Universidad Zaragoza

1542

# Proyecto Fin de Carrera

## Sistema de procesamiento de data streams


Autor

Óscar Cardiel Enguita

Director

Sergio Ilarri Artigas

Escuela de Ingeniería y Arquitectura


Año 2014

# RESUMEN

Durante años, los sistemas de bases de datos tradicionales han resultado suficientes para procesar la cantidad de datos que se generaban en el día a día. Sin embargo estos datos siempre han sido fundamentalmente estáticos y una vez almacenados ya no suele ser necesario modificar.

Hoy en día, con el auge de nuevas tecnologías de comunicación ha aparecido la necesidad de recibir y procesar todo tipo de datos que modifican su valor constantemente. Un ejemplo de esto sería el procesamiento de datos que provienen de sensores de muy diverso tipo, como podrían ser los GPS (*Global Positioning System*). Este tipo de información se escapa del ámbito de los gestores tradicionales y hace necesario el desarrollo de una nueva tecnología que procese los datos de estos sensores conforme van llegando, es decir, en tiempo real.

Aprovechando diversas políticas de actualización, el coste energético de estos envíos también se puede minimizar, ya que aplicando diversas estrategias podemos reducir el número de datos enviados, acompañando información adicional al valor del sensor.

Bajo esta serie de premisas el grupo SID de la Universidad de Zaragoza elaboró un procesador de consultas para datos provenientes de flujos de datos (o *data streams*). En este prototipo, los sensores acompañan la información que envían con una función que permite calcular el valor durante un periodo de tiempo en el cual esta función tiene validez.

Al no recibir la totalidad de la información, sino funciones de predicción validas durante un periodo de tiempo, se hace necesario una gestión de la información totalmente diferente a los gestores tradicionales. Para ello, este procesador permite la realización de dos tipos de consultas: Window Join y Value. La primera permite comparar el valor entre dos clases de sensores durante una ventana de tiempo determinada en la consulta y el segundo permite comprobar el valor de una clase de sensores. Este procesador de consultas se trataba de un programa que funcionaba en línea de comandos y que tenía toda su configuración codificada en código fuente, y usaba un simulador muy sencillo y poco versátil.

Con todo ello, se ha realizado este proyecto que intenta proveer de una potente herramienta de simulación para poder realizar experimentos con el procesador de consultas y además proveerle de una interfaz gráfica que facilite el trabajo con el programa, además de incluirle algunas operaciones adicionales. Con este proyecto se espera que con las modificaciones y adaptaciones pertinentes contribuya a la investigación en el ámbito de los flujos de datos del grupo SID.

# ÍNDICE

RESUMEN ..... 3

ÍNDICE ..... 5

TABLA DE FIGURAS ..... 11

1 INTRODUCCIÓN ..... 15

1.1 Contexto y motivación ..... 15

1.2 Prototipo anterior del SID ..... 16

1.3 Objetivos ..... 18

1.4 Tecnologías utilizadas ..... 19

1.5 Fases del proyecto ..... 19

1.5.1 Búsqueda de Información ..... 19

1.5.2 Análisis, Diseño e Implementación de un sistema de simulación ..... 19

1.5.3 Familiarización con el prototipo anterior. ..... 19

1.5.4 Fusión del sistema de simulación con el prototipo ..... 20

1.5.5 Análisis y Diseño de la Interfaz ..... 20

1.5.6 Implementación de la interfaz ..... 20

1.5.7 Pruebas globales ..... 20

1.6 Estructura de la memoria ..... 20

1.7 Planificación del proyecto ..... 21

1.7.1 Listado de tareas ..... 21

1.7.2 Control de tiempos ..... 23

2 CONTEXTO TECNOLÓGICO ..... 25

2.1 Java ..... 25

2.2 Eclipse ..... 25

2.3 Git ..... 26

2.4 Trazas GPS ..... 26

2.5 Sistema de procesamiento de *data streams* ..... 27

2.6 Hadoop y Hadoop Streaming ..... 27

3 PLANTEAMIENTO Y DESARROLLO DEL SISTEMA ..... 29

3.1 Análisis de requisitos ..... 29

3.2 Diseño e implementación del sistema ..... 31

3.2.1 Estrategias de movilidad ..... 35

3.2.2 Sistema de carga y almacenamiento de parámetros ..... 35

3.2.4 Interfaz de usuario ..... 36

3.2.5 Interpolación para obtener la función de predicción ..... 38

3.2.6 Conversión de coordenadas ..... 39

3.2.7 Alarmas ..... 39

3.2.8 Snapshots ..... 40

3.2.9 Estadísticas ..... 40

4 PRUEBAS Y PROBLEMAS ENCONTRADOS ..... 43

4.1 Pruebas realizadas ..... 43

4.2 Problemas encontrados ..... 44

5 CONCLUSIONES ..... 45

5.1 Resultados obtenidos ..... 45

5.2 Futuras ampliaciones ..... 46

5.3 Valoración personal ..... 47

ANEXO A: Manual de Usuario ..... 51

A1 Introducción ..... 51

A2 Inicio de la aplicación ..... 51

A2.1 Compilación y ejecución ..... 51

A2.2 Pantalla Inicial ..... 52

A3 Combinación de teclas de acceso rápido. ..... 60

A4 Simulación ..... 61

A4.1 Realizar una simulación GPS ..... 61

A4.2 Realizar una simulación de cotizaciones ..... 64

A4.3 Realizar una simulación genérica ..... 67

A4.4 Realizar una simulación de una red de sensores ..... 70

A4.5 Guardar simulación de sensores ..... 71

A4.6 Importar ficheros .sim ..... 72

A4.7 Importar Trazas GPS ..... 72

A5 Procesamiento de *data streams* ..... 74

A5.1 Crear una consulta ..... 74

A5.2 Visualizar una consulta del sistema ..... 75

A5.3 Visualizar una consulta desde fichero ..... 78

A5.4 Cancelar consulta ..... 78

A5.5 Crear una alarma ..... 78

A5.6 Cancelar una alarma ..... 79

A5.7 Funciones de Agregación ..... 80

A5.8 Snapshots ..... 81
A6 Configuración general de la aplicación ..... 82
A6.1 Predicción ..... 82
A6.2 Buffer ..... 82
A6.3 Simulación ..... 83
A6.4 R-tree ..... 84
A7 Tutorial ..... 85
A7.1 Manejo básico de la aplicación ..... 85
A7.2 Ejemplo con escenario real ..... 91
ANEXO B: Formato de Ficheros ..... 94
B1 Ficheros GPX ..... 94
B2 Ficheros GeoLife ..... 95
B3 Ficheros CSV ..... 96
B4 Ficheros SIM ..... 97
B6 Ficheros DSM ..... 98
B2 Fichero de configuración ..... 100
ANEXO C: Análisis del sistema ..... 102
C1 Requisitos de la Aplicación ..... 102
C2 Casos de Uso ..... 109
C2.1 Casos de uso de simulación ..... 109
C2.2 Casos de uso de procesamiento ..... 111
C3 Prototipado de Ventanas ..... 112
C4 Navegación de Ventanas ..... 123

ANEXO D: Diseño e implementación del sistema ..... 126

D1 Paquete DSMS ..... 126

D1.1 Buffers ..... 126

D1.2 Constants ..... 126

D1.3 Constraints ..... 127

D1.3 Delayer ..... 127

D1.4 Geo ..... 127

D1.5 Joins ..... 127

D1.6 Logs ..... 127

D1.7 Maths ..... 128

D1.8 Output ..... 128

D1.9 Policies ..... 128

D1.10 Predictions ..... 129

D1.11 Qp (Query Processor) ..... 129

D1.12 Statics ..... 129

D1.13 Test ..... 129

D1.14 Util ..... 129

D2 Paquete GUI ..... 130

D3 Paquete Communication ..... 132

D3.1 Clase AbstractCommunication ..... 132

D3.1 Clase Message ..... 132

D3.1 Clase PredictionComunnication ..... 132

D4 Paquete GenericModel ..... 133

D5 Paquete Mobility ..... 133

D6 Paquete PriceModel ..... 134

D7 Paquete Sensor ..... 135

D7.1 AbstractPredictionFunction ..... 135

D7.2 AbstractSensor ..... 136

D7.3 GPSPredictionSensor ..... 136

D7.4 PricePredictionSensor ..... 137

D7.5 GenericPredictionSensor ..... 137

D8 Paquete Importer ..... 137

D9 Paquete Simulator ..... 138

D10 Paquete Operations ..... 139

D11 Paquete Triggers ..... 139

D12 Paquete Snapshots ..... 140

D13 Paquete Utils ..... 140

D14 Diagramas de Clases ..... 140

ANEXO E: Estrategias de movilidad ..... 144

E1 Modelo Aleatorio ..... 144

E2 Modelo Manhattan ..... 145

E3 Modelo Gauss-Markov ..... 146

ANEXO F: Modelo de Black-Scholes ..... 148

BIBLIOGRAFÍA ..... 151

# TABLA DE FIGURAS

Figura 1 Arquitectura del prototipo, adaptado del original [1] ..... 17

Figura 2 Distribución de Tiempos ..... 23

Figura 3 Diseño Global de la Aplicación ..... 33

Figura 4 Diseño en alto nivel del Simulador ..... 34

Figura 5 Modelos de Movilidad ..... 35

Figura 6 Apariencia de la interfaz ..... 37

Figura A1 Pantalla Principal ..... 53

Figura A2 Barra de Menú ..... 53

Figura A3 Menú Archivo ..... 54

Figura A4 Menú Consultas ..... 54

Figura A5 Menú Alarmas ..... 55

Figura A6 Menú Simulación ..... 56

Figura A7 Menú Herramientas ..... 56

Figura A8 Menú Ayuda ..... 56

Figura A9 Menú Lateral ..... 57

Figura A10 Menú Contextual Sensores ..... 58

Figura A11 Menú Contextual Alarmas ..... 59

Figura A12 Menú Contextual Alarmas ..... 59

Figura A13 Configuración GPS ..... 61

Figura A14 Configuración Cotizaciones ..... 65

Figura A15 Configuración Genérico ..... 68

Figura A16 Red de Sensores ..... 70

Figura A17 Importar Traza GPS ..... 73

Figura A18 Crear Consulta ..... 74

Figura A19 Crear Restricción ..... 75

Figura A20 Listado de Consultas ..... 76

Figura A21 Consulta Parte Superior ..... 76

Figura A22 Consulta parte media ..... 77

Figura A23 Consulta Resultados ..... 77

Figura A24 Consulta con Alarma ..... 77

Figura A25 Crear Alarma ..... 79

Figura A26 Cancelar Alarma ..... 79

Figura A27 Resultado Agregación ..... 80

Figura A28 Visualización de Snapshot ..... 81

Figura A29 Configuración Predicción ..... 82

Figura A30 Configuración Buffer ..... 83

Figura A31 Configuración Sensor ..... 84

Figura A32 Configuración R-TREE ..... 85

Figura A33 Parámetros de la consulta ..... 86

Figura A34 Configuración del sensor "COCHE" ..... 87

Figura A35 Configuración sensor "PERSONA" ..... 88

Figura A36 Estado de la pantalla de la red ..... 88

Figura A37 Simulación en tiempo real ..... 89

Figura A38 Menú lateral para acceder a la consulta ..... 89

Figura A39 Resultado de Consulta ..... 90

Figura A40 Estadístico ..... 90

Figura A41 Parámetros de transformación ..... 92

Figura A42 Parámetros de la consulta ..... 92

Figura A43 Resultados del ejemplo ..... 93

Figura C1 Diagrama de Navegación ..... 124

Figura D1 Diagrama de clases del simulador ..... 141

Figura D2 Diagrama de clases resumido del sistema de procesamiento con el simulador ..... 142

Figura E1 Simulación aleatoria con probabilidad 50% ..... 145

Figura E2 Simulación con Modelo Manhattan ..... 146

Figura E3 Simulación con modelo Gauss-Markov ..... 147

# 1 INTRODUCCIÓN

El presente documento surge como resultado del trabajo desarrollado para el proyecto final de carrera "Sistema de procesamiento de data streams". Este capítulo en concreto intenta poner en situación al lector para facilitar la compresión de los siguientes capítulos del documento y trata sobre la motivación que tiene el proyecto, cuáles son los objetivos a alcanzar, cuál es el contexto en el que está situado, que tecnologías y sistemas se utilizan para desarrollarlo, que fases se han seguido, su planificación y la estructura de esta memoria.

## 1.1 Contexto y motivación

Los sistemas de bases de datos tradicionales resultan insuficientes cuando hay que procesar grandes volúmenes de datos que cambian constantemente (por ejemplo, localizaciones de objetos móviles, datos provenientes de sensores, información de cotizaciones de acciones, etc.). Frente al almacenamiento persistente de información, en estos casos se proponen técnicas para procesar los datos de forma eficiente conforme llegan (y sin necesidad de darles persistencia previamente), teniendo en cuenta las características y dificultades que supone este flujo continuo en la entrada.

Por ello, frente a los sistemas gestores de bases de datos (SGBD) tradicionales, se han propuesto diversos "sistemas gestores de flujos de datos" (*data stream management systems*" o DSMS) que tienen en cuenta estas consideraciones.

El grupo SID de la Universidad de Zaragoza desarrolló un trabajo previo que permite la gestión de funciones de predicción asociadas a los datos medidos por los sensores (con objeto de disminuir la carga del procesamiento de consultas y las comunicaciones realizadas por los sensores) para un par de operadores de consultas. [1]

En este PFC, basándose en ese desarrollo anterior, se ampliará el prototipo existente con objeto de facilitar su utilización y la realización de pruebas experimentales, se estudiará su ampliación con el soporte de operadores adicionales, y se considerará la posible utilización de *Hadoop Streaming.*

## 1.2 Prototipo anterior del SID

El prototipo sobre el que se sustenta este proyecto se trata de un programa que permite procesar tuplas a las que se acompaña una función de predicción. Esta función permite calcular el valor de un determinado sensor en un instante de tiempo.

Para que esta aproximación funcione, hace falta que los sensores tengan cierta inteligencia para poder calcular una función de predicción que permita determinar los próximos valores que tomará el sensor hasta que éste incumpla alguna política de actualización. Una forma sencilla de obtener estos datos se consigue mediante la simulación de los mismos.

La necesidad de crear un simulador surge de la escasa o nula existencia de simuladores que cumpliesen las necesidades del problema (envío de funciones de predicción bajo una serie de políticas de actualización.) En el prototipito anterior se usaba el programa *IBM Location Transponder* para realizar simulaciones, aunque este proyecto se descontinuó y ya no está accesible en la red.

IBM desarrolló un sistema de simulación de trazas GPS que permitía realizar simulaciones leyendo ficheros de trazas, u obtenerlas directamente. Este proyecto estaba dentro de Alfaworks, una comunidad enfocada a desarrollar software que pueda ser usado en investigación. La capacidad de este simulador era muy limitada (solo permitía pequeñas variaciones aleatorias en la dirección del movimiento, lo cual se ha intentado imitar en el modelo aleatorio de simulación).

También, al tratarse de un proyecto de investigación, toda la configuración y ejecución de experimentos se encontraba codificada en código y se le pasaba unas mínimas opciones por línea de comandos.

Este procesador tenía dos tipos de consultas:

- Window Join: Se trata de una consulta que se aplica a dos clases de sensores (pudiendo ser la misma). A ella se le aporta un operador relacional con un valor asociado y una ventana temporal sobre la que se aplicará la consulta. Es decir, esta consulta es del tipo “la clase location y location tienen una diferencia de valor mayor que 20 durante tres segundos”. Es útil por ejemplo para detectar colisiones entre elementos móviles. [2]
- Value: Se trata de una consulta que se aplica a una clase de sensor y detecta casos del tipo “el valor de la clase location es mayor que 20”. Es útil para conocer valores individuales en una clase de sensores. [1] [2]

Este prototipo contiene toda la infraestructura necesaria para la realización de consultas y el almacenamiento de tuplas.

Estas tuplas son almacenadas en una estructura denominada *R-tree* [3] usada para espacios bidimensionales, minimizando el tiempo de búsqueda a la hora de realizar las consultas.

El siguiente diagrama muestra un diagrama de la arquitectura de este prototipo:

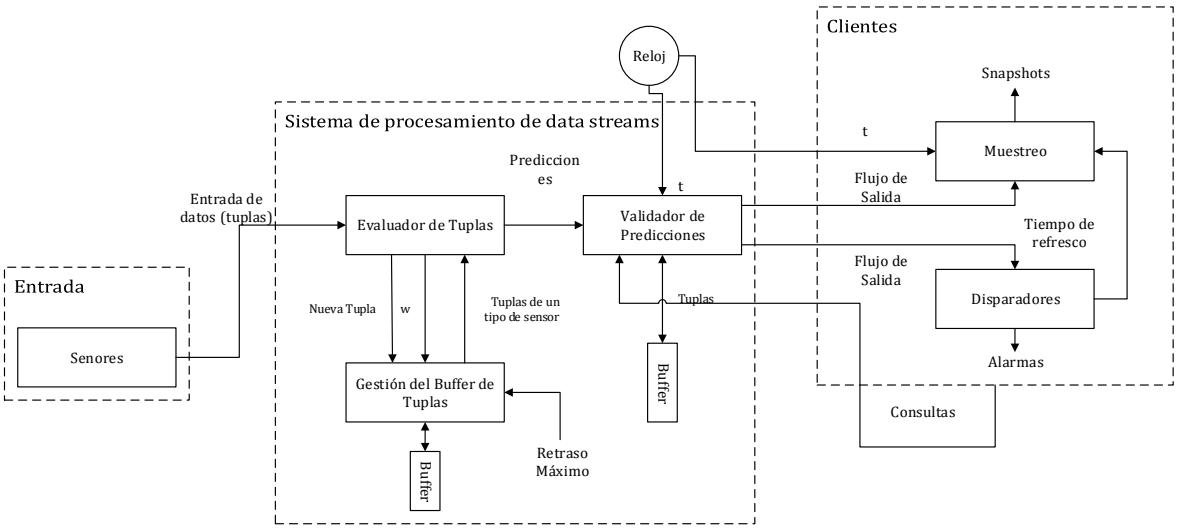


*Figura 1 Arquitectura del prototipo, adaptado del original [1]*

Este prototipo recoge la infraestructura para los módulos EVALUADOR DE TUPLAS, GESTIÓN DEL BUFFER y VALIDADOR DE PREDICCIONES dentro del diagrama de la arquitectura.

El módulo EVALUADOR DE TUPLAS se encarga de recibir las tuplas recibidas por el sensor, comprobar si esta nueva tupla provoca un cambio en el resultado de una consulta activa y realizar su almacenamiento en el buffer para posteriores análisis con tuplas nuevas.

El módulo GESTIÓN DE TUPLAS se encarga del almacenamiento de tuplas recibidas por el EVALUADOR DE TUPLAS. En el caso de que este buffer alcance su capacidad máxima, existen diversas políticas de eliminación de tuplas para crear espacio a las nuevas. Estas políticas pueden ser un sistema FIFO, un sistema de

descarte aleatorio de tuplas, descarte de aquella que tenga menos peso para el resultado de una consulta, etc.

Por último el módulo VALIDADOR DE PREDICCIONES se encarga de calcular valores para instantes de tiempos determinados, pudiéndose considerar un cierto retraso a la propagación de la tupla.

Este prototipo cuenta con dos limitaciones importantes:

1. Las consultas de tipo WINDOW JOIN solo pueden tener asignada una consulta VALUE como restricción.
2. Las consultas de tipo VALUE no pueden definirse solas, siempre tienen que ir acompañadas de una WINDOW JOIN.

Estas limitaciones se han tenido en cuenta en la elaboración de los módulos desarrollados para ampliar el sistema.

Para el desarrollo de este proyecto se ha conservado la implementación del sistema de procesamiento de *data streams* (es decir, los módulos EVALUADOR DE TUPLAS, GESTIÓN DEL BUFFER y VALIDADOR DE PREDICCIONES), por no ser objeto del presente proyecto, añadiendo el cliente de alarmas y el cliente de *snapshots*, además de la simulación de sensores.

## 1.3 Objetivos

Los objetivos que se marcaron al inicio de este proyecto fueron:

1-Estudio de nuevas operaciones interesantes a añadir al prototipo previo desarrollado por el grupo SID.

2-Realización de una interfaz gráfica apropiada para facilitar el uso de la aplicación (definición de consultas, ejecución, visualización de resultados y recogida de datos de evaluación experimental).

3-Desarrollo de una aplicación que permita simular la generación de datos procedentes de sensores para diversos entornos, así como la importación de fuentes externas.

4-Estudiar la posible utilización de *Hadoop Streaming* como middleware de soporte, analizando las ventajas e inconvenientes de dicha aproximación.

## 1.4 Tecnologías utilizadas

Para este proyecto se han utilizado los siguientes elementos:

Lenguaje de Programación: Java 1.7 [3]

Entorno de Desarrollo: Eclipse Juno y Eclipse Luna [4]

Control de Versiones: Git [5]

Persistencia de información: XML [6] [7]

Formatos Admitidos: GPX [7], PLT [8], CSV [9]

## 1.5 Fases del proyecto

Este proyecto se dividió en las siguientes fases:

### 1.5.1 Búsqueda de Información

En primer lugar se hizo necesario obtener información sobre redes de sensores y procesamiento de *data streams*. Para ello se realizó el estudio y análisis de diferentes artículos de investigación desarrollados sobre el tema [1] [2]. También para el desarrollo del simulador hubo que encontrar modelos matemáticos que realizaran simulaciones lo más verosímiles posibles. [10] [11] [12]

### 1.5.2 Análisis, Diseño e Implementación de un sistema de simulación

Debido a que el anterior prototipo usaba una tecnología de simulación ya descontinuada y retirada, software proporcionado por IBM, se vio la necesidad de realizar un simulador que supliera esta carencia, y además ampliara el número de fuentes de datos que pudiese manejar, incluyendo modelos de simulación que fuesen de lo más sencillos, hasta lo más reales y complejos posibles. Debido a la naturaleza del mismo se realizó un análisis previo, un diseño de las interfaces y del modelo de simulación, implementación y pruebas propias para este módulo.

### 1.5.3 Familiarización con el prototipo anterior.

Este prototipo se trataba de un sistema en fase de investigación, con lo cual no existía interfaz gráfica de usuario. Solo era posible realizar un número reducido de opciones por línea de comandos con una serie de baterías de test predefinidas en código. Se hizo necesario aprender el manejo de las funciones principales del

sistema así como el formato de las salidas de datos para poder darles la apariencia adecuada.

### 1.5.4 Fusión del sistema de simulación con el prototipo

Una vez conocido el funcionamiento del prototipo y desarrollado el sistema de simulación hubo que realizar la conexión de ambos sistemas, ajustando diversas clases para que ambos sistemas manejasen el mismo sistema temporal.

### 1.5.5 Análisis y Diseño de la Interfaz

Después de comunicar ambos sistemas y comprobar que los resultados eran adecuados, se procedió a realizar el análisis y diseño de la interfaz, así como funcionalidades ampliadas que no existían en el prototipo (sistema de alarmas, cálculo de funciones de agregación y el módulo de *snapshots* para monitorización).

### 1.5.6 Implementación de la interfaz

Una vez realizado el diseño de las nuevas operaciones y de la interfaz se procedió a su implementación.

### 1.5.7 Pruebas globales

Una vez finalizado se realizaron las pruebas de funcionamiento globales de la aplicación, con diversos casos de simulación para el correcto funcionamiento del sistema.

## 1.6 Estructura de la memoria

La memoria del proyecto consta de los siguientes capítulos:

- **Capítulo 1**: Introducción.

- **Capítulo 2**: Contexto tecnológico. Este capítulo pone en contexto la aplicación según las tecnologías usadas.

- **Capítulo 3**: Planteamiento y desarrollo del sistema.

- **Capítulo 4**: Pruebas y problemas encontrados. Baterías de pruebas y solución de problemas.

• **Capítulo 5**: Conclusiones. Conclusiones personales y trabajo futuro.

Por otra parte, los anexos que forman parte de esta memoria son:

• **Anexo A**: Manual de usuario y tutorial de la aplicación

• **Anexo B**: Formatos de los ficheros. Se explican los formatos de los diversos ficheros utilizados en la aplicación.

• **Anexo C**: Análisis del sistema. Este capítulo recoge el análisis detallado del sistema.

• **Anexo D**: Diseño e implementación del sistema. Explicación exhaustiva del diseño de la aplicación.

• **Anexo E**: Estrategias de movilidad. Se explica de forma detallada cada estrategia de movilidad implementada en este PFC.

• **Anexo F**: Modelo de Black-Scholes. Se explica de forma detallada el modelo de Black-Scholes para realizar simulación de cotizaciones.

## 1.7 Planificación del proyecto

Para la realización de este proyecto se dividió en dos grandes grupos: la creación del simulador y la integración con el prototipo anterior, añadiéndole la capa de interfaz de usuario que homogeneizaría toda la aplicación.

El proyecto se desarrolló durante el curso académico 2013/2014, y el primer trimestre del 2014/2015 dedicando una media de dos o tres horas diarias por razones laborales y otros cursos ajenos a la universidad.

### 1.7.1 Listado de tareas

Este es el desglose de tareas realizadas durante la elaboración del proyecto:

**Inicio del proyecto**: Se trató de la primera fase, en ella después de una serie de reuniones con el tutor se trataron los objetivos a alcanzar y se proporcionaron diversos artículos de investigación del departamento que recogen los resultados del prototipo anterior.

**Búsqueda de Información (45 horas)**: Después de definidos los objetivos, se hizo necesario el estudio de los documentos de investigación, el tratamiento de la información y el sistema de consultas, todo ello de forma teórica.

**Realización del prototipo de simulación (30 horas)**: Una vez conocido el sistema de predicción que usa el prototipo, se realizó un pequeño simulador en modo texto como prueba para comprobar si era viable realizar uno más completo que calculase funciones de predicción. Una vez validado por el tutor, se procedió a ampliar este simulador para realizar más tipos de simulación (Sin función de predicción, aplicar modelos de movilidad, otros sistemas de comunicación, etc.)

**Análisis y Diseño e Implementación del Simulador (150 horas)**: Tras el validado del prototipo de simulación se realizó el análisis del mismo, el diseño y la implementación incluyendo la interfaz gráfica. Esta interfaz sirvió como base del diseño de la interfaz de ambas aplicaciones integradas.

**Estudio del prototipo del sistema de procesamiento de data streams (25 horas):** Tras la verificación de que el simulador cumplía las expectativas, se proporcionó el código del prototipo, el cual contenía un módulo de simulación básico basado en el simulador de IBM ya descontinuado, por lo que hubo que, primeramente, familiarizarse con el código del simulador y segundo averiguar cómo comunicar el nuevo módulo de simulación a la aplicación anterior.

**Análisis de nuevas operaciones (30 horas):** Tras la familiarización, se estudió la aplicación de nuevas operaciones (agregación, sistema de alarmas y *snapshots*).

**Análisis y diseño de la interfaz del sistema de procesamiento (20 horas)**. Utilizando como base la interfaz desarrollada para el simulador, se realizó el análisis previo de la interfaz, así como el diseño de la misma.

**Implementación de la interfaz y de las nuevas operaciones (100 horas).** Se desarrolló la interfaz y las nuevas operaciones.

**Estudio de Hadoop Streaming (10 horas):** En un principio se consideró la utilización de Hadoop Streaming, en este punto se analizó su conveniencia.

**Pruebas (15 horas).** Se desarrollaron diferentes pruebas para comprobar el correcto funcionamiento de todo el sistema. Estas pruebas se realizaron en un sistema Windows 7, tanto en el entorno eclipse, como ejecuciones separadas con la máquina virtual Java.

**Elaboración de la memoria (90 horas):** Desarrollo de esta memoria

**Reuniones (6 horas aprox.)** Durante el transcurso del proyecto se realizaron diversas reuniones con el tutor de unos 45 minutos aproximadamente cada una, donde se trataban diversos puntos a lo largo del desarrollo del proyecto

### 1.7.2 Control de tiempos

El proyecto se ha realizado en unas 550 horas, en este gráfico se puede observar la distribución del trabajo:

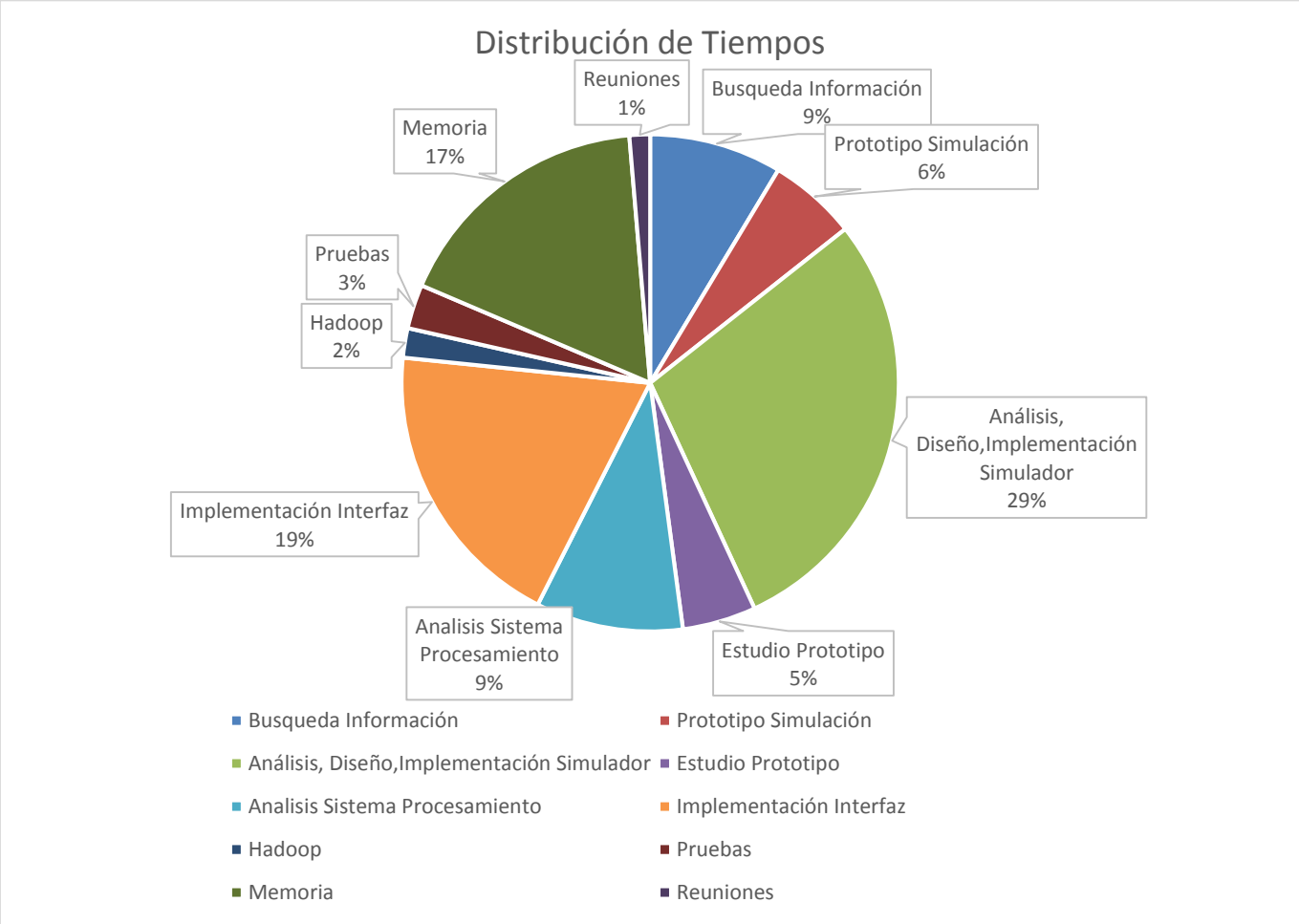


*Figura 2 Distribución de Tiempos*

# 2 CONTEXTO TECNOLÓGICO

En este capítulo se van a presentar todas las tecnologías que se han utilizado para la realización de este proyecto, en él se incluyen lenguajes, plataformas, herramientas de soporte, asistentes de programación, etc.

## 2.1 Java

Java es un lenguaje de programación orientado a objetos creado por *Sun Microsystems* y perteneciente a Oracle gracias a la adquisición de ésta empresa [3]. Se trata de un lenguaje muy utilizado en la comunidad de software libre por su simple manejo de objetos y la posibilidad de ejecutarlo en cualquier máquina gracias a la Máquina Virtual donde se ejecutan los programas desarrollados en este lenguaje [13]

Para la realización del proyecto se ha optado por este lenguaje de programación y las razones que han llevado a esta decisión son:

1) El prototipo anterior del grupo estaba desarrollado en este lenguaje, luego era lógico continuar desarrollando sin cambiar el lenguaje
2) Tiene una librería gráfica potente (Java Swing) que permite realizar una interfaz gráfica sin depender de librerías de terceros (como puede ser Qt) [14]
3) Es un lenguaje multiplataforma, lo cual amplía el número de computadoras que pueden manejar el programa.
4) Tiene un manejo simple de la concurrencia.

Este proyecto ha utilizado la versión 1.7 de Java.

## 2.2 Eclipse

Eclipse es el entorno de desarrollo (IDE) por excelencia de Java, junto con Netbeans. Este mantenido por la *Eclipse Foundation*, una comunidad de software abierto que cuenta con su propia licencia de software libre. Este entorno ofrece una gestión integral de la fase de implementación del proyecto.

Para este proyecto se han utilizado las versiones Juno del año 2013, y Luna del año 2014. Ambas versiones tienen integrados diversos *plugins* para el diseño de interfaces gráficas (*Window Builder*) y compatibilidad con varios sistemas de

control de versiones (*Mercurial*, *Git* y *SVN*). Además existen una gran variedad de *plugins* desarrollados por la comunidad para la automatización de tareas, baterías de test, construcción de dependencias etc. [4]

## 2.3 Git

Git es un sistema de control de versiones desarrollado por *Linus Torvald*s muy usado en la comunidad de software libre (es el sistema de control de versiones utilizado por la Linux Fundation, encargada del mantenimiento del Kernell Linux.) Se trata de un sistema de control de versiones distribuido (a diferencia de SVN que es centralizado), en el que cada usuario tiene una copia de trabajo en la que desarrolla sus modificaciones y después se fusionan con el repositorio central (denominado *origin*). Permite trabajar en varias ramas diferenciadas de trabajo (denominadas *branche*s) para después unirlas en una misma rama y asignar etiquetas a distintos puntos del trabajo (*tags*) [15].

Se decidió utilizar este sistema de versiones a pesar de que solo había un desarrollador por su facilidad de uso y la cantidad de servidores gratuitos que ofrecen *hosting* para pequeños proyectos de forma gratuita, pudiendo compartir el proyecto entre varios equipos, asegurando siempre una copia de respaldo en la nube.

## 2.4 Trazas GPS

Se define traza como una serie de puntos que son almacenados por un dispositivo con capacidad de conexión GPS en los que aparece información como la posición geográfica (latitud y longitud), fecha de registro, hora de registro, altitud, etc.

Existen una gran variedad de formatos. Uno de los más usados es GPX (*GPS eXchange Format*) se trata de un formato de fichero que sigue un esquema XML donde se almacena la posición obtenida, la hora de registro y la altitud a la que se registró. Este formato se utiliza en diversas aplicaciones de monitorización deportiva. [7]

Otra plataforma que usa trazas GPS es *GeoLife*. Esta plataforma es una red social creada por Microsoft nacida con la finalidad de conectar personas según sus trazas GPS. Durante más de tres años se monitorizó las posiciones de 165 personas obteniendo datos totales equivalentes a más de un millón de kilómetros. Esta red tuvo gran éxito en Asia, donde se tomaron casi todas las monitorizaciones. [8]

*Movebank* es otra comunidad de rastreo GPS, pero esta vez aplicado a animales. Este repositorio esta mantenido por la asociación *Max Plank* de ornitología. En esta asociación se monitorizan movimientos migratorios con una gran variedad de sensores, entre ellos GPS y utiliza un formato CSV (ficheros de texto separados por comas) [9]

Este proyecto usa como formatos de entrada estos tres formatos de trazas reales GPX, PLT y CSV, este último siempre que contenga datos sobre trazas reales (ya que puede contener muy variada información ya que no es un formato propio de trazas de posicionamiento)

## 2.5 Sistema de procesamiento de *data streams*

Los sistemas de procesamiento de flujos de datos (data streams) son programas informáticos encargados de manejar flujos continuos de información [16]. Se asemejan a las bases de datos relacionales, pero a diferencia de éstos no están pensados para datos estáticos. También sus lenguajes de manipulación e interrogación son diferentes ya que en los flujos continuos una consulta tiene que actualizarse constantemente para ofrecer el resultado de las consultas considerando los datos actualizados, para ello una de las alternativas es la utilización del CQL o *Continuos Query Language*, donde se pueden definir consultas con una sintaxis similar al SQL, incorporando una condición de ventana (*window*) en la que aplicar la consulta.

Los sistemas de procesamiento de data streams que se pueden encontrar actualmente son SQLStream [17], Stream [18] , Aurora [19], etc.

## 2.6 Hadoop y Hadoop Streaming

*Apache Hadoop* [20] es un framework pensado para aplicaciones distribuidas que trabajan con miles de nodos y una gran necesidad de almacenamiento de datos (del orden de petabytes). Consta de los siguientes módulos:

**Hadoop Common**: Utilidades comunes para los demás módulos.

**Hadoop Distributed File System (HDFS™):** Es un sistema de ficheros distribuido de acceso rápido.

**Hadoop YARN:** Módulo que se encarga de la secuenciación de trabajos y la gestión de clústeres.

**Hadoop MapReduce**: Permite el proceso paralelo de grandes cantidades de datos, basado en el módulo YARN.

Poco antes de la inicialización de este proyecto, apareció la herramienta *Hadoop Streaming* [21] asociada al proyecto Hadoop de Apache. En principio la utilización de operaciones de *Map/Reduce* podría parecer interesante a la hora del cálculo de las operaciones de consulta. Tras la lectura de diversos manuales y documentación se comprobó que *Hadoop* no está indicado para aplicaciones en tiempo real (es decir, los datos se están actualizando constantemente, como es nuestro caso) sino que está pensado para trabajos en lote, con grandes cantidades de datos distribuidos, siendo este prototipo centralizado. Es decir, se manda un trabajo a Hadoop y este se encarga de establecer que nodos realizan el trabajo y recoge los resultados. *Hadoop Streming* no es más que una utilidad para poder definir funciones de Map y Reduce propias, en cualquier lenguaje de programación en vez de las predeterminadas por el sistema [22].

*Hadoop Streaming* realiza las operaciones de Map primero a todos los datos de entrada, antes de pasarle estos datos mapeados a la operación *reduce*. Si están llegando datos de entrada continuamente (como es un sistema de monitorización), la operación de *Map* no terminaría nunca con lo cual parece que esta opción no es viable para nuestro sistema.

Sin embargo la fundación Apache ha realizado un sistema de alta capacidad denominado Apache Storm [23], pensado para trabajos en tiempo real en sistemas distribuidos, utilizando la plataforma *Hadoop.* Esta solución podría considerarse para realizar una ampliación futura del presente proyecto.

# 3 PLANTEAMIENTO Y DESARROLLO DEL SISTEMA

Una vez conocidos los objetivos del proyecto, la búsqueda de información sobre flujos de datos y la realización de un pequeño prototipo de simulación como prueba, se establecieron los requisitos del sistema, su análisis, el diseño e implementación del mismo.

## 3.1 Análisis de requisitos

Tras la realización del prototipo de pruebas, se establecieron los puntos que se consideraron interesantes para incluir en la simulación de datos. Uno de los primeros objetivos era que el simulador pudiese incluir varios modelos de movilidad. En un primer momento, el prototipo de prueba solo realizaba un sorteo aleatorio para determinar si cambiaba de dirección o no, lo cual no se ajustaba a la realidad. Para ello se produjo una búsqueda de información sobre políticas de movilidad y algoritmos de simulación de movimientos.

Los modelos que se estudiaron para implantar fueron:

Modelo Manhattan: Se trata de un modelo en el que el elemento móvil puede cambiar de trayectoria a norte sur este y oeste (como la distribución de edificios en una ciudad) [10].

Modelo Gauss-Markov: Se trata de un modelo de movilidad basado en la distribución normal, se obtienen trayectorias mucho más reales [12].

Modelo de Dijkstra: Se trata de un modelo basado en el algoritmo de Dijkstra, su uso es principalmente para el cálculo de rutas.

Como lo interesante en este simulación era realizar trazas de movimiento (sin ninguna restricción de por dónde puede o no pasar el elemento móvil) se determinó añadir el modelo Manhatan y el modelo de Gauss-Markov, rechazando el modelo de Dikstra ya que no se iban a realizar rutas.

También en este punto se decidió ampliar la simulación no solo a elementos móviles, sino incluir simulaciones de elementos unidimensionales, para ello se buscaron diferentes modelos matemáticos que permitiesen una buena adaptación

con el simulador. Durante la investigación de estos modelos matemáticos se encontró el modelo de Black-Scholes [11] [24]. Este modelo permite la simulación del precio de una acción en N periodos de tiempo.

Puesto que estos modelos parecían escasos, se decidió añadir un tercer tipo de simulación, denominado genérico, que permitiese añadir un número aleatorio a una cantidad inicial determinado por una distribución de probabilidad, permitiendo al usuario modelar su propia simulación.

Una vez determinados los modelos de movimiento en la simulación GPS y que otros simuladores se implementarían se determinó que políticas de actualización se seguirían. El sistema de procesamiento funciona con funciones de predicción que se modifican según se incumpla una política de error máximo (el dato se encuentra a una distancia mayor que el predicho por la función) o de tiempo máximo (el último dato se ha enviado un tiempo muy anterior al determinado). Se decidió que estas políticas se pudieran dejar a elección del usuario, pudiendo usar una, las dos, o ninguna.

Como ya se ha dicho el sistema de procesamiento trabaja con funciones de predicción, con lo cual había que mantener esta funcionalidad. También se decidió dejar la posibilidad de enviar el valor medido por el sensor directamente, en vez de realizar el cálculo de la función de predicción.

Una vez determinadas las funcionalidades del sistema, se vio que era necesario importar trazas reales. En un principio se contempló solo el uso del formato GPX y una vez realizado el simulador se decidió probar la versatilidad del mismo añadiendo trazas de GeoLife y Movebank.

Realizado el análisis para un solo sensor, la realización de la simulación de N sensores era el siguiente paso ya que se podía aprovechar toda la infraestructura para un número mayor de sensores.

Tras esto se determinaron los objetos de la interfaz. En principio se trataba de una interfaz sencilla, visualizar la simulación y mostrar las comunicaciones que se producen, Posteriormente se añadió la posibilidad de pausar y detener la simulación, así como exportar e importar simulaciones ya realizadas.

Con todo esto determinado se procedió a diseñar y desarrollar el simulador.

Una vez realizado se efectuó el análisis de las nuevas operaciones a añadir al sistema. Se había planteado la posibilidad de añadir operaciones de agregación al

simulador, esto sería útil para realizar cálculos sobre los datos almacenados en tiempo real por el sistema. Se determinó realizar las funciones típicas de agregación de un sistema gestor de bases de datos, estas son AVG (media aritmética de los datos almacenados en sistema), DESVEST (desviación típica), MIN (dato mínimo almacenado por el sistema), MAX (dato máximo almacenado por el sistema), COUNT (número de datos almacenados del sistema). El sistema de procesamiento trabaja principalmente por clases (las consultas se realizan sobre clases de sensores), con lo cual se realizaron estas operaciones para clases de sensores, si bien, posteriormente, se determinó que también sería interesante incluirlas para los sensores de forma individual. Debido a las limitaciones del prototipo, estas operaciones solo se aplican a las tuplas que permanezcan activas en el buffer y no se hayan eliminado por no ser útiles para la consulta.

También se incluyó la posibilidad de asociar una alarma a una consulta para mostrar al usuario cuando se produzcan resultados (puede ser útil para monitorizar datos críticos) y la visualización de estados concretos de las consultas.

Éste es un pequeño resumen de los requisitos de la aplicación que pueden ser consultados en el Anexo C: Análisis del sistema.

## 3.2 Diseño e implementación del sistema

Una vez realizado el análisis, y marcadas las pautas generales para el transcurso del proyecto, se procedió al diseño de la solución y la implementación de la misma.

Se ha intentado seguir una estructura modular, distribuida en paquetes que contienen las clases según su finalidad, así como un paquete con utilidades varias que son comunes a toda la aplicación.

Aprovechando la capacidad de herencia de los lenguajes orientados a objetos como es Java, se ha realizado una serie de clases abstractas (*AbstractSimulator*, *AbstractSensor* y *AbstractCommunication*). Con ello se consigue que realizar una ampliación del simulador sea tan sencillo como declarar una clase que descienda de ella, implementar las variaciones que sean necesarias y asociarla al simulador, Esta misma idea se aplica a los modelos de movilidad, donde existe la clase *AbstractMobility* de la que descienden los modelos de movilidad.

También se ha implementado una clase *Controller* que contiene todas las funciones que se pueden realizar a través de la interfaz gráfica. Esto permite una separación entre las funcionalidades del sistema y la interfaz gráfica, lo que permite que los cambios en funcionalidad (como puede ser una mejora, una solución a un error

etc.) sean realizados en las clases correspondientes sin tener que modificar otras partes del código de la interfaz, y viceversa, un cambio en la interfaz no tiene que afectar a las clases de tratamiento de datos.

Para el tratamiento de ficheros se ha optado también por esta estrategia modular, se han creado varias clases que importan los tipos según fichero, con una serie de operaciones comunes, y dependiendo la extensión del fichero automáticamente se selecciona la clase necesaria. Para la exportación de simulaciones y resultados de las consultas se ha realizado una clase que se encarga de escribir ficheros en formato XML. Para más información acerca del diseño de la aplicación consultar el Anexo D: “Diseño e implementación del sistema”.

En el siguiente gráfico se puede observar el esquema global de la arquitectura de la aplicación a muy alto nivel:

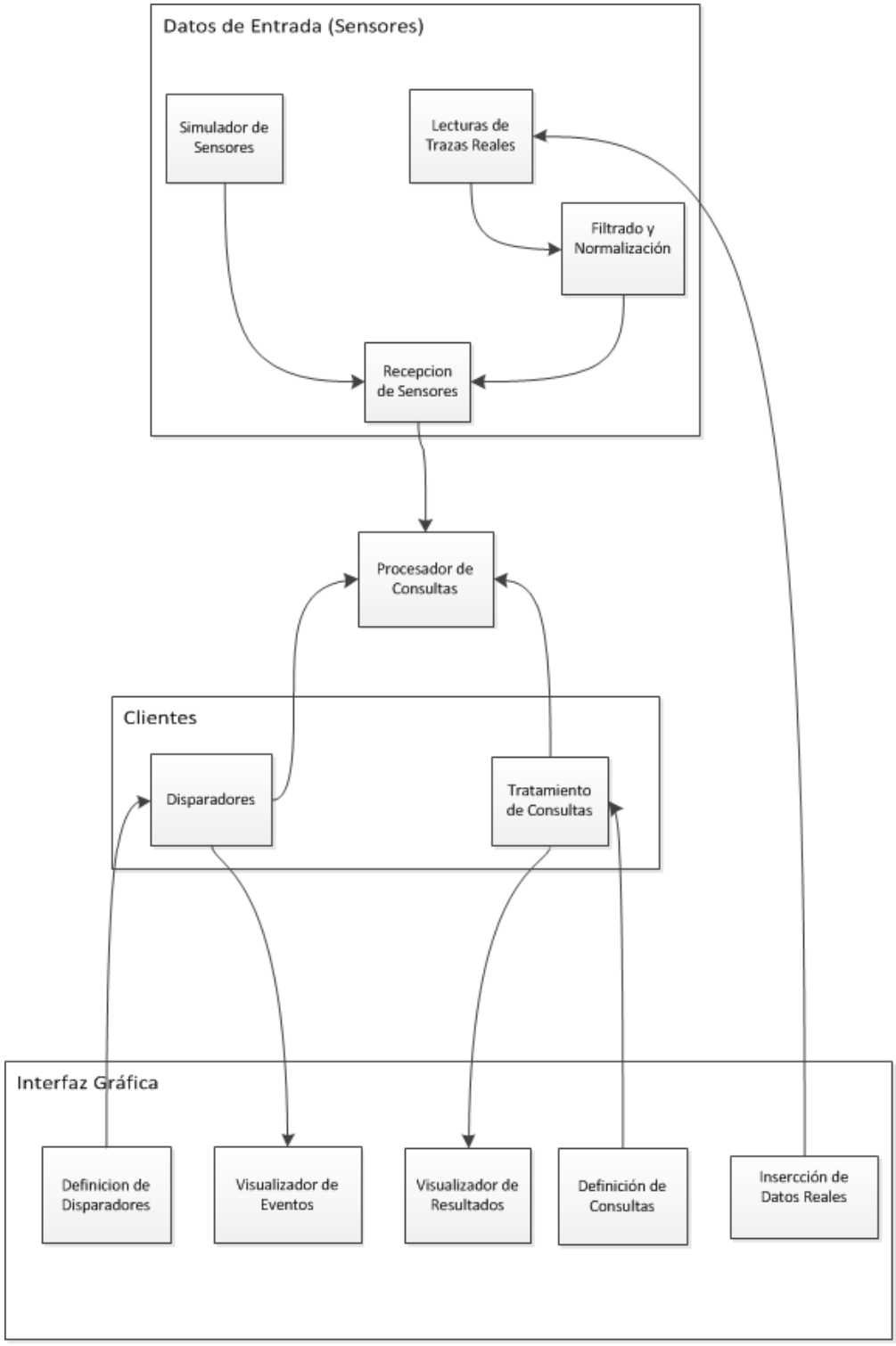


*Figura 3 Diseño Global de la Aplicación*

Y el siguiente muestra la estructura de clases abstractas que sigue el sistema de simulación.


*Figura 4 Diseño en alto nivel del Simulador*

### 3.2.1 Estrategias de movilidad

Todas las estrategias de movilidad que existen en la aplicación heredan de la clase *AbstractMobility*, siendo con ésta la que se comunica el sensor GPS, Estas estrategias, implementan la interfaz *Runnable* ya que dependiendo del modo de ejecución (tiempo real, o iterativa) el propio actualizador debe ser capaz de lanzarse en un nuevo *Thread* que sea el encargado de proporcionar el nuevo dato al simulador. El esquema sería el siguiente:

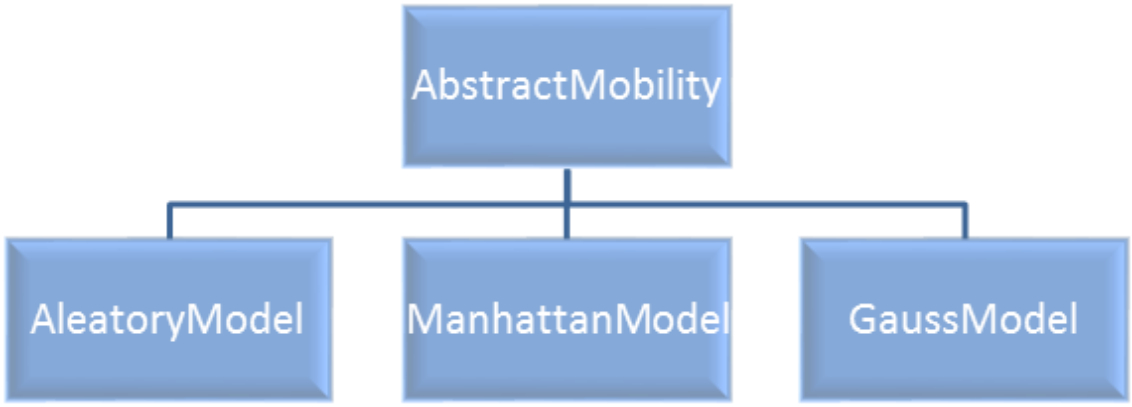


*Figura 5 Modelos de Movilidad*

Para ampliar la información de la definición e implementación de los modelos de movilidad consultar el Anexo E: Modelos de Movilidad.

### 3.2.2 Sistema de carga y almacenamiento de parámetros

El sistema anterior contiene una clase llamada *Constants* que incluye todas las configuraciones que se pueden realizar en el programa, pero estas debían ser cambiadas en el código fuente cada vez que se quisiera cambiar un parámetro, algo totalmente contraindicado para el desarrollo de una aplicación usable y pensada para otras personas además del desarrollador. Para ello se estableció un fichero denominado config.xml que crea persistencia de la configuración de la aplicación y que mediante un menú puede modificarse. Este fichero tiene el siguiente aspecto:

```
<config>

	<sensorTime>1</sensorTime>

	<simulatorTime>3</simulatorTime>

	<bufferPolicy>0</bufferPolicy>

	<simulateDelay>false</simulateDelay>

	<probDelay>50.0</probDelay>

	<maxDelay>1</maxDelay>

	<R-tree>1</R-tree>

	<prediction>1</prediction>

</config>
```

En base a estas opciones, se ajustan todas las demás opciones de la aplicación.

### 3.2.4 Interfaz de usuario

La aplicación previa desarrollada por el SID carecía de interfaz gráfica. Solo podía configurarse la simulación a través de la línea de comandos a la hora de lanzar la aplicación de test. Puesto que hoy en día este uso está muy desactualizado, se hizo patente la necesidad de realizar una primera interfaz gráfica que permitiese al usuario interaccionar con el sistema de forma amigable y poder desarrollar y tener almacenadas todas las consultas que desease hacer, así como la posibilidad de crear nuevas consultas. Aprovechando la necesidad de crear el simulador, se optó por realizar una interfaz conjunta y no tener dos aplicaciones separadas.

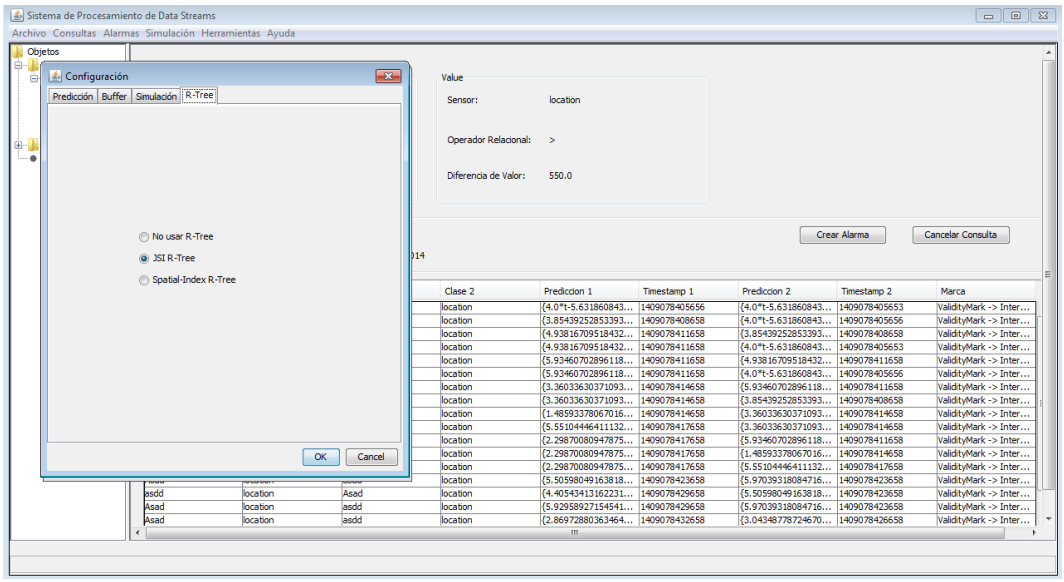


*Figura 6 Apariencia de la interfaz*

En concreto las posibilidades de la interfaz son:

1-Manejo de una simulación

El usuario puede realizar una nueva simulación, pausarla, detenerla, guardarla en un fichero e importar una simulación realizada previamente. La simulación puede constar de un solo sensor o de varios sensores.

2-Configurar un sensor

Según el tipo de simulación elegido, el usuario podrá aportar una serie de parámetros mediante una pantalla de configuración para realizar la simulación a sus necesidades. En el caso de que este realizando una simulación de una red de sensores, el usuario puede clonar un simulador para no tener que introducir en todos y cada uno los datos requeridos.

3-Manejo de Consultas

El usuario podrá crear consultas nuevas, visualizar los resultados de las mismas, guardarlos en un fichero y recuperarlos posteriormente. En base de estas consultas se podrán definir alarmas para su monitorización automática y visualizar resultados para tiempos de refrescos definidos.

Para más información sobre la interfaz se puede consultar los anexos A Manual de Usuario y D Diseño de la Aplicación (en concreto el apartado sobre navegación de ventanas)

### 3.2.5 Interpolación para obtener la función de predicción

Para que el simulador pueda proporcionar la función de predicción en la comunicación, se hace necesario realizar una interpolación [25] entre el valor del sensor en el punto de la anterior comunicación y en el punto actual. Como conocemos el tiempo $T$ en el que se ha enviado la información anterior, podemos recuperar el dato en base a su función de predicción con el que obtendremos el valor $f(T)$. El nuevo punto es el dato medido con el sensor, luego realizando una interpolación lineal entre el valor anterior y el nuevo podemos obtener una función cuyo valor en $T$ será el valor anterior y en $T'$ será el valor nuevo.

Matemáticamente:

Sea $x''$ el nuevo valor medido por el sensor en el instante $T''$ y x' el valor $f(t')$ calculado con la función de predicción anterior. Se tiene que:

$$\frac{x - x'}{x'' - x'} = \frac{t - t'}{t'' - t'} => (x - x')(t'' - t') = (t - t')(x'' - x')$$

Agrupando y reduciendo términos se obtendrá que la nueva función de predicción será:

$$f'(t) = \frac{(x'' - x')t - (x'' + x')t1 + \ 2x't''}{t'' - t'}$$

Está interpolación será necesaria realizarla en todas las dimensiones de las que conste la simulación (2 en el caso GPS, 1 en las demás).

### 3.2.6 Conversión de coordenadas

Las trazas GPS al hacer uso de los datos de longitud y latitud suelen utilizarlas en grados decimales (es decir, 32.2931º sería un valor válido para cualquiera de las dos dimensiones) Este sistema es muy difícil de manejar a la hora de medir distancias para realizar una política de error máximo. Es por esto que se hace necesario transformar estas coordenadas a un eje X-Y tradicional, para ello se han utilizado estas funciones de transformaciones simples. [26]

```
public Vector2D toXY (double lon, double lat) {

	double x = Math.toRadians(lon)*6378137.0;

	double Y =
	Math.log(Math.tan((Math.toRadians(lat)/2)+(Math.PI/4)))*6378137.0;

}

public Vector2D toLatLong (float x, float y) {

	float lon = (float) ((x / 20037508.34) * 180);

	float lat = (float) ((y / 20037508.34) * 180);

	lat = (float) (180/Math.PI * (2 * Math.atan(Math.exp(lat * Math.PI /
	180)) - Math.PI / 2));

	return new Vector2D(lon, lat);

	}
```

Estas funciones permiten la transformación de una longitud y latitud a un eje cartesiano y viceversa.

### 3.2.7 Alarmas

Dentro de la esta aplicación se define alarma como una consulta que se encuentra monitorizada por la aplicación y cuando se producen resultados avisa al usuario y le permite acceder a la monitorización del sistema. Podrían asemejarse a los *Triggers* de base de datos que se disparan en el momento que se realiza una acción

concreta contra el gestor. Para el usuario basta con marcar como alarma una consulta ya creada (o crear una consulta al crear la alarma) para que este fenómeno se produzca. Internamente se ha creado la clase *Trigger* que se encarga de almacenar todas las alarmas (clase *Alarm*, que están asociada a un objeto tipo *Query*). Esta clase *Trigger* se trata de un *thread* que se lanza al iniciar la aplicación, y consulta todas las alarmas que se han asociado cada cierto periodo de tiempo (1 segundo). Si se ha producido resultado en alguna de ellas, el sistema lanzará un mensaje por pantalla informando del resultado obtenido y dando la oportunidad al usuario de saltar a la pantalla de resultados de la consulta. En cualquier momento se puede cancelar una alarma para dejar de monitorizar y obtener mensajes. Para más información sobre las alarmas acudir al capítulo de ALARMAS del anexo A Manual de Usuario.

### 3.2.8 Snapshots

Un *snapshot* es el estado del cálculo de una consulta en cada instante de tiempo determinado, es decir, es como una fotografía del resultado actual del sistema. En ella la respuesta se va actualizando según un intervalo de refresco que proporciona la tupla (o tuplas) que se pueden utilizar en el instante actual de tiempo para determinar el valor futuro de una pareja de sensores. Esta tuplas admiten parejas de valores temporales en lo que se denomina *"Timestamp-Matching-Boundary"* que indica qué par de valores de tiempo pueden aplicarse para proporcionar un resultado dentro del intervalo de validez de esta consulta.

Esta aproximación es útil para un cliente que necesite monitorizar continuamente una respuesta.

### 3.2.9 Estadísticas

Se ha decidido mantener el cálculo de las estadísticas de procesamiento que ya estaba incluido en el prototipo. Estas estadísticas calculan diversos valores producidos durante ejecución del sistema de procesamiento. Estas mediciones recogen datos como el número de tuplas procesadas, el tiempo de proceso, el retraso producido, el número de comparaciones y su tiempo de proceso, el tiempo de utilización (inserciones, ordenaciones y borrados) del R-tree y el tiempo utilizado para gestionar el buffer. Todas estas estadísticas pueden utilizarse para elaborar estudios como el impacto del número de objetos simulados en el tiempo de proceso en el sistema de procesamiento, el impacto de las políticas de actualización respecto al número de comparaciones realizadas para calcular los

resultados de las consultas, el impacto que tiene el tamaño del escenario respecto al número de comparaciones [1] [2].

El sistema de simulación desarrollado para la aplicación permite calcular el número de tuplas enviadas, el porcentaje de ellas que se envían según su política de actualización, el tiempo de proceso y el tiempo de simulación y el error medio producido por usar funciones de predicción. Todas estas estadísticas salvo el porcentaje según su política de actualización ya se encontraban en el anterior sistema, pero ha sido necesaria su adaptación para el nuevo simulador.

Estas estadísticas aparecen al usuario una vez ha terminado la ejecución de la aplicación en un cuadro de texto para poder ser copiado a un fichero para realizar el trabajo experimental que se desee sobre ellos.

En el artículo *"A Query Processor for Prediction-Based Monitoring of Data Streams"* [2] se presentan tres gráficas que muestran experimentos realizados con el prototipo sobre el que se trabaja en este proyecto.

En primer lugar se estudia la escalabilidad del procesador de consultas según su número de objetos. Este estudio podría realizarse en la aplicación simulando cada vez un número mayor de objetos y obteniendo los resultados al final de la ejecución del programa.

Como segundo estudio, se realiza una estimación del beneficio del uso de funciones de predicción. En concreto se realiza una comparación entre una serie de simulaciones usando funciones de predicción y otras sin usarlas. Para ello habría que configurar el programa con las opciones adecuadas (uso o no de estas funciones) y luego realizar simulaciones con el número de objetos deseado.

Por último se realiza un estudio comparando las diferentes políticas de gestión de buffer, dependiendo de la ventana temporal definida para la consulta. Para realizar este estudio bastaría con realizar una simulación, guardarlas en un fichero y después crear consultas con diferentes ventanas temporales. Una vez creadas bastaría con importar el fichero de simulación para cada política de gestión de buffer seleccionando la opción deseada dentro de la configuración de la aplicación

El coste de realizar la replicación de estos experimentos exigiría la creación de una serie de baterías de test dentro de un programa Java que simulara la ejecución de los experimentos mencionados anteriormente, y su ejecución mediante una serie de scripts para automatizar el proceso. Se ha optado por dejar esto fuera del

desarrollo del proyecto, pero es presumible que llevaría un esfuerzo moderado realizar todas estos estudios.

# 4 PRUEBAS Y PROBLEMAS ENCONTRADOS

En este capítulo se van a presentar diferentes pruebas realizadas a la aplicación de simulación, como al sistema de procesamiento, así como los problemas encontrados a la hora de la realización del proyecto.

## 4.1 Pruebas realizadas

Para verificar el correcto funcionamiento de la aplicación las pruebas a las que fue sometida la aplicación en sus diferentes fases han sido las siguientes:

• **Pruebas unitarias**: estas pruebas se realizaron para comprobar el adecuado funcionamiento de las diferentes clases de las que se compone el proyecto y las funcionalidades implementadas de forma individual. Estas pruebas se realizaron en paralelo con el desarrollo de las funciones, asegurando que durante el desarrollo éstas quedaban correctamente realizadas.

• **Pruebas de integración**: Una vez realizada la correcta ejecución de los métodos de los que constan las clases, se procedió a comprobar el correcto funcionamiento de la comunicación entre ellas. Sobre todo fue importante el punto donde la aplicación de simulación acaba comunicando sus resultados al sistema de procesamiento, este punto era crítico por lo cual se realizaron pruebas exhaustivas de que el sistema comunicaba los datos de simulación, teniendo que realizar ajustes ya que el sistema temporal difería entre ambos módulos.

• **Pruebas de sistema y aceptación**: Una vez terminada la aplicación, se procedió a realizar pruebas de sistema y de aceptación. Para ello se generaron una serie de scripts para Windows y Linux para su compilación y ejecución. Con ello se ejecutaron todas y cada una de las funcionalidades del sistema. En este punto se corrigieron varios fallos de denominación en los menús (palabras repetidas, etc.). Aprovechando estas pruebas se comprobó que la aplicación se ajustaba a los requisitos planteados y los objetivos previstos. Estas pruebas se han realizado en un Pc de 64 bits con Windows 7 y versión de JDK 1.7 y Ubuntu sobre una Máquina Virtual de 64 Bits con versión OpenJDK1.7

## 4.2 Problemas encontrados

Durante el transcurso de las pruebas se encontraron estos problemas graves que fueron solucionados antes de la versión definitiva:

-**Ajustes Temporales**: Mientras que la aplicación original de procesamiento de *data streams* trataba el tiempo como absoluto (es decir en base a la hora actual se procedía a calcular el valor con las funciones de predicción), el sistema de simulación trabaja con tiempos relativos (aunque se almacena la hora en que se envía la información, se usa para determinar un valor temporal tomando como referencia el primer dato enviado, que se considera tiempo 0). Esto hacia que los resultados obtenidos por las funciones de predicción en ambas aplicaciones fueran distintos y por consiguiente erróneo. Para solucionar esto, se realizó una función que cambiase el sistema temporal de las funciones de predicción el momento que se envían al sistema de procesamiento, gracias a este cambio ambos datos pasaron a ser iguales y correctos.

-**Valores Erróneos en la importación de trazas**: Durante las pruebas se comprobó que en el caso de GPX al realizar la importación se obtenían valores infinitos a la hora de transformar las coordenadas a eje X e Y. Se limitó el rango de valores que admitían las coordenadas para asegurar que siempre se importa un dato correcto y se comprobó que los atributos estaban cambiados a la hora de importar, por lo que tomaba la longitud como latitud y viceversa, provocando un desajuste en la transformación.

-**La simulación no se paraba o detenía:** Debido a la arquitectura de la clase *controller,* a la hora de realizar simulación de redes, no se era capaz de pausar o detener una simulación. Esto se solucionó teniendo una lista de simuladores que están actualmente en ejecución para poder mandar la información de pausa.

-**Almacenamiento de valores unidimensionales en un R-tree:** La estructura de datos R-tree está pensada para realizar búsquedas de elementos bidimensionales. Esto en primer momento parece un problema para realizar consulta sobre valores unidimensionales. En un primer momento se planteó utilizar una estructura de datos diferente, pero la implementación del prototipo estaba muy apoyada en la búsqueda en arboles R-tree, con lo cual se optó por utilizar un valor constante como segunda dimensión del mismo, comprobando que se producían resultados correctos.

# 5 CONCLUSIONES

En este apartado se van a presentar los resultados obtenidos con la realización del proyecto, así como una valoración personal de la elaboración del proyecto.

## 5.1 Resultados obtenidos

Tras la realización de la aplicación se han cumplido los objetivos planteados:

Se han estudiado nuevas operaciones interesantes a añadir al prototipo previo desarrollado por el grupo SID, en este caso se han desarrollado varias operaciones de agregación (la media de los datos, desviación típica, mínimo, máximo y número de datos que almacena el sistema), que se aplican a los datos que residan en el buffer de la aplicación, y no se hayan descartado para el cálculo de una consulta.

Se ha realizado una interfaz gráfica apropiada para facilitar el uso de la aplicación (definición de consultas, ejecución, visualización de resultados y recogida de datos de evaluación experimental). Esta interfaz permite definir consultas, alarmas y simulaciones para probar el sistema. Además muestra por pantalla las estadísticas de ejecución y simulación como datos experimentales.

Se ha desarrollo una aplicación que permite simular la generación de datos procedentes de sensores para diversos entornos, así como la importación de fuentes externas. Este punto ha ocupado la mayoría del desarrollo ya que según transcurría el proyecto se observó la potencia que tenía y su versatilidad permitía una utilización mayor del sistema de procesamiento. Este simulador es fácilmente ampliable y permite la importación de trazas de diversos formatos así como la simulación de varios tipos de datos. También permite el envío de valores puros o funciones de predicción, pudiendo usar el módulo en otras aplicaciones que no requiera el uso de funciones de predicción.

Se ha estudiado la posible utilización de *Hadoop Streaming* como middleware de soporte y se ha determinado no útil para este caso aunque se puede plantear otra solución con otro framework aparecido recientemente, *Apache Storm*, pensado para tiempo real.

Con todo ello, se espera que el proyecto realizado, con las adaptaciones, modificaciones, y extensiones que resulten necesarias, pueda ser utilizado en el

futuro por el grupo SID en sus investigaciones sobre procesamiento de flujos de datos.

## 5.2 Futuras ampliaciones

A continuación se establecen una serie de futuras ampliaciones que pueden dar cabida a proyectos futuros para otros estudiantes:

**1 Apache Storm**

A diferencia de *Apache Hadoop*, *Apache Storm* está pensado para sistemas en tiempo real. Podría ser interesante estudiar esa aproximación como middleware de soporte para la aplicación de procesamiento de *data streams*.

**2 Utilización de Mapas Reales**

En este proyecto se usan trazas reales, sin ninguna información de lugar ni de rutas posibles. Sería interesante integrar el simulador con algún sistema de rutas reales que permitan simular el movimiento en una carretera, ciudad etc…

**3 Creación de un sensor real que se comunique con la aplicación**

Una vez realizado el simulador y comprobada su viabilidad, podría ser interesante aprovechar el amplio uso de *smartphones* hoy en día, para crear un sensor real que se comunique por red (ya que también tienen conexión a internet). Para ello habría que adaptar el sistema de procesamiento para que admitiese tráfico de red y crear una aplicación móvil que recogiese los datos GPS, calculase la función de predicción (solo necesitaría almacenar como mínimo el dato anterior enviado) y enviase esta información al sistema de procesamiento.

**4 Ampliación de estrategias de movilidad y simuladores**

Con el diseño modular, añadir nuevas estrategias (e incluso simuladores) resulta sencillo, solo habría que conocer el modelo matemático (o lógico) del simulador e implementarlo para añadirlo a los existentes.

**5 Creación de un lenguaje de consultas similar a SQL**

Desarrolladas operaciones de agregación, y creación de consultas, se podría definir un lenguaje que permitiese interrogar al sistema de forma similar a como se interroga a un gestor de base de datos tradicional.

**6 Eliminación de las limitaciones del prototipo**

Una vez ampliado el prototipo, se podría considerar eliminar aquellas limitaciones existentes, pudiendo definir consultas VALUE independientes y que las funciones de agregación se apliquen a todos los datos y no solo a aquellos que se almacenan en el buffer.

**7 Estudios estadísticos**

Con el mantenimiento de los datos estadísticos, un posible trabajo futuro sería utilizar la aplicación (modificándola adecuadamente) para someterla a todo tipo de test para comprobar su rendimiento.

## 5.3 Valoración personal

Realizar un proyecto en solitario sobre un tema de investigación totalmente desconocido para mí ha sido enriquecedor ya que he conocido tecnologías e investigaciones alejadas a mi vida académica.

En un principio resulto algo frustrante ya que eran demasiados términos nuevos de los cuales no había tenido conocimiento a lo largo de la titulación, con lo cual eran sistemas a los que no estaba acostumbrado.

Tras la realización del primer prototipo de simulación el resto de las ampliaciones surgieron de forma más o menos fluidas ya que los términos ya no me eran ajenos, tenía una herramienta que al menos hacia una pequeña simulación tal y como ponían en los documentos de investigación leídos.

También ha sido un reto el comprender un proyecto de una envergadura mediana que había desarrollado otra persona. Coger el proyecto y averiguar cómo comunicar ambos programas ha sido el punto más crítico del proyecto.

El compaginar el proyecto con la vida laboral ha sido una ardua tarea, ya que disponía de poco tiempo para realizarlo, pero a la vez me ha servido para aplicar técnicas de gestión del tiempo y de la productividad.

Los objetivos se han cumplido en la gran mayoría y se ha creado una aplicación usable con las mismas funcionalidades del prototipo además de permitir una versatilidad en el simulador que antes no se tenía.

También he adquirido un mayor conocimiento del lenguaje Java y de modelos matemáticos de simulación. En definitiva, ha sido un buen colofón para terminar la titulación.