



Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMÀTICA

**Facultat de Matemàtiques
Universitat de Barcelona**

**APLICACIÓ ANDROID PER A LA GESTIÓ DE
CURSES DE F1**

Ernest Pastor i Díaz

Director: Lluís Garrido
Realitzat a: Departament de Matemàtica
Aplicada i Anàlisi. UB

Barcelona, 20 de Juny de 2013

Index

0. Resum - Resumen - Summary	pàg. 5-7
1. Introducció i motivació	pàg. 8
2. Objectius i Metodologia	pàg. 9
2.1 Pla de treball	pàg. 9
2.2 Diagrama de Gantt	pàg. 11
2.3 Funcionalitats. Requisits.	pàg. 12
2.4 Organització de la Memòria	pàg. 13
3. Aplicació F1LT v0.8	pàg. 14
3.1 Casos d'Ús	pàg. 15
3.2 Login	pàg. 16
3.3 Key Frame	pàg. 19
3.4 Protocol de recepció de Dades	pàg. 24
3.5 Desencriptar Dades	pàg. 26
3.6 Tipus de Paquets	pàg. 27
3.6.1 Paquets de Sistema	pàg. 29
3.6.2 Paquets de Cotxe	pàg. 32
3.7 LTData	pàg. 33
3.8 Gestió de Dades. On guardem la informació generada.	pàg. 34
3.8.1 EventData	pàg. 34
3.8.2 DriverData	pàg. 35
3.9 Season.xml	pàg. 38
3.10 Activities de l'aplicació	pàg. 39

4. Desenvolupament	pàg. 42
4.1 Casos d'Ús	pàg. 43
4.2 Guardar les sessions	pàg. 44
4.2.1 Ruta on Guardem Dades al Login	pàg. 46
4.2.2 Key Frame i Data	pàg. 46
4.2.3 Guardar Dades	pàg. 46
4.2.4 Sincronitzar les Captura de Dades	pàg. 46
4.3 Reproduir les sessions	pàg. 48
4.3.1 Funció Diferit. Descomprimir dades, carregar ...	pàg. 50
4.3.2 Funció Diferit. Llegir arxius i Sincronització	pàg. 51
4.3.3 Funció Diferit. Bundle – Handler.	pàg. 52
4.3.4 Funció Diferit. Fi de paquets per llegir.	pàg. 52
4.4 Gràfics del Temps	pàg. 53
4.4.1 Creació d'Activities	pàg. 55
4.4.2 Composició de les Gràfiques	pàg. 55
4.4.3 Particularitats de les Gràfiques. Aspectes addicionals.	pàg. 56
4.5 Comprimir - Descomprimir	pàg. 58
4.5.1 Comprimir	pàg. 59
4.5.2 Descomprimir	pàg. 59
4.6 Netejar les dades temporals de l'aplicació	pàg. 61
5. Resultats obtinguts	pàg. 62
5.1 Grans Premis que s'han provat	pàg. 62
5.2 Anàlisi : Captura WI-FI / Paquets de Dades	pàg. 68
6. Manual d'Usuari	pàg. 69
7. Conclusions	pàg. 76
7.1 Dificultats	pàg. 77
7.2 Discussió crítica	pàg. 78
7.3 Línies Futures	pàg. 78
8. Bibliografia	pàg. 79
ANNEX - Vocabulari específic	pàg. 80

0. Resum

La motivació a l'hora de fer aquest Projecte Final de Grau ve del repte que suposa poder desenvolupar individualment un treball de gran dimensió, emprant els coneixements que hem adquirit durant els estudis universitaris . La tria del temari ha sigut degut a l'afició per la Fórmula 1.

L'objectiu d'aquest projecte és el desenvolupament d'una aplicació per dispositius mòbils per a poder seguir les curses de F1 en directe i en diferit. Per altra banda, també volem poder veure informació generada per les curses en forma de gràfics.

Per assolir els objectius hem emprat la tecnologia Android. S'ha partit d'una aplicació ja existent que permet seguir les sessions de la F1 en directe, i serà ampliada i millorada per cobrir les funcionalitats demanades.

Tal com es mostrarà al capítol de resultats, hem assolit els objectius que ens havíem plantejat ja que hem pogut gravar i reproduir les sessions de la manera que ens havíem proposat. També hem aconseguit visualitzar gràfics sobre dades generades en les curses.

0. Resumen

La motivación a la hora de hacer este Proyecto Final de Grado viene del reto que supone poder desarrollar individualmente un trabajo de gran dimensión, utilizando los conocimientos que hemos adquirido durante los estudios universitarios. La elección del temario ha sido debido a la afición por la Fórmula 1.

El objetivo de este proyecto es el desarrollo de una aplicación para dispositivos móviles para poder seguir las carreras de F1 en directo y en diferido. Por otra parte, también queremos poder ver información generada por las carreras en forma de gráficos.

Para alcanzar los objetivos hemos utilizado la tecnología Android. Se ha partido de una aplicación ya existente que permite seguir las sesiones de la F1 en directo, y será ampliada y mejorada para cubrir las funcionalidades solicitadas.

Como se mostrará en el capítulo de resultados, hemos alcanzado los objetivos que nos habíamos planteado ya que hemos podido grabar y reproducir las sesiones de la forma que nos habíamos propuesto. También hemos conseguido visualizar gráficos sobre datos generados en las carreras.

0. Summary

The reason to carry out this Final Degree Project is to meet the challenge it represents to be individually able to develop a big scope work. This has been done by using the knowledge acquired in our university studies. The reason to have chosen this subject is my being a Formule 1 fan.

The aim of this project is to develop an application for mobile devices that enables to follow F1 live sessions as well as recorded ones. On the other hand, it is also interesting to see the during the races generated data by means of graphic charts.

Technology used is Android, taking as a basis an already existing application that gives the possibility of following the live F1 sessions. This application has been further developed to cover other needed functionalities.

As will be shown in the results chapter, the aimed objectives have been reached, as we could record and play the sessions as it had been foreseen. The application also allows to have graphic charts on display as per data generated during the races.

1. Introducció i motivació

Arribem al Projecte Final de Grau, final dels Estudis Universitaris, i en aquest punt és on hem de demostrar la capacitat d'aprenentatge que hem adquirit durant tots aquests anys, així com els coneixements en les diferents àrees que es tracten durant la carrera.

Per altra banda, hem d'escollir un tema relacionat que haguem fet durant la carrera i desenvolupar-ho, a fi d'aprofundir més en base als coneixements adquirits i realitzar un treball autònom.

En la part personal, la F1 és una gran passió que he tingut des de petit i he seguit aquest esport tant des de la Televisió com anant a diferents circuits, a més de seguir l'actualitat que es genera entre les curses. És per això que tinc la motivació de fer un projecte relacionat amb aquest esport, ja que al veure les curses m'agrada anar seguint el *Live Timing* (veure Annex) que es genera a totes aquestes curses i sempre hi he tingut molt d'interès en treure-hi conclusions d'aquestes dades.

També tinc un gran interès en l'àrea de les aplicacions Android per a dispositius mòbils. A dia d'avui, es viu un gran auge d'aquestes tecnologies i el seu llenguatge (Java) ha sigut força utilitzat en el transcurs de la carrera cursada a la UB. També s'afegeix a això, que al cursar la carrera d'ETIS (Enginyeria Tècnica en Informàtica de Sistemes) en gran part, no he tocat pràcticament gens les aplicacions Android, i aquest ha sigut un punt important a decidir-me en aquesta tecnologia de cara al Projecte Final.

Per tot això, la meva decisió va ser de iniciar aquest projecte d'una aplicació Android que gestioni les dades generades per les curses de F1, ja que uneix motivació per la tecnologia, interès en aprendre una nova àrea que no he cursat pràcticament en la carrera i l'afició per la Fórmula 1.

2. Objectius i Metodologia

Els objectius per aquest projecte és poder treballar amb totes les dades que genera el Live Timing i a partir d'aquesta informació obtinguda, poder reproduir sessions disputades amb anterioritat així com poder mostrar la informació de diferents maneres per poder treure conclusions o fer comparacions entre els diferents pilots participants.

Per tant, l'objectiu d'aquest projecte és desenvolupar una aplicació o eina Android capaç de treballar amb les dades que genera el Live Timing de la F1, amb les següents característiques:

1. L'aplicació ha de poder treballar tant amb les dades en directe (capturades directament del servidor web) com en diferit, amb les dades capturades.
2. L'aplicació ha de permetre visualitzar la informació de diverses formes (taules, gràfiques, ...) perquè l'usuari pugui analitzar amb detall el decurs d'una cursa o entrenaments.

Per arribar a poder fer tot això, m'hauré de familiaritzar en tot l'entorn Android ja que només abans ho vaig fer en una petita pràctica a Nous Usos de la Informàtica. Per tant, es pot dir que Android no l'he arribat a fer pràcticament fins a iniciar aquest Projecte Final de Grau.

Per altra banda, parteixo d'un codi que ja estava fet parcialment per la versió Android. El què ja estava fet de l'aplicació és la part de desenscriptació de dades i emmagatzematge. Per tant, també m'hauré de familiaritzar amb tot el codi existent i entendre bé com funciona, per posteriorment poder implementar diferents funcionalitats que respectin el què ja hi havia, ampliant la informació de l'aplicació.

2.1 Pla de treball

2.1.1 Pla de treball inicial

Al iniciar el projecte teníem el dubte de si iniciar el projecte de 0 fent nosaltres el parser de les dades, la desenscriptació i l'emmagatzematge, o per altra banda, sortir d'un codi inicial en què les dades venen parsejades de la font (pàgina F1.com) i en què nosaltres ens des preocupem del procés, dedicant-nos a les implementacions de noves funcionalitats i part d'interfície. En els primers mesos s'ha fet una planificació inicial per tal de decidir entre les diferents opcions que tenim.

El planning inicial dels primers dos mesos és el següent :

Setmana 1 (4-10 febrer)

Llegir sobre Android, provar-ho tant al PC (amb el desenvolupador) com al mòbil. Només cal fer una petita aplicació per assegurar-se que tot funciona bé i que el tema està controlat.

Setmana 2 (11-17 febrer)

Analitzar el parser. Què fa exactament ? Rep dades en forma de streaming ? Accedeix al web de F1 cada X temps ? Com es criden dels funcions del parser per obtenir la informació que volem ?

Setmana 3 (18-24 febrer)

Analitzar la connexió d'Android amb el parser. Hem de trobar la forma de poder cridar al parser i així no haver de re-escriure el parser. Fes una petita aplicació en Android que cridi al parser.

Setmana 4 (25 febrer-3 març)

Modificar parser perquè guardi en un log les dades parsejades. Aquestes es podran aprofitar pel simulador.

Cal fer un simulador, que serà el mateix parser. A l'hora d'inicialitzar l'aplicació l'usuari podrà escollir si inicialitzar amb el simulador o amb el mode real.

Setmana 5 (4-10 març)

Fer una aplicació Android molt senzilla que mostri per pantalla les dades que arriben del parser.

Durant aquestes setmanes, es va analitzar la complexitat començar de zero l'aplicació, descartant de bon inici aquesta opció ja que no teníem el temps necessari. També és cert que existeix un parser fet que facilitaria la feina, i arribats a aquest punt s'havia de decidir entre un parser en llenguatge C i aplicar-ho a Android o partir d'una versió inicial de l'aplicació [F1LT](#) en la versió 0.8 per a Android.

Finalment, la decisió va ser agafar la versió 0.8 d'Android de F1LT, ja que l'objectiu en aquest projecte l'enfoco més al desenvolupament dins l'entorn Android, a la gestió de les dades obtingudes de l'aplicació més que l'obtenció i descriptació de les dades originals que rebí l'aplicació, i al desenvolupament d'una aplicació en l'entorn.

He agafat aquest codi per diferents raons, la principal és perquè evita parsejar dades del punt origen fins a l'aplicació. També perquè ja està en el llenguatge Android (Java) el qual volia treballar, per tant no hem d'aplicar operacions entre diferents llenguatges dins la mateixa aplicació, el qual li evitarà dificultat afegida al projecte. Per últim, també és una raó important el partir d'aquesta aplicació ja que és una versió inicial no definitiva, i crec poder aportar funcionalitats molt útils per als usuaris.

La meua contribució que vull aportar al projecte és donar noves funcionalitats addicionals a les que ja ofereix l'aplicació. En la fase inicial del codi en què començo, aquesta aplicació només està operativa en temps real mentre es disputa cada una de les sessions.

Per tant, m'agradaria poder reproduir les dades en qualsevol moment, per tant serà essencial guardar les dades mentre es disputen les diferents sessions per a posteriorment ser reproduïdes per els usuaris.

2.2 Diagrama de Gantt

Un cop escollida la direcció del projecte partint de l'aplicació F1LT, fem el diagrama de Gantt per organitzar les diverses tasques previstes que es duren a terme en el transcurs del Projecte.

Aquest diagrama ens ajudarà a planificar millor el temps i a la seva distribució d'hores a les diferents tasques i noves funcionalitats.

Activitat	Inici	Final	Dies
Llegir sobre Android	04/02/13	10/02/13	6
Entendre el Parser en C	11/02/13	17/02/13	6
Relació Parser-Android	18/02/13	24/02/13	6
Fer un log de les dades	25/02/13	03/03/13	6
Entendre la App F1LT	25/02/13	11/05/13	75
Capturar les dades	16/03/13	11/05/13	56
Llegir sobre Android 2	22/04/13	05/05/13	13
Reproduir les dades	01/04/13	19/05/13	48
Guardar KeyFrame Sessió	06/05/13	12/05/13	6
Gràfics de l'Event	12/05/13	26/05/13	14
Comprimir – Descomprimir	27/05/13	02/06/13	6
Interfície Carregar Dades	27/05/13	02/06/13	6

I la representació gràfica segons el temps és la següent :

[illegible]

2.3 Funcionalitats. Requisits.

Els requisits genèrics per cada una de les noves funcionalitats que s'implementin són que no siguin incompatibles amb els que ja venen amb la versió bàsica amb que ens hem posat a treballar. Òbviament, tampoc volem que distorsionin les dades de les altres funcionalitats o que modifiquin el comportament de la aplicació a pitjor.

L'objectiu a l'hora d'introduir cada nova implementació a l'aplicació és que el rendiment global de l'aplicació no empitjori ni pateixi alentiment. És a dir, no només implementar-ho de la forma més correcta i eficient, sinó que també respecti l'entorn que existia i les millores no siguin a costa de cap altre rendiment existent.

Les funcionalitats que es vol implementar seriem les següents:

- Gravació de les dades obtingudes en temps real.

Requisits :

1. Que les dades gravades de la realitat siguin reproduïbles en més d'una ocasió.
2. Fer-les accessibles perquè el mode de lectura hi pugui accedir.
3. Que les dades gravades ocupin la mínima mida possible.
4. Les dades gravades de diferents sessions no estiguin mesclades.
5. Que les dades gravades siguin intercanviables entre diferents dispositius amb l'aplicació.

- Reproducció de les Dades de qualsevol sessió.

Requisits :

1. Poder reproduir les dades a temps real (tal qual s'han obtingut).
2. Poder reproduir els diferents tipus de sessió gravats.
3. Que no difereixin les dades gravades de les reproduïdes en posterioritat.

- Gràfics de les Dades obtingudes per l'esdeveniment.

Requisits :

1. Mostrar la representació gràfica de les dades capturades durant la Sessió.
2. Representar les dades d'una manera semblant a com es fa a F1.com
3. Que les dades representades corresponguin amb les capturades.
4. Ajustar la representació a les dimensions de forma intel·ligible.

2.4 Organització de la Memòria

En els propers punts, s'explicarà el funcionament intern de l'aplicació en dos moments ben diferents de l'execució d'aquest projecte.

Al **Capítol 3**, el següent, s'explicarà el funcionament de l'aplicació primitiva que s'ha agafat i s'ha partit de base en aquest projecte. En aquest apartat s'explica tant el flux d'execució del programa, com els diferents casos d'ús que es poden executar en l'aplicació, així com d'interacció de l'aplicació amb la web F1.com, d'on obtindrà les dades en temps real.

Al **Capítol 4**, s'explicarà com ha quedat l'aplicació després d'aplicar-hi les noves funcionalitats, i es compararà amb el punt anterior 3 per veure-hi les modificacions aplicades al codi. Òbviament, també s'explicarà detalladament les modificacions de les noves funcionalitats, així com els nous casos d'ús.

Al **Capítol 5**, s'avaluaren els resultats obtinguts en els diferents Grans Premis capturats un cop afegides les noves funcionalitats al codi, així com mostrarem com les veu l'usuari final de l'aplicació.

Al **Capítol 6** hi ha un Manual d'Usuari en què s'explica directament per a l'usuari la navegació de l'usuari per l'aplicació així com les diferents modalitats de rebre la informació que pot escollir. S'informarà a l'usuari com guardar les curses i com pot reproduir-les posteriorment.

Per últim, al **Capítol 7** s'exposarà les conclusions obtingudes al llarg del Projecte en els diferents apartats treballats.

També s'exposarà les dificultats que ens hem enfrontat en les diferents fases, així com les línies futures que es treballaran en l'aplicació i farem una discussió crítica del treball realitzat.

Per últim, al llarg d'aquest document sortiran paraules específiques que mereixen ser definides en una secció apart. Aquestes paraules aniran acompanyades d'un subratllat i cursiva (*ANNEX*) i estan explicades al final del document, a la secció Annex. Per últim, també pot ser útil llegir el punt 6 (Manual d'Usuari) per tal d'entendre molts aspectes que en explicacions detallades de funcionalitats no puguin quedar del tot clars.

3. Aplicació F1LT v0.8

En aquest Capítol explicarem, el més acuradament possible, el funcionament de l'aplicació inicial F1LT v0.8 del qual es va partir per introduir-hi les funcionalitats addicionals

Posteriorment, en el **Capítol 4** detallarem les noves funcionalitats introduïdes així com compararem les diferències amb l'aplicació inicial, per a poder entendre més concretament les modificacions que s'han fet i el seu funcionament inicial i final.

L'aplicació usa la tecnologia Android, i està escrita en llenguatge Java.

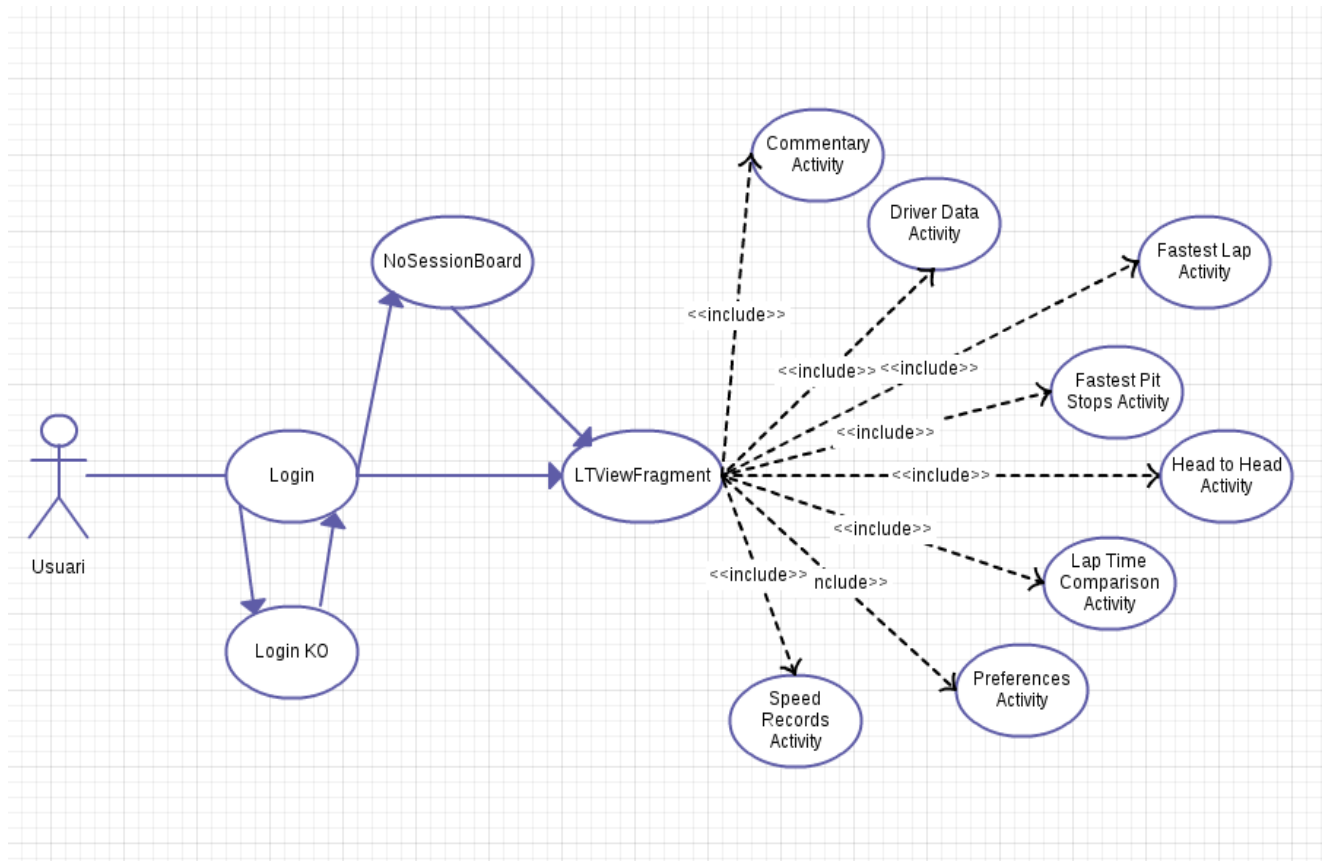
En l'aplicació de la qual partim **no hi ha documentació** que expliqui el seu funcionament i per tant hem hagut de dedicar una gran quantitat de temps a entendre el funcionament intern d'aquesta. Aquest capítol es concentra doncs en explicar els detalls de funcionament de l'aplicació F1LT 0.8.

Aquesta aplicació que comencem a treballar, només treballa en mode Online. És a dir, es reben dades en tot moment de l'última cursa o sessió disputades tal com s'ha acabat, però el streaming continu de dades només succeeix mentre les sessions s'estan disputant.

IMPORTANT : Hem optat per dividir els diagrames de classes i de seqüència per seccions o funcionalitats, ja que sinó era totalment indesxifrable.

3.1 Casos d'Ús

Aquí tenim el diagrama de Casos d'Ús per a l'Aplicació :



En els propers capítols s'explicaran amb més detall què conté cada activity, concretament en l'apartat 3.10.

3.2 Login

Començarem a explicar en primer lloc la secció de la classe *Login*, per accedir a l'aplicació i la pantalla inicial un cop hi hem accedit, que és on mostra la informació tal qual la veiem al *Live Timing* de F1.com

El flux de l'aplicació s'inicia en la classe *F1LTActivity* que és el main (principal d'execució). En aquesta classe hi han totes les funcions que una aplicació Android ha de tenir, onCreate (crea l'aplicació i tot el necessari), onStart (Inicia l'Aplicació), onDestroy (al sortir de l'aplicació), onRestart, etc.. També hi tenim les funcions principals que gestionen tot el flux principal de l'aplicació així com la crida a l'obtenció de dades o la crida a les diferents Activities (funcionalitats) implementades.

Al inicial l'aplicació, es llegeix el fitxer XML per obtenir les dades dels pilots, Grans Premis, etc. Tota la informació relativa i necessària per a la reproducció correcta dels esdeveniments. Es carrega la vista inicial a reproduir així com també es carregaran les preferències de l'usuari (mostrar el dibuix del cotxe al costat del pilot, mostrar el nom curt – abreujat - o normal, i mostrar una barra de comentaris al marge inferior de la pantalla. Aquestes preferències són carregades inicialment en valors per defecte, posteriorment l'usuari pot modificar-ho a través del menú principal.

Per altra banda, s'instancia la classe *DataStreamReader* on s'enviaran les dades obtingudes a la disputa de la sessió per ser processades.

En cas de no estar connectats a l'stream de dades provinent de la pàgina web, s'instancia la Activitat de la classe *Login* per tal fi que l'usuari introdueixi el seu e-mail registrat i contrasenya per accedir a dins de l'aplicació.

Un cop introduïdes les dades, es retornarà a la classe *F1LTActivity* els valors per a què es connecti mitjançant la funció *onActivityResult* . En aquesta funció en cas que s'hagués demanat el login i s'hagin omplert els dos camps corresponents, es crida a *onConnectedClicked* que farà una crida a la classe *HttpReader* on es gestionarà tot el procés de Login. Aquesta classe implementa accions de HTTP per l'obtenció de les dades vitals per l'accés i descriptació de dades, com son el id de la sessió i la cookie.

Aquesta cookie d'autorització és imprescindible per accedir al *Live Timing* ja que permet la interacció del programa amb l'obtenció de dades. Per obtenir-la, hem d'actuar com un navegador web, i es fa construint el mètode **POST** a la URL del *Live Timing*, amb els paràmetres del “e-mail” i “password” introduïts per l'usuari. Un cop enviat, se'ns retornarà la cookie al **Header** de resposta.

En resum :

```
-->  POST /reg/login.asp HTTP/1.1
      Host : secure.formula1.com
      ...

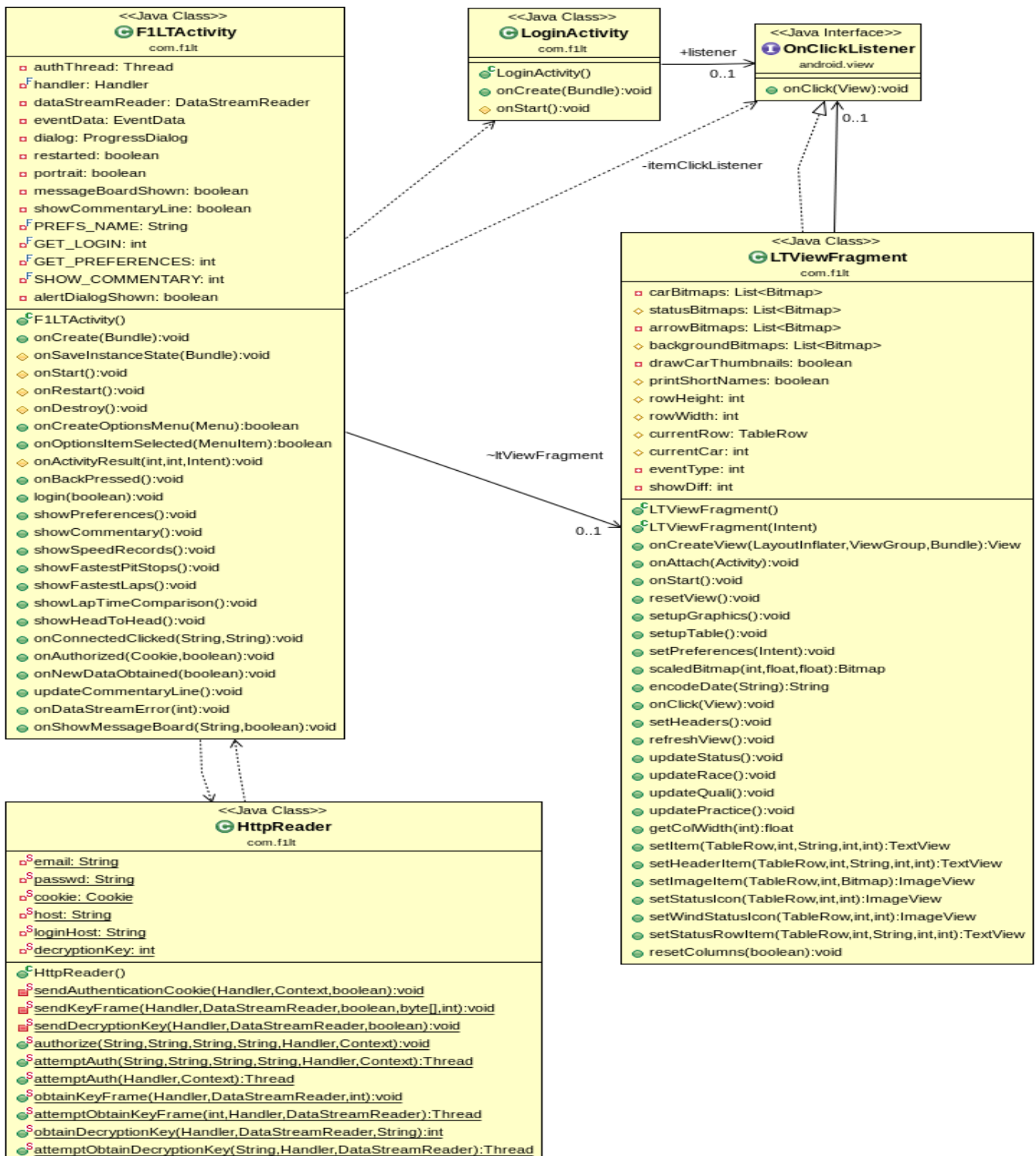
      email=$email&password=$password

<--  HTTP/1.1 302 Object Moved
      Location : http://www.formula1.com/race/livetiming/popup/0.html
      Set-Cookie : USER=$cookie;
```

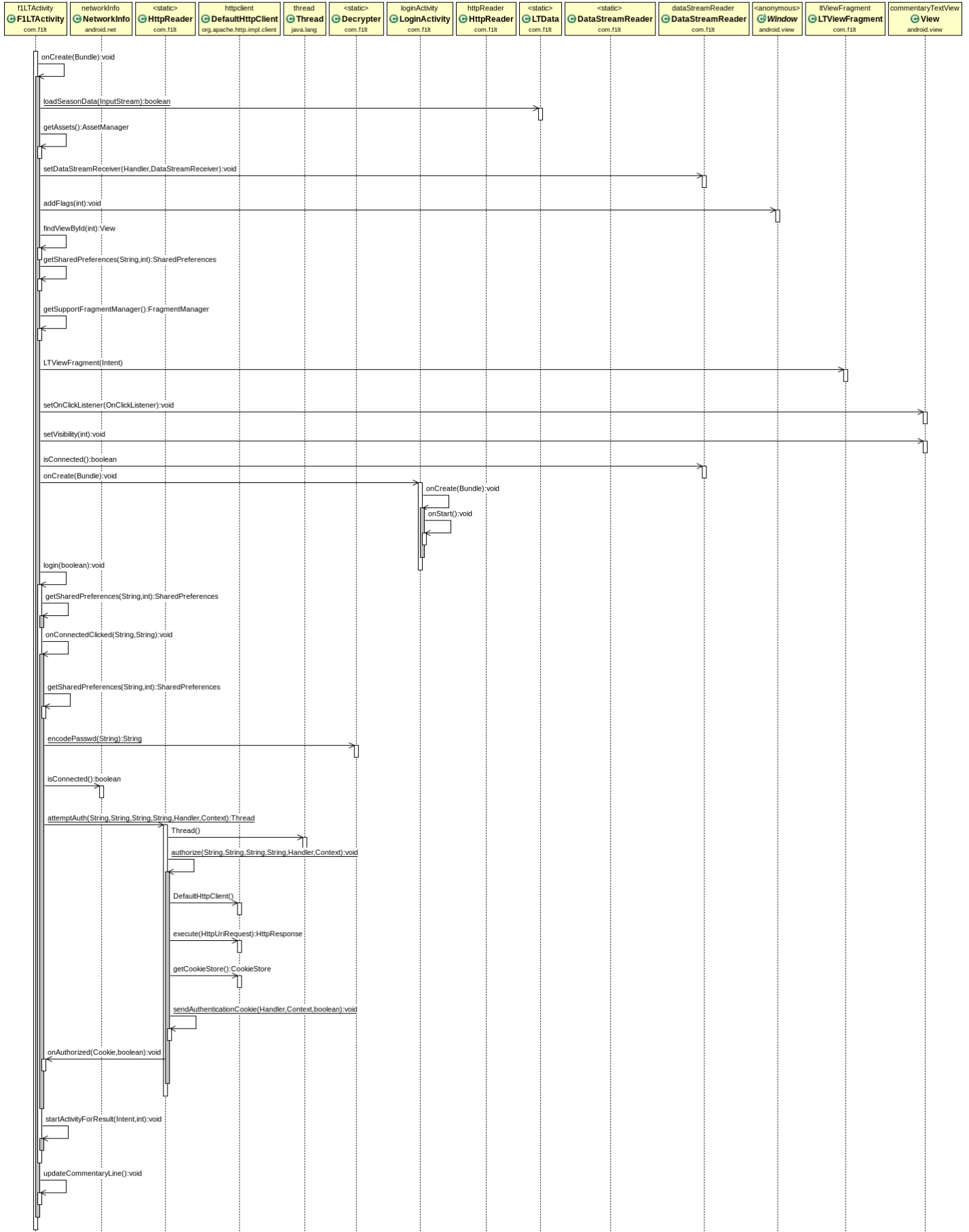
Observem que se'ns redirigeix però no importa ja que tenim la cookie i la redirecció és per dirigir-te al Applet Java del Live Timing, un cop loguinejat.

Per tant, guardem la cookie d'autorització i la resta no ens importa. I fet això, retornem l'execució de la classe *HttpReader* a *onAuthorized* retornant la cookie i el resultat en valor true o false (true si l'ha obtingut, false si ha fallat).

El diagrama de classes del procés que hem explicat és el següent :



I el diagrama de seqüència :



3.3 Key Frame

La propera acció a realitzar, és obtenir el Key Frame. **ÉS VITAL**, ja que a partir d'aquest nombre podrem descriptar totes les dades que obtinguem posteriorment. En cas que guardem les dades que es produeixen però no el Key Frame, al descriptar les dades no seran llegibles.

El procés s'inicia al `onAuthorized` un cop obtinguda la cookie d'autorització. Fet això es crida a `connectToStream` de la classe *DataStreamReader* on s'inicia a partir d'aquí la recepció de dades. Però abans de començar a rebre dades, necessitem la clau per descriptar-les.

IMPORTANT : La classe *F1LTActivity* implementa la Interfície *DataStreamReceiver*, que és una interfície per a l'obtenció de les dades.

Abans d'iniciar l'explicació, necessitem apuntar que apart de necessitar el KeyFrame, també és necessari saber que el Live Timing utilitza un Frame per anar recomptant les dades que s'envien. Va per ràfegues actualitzant-ho incrementalment (inicia amb 1, 2, etc..), i és necessari a tal fi de poder incorporar-se un usuari en el moment que vulgui i no haver de necessitar rebre totes les dades anteriors al moment que s'ha iniciat la sessió.

Així doncs, el Frame és un nombre enter que es va augmentant cada cop que l'aplicació ens ho dicta, per tant ens ho hem d'anar guardant a la variable de la classe *DataStreamReader* anomenada Frame perquè sinó perdríem la correcta descriptació.

NOTA : Durant la fase de proves, de tant en tant no arriba bé el Frame. El resultat en la descriptació és que totes les dades rebudes per el nombre de Frame determinat, no podrà ser correctament mostrat i les properes dades a ser llegides i processades correctament serà en el moment d'arribar-nos el següent Frame.

Per obtenir el Key Frame, farem la crida a la URL passant per paràmetre el Frame amb valor 0. Això ho fem perquè quan rebí un 0 amb la nostra petició, entendrà que estem reclamant el Key Frame, i en la mateixa operació posteriorment ja ens enviarà el Frame que actualment estigui, més les dades referents a la sessió.

Per tant, es cridarà a la classe *HttpReader* la funció `attemptObtainKeyFrame` , passant per paràmetre 0 del Frame. S'inicia un Thread perquè aquesta part de funcionament sigui paral·lela a l'obtenció de les dades de programa (es reben en per separat el KeyFrame per petició HTTP i les dades per el socket).

Llavors, per gestionar això farem un GET a la URL passant el id de la sessió i la cookie d'autorització, i d'aquesta manera obtindrem en el cos de la resposta el Key Frame que ens descriptarà tota la sessió. Ens ho torna en HTML la resposta, però el key és un text pla que no tindrem problemes agafar-lo.

Afegim que la longitud del Key Frame pot oscil·lar, havent vist durant les curses gravades xifres negatives de longitud 10 o 9 enters.

En resum:

```
--> GET /reg/getkey/$session.asp?auth=$cookie HTTP/1.1  
Host : secure.formula1.com
```

```
<-- HTTP/1.1 200 OK  
Content-Type : text/html  
Content-Length : 10
```

\$key

Per tant, un cop s'ha obtingut el Key Frame des de la classe *HttpReader* en la funció *obtainKeyFrame* queda cridar a la funció *sendKeyFrame* perquè s'invoqui a la classe *DataStreamReader* que gestionarà com hem de guardar el Key Frame per tal de descriptar les dades que posteriorment es rebran.

A partir d'aquí, es cridarà la funció *onKeyFrameObtained* de la classe *DataStreamReader* que gestionarà la subclasse *Decrypter*, l'encarregada de descriptar els paquets que anem rebent. Aquí també es rebran els paquets per a inicialitzar els pilots participants en la sessió junt amb el Key Frame.

Per tant, es cridarà la funció *parseBlock* en què parsejarà les dades dels 22 pilots participants (aquesta temporada en són 22, però podria variar segons els inscrits al campionat).

Aquest és el Log de la inicialització que parlem :

D/obtainDecryptionKey

```
D/CAR=1(3431): CAR TYPE=0 CAR DATA= 2 CAR LEN= 0 LONG DATA=  
D/CAR=2(3431): CAR TYPE=0 CAR DATA= 5 CAR LEN= 0 LONG DATA=  
D/CAR=3(3431): CAR TYPE=0 CAR DATA= 3 CAR LEN= 0 LONG DATA=  
D/CAR=4(3431): CAR TYPE=0 CAR DATA= 8 CAR LEN= 0 LONG DATA=  
D/CAR=5(3431): CAR TYPE=0 CAR DATA= 14 CAR LEN= 0 LONG DATA=  
D/CAR=6(3431): CAR TYPE=0 CAR DATA= 7 CAR LEN= 0 LONG DATA=  
D/CAR=7(3431): CAR TYPE=0 CAR DATA= 4 CAR LEN= 0 LONG DATA=  
D/CAR=8(3431): CAR TYPE=0 CAR DATA= 9 CAR LEN= 0 LONG DATA=  
D/CAR=9(3431): CAR TYPE=0 CAR DATA= 6 CAR LEN= 0 LONG DATA=  
D/CAR=10(3431): CAR TYPE=0 CAR DATA= 1 CAR LEN= 0 LONG DATA=  
D/CAR=11(3431): CAR TYPE=0 CAR DATA= 15 CAR LEN= 0 LONG DATA=  
D/CAR=12(3431): CAR TYPE=0 CAR DATA= 16 CAR LEN= 0 LONG DATA=  
D/CAR=13(3431): CAR TYPE=0 CAR DATA= 10 CAR LEN= 0 LONG DATA=  
D/CAR=14(3431): CAR TYPE=0 CAR DATA= 13 CAR LEN= 0 LONG DATA=  
D/CAR=15(3431): CAR TYPE=0 CAR DATA= 18 CAR LEN= 0 LONG DATA=  
D/CAR=16(3431): CAR TYPE=0 CAR DATA= 17 CAR LEN= 0 LONG DATA=  
D/CAR=17(3431): CAR TYPE=0 CAR DATA= 12 CAR LEN= 0 LONG DATA=  
D/CAR=18(3431): CAR TYPE=0 CAR DATA= 11 CAR LEN= 0 LONG DATA=  
D/CAR=19(3431): CAR TYPE=0 CAR DATA= 22 CAR LEN= 0 LONG DATA=  
D/CAR=20(3431): CAR TYPE=0 CAR DATA= 19 CAR LEN= 0 LONG DATA=  
D/CAR=21(3431): CAR TYPE=0 CAR DATA= 20 CAR LEN= 0 LONG DATA=  
D/CAR=22(3431): CAR TYPE=0 CAR DATA= 21 CAR LEN= 0 LONG DATA=
```

Observem al Log com rebem el Key Frame i, posteriorment, obtenim els 22 pilots participants. El tipus de paquet (CAR TYPE) és 0 degut que és el tipus que s'estableix com inicialitzant. Per últim, el Car Data és el dorsal.

Apuntem que no té relació la creació dels diferents pilots amb l'ordre de dorsals.

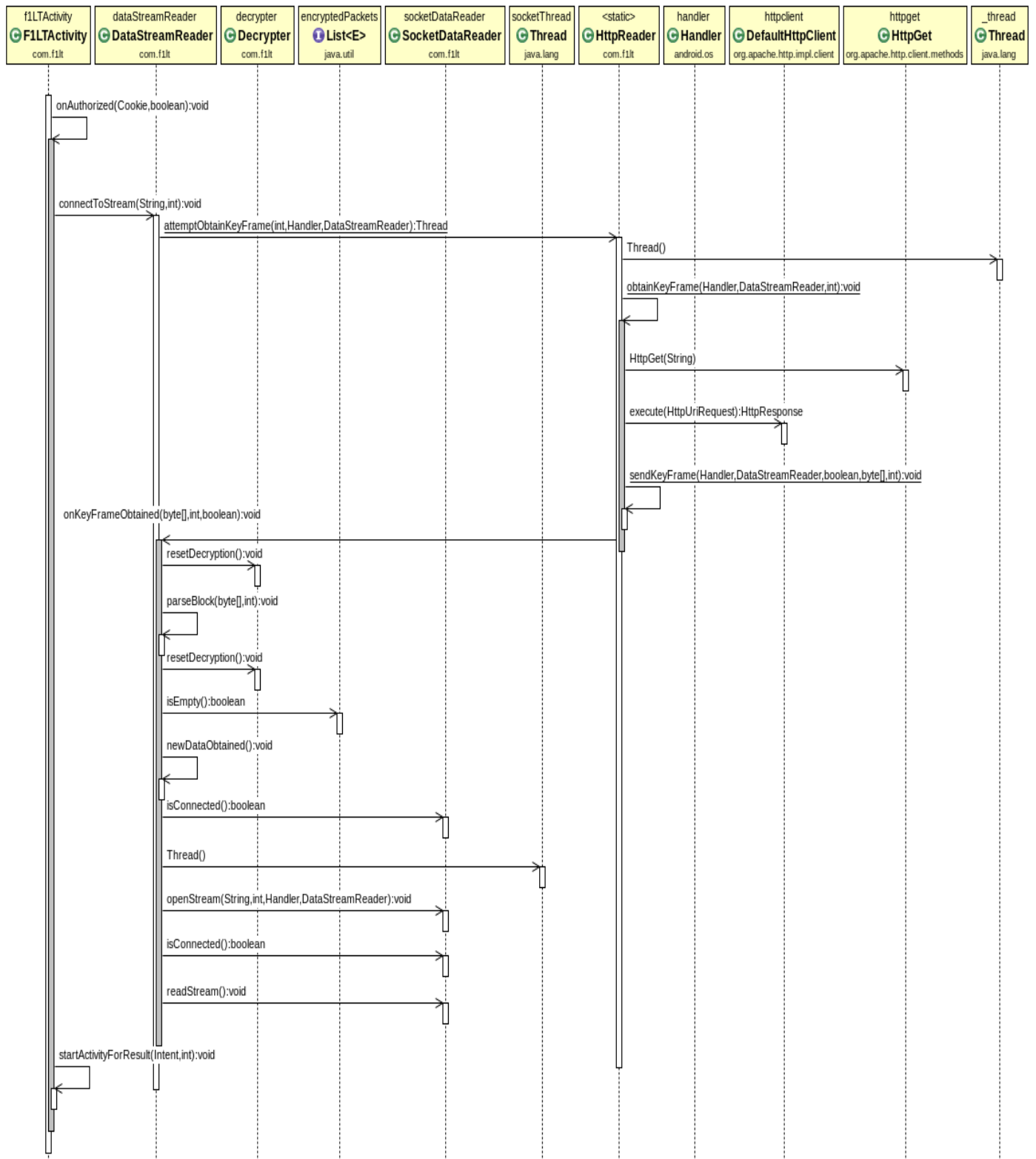
Per tant, ara només queda obrir l'Streaming de Dades entre l'aplicació i el Live Timing, això es farà a través de la classe *SocketDataReader* que habilita un socket en què es van rebent les dades generades per la sessió en disputa. Aquest socket es mantindrà obert mentre estigui connectat.

Per tant, ja tenim obert el socket de dades i ja estem preparats per rebre totes les dades que vindran corresponents a pilots o a l'esdeveniment. En la funció de `readStream` de *SocketDataReader* anirem llegint les dades que anem rebent.

El diagrama de classes del procés que hem explicat és el següent :



I el diagrama de seqüència :



Un cop obtinguts els elements indispensables, ja ens mostra la pantalla principal del Live Timing a la aplicació. Farem una comparació del què veiem per la web i el què veiem un cop hem accedit dins la zona registrada.

A la web i a l'aplicació ho veiem així :

Formula 1™ - The Official F1™ Website - Live Timing - Mozilla Firefox

www.formula1.com/live_timing/live_timing_popup.html

FORMULA 1 GRAND PRIX DU CANADA 2013 (Race)

P	Name	Gap	Interval	Sector 1	Sector 2	Sector 3	Pit
1	S. VETTEL	LAP	70	1:17.914	21.1	25.0	31.6
2	F. ALONSO	14.4	14.4	1:17.017	21.2	24.8	30.9
3	L. HAMILTON	15.9	1.5	1:16.797	21.6	24.5	30.5
4	M. WEBBER	25.7	9.7	1:17.418	21.7	24.7	30.9
5	N. ROSBERG	69.7	43.9	1:16.534	21.4	24.4	30.6
6	J. VERGNE	1L	1L	1:19.917	23.2	25.2	31.4
7	P. DI RESTA	1L	9.3	1:20.693	23.6	25.4	31.5
8	F. MASSA	1L	1.9	1:16.939	21.6	24.7	30.5
9	K. RAIKKONEN	1L	2.6	1:19.089	22.2	25.2	31.6
10	A. SUTIL	1L	2.4	1:17.717	22.0	24.9	30.7
11	S. PEREZ	1L	1.2	1:17.369	21.7	24.8	30.8
12	J. BUTTON	1L	2.6	1:17.587	21.7	24.7	31.0
13	R. GROSJEAN	1L	12.5	1:17.610	21.8	24.8	30.8
14	V. BOTTAS	1L	21.9	1:21.362	23.9	25.6	31.8
15	D. RICCIARDO	2L	1L	1:18.257	22.0	25.1	31.0
16	P. MALDONADO	2L	35.4	1:18.105	22.0	25.0	31.0
17	J. BIANCHI	2L	21.7	1:18.873	22.0	25.2	31.5
18	C. PIC	3L	1L	1:20.913	22.3	26.0	32.5
19	M. CHILTON	3L	21.1	1:19.566	22.3	25.5	31.7
20	E. GUTIERREZ	7L	4L	STOP	L21		
21	N. HULKENBERG	25L	18L	STOP			
21	G. VAN DER GAR	27L	2L	32.6	36.2	STOP	

BEST LAP: 2 M. WEBBER 1:16.182 **ON LAP:** 69

Track Status: ● © Formula One World Championship Ltd LG

Reduce Window

He collided with the wall but is safely out of the car.

Lap 69/70 - Vettel set for his first Canadian win - and a very dominant one it will be.

Lap 69/70 - Massa finally passes Raikkonen for P8 into the final chicane.

Chequered flag - Vettel wins in some style in Canada.

Alonso comes home second, almost 15s down, with Hamilton completing the podium.

Webber just misses out on the podium in fourth, ahead of Rosberg. Vergne finishes in a F1 career best of sixth.

One-stopping Di Resta takes seventh ahead of Massa and Raikkonen, who equals Michael Schumacher's record of 24 consecutive points finishes.

And Sutil takes the final point for Force India.

That concludes our coverage from Montreal. Join us again in three weeks' time for the British round, which takes place at Silverstone from June 28-30th.

3.1 - Live Timing

F1LT

1/70 0:35:02 24.0°C 30.0°C 1016.4mb 3.8m/s 32.0%

P	Name	Gap	Int.	Time	S1	S2	S3	Pit
1	S. Vettel	LA	70	1:17.9	21.1	25.0	31.6	2
2	F. Alonso	14.4	14.4	1:17.0	21.2	24.8	30.9	2
3	L. Hamilton	15.9	1.5	1:16.7	21.6	24.5	30.5	2
4	M. Webber	25.7	9.7	1:17.4	21.7	24.7	30.9	2
5	N. Rosberg	69.7	43.9	1:16.5	21.4	24.4	30.6	3

That concludes our cover

3.2 - Aplicació F1LT

Per tant, veiem com la vista principal és molt similar al què repliquem per a l'aplicació.

3.4 Protocol de recepció de Dades

Un cop estem dins de l'aplicació, explicarem quin és el protocol per a anar rebent dades i com es fa perquè l'aplicació estigui permanentment connectada a l'streaming de dades.

El protocol per anar rebent dades contínuament és una mica rudimentari. Utilitzem la classe *SocketDataReader* com hem comentat anteriorment.

En aquesta classe, anirem fent *ping* repetidament cada segon al *Live Timing* (per ser precisos, cada 0,9 segons, ja que sumat al temps de processat vindria a ser l'actualització de l'aplicació cada segon).

Aquest ping, no és res més que anar enviant el 0x16 byte (podria ser un altre nombre) ja que amb això aconseguirem establir connexió amb el *Live Timing* i ens enviarà les dades generades en l'espai de temps d'aquest ping i l'anterior. **Obtindrem un block cada segon, i aquest inclourà tots els paquets nous.**

La decisió de fer-ho d'1 segon és propietat de qui ho va programar el programa, però la podria explicar a una decisió arbitrària buscant un compromís entre que l'aplicació no s'actualitzi masses vegades i col·lapsar-la , i actualitzar-ho en un període de temps massa gran i que el nombre de dades generat sigui massa elevat. Per això, 1 segon és un bon interval per tal de reproduir-ho de manera fidedigna.

Un cop fet el ping pertinent, adormim el *Thread* per tal de gestionar el temps entre crides a l'aplicació. En cas que s'hagi desconnectat, hi hauran tres intents automàtics de reconexió al servidor web de l'aplicació. Si no es connecta, l'aplicació ens mostrarà un missatge que ens hem desconnectat donant l'oportunitat a escollir si volem re-intentar connectar-nos o volem deixar-ho.

Un cop fet el ping, hem de cridar a la funció *readStream* que la seva funció serà llegir totes les dades que s'hagin obtingut de resposta al ping, que estan al Socket de Dades.

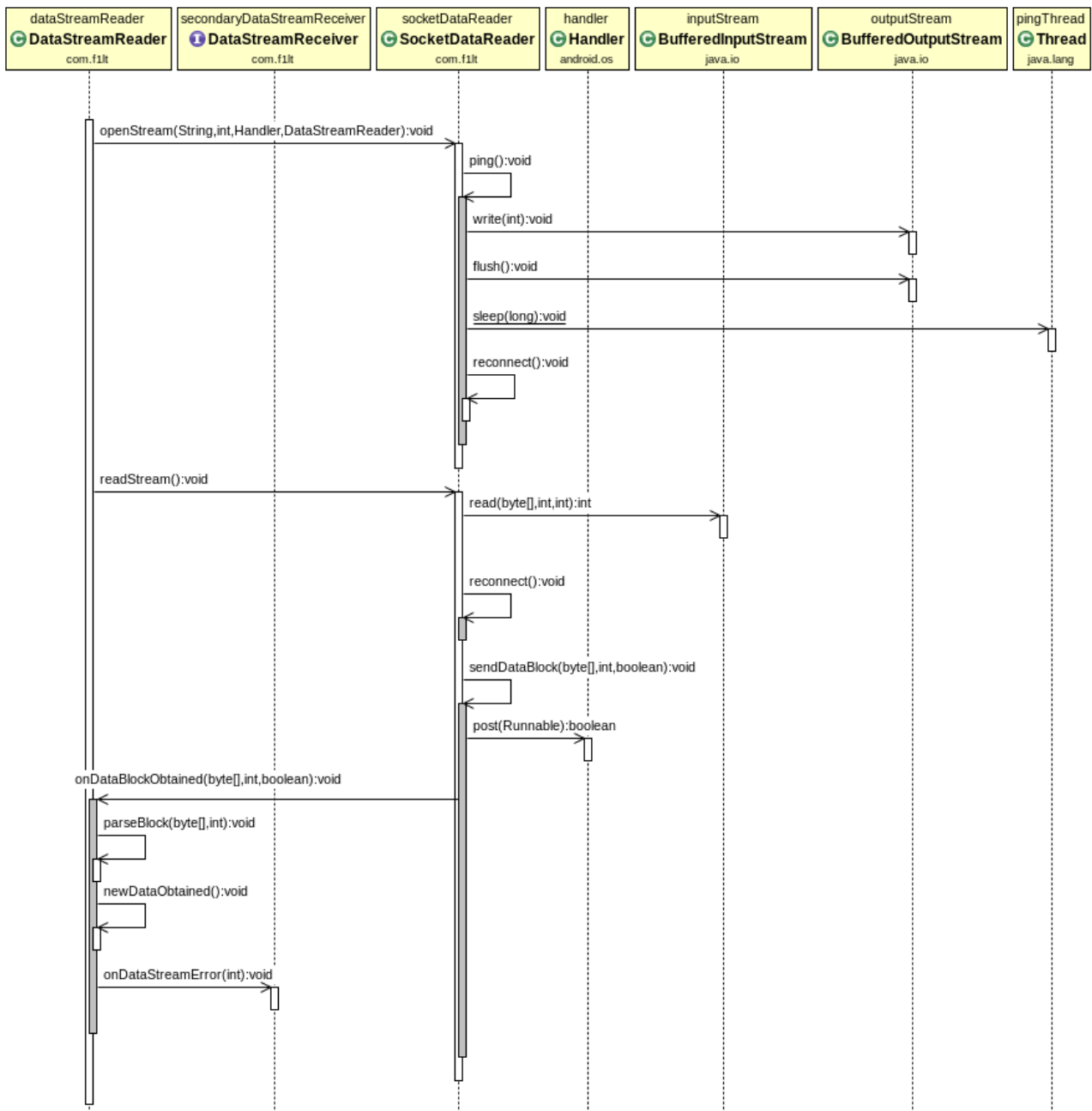
El buffer de lectura té una longitud de 8192, per tant si les dades obtingudes superessin aquest tamany tornàriem a accedir fins no tenir cap més dada per llegir.

Volem remarcar un aspecte important que es pot visualitzar als diagrames, la classe *DataStreamReader* crearà un via secundària per rebre dades que s'anomenarà *secondaryDataStreamReceiver* per tal que si s'està operant en la interfície qualsevol operació, habilitar una segona interfície per no perdre cap dada de les que s'obtinguin en cada petició al socket de dades o en el cas que estiguem dins d'alguna vista concreta de l'aplicació.

Un cop llegits, la funció *sendDataBlock* operarà l'enviament d'aquestes dades obtingudes cap a la Interfície *DataStreamReceiver* per a ser processades. Des d'allà, es processaran els paquets corresponents com explicarem en els propers apartats.

En l'interfície *DataStreamReceiver* són obtingudes les dades i a la classe *DataStreamReader* son processades. És important tenir clar la diferència per no mesclar els conceptes sobre els dos ja que operen en col·laboració.

Aquest es el diagrama de seqüència del protocol d'anar llegint les dades que rebem:



3.5 Desencriptar Dades

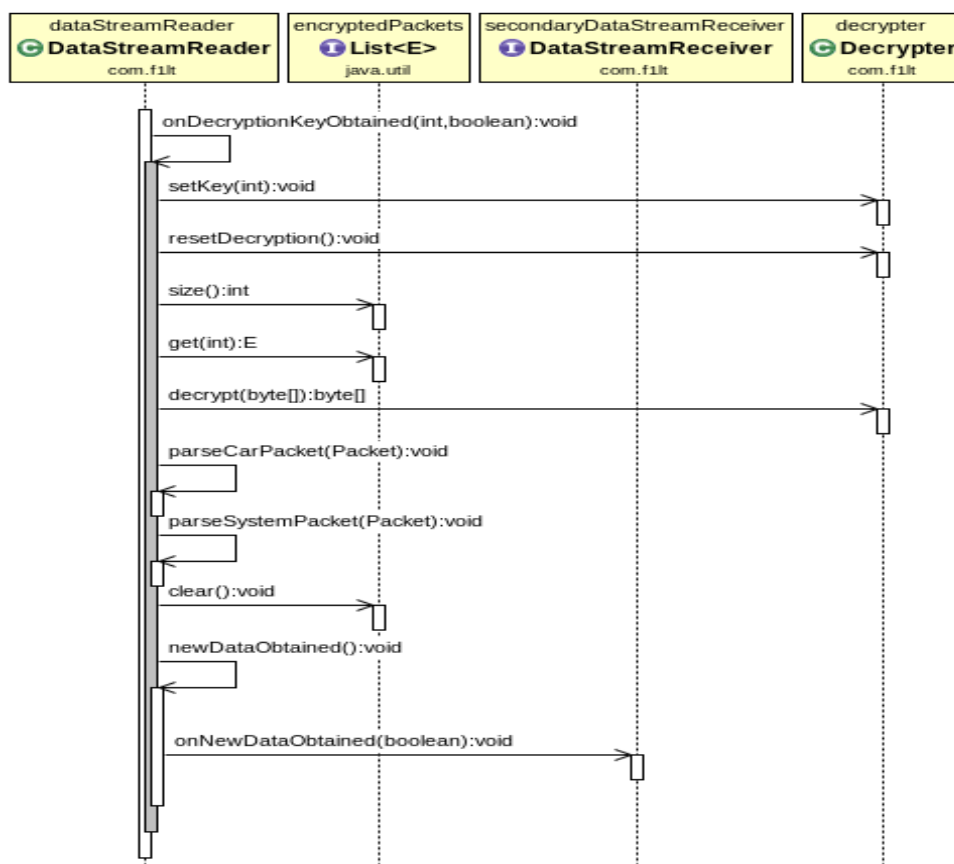
Però abans de ser processats els paquets, necessitem desencriptar. Bastants paquets venen encriptats, per tant haurem de separar entre els encriptats i els no encriptats. Els no encriptats podran ser directament processats, mentre que els encriptats seran emmagatzemats en una llista especial on ens dedicarem a desencriptar-los.

No explicarem la desencriptació detalladament, ja que creiem que no és el propòsit d'aquest projecte, però si que farem quatre pinzellades de com es gestiona.

La desencriptació no és res més que operacions de desplaçament de bits i de operacions XOR. Aquestes venen donades per el Key Frame obtingut. D'aquí la vital importància que remarcàvem anteriorment de obtenir-lo i guardar-ho, ja que d'altre manera el resultat de la desencriptació serà erroni.

Al moment d'obtenir la Key Frame, ens guardarem el valor, i posteriorment cada cop que volem desencriptar els paquets, es cridarà la funció decrypt amb els bytes a procedir per tal d'aplicar les operacions pertinents amb la màscara que parteix del Key Frame. Acabada aquesta operació, els paquets aniran al mètode que explicarem posteriorment per ser processats.

Aquest es el diagrama de seqüència del procés de desencriptació dels paquets encriptats:



És important remarcar que en aquest diagrama es veu clarament la diferència entre la classe *DataStreamReader* i la Interfície *DataStreamReceiver*, ja que al obtenir un paquet es crida a la lectura i processat d'aquest, mentre al acabar es crida a la interfície.

3.6 Tipus de Paquets

Un cop tenim desencriptats els paquets, és bo saber que cada un d'ells porta 2 bytes inicials que són els identificadors de paquets.

Hi han diferents tipus de longitud de paquets : paquets llargs, paquets curts, paquets especials, etc. Però no entrarem en això i ens centrarem en el contingut dels paquets.

Els paquets els podrem distingir entre dos tipus ben diferenciats :

- Paquets de Sistema (SYSTEM PACKETS)
- Paquets de Cotxe (CAR PACKETS)

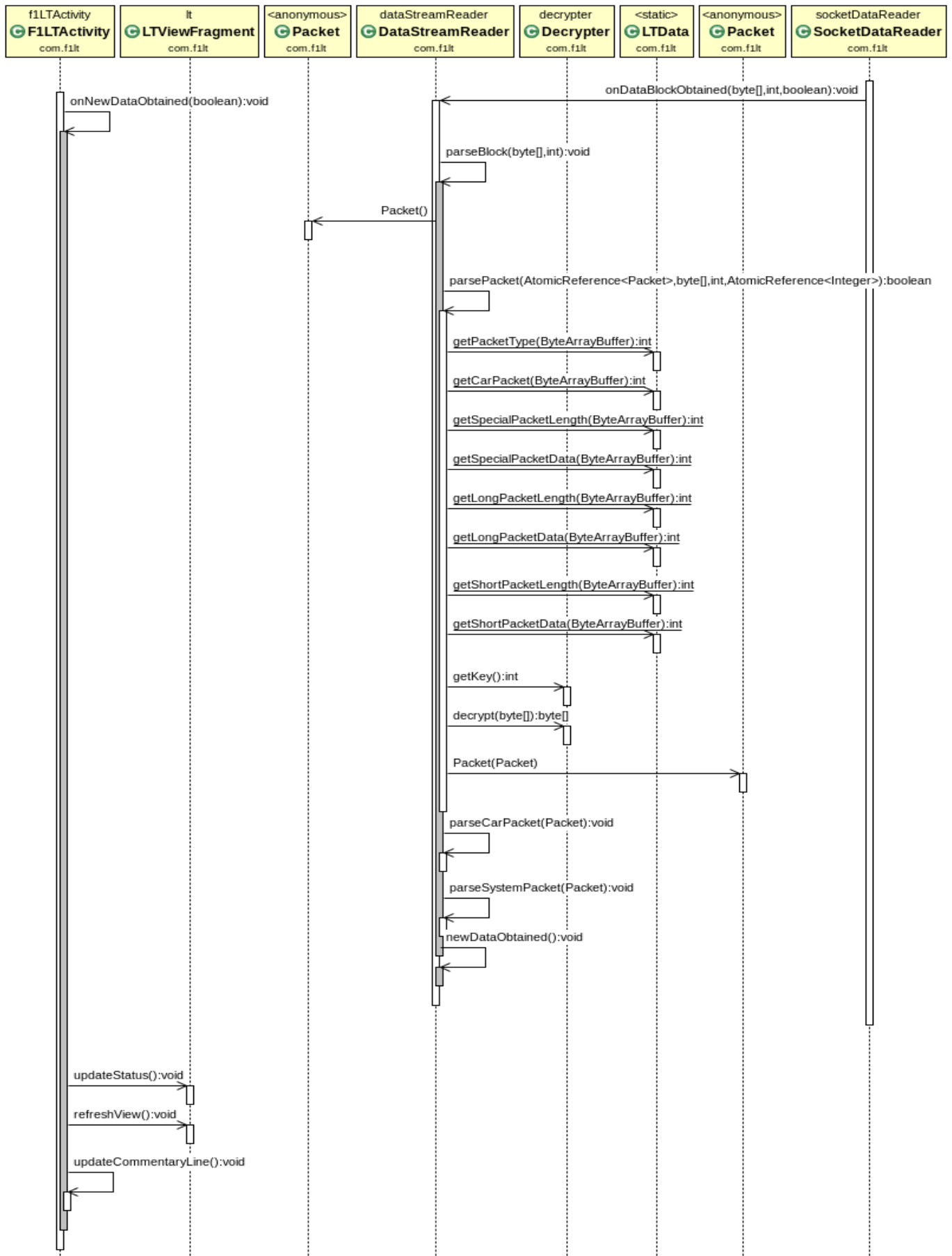
Quines són les diferències ?

Bé, els de Sistema consisteixen en la informació relativa a la sessió que s'està disputant. Per altra banda, els de Cotxe consisteixen en la informació relativa a cada un dels pilots.

Hi ha una excepció, les velocitats màxima i les velocitats màximes a cada parcial consten dins dels paquets de Sistema. Té una explicació lògica, i seria que qui ho va decidir així va veure que les dades són úniques en la sessió ja que no es controla cada velocitat de cada pilot en passar per aquell punt sinó que es controlen les màximes.

Tot seguit, explicarem concretament quins són cada un ja que tota aquesta informació serà molt important a l'hora d'emmagatzemar-ho.

Aquest es el diagrama de seqüència previ a definir si és System paquets o Car paquets :



3.6.1 Paquets de Sistema

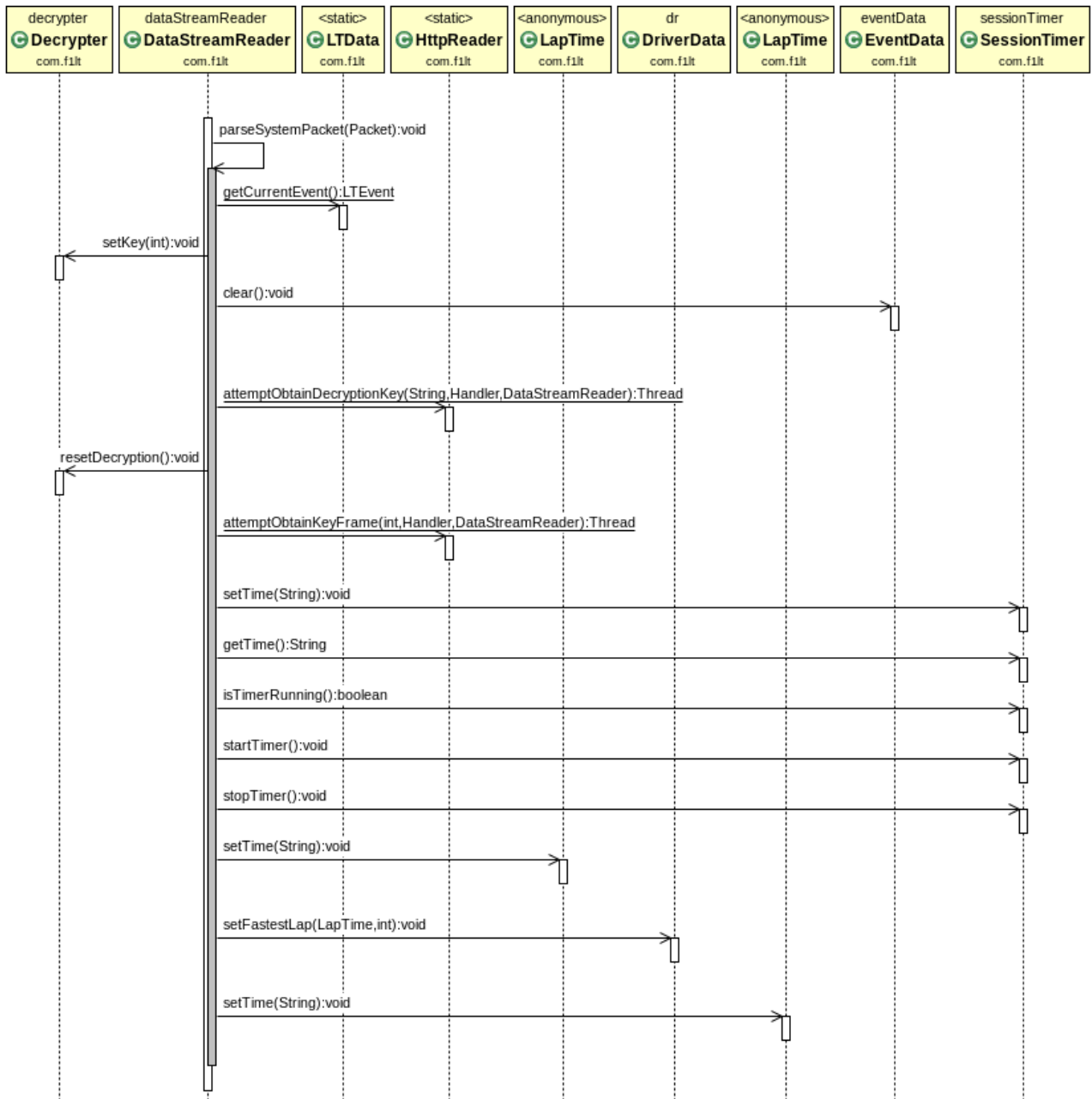
Aquesta és una relació de quin són els paquets de Sistema i la seva descripció :

<i>SYS_EVENT_ID</i>	<p>Id de la Sessió, o dit d'altra manera, <i>Key Frame</i>.</p> <p>A partir d'aquest nombre, identificarem al tipus de sessió que som de cara a mostrar-la correctament (Entrenaments Lliures, Qualificació o Cursa).</p> <p>En aquest punt, llegirem el fitxer XML per tal d'obtenir la data de la disputa de la sessió. Això pren una especial importància de cara a quan reproduïm una sessió en diferit ja que la data actual de reproducció no coincidiria amb la de disputa i no es podria reproduir al no entendre el sistema la data correctament.</p>
<i>SYS_FRAME</i>	<p>Frame actual de dades que rebem. S'actualitza cada cop que ens ho dicta l'aplicació <i>Live Timing</i>.</p> <p>En cas de no obtenir-ho correctament, cridarà a la funció de la classe <i>HttpReader</i> per obtenir-ho de nou de la web.</p>
<i>WEATHER_SESSION_CLOCK</i>	<p>Envia periòdicament el temps restant de la Sessió. No ho envia a cada sessió, per tant es poden produir petits salts en el temps que corregeixen la sincronia.</p>
<i>WEATHER_TRACK_TEMP</i>	<p>Temperatura a la pista (°C)</p>
<i>WEATHER_AIR_TEMP</i>	<p>Temperatura al aire (°C)</p>
<i>WEATHER_WIND_SPEED</i>	<p>Velocitat del Vent (m/s)</p>
<i>WEATHER_WIND_DIRECTION</i>	<p>Direcció del Vent (Graus - Direcció)</p>
<i>WEATHER_HUMIDITY</i>	<p>Humitat a l'aire (%)</p>
<i>WEATHER_PRESSURE</i>	<p>Pressió atmosfèrica (mB)</p>
<i>WEATHER_WET_TRACK</i>	<p>Descriu si plou (1) o no ho fa (0)</p>
<i>SYS_TRACK_STATUS</i>	<p>Estat de la pista, referent a les Banderes.</p> <p>Vermella : Sessió aturada Verd : Pista lliure Groc : Perill per accident o incident a la pista Safety Car : Cotxe de seguretat a la pista</p>
<i>SYS_NOTICE</i>	<p>Notificació</p>
<i>SYS_TIMESTAMP</i>	<p>Temps enviat, serveix per relacionar dades en el temps</p>

<i>SYS_SPEED</i>	<p>Velocitats en diferents punts del circuit.</p> <p>Es divideixen en :</p> <ul style="list-style-type: none"> • <i>SPEED_SECTOR1</i> • <i>SPEED_SECTOR2</i> • <i>SPEED_SECTOR3</i> • <i>SPEED_TRAP</i> (Punt de velocitat màxima al circuit) • <i>FL_CAR</i> (Volta ràpida, identifica el cotxe) • <i>FL_DRIVER</i> (Volta ràpida, identifica el pilot) • <i>FL_LAP</i> (Volta ràpida, identifica la volta) • <i>FL_TIME</i> (Volta ràpida, identifica el temps fet)
<i>SYS_COMMENTARY</i>	Comentari relatiu a la sessió

Un cop parsejades correctament les dades, aquestes es guardaran en el lloc que pertorquin de les classes *EventData* o *DriverData*. Els paquets de Sistema es guarden *EventData*.

Aquest es el diagrama de seqüència per als diferents paquets de sistema que obtenim:
(no s'inclou condicions ja que seria indesxifrable)



3.6.2 Paquets de Cotxe

Aquí es gestiona la informació generada per cada pilot.

Dins la gestió de les dades de cada cotxe, serà important a l'hora de guardar la informació i mostrar-la, que tinguem clar la sessió en què estem.

Els tres tipus de paquets que identifiquen les diferents sessions són : *RACE_EVENT*, *QUALI_EVENT*, *PRACTICE_EVENT*

Aquesta és una relació de quin són els paquets de Cotxe i la seva descripció :

<i>CAR_POSITION_UPDATE</i>	Canvi en la posició d'un pilot (o cotxe)
<i>RACE_POSITION</i>	Envia la posició d'un pilot en cursa
<i>RACE_NUMBER</i>	Número de la cursa
<i>RACE_DRIVER</i>	Nom del pilot del paquet
<i>RACE_GAP</i>	Diferència respecte al líder de la cursa
<i>RACE_INTERVAL</i>	Diferència respecte al pilot situat al davant (si el pilot està en la Posició 3, la referència seria respecte el que està en Posició 2)
<i>RACE_LAP_TIME</i>	Temps de volta marcat per el pilot
<i>RACE_SECTOR_1</i>	Temps al Sector 1 del circuit
<i>RACE_PIT_LAP_1</i>	Aturada a box durant la cursa
<i>RACE_SECTOR_2</i>	Temps al Sector 2 del circuit
<i>RACE_SECTOR_3</i>	Temps al Sector 3 del circuit
<i>RACE_PIT_LAP_2</i>	Aturada a box durant la cursa
<i>RACE_PIT_LAP_3</i>	Aturada a box durant la cursa
<i>RACE_NUM_PITS</i>	Nombre d'aturades al box realitzades durant la cursa
<i>CAR_POSITION_HISTORY</i>	Històric de posicions del pilot durant la cursa

Un cop parsejades correctament les dades, aquestes es guardaran en el lloc que pertorquin de les classes *EventData* o *DriverData*. Dels paquets de Cotxe, es guarden majoritàriament a *DriverData*, encara que hi ha algunes dades que també es guardaran en *EventData*.

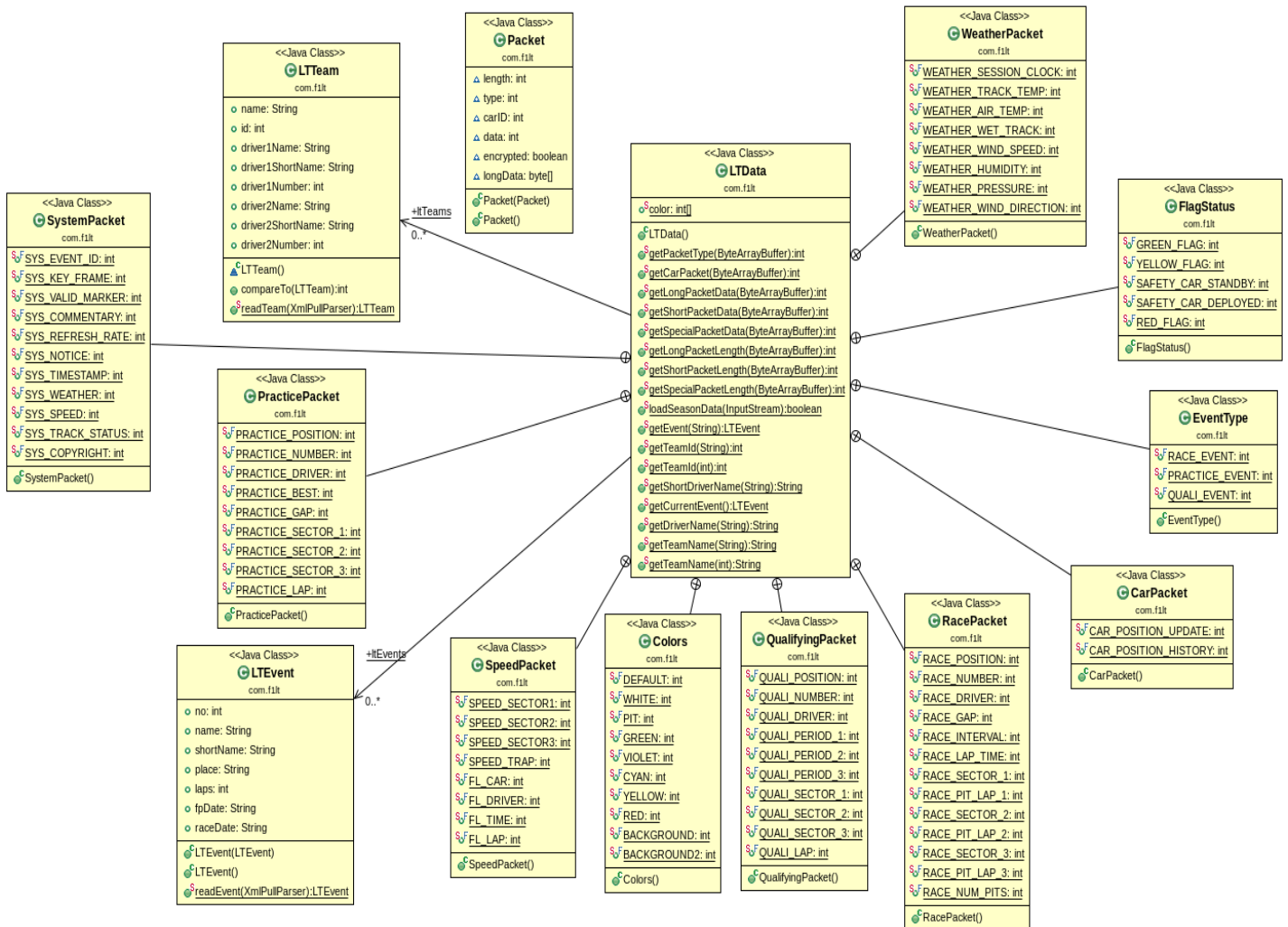
Paral·lelament a aquestes dades que es generen i es guarden, també es gestionen els colors a mostrar cada una de les dades, en cas que sigui pertinent. Aquest codi de colors està explicat al Manual d'Usuari a la secció dels entrenaments lliures, sota la imatge (6.4).

3.7 LTData

Com hem vist al darrer diagrama, LTData apareix en les crides a paquets. Bé, aquesta classe engloba tota la descripció necessària per al perfecte comportament de l'aplicació.

Com veurem al proper diagrama de classes, aquí està definit tots els paquets (contingut), com la llargària dels paquets, els equips i pilots participants, distinció de paquets entre entrenaments, qualificació i cursa, colors que necessitem per als parcials o temps de voltes dels pilots, etc. En resum, és la classe on tenim definits tots els atributs que ens dirigiran al flux correcte de funcionament.

El diagrama de classes de LTData és el següent :



3.8 Gestió de Dades. On guardem la informació generada.

La informació es guarda en dos grans blocs, les classes *EventData* i *DriverData*.

EventData inclou totes les dades relatives a la sessió, i *DriverData* les dades de pista referents a cada uns dels pilots. Amb la particularitat que *EventData* engloba les dades de l'esdeveniment, i inclou el *DriverData* com a atribut dins seu, ja que és una taula de pilots.

3.8.1 EventData

Els atributs de la classe *EventData* són els següents:

key	Defineix la clau de descriptació per l'esdeveniment
frame	Identificador de la ràfega de dades, incremental
cookie	Cookie d'accés al <i>Live Timing</i> per al usuari
eventInfo	Informació relativa a l'esdeveniment. Carregat del fitxer season.xml
eventType	Tipus de sessió (Practice 1, Practice 2, Qualyfying, Cursa..)
flagStatus	Estat de les banderes de la sessió
sessionId	Id de la sessió
firstSessionId	Id primer de la sessió
remainingTime	Temps restant de la sessió
lapsCompleted	Voltes restants que li queden a la cursa
qualiPeriod	Període que som de la Qualificació (Q1 o Q2 o Q3)
airTemp	Temperatura de l'aire
humidity	Humitat
windSpeed	Velocitat del vent
windDirection	Direcció del vent
pressure	Pressió de l'aire
sessionStarted	Atribut que defineix si la sessió està iniciada o no
wetdry	Ens diu si està plovent o no
notice	Notificació
commentary	Comentari
Sec1Speed, sec2Speed, sec3Speed	Vector de les velocitats al primer, segon i tercer parcial
speedTrap	Vector de millors velocitats al punt de màxima velocitat del circuit
FLNumber	Número de volta ràpida (és incremental a mida que van succeint)
FLDriver, FLTime, FLLap	Volta ràpida del pilot, número de volta en què s'ha marcat i temps
Sec1Record,	Vectors de les millors velocitats al primer, segon i tercer parcial

sec2Record , sec3Record	Vectors de les millors velocitats al primer, segon i tercer parcial
driversData	Llista de tots els pilots (s'explicarà el contingut al següent punt)

Per últim, aquesta classe també té la funció `calculateInterval` que donant dos pilots determinats i una volta concreta, ens calcularà la diferència entre les voltes dels dos pilots. Això serà útil per algunes *Activities* que veurem en els propers apartats.

3.8.2 DriverData

Driver Data és una classe formada per diverses subclasses, a fi d'encapsular correctament la informació sense crear una classe mare amb tota la informació.

Les subclasses de la classe *DriverData* són els següents:

Primerament tenim *LapTime* que ens defineix el temps per volta d'un pilot.

Aquesta classe fa les següents funcions que són interessants comentar :

<code>LapTime</code>	Entra el temps en milisegons i ho passa a format String (p. Ex 1:25.482)
<code>setTime</code>	Fixa el temps de volta
<code>isValid</code>	Comprova que el temps obtingut sigui correcte
<code>calc107p</code>	Calcula el 107 % d'aquest temps. Útil per qualificació ja que un temps superior al 107% del temps en primera posició significa no poder participar a la cursa. Per exemple : Temps pole : 1:20.000 -> Temps 107% : 1:25.600
<code>LessThan</code> , <code>lessEqual</code> , <code>equals</code>	Compara dues voltes per establir si milloren els seus propis registres (o els absoluts)

A continuació, tenim *PitData* que ens guarda cada una de les aturades al box del pilot.

Té dos atributs :

- [pitTime](#) que és el temps que ha estat a boxes (el total del pit-lane, per ex: 25.6).
- [pitLap](#) que és la volta en què s'ha fet l'aturada.

Aquesta classe fa les següents funcions que són interessants comentar :

<code>compareTo</code>	Compara dues parades. Funció que s'usa en l'activity de millors temps d'aturades al box.
------------------------	--

Posteriorment, tenim LapData que ens guarda la informació cada cop que es completa la volta.

Té aquests atributs :

- `carID` Id del cotxe.
- `pos` Posició del pilot.
- `gap` Diferència de temps amb el líder.
- `interval` Diferència de temps amb el pilot que té davant.
- `lapTime` Temps de la volta.
- `Sector1`, `sector2`, `sector3` Temps de cada sector de la volta.
- `numLap` Número de volta.
- `sessionTime` Temps que falta de sessió al marcar la volta.
- `qualiPeriod` Període de la Qualyfyng.
- `sclap` Atribut booleà que indica si la volta ha sigut marcada amb Safety Car.
- `approxLap` Temps aproximat marcat (en entrenaments, si no es marca el millor temps del pilot no s'envia la volta, per tant s'ha de calcular sumant els tres sectors).

Aquesta classe fa les següents funcions que són interessants comentar :

<code>sumSectors</code>	Suma els tres sectors de la volta realitzada.
-------------------------	---

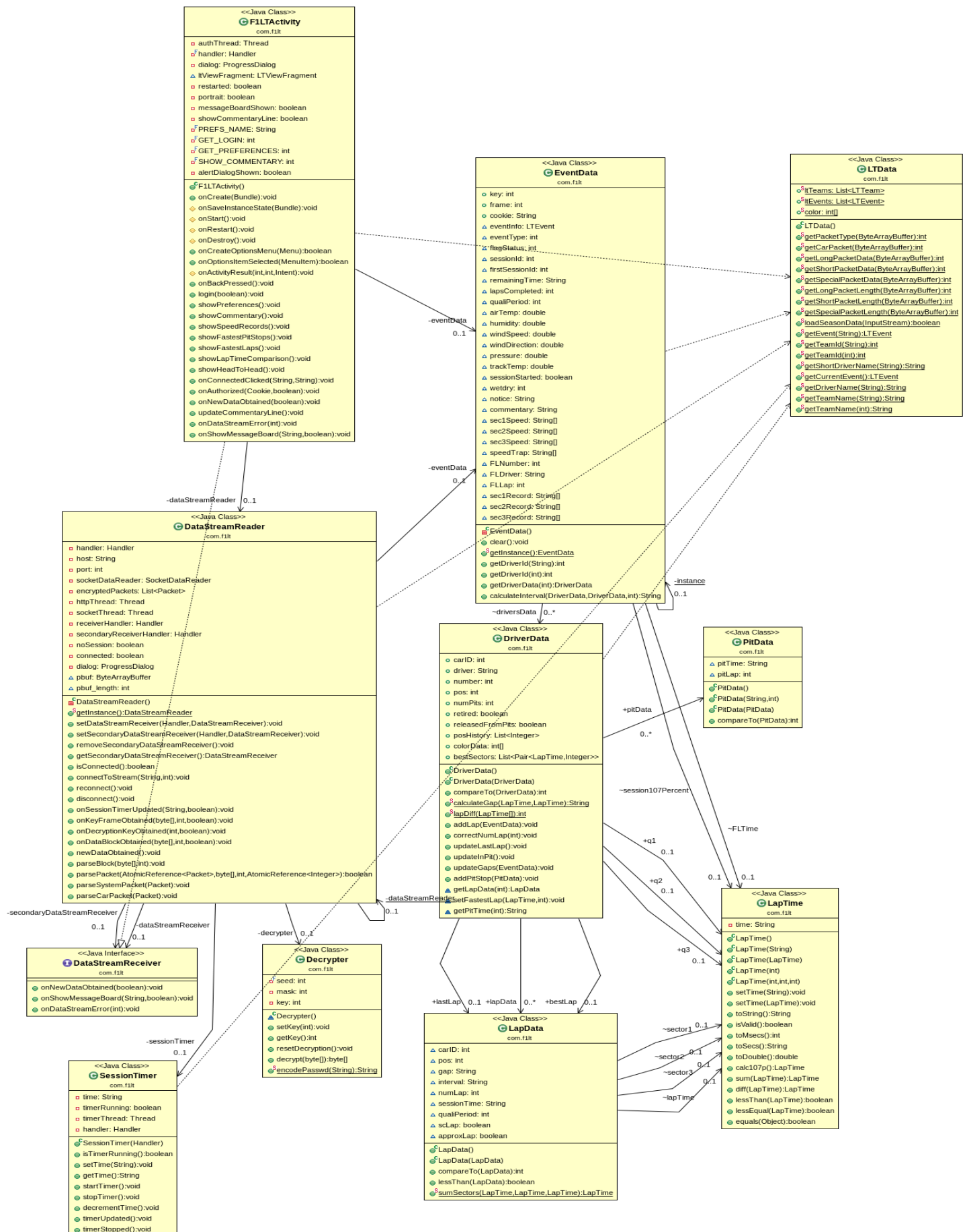
Per últim, la classe mare `DriverData` que engloba les descrites anteriorment dins seu.

Té aquests atributs (no repetirem els anteriorment comentats) :

- `driver` Nom del pilot.
- `numPits` Nombre d'aturades a box.
- `retired` Booleà que indica si el pilot està retirat o no.
- `releasedFromPits` Booleà que indica si el pilot acaba de sortir de box en aquesta volta.
- `q1`, `q2`, `q3` Millors temps que ha realitzat en la Qualyfyng, en cas de ser aquesta la sessió que s'estigui processant.
- `posHistory` Històric de posicions volta a volta durant la cursa.
- `bestSectors` Millors sectors del pilot.
- `lastLap` Últim temps marcat.
- `bestLap` Millor volta del pilot.

En aquesta classe hi haurà les funcions per accedir a cada una de les voltes, així com calcular diferències entre elles, aturades a boxes, etc.

El diagrama de classes de la gesti  de dades   el seg ent :



3.9 Season.xml

Aquest fitxer que trobem a */assets/season.xml* ens ajuda per inicialitzar :

- pilots
- equips
- grans premis (inclou dates de disputa dels entrenaments i cursa , així com les voltes de la pròpia cursa).

Aquesta informació és especialment important a l'hora de dibuixar els cotxes en la pantalla principal (les seves figures es guarden en les carpetes *drawable*), així com la data de disputa de la cursa. Això és molt important ja que compararem la data actual amb les carregades del XML per a saber l'esdeveniment que estem reproduint, entre la informació que necessitem són les voltes de l'esdeveniment a disputar. Les voltes a disputar no ens ho envia l'aplicació, per tant nosaltres ho haurem de gestionar.

3.10 Activities de l'aplicació

Fins aquí, ja s'ha explicat tota l'obtenció i gestió de les dades. Però encara ens queda una part important de l'aplicació que seran les diferents vistes en que reproduïm la informació de diferents maneres per tal de comparar diferents voltes dels pilots entre ells o veure les millors aturades al box, etc. A les diferents vistes de l'aplicació és el què en Android es coneix com Activities i són accessibles en la seva majoria a través del Menú Principal.

Les Activities en l'aplicació són les següents :

1. Commentary Activity

Aquesta vista consta de tots els comentaris realitzats durant la sessió.

2. Driver Data Activity

Al seleccionar el nom d'un pilot en la vista principal, accedim a la informació personalitzada del pilot. Consisteix en informació volta a volta del pilot, a més de l'imatge del seu cotxe així com la diferència amb els pilots amb posició precedent i posterior, etc.

3. Fastest Laps Activity

Vista que recapitula la volta més ràpida per a cada pilot, ordenades de temps més ràpid a menys.

4. Fastest Pits Stops Activity

Vista que recapitula les aturades a boxes, ordenades de la més curta a la més llarga.

5. Head to Head Activity

Vista que compara entre dos pilots, triats per l'usuari, el volta a volta així com els sectors de cada volta entre ells.

6. Lap Time Comparison Activity

Vista que compara els temps entre 4 pilots triats per l'usuari

7. Login Activity

Vista que permet entrar a l'aplicació disposant l'e-mail i password.

8. No Session Board Activity

Quan no hi ha una sessió activa, sol haver-hi les dades de l'última sessió disputada. De totes formes si el *Live Timing* està deshabilitat o queden entre 15 i 5 minuts per la disputa de la propera sessió, el *Live Timing* està deshabilitat mostrant el temps que queda per la propera sessió. Aquest canvi és degut a canvis interns que es faran, com generar un nou *Key Frame*, etc.

9. Preferences Activity

Aquesta vista serveix per triar les preferències.

10. Speed Records Activity

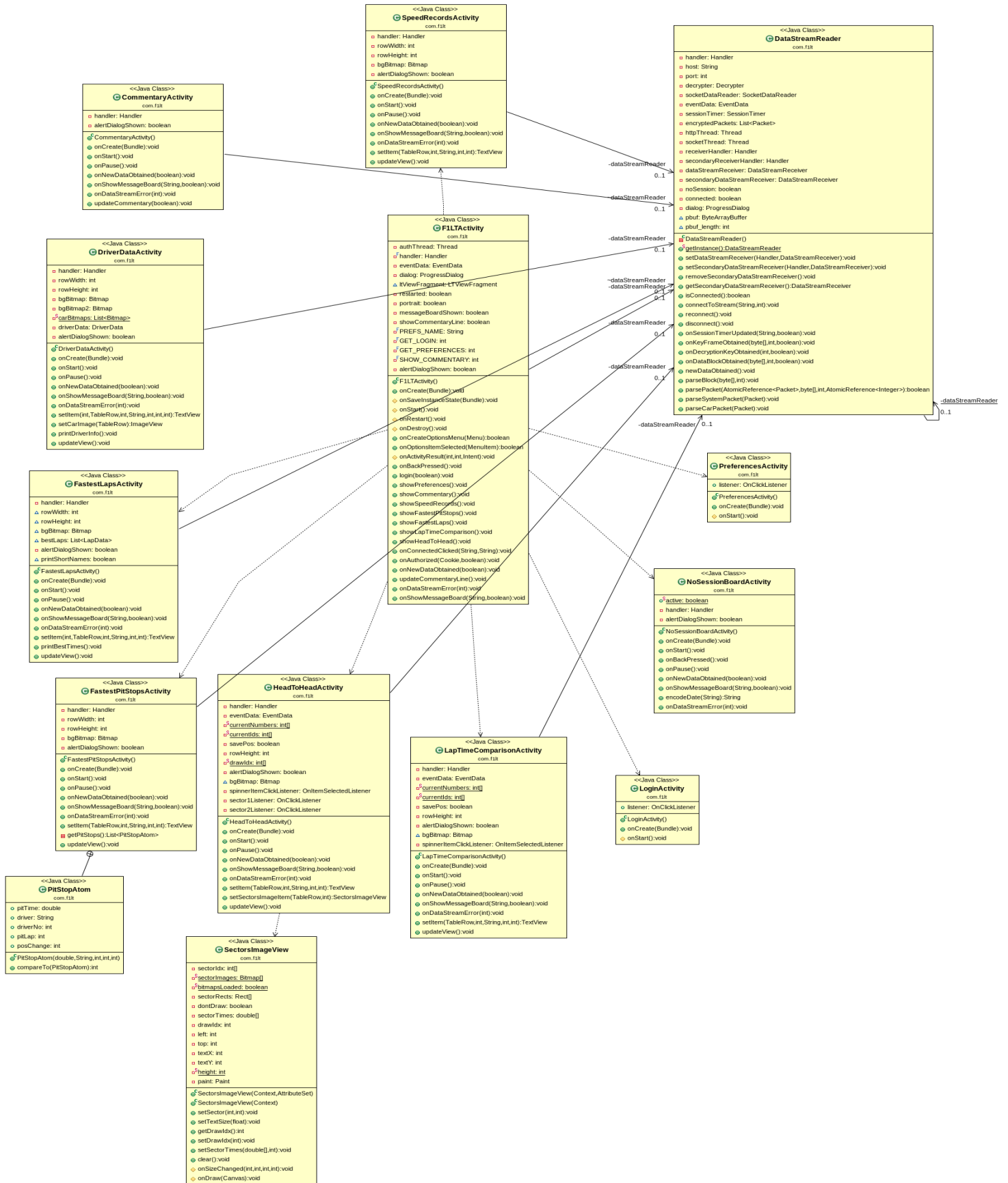
Aquesta vista consta de les millors velocitats registrades als sectors i al punt de velocitat màxima del circuit.

És important comentar un aspecte que potser teníem en compte d'apartats anteriors però que creiem que cal tornar a remarcar. En cas d'accedir a una *Activities*, l'obtenció de dades segueix en curs ja que com s'ha comentat abans, s'obre un *SecondaryDataStreamReceiver* per tal d'anar rebent els paquets de dades generats al *Live Timing*.

Al rebre dades, s'invocarà el mètode *onNewDataObtained* que gestionarà el seu processament per la classe *DataStreamReader* i també cridarà la funció *updateView* de la pròpia *Activity*.

Això no només serà útil per el fet que la navegació per les diferents vistes no suposa el perdre les dades que s'estiguin generant, sinó que també farà que dinàmicament les dades que es van rebent i llegint es puguin reproduir dins la vista sense fer cap acció addicional. La funció *updateView* ens va refrescant les dades que anem veient dinàmicament.

Aquest és el diagrama de classes per les Activities :



4. Desenvolupament

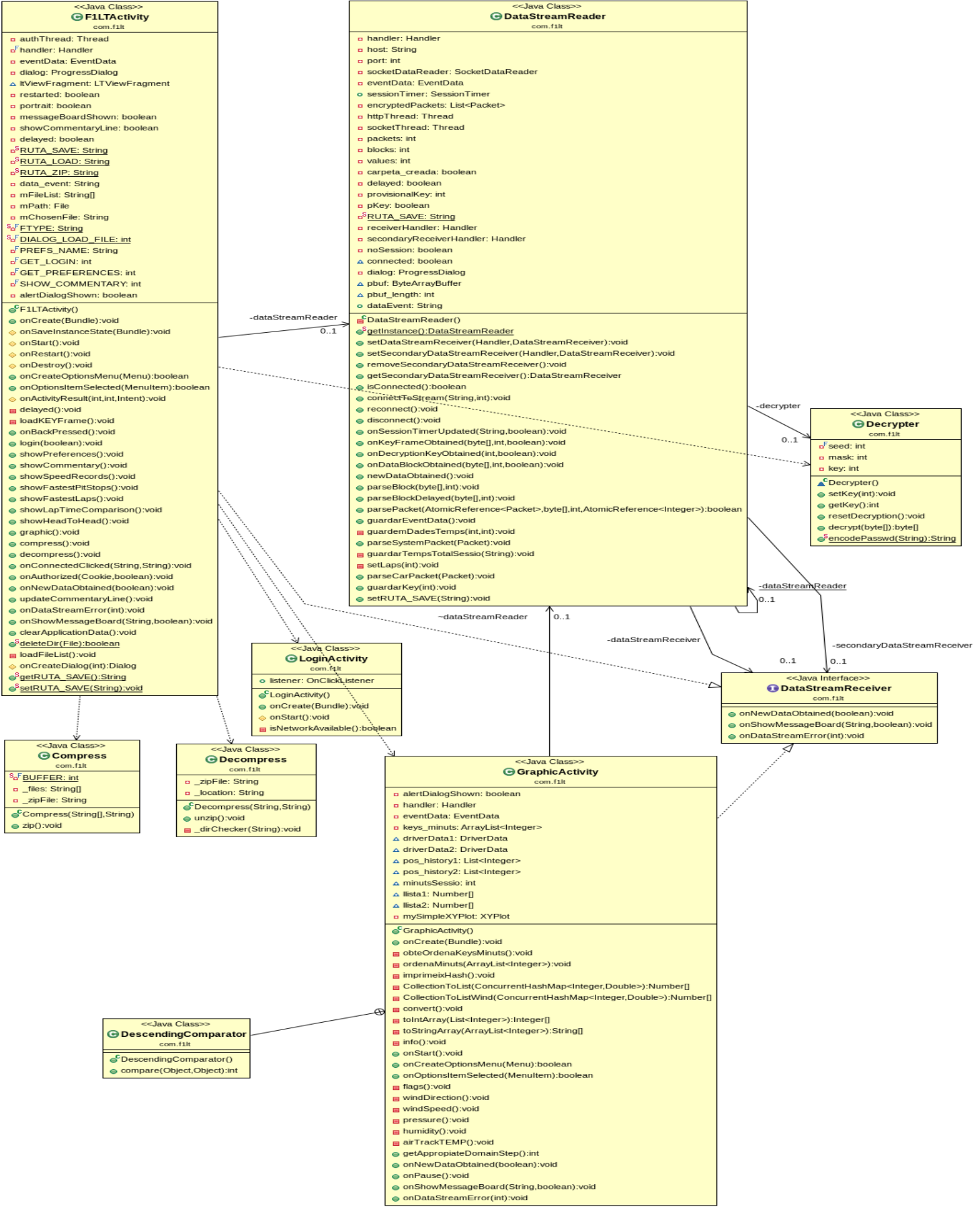
A partir d'aquí, explicarem la feina de modificació del codi per part nostra implementant les noves funcionalitats.

Continuem treballant amb tecnologia **Android – llenguatge Java**. A partir d'aquí, volem incloure noves funcionalitats al codi que siguin útils i necessàries per als usuaris de l'aplicació que els hi agrada la F1. Creiem que seria molt útil poder capturar les sessions per posteriorment poder-les reproduir.

Aquestes modificacions volem que no restin rendiment a l'aplicació i que cap funcionalitat que ja estava implementada en la versió inicial deixi de funcionar.

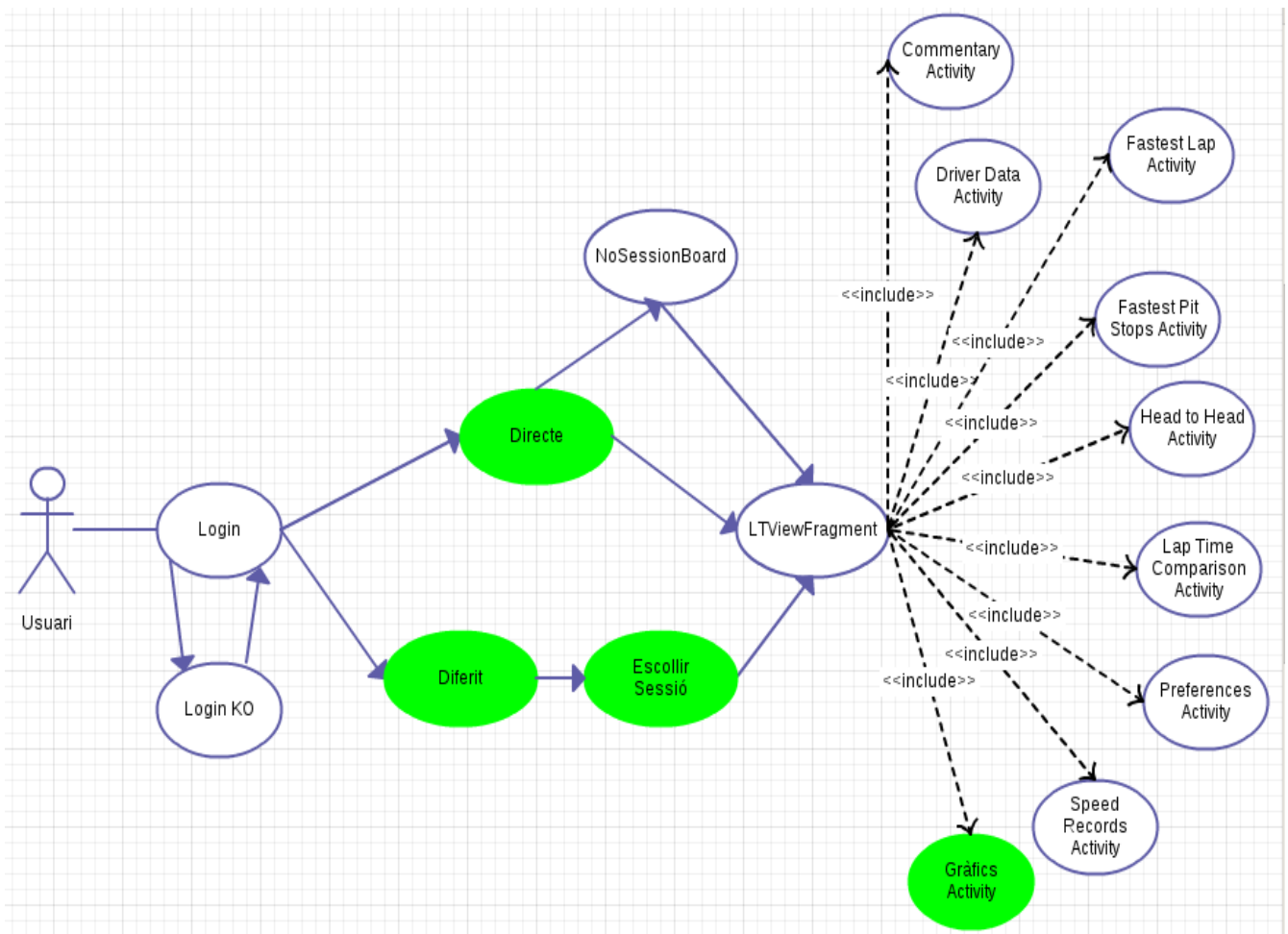
Aquí mostrem el nou Diagrama de Classes de les noves implementacions. Només afegim les noves classes implementades junt amb les classes necessàries per al seu funcionament. Les altres segueixen tenint el mateix funcionament que s'ha explicat en el **Capítol 3**.

Aquest és el diagrama de classes :



4.1 Casos d'Ús

Aquí tenim el diagrama de Casos d'Ús per a l'Aplicació :



En verd observem les funcionalitats afegides al codi de la versió inicial definida al Capítol anterior.

En el diagrama anterior podem observar com hem introduït dos camins diferents que es donaran per reproduir una sessió :

- **Directe.** Quan la sessió s'estigui disputant en directe, es conserva el procés anterior d'operació per l'aplicació.

IMPORTANT: Afegim un petit matís, per accedir al Live Timing en directe, hem d'estar connectats a Internet. Ho remarcuem perquè abans era així, però ara serà un tret important a l'hora de diferenciar-ho de la reproducció en diferit.

- **Diferit.** Quan no estem connectats a Internet, accedirem a un diàleg en què triarem la sessió que volem reproduir. Un cop descomprimida, serà reproduïda de la mateixa manera que es reprodueix en la funció directe.

IMPORTANT: Per tal de reproduir sessions, abans han d'haver sigut capturades i comprimides. Al Manual d'Usuari s'explica com fer-ho, de totes formes en aquest capítol ho explicarem amb més detall, però en el seu sentit de funcionament intern implementat.

- **Gràfics.** S'introdueixen la Gràfics Activity que mostrarà els gràfics de les dades del temps que es vagin generant durant la sessió. Aquests gràfics seran accessibles en tot moment, tant directe com diferit.

És important apuntar que Directe i Diferit no són activitats pròpiament dites, sinó una manera de diferenciar els dos camins que s'adoptaran.

4.2 Guardar les sessions

4.2.1 Ruta on Guardem Dades al Login

En l'activitat Login, si estem connectats a Internet, ens demanarà la carpeta on desarem els arxius que capturem en aquesta sessió. Al connectar, aquesta ruta serà salvada i enviada a la funció principal de la classe *F1LTActivity* on serà gestionada cap a la classe *DataStreamReader* i *HttpReader* per guardar tots els fitxers en aquesta mateixa carpeta.

4.2.2 Key Frame i Data

En aquest punt explicarem com operarem el codi per afegir la funcionalitat de guardar les dades d'una sessió. Aquesta nova funcionalitat serà implementada a les classes *DataStreamReader* i *HttpReader*.

Començarem per la part de la classe *HttpReader*, que és on ens guardem el Key Frame i la Data de l'esdeveniment.

Al accedir a la funció *obtainDecryptionKey*, que es crida des de la funció *attemptObtainDecryptionKey* ens guardarem en dos fitxers separats aquests dos atributs, en la carpeta que l'usuari ha seleccionat en la finestra Login.

4.2.3 Guardar Dades

Per poder guardar les dades, modifiquem la funció *onAuthorized* i *connectToStream* per tal d'incloure en les crides posteriors a funcions la ruta on guardarem les dades a la carpeta de la targeta SD.

Un cop estem dins la lectura de les dades dins els blocs, ens guardarem les dades a la funció *parseBlock* que és cridada cada segon que rebem dades. Aquesta captura ens l'administrarem creant una variable anomenada **blocks** que és incremental cada cop que es crida aquesta funció. Això ho fem per tal d'identificar cada bloc de dades que ens guardem de forma ordenada en el temps.

Per altra banda, també ens servirà a l'hora de donar nom al fitxer on ens ho guardem, posar-li nombre.

Exemple : Dades1.txt, Dades2.txt, Dades3.txt, etc.

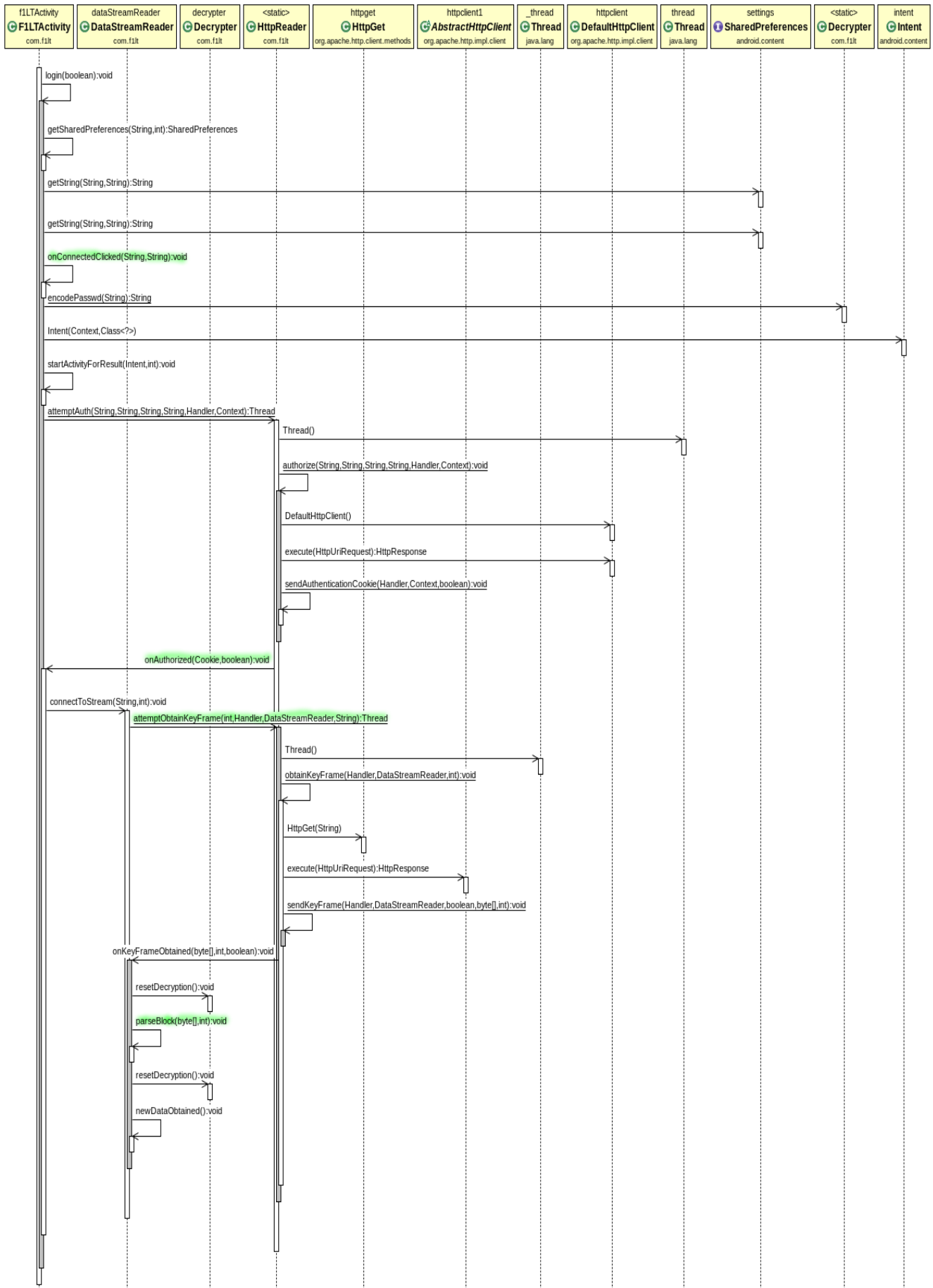
4.2.4 Sincronitzar les Captura de Dades

De la mateixa manera, en el mateix moment ens capturem un altre fitxer que hi conté l'instant del temps en què s'ha obtingut aquest paquet de dades. Això ho fem possible ja que hi ha una funció que ens proporciona el temps actual del sistema en Milisegons : *System.currentTimeMillis()*. Ens guardem les dades amb segons (obviant els 3 nombres de menor pes) als fitxers indicats.

Exemple : Milis1.txt, Milis2.txt, Milis3.txt, etc.

Per tant, tenim les dades i temps agrupades en parelles : Dades1 – Milis1 , Dades 2 – Milis 2, etc.

Aquest és el diagrama de seqüència per la captura de dades en directe :



4.3 Reproduir les sessions

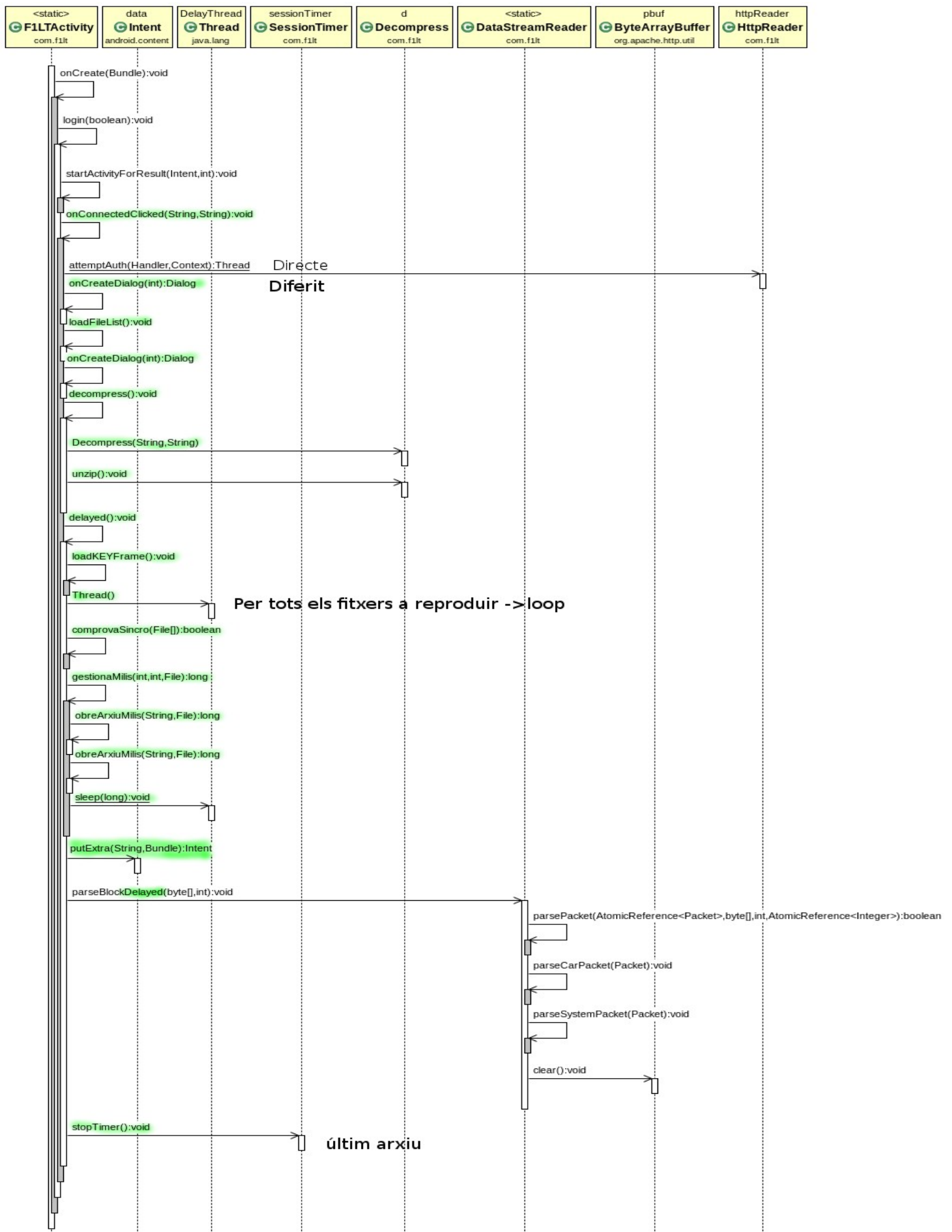
A l'hora de reproduir les dades que ja s'han guardat en el moment de la sessió en directe, hem optat per tenir una [RUTA_LOAD](#) que ens apuntarà a la ruta (directori) que volem reproduir les dades.

Per tant, serà tan senzill com anar a la sessió que l'usuari ens indiqui i reproduir-la. Però com ens hem comprimit les sessions, al descomprimir-les ho farem sempre a una carpeta que tindrem habilitada per la fi de reproduir les dades sense necessitar d'haver de canviar on apunti [RUTA_LOAD](#).

Per tant, al **Capítol 4.5** s'explica detalladament el procés de compressió i descompressió.

Ara explicarem detalladament el funcionament de la funcionalitat, però abans mostrarem el diagrama de seqüència abans de explicar-ho, ja que en aquest cas serà més útil per no embolicar-nos.

IMPORTANT: En **VERD** es mostraran les noves funcionalitats implementades perquè constin les diferències respecte la versió inicial.



4.3.1 Funció Diferit. Descomprimir dades, carregar Key Frame i Data

Fins arribar a la invocació de Login, segueix el mateix esquema que a la versió inicial. Però aquí hi ha el primer canvi que és guardar-se la ruta de la SD de gravació de dades.

I arribem a la primera modificació en la funció `onConnectedClicked` on comprovarem si estem connectats a Internet o no.

- En cas d'estar **connectats** a Internet : Accedim al *Live Timing*, invocant la funció `attemptAuth` de la classe *HttpReader*. Aquesta funcionalitat és igual que l'anterior.
- En cas de **NO** estar **connectats** : Se'ns crea un Dialog, i se'ns mostra totes les sessions comprimides en arxius .zip a la carpeta F1LT/ZIP. La funció `loadFileList` ens ajuda a llistar totes les sessions per mostrar-les a la vista.

Fet això, descomprimeix les dades a la carpeta F1LT/UNZIPPED/ mitjançant la funció `decompress` . Les funcions homònima i `unzip` són internes i s'encarreguen d'aquesta missió.

En aquest punt, invoquem la funció `delayed` que serà l'encarregada de tota la gestió de la reproducció de sessions capturades anteriorment.

En primer lloc, es cridarà la funció `loadKEYFrame` que serà l'encarregada de carregar el *Key Frame* de la Sessió així com la Data de disputa de la sessió. Aquestes dues operacions son vitals perquè :

1. Si no carreguem el **Key Frame**, no es podrà reproduir de forma correcta la sessió. A més, s'ha de carregar abans de iniciar el processat de dades, ja que quan s'inicialitzin els primers paquets de cotxe, aquesta dada es carregada automàticament per al programa. Per tant, ens ho guardarem localment a la classe *DataStreamReader*, i quan aquest vulgui carregar-la, serà tant senzill com transferir-ho de la variable local guardada a la localització definitiva. Aquesta funció local que ho operarà serà `guardarKey` i l'atribut `pKey` de *DataStreamReader*.

Sembla rebuscat aquest mètode, però la metodologia i **carregar el Key Frame en el moment correcte fa que una sessió es reproduïxi bé o no.**

2. Al carregar la **Data**, ens assegurem que reproduïm l'esdeveniment adequat. D'altre manera, si per exemple intentem reproduir la cursa de Barcelona el dia que es disputa la cursa de Mònaco, ens apareixeria a la pantalla les voltes del GP de Mònaco enlloc del de Barcelona.

Perquè succeïx això ? Com hem explicat en el Capítol anterior, el arxiu *Season.xml* es la inicialització dels Gran Premis, a més dels pilots i equips. Als Grans Premis ens guardem el nombre de voltes de l'esdeveniment, per tant és IMPRESCINDIBLE guardar la data de l'esdeveniment capturat sinó en el moment de reproduir, les voltes serien errònies.

Per assegurar-nos que no fallarà en cap cas, hem previst també en el cas que la data guardada sigui errònia o altres casos, la funció de la classe *DataStreamReader* corregirà les voltes a 78 que son les que té el GP de Mònaco. Aquest esdeveniment és el que té el major nombre de voltes, per tant en cas que no fos aquest l'esdeveniment, sempre comptaríem les voltes per excés i tots els grans premis es reproduirien correctament per excés.

ACLARIMENT : La correcció de voltes és important ja que si al iniciar la reproducció ens diu que hi haurà 60 voltes i l'esdeveniment n'ha tingut 65, les últimes 5 voltes serien reproduïdes però no serien guardades. Aquesta funció que ens corregeix les voltes és `setLaps` dins de la classe `DataStreamReader`.

Un cop carregades aquestes dues dades, obrim un nou Thread per tal d'operar la lectura en paral·lel al procés principal. A més, també es fa ja que el procés principal és qui ha d'operar el flux de l'aplicació mentre la lectura s'encarregarà el Thread que anirà cridant la classe `DataStreamReader` per operar les dades.

4.3.2 Funció Diferit. Llegir arxius i Sincronització

Ara llegirem els arxius de dades. Aquí hi hauran dos tipus de reproducció de la sessió :

- Sincronitzada
- No Sincronitzada

Aquesta diferència ve donada perquè els primers GP en ser enregistrat, estàvem en fase de refinament de la gravació. Per tant, no varen poder ser gravats en quin moment era capturat cada dada. S'ha explicat la gravació de dades al apartat 4.2.

Per tant, hem buscat una manera de poder reproduir totes les sessions sense que influeixi que estigui o no sincronitzada.

Els temps de retràs entre paquets serà efectuat amb el següent càlcul :

TEMPS ENTRE PAQUETS = TEMPS RETRÀS AL SEGÜENT * TEMPS PING (1 segon)

Per tant, hem d'obtenir el “TEMPS DE RETRÀS AL SEGÜENT”

- No Sincronitzada

Al no posseir els instants del temps quan hem rebut les dades, tampoc podem saber quan mostrar-les amb exactitud. Però hem optat a una decisió arbitrària que consisteix que mostrarem cada dada que tinguem amb un interval de 1 segon.

Això ho hem fet ja que l'aplicació fa ping (demanar dades al Live Timing) cada segon, per tant intentarem reproduir la sessió, encara que sigui de forma accelerada ja que si incrementéssim la diferència entre cada reproducció, el temps total seria més fidedigne però alguns temps es mesclarien la distorsió de lent i altres ràpid. En els primers 4 paquets, fixem la diferència a 5 segons per tal que es carreguin ordenadament el Key Frame i Data.

Per tant, preferim fer-ho tot més ràpid. TEMPS DE RETRÀS AL SEGÜENT = 1 SEGON

- Sincronitzada

Al tenir capturats els instants del temps per cada paquet, el què farem serà accedir al paquet actual i obtenir el temps d'aquest paquet. Paral·lelament, entrarem al proper paquet per saber el temps de captura.

Per tant, operarem així per obtenir :

TEMPS DE RETRÀS AL SEGÜENT = TEMPS FINAL – TEMPS INICIAL * 1 SEGON

Per l'últim paquet a reproduir, establim 1 segon de retràs.

Per tant, aquest temps de retràs serà el temps que adormirem el *Thread* per tal de simular correctament la sincronització de les dades com si fos la recepció del Directe. Això serà la funció *Thread.sleep(segonsDelay*Delay)*

4.3.3 Funció Diferit. Bundle – Handler.

Aquest apartat és de vital importància en tot aquest procés de reproducció. Ens arriscaríem a dir que és el més important.

En el procés per simular que estem en directe en la sessió, necessitem per un cantó anar llegint les dades i per l'altre, anar processant. Doncs això vol dir que del *Thread* que hem creat per anar llegint les dades, hem de buscar una manera de traspasar tota aquesta informació al flux principal de l'aplicació. Doncs bé, això ho farem amb un Bundle, que no és més que una estructura que ens permet passar qualsevol tipus de dada al flux principal de l'aplicació (*Handler*).

Un cop definit el Bundle, inserim les dades a la funció *sendMessage* que transferirà des de la lectura de dades cap al Handler. Però per fer això, necessitarem definir-nos una funció que reculli la informació que hem dipositat al Bundle des de el flux principal de l'aplicació.

Aquesta funció que ens recollirà el Bundle des de el flux principal serà *handleMessage* que l'haurèm creat dins la inicialització de *Handler* a la classe principal *FLIActivity*. I dins d'aquesta funció, obtindrem les dades del Bundle.

Per últim, només ens caldrà fer dues coses més :

1. Cridar la funció *parseBlockDelayed* de la classe *DataStreamReader* perquè ens llegeixi i processi les dades.
2. Cridar la funció *onNewDataObtained* de la Interfície *DataStreamReceiver* per fer-li saber que hem rebut dades. Aquest procés ens les reproduirà per pantalla, de la mateixa manera que ens ho feia a la versió inicial de l'aplicació.

MÉS INFO : Per obtenir informació addicional a l'explicada o per entendre com funciona, recomanem aquest [LINK](#) o [LINK2](#) on explica la idea del que hem aplicat en l'aplicació.

4.3.4 Funció Diferit. Fi de paquets per llegir.

Per últim, només ens queda senyalar una última cosa. Per tal de facilitar per l'usuari la visualització de quan s'ha acabat de reproduir una sessió, s'ha optat per parar el temps quan es llegeixi l'últim paquet. D'aquesta manera si la sessió no està acabada de gravar fins el final, l'usuari podrà localitzar que no hi han mes dades a reproduir quan vegi que s'ha parat el temps.

4.4 Gràfics del Temps

En aquest punt, per dibuixar les gràfiques ens hem ajudat de la llibreria [Androidplot.com](https://www.androidplot.com/) que dona cobertura en la seva implementació.

En aquest punt, volíem reproduir les dades del temps que es van produint durant la sessió. Però per realitzar aquesta funcionalitat, ens enfrontem a una limitació que és la aplicació en què tenim aquests atributs però no conservem els valors històrics al llarg de tota la sessió.

Per tant, per guardar l'històric de valors havíem de prendre la decisió arbitrària de en quins moments fer-ho, però fer-ho en un interval coherent de dades per tal de no sobrecarregar l'aplicació de dades repetides.

La decisió ha sigut de guardar les dades cada cop que en la sessió queden unitats de temps exactes, és a dir : quan queden 1:00, quan queda 2:00, etc.

Per tant, ens hem creat una operació que va comprovant si el temps restant de la sessió és X minuts exacte, i en cas de complir-se la condició, guardem les dades.

La gravació de dades hi hem fet amb Hash-Tables Concurrents, en què guardem en cada Hash conté un tipus de valor diferents. **Aquestes taules estan guardades a *eventData***, igual com els atributs que ens fan mostrar-ho per pantalla les dades actuals. Aquesta funció que guarda les dades és `guardarEventData` a la classe *DataStreamReader*. Al final d'aquest punt explicarem les crides internes.

Hem escollit les Hash-Tables Concurrents perquè el accés als elements és directe quan hi accedim per el seu identificador. Aquesta és l'avantatge que buscàvem ja que hi haurà un valor únic per cada minut. D'aquesta manera l'accés és automàtic i ens evitem la cerca entre molts valors d'una taula.

Resumint, l'exemple de guardar la temperatura de 44° quan falten 30 minuts per acabar seria així : 44.0 (el valor de la temperatura està en Double – enters amb decimals) i l'identificador seria 30.

Així, per els **entrenaments lliures** es guardaran de manera decreixement, seguint el valor del temps restant de la sessió.

En la **cursa**, guardarem el temps per el temps restant de la cursa. (La cursa va per voltes, però el temps màxim de disputa d'una cursa són 2 hores. De totes formes era mes senzill implementar-ho també per temps a la cursa, ja que capturem les dades en un **interval regular** , en canvi si es capturés per voltes, en cas d'haver Safety Car, el temps de volta es pot arribar a doblar.

De totes formes, a la **Qualificació** s'ha hagut de improvisar una nova forma de guardar-ho.

Les sessions de Qualificació tenen el següent format :

Q1 : 20 minuts.

Descans 7 minuts

Q2 : 15 minuts.

Descans 8 minuts

Q3 : 10 minuts.

La qualificació dura 1 hora, en total, amb la suma de les sessions i descansos. Poden haver-hi circumstàncies que l'allarguin, com bandera vermella per accident. Però en aquests casos, com no avança el temps, seguim partint de la base que durarà 1 hora.

Per tant, a l'hora de guardar dades de les diferents s'ha establert que els minuts es guardaran :

Q1 : 60 minuts a 40 minuts.

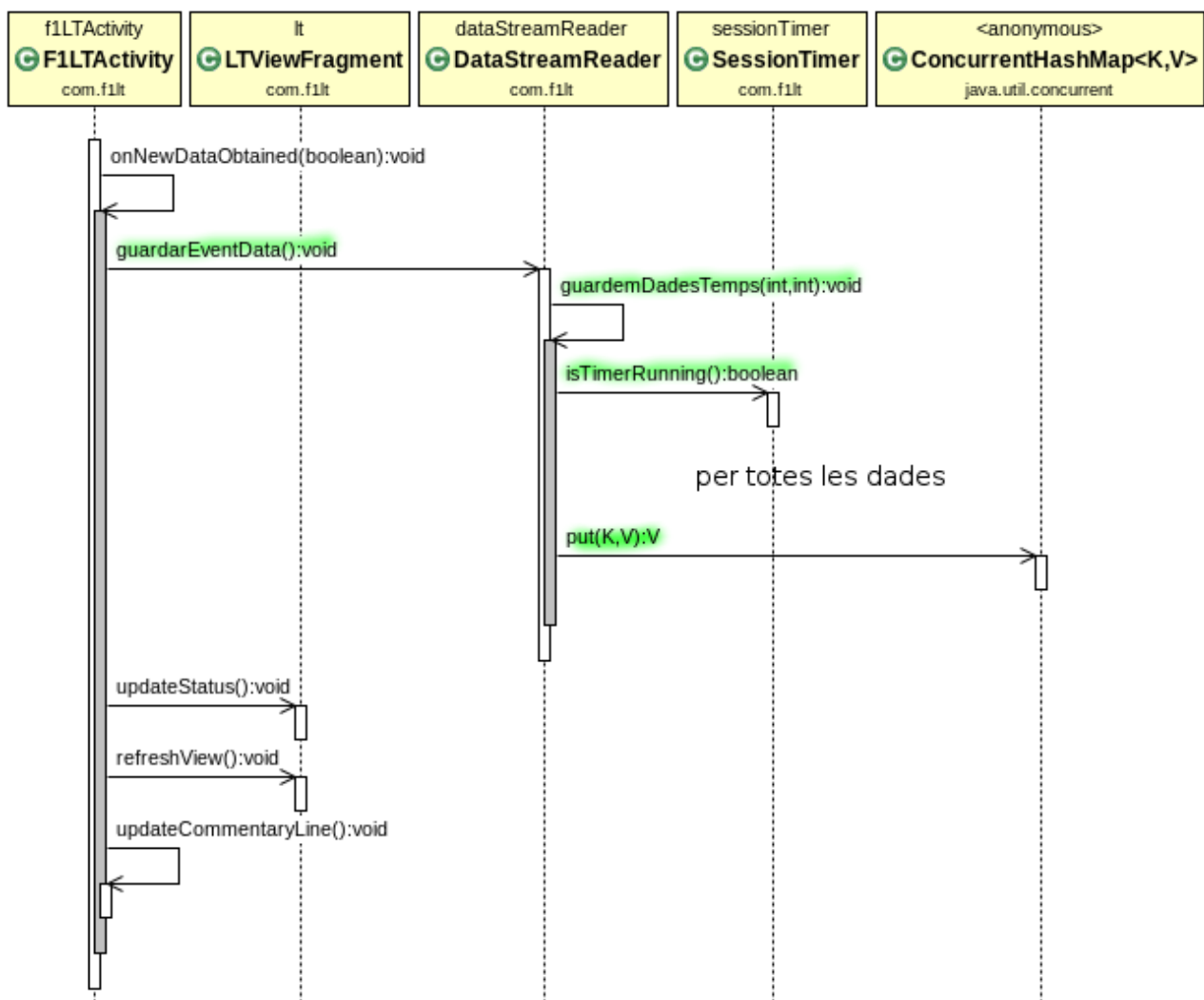
Q2 : 33 minuts a 18 minuts.

Q3 : 10 minuts a 0 minuts.

Per tant, és una qüestió simplement de depenent de quina Q estem, acumular els minuts que queden per acabar la sessió (sumant el temps de descans també).

Un cop fet això, ja tenim les dades guardades, ara s'ha de crear la nova Activity i mostrar les dades.

Aquest és el diagram de seqüència del procés de guardar les dades als minuts exactes :



Amb verd, indiquem les noves funcionalitats que capturen les dades en Hash-Table cada minut exacte.

4.4.1 Creació d'Activities

Per realitzar aquesta funcionalitat, necessitem crear una nova Activity.

Genèricament, no és difícil crear-la ja que els diferents programes que permeten programar en Android ja tenen bones Interfícies que et guien en la creació. De totes formes, és important que al utilitzar la llibreria que ens hem referit anteriorment, aquesta sigui introduïda dins l'arxiu "graphic.xml" dins de la carpeta del projecte F1LT/res/layout/ sinó no es visualitzaria la gràfica al accedir dins la vista pròpia.

IMPORTANT: Dins la carpeta /res/layout/ es defineix l'arquitectura de les Interfícies d'Usuaris dins de l'activity.

4.4.2 Composició de les Gràfiques

En aquestes gràfiques, hem intentar reproduir-les el màxim de semblants a com està en el Live Timing per web, reproduint també els colors que representen cada una de les dades.

Al iniciar l'activity, haurem d'ordenar els identificadors dels minuts capturats. Això és necessari perquè un cop ordenats de manera descendent (igual com el temps de la sessió), podrem accedir a les dades corresponents de cada Hash amb els minuts de forma seqüencial per obtenir les dates. Tenim la funció que implementa l'ordenament dels minuts de forma descendent és `obteOrdenaKeysMinuts`.

La ordenació es farà al iniciar l'activitat o al entrar en la funció concreta de pintar cada gràfica. Això ho hem implementat així de cara a que l'**actualització de dades en la gràfica** sigui **dinàmica** a mida que es van llegint dades i processant-les a la classe *DataStreamReader*.

En resum, que si som dins d'una gràfica durant més de 1 minut, al anar canviant d'una a l'altre es veuran actualitzades amb els valors processats fins al moment i no fins al moment de crear l'activitat.

També és important comentar que al enviar els valors per mostrar la gràfica, aquests han de ser en format Number (pot ser enter, double, etc). De totes formes, aquests han hagut de ser transformats a Number ja que al passar els valors primitius, la llibreria no era capaç de llegir-los. Per això hem creat la funció `CollectionToList` en què es converteix a una llista de Numbers.

Per altra banda, predefinidament al iniciar l'activitat mostrarem la gràfica de Temperatura a l'Aire i Temperatura al Asfalt.

Per canviar entre les diferents gràfiques, hem creat un menú que al polsar la tecla Menú ens apareixerà a sota de la pantalla, la funció `onOptionsItemSelected` ens gestionarà a quina gràfica ens mostra en funció de la què haguem triat.

Podrem escollir entre les 6 diferents que hem fet :

1. Air Temperature / Track Temperature . Temperatura a l'aire i a l'asfalt.
2. Humidity . Humitat.
3. Pressure . Pressió a l'aire.
4. Wind Speed . Velocitat de l'aire.
5. Wind Direction . Direcció de l'aire.
6. Flag Status / Wet – Dry . Estat de les banderes i Mullat-Sec.

4.4.3 Particularitats de les Gràfiques. Aspectes addicionals.

Quan una sessió no estava activa, als moments abans d'iniciar, la gràfica no tenia valors ja que només guardàvem valors amb el temps actiu, per tant vàrem decidir guardar també valors del temps abans d'iniciar la sessió amb el valor dels minuts totals que tindrà aquesta per tal que no ens produís error si entraven a l'activity abans de córrer el temps.

També important que ens hem fet una funció anomenada `getAppropriateDomainStep` que per l'eix X (minuts) gestiona la forma eficient per mostrar els passos entre cada minuts. Hem definit que pot haver-hi un màxim de 16 minuts a mostrar, a partir de superar aquests, els passos s'anirà augmentant mostrant un salt de 2 minuts en el label inferior.

Per exemple, si tenim menys de 16 minuts, es reproduïxen tots, si hi han entre 16 i 32, els salts al label seran de 2 minuts, etc.

Funcions interessants que hem implementat en aquesta Activity :

<code>SimpleXYSeries</code>	Crea una nova serie de dades.
<code>SimpleXYSeries. ArrayFormat. Y_VALS_ONLY</code>	Associa les dades al eix Y.
<code>LineAndPointFormat ter(Color.rgb (238, 0, 0)</code>	Afegeix la serie amb línia entre punts, i punts en la dada. A més, defineix el color de la serie de dades.
<code>addSeries</code>	Afegeix la serie al gràfic.
<code>setTitle</code>	Fixa el títol de la gràfica.
<code>GetGraphWidget(). getGridBackgroundP aint().SetColor (Color.BLACK)</code>	Fixa el fons de la gràfica a negre.
<code>SetDomainValue Format</code>	Crea un nou Format per les dades del eix X. En aquest cas, ens hem creat el format Number per pintar els intervals de minuts.
<code>GetGraphWidget(). setRangeLabelWidth (25)</code>	Defineix la separació de l'eix Y.
<code>Range Domain</code>	Range = Eix Y ; Domain = Eix X.

<code>setRangeLabel</code>	Llegenda o títol per l'eix Y.
<code>getLegendWidget().setMarginBottom(float 2.5)</code>	Defineix la separació de la llegenda amb el marge inferior.
<code>SetRangeBoundaries(0, 5, BoundaryMode.<i>FIXED</i>);</code>	Fixa el domini de valors per el eix Y (en aquest cas, entre 0 i 5).
<code>setRangeStep(XYStepMode.<i>INCREMENT_BY_VAL</i>, 1);</code>	Fixa el passos que augmenta els valors del eix Y.

Altres apreciacions per gràfiques concretes :

- Per la Direcció del vent s'ha fixat el marge entre 0° i 360°.
- Per la gràfica de banderes s'usa entre els valors 1 i 5. I en Wet / Dry son valors binaris, Wet (1) i Dry(0), per tant les dues series poden ser complementàries i reproduir-les juntes.

4.5 Comprimir - Descomprimir

Un cop estàvem provant la gravació i reproducció de les dades, ens vàrem trobar amb un problema important. Al gravar dades, arribava un moment que ens deia que no hi havia suficient espai a la targeta SD per gravar més dades (això en el cas que comprovéssim la gravació per Log, en cas que no ho féssim, no es gravaven les dades arribat aquest punt). El problema era estrany ja que les dades guardades en una cursa arriben com a màxim a 2MB.

Aquest problema és degut al **sistema de fitxers de les targetes SD**, que estan en format FAT32 i permet un emmagatzematge determinat en una mateixa carpeta.

- Màxim nombre de fitxers al disc : 268,435,437
- Màxim nombre de fitxers a una carpeta : **65,534**

Per tant, la segona condició és la que ens limita en la gravació d'arxius.

Això és degut que agrupem les dades gravades en la carpeta F1LT/ i la carpeta corresponent del GP i Sessió. En una cursa com Mònaco, es graven uns 8000 fitxers. Per tant, tenint a disc gravats les sessions de dos Grans Premis diferents, ja som a prop del límit dels fitxers en una carpeta.

Per tant, vàrem optar per afegir la funcionalitat de Comprimir i Descomprimir de ZIP les sessions per tal d'estalviar-nos lloc a disc, un cop acabades les sessions.

És important apuntar que les dades seguiran gravades a disc, i queda a disposició de l'usuari el esborrar les dades. Això s'ha decidit així perquè en cas d'haver algun problema en la compressió o descompressió de dades, hi hagi una segona opció de fer-ho manualment.

RECOMANACIÓ : Esborrar les dades un cop estiguem segurs que son correctes i ho hem comprovat al programa. Deixar com a màxim les sessions de 1 Gran Premi a disc.

4.5.1 Comprimir

A l'hora de comprimir les dades, aquesta opció és accessible des de el Menú principal de l'aplicació. Aquesta opció es pot fer durant el transcurs de la sessió o un cop acabada, recomanant fer-ho quan s'hagi acabat.

La sessió es guardarà a la carpeta especial habilitada en guardar totes les sessions comprimides, que serà /F1LT/ZIP/

El nom on es guardarà l'arxiu es la ruta que s'ha introduït en el diàleg de Login, al iniciar la aplicació, però canviant el caràcter “/” per “_”.

Per implementar aquesta funcionalitat, ens ajudem de la llibreria *java.util.zip*

Per últim, també s'ha controlat que mentre estem reproduint una sessió (DIFERIT), no es pugui comprimir-la ja que ja l'haurem obtingut d'un fitxer comprimit. Per tant, aquesta funcionalitat només s'usa en **Directe**.

4.5.2 Descomprimir

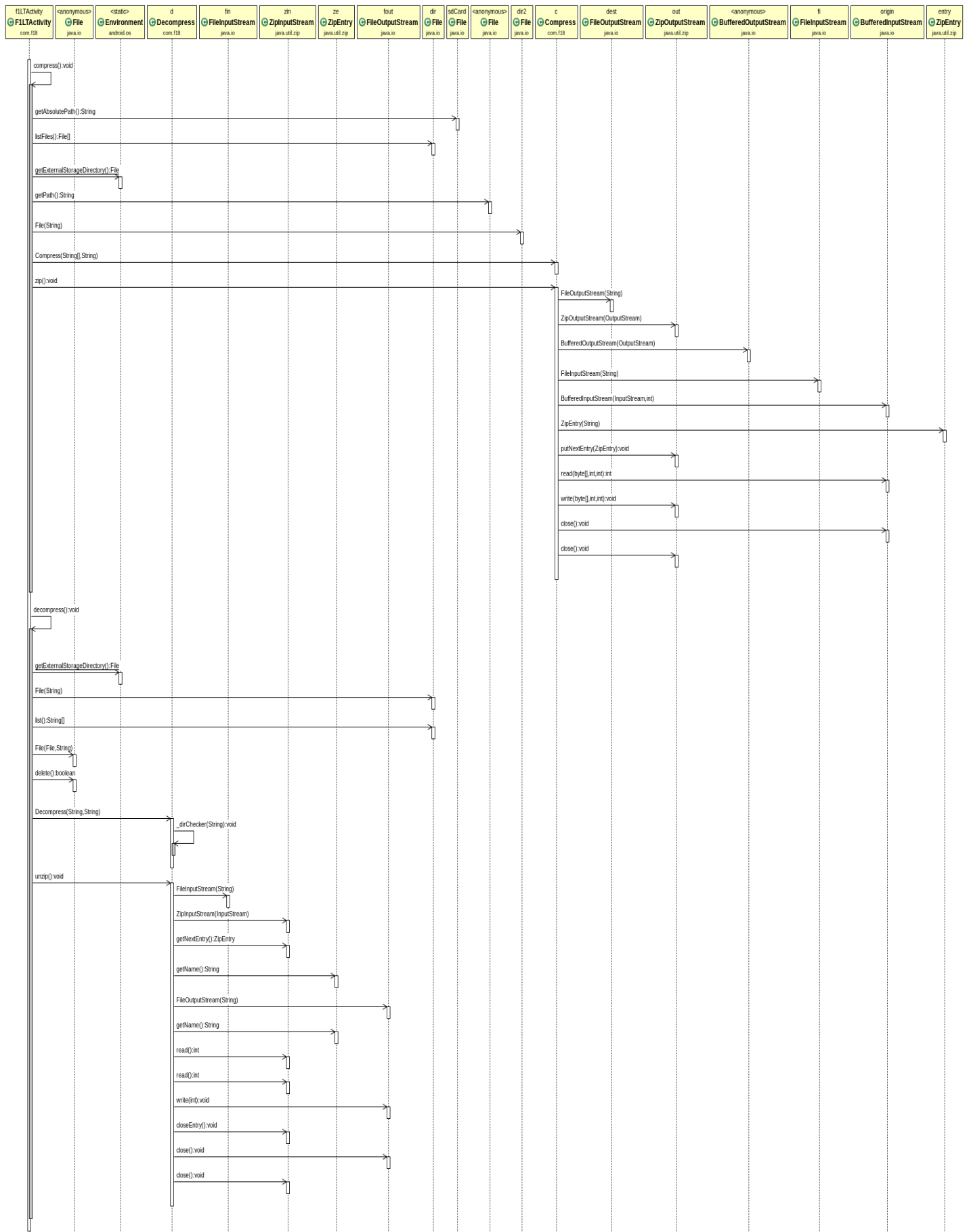
A l'hora de descomprimir, el procés és semblant a comprimir però a l'invers. La major diferència és que en el procés de descomprimir s'utilitzarà al anar a reproduir una sessió ja enregistrada, en la funció **Diferit**.

El procés es cridarà quan estiguem desconnectats d'Internet, i seleccionem que volem reproduir una sessió. Ens apareixerà un diàleg on seleccionarem la sessió desitjada, que està en format .zip a la carpeta F1LT/ZIP/

Un cop seleccionada, es descomprimirà a la carpeta F1LT/UNZIPPED/. Aquí es dipositaran tots els fitxers i un cop acabada la descompressió, s'iniciarà la reproducció d'aquests fitxers.

Important remarcar que els fitxers que hi hagin a la carpeta /UNZIPPED/ abans de descomprimir una sessió seran esborrats quan intentem descomprimir una sessió, de manera que el procés de descompressió serà una mica llarg ja que primer s'han de esborrar els fitxers existents a la carpeta i després descomprimir la sessió.

Aquest es el diagrama de seqüència per la compressió i descompressió :



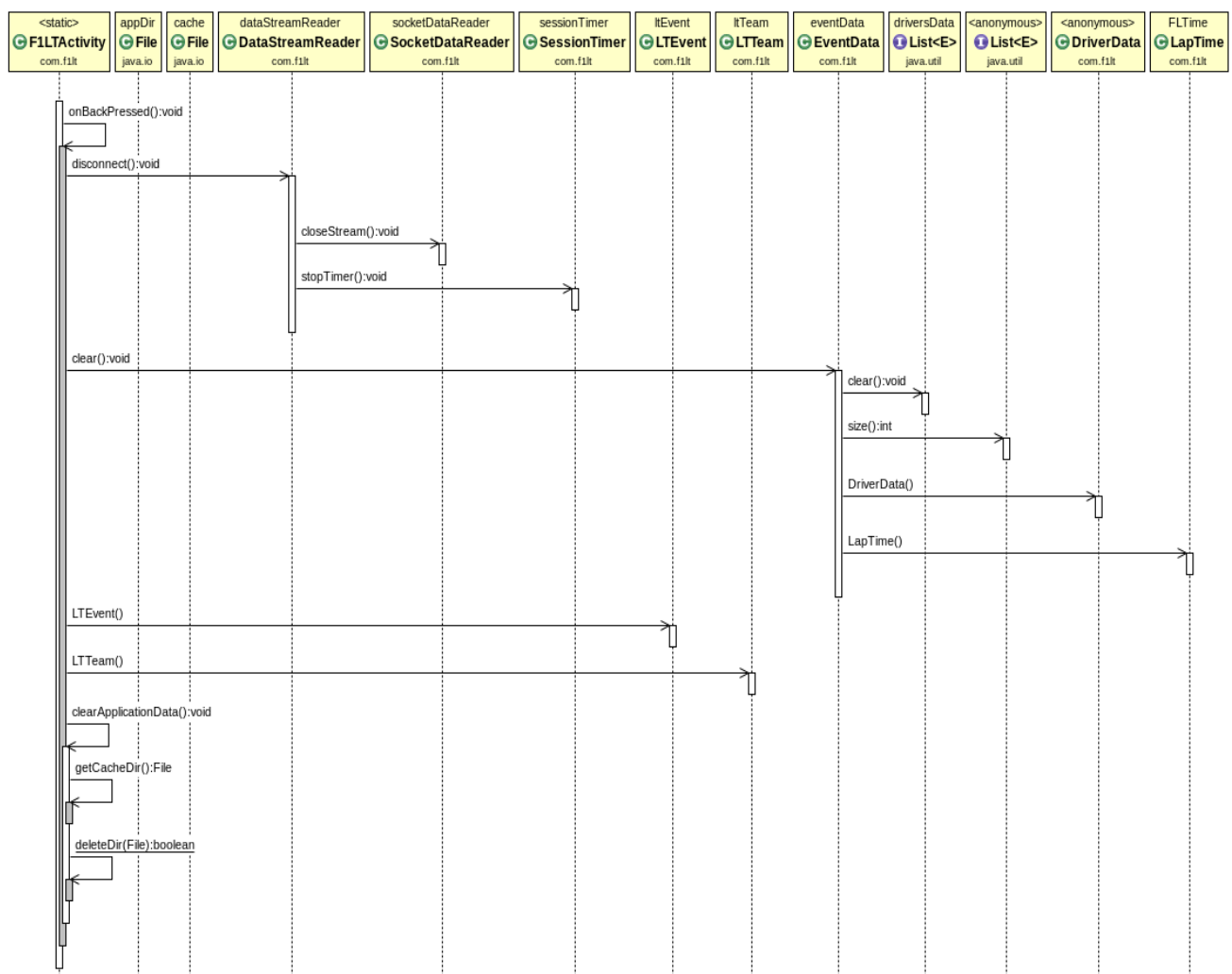
4.6 Netejar les dades temporals de l'aplicació

Ha sigut necessari buscar alguna manera d'esborrar algunes dades temporals que se'ns generen durant l'execució de la sessió. Això és degut que algunes vegades quan re-obrim l'aplicació, els valors que compten el nombre d'arxius actuals capturats eren els mateixos que al haver tancat l'aplicació.

Per això, amb l'ajuda d'algunes pàgines d'Internet especificades a la Bibliografia, hem utilitzat funcions que ens netegen els valors i atributs de l'aplicació.

D'aquesta manera hem aconseguit que al iniciar l'aplicació, tots els valors s'iniciïn als valors habituals del principi i el funcionament sigui l'esperat.

Aquest és el diagrama de seqüència al tancar l'aplicació :



5. Resultats obtinguts

5.1 Grans Premis que s'han provat

Des de l'inici de temporada hem estat enregistrant les curses sistemàticament, excepte la primera d'Austràlia que va ser a mode de prova per veure com s'executava les diferents sessions en directe en el mode analític de les diferents funcions internes de l'Aplicació.

En el Gran Premi de Bahrain ja es va finalment poder gravar de forma correcta les dades i reproduir-les en l'aplicació, per tant aquest Gran Premi va ser la prova definitiva que el treball fet anteriorment de comprensió del codi, saber on s'utilitza cada funció i on aplicar els canvis, havia finalment funcionat.

Tornem a remarcar el fet diferencial que vàrem entendre que era clau per guardar les dades, guardar el Key Frame de cada sessió. Al tenir-ho guardat, fa que les dades guardades puguin ser descriptades correctament. D'altra manera, la reproducció no és correcta ja que el caràcters descriptats no són consistents (il·legibles).

De totes formes, la reproducció no estava perfeccionada ja que no es guardava l'instant de temps en què es rebia cada paquet de dades, per tant la reproducció d'aquesta cursa és desincronitzada en el temps, durant menys temps la reproducció del Gran Premi al que realment va ser en la seva disputa.

Tal com s'ha explicat en el **Capítol 4**, la solució per reproduir-ho de la forma més fidedigna possible ha sigut aplicar una diferència de 1 segon entre cada dada a reproduir.

Aquesta ha sigut una decisió arbitrària, ja que s'hagués pogut triar-ne una altre, però al saber que les dades es reben cada segon (podent no rebre dades durant un període de temps), preferíem que es mostrés la cursa més accelerada que a la realitat que no pas aplicar més diferència de reproducció entre les dades en què algunes fases de la cursa les dades s'haguessin mostrat més lentes del que succeïen a la realitat.

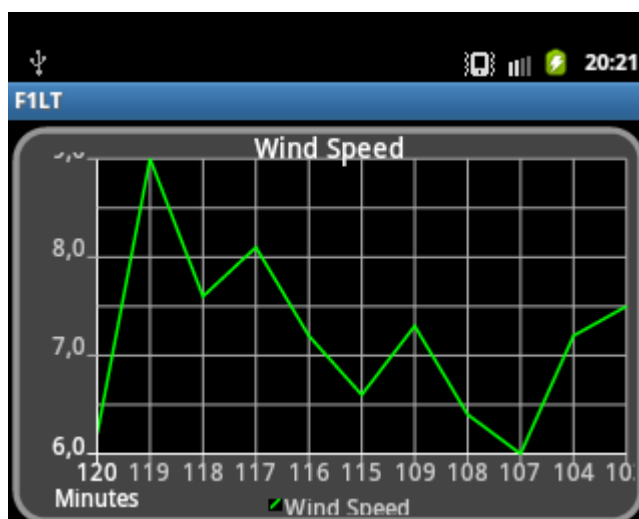
Per últim, també remarcar que els primers 4 blocs de dades que reproduïm, els separem amb 5 segons cada un ja que són paquets d'inicialització de dades i com operem Threads en diferents part de l'aplicació, algunes dades podien ser enviades a descriptar abans que les anteriors. Això provocava errors de reproducció que feien impossible la correcta reproducció de l'esdeveniment, per això que aplicant aquest criteri la reproducció de les dades, i en general de la sessió, és correcta.

A partir del GP de Montmeló, tots els Gran Premis han sigut capturats i reproduïts sincronitzadament.

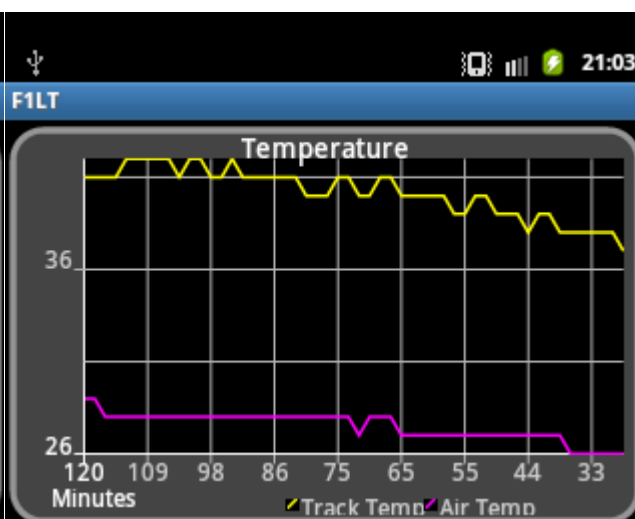
GP de Bahrain

Com veurem en els exemples, la cursa de Bahrain es reproduceix correctament però a velocitat accelerada. A simple vista, pot costar trobar diferències però ara en mostrarem algunes.

On veurem clarament que la reproducció va accelerada és en el fet que no es capturen les informacions del temps, es visualitza en les gràfiques del temps (eix X) en què als minuts hi han salts irregulars de temps.



5.1.2 - Primers minuts de Cursa



5.1.3 - Tota la Cursa

Veiem com en detall (5.1.2) si es nota els salts de captura entre minuts, però a la llarga no es nota ja que periòdicament l'aplicació envia el temps corregit de la sessió el qual maquilla una desincronització en la reproducció de dades a la llarga (5.1.3).

Per altra banda, la **reproducció** del GP dura **51 minuts** mentre que el **guanyador de la cursa** ho va fer en **1h 36 minuts**. Per tant, es comprova que el temps és sensiblement més curt que el real, la meitat aproximadament.

També observem que les primeres voltes el temps avança més ràpidament que no pas més tard en la cursa. Això té sentit ja que al anar els diferents cotxes agrupats, les dades es concentren en moments, fent que hi hagi forces segons que no es reben dades.

Més tard en la cursa, al estar els cotxes repartits per el circuit, les dades son més fàcil que es produeixin repartides en el temps i hi hagi menys segons sense generar-se dades.

Respecte als altres resultats de la reproducció, són els esperats i correctes.

GP de Montmeló

En aquest GP ja el varem poder reproduir sincronitzadament després de guardar l'instant de temps que obteníem les dades. De totes formes, el dispositiu mòbil no funcionava del tot bé aquest cap de setmana.

Al enregistrar les sessions en dos dispositius diferents i en dues modalitats diferents, es poden observar diferències de comportament entre ambdós.

En un dispositiu, el Samsung Galaxy Mini 1 es captura a través de Wi-Fi.
A l'altre dispositiu, Samsung Galaxy Ace es captura a través de Paquets de Dades.

Doncs bé, les dades capturades a través de Wi-Fi no eren reproduïbles (bé, es reproduïen però era il·legible) mentre les capturades amb Paquets de Dades eren correctes. Amb l'excepció de la cursa, la qual si va ser reproduïda en les dues modalitats capturades, les sessions de qualificació i les de proves lliures del divendres capturades per Wi-Fi eren errònies, com veiem a la imatge.

5.1.4 - Reproducció correcta

5.1.5 - Reproducció incorrecta

Veiem que a la imatge esquerra (5.1.4), la reproducció és correcta, mentre a l'esquerra (5.1.5 – capturat amb Wi-Fi), els primers paquet es reben correctament, però els següents no són correctes creant que tota nova dada sigui il·legible. A més, col·loca colors estranys a la columna del dorsal, posició i nom de pilot.

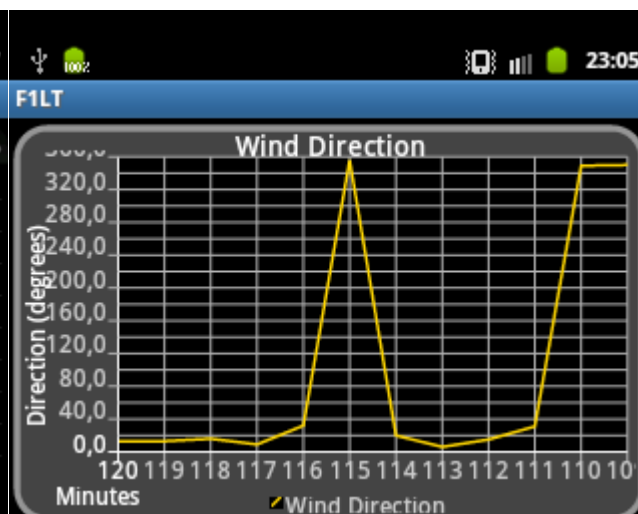
No es va localitzar amb exactitud la causa de l'error, però imaginem que seria alguna disfunció interna en la recepció de dades del mòbil, ja que el mateix amb l'altre dispositiu funcionava correctament amb la mateixa versió de l'aplicació instal·lada.

La cursa és gravada correctament amb Paquets de Dades, i els resultats són els esperats tant en sincronització com en reproducció.

Com a particularitat, observem (5.1.7) un canvi de direcció del vent lleugera però el gràfic al anar de rang 0° a 360° al variar 30° a 355° es produeix el que sembla un canvi pronunciat en la tendència de la gràfica.



5.1.6 - Cursa Montmeló

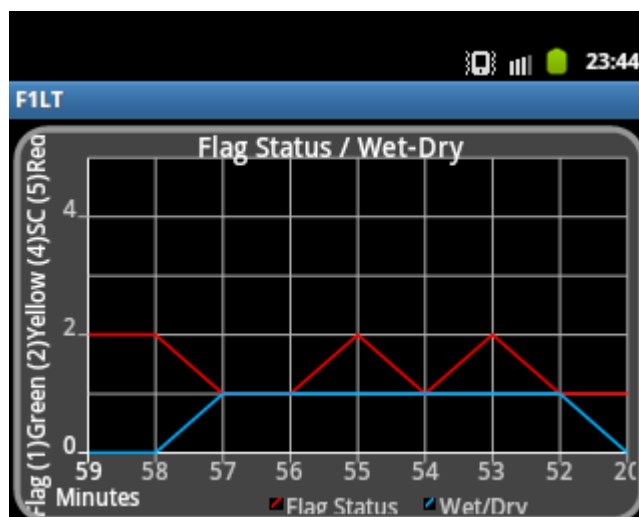


5.1.7 - Canvi en la direcció del vent.

GP de Mònaco

En aquest GP vàrem aplicar un canvi més, ja que la quantitat d'arxius que es poden emmagatzemar en una SD és limitat degut al sistema d'arxius FAT32 que s'utilitza. El nombre total d'arxius és 65536, per tant hem aplicat compressió a .zip en els arxius capturats a fi de no topar amb el limit. Això fa que al descomprimir la cursa (mes de 8000 arxius), trigui uns 10 minuts en descomprimir els arxius necessaris i esborrar els arxius anteriors de la carpeta de reproducció.

Per altra banda, en aquest GP varen haver-hi força incidents tant en la cursa com en vèries sessions, per tant es poden observar bé en les gràfiques el canvi d'estat de les banderes durant el transcurs de la sessió. També va ploure, amb lo qual s'observa variacions entre quan està sec i mullat en la gràfica de Wet-Dry.



5.1.8 - Banderes a la Qualyfyng de Mònaco



5.1.9 - Qualyfyng de Mònaco

Ens ha aparegut un petit problema a l'hora de gestionar les dades que guardem cada minut en les sessions de Qualificació.

(RECORDATORI DEL PUNT 4.4) Les sessions de Qualificació tenen el següent format :

Q1 : 20 minuts (s'elimina els 6 últims).

Descans 7 minuts

Q2 : 15 minuts (s'elimina els 6 últims).

Descans 8 minuts

Q3 : 10 minuts. Participen els 10 millors.

La qualificació dura 1 hora, amb la suma de les sessions i descansos.

Per tant, a l'hora de guardar dades de les diferents s'ha establert que els minuts es guardaran :

Q1 : 60 minuts a 40 minuts.

Q2 : 33 minuts a 18 minuts.

Q3 : 10 minuts a 0 minuts.

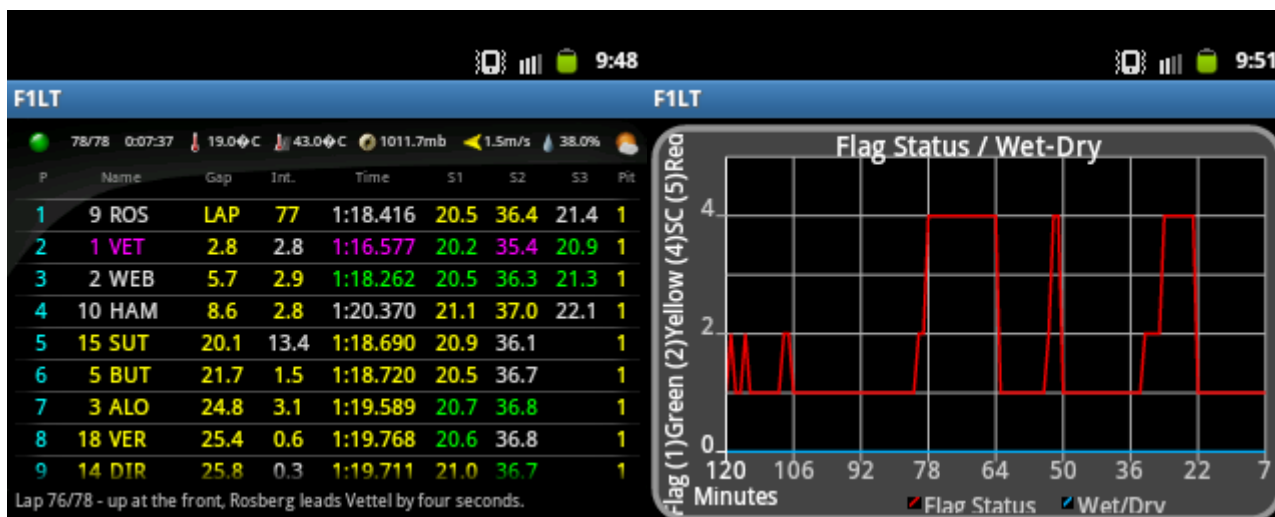
Però el problema apareix quan no es canvia la sessió (Q1 a Q2, Q2 a Q3) fins que la sessió ja s'ha iniciat mentre el temps de la nova sessió ja està vigent. Això fa que es guardi un valor petit llunya que amb el pas de la sessió es corregirà.

És a dir, aquest error farà escriure el valor 20 abans que s'iniciï la Q1, per exemple. En la imatge superior esquerra es pot observar (5.1.8).

A les imatges de sota es poden veure les diferents situacions que es van produir durant la cursa a Mònaco, com la Bandera Vermella (Es para la cursa per arreglar els desperfectes de l'accident entre Chilton i Maldonado), Safety Car per reprendre la cursa (el temps de volta es 25 minuts ja que és el temps que va estar parada la cursa), i la gràfica amb l'estat de les banderes durant el transcurs de la cursa.

F1LT 8:55										F1LT 9:03									
46/78 0:52:38 19.0°C 43.0°C 1012.0mb 2.1m/s 37.0%										SC 47/78 0:52:03 19.0°C 43.0°C 1011.9mb 2.4m/s 38.0%									
P	Name	Gap	Int.	Time	S1	S2	S3	Pit		P	Name	Gap	Int.	Time	S1	S2	S3	Pit	
1	9 N. Rosberg	LA	45	1:22.8	22.	62.		1		1	1 VET	LAP	46	25:18.412	22.1	66.0		1	
		P		85	1	0				2	2 WEB	1.7	1.7	25:19.348	22.1	66.5		1	
2	1 S. Vettel	2.4	2.4	1:23.2	22.	66.		1		3	7 RAI	4.1	2.4	25:20.667	22.1	70.1		1	
				62	1	0				4	6 PER	6.0	1.8	25:21.293	22.0	72.5		1	
3	2 M. Webber	3.2	0.8	1:23.1	22.	66.		1		5	3 ALO	7.0	0.9	25:22.622	21.9	71.4		1	
				29	1	5				6	15 SUT	7.9	0.9	25:21.999	21.8	75.7		1	
4	1 L. Hamilton	3.6	0.4	1:23.2	21.	68.		1		7	5 BUT	7.9	0.9	25:22.801	21.8	74.8		1	
				69	9	1				8	14 DIR	10.5	2.6	25:23.136	22.1	81.3		1	
5	7 K. Raikkonen	4.3	0.6	1:23.1	22.	70.		1		9	18 VER	11.3	0.8	25:24.590	21.8	80.1		1	
				39	1	1				Red flag - the race will, of course, be restarted behind the safety car. The tyre w									
Red flag - the race will resume at 15:35 local time.																			

5.1.10 - Bandera Vermella a la cursa de Mònaco 5.1.11 - Safety Car a la cursa de Mònaco



5.1.12 - Cursa de Mònaco

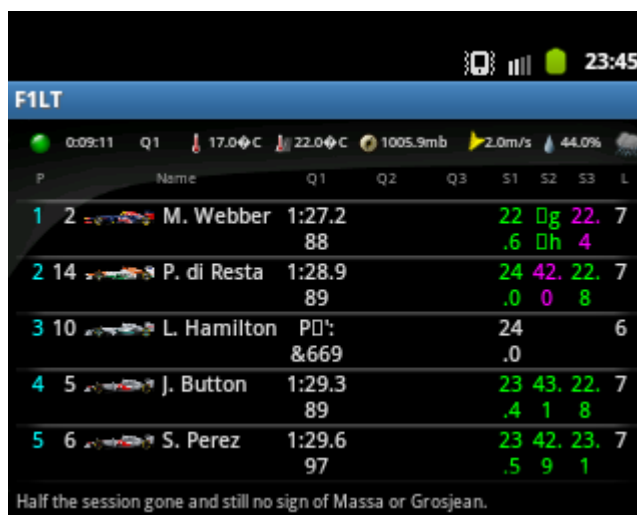
5.1.13 - Banderes durant la cursa de Mònaco

Observem en la última fila d'imatges (5.1.12 i 5.1.13) com la captura i reproducció és correcta i els gràfics generats de les banderes durant tota la cursa.

El estat 1 de les banderes (Flag) vol dir bandera verda (pista lliure), bandera groga – estat 2 (accident, precaució), estat 4 – Safety Car / Cotxe de Seguretat a pista. Per altra banda, en la línia de Wet/Dry veiem com la cursa ha sigut en sec en tot moment.

GP de Canadà

En aquest GP també obtenim correctament les dades, de totes formes es pot observar que la recepció de les dades no és infal·lible ni amb Wi-Fi ni amb Paquets de Dades. Com es mostra en les properes dues imatges, aquestes produeixen ocasionalment fallades de descryptació al no rebre correctament la trama. De totes formes, aquest tipus d'error són pràcticament inapreciables ja que sol ocórrer com a màxim, 2 o 3 cops per cursa (de 2h). No implica un gran trasbals en les dades.



5.1.14 - Q1 amb dades incorrectes

Observem en la imatge (5.1.14) com el S2 de Webber surt distorsionat, així com la volta de Lewis Hamilton.

5.2 Anàlisi : Captura WI-FI / Paquets de Dades

Per altra banda, també hi ha una diferència important entre capturar amb Wi-Fi i Paquets de Dades. En la primera opció, hem aconseguit capturar un màxim de 45 minuts. Però l'habitual és que la recepció continuada duri entre 20 i 30 minuts, podent ser inclús menys, arribat el moment el temps segueix baixant però no es reben més dades. Això fa necessari el tancar l'aplicació i tornar a iniciar amb una ruta diferent per no sobre-escriure les dades ja gravades.

S'ha provat per Wi-Fi tant a casa, com al Wi-Fi de la UB amb els mateixos resultats de talls. També s'ha mirat detalladament el codi i el funcionament és correcte. Inclús amb dos dispositius diferents s'ha provat, i els talls succeïxen en els dos.

També es va provar diferents configuracions avançades del Wi-Fi del dispositius, amb el mateix resultat en la captura de dades, segueix fallant.

En canvi, la captura amb paquets de dades sempre és infal·lible i continua des de l'inici de la sessió fins al final. S'ha provat també canviant de situació, anant en cotxe... i evidentment es talla en algun moment. Però no és necessari reiniciar la connexió com si cal amb Wi-Fi.

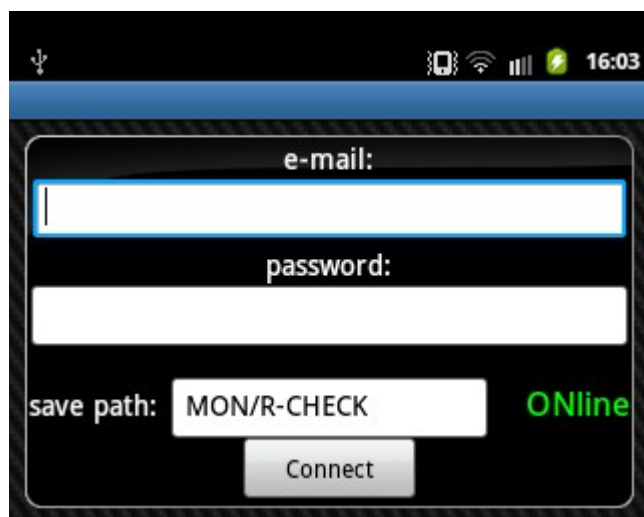
No sabem molt bé a què deu ser degut. Pot ser a algun tipus d'Inactivitat per Wi-Fi que no passa per Paquets de Dades. També podria ser degut a que la Xarxa de Dades sigui més estable que no pas la recepció per Wi-Fi.

6. Manual d'Usuari

Aquesta és una petita guia per l'usuari, per tal de guiar-lo a través de la seva experiència per l'Aplicació F1LT.

Per obtenir dades de l'aplicació necessitarem estar registrats a la pàgina F1.com , el registre és gratuït.

Per tant, al entrar a l'aplicació veurem com ens demana el e-mail de registre i la contrasenya. Apart d'això, si estem connectats a Internet ens demanarà la ruta on volem salvar, a la targeta SD, les dades que generarà la sessió. Per exemple si la ruta a guardar és MON/Race, les dades se'ns guardaran a la targeta SD amb ruta : /F1LT/F1/MON/RACE.

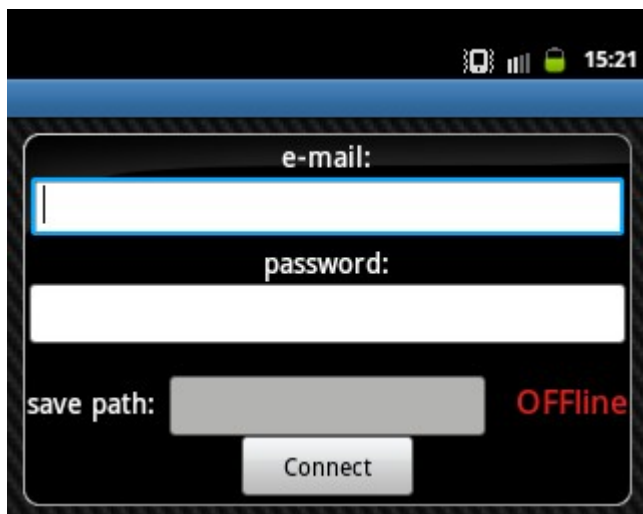


6.1 - Login Online

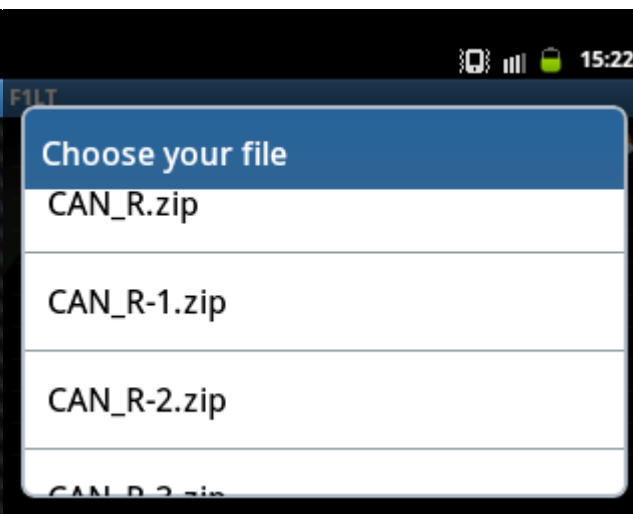
MODE OFFLINE

En el cas que volem accedir a una sessió gravada, hem de desconnectar d'Internet (tant de Wifi com Paquets de Dades). Un cop entrades les dades de registre, llavors ens apareixerà un missatge confirmant que volem carregar una sessió ja disputada. Al respondre afirmativament, se'ns genera una finestra on escollirem l'arxiu comprimit de la sessió que volem reproduir.

IMPORTANT – 1 : S'han de comprimir les sessions a zip (els arxius interiors de la carpeta guardada) per tal de poder-la reproduir. Hi han dues opcions, comprimir-ho manualment des de una finestra de l'explorador o comprimir-ho dins l'aplicació com explicarem posteriorment.



6.2 - Login Offline



6.3 - Triar sessió diferida

Un cop seleccionat la sessió a reproduir, trigarà uns minuts (en el cas d'una cursa, pot arribar a trigar una mica més de 10 minuts) a carregar les dades, i immediatament després ja es reproduirà la sessió desitjada. Un cop som dins la sessió reproduïda o en directe, les funcionalitats son les mateixes.

LIVE TIMING

Per defecte, ens apareix la pantalla d'informació genèrica de cada pilot però hi ha diferències entre el què es mostra en una sessió d'entrenaments lliures, qualificació i cursa.

El què no varia és la informació de la part superior en què indica si hi ha bandera verda (tot OK), bandera groga (accident en algun punt del circuit), bandera vermella (sessió aturada), SC (Safety Car – Cotxe de Seguretat). Així com també mostra el temps per finalitzar la sessió (en cas de ser cursa, també marca les voltes completades sobre les voltes totals). Mostra també la temperatura de l'aire, la temperatura de l'asfalt, la pressió atmosfèrica, la direcció del vent així com la seva direcció, la humitat i, la icona última, mostra si la sessió és en sec (icona amb sol i núvol) o si plou (pluja).

P	Name	Best	Gap	S1	S2	S3	L
1	1 S. Vettel	1:23.04		25.			19
		7		1			
2	9 N. Rosberg	1:23.68	0.633	23.	26.	33.	16
		0		6	8	1	
3	12 E. Gutierrez	1:23.95	0.910	23.	26.	33.	28
		7		9	8	0	
4	8 R. Grosjean	1:24.27	1.230				15
		7					
5	10 L. Hamilton	1:25.05	2.007	23.	27.	33.	16
		4		8	4	5	

Hulkenberg does the same.

6.4 - Entrenaments lliures

En els **entrenaments lliures** :

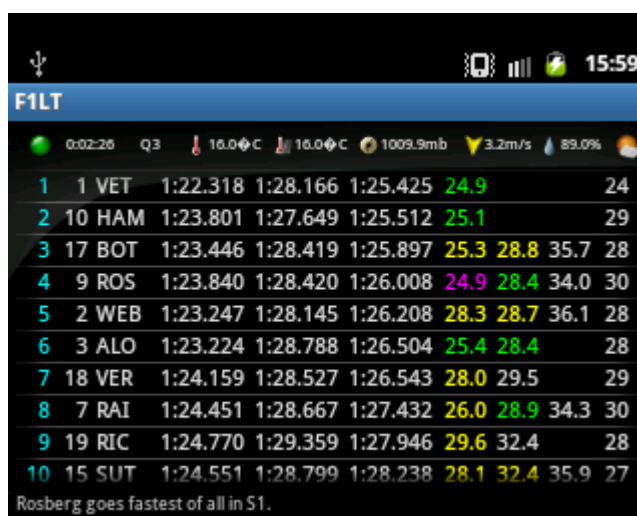
- Mostra els pilots ordenats segons la seva millor volta ràpida
- El dorsal del pilot (a l'esquerra del nom) és mostrat en blanc si està a pista, i en vermell si està aturat al box.
- Mostra el seu millor temps de volta
- Mostra la diferència respecte al millor temps.
- Mostra els tres parcials actuals. En cas d'estar aturat a boxes, mostra els seus tres millors parcials realitzats anteriorment.
- Voltes totals realitzades.
- Mostra al final de tot el 107% del millor temps (temps a partir del qual, en qualificació i si és superior, queda exclòs de participar en la cursa)

En el cas dels parcials, hi han diferents colors representatius :

- **Verd** : Millor parcial del pilot
- **Magenta** : Millor parcial absolut (de tots els pilots)
- **Blanc** : Ultim parcial marcat per el pilot (en cas de no ser ni verd, ni magenta)
- **Groc** : Parcial marcat anteriorment.
(en la propera imatge 6.5 s'observa correctament els diferents colors)

En la **qualificació** :

- Mostra els pilots ordenats segons la seva millor volta ràpida
- El dorsal del pilot (a l'esquerra del nom) és mostrat en blanc si està a pista, i en vermell si està aturat al box.
- Mostra el seu millor temps de volta
- Mostra els millors temps a cada part de la qualificació (Q1, Q2, Q3).
- Informació dels parcials.
- Voltes totals realitzades.
- Mostra al final de tot el 107% del millor temps (temps a partir del qual, en qualificació i si és superior, queda exclòs de participar en la cursa)



F1LT						
0:02:26 Q3 16.0°C 16.0°C 1009.9mb 3.2m/s 89.0%						
1	1 VET	1:22.318	1:28.166	1:25.425	24.9	24
2	10 HAM	1:23.801	1:27.649	1:25.512	25.1	29
3	17 BOT	1:23.446	1:28.419	1:25.897	25.3 28.8	35.7 28
4	9 ROS	1:23.840	1:28.420	1:26.008	24.9 28.4	34.0 30
5	2 WEB	1:23.247	1:28.145	1:26.208	28.3 28.7	36.1 28
6	3 ALO	1:23.224	1:28.788	1:26.504	25.4 28.4	28
7	18 VER	1:24.159	1:28.527	1:26.543	28.0	29.5 29
8	7 RAI	1:24.451	1:28.667	1:27.432	26.0 28.9	34.3 30
9	19 RIC	1:24.770	1:29.359	1:27.946	29.6	32.4 28
10	15 SUT	1:24.551	1:28.799	1:28.238	28.1 32.4	35.9 27

Rosberg goes fastest of all in S1.

6.5 - Qualificació

En la **cursa** :

- Mostra els pilots ordenats segons la seva posició en cursa
- El dorsal del pilot.
- Mostra la volta del líder i la diferència amb el primer. En la segona columna, marca la diferència amb el pilot que té davant.
- Mostra el seu darrer temps de volta.
- Informació dels parcials.
- Cops que ha entrat a boxes.
 - Quan entra a box, mostra “IN PIT” a Parcials. Un cop ha sortit de boxes, en la primera volta després de parar es mostra “OUT” i el temps que ha trigat per boxes a la columna de parcial S3 (mostrant també les parades anteriors, si les ha fet, movent-les cap a S1. És a dir, en la primera parada es mostra a S3, a la segona parada es mostra a S3, movent la primera a S2. I així tants cops com parades faci, amb l'inconvenient que màxim mostrarà només les 3 últimes).

Com observem a la imatge de la dreta (6.7), apareixen les dues aturades a Boxes de HAM (Hamilton).

F1LT 12:58									
P	Name	Gap	Int.	Time	S1	S2	S3	Pit	
1	1 VET	LAP	39	1:17.795	21.6	24.9	31.1	1	
2	10 HAM	16.2	16.2	1:19.116	21.7	25.0		1	
3	2 WEB	23.8	7.6	1:17.934	21.7	24.9		1	
4	3 ALO	24.5	0.6	1:17.844	21.8	24.8		1	
5	9 ROS	51.0	26.5	1:18.652	21.8			2	
6	18 VER	58.8	7.7	1:19.677	22.2			1	
7	14 DIR	59.9	1.1	1:19.506	22.4				
8	8 GRO	69.6	9.7	1:20.242	22.6	25.7	31.8		
9	4 MAS	76.7	7.0	1:19.711	22.3	25.6	31.6	1	

Lap 39/70 - Di Resta, Grosjean, Bianchi and Chilton still to pit.

6.6 - Codi de colors

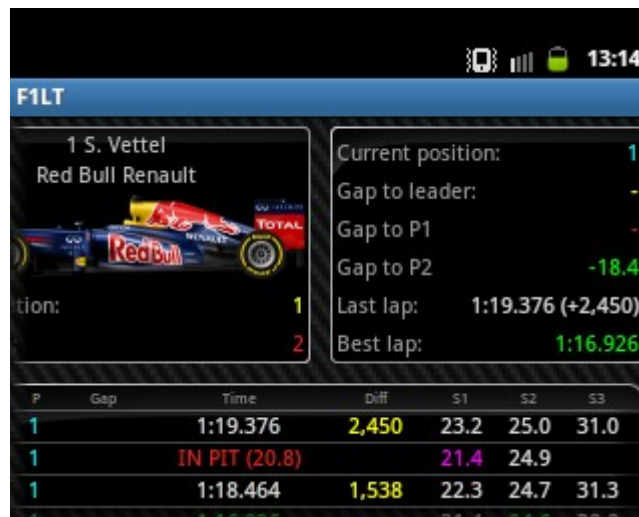
F1LT 13:10									
P	Name	Gap	Int.	Time	S1	S2	S3	Pit	
1	1 VET	LAP	48	1:18.464	21.4	24.9		1	
2	10 HAM	30.4	30.4	OUT		20.8	20.6	2	
3	3 ALO	39.9	9.4	1:19.764	23.0	25.4	31.2	2	
4	2 WEB	44.1	4.2	1:17.907	21.6	24.9	31.2	2	
5	9 ROS	58.5	13.8	1:20.921	22.2	26.1		2	
6	18 VER	70.4	11.8	1:19.526	22.1	25.5		1	
7	14 DIR	72.6	2.2	1:19.898	22.2	25.6			
8	15 SUT	1L	23.8	1:19.983	22.4			2	
9	7 RAI	1L	4.4	1:19.453	22.3			1	

Lap 49/70 - Hamilton pits from second.

6.7 - Codi de colors i aturades a box

A part de la vista comentada, també hi han altres maneres de recopilar informació.

Clicant sobre el nom del pilot, entrarà a la vista que se'ns mostra les voltes per el pilot seleccionat.

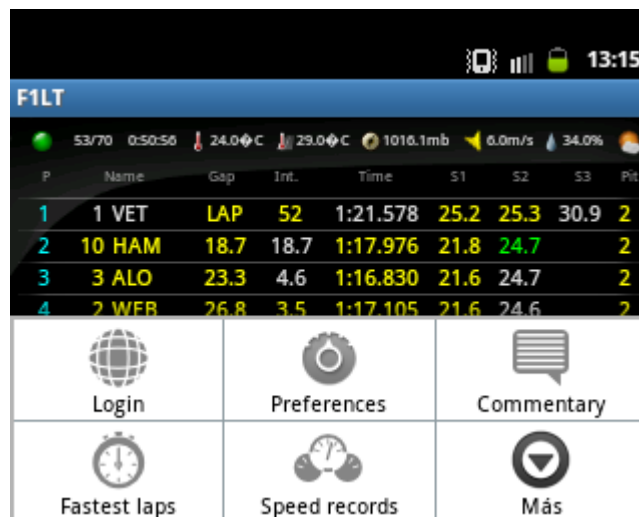


P	Gap	Time	Diff	S1	S2	S3
1		1:19.376	2,450	23.2	25.0	31.0
1	IN PIT (20.8)			21.4	24.9	
1		1:18.464	1,538	22.3	24.7	31.3







6.8 - Vista del pilot seleccionat

ACTIVITIES

També podem accedir a través del Menú a diferents opcions.



P	Name	Gap	Int.	Time	S1	S2	S3	Pit
1	1 VET	LAP	52	1:21.578	25.2	25.3	30.9	2
2	10 HAM	18.7	18.7	1:17.976	21.8	24.7		2
3	3 ALO	23.3	4.6	1:16.830	21.6	24.7		2
4	2 WEB	26.8	3.5	1:17.105	21.6	24.6		2

 Login	 Preferences	 Commentary
 Fastest laps	 Speed records	 Más

6.9 - Menú principal

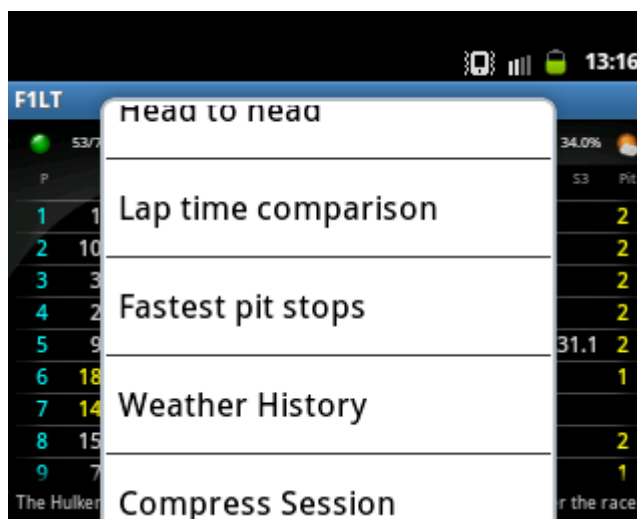
1. Login, per tornar a accedir.
2. Preferències, seleccionem si volem veure els noms normals o abreujats, imatge del cotxe o no, comentaris a sota de la finestra o no.
3. Comentaris de la sessió
4. Fastest Laps, Volta ràpida de cada pilot.
5. Speed Records, velocitats màximes registrades a cada parcial (punt de referència que separa cada parcial durant la volta, no la velocitat màxima durant aquest parcial).

6. MÉS

- Head to Head, comparació de voltes entre pilots
- Lap Time Comparison, comparació de voltes entre 4 pilots
- Fastest Pit Stops, aturades més ràpides
- Weather History
 - Temperatura al Aire i Temperatura al Asfalt.
 - Humitat a l'Aire
 - Pressió a l'Aire
 - Velocitat del Vent
 - Direcció del Vent
 - Estatus de les Banderes / Sec-Mullat
- **COMPRESS SESSION.** Un cop acabada la Sessió, al pulsar en aquesta opció es comprimeix la sessió en format .zip a la carpeta F1LT/F1/ZIP on després podrà ser accedida per a reproduir.

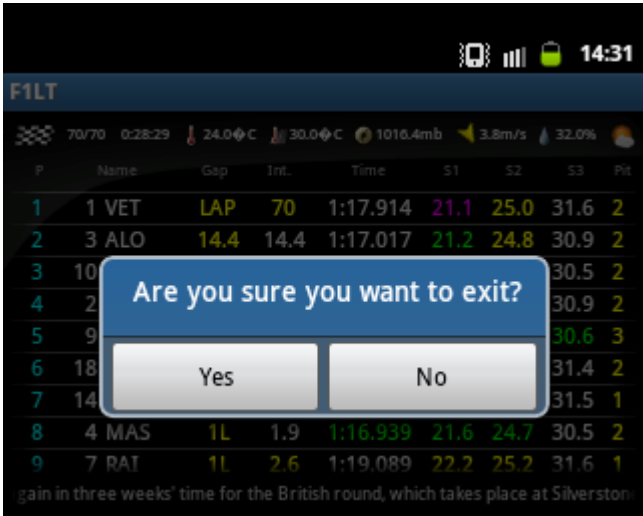
En cas que no guardem la sessió en finalitzar aquesta, s'haurà de guardar a mà des de l'explorador, seleccionant tots els arxius dins de la carpeta i comprimint-ho. Posteriorment, s'haurà de col·locar el ZIP resultant a la carpeta F1LT/F1/ZIP/

És **IMPORTANT** també esborrar regularment les carpetes generades per cada sessió ja que les SD Card del mòbil tenen un límit de 65.600 fitxers, el qual s'hi arriba capturant 2 GP's. Per tant, no és recomanable tenir més d'1 GP no comprimit ja que podríem perdre dades capturades per ja tenir la targeta al límit d'aquesta quantitat de fitxers.



6.10 - Menú adicional

Per últim, per abandonar l'aplicació polsem la tecla de retorn i ens apareix una finestra per confirmar que volem abandonar l'aplicació. Es suprimirà els arxius temporal al sortir de l'aplicació.



6.11 - Sortir

7. Conclusions

En aquest Projecte Final de Grau he après a treballar en l'entorn Android – llenguatge Java. Encara que Java s'havia usat de forma recurrent durant ETIS i també al Grau, hem après a integrar els coneixements en l'entorn Android. Com s'havia indicat al inici de la memòria, aquest era un punt important a l'hora de triar aquesta tecnologia per triar fer aquest Projecte.

L'entorn Android no m'ha semblat gaire complicat en la seva globalitat, però si hi han alguns aspectes que han sigut costosos a l'hora d'implementar-ho.

Ha sigut costós el fet de passar dades d'un fil secundari al flux principal de dades, ja que era un concepte un pèl difícil d'entendre si no s'havia programat en Android.

Per altra banda, també ha sigut costós el fet d'entendre un codi que no era propi i que era molt extens. Això afegit a que el protocol de desencryptació de dades i gestió d'elles fos propi del Live Timing, han afegit dificultat a processar-ho.

El reball individual ha sigut un aspecte important en aquest projecte. Durant els Estudis Universitaris estem acostumats a cooperar entre els alumnes a l'hora d'elaborar les pràctiques. Però en aquest projecte final, hem d'enfrontar-nos sols a tots aquests reptes de aprendre el necessari i implementar-ho. Per tant, una bona utilització del temps i la motivació ha sigut essencial a l'hora d'assolir l'èxit en aquest projecte.

També ha sigut un repte el fer una memòria tant extensa i amb tants aspectes importants a ser comentats. Durant els estudis, no s'havien fet documentacions tant extenses, i en aquest sentit hem hagut d'organitzar bé el document per tal de plasmar de la millor manera possible tot el nostre treball realitzat.

Per últim, i tan important que els anteriors. El tutor d'aquest projecte, el Lluís Garrido, des de que havia fet classes amb ell de diferents matèries vaig tenir clar que m'agradaria fer el projecte amb ell, ja que ha sigut el millor professor que he tingut durant la carrera. Sempre s'ha ofert a solucionar qualsevol tipus de dubte, i en cas de dificultat ha sigut vital a l'hora d'incentivar i motivar a solucionar-los. La seva ajuda i col·laboració ha sigut imprescindible.

7.1 Dificultats

- La primera dificultat que ens vàrem trobar va ser entendre el projecte sense documentació. Al haver de crear Logs on imprimir un rastre d'on passa l'execució del codi, per un cop sabíem per on passava cada una de les funcionalitats, entendre pròpiament què estava fent cada funció. Un cop acabat, entendre el lloc on havíem de modificar el codi per afegir les noves funcionalitats.
- Les primers 4 curses del campionat han anat agrupades de dos en dos, però separades en 3 setmanes.

Austràlia 17 de Març, Malàisia 24 de Març.

Xina 14 d'Abril, Bahrain 21 d'Abril.

Barcelona 12 de Maig.

Això feia que tot el que no estigués fet i funcionant un cop acabat Malàisia, no es podria tornar a provar fins al cap de 3 setmanes. A més, també hi hem d'afegir una dificultat extra, en què les tres primeres curses han sigut disputades a la matinada, hora nostra. Per tant, no era possible enregistrar o provar amb totes les sessions ja que estar despert tota la nit durant 3 dies, tenint classe el dia posterior era una quimera.

- Relacionat amb el punt anterior està el fet que no fos possible afinar la sincronització de les sessions per a la seva reproducció en temps real.

Fins que no vàrem trobar la manera correcta, després de provar algunes altres, d'obtenir l'instant del temps de captura del paquet. Fins aquest punt, no vàrem poder sincronitzar les sessions al temps real, per tant algunes sessions les reproduïm desincronitzades de la realitat.

Sincronització de la Sessió

- Sistema de fitxers FAT32 a la targeta SD. Aquest ha sigut un problema que ens hem trobat. Degut al sistema de fitxers que accepta un nombre determinat de fitxers dins una carpeta, en la captura de dades es deixava de capturar paquets quan s'arribava a aquest límit. Al localitzar-ho, varem aplicar la compressió en les sessions per pal·liar aquest dèficit.
- Wi-Fi / Paquets de Dades (comentat al Capítol 5.2)

No ens estendrem més en aquest tema, però ha sigut un problema no resolt el saber perquè la captura per xarxa Wi-Fi se'ns atura abans de la mitja hora mentre amb Paquets de Dades funciona perfectament d'una tirada. Hem mirat tot el codi i provat diferents solucions que no ens han reportat cap millora.

7.2 Discussió crítica

Estic força content amb tot el treball realitzat, però m'he quedat amb ganes de implementar més funcionalitats a l'aplicació.

El fet que l'aplicació no tingués documentació ha fet que el procés d'aprenentatge fos llarg i costós, també influenciat en què les primeres curses habitualment anaven separades cada 3 setmanes. Per tant, s'ha trigat un mes i mig des de l'inici del Campionat en poder implementar la gravació i reproducció de les dades.

També he de fer autocrítica en què vàrem estar durant 2 o 3 setmanes en què no guardar el Key Frame, feia que la reproducció es produís distorsionada i les dades no es poguessin llegir. Això va fer endarrerir el procés de reproducció.

M'hagués agradat tocar bastant més temps la part d'interfície i noves implementacions d'activitats, ja que la part de programació útil d'aquest projecte s'ha acabat fent els últims dos mesos.

D'aquí que m'hagi quedat amb la sensació que al final el temps de programació en el projecte ha sigut més aviat curt.

7.3 Línies Futures

Tal com s'ha indicat en la introducció, la tria d'aquesta aplicació ha sigut per donar-li un us quotidià, i que no es quedi tal com està un cop s'hagi entregat el projecte.

De cara a un futur, seria interessant **millorar la captura de dades** de forma **més eficient** ja que la compressió i descompressió és força costós per al dispositiu de cara a consum de recursos i temps que dura.

Seria interessant **poder entrar** a la part de **Diferit estant connectar Internet** del dispositiu mòbil, és una part incòmoda per l'usuari haver de desconnectar-se.

Per altra banda, també seria útil implementar més gràfics referents als pilots. Per exemple, implementar un **gràfic en cursa** que digui la **evolució de posicions dels pilots cada volta**.

Una implementació addicional de cara a consultar les dades sense haver d'estar a l'aplicació seria bolcar de forma ordenada totes les **dades generades a un arxiu PDF**, per exemple. Ordenar els temps de voltes per els pilots, així com l'evolució de les Dades meteorològiques i gràfiques.

També seria interessant poder introduir el tipus de **pneumàtics que porten els pilots**, i les **voltes** que porta cada pneumàtic durant el cap de setmana. De totes formes, aquesta funcionalitat seria complicada ja que aquestes dades s'haurien d'introduir manualment per l'usuari ja que no són accessibles via el Live Timing.

Per últim, afegir una Activitat que mostrés **comparació de volta entre pilots** en un marge de **voltes concretes**. Per exemple, des de la primera aturada fins a la segona. O inclús, **per uns pneumàtics determinats** fer una comparativa directa dels pilots que l'han utilitzat.

8. Bibliografia

- Llibre “Learning Android” by Marko Gargenta. Editat: O'Reilly Media
- Fòrum especialitzats de consulta sobre codi Java-Android (stackoverflow.com/ , <http://www.android.es/foro/>)
- Llibreria de suport en la creació de gràfics (<http://androidplot.com/>)
- Tutorials sobre Android (<http://elbauldelprogramador.com/> , <http://www.elandroidelibre.com/>)
- API Android (<http://developer.android.com/>)
- Guia per comprimir / descomprimir arxius (<http://www.jondev.net/>)

ANNEX

Vocabulari específic

Live Timing : Aplicació WEB ([Applet Java](#)) que permet la recepció a temps real de les dades generades durant cada una de les sessions disputades de la Fórmula 1. Aquesta aplicació web ve proporcionada per la F1.com (pàgina oficial de la Fórmula 1), i per [ACCEDIR-HI](#) cal estar registrat (és gratuït).

Applet Java : És un component d'una aplicació que s'executa en el context d'un altre programa (en el cas que ens referim del Live Timing, serà en el context del nostre navegador web). Permet l'execució de manera senzilla de l'aplicació en el navegador per a l'usuari. Els applets son multi-plataforma, per tant permet executar-ho en entorns Windows, Linux, Mac, etc. Només amb el requeriment que la màquina tingui la "Java Virtual Machine", contingut a les edicions de Java Standard Edition.

Parser - Parsejar : Motor de processament de llenguatge aplicat de qualsevol llenguatge (en aquest cas, el tipus en què s'envien les dades). Es refereix a la part del programa en què obté les dades fent el processament necessari al tipus desitjat (llegir, guardar, operar, etc).

Log : Arxiu que porta un registre de les operacions o esdeveniments que es duen a terme en l'aplicació.

Key Frame : Nombre únic que genera la pàgina F1.com. Serveix per desencripar les dades que s'envien per la sessió actual. Aquest nombre és únic i sol tenir aquest format : -1148789007

Threads : Fils d'execució paral·lels dins d'un procés (Aplicació).

Ping : Utilitat que serveix per comprovar la comunicació en una xarxa de comunicacions TCP/IP amb el servidor web o host. Aquesta operació es fa mitjançant l'enviament de sol·licitud i resposta de paquets (en aquest cas, bytes).

Handler : Pont que permet traslladar informació d'un fil secundari al flux principal d'execució de l'aplicació.