elegit desenvolupar una aplicació per a dispositius amb sistema operatiu Android per varis motius.

El primer motiu d'elecció Android envers altres possibilitats és que es tracta d'una plataforma de codi obert.

El segon motiu és l'innegable èxit d'aquesta plataforma, que a dia d'avui arriba a 1,5 milions d'activacions per dia.

El tercer motiu és la necessitat de l'aplicació de funcionar nativament, en contra de html 5 per exemple.

Així doncs, l'aplicació que es desenvolupa durant aquest projecte és un llibre de registre d'hores de vol, "logbook" , per al sistema operatiu Android.

1.1 Motivació.

Després d'un estudi de les aplicacions disponibles per a Android per a l'anotació de hores de vol s'ha trobat que les aplicacions disponibles no complien correctament amb el seu propòsit, que presentaven problemes de disseny o simplement estan optimitzades per a pilots de línia aèria amb opcions que no interessen a una gran part del col·lectiu. A continuació passo a detallar alguns dels errors trobats en aquestes aplicacions, encara que aquest no és l'objectiu d'aquest treball.

Moltes de les aplicacions provades per aquest propòsit inunden la pantalla amb una quantitat d'informació innecessària que provoca que l'usuari quedi desbordat i perdi de vista l'objectiu principal de l'aplicació, que és, obtenir un registre de les hores volades.

Per exemple, en algunes aplicacions al introduir el codi OACI, *Organització d'Aviació Civil Internacional*, de un aeroport apareixen un munt de dades sobre l'aeroport. El primer problema d'aquestes opcions és que hi ha molta gent que vola en aeròdroms, heliports, vaixells, etc... que no tenen possibilitat d'adquirir aquesta informació perquè no està publicada, i el segon problema és que aquesta informació és necessària avanç de començar el vol i no quan ja s'ha acabat.

Seguint amb l'excés d'informació, s'ha trobat varies aplicacions que tenen varies opcions per a la planificació del vol, com ara, la informació meteorològica, control de combustible, etc... . El problema és el mateix d'abans, un "*logbook*" no és un llibre de planificació de vols sinó un llibre de registre de vols i totes aquestes opcions provoquen que l'ús de l'aplicació sigui molt més complicat.

Un altre problema trobat ve donat per culpa dels formats de data, ja que algunes de les aplicacions provenen d'estats units i les dates són mostrades en format americà, mes dia i any, en comptes de dia mes i any, i sense opció a utilitzar-les en format europeu.

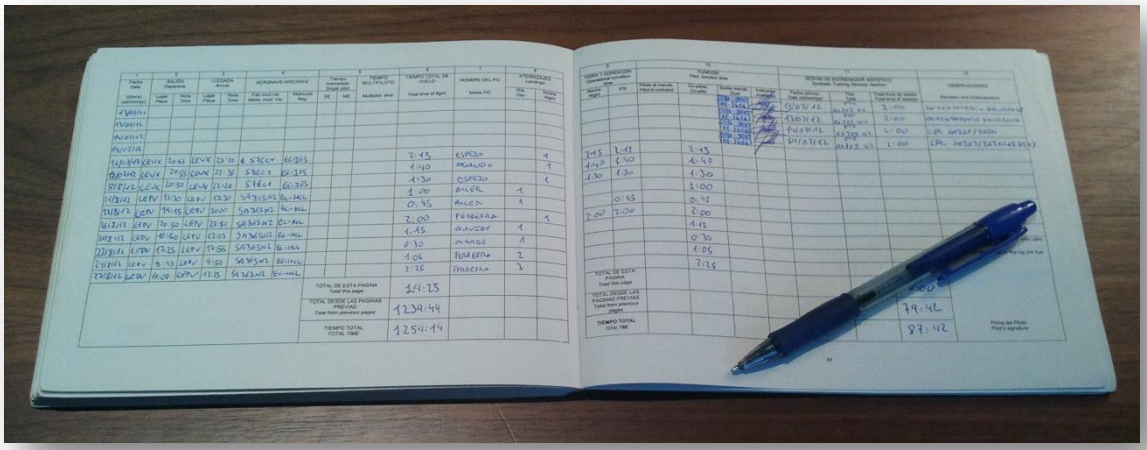
Però també s'han trobat problemes amb la suma d'hores de vol donat que alguna aplicació arrodonia els minuts volats de hexadecimal a decimal. Això és així perquè algunes aeronaus tenen un taquímetre que dona el temps de vol en format decimal, per exemple, una hora i sis minuts serien 1.1 hores de vol, el problema ve quan volem en minuts no múltiples de 6. Per exemple, una hora i cinc minuts, aleshores el temps de vol continua sent 1.1, però ara hem guanyat un minut de vol.

Així doncs la motivació d'aquest treball és fer una aplicació de registres de vol que sigui senzilla d'utilitzar, que sigui específica per al seu propòsit i que resulti natural d'utilitzar per a pilots que utilitzen "*logbooks*" habitualment. Per això el disseny està centrat en que l'ús s'assembli com mes possible en la introducció de dades en un diari

de vols de paper tradicional, i sense extres innecessaris que fan de la introducció de dades una experiència tediosa.

1.2 Descripció.

Tal i com ja s'ha comentat, el treball consisteix en crear una aplicació per al sistema operatiu Android que permeti substituir el llibre de registres de vols tradicional en paper.



Il·lustració 1 - "Logbook"

El "*logbook*" o llibre de registre de vols és un llibre amb un format establert on s'apunten les hores de vol. Quan omplim una plana aleshores sumem totes les hores de forma que tenim una suma del total de la pàgina i després sumem la suma de les pàgines anteriors per a obtenir el total de hores sumades. Aquestes sumes es fan per al total de hores de vol però també per a altres camps com poden ser hores nocturnes, hores de vol instrumental (IFR), de copilot, comandant, etc...

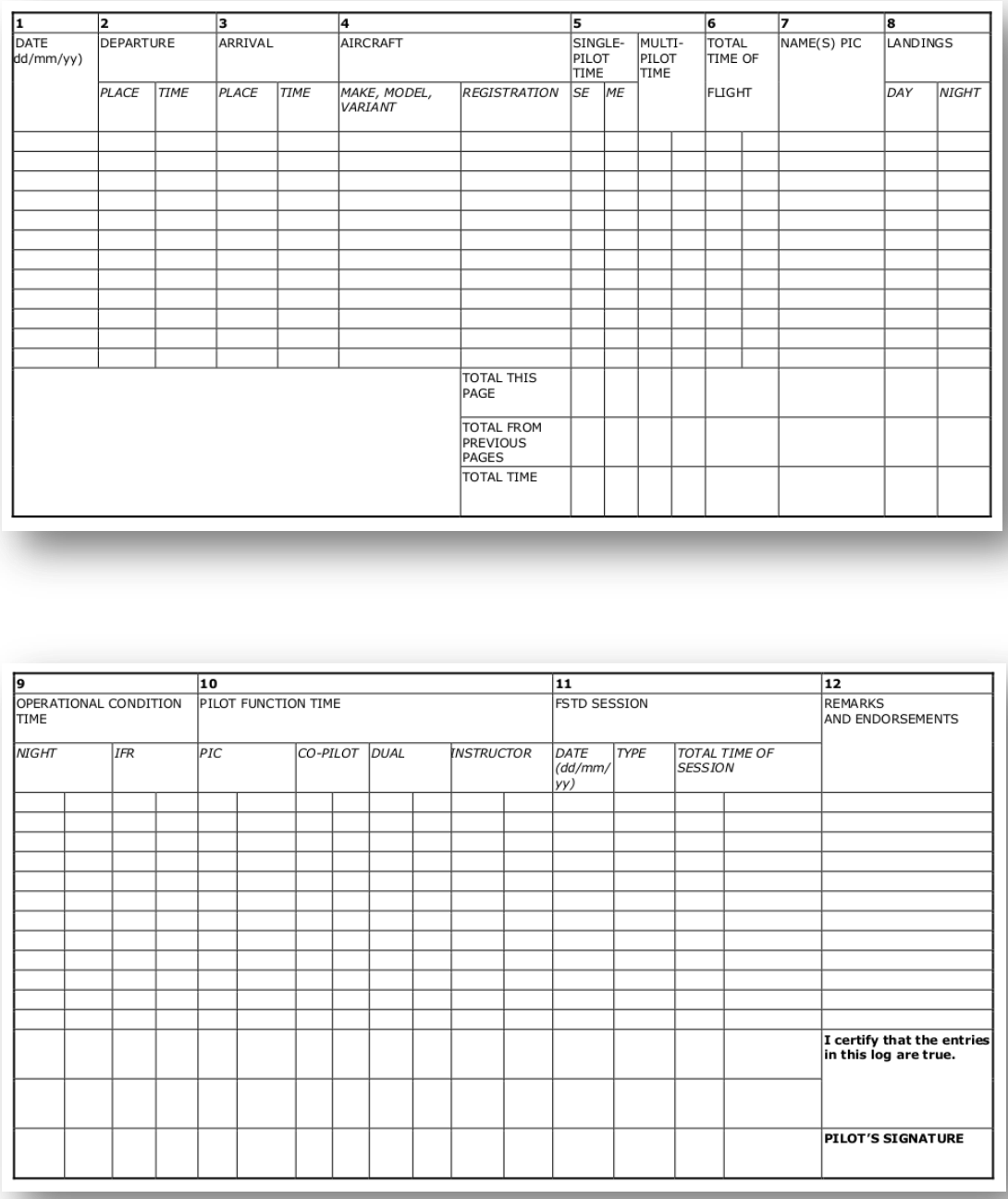
El principal problema d'aquest sistema tradicional és que el sumatori de hores després de varies pàgines mai coincideix. Per exemple, si sumem hores diürnes més hores nocturnes hauríem d'obtenir el total de hores de vol però normalment s'arrosseguen errors, ja que la suma és manual, i mai coincideix.

Aquesta aplicació pretén obtenir un registre de vols el qual solucioni els problemes que suposa mantenir un registre manual. Com ja s'ha comentat amb aquesta aplicació

s'obtenen les sumes de hores sense errors produïts per la suma manual. Però a més a més ofereix altres avantatges com ara la possibilitat de obtenir còpies de seguretat per evitar-ne la pèrdua o la possibilitat d'obtenir el registre de vols en format "pdf". També ofereix, encara que no és una opció imprescindible, la possibilitat de veure gràficament les estadístiques de hores de vol.

Com que l'aplicació està pensada en ser utilitzada a Europa s'ha elegit el format de "logbook" europeu segons la "European Joint Aviation Authorities" . No es objectiu del projecte, però seria fàcil implementar una sortida dels registres de vols que fos coincident amb el format americà, per exemple.

A continuació espot veure el format europeu de un diari de vol.



1	2		3		4		5				6		7	8	
DATE dd/mm/yy)	DEPARTURE		ARRIVAL		AIRCRAFT		SINGLE-PILOT TIME		MULTI-PILOT TIME		TOTAL TIME OF		NAME(S) PIC	LANDINGS	
	PLACE	*TIME*	*PLACE*	*TIME*	*MAKE, MODEL, VARIANT*	*REGISTRATION*	*SE*	*ME*			FLIGHT			*DAY*	*NIGHT*
						TOTAL THIS PAGE									
						TOTAL FROM PREVIOUS PAGES									
						TOTAL TIME									

9				10								11				12
OPERATIONAL CONDITION TIME				PILOT FUNCTION TIME								FSTD SESSION				REMARKS AND ENDORSEMENTS
NIGHT		*IFR*		*PIC*		*CO-PILOT*		*DUAL*		*INSTRUCTOR*		*DATE (dd/mm/yy)*	*TYPE*	*TOTAL TIME OF SESSION*		
																I certify that the entries in this log are true.
																PILOT'S SIGNATURE

Il·lustració 2 - Format de "*logbook*" europeu. Font: European Aviation Safety Agency.

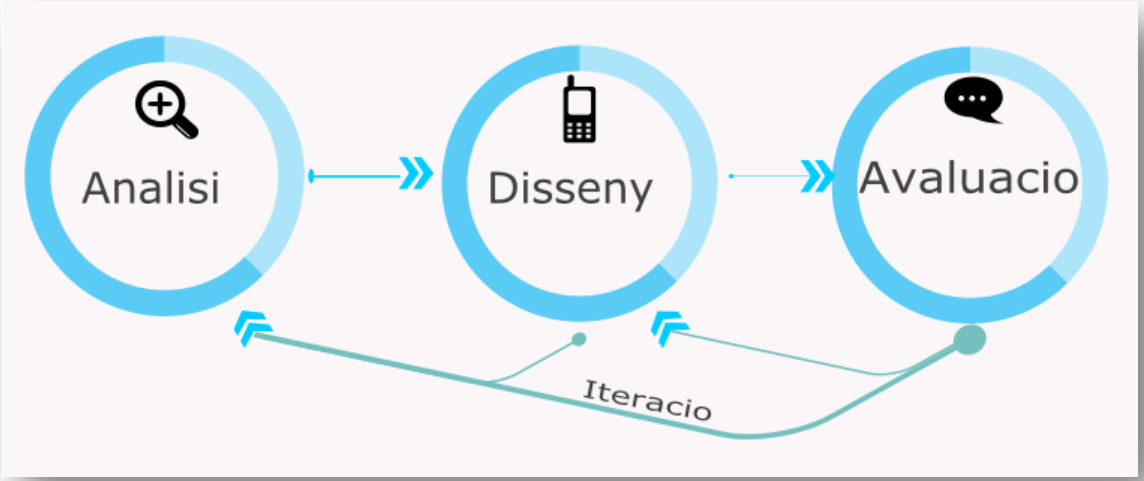
1.3 Objectius.

L'objectiu d'aquest treball s'emmarca dins l'assignatura de desenvolupament d´aplicacions per dispositius mòbils, en concret, els objectius són:

- Aplicar en un projecte els conceptes adquirits durant la carrera. Per exemple, programació orientada al objecte amb el llenguatge Java. Base de dades perquè l'aplicació utilitza *SQLite*. I tècniques de desenvolupament de programari com a assignatures més rellevants per al desenvolupament d'aquest projecte.
- Dissenyar una aplicació des de el seu borrador fins al seu producte final.
- Aprendre a desenvolupar una aplicació per a Android. Des de la instal·lació del SDK, passant per al *debuging*, *logs* ... fins a la compilació final.
- Implementar la interfície gràfica d'una aplicació Android.
- Utilitzar base de dades *SQLite* dins l'entorn Android.
- Finalment, obtenir una aplicació per a un dispositiu Android.

1.4 Mètode de treball.

El mètode de treball utilitzat és el disseny centrat en l'usuari (DCU). Aquest mètode es divideix en tres grans blocs, anàlisis, disseny i avaluació. Aquests mètode és un mètode iteratiu quan acabem una fase passem a la següent, però les fases es retro alimenten, per tant podem tornar a fases anteriors per millorar-les. Durant l'elaboració d'aquest projecte, per exemple, a sigut el cas de la pantalla principal de l'aplicació que ha sigut redissenyada per millorar-ne la implementació i disseny, com més endavant es comentarà.



Il·lustració 3 - Mètode centrat en l'usuari (DCU)

1.5 Recursos de desenvolupament.

Per al desenvolupament de l'aplicació s'utilitzarà una plataforma de desenvolupament i un maquinari per a provar l'aplicació.

Plataforma de desenvolupament:

Ordenador amb sistema operatiu Ubuntu 12.10 64bits.

ADT Bundle (Android Developer Tools), que inclou:

- Eclipse més ADT plugin.
- Android SDK Tools.
- Android Platform-tools.
- Plataforma Android.
- Imatges del sistema Android per a el emulador.

Java JDK 1.6.

Biblioteques externes:

- Itext. Aquesta llibreria permet crear arxius pdf.
- AndroidPlot. Aquesta llibreria permet crear gràfiques de dades.

Maquinari per a test:

Samsung Galaxy S I9000 amb Android v2.3.3

2 Requisits de l'aplicació.

En els següents apartats es descriu els requisits de l'aplicació obtinguts durant la fase d'anàlisis del procés de disseny centrat en l'usuari. Les tècniques utilitzades per a la indagació i obtenció dels usuaris i context d'ús són la anàlisis competitiva o comparativa (Benchmarking), l' observació i investigació textual i l'entrevista personal.

La anàlisis comparativa ha sigut breument explicada durant l'apartat 1.1 sobre la motivació del projecte.

La observació i investigació contextual ha sigut possible gràcies a la condició de pilot d'helicòpter de l'autor del treball. Això ha permès la realització d'entrevistes personals a companys per a la obtenció de requisits.

2.1 Usuaris.

Les aplicacions mòbils són aplicacions molt més acotades, en quant a funcionalitats, que les aplicacions que estem acostumats a utilitzar en els ordenadors personals. Aquest fet provoca que les aplicacions siguin dissenyades per uns perfils d'usuari molt mes específics també. La especificitat d'aquestes aplicacions arriba al punt que per a una funcionalitat molt similar dos actors diferents utilitzin aplicacions diferents.

Aquesta aplicació no és una excepció al cas comentat ja que ha sigut dissenyada per a un usuari bastant específic. L'usuari de l'aplicació serà un pilot de l'aviació, queden exclosos aquells pilots de línies aèries que, a més a més del registre de vols legal, vulguin guardar informació extra, com ara els salts realitzats durant un vol. Durant la fase d'anàlisi comparativa s'ha pogut observar que són varies les opcions disponibles dissenyades per a aquests usuaris.

Així doncs podem definir l'usuari com a un pilot d'avió, avioneta, ultralleuger, helicòpter, globus, etc... que vol mantenir un diari de vols en un format legal estàndard.

Es de esperar que la majoria d'usuaris seran persones amb estudis mitjans o superiors, encara que existeix una categoria de pilots privats dels quals pot ser que no tinguin estudis.

Demogràficament parlem de persones d'edat compresa entre 17 anys com a mínim, a 99 anys si són pilots privats que superen el certificat mèdic, cas molt excepcional. La majoria parlarem de persones de 20 a 60 anys.

L'experiència en l'ús de tecnologies de la informació es molt variada, encara que la majoria tenen coneixements mínims d'ofimàtica, correu electrònic,... Respecte a l'ús de tecnologia mòbil, encara que els resultats són variats, si que es pot observar que una gran majoria utilitza telèfons intel·ligents.

En quant al coneixement de l'anglès, encara que el seu coneixement és molt variat des de res a un nivell OACI 6 (pràcticament idioma matern) , si que podem esperar que coneguin vocabulari i expressions pròpies del món de l'aviació, com pot ser, flight, pilot in command, multipilot time, etc...

2.2 Requisits funcionals.

A continuació es detallen els requisits funcionals recopilats durant la fase d'anàlisi.

- **Funcionalitats referents a la informació:**
  - Crear, modificar, esborrar un vol nou.
  - Crear, modificar, esborrar una aeronau.
  - Crear, modificar, esborrar un pilot nou.
- **Introducció intel·ligent de dades**, l'aplicació mostrarà el valor mes probable al introduir les dades. Aquesta funcionalitat ha d'ajudar a fer de l'experiència, sempre carregosa, de introducció de dades una experiència molt més àgil, que a més a més ajudi a no cometre errors.
- **Creació de camps a mida**. L'aplicació podria tenir la possibilitat d'afegir algun camp nou als camps estrictament legals, així l'usuari podria tenir un seguiment, per exemple, de descàrregues d'aigua fetes des de un helicòpter d'incendis.
- **Resum de totals de hores per camps**. Aquesta és una de les funcions principals d'un registre de vols, obtenir la suma total de hores de vol i també diferenciades per tipus, com ara vol instrumental, nocturn,...
- **Resum personalitzat per dates**. Així com normalment només ens interessa el còmput total dels vols, és un extra del fet de tenir el "logbook" digital poder per exemple saber quantes hores s'han volat durant l'últim any.
- **Visualització dels vols realitzats per pàgines**. Aquesta funcionalitat prové del fet que l'aplicació està dissenyada per a que l'experiència sigui tant propera com sigui possible a la introducció i consulta de dades en un llibre tradicional en que es passen pàgines per a veure els vols.
- **Capacitat d'exportar les dades a csv o algun altre format per a còpia de seguretat**. L'usuari ha de tenir confiança en que no perdrà la informació introduïda, recuperar-la mes tard es casi impossible en molts casos.
- **Capacitat d'exportar les dades a format JAA en un fitxer pdf.** Com s'ha comentat es pretén obtenir un registre de vols que segueixi la legalitat europea, per tant, també s'haurà de poder imprimir en aquest format.
- **Optimització per a telèfon i tauleta.** L'ideal seria obtenir un producte que fos capaç de funcionar amb pantalles més o menys petites, el mòbil possiblement serà la font d'introducció de dades, ja que s'estima que es el que l'usuari sempre porta amb ell. En canvi al utilitzar l'aplicació en una tauleta podria aprofitar les avantatges d'una pantalla més gran per mostrar major informació.
- **Idiomes català, anglès.** A quants més idiomes estigui traduïda l'aplicació més usuaris podran utilitzar-la amb mes naturalitat. Però com ja s'ha explicat una gran majoria de usuaris tindran un nivell d'anglès aeronàutic suficient per utilitzar-la.

- **Gràfiques per anys, mesos, etc.. i també per totals per aeronau, nocturn/diürn, etc..** Aquesta suposa una altre avantatge addicional d'utilitzar una aplicació envers a un mètode tradicional, i és la visualització de la informació en forma de gràfica.
- **Sincronització amb altres dispositius.** Si l'aplicació es usable en mòbils i tauletes és lògic pensar que un mateix usuari la utilitzarà en els dos dispositius, per tant hauràn d'estar sincronitzats.

2.3 Requisits d'informació.

Els requisits d'informació d'aquesta aplicació són fàcilment visibles a través de la capçalera d'un diari de vols.

1	2		3		4		5			6	7	8	
DATE dd/mm/yy)	DEPARTURE		ARRIVAL		AIRCRAFT		SINGLE-PILOT TIME		MULTI-PILOT TIME	TOTAL TIME OF FLIGHT	NAME(S) PIC	LANDINGS	
	PLACE	*TIME*	*PLACE*	*TIME*	*MAKE, MODEL, VARIANT*	*REGISTRATION*	*SE*	*ME*				*DAY*	*NIGHT*

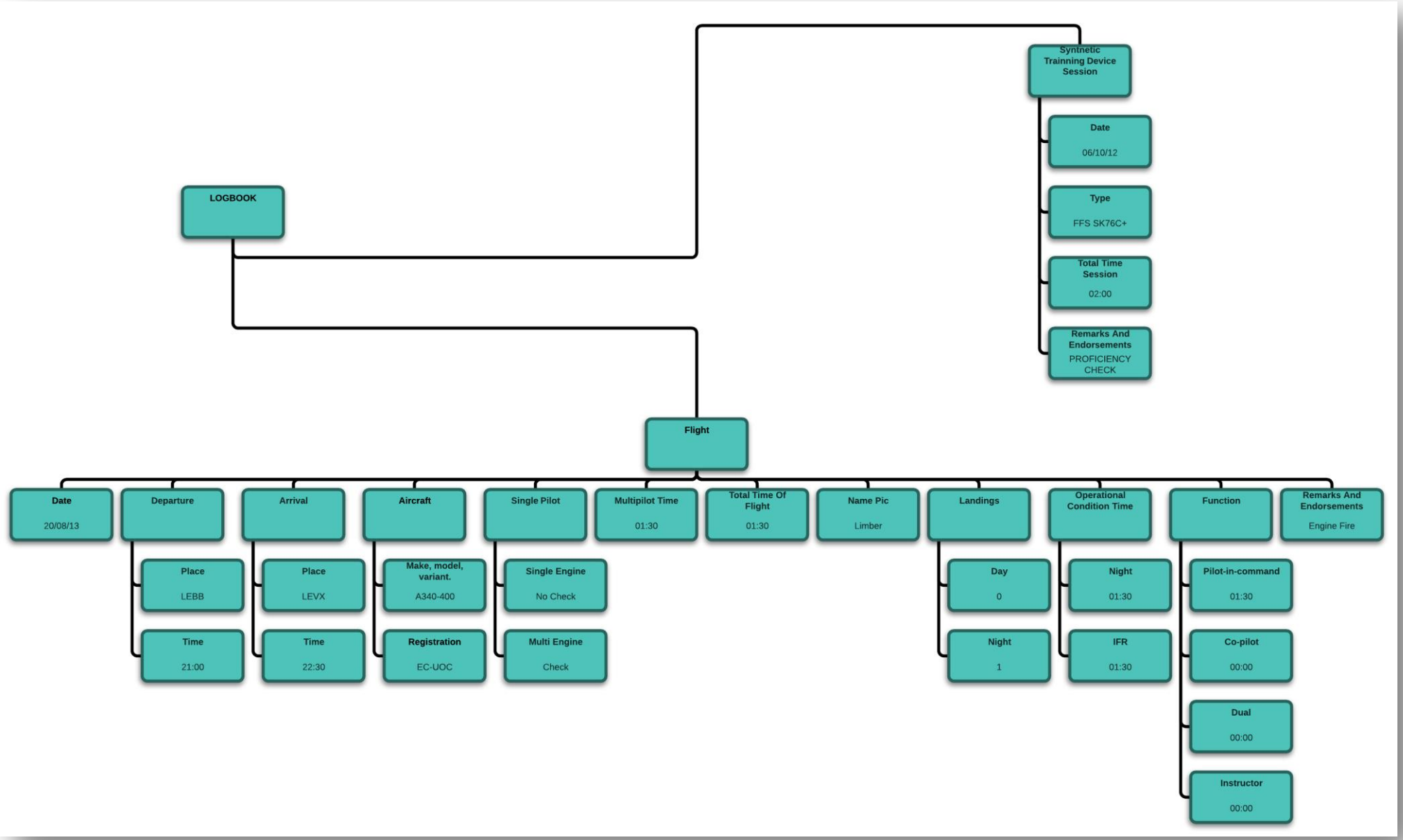
9		10				11			12
OPERATIONAL CONDITION TIME		PILOT FUNCTION TIME				FSTD SESSION			REMARKS AND ENDORSEMENTS
NIGHT	*IFR*	*PIC*	*CO-PILOT*	*DUAL*	*INSTRUCTOR*	*DATE (dd/mm/yy)*	*TYPE*	*TOTAL TIME OF SESSION*	

Il·lustració 4- Capçalera d'un diari de vol europeu.

Cada columna correspon a alguna dada que ha de ser guardada. En l'apartat de glossari i termes s'hi descriu que significa cada terme.

En l'apartat disseny de la persistència, s'explicarà com s'ha dissenyat la base de dades per a contenir aquesta informació.

A continuació es mostra un esquema de la informació en que cada terme, que li correspon, conté una dada d'exemple. També es pot observar com la informació es divideix en dos grans blocs, els vols (*Flight*) i vols de simulador (*Synthetic Training Device Session*). Només el camp "*Remarks and endorsements*" es compartit per als dos tipus de vols.



Il·lustració 5-Informació continguda en un registre de vol.

3 Anàlisis del sistema.

Aquest apartat mostra els casos d'ús, primerament de forma gràfica i posteriorment podrem veure'n la fitxa detallada de cada un. Aquest apartat doncs, és el resultat de l'anàlisi, dins el procés de disseny centrat en l'usuari.

3.1 Diagrames de casos d'ús.

Il·lustració 6- Esquema de casos d'ús.

3.2 Descripció de casos d'ús.

A continuació s'ha creat una fitxa per a cada cas d'ús de l'aplicació, només s'ha omes l'actor ja que, com s'ha explicat anteriorment, només n'hi ha un que és un pilot d'aviació.

Nom	**Logbook**
Resum	L'actor serà capaç de veure tots els registres de vols i vols de simulador en forma de pàgines.
Pre condicions	Cap
Post condicions	Cap
Flux normal	- El cas comença després de la visualització de la portada de l'aplicació o a través del menú d'opcions seleccionant "Logbook". - La primera pàgina que apareix són els últims vols ordenats per data. - L'actor pot navegar una pàgina enrere prement el botó menys, o al principi del llibre prement llargament el botó menys. - L'actor navegarà pàgines endavant prement el botó més, i si s'aguanta la pressió es navega a l'última pàgina. - Si es prem sobre un registre s'obre el cas d'ús visualització de registre.
Flux alternatiu	- En cas de clicar llargament sobre un registre s'inicia el cas d'ús edició de vol. - En cas de clicar llargament sobre un registre buit s'inicia el cas d'us introducció d'un vol.
Inclusions	Cap
Extensions	- Visualització d'un registre. - Introducció de un registre. - Edició de registre.

Nom	**Visualització d'un registre**
Resum	L'actor visualitzarà totes les dades d'un registre.
Pre condicions	Es rep el ID que identifica el registre a la base de dades.
Post condicions	Cap
Flux normal	- L'actor visualitzarà totes les dades que conté el registre. - Es prem cancel·la i es tanca la pàgina.
Flux alternatiu	Es podrà iniciar el cas d'us de edició de registre.
Inclusions	Cap
Extensions	Edició registre

Nom	Introducció d'un registre
Resum	L'actor podrà introduir les dades d'un registre.
Pre condicions	Cap
Post condicions	S'haurà gravat un nou registre a la base de dades.
Flux normal	- L'actor introdueix les dades d'un registre. - Es guarden les dades al prémer el botó guardar. - Apareix un missatge de confirmació de dades guardades.
Flux alternatiu	Es prem cancel·lar i es tanca la pantalla sense introduir les dades.
Excepcions	En cas de no poder introduir les dades a la base de dades apareix un missatge d'error.
Inclusions	Cap
Extensions	Cap

Nom	Edició d'un registre
Resum	L'actor podrà modificar un registre ja introduït.
Pre condicions	Es rep el ID que identifica el registre a la base de dades.
Post condicions	S'actualitza el registre a la base de dades.
Flux normal	- L'actor modifica o amplia les dades ja introduïdes. - S'actualitzen les dades al prémer el botó guardar. - Apareix un missatge de confirmació de dades guardades.
Flux alternatiu	Es prem cancel·lar i es tanca la pantalla sense actualitzar les dades.
Excepcions	Si el registre no pot ser actualitzat apareix un missatge d'error.
Inclusions	Cap
Extensions	Cap

Nom	Visualització d'un registre de simulador
Resum	L'actor podrà veure la informació del registre de simulador.
Pre condicions	Es rep el ID que identifica el registre a la base de dades.
Post condicions	Cap
Flux normal	- L'actor visualitza les dades. - Es prem cancel·la i es tanca la pàgina.
Flux alternatiu	Es podrà prémer el botó editar per obrir la pantalla d'edició del registre.
Inclusions	Cap
Extensions	Edició d'un registre de simulador.

Nom	Introducció d'un registre de simulador
Resum	L'actor podrà introduir les dades de una sessió de simulador.
Pre condicions	Cap
Post condicions	S'haurà introduït un nou registre de simulador a la base de dades.
Flux normal	- L'actor introdueix les dades. - Es prem el botó guardar i s'introdueixen les dades a la base de dades. - Apareix un missatge de confirmació de dades guardades.
Flux alternatiu	Es prem cancel·la i es tanca la pantalla.
Excepcions	No es poden introduir les dades a la base de dades i apareix un missatge d'error.
Inclusions	Cap
Extensions	Cap

Nom	Edició d'un registre de simulador
Resum	L'actor podrà modificar un registre de simulador ja introduït.
Pre condicions	Es rep el ID que identifica el registre de simulador a la base de dades.
Post condicions	S'actualitza el registre a la base de dades.
Flux normal	- L'actor modifica o amplia les dades ja introduïdes. - S'actualitzen les dades al prémer el botó guardar. - Apareix un missatge de confirmació de dades guardades.
Flux alternatiu	Es prem cancel·lar i es tanca la pantalla sense actualitzar les dades.
Excepcions	Si el registre no pot ser actualitzat apareix un missatge d'error.
Inclusions	Cap
Extensions	Cap

Nom	Introducció – Edició d'un pilot
Resum	L'actor podrà introduir les dades d'un pilot nou, o si ja existeix s'actualitzaran en la base de dades.
Pre condicions	Cap
Post condicions	S'introduirà o actualitzarà un pilot a la base de dades.
Flux normal	- L'usuari introdueix les dades. - Es prem el botó guardar. - Apareix un missatge de confirmació de dades guardades
Flux alternatiu	Es prem el botó cancel·la i se surt de la pantalla.
Excepcions	Si no es poden introduït les dades a la base de dades apareix un missatge d'error.
Inclusions	Cap
Extensions	Cap

Nom	Introducció – Edició d'una aeronau
Resum	L'actor podrà introduir les dades d'una aeronau nova o si ja existeix s'actualitzaran en la base de dades.
Pre condicions	Cap
Post condicions	S'introduirà o actualitzarà una aeronau a la base de dades.
Flux normal	- L'usuari introdueix les dades. - Es prem el botó guardar. - Apareix un missatge de confirmació de dades guardades
Flux alternatiu	Es prem el botó cancel·la i se surt de la pantalla.
Excepcions	Si no es poden introduït les dades a la base de dades apareix un missatge d'error.
Inclusions	Cap
Extensions	Cap

Nom	Veure totals
Resum	L'actor podrà veure la suma de hores de vols.
Pre condicions	Cap
Post condicions	Cap
Flux normal	- Es poden consultar els totals de tots els camps del diari de vol. - Es pot navegar amunt i avall per els totals. - Es torna al menú prement enrere.
Flux alternatiu	Es prem el botó personalitzats i s'obra la pantalla de totals parcials.
Inclusions	Cap
Extensions	Veure totals parcials.

Nom	Veure totals parcials
Resum	L'actor podrà veure la suma de hores de vols personalitzada
Pre condicions	Cap
Post condicions	Cap
Flux normal	- L'actor seleccionarà un espai temporal (per exemple: 1 de Gener de 2012 a 31 de Desembre de 2012) o bé seleccionarà algun dels camps (per exemple: model d'aeronau) o ambdues coses a la vegada. - Es mostrarà el total de hores segons els criteris escollits. - Es torna al menú prement enrere.
Flux alternatiu	Cap
Inclusions	Cap
Extensions	Cap

Nom	Exportar a pdf
Resum	L'actor obtindrà un arxiu pdf amb el contingut de la base de dades.
Pre condicions	Cap
Post condicions	Obtenim un arxiu pdf amb el format europeu de "logbook" i tots les registres de l'aplicació.
Flux normal	- L'actor seleccionarà el botó exportar pdf. - Visualitzarà un missatge d'espera. - Visualitzarà un missatge de finalització.
Flux alternatiu	L'actor podrà canviar el destí de l'arxiu.
Excepcions	En cas de no poder crear l'arxiu.
Inclusions	Cap
Extensions	Cap

Nom	Gràfiques estadístiques
Resum	L'actor podrà visualitzar les dades en gràfiques.
Pre condicions	Cap
Post condicions	Cap
Flux normal	- L'actor tria quin tipus de gràfica vol visualitzar. - Es visualitzarà la gràfica segons el criteri escollit. - Se surt amb el botó enrere.
Flux alternatiu	Es cancel·la el cas d'ús i se surt.
Excepcions	No es pot accedir a la base de dades
Inclusions	Cap
Extensions	Cap

Nom	Exportar base de dades
Resum	L'actor obtindrà un arxiu amb el contingut de la base de dades.
Pre condicions	Cap
Post condicions	Obtenim un arxiu amb la base de dades de l'aplicació que serà guardat al mòbil.
Flux normal	- L'actor seleccionarà el botó exportar base de dades. - Visualitzarà un missatge de finalització.
Flux alternatiu	L'actor podrà canviar el destí de l'arxiu.
Excepcions	En cas de no poder crear l'arxiu.
Inclusions	Cap
Extensions	Cap

Nom	Exportar base de dades en format .csv
Resum	L'actor obtindrà un arxiu amb el contingut de la base de dades.
Pre condicions	Cap
Post condicions	Obtenim un arxiu en format csv amb el contingut de la base de dades.
Flux normal	- L'actor seleccionarà el botó exportar base de dades. - Visualitzarà un missatge de finalització.
Flux alternatiu	L'actor podrà canviar el destí de l'arxiu.
Excepcions	En cas de no poder crear l'arxiu.
Inclusions	Cap
Extensions	Cap

Nom	Creació de camps
Resum	L'actor pot afegir un camp personalitzat a la base de dades.
Pre condicions	Cap
Post condicions	La base de dades contindrà un nou camp per registre.
Flux normal	- Se selecciona el tipus de dada a guardar (text, nombre o hores i minuts). - Es dona un títol al nou camp.
Flux alternatiu	Cap
Excepcions	No s'aconsegueix actualitzar la base de dades.
Inclusions	Cap
Extensions	Cap

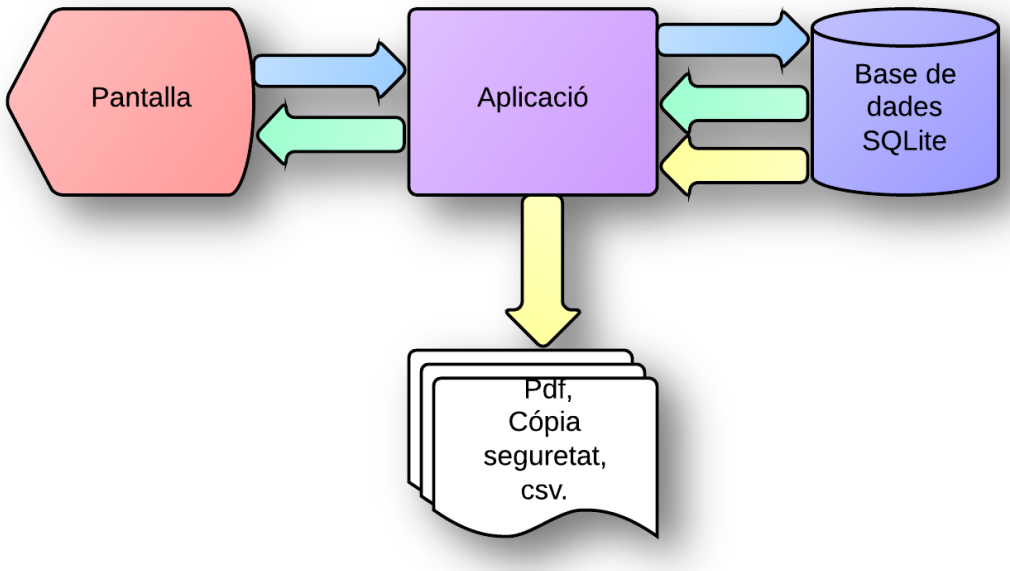
Nom	Sincronització
Resum	L'actor pot forçar la sincronització de la base de dades (encara que l'aplicació s'autosincronitzarà)
Pre condicions	El dispositiu té connexió a internet.
Post condicions	La base de dades està sincronitzada al núvol.
Flux normal	- Es premerà el botó sincronitza. - Apareix un missatge de sincronitzant. - Apareix un missatge de sincronitzat.
Flux alternatiu	Es prem el botó cancel·la i es retorna a la pantalla anterior.
Excepcions	No hi ha connexió a internet.
Inclusions	Cap
Extensions	Cap

4 Disseny.

A continuació es passa a detallar el disseny de l'aplicació, aquests pròxims apartats corresponen a la fase de disseny de l'aplicació dins els procés DCU.

4.1 Arquitectura de l'aplicació.

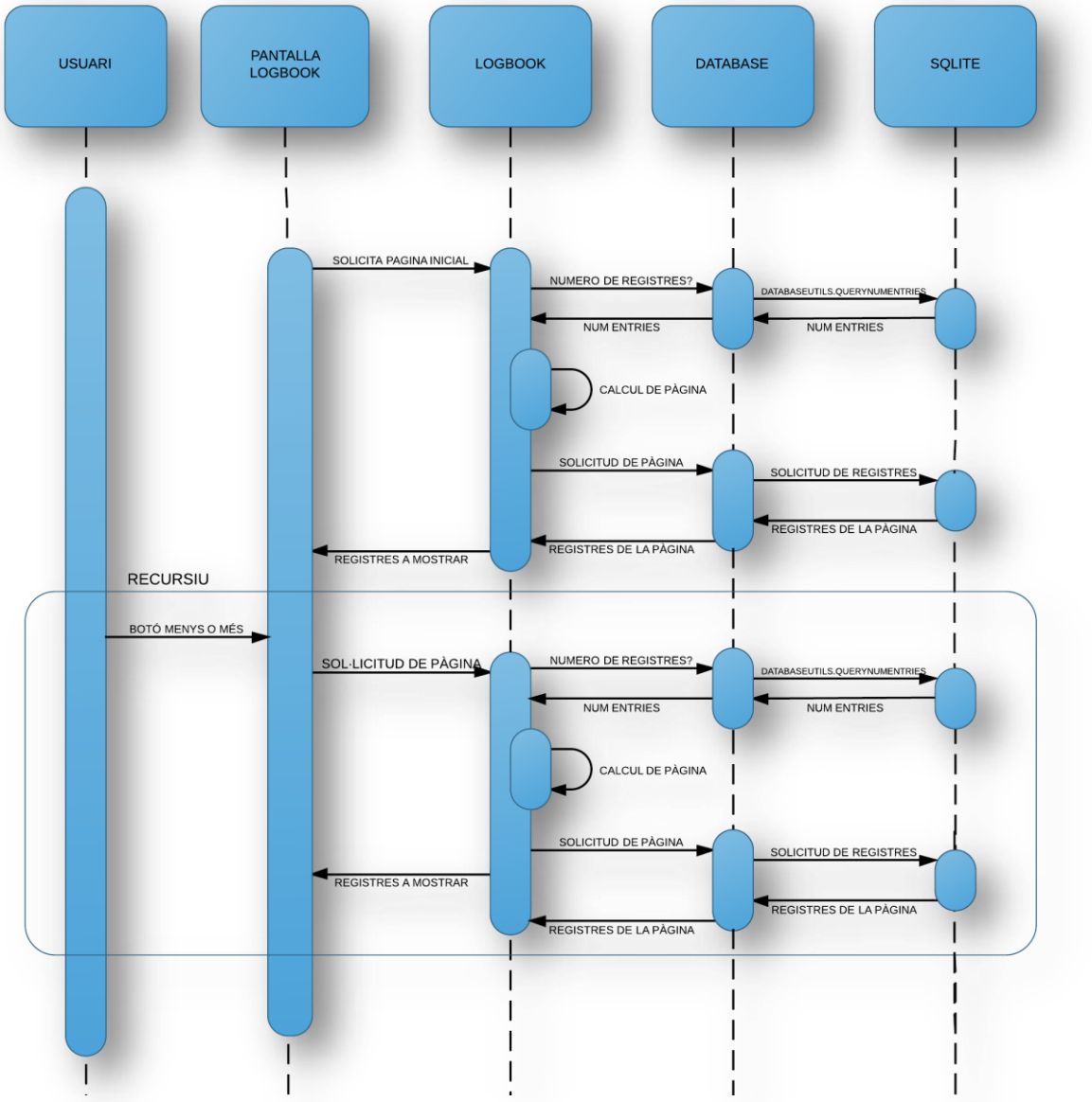
L'arquitectura de l'aplicació no és molt complexa, consisteix en que l'usuari introdueix les dades per pantalla, l'aplicació guarda les dades a la base de dades i a la vegada la base de dades retorna els registres per a ser mostrats a l'usuari o bé per exportar-los a un arxiu extern a l'aplicació.



Il·lustració 7 - Arquitectura de l'aplicació.

4.2 Diagrama de seqüencia.

A continuació es mostra el diagrama de seqüencia del cas d'ús titulat "*Logbook*". Es mostra aquest cas d'ús per la importància que té dins l'aplicació, és el cas d'ús que defineix l'aplicació. I també perquè permet veure com és l'accés a la base de dades.



Il·lustració 8 - Cas d'ús Logbook

4.3 Disseny de la persistència.

4.3.1 Model relacional de la base de dades

A continuació es descriu la base de dades segons el model relacional:

FLIGHTS (RowId, Date, DepartureTime, DeparturePlace, ArrivalTime, ArrivalPlace, AircraftRegistration, AircraftModel, SingleEngine, MultiEngine, MultiPilotTime, TotalTime, NamePic,LandingsDay, LandingsNight, OperationalNight, OperationalIFR, TimeInCommand, TimeCopilot, TimeDual, TimeInstructor, Remarks)

AircraftRegistration és clau forana de AIRCARFTREGISTRATION, AircraftModel és clau forana de AIRCRAFTMODEL NamePIC és clau forana de PIC

SYNTHETICTRAINING (RowId, DateSyn, Type, TimeSession, Remarks)

AIRCRAFTREGISTRATION (RowId, Registration, Model)

Model és clau forana de AIRCARFTMODEL

AIRCRAFTMODEL (RowId, Model, Make)

PIC (RowId, NamePic, Name, Surname)

Es pot observar que contràriament al que es recomana en un model relacional, totes les taules conten un camp *RowId*, que identifica cada fila dins la base de dades, en comptes d'utilitzar aquells camps que fan de cada entrada única. Això és així degut a que la base de dades serà implementada en SQLite.

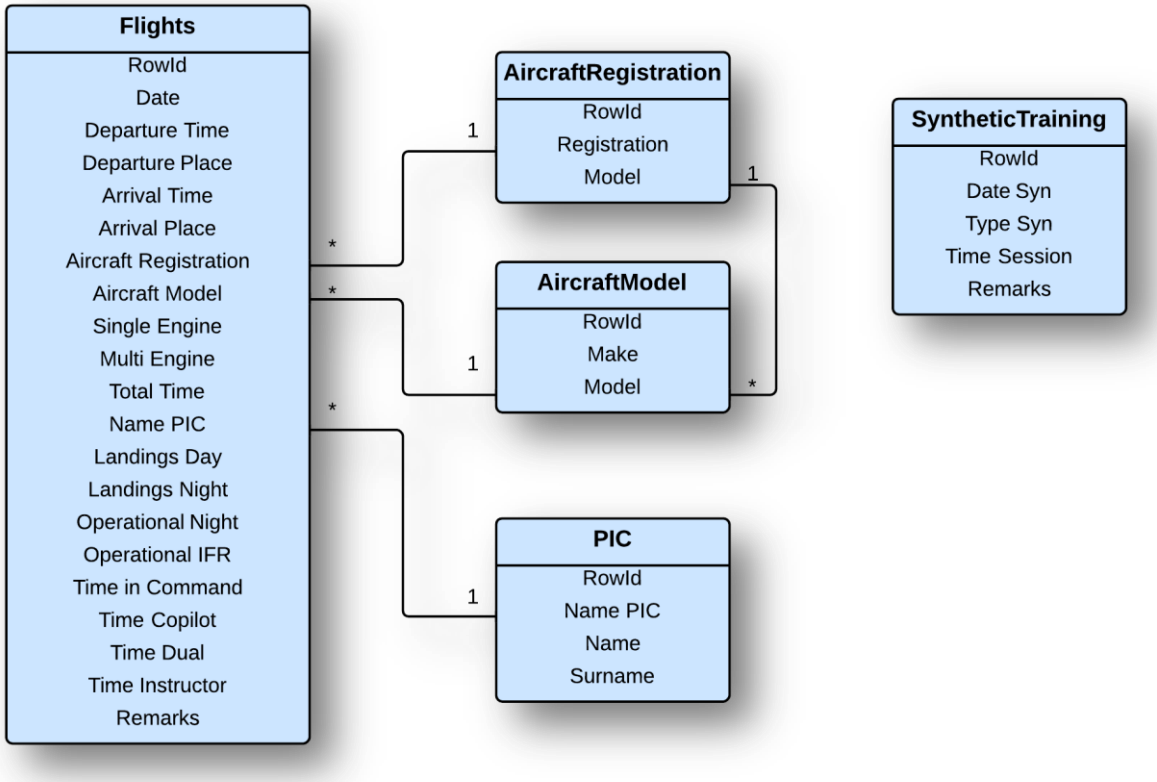
SQLite provoca un error si intentem crear una taula que utilitza dos camps com a clau primària. Aquest seria el cas de la taula Flights, per exemple, que utilitzaria el camp "*Date*" i el camp "*DepartureTime*" per identificar cada entrada de la base de dades. Concretament la documentació diu: "*If there is more than one PRIMARY KEY clause in a single CREATE TABLE statement, it is an error.*"

De fet SQLite accepta valors "*NULL*" dins un camp identificat com a "*PRIMARY KEY*", amb tots els problemes que això pot portar. El motiu d'aquesta implementació, segons la documentació, és, "*We could change SQLite to conform to the standard (and we might do so in the future), but by the time the oversight was discovered, SQLite was in such wide use that we feared breaking legacy code if we fixed the problem*."

Per això SQLite crea automàticament un camp *RowId*, que dona un valor numèric únic per fila, que no permet valors "*NULL*". Els valors d'aquest camp serà del tipus numèric

auto increment, quan ja no queden mes nombres per seguir el increment aleshores comença a buscar nombres buits.

4.3.2 Esquema de la base de dades

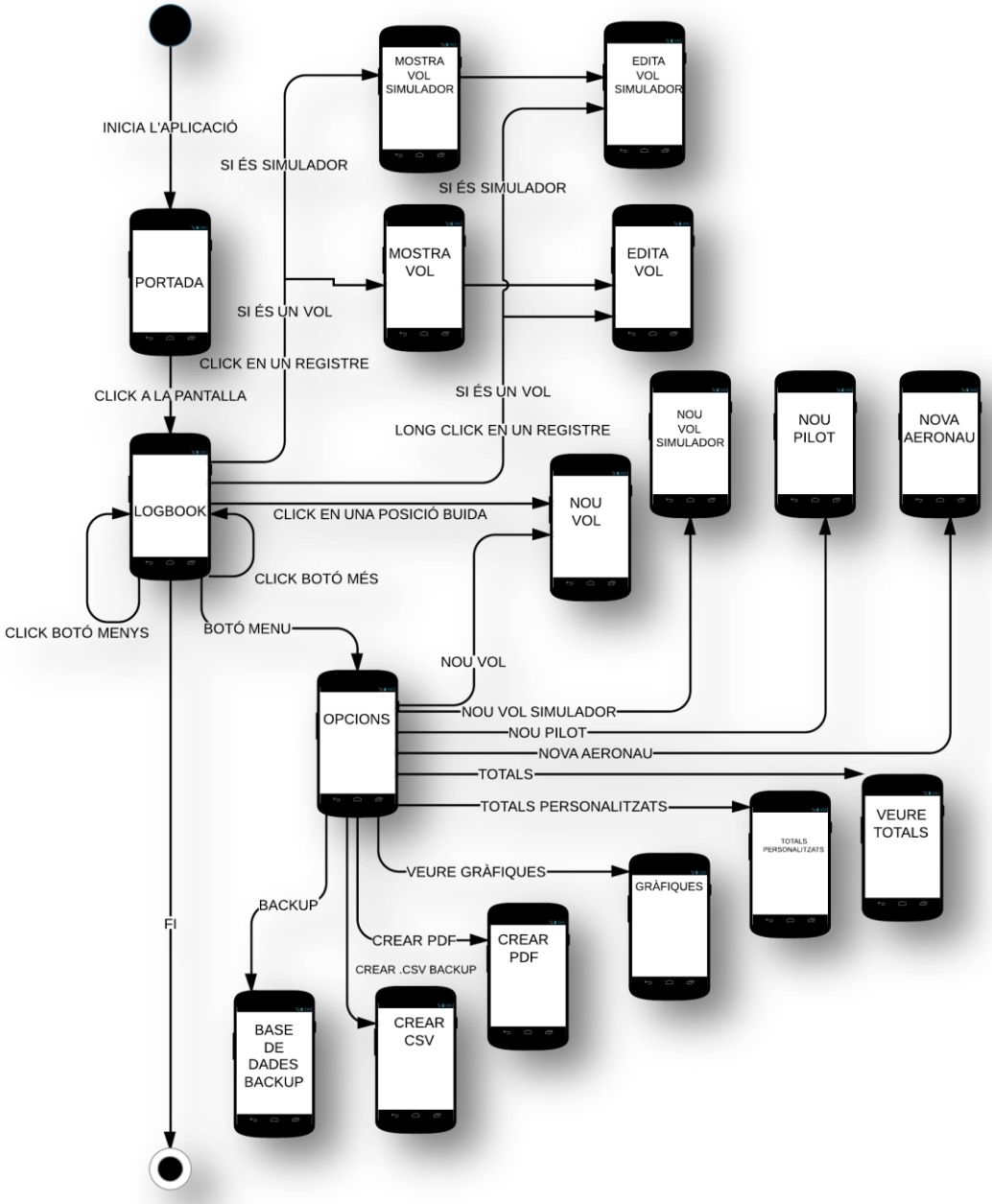


Il·lustració 9 - Persistència

4.4 Prototipatge.

En aquest apartat es mostra el disseny gràfic de l'aplicació. Primerament veurem el flux d'interacció de les diferents pantalles i seguidament els prototips.

4.4.1 Flux d'interacció.



Il·lustració 10 - Flux d'interacció.

S'ha omes, per no posar masses línies, que cada pantalla pot tornar a la pàgina anterior quan l'usuari vol. A comentar que les pantalles nou vol i edita vol són iguals

però en la de edició les dades venen pre-omplertes en la pantalla editar. Passa el mateix amb nou vol simulador i edita vol simulador.

4.4.2 Prototips.

Seguidament es mostren els prototips de les pantalles principals de l'aplicació, els "*sketches*" no es mostren perquè no aporten gran informació, ja que en realitat són els esborranys dels prototips.

Il·lustració 11-Pantalla portada

Algunes de les pantalles han sofert modificacions posteriors, aquestes modificacions es justifiquen i mostren en el següent apartat conclusions del procés DCU.

Aquesta pantalla correspon a la portada de l'aplicació. La intenció es que l'usuari tingui una experiència similar a la d'un llibre, on sempre comencem per la portada.

Il·lustració 12-Pantalla Logbook

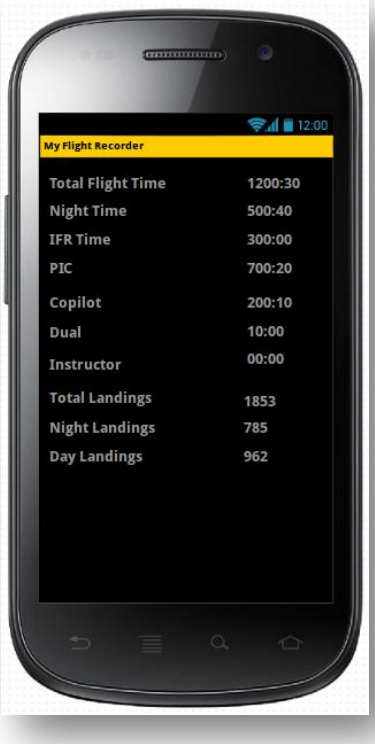
Aquesta és la segona pantalla de l'aplicació, i és la pantalla principal ja que és on podem consultar les dades introduïdes.

En un principi s'havia pensat en que des d' aquesta pantalla es vegues tota la informació de cada registre. Però sens dubte és massa informació per a una pantalla petita, seria una possible implementació per a una versió per a tauleta. Per tant, s'ha elegit la informació més important de cada registre. Per exemple, surt l'hora de enlairament i la durada del vol, així que es fàcil imaginar l'hora d'aterratge. També podem veure la data, l'aeronau, els aeroports de sortida i arribada i els possibles comentaris (*Remarks*). Suficient informació per a que el registre quedi ben definit.

Il·lustració 13-Pantalla introducció de vols

Aquest prototip correspon al disseny original de la pantalla d'introducció de dades d'un vol. La idea era anar seleccionant els camps i anar introduït les dades mentrestant es navegava descendent-ment.

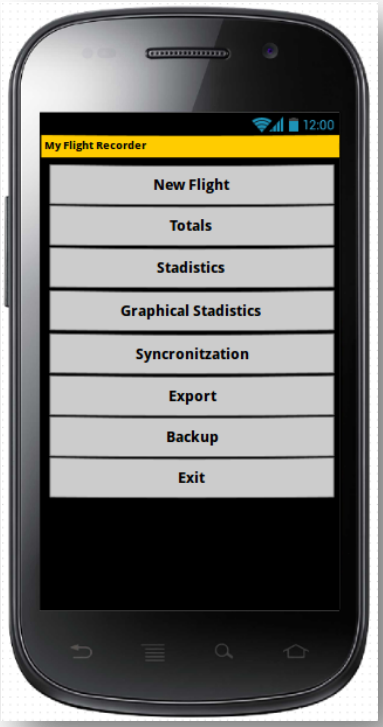
Però els prototips accepten qualsevol escala i mida, la realitat és, per exemple, que un "*Date Picker*", ocupa molt més o és pràcticament inusable ja que el dit és molt més gros.



Il·lustració 14-Pantalla totals

Aquesta pantalla correspon a la pantalla de totals. En ella podem veure el sumatori de les dades del diari de vols.

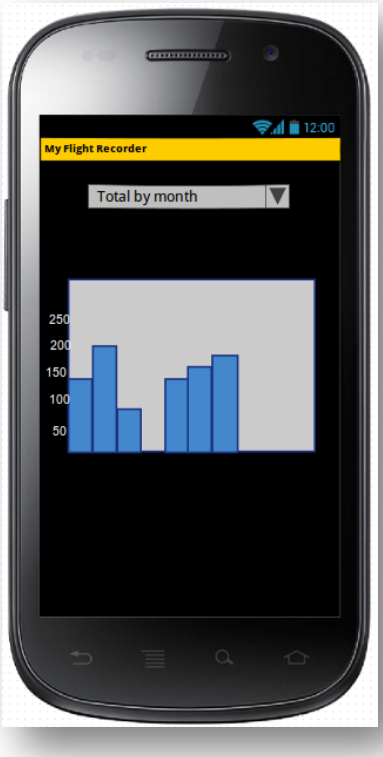
Les sumes corresponen a les sumes estàndard que obtenim en un "*logbook*" de paper. On tenim el total de hores volades i les hores volades per tipus, nit, IFR, copilot, etc..



Il·lustració 15-Pantalla opcions

Aquesta pantalla ofereix les opcions que l'usuari pot elegir dins l'aplicació.

Es base en un "*ListActivity*" que Android implementa nativament. Aquesta opció, a part de facilitar la implementació, també permet una ràpida adaptació de l'usuari que esta acostumat a utilitzar aquest tipus de menús.



Il·lustració 16-Pantalla gràfiques

El prototip mostra com l'usuari pot elegir quin tipus de gràfica vol visualitzar. Podria ser per mes, any, dies, etc...

Encara que en el prototip no es posa també podria incloure la possibilitat de seleccionar dues dates entre les quals veure la gràfica. Però s'ha desestimat de moment per no sobrecarregar el projecte.

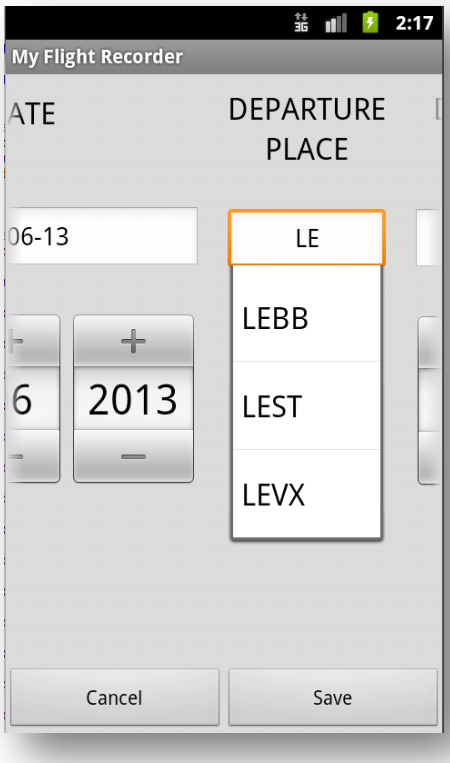
4.5 Conclusions del procés DCU.

El procés de disseny centrat en l'usuari és un procés iteratiu. Per tant quan acabem una fase no es dona per enllestida i se segueix amb la següent sinó que d'alguna manera les fases queden obertes per a possibles canvis.

La realitat és que quan millor és el disseny inicial més treball ens estalviem en les posteriors fases, ja que un canvi en els requisits de l'aplicació quan ja estem implementant, per exemple, poden suposar hores de feina perduda. Però per una altre part aconseguirem un millor producte final si modifiquem el disseny en cas de trobar-hi inconvenients.

Durant el procés d'aquest projecte s'han fet canvis en el disseny original. Alguns els podem atribuir a una millora de la aplicació, però també hi ha el factor aprenentatge.

Al ser aquesta la primera aplicació per Android que es dissenya, s'ha combinat el disseny amb l'adquisició dels coneixements necessaris. Lògicament, això a portat a canvis en les decisions inicials degut a les limitacions d'implementació.

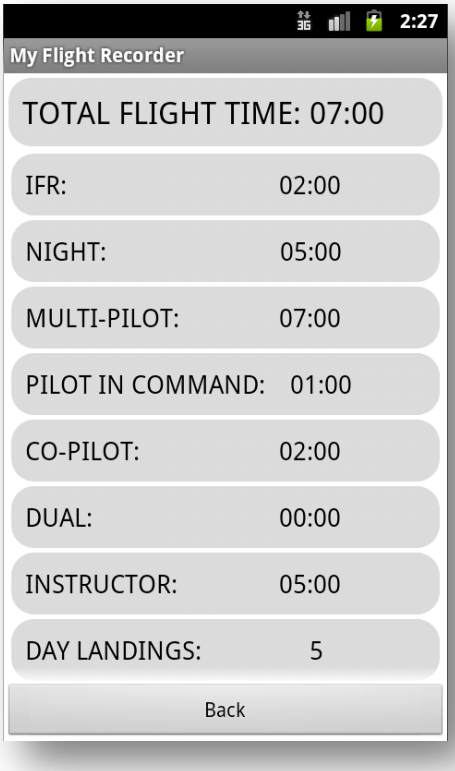


Il·lustració 17-Pantalla final d'introducció de vols

A l'esquerra podem observar el disseny final de la pàgina d'introducció de vols.

Degut a que era molt poc usable el disseny inicial per les mides mínimes del "*DatePicker*" i "*TimePickers*". S'ha optat per una idea que originalment s'havia descartat.

La pantalla comença per l'esquerra amb el camp "*DATE*" segueix amb "*DEPARTURE PLACE*" , "*DEPARTURE TIME*", etc... seguint l'odre establert de un "*Logbook*". Així, l'usuari, que ja està acostumat a aquest ordre, pot utilitzar l'aplicació amb mes naturalitat i anar omplint els camps tal i com h faria en un llibre de paper.



La pantalla d'estadístiques gràfiques finalment, no ofereix la possibilitat de triar entre mesos, anys, etc.. i només ofereix els vols volats per mesos, però en aquest cas ha sigut per la falta de temps en la implementació. I s'ha decidit implementar el cas mes comú, veure tots els vols dividits per mesos.

Per últim comentar que encara que s'ha conservat el disseny original de la pantalla de totals, tal i com es pot veure a l'esquerra, s'ha remarcat cada camp per separat per facilitar-ne la lectura i fer-la mes atractiva i s'ha augmentat la mida del camp "*TOTAL FLIGHT TIME*" ja que en realitat és la suma més important.

4.6 Qüestionaris.

Per a la realització d'aquest projecte s'han preparat uns qüestionaris per tal de dur a terme una fase d'avaluació davant d' usuaris.

L'avaluació és faria amb test d'usuaris, ja que és el test que permet avaluar la usabilitat de manera més fiable.

Per fer el test, primer es deixaria al usuari arrancar l'aplicació per veure si identifica sense informació prèvia de quina aplicació es tracta. Un cop l'usuari hagi identificat l'aplicació s'observaria que intenta fer per iniciativa pròpia i que n'espera en un primer moment.

Després s'encarregaria a l'usuari fer diferents tasques per veure si les troba de manera lògica o es perd buscant-les.

- El recull de preguntes d'informació sobre l'usuari que realitzaria el test.
  - Home o dona?
  - Quina edat tens?
  - Utilitzes ordenadors habitualment?
  - Tens experiència amb dispositius Android?
  - Has utilitzat mai un "*logbook*" electrònic?
  - Has trobar l'aplicació àgil?
  - Has trobat la informació que buscaves fàcilment?
  - En algun moment t'ha semblat que les dades eren tractades incorrectament?
  - Has sentit en algun moment que podries perdre les dades?
  - T'ha semblat visualment agradable?

- Les tasques que els usuaris haurien de realitzar.
  - Veure i navegar per les pàgines de vols.
  - Introduir un nou vol.
  - Editar un vol introduït.
  - Veure estadístiques.

- Les preguntes referents a les tasques.
  - Veure i navegar per les pàgines de vols.
    - T'ha semblat la pàgina de vols comprensible?
    - Has trobat tota la informació sobre vols introduïts suficient?
    - Et recorda la pàgina al teu "*logbook*"?
    -
  - Introduir un nou vol.

- T'ha ajudat l'aplicació a introduir les dades?
- T'ha semblat un procés llarg introduir les dades?
- Has sigut conscient d'haver guardar les dades correctament?
- T'ha ajudat l'aplicació a corregir alguna dada incorrecte?

- Editar un vol introduït.
  - Has trobat fàcilment com modificar un vol ja introduït?
  - T'ha sigut fàcil editar el camps?
  - Has sigut conscient de que les dades s'han guardat correctament?
  - T'ha sigut fàcil trobar el vol que volies editar?

- Veure estadístiques.
  - T'ha sigut fàcil trobar els totals?
  - Et sembla que hi falta alguna informació important?
  - Les estadístiques per camps t'han semblat fàcils de consultar?
  - Trobes a faltar alguna consulta de les dades?
  - Les gràfiques et semblen fàcils d'interpretar?
  - Falta alguna gràfica important?

5 Implementació.

En els següents apartats s'explica el procés d'implementació i el resultat final.

5.1 Procés d'implementació.

Per a la implementació és necessari primerament configurar l'entorn de desenvolupament que ens proporciona Android. Els pre-requisits d'aquesta plataforma utilitzats són:

- Ubuntu Linux versió 12.10 de 64 bits.
- Llibreria GNU C (glibc).
- Llibreria ia32-libs per a l'execució de programes de 32 bits.
- Java JDK 1.6

Amb aquests requisits s'ha descarregat el ADT "*Bundle*" (Android Developer Tools), que inclou:

- Eclipse més ADT plugin.
- Android SDK Tools.
- Android Platform-tools.
- Plataforma Android.
- Imatges del sistema Android per a el emulador.

Aquest paquet ADT de la pàgina oficial d'Android facilita la instal·lació, ja que en un sol paquet trobem totes les eines necessàries preconfigurades per a començar a desenvolupar l'aplicació. Abans de l'aparició d'aquest paquet s'havia de configurar manualment el sistema, el IDE eclipse, etc.. i portava molts més problemes.

Version	Codename	API	Distribution
1.6	Donut	4	0.1%
2.1	Eclair	7	1.5%
2.2	Froyo	8	3.2%
2.3 - 2.3.2	Gingerbread	9	0.1%
2.3.3 - 2.3.7		10	36.4%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	25.6%
4.1.x	Jelly Bean	16	29.0%
4.2.x		17	4.0%

Il·lustració 18-Dades sobre les versions Android en els dispositius actuals. Font: Android Developers

Amb tot això instal·lat ja podem començar a implementar. El primer que haurem de fer serà decidir quina és el versió mínima d'Android en la que es podrà executar l'aplicació.

En aquest cas s'ha decidit que l'API mínim serà el 10, corresponent a la versió 2.3.3 d'Android.

Perquè es disposa d'un dispositiu Android Sansung Galaxy i9000 amb la versió 2.3.3 i perquè l'API 10, a 3 de Juny del 2013, permet executar l'aplicació en un

95.1% de dispositius, tal i com es pot veure en la infografia de la esquerra.

També haurem de decidir la versió màxima en que l'aplicació s'ha provat i funciona, evidentment intentarem que sigui el màxim a dia d'avui, l'API 17.

Seguidament el IDE Eclipse ens permet crear automàticament una sèrie de paràmetres per a l'aplicació. Un dels paràmetres que s'ha utilitzat és la creació automàtica d'una icona per a la aplicació.

Degut a que no s'ha trobat cap imatge sense copyright que es pogués aplicar amb un mínim de qualitat, s'ha utilitzat una de les icones que ofereix el IDE Android com a de icona de l'aplicació. Per seguir la coherència en l'aplicació aquesta imatge icona s'ha utilitzat de portada, també. No era propòsit d'aquest treball dissenyar imatges personalitzades per a l'aplicació, encara que, de cara a publicar aquesta aplicació al públic general seria necessari crear una icona i una portada personalitzada. A la fi i al cap, estem dissenyant una aplicació centrada en l'usuari i els usuaris agraeixen molt un disseny personalitzat i cuidat.

Un cop finalitzats els primers passos podem començar a implementar l'aplicació.

Per a la implementació Android s'han de tenir en compte una sèrie d'aspectes respecte a la programació en Java. Que es detallen en el següent punt.

5.1.1 Programació per a Android

El llenguatge de programació Android és el Java, però fins aquí les similituds.

Java utilitza una "*Java Virtual Machine*" per a l'execució d'aplicacions, Android en canvi utilitza la màquina virtual Dalvik. Les Java *VM's* permeten la portabilitat de les aplicacions a múltiples sistemes operatius i plataformes, en canvi, Dalvik no permet aquesta portabilitat però així tenim una màquina virtual molt més optimitzada, de fet, Android no recomana escriure aplicacions natives per motius de rapidesa d'execució.

A continuació es comenten els aspectes més importants en quant a la programació Android, aquests aspectes es comenten perquè són nous o perquè són diferents en comparació a la programació Java.

- Les funcions System.out i System.err no s'utilitzen en Android en canvi utilitzarem la classe Log.
  Per exemple:

```
private static final String TAG = "DataBase";
```

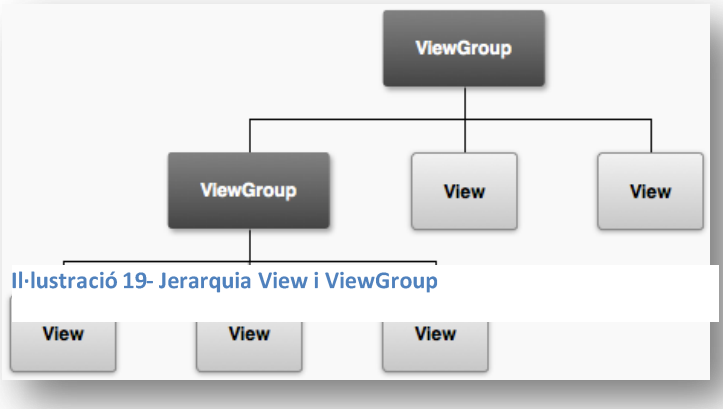
```
} catch (Exception e) {

        Log.e(TAG, "Error accesing to the database: Some Timestamp values
                    are null in the db");
        Log.e(TAG, e.getMessage());
              }
```

  Aquesta classe ens permetrà ensenyar missatges d'error però també del tipus WARN, INFO, DEBUG, VERBOSE.

- El "*framework*" en Android es troba en la classe View. La qual necessita de la classe Content. La interfície de l'usuari pot ser declarada en el propi codi utilitzant la jerarquia de Views i ViewGroup, tal i com mostra la il·lustració següent.

Encara que la forma més utilitzada per simplicitat i efectivitat és declarar la interfície gràfica a través d'un arxiu XML.



Il·lustració 19- Jerarquia View i ViewGroup

Quan l'aplicació crida un recurs XML aleshores, en temps d'execució, crea un objecte que podrà ser modificat, editat, etc...

Exemple d' ús de pantalla XML en la portada de l'aplicació:

La classe de Java que crida la pantalla és la classe COVER.

```
public class Cover extends Activity {

        ImageView image;

        @Override
        protected void onCreate(Bundle savedInstanceState) {

                super.onCreate(savedInstanceState);
                setContentView(R.layout.cover);
                image = (ImageView) findViewById(R.id.imageLogo);
                        image.setOnClickListener(new View.OnClickListener()

                                                etc ...
```

En concret la línia de codi "setContentView(R.layout.*cover*);" carga la pantalla cover.xml.

Les pantalles seran definides dins la carpeta "*layout*", en arxius amb noms en minúscules i acabats per .xml. Per a recuperar aquests recursos utilitzem la classe "*R.layout*". Però com recuperar recursos a través de la classe "*R*" o veurem en el pròxim punt. Seguidament podem observar l'arxiu XML de la pantalla 'portada' de l'aplicació. Que és formada per un "*LinearLayout*" vertical que conté una imatge central.

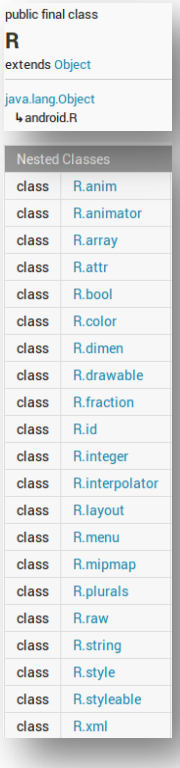
L'arxiu cover.xml, és:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/grey"
    android:contentDescription="@string/appLogo"
    android:gravity="center"
    android:orientation="vertical" >

    <ImageView
        android:id="@+id/imageLogo"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:clickable="true"
        android:contentDescription="@string/appLogo"
        android:src="@drawable/ic_launcher" />

</LinearLayout>
```

- Els recursos en Android tenen un nom que és únic. Així el sistema pot identificar-los des del codi Java. Per utilitzar-los farem servir les classes R, que podem visualitzar en la il·lustració de la dreta.



  Per exemple, en el punt anterior veiem que la classe "*Cover*" de Java utilitzava l'arxiu cover.xml a través de la crida:
  "setContentView(R.layout.*cover*);" .

  També, en l'exemple anterior, podem veure que en l'arxiu cover.xml el "ImageView" té un camp que és: "android:id="@+id/imageLogo" " amb aquest camp donem una identificació única a la imatge ("imageLogo"), per després utilitzar-la en el codi.
  I efectivament es pot veure com la classe "Cover" utilitza la línia:
  " image = (ImageView) findViewById(R.id.*imageLogo*); "
  que permet que des de aquest punt puguem cridar, modificar, etc.. la imatge logo a través de la variable "*image*".

  Altres classes R utilitzades en l'aplicació a part de "*R.layout*" i "*R.id*" són "*R.drawable*", "*R.color*", "*R.style*" i "*R.string*".

"*R.drawable*" l'utilitzarem per a recursos gràfics. "*R.color*" s'utilitza per a definir colors. "*R.style*" per a definir estils predeterminats i "*R.string*" proporciona cadenes de caràcters segons necessitem.
Podem trobar més informació sobre els recursos a Android Developers.

Il·lustració 20- Classes R

- Un altre aspecte a destacar és el "*Android Manifest*", aquest és un arxiu que es troba a la carpeta arrel del codi.
  Cada aplicació Android l'ha de tenir i serveix per donar informació al SO Android abans de començar a executar la aplicació. Els punts més importants són:
  - Donar un nom únic al paquet de Java.

```
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
package="com.myflightrecorderv1"
```

  - Descriu els components de l'aplicació, "*activities*", "*broadcast receivers*", etc..

```
<activity
        android:name="com.myflightrecorderv1.Cover"
        android:label="@string/app_name"
        android:screenOrientation="portrait" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category
         android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
</activity>
```

  - Descriu quins processos hostatgen els components de l'aplicació.
  - Declara els permisos que l'aplicació necessita per a l'execució.

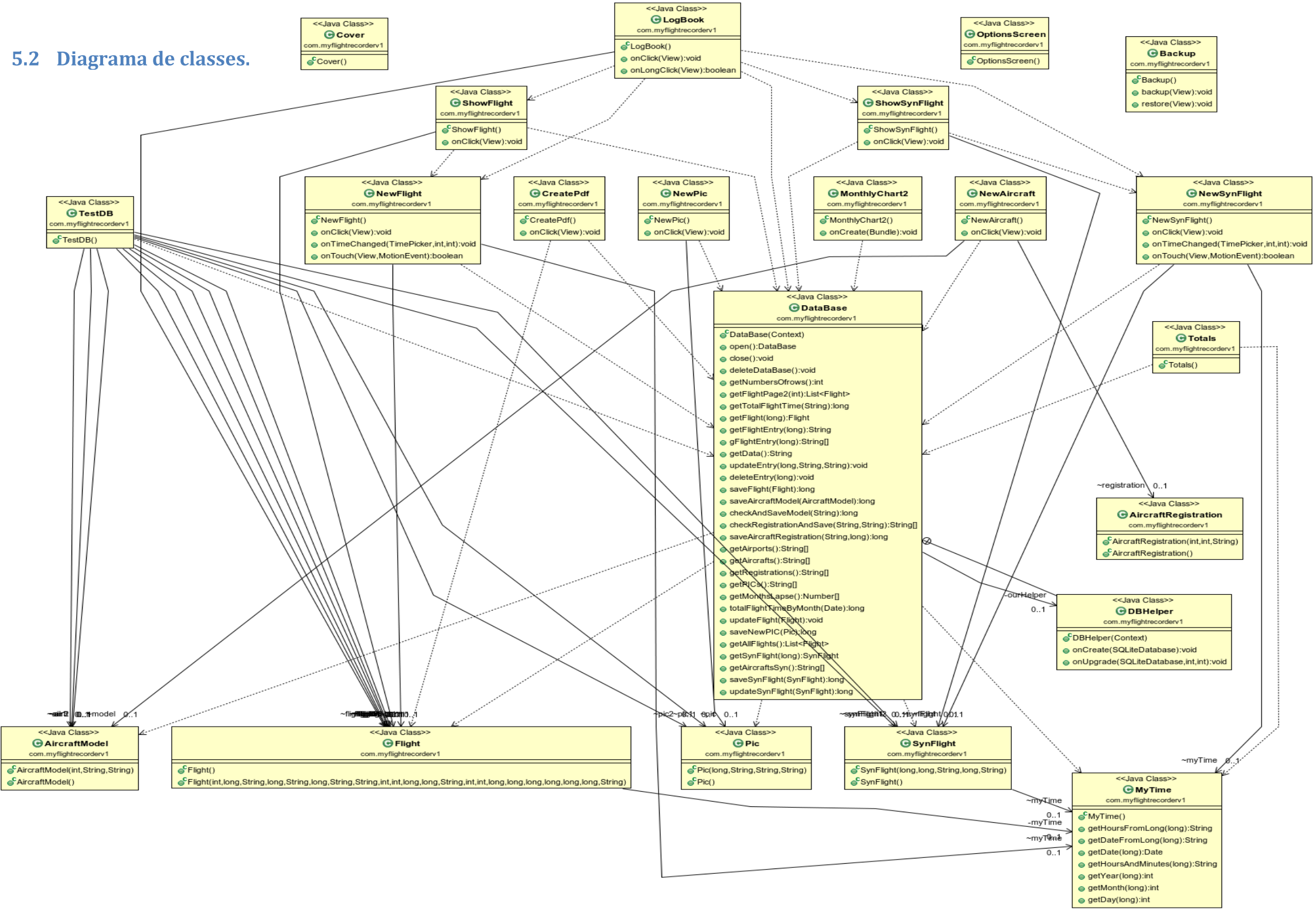
```
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"
/>
```

  - Declara el nivell mínim i màxim de l'API que l'aplicació necessita.

```
<uses-sdk
        android:minSdkVersion="10"
        android:targetSdkVersion="17" />
```

  - Llista també les llibreries que utilitza.

5.2 Diagrama de classes.



5.3 Descripció de les classes.

Vist com és la programació en Android anem a veure quina és la utilitat de cada una de les classes dins el projecte.

Les primeres classes a comentar són les que pertanyen a "objectes" tal i com els entenem dins la programació orientada a objectes.

- **Flight.java:** Aquesta classe fa referència a un registre de vol i conté tots els paràmetres d'aquest, data, hora d'enlairament, aeroport d'enlairament, etc... Amb tots els seus "*getters*" i "*setters*". En la base de dades correspon a la taula "Flights".
- **SynFlight.**java: Aquesta classe fa referència a un registre d'un vol en un simulador, i conté tots els paràmetres i "*getters*" i "*setters*". Correspon a la taula "SyntheticTraining" de la base de dades.
- **Pic.java:** Aquesta classe correspon a les dades de un pilot, el nick, el nom, cognom. I conté tots els seus "*getters*" i "*setters*". Correspon a la taula "PIC" de la base de dades.
- **AircraftModel.java:** Aquesta classe correspon a un model d'aeronau, conté tots els seus paràmetres i "*getters*" i "*setters*" corresponents. Correspon a la taula "AircraftModel" de la base de dades.
- **AircraftRegistration.java:** Aquesta classe correspon al registre d'aeronau, conté els paràmetres i "*getters*" i "*setters*" corresponents. Correspon a la taula "AircraftRegistration" de la base de dades.

Les següents classes són classes útils, no contenen informació per sí mateixes ni tampoc gestionen una pantalla. Però ofereixen mètodes útils a la resta de les classes.

- **DataBase.java**: Aquesta classe conté la classe privada DBHelper. Aquestes dues classes juntes s'encarreguen d'obrir i tancar l'accés a la base de dades i ofereixen els mètodes de consulta i modificació. És la classe pont entre l'aplicació i la base de dades Android (SQLite).
- **MyTime.java**: És una classe creada per oferir mètodes de tractament de les dades de temps. Com que la gestió de les dades temporals és una part important de l'aplicació, s'ha implementat aquesta classe per poder reutilitzar els mètodes i no haver de fer que cada classe repetís els mateixos mètodes.

I per últim comentar les classes que porten alguna pantalla associada.

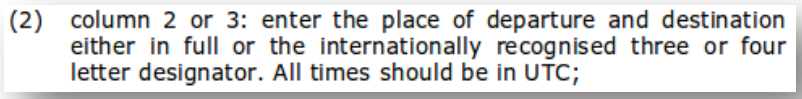
- **Cover.java (cover.xml):** Aquesta és la classe que obra la aplicació, la seva funció no es altre que mostrar una portada de l'aplicació.
- **LogBook.java (logbook.xml):** Aquesta és la classe de la pantalla principal de l'aplicació, la pantalla on podem veure els registres dels vols. S'ocupa de gestionar les pàgines dels registres que es mostren i obtenir-los de la base de dades a través de la classe DataBase.java.
- **ShowFlight.java (showflight.xml):** Aquesta pantalla s'ocupa de mostrar totes les dades d'un vol concret.
- **ShowSynFlight.java (showsynflight.xml):** Aquesta pantalla s'ocupa de mostrar el registre d'una sessió de simulador.
- **NewFlight.java (newflight.xml):** Aquesta pantalla ens permet introduir les dades d'un vol nou i guardar-les a la base de dades. Però si la classe rep la "RowID" d'un registre de vol, aleshores ens mostrarà la pantalla amb les dades del registre perquè les puguem editar.
- **NewSynFlight.java(newsynflight.xml):** Aquesta pantalla ens permet introduir les dades d'una sessió de simulador nova. Però si la classe rep la "RowID" d'un registre de simulador, aleshores ens mostrarà la pantalla amb les dades del registre perquè les puguem editar.
- **NewPic.java(newpic.xml):** Aquesta pantalla permet introduir les dades d'un nou comandant.
- **NewAiracraft.java(newaircraft.xml):** Aquesta pantalla te la utilitat d'introduir una nova aeronau a la base de dades i també la seva matricula (registration).
- **OptionsScreen.java**: Aquesta classe no es pròpiament una pantalla sinó que proporciona el menú d'opcions dins l'aplicació.
- **Backup.java(backup.xml):** En aquesta pantalla podem fer una còpia de seguretat de la base de dades, que es guardarà a la targeta de memòria externa dins la carpeta "MyFlightRecorder-Backups". També permet recuperar la base de dades.
- **MonthlyChart2.java(monthlychart2.xml):** Aquesta pantalla es formada per la gràfica de vols dividits per mesos.
- **CreatePdf.java(createpdf.xml):** Aquesta pantalla crea un arxiu pdf amb el contingut de la base de dades i el guarda dins la carpeta "Downloads".
- **TestDB.java(testdb.xml):** Aquesta és una classe per provar l'aplicació, permet eliminar la base de dades, introduir-hi dades automàticament i veure el numero de registres. Aquesta classe ha de desaparèixer en una versió definitiva o be comentar un parell de línies de codi a la classe "OptionsScreen.java" perquè l'usuari final no hi tingui accés.

5.4 Tractament de les dades temporals.

He volgut dedicar un apartat al tractament de les dades de temps perquè és un punt important de l'aplicació. Per veure'n el tractament veurem com s'introdueixen, es transformen, es guarden i es recuperen. Però abans anem a veure les necessitats de l'aplicació.

Com s'ha comentat en la motivació, al fer la comparativa d'aplicacions disponibles moltes de elles només tenen la possibilitat d'utilitzar les dates en format americà. Per això s'ha decidit que l'aplicació utilitzi el format dia, mes i any per a les dates. I així proporcionar una alternativa útil a Europa.

Pel que fa a les hores i el canvi horari, la decisió ha sigut fàcil, ja que en l'àmbit aeronàutic s'utilitza sempre les hores UTC (Universal Time Coordinated). Seria una


(2) column 2 or 3: enter the place of departure and destination either in full or the internationally recognised three or four letter designator. All times should be in UTC;

Il·lustració 21-Temps UTC. Font: EASA

bona opció a desenvolupar que l'aplicació pogués calcular l'hora UTC respecte l'hora local segons el fus horari establert en el sistema, però no ha sigut objectiu del treball.

Així doncs, guardarem les dates en format dd/MM/yy, les hores en UTC i finalment, com s'havia comentat en l'apartat motivacions, les hores de vol es guardaran en format hexadecimal de hores i minuts (HH:mm) i no en format decimal.

La introducció de la data es fa a través d'un "*date picker*", que mostra la data del dia. Per què és la data més probable d'introducció del vol ja que omplim el "*logbook*" normalment quan hem arribat de volar.

Aquesta data és transformada a un nombre "*long*" corresponent al temps UNIX o temps POSIX a les 00:00 del dia en qüestió en milisegons.

Per exemple: Si introduïm la data 10/06/2013 obtenim el long 1370822400000.
Corresponent al dia 10 de Juny de 2013 a les 00:00 en milisegons.

Aquest "long" és el que es guarda a la base de dades. Quan recuperem la data de la base de dades i volem obtenir el format dia, mes i any, ho fem cridant la funció de la classe MyTime.java "getDateFromLong".

```
public String getDateFromLong(long l) {
        Date data = new Date(l);
        DateFormat df = new SimpleDateFormat("dd-MM-yy");
        return df.format(data);
                                }
```

En el cas de les hores d'enlairament i aterratge, utilitzem "*time pickers*" per a la seva introducció. I el que fem és guardar l'hora com si fos l'hora del dia zero (1 de Gener de 1970).

Il·lustració 22-Time Picker

Per exemple: Un enlairament a les 00:00 correspondrà al long 0. I un aterratge a les 00:30 correspondrà al long 1800000 milisegons.

Per a calcular el temps de vol només haurem de restar els milisegons d'enlairament amb els d'aterratge, per tant 1800000 – 0 = 1800000 milisegons de vol, que dividit per mil ens dona 1800 segons i dividit per 60 tindrem 30minuts de vol.

Només haurem de vigilar quan l'hora d'aterratge no correspongui al dia següent, aleshores tindríem un nombre menor de milisegons de l'hora d'aterratge que d'enlairament. I aquest fet és el que s'utilitza per calcular el temps de vol. En aquest cas haurem de sumar els milisegons de l'hora d'enlairament fins a les 00:00 més els milisegons desde les 00:00 fins l'hora d'aterratge.

L'hora d'enlairament i aterratge es guarden a la base de dades com a un nombre "*long*" corresponent als milisegons. Al recuperar el nombre de milisegons utilitzem la funció de la classe MyTime.java "getHoursFromLong" i així obtenim la hora en format text de hores i minuts.

```
public String getHoursFromLong(long l) {
        Date hours = new Date(l);
        DateFormat df = new SimpleDateFormat("HH:mm");
        return df.format(hours);
                                }
```

I per últim tenim les hores de vol. Les quals introdueix l'usuari amb un "*Time Picker*". El truc utilitzat és similar al de les hores d'enlairament i aterratge. Guardem les hores de vol com s'hi fos el temps passat des de l' 1 de Gener de 1970.

Per exemple: 1:00 de vol correspon a 3600000 milisegons que seria l'1 de Gener de 1970 a les 01:00 GMT.

Després guardem el nombre de milisegons de vol a la base de dades com a nombre "*long*".

Això ens permet sumar fàcilment el temps de vol ja que només hem de sumar el nombre de milisegons volats directament.

Per recuperar el temps de vol de la base de dades hem de recuperar el nombre guardat i hem d'utilitzar la funció de la classe MyTime.java "getHoursAndMinuts" que ens retornà la quantitat de hores i minuts d'un nombre "long" en format hh:mm.

```
public String getHoursAndMinutes(long l) {
                String hour, minute;
                Long hours = (long) 0;
                Long minutes = (long) 0;
                // Get seconds from milliseconds
                l = l / 1000;
                // Get minutes from seconds
                l = l / 60;
                // Get hours
                hours = l / 60;
                // Get minutes
                minutes = l % 60;
                // format to a String HH:mm
                hour = hours.toString();
                minute = minutes.toString();
                if (hour.length() == 1) {
                        hour = "0" + hour;
                }
                if (minute.length() == 1) {
                        minute = "0" + minute;
                }

                return hour + ":" + minute;

                                        }
```

6 Conclusions.

L'objectiu del treball final de carrera és posar en pràctica els coneixements adquirits durant la carrera per a la realització d'un projecte en concret. Aquest objectiu ha sigut assolit ja que s'han utilitzat coneixements de programació orientada a objectes en Java, base de dades amb SQLite, tècniques de desenvolupament de programari, enginyeria de programari, etc.. I a més a més el treball a aportat coneixements sobre la metodologia de Disseny Centrat en l'Usuari.

Al començament del projecte ens proposàvem realitzar una aplicació per a dispositius mòbils Android. Aquest objectiu ha sigut assolit. Encara que degut al poc temps d'implementació han quedat algunes funcions sense implementar. Però gràcies a la decisió d'implementar aquelles funcions més importants per a la aplicació s'ha aconseguit un producte funcional.

Quedarien alguns aspectes a millorar en l'aplicació abans de ser publicada al públic general, com ja s'han anat comentant durant la memòria, a més d'algunes funcions que no han sigut mai objectiu del projecte, com la creació d'un "*widget*" per a l'escriptori, que no es necessari però segur que els usuaris agrairien.

Però, tot i així, aquest treball ha permès a l'autor crear una aplicació des de l'anàlisi inicial de requisits fins a la implementació final. Per tant puc afirmar que l'objectiu del treball ha sigut assolit.

S'han practicat els coneixements adquirits durant la carrera, s'ha dissenyat una aplicació segons el disseny centrat a l'usuari i s'ha implementat en Android, tot això no hagués estat possible fa uns anys quan vaig matricular-me a Enginyeria Tècnica Informàtica de Sistemes i els meus coneixements d'informàtica no eren mes grans que la instal·lació de "Microsoft Windows" en un ordenador i una mica de programació seqüencial. Per tant, només hem queda agrair a tota la comunitat UOC la possibilitat que m'ha brindat per ampliar els meus coneixements i competències professionals.

7 Glossari de termes.

Dual: Vol amb doble autoritat.

EASA (European Aviation Safety Agency): Agencia europea de seguretat aèria.

GMT (Greenwich Mean Time): Mesura del temps basada en el temps solar mitjà al meridià de Greenwich, a efectes pràctics equival a l'hora UTC.

IFR (Instrumental Flights Rules): Vol que es realitza sota regles de vol per instruments.

LogBook: Diari de vol, llibre de registres de vols realitzats que serveix per portar un control dels vols realitzats i un sumatori total i parcial d'aquests.

ME (Multi Engine): Múltiples motors.

OACI o ICAO: Organització d'Aviació Civil Internacional.

PIC (Pilot In Command): pilot comandant de l'aeronau.

PICUS(Pilot In Command Under Supervision): pilot en funció de comandant de l'aeronau sota supervisió.

Registration: Matricula de l'aeronau.

SE (Single Engine): Un sol motor.

Synthetic Training Device Sesion: Sessió d'entrenament en un simulador de vol.

Synthetic Training Device: Simulador de vol per a entrenament.

UTC (Universal Time Coordianted): Temps coordinat universal, és la zona horària de referència respecte de la qual es calculen totes les hores corresponents a les altres zones horàries del món.

8 Bibliografia i recursos.

Programming Android, 2nd Edition. O'Reilly Media.

The Definitive Guideto SQLite Second Edition. Grant Allen and Mike Owens.

Disseny Android:

- DevelopersGuide_10thEdition

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

- http://androideity.com/

Programació Android:

- http://thenewboston.org/list.php?cat=6

- http://www.mybringback.com/

- http://androidcookbook.com

- http://developer.android.com

- https://www.youtube.com/playlist?list=PLE08A97D36D5A255F&feature=plcp

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

- http://stackoverflow.com/

Llibreries externes:

- http://androidplot.com/

- http://itextpdf.com/

SQLite:

- http://www.sqlite.org

- http://www.vogella.com/articles/AndroidSQLite/article.html

- http://www.codinghorror.com/blog/2007/10/a-visual-explanation-of-sql-joins.html

```
finish();
```