

Captura, representación, consulta y clasificación semántica de trayectorias

Autor: Cristina Martínez Pérez

Tesis doctoral UDC / 2022

Directores:

Miguel Ángel Rodríguez Luaces

Ángeles Saavedra Places

Programa Oficial de Doutoramento en Computación



**Tesis doctoral dirigida por
Miguel Ángel Rodríguez Luaces**

Departamento de Ciencias de la Computación y Tecnologías de la Información
Facultade de Informática
Universidade da Coruña
15071 A Coruña (España)
Tel: +34 981 167000 ext. 1254
Fax: +34 981 167160
miguel.luaces@udc.es

Ángeles Saavedra Places

Departamento de Ciencias de la Computación y Tecnologías de la Información
Facultade de Informática
Universidade da Coruña
15071 A Coruña (España)
Tel: +34 981 167000 ext. 1249
Fax: +34 981 167160
angeles.saavedra.places@udc.es

Miguel Ángel Rodríguez Luaces y Ángeles Saavedra Places, como directores, acreditamos que esta tesis cumple los requisitos para optar al título de doctor y autorizamos su depósito y defensa por parte de Cristina Martínez Pérez cuya firma también se incluye.

A mis padres y hermana

Agradecimientos

En primer lugar, quiero agradecer a la coordinadora del grupo de investigación, la catedrática Nieves Rodríguez Brisaboa, no solo todo el tiempo que ha dedicado a este trabajo de tesis, sino también el que me haya dado la oportunidad de formar parte del grupo de investigación en el Laboratorio de Bases de Datos. Asimismo, agradezco a mis directores de tesis, el doctor Miguel Ángel Rodríguez Luaces y a la doctora Ángeles Saavedra Places toda su ayuda, dedicación y esfuerzo a lo largo de esta tesis. Cada vez que he cruzado la puerta de vuestros despachos me he encontrado con una solución a cualquier problema u obstáculo encontrado en el camino y os estaré eternamente agradecida.

El período de realización de esta tesis ha sido una de las etapas más enriquecedoras hasta el momento y esto ha sido posible gracias al gran equipo de personas y profesionales que componen el LBD. Por ello, quiero dar gracias a todos/as mis compañeros/as, especialmente a Fernando, Álex, Tirso, Nelly, Suilen y Adrián, gracias por las comidas, los *breaks*, los viajes, las cenas cada mes, etc., os convertisteis en mi segunda casa. Gracias también a Carmen por toda la ayuda con gestiones administrativas, reservas, tarjetas de embarque, etc., pero sobre todo, gracias por las charlas y los cafés.

En tercer lugar, gracias a mis padres y hermana, por su apoyo incondicional, por confiar en mí siempre, darme ánimos y guiarme hasta convertirme en quien soy. Sois el lugar donde siempre quiero estar. Gracias también a Eduardo por animarme, apoyarme y acompañarme en cada camino que he escogido, pero también, perdón por todos los momentos en los que estuve ausente. Gracias por haber sido “casa” siempre, eres mi norte y mi sur.

Por último, gracias a mis amigas y amigos, por todos los momentos de desconexión y los consejos a lo largo de toda esta tesis, sois un gran tesoro del que me siento orgullosa.

Abstract

As a consequence of the competition between different manufacturers, current smartphones improve their features continuously and they currently include many sensors. *Mobile Workforce Management (MWM)* is an industrial process that would benefit highly from the information captured by the sensors of mobile devices. However, there are some problems that prevent MWM software from using this information: i) the abstraction level of the activities currently identified is too low (e.g., *moving* instead of *performing an inspection on a client*, or *stopped* instead of *loading a truck in the facility of a client*; ii) research work focuses on using geographic information algorithms on GPS data, or machine learning algorithms on sensor data, but there is little research on combining both types of data; and iii) context information extracted from geographic information providers or MWM software is rarely used.

In this thesis, we present a new methodology to turn raw data collected from the sensors of mobile devices into trajectories annotated with semantic activities of a high level of abstraction. The methodology is based on *activity taxonomies* that can be adapted easily to the needs of any company. The activity taxonomies describe the expected values for each of the variables that are collected in the system using predicates defined in a *pattern specification language*. We also present the functional architecture of a module that combines context information retrieved from MWM software and geographic information providers with data from the sensors of the mobile device of the worker to annotate their trajectories and that can be easily integrated in MWM systems and in the workflow of any company.

Resumen

Como consecuencia de la competencia entre los diferentes fabricantes, los *smartphones* actuales presentan continuamente mejoras en sus características y en la actualidad incluyen numerosos sensores. *Mobile Workforce Management (MWM)* es un proceso industrial que podría beneficiarse mucho de la información recogida por los sensores de los teléfonos móviles. Sin embargo, existen varios problemas que lo impiden: i) hoy en día el nivel de abstracción de las actividades que son identificadas es demasiado bajo (por ejemplo, *moviéndose* en vez de *realizando una inspección en un cliente*, o *parado* en vez de *cargando un camión en la instalación de un cliente*); ii) los *trabajos* de investigación se centran en el uso de algoritmos que contrastan la información geográfica con los datos del GPS, o en algoritmos de aprendizaje aplicados a los datos de los sensores, pero existen pocos resultados de investigación que combinen ambos tipos de datos; y iii) la información contextual procedente de los repositorios de información geográfica o del software MWM es raramente usada.

En esta tesis se presenta una nueva metodología que convierte los datos crudos capturados por los sensores de los dispositivos móviles en trayectorias anotadas con actividades semánticas con un alto nivel de abstracción. La metodología está basada en la definición de *taxonomías de actividades* que pueden ser adaptadas fácilmente a las necesidades de cualquier empresa. Estas taxonomías describen los valores esperados para cada una de las variables que son recogidas en el sistema usando predicados definidos mediante un *lenguaje de especificación de patrones*. Por último, también se describe la arquitectura del sistema que combina la información contextual del software MWM y las fuentes de información geográfica con los datos de los sensores del dispositivo móvil del trabajador para así, finalmente, anotar sus trayectorias y que pueden integrarse fácilmente en sistemas MWM y en el flujo de trabajo de cualquier empresa.

Resumo

Como consecuencia da competencia entre os diferentes fabricantes, os *smartphones* actuais presentan continuamente melloras nas súas características e na actualidade inclúen numerosos sensores. *Mobile Workforce Management (MWM)* é un proceso industrial que podería beneficiarse moito desta información recollida polos sensores dos teléfonos móbiles. Con todo, existen varios problemas que o impiden: i) hoxe en día o nivel de abstracción das actividades que son identificadas é demasiado baixo (por exemplo, *movéndose* en lugar de *realizando unha inspección nun cliente*, ou *parado e cargando un camión na instalación dun cliente*); ii) os traballos de investigación céntranse no uso de algoritmos que contrastan a información xeográfica cos datos do GPS, ou en algoritmos de aprendizaxe aplicados aos datos dos sensores, pero existen poucos resultados de investigación que combinen ambos os tipos de datos; e iii) a información contextual procedente dos repositorios de información xeográfica ou do software MWM é raramente usada.

Nesta tese preséntase unha nova metodoloxía que converte os datos crus capturados polos sensores dos dispositivos móbiles en traxectorias anotadas con actividades semánticas cun alto nivel de abstracción. A metodoloxía está baseada na definición de *taxonomías de actividades* que poden ser adaptadas ás necesidades de calquera empresa dun xeito doado. Estas taxonomías describen os valores esperados para cada unha das variables que son recollidas no sistema usando predicados definidos mediante unha *linguaxe de especificación de patróns*. Por último, tamén se describe a arquitectura do sistema que combina a información contextual do software MWM e as fontes de información xeográfica cos datos dos sensores do dispositivo móbil do traballador para así, finalmente, anotar as súas traxectorias e que poden integrarse dun xeito doado en sistemas MWM e no fluxo de traballo de calquera empresa.

Índice general

1. Introducción	1
1.1. Motivación	1
1.2. Contribuciones de esta tesis	3
1.3. Estructura de la tesis	5
2. Estado del arte	7
2.1. Metodología de revisión sistemática de la literatura	7
2.2. Revisión sistemática de la literatura	8
2.3. Resultados de la revisión sistemática de la literatura	11
2.3.1. Campos de aplicación de la identificación de actividades semánticas	11
2.3.2. Representación de las trayectorias semánticas	12
2.3.3. Técnicas de enriquecimiento semántico	13
2.4. Conclusiones	18
3. Conceptos básicos	19
3.1. Captura de datos de sensores	19
3.1.1. Captura de datos de localización	19
3.1.2. Captura de datos de movimiento	22
3.2. Técnicas de análisis de información geográfica	25
4. Sistema de identificación de actividades semánticas	33
4.1. Arquitectura funcional del sistema	33
4.2. Componente de captura de datos (<i>SensorCollector</i>)	35
4.3. Componente de recogida de datos (<i>SensorReceiver</i>)	42
4.4. Componente de anotación semántica (<i>Annotator</i>)	45
4.5. Conclusiones	52
5. Metodología de anotación de actividades semánticas	55
5.1. Taxonomía de actividades	55
5.1.1. Definición de una taxonomía de actividades	56

5.1.2. Representación formal de una taxonomía de actividades . . .	60
5.2. Lenguaje de especificación de patrones de actividad	64
5.3. Anotación de actividades	68
5.4. Conclusiones	69
6. Evaluación del sistema	71
6.1. Evaluación experimental	71
6.1.1. Conjuntos de datos	71
6.1.2. Experimento con <i>ParadoYMovimiento</i>	73
6.1.3. Experimento con <i>Decisión</i>	75
6.1.4. Conclusiones	78
6.2. Prueba de concepto	79
6.2.1. Identificación de actividades en GESUGA	80
6.2.2. Identificación de actividades en Mayores	87
6.3. Conclusiones	94
7. Conclusiones	95
7.1. Conclusiones y aportaciones principales	95
7.2. Lineas de trabajo futuro	97
Bibliografía	99

Índice de figuras

1.1. Ejemplo de trayectoria semántica	2
2.1. Aplicación de la metodología de la revisión sistemática de la literatura	10
3.1. Proceso de trilateración	20
3.2. Sistema de coordenadas utilizado por el <i>framework</i> de sensores . . .	24
3.3. Proceso de modelado en un sistema de información	26
3.4. Tipos de datos de <i>Simple Feature Access</i> , extraído de [ISO04]	28
3.5. Ejemplos de valores de los tipos de datos de <i>Simple Feature Access</i> .	29
3.6. Borde, interior y exterior de objetos geográficos	29
3.7. Cambios en la relación topológica de los objetos se reflejan en la matriz calculada por el <i>Dimensionally Extended Nine-Intersection Model (DE9IM)</i>	30
3.8. Resumen de los predicados espaciales	31
4.1. Arquitectura funcional	34
4.2. Arquitectura del componente de captura de datos (<i>SensorCollector</i>)	36
4.3. Diagrama de clases del componente de captura de datos (<i>SensorOb- server</i>)	37
4.4. Diagrama de clases de la etapa de procesamiento de datos y preparación del envío (<i>SensorProcessor</i>)	38
4.5. Modelo de datos de los ficheros JSON preparados para el envío . . .	39
4.6. Diagrama de clases del componente de envío de datos (<i>SensorSender</i>)	41
4.7. Arquitectura del componente de recogida de datos (<i>SensorReceiver</i>)	42
4.8. Diagrama de clases del componente de recogida de datos (<i>SensorRe- ceiverr</i>)	43
4.9. Datos de sensores y ubicaciones registrados por el dispositivo móvil .	43
4.10. Modelo conceptual de la tabla de registro de incidencias	44
4.11. Arquitectura del componente de anotación semántica (<i>Annotator</i>) .	46
4.12. Diagrama de clases de un Segmento	47
4.13. Lista de ubicaciones y datos de sensores	48

4.14. Modelo de datos para la información procedente del sistema MWM	50
4.15. Modelo de datos para la información de contexto	51
4.16. Modelo de datos para las actividades identificadas	52
5.1. Taxonomía clasificación animales vertebrados e invertebrados	56
5.2. Ejemplo de taxonomía usando <i>Decisión</i> y <i>Etiquetado</i>	57
5.3. Ejemplo de taxonomía usando el nodo <i>Agrupar</i>	58
5.4. Ejemplo de taxonomía usando el nodo <i>ParadaYMovimiento</i>	59
5.5. Diagrama de clases de una taxonomía de actividades	60
5.6. Predicados simples	65
5.7. Predicados complejos	66
5.8. Predicado de combinación lógica para formar predicados complejos	67
6.1. Taxonomía de actividades para el experimento con <i>ParadoYMovimiento</i>	73
6.2. Taxonomía de actividades para el experimento con <i>Decisión</i>	76
6.3. Taxonomía de actividades para GESUGA	83
6.4. Esquema de funcionamiento de la solución mixta	84
6.5. Interfaz de explotación de las actividades identificadas en GESUGA	86
6.6. Colocación de receptores y <i>beacons Bluetooth</i> en Mayores	88
6.7. Taxonomía de actividades para residentes en Mayores	90
6.8. Taxonomía de actividades para auxiliares en Mayores	92
6.9. Interfaz de explotación de las actividades identificadas en Mayores	93

Índice de tablas

2.1. Criterios de inclusión	9
2.2. Selección de términos de búsqueda	9
6.1. Descripción de los datos recogidos en el experimento	72
6.2. Resultados de la identificación preliminar de actividades realizadas en el trabajo de investigación [GGRFBL20]	73
6.3. Resultados del experimento con <i>ParadoYMovimiento</i>	74
6.4. Resultados del experimento con <i>Decisión</i>	77

Lista de algoritmos

4.1. Obtener los segmentos a partir de ubicaciones GPS y datos de sensores	49
5.1. Algoritmo de TagNode	61
5.2. Algoritmo de DecisionNode	61
5.3. Algoritmo de GroupNode	62
5.4. Algoritmo de StopAndMoveNode	63
5.5. Algoritmo Annotator	69

Capítulo 1

Introducción

1.1. Motivación

En la última década el uso de dispositivos móviles como *smartphones*, *tablets*, y *wearable devices* ha crecido incesantemente tanto en el ámbito personal como laboral y más de un 60% de las personas que viven en economías avanzadas posee un *smartphone* [Pou16]. Como consecuencia, las características de dispositivos móviles como *smartphones*, *tablets* y *wearable devices* han ido incrementándose constantemente. En los últimos años su poder computacional es comparable al de un ordenador de escritorio, y estos dispositivos incluyen múltiples sensores que pueden ser usados para medir diferentes variables como la posición geográfica usando el GPS, la actividad del usuario usando el acelerómetro, o información del entorno usando por ejemplo el termómetro. Un proceso industrial que podría beneficiarse especialmente del uso de esta información capturada por los dispositivos móviles es *Mobile Workforce Management (MWM)*. Los sistemas MWM son usados por las compañías para manejar y optimizar la programación de tareas de sus trabajadores (por ejemplo, asegurarse de que una empresa tenga el mínimo número de empleados activos desplegados en cualquier momento de la jornada laboral) y para mejorar la actuación de sus procesos de negocio (por ejemplo, detectar qué tareas son costosas para una compañía). Como un ejemplo, sería beneficioso para una empresa que recoge residuos orgánicos en zonas rurales detectar si los trabajadores invierten demasiado tiempo en la actividad repostar, en ese caso sería una buena idea decidir que se contrate a alguien para realizar esta tarea por las noches.

Este tipo de estudios son imposibles de llevar a cabo sin un histórico de los datos de los movimientos de los trabajadores y un procedimiento para detectar y analizar, a un alto nivel de abstracción, las actividades que los empleados llevan a cabo y cuanto tiempo invierten en cada una de ellas. Si el sistema MWM puede detectar qué ha pasado realmente en la jornada laboral y lo compara con lo que ha

siendo previamente planificado, entonces se podrá generar información muy útil para gestionar procesos de negocio y detectar aquellos puntos críticos. Sin embargo, los sistemas MWM actuales no usan la información que es recogida por los dispositivos móviles. A lo sumo, usan los servicios de localización para registrar cuando llega a la ubicación del cliente, pero no hacen uso de la gran variedad de datos recogidos por los sensores de los dispositivos móviles para responder a consultas tales como *¿Estaba el trabajador en un atasco y por eso llegó tarde?* o *¿Cuánto tiempo invierte en cada visita a la empresa del cliente?*, ya que se trata de datos masivos y de gran complejidad.



Figura 1.1: Ejemplo de trayectoria semántica

El motivo que evita el uso de los datos generados por estos sensores en software MWM es que son muy voluminosos y complejos. El campo de investigación en *trayectorias semánticas* [PSR⁺13] focaliza su interés en los problemas de recolección, modelado, carga y análisis de estos datos. Una trayectoria semántica es un conjunto de segmentos, donde cada segmento tiene asignado una hora de inicio, una hora de fin y una etiqueta de valor semántico, en la Figura 1.1 se muestra un ejemplo de trayectoria anotada semánticamente. Cada segmento que forma una trayectoria semántica representa una actividad realizada. En el momento que se obtienen este conjunto de actividades se conoce en cada momento dónde y qué están haciendo los empleados de la empresa, pudiendo responder a preguntas como las planteadas anteriormente.

Una parte de las técnicas propuestas en el campo de investigación en trayectorias semánticas se han centrado en el uso de técnicas de aprendizaje máquina para la detección de actividades. Sin embargo, no es posible aplicar estas técnicas en entornos MWM ya que resultan muy costosas debido a que requieren la recogida de datos para entrenar al sistema. En un entorno MWM esto implicaría que los

empleados deberían anotar cada vez que realizan una tarea el instante de inicio y fin, un proceso muy tedioso y costoso a la vez. Además se añade un gran margen de error, es muy difícil que un empleado se ocupe de realizar su trabajo y al mismo tiempo recoja los datos de muestra para el conjunto de entrenamiento es posible que se olvide de etiquetar el inicio o el fin de una actividad y por lo tanto, el conjunto de datos de muestra sea erróneo, sin tener en cuenta que esta labor estaría interfiriendo en la adecuada realización de su trabajo.

Por otra parte, los resultados de investigación en trayectorias semánticas que realizan un análisis de la información mediante técnicas tradicionales tampoco se pueden utilizar directamente ya que se centran en la detección de actividades a bajo nivel tales como *parado*, *en movimiento*, *caminando*, o *corriendo* en lugar de actividades de alto nivel, tales como *parado en un atasco de tráfico en el camino a la instalación del cliente*, o *cargando el camión en la instalación del cliente*. El motivo de la diferencia de estos niveles de abstracción es que la investigación en trayectorias semánticas se centra en el caso general y hace un uso limitado de información del contexto, mientras que en un sistema MWM la información de contexto (por ejemplo, la agenda del trabajador) es extremadamente significativa.

Por todo ello, el objetivo de esta tesis es presentar una metodología basada en taxonomías de actividades para el anotado de trayectorias en entornos MWM. Este sistema puede ser utilizado para recoger la información capturada por los sensores de los dispositivos móviles, analizar y anotar la trayectoria con actividades con un alto nivel de abstracción utilizando información del contexto del sistema MWM y proveedores de información geográfica, y finalmente proporcionar la información de la trayectoria semántica de nuevo al sistema MWM con el fin de apoyar el proceso de inteligencia de negocios. El resto de este documento se organiza de la siguiente manera.

1.2. Contribuciones de esta tesis

Esta tesis se centra en explorar un campo de investigación en auge y que ofrece un gran abanico de posibilidades: la identificación de actividades semánticas en entornos MWM. Las aportaciones de esta tesis pueden resumirse en dos. La primera, aportar una arquitectura de un sistema completo para la identificación de actividades semánticas de trabajadores en movilidad que puede ser integrado en los procesos de negocio de cualquier empresa mediante la interacción con su sistema MWM (*Mobile Workforce Management*). La segunda, definir una metodología flexible e intuitiva para la identificación de actividades semánticas.

Respecto a la primera contribución, aportamos una arquitectura que cumple con los requisitos iniciales necesarios para su empleo en empresas reales que utilizan una metodología MWM. Esta arquitectura incluye desde la aplicación móvil de recogida de datos hasta la identificación de las actividades, pasando por el almacenamiento de los datos crudos y la interacción con el sistema MWM. Concretamente, esta tesis

aporta una arquitectura que cumple con las siguientes premisas:

- Mínima transferencia de datos recogidos por el componente de captura de datos en el dispositivo móvil.
- Alta eficiencia energética en su funcionamiento.
- Robustez frente a problemas de conectividad. La arquitectura construida presenta un diseño que permite no tener conexión permanente con el sistema central.
- Datos unificados procedentes de diversas fuentes, desde la información de contexto hasta los datos de los sensores y la ubicación GPS. Además, el diseño de la arquitectura permite dar solución al problema de las distintas granularidades que presentan los sensores y los datos de localización, permitiendo agregarlos y fusionarlos.

Respecto a la segunda contribución, una metodología de identificación de actividades semánticas, aportamos un componente de detección de actividades semánticas que alcanza tres objetivos fundamentales para que su uso sea posible en empresas reales:

- Definimos un modelo simple e intuitivo que permite representar las actividades semánticas. Para ello, proponemos el concepto de *taxonomía de actividades* junto con los conceptos necesarios para definirlas (esto es, tipos de nodos de procesamiento).
- Definimos un *lenguaje de especificación de patrones* para representar los predicados utilizados en las decisiones que se toman en la taxonomía. Este lenguaje es muy expresivo y tiene en cuenta no sólo los datos del sensor sin procesar sino también los datos almacenados en el sistema MWM, y de la información geográfica del contexto relacionada con el dominio.
- Definimos un proceso para la detección de actividades semánticas con un alto nivel de abstracción semántico y con un grado de precisión elevado en comparación con otros trabajos del estado del arte.

Por último, cabe destacar, que esta tesis aporta cuatro experimentos, dos pruebas experimentales en entornos reales para la identificación de actividades básicas, y dos pruebas de concepto en dos empresas reales en las que se identifican actividades de alto nivel de abstracción semántica. En cuanto a las dos primeras pruebas experimentales, cabe señalar que los resultados obtenidos se compararon con los de otro trabajo de investigación del estado del arte [GGRFBL20] e incluso han mejorado sus resultados. Por último, en cuanto a las pruebas de concepto, cabe resaltar que los resultados obtenidos han sido óptimos y ambas empresas han conseguido explotar la información que engloban los datos crudos de las trayectorias sin procesar, permitiendo realizar consultas filtrando por empleado, fecha, visualizar trayectorias, tareas realizadas, etc.

1.3. Estructura de la tesis

En el Capítulo 2 se describe el trabajo previo relacionado, se detalla la metodología de revisión sistemática de la literatura que se ha seguido para estudiar el estado del arte, y por último, se describen los resultados obtenidos de esta revisión, concretamente, se describe el estado del arte de los campos de aplicación de la identificación de actividades semánticas, la representación de las trayectorias semánticas, y por último, las técnicas de enriquecimiento semánticos empleadas en el literatura hasta el momento.

En el Capítulo 3 se describen los conceptos básicos en el área de captura de datos de sensores y se abordan las diferentes técnicas de tratamiento de información geográfica. En una primera parte se describen las técnicas empleadas para recoger datos de los sensores de los dispositivos móviles y de ubicación mediante GPS en *Android*. A continuación, en una segunda parte de este capítulo, se describen las técnicas de modelado de información geográfica, de los predicados y operadores de procesamiento de este tipo de información, y de las fuentes de datos de utilidad para la detección de actividades de interés

En el Capítulo 4 se muestra la arquitectura funcional del sistema de identificación de actividades semánticas de trabajadores en movilidad, detallando el funcionamiento, la arquitectura y requisitos de cada uno de los componentes que lo integran.

En el Capítulo 5 se describe la metodología de anotación y las taxonomías de actividades que forman la base de la metodología. Además, se detalla el proceso de anotación y se describen los diferentes algoritmos empleados en el proceso de etiquetado. Por último, se detalla el lenguaje de especificación de patrones de actividad que se utiliza en las taxonomías para decidir qué actividad se está realizando.

En el Capítulo 6 se describen los experimentos realizados. En primer lugar, se describe las evaluaciones experimentales de la metodología utilizando los datos recogidos por voluntarios de nuestro grupo de investigación. En segundo lugar, se detallan dos pruebas de concepto aplicando la metodología a dos empresas reales e identificando actividades de interés para su día a día.

Finalmente, en el Capítulo 7 se describen nuestras conclusiones, detallando las aportaciones principales del trabajo realizado. Para concluir, se describen las líneas de trabajo futuro que abre esta tesis.

Capítulo 2

Estado del arte

2.1. Metodología de revisión sistemática de la literatura

Para elaborar el estado del arte de esta tesis realizamos una revisión sistemática de la literatura (SLR, del inglés *systematic literature review*). Una SLR es una forma de estudio secundario que usa una metodología bien definida para identificar, analizar e interpretar todas las evidencias relacionadas con una pregunta de investigación específica de una forma que es imparcial y repetible ([KC07]). Una SLR recopila y analiza críticamente múltiples estudios o trabajos de investigación a través de un proceso sistemático para identificar, evaluar e interpretar el trabajo de investigadores, académicos y profesionales en un campo elegido ([Fri02]). En definitiva, el objetivo de una SLR es proporcionar un resumen exhaustivo de la literatura disponible pertinente a una o varias preguntas de investigación.

Para llevar a cabo esta revisión sistemática de la literatura hemos seguido la siguiente metodología inspirada en [GP17], que a su vez se basa en los trabajos [Shu11, LSLB⁺11].

1. **Definición de las preguntas de investigación.** El primer paso consiste en definir cuáles eran las preguntas de investigación (y objetivos) a las que buscamos obtener respuesta.
2. **Definición de criterios de inclusión y exclusión.** En el segundo paso seleccionamos los criterios de exclusión e inclusión que se van a aplicar al resultado obtenido en la búsquedas.
3. **Identificación de bases de datos.** En el tercer paso, identificamos las bases de datos y motores de búsqueda que se van a utilizar.

4. **Selección de términos de búsqueda.** En el cuarto paso, realizamos varias pruebas con diferentes términos de búsqueda hasta que el resultado obtenido tenga un número suficiente de trabajos de investigación para realizar una revisión sistemática de la literatura completa y exhaustiva.
5. **Búsqueda y revisión de los resultados.** Una vez completados los anteriores pasos obtenemos el conjunto final de artículos a analizar. En este paso se revisa cada artículo, siguiendo el orden de su fecha de publicación, para verificar que el artículo tiene relación con nuestra área y/o con nuestro objetivo a estudiar. Esta revisión se hace en primer lugar sobre el título y el resumen, y a continuación sobre el texto completo mediante una lectura preliminar.
6. **Lectura y análisis de los resultados.** En esta fase se leen y se analizan los artículos seleccionados para dar respuesta a las preguntas de investigación planteadas. En primer lugar se revisan los artículos que a nuestro juicio tienen un estudio del estado del arte más completo y reciente, y que por lo tanto describen un problema más cercano a nuestra área de estudio. Esto permite construir una base sólida desde de la que comenzar a responder las preguntas de investigación y proporciona en los primeros pasos del análisis nuevas referencias que añadir al conjunto de resultados siguiendo el último punto de la metodología.
7. **Inclusión de artículos a partir de las referencias de los resultados más sobresalientes** Al leer y analizar los artículos, se añadieron nuevos artículos de sus referencias que se consideraron interesantes. De esta forma el proceso es iterativo y asegura que se cubre el campo de estudio de la forma más exhaustiva posible.

2.2. Revisión sistemática de la literatura

Las preguntas de investigación a las que se busca dar respuesta con la revisión sistemática de la literatura son las que siguen:

- **Q1. ¿Cuales son los campos de aplicación de la identificación de actividades semánticas?** Con esta pregunta pretendemos saber cuáles son los campos de aplicación de la identificación de actividades semánticas, qué utilidades tienen y qué líneas de investigación están abiertas.
- **Q2. ¿Cómo se representan las trayectorias semánticas?** Con esta pregunta pretendemos averiguar qué son y cómo se representan conceptualmente una trayectoria y una trayectoria semántica respectivamente.
- **Q3. ¿Qué actividades se identifican y cómo se identifican?** Con esta pregunta pretendemos conocer qué actividades identifican los trabajos de

investigación, qué métodos emplean y qué información y pasos utilizan para hacerlo.

Los criterios de inclusión definidos son los que se especifican en la tabla 2.1.

Identificador	Descripción
CI1	El artículo está publicado en un congreso o revista de reconocido prestigio
CI2	El artículo es posterior a 2008.
CI3	El artículo responde a las preguntas de investigación
CI4	El artículo incluye pruebas/evaluación sobre un conjunto de datos

Tabla 2.1: Criterios de inclusión

El criterio de exclusión ha sido excluir aquellos artículos que no estuviesen relacionados con las áreas de investigación relevantes (*Computer Science, Engineering, Decision Sciences y Environmental Science*).

La bases de datos y motor de búsqueda seleccionada ha sido *Scopus*. En una inspección preliminar se comprobó que los resultados eran buenos, y utilizar una única bases de datos evita tener que procesar los resultados para eliminar duplicados. Además, *Scopus* proporciona resultados más precisos con menos ruido en comparación con otras bases de datos, tiene un lenguaje de consulta más flexible y permite guardar las consultas.

Una vez definidos los criterios, hicimos varias pruebas con términos de búsqueda para determinar el que devolviera un conjunto de resultados suficientemente amplio para empezar. La tabla 2.2 muestra las pruebas realizadas junto con su número de resultados. En todos los casos se realizó la búsqueda en el título y en las palabras clave de los artículos utilizando la sintaxis *TITLE-ABS-KEY* (*palabras clave*) de *Scopus*. Decidimos seleccionar el último conjunto de términos de búsqueda (es decir, *TITLE-ABS-KEY* (*semantic AND trajectory AND mobility*))

Palabras clave	Resultados
<i>activity AND trajectories AND sensor AND data AND moving</i>	48
<i>activity AND trajectories AND sensor AND moving</i>	81
<i>semantic AND trajectory AND annotation</i>	96
<i>semantic AND trajectory AND mobility</i>	186

Tabla 2.2: Selección de términos de búsqueda

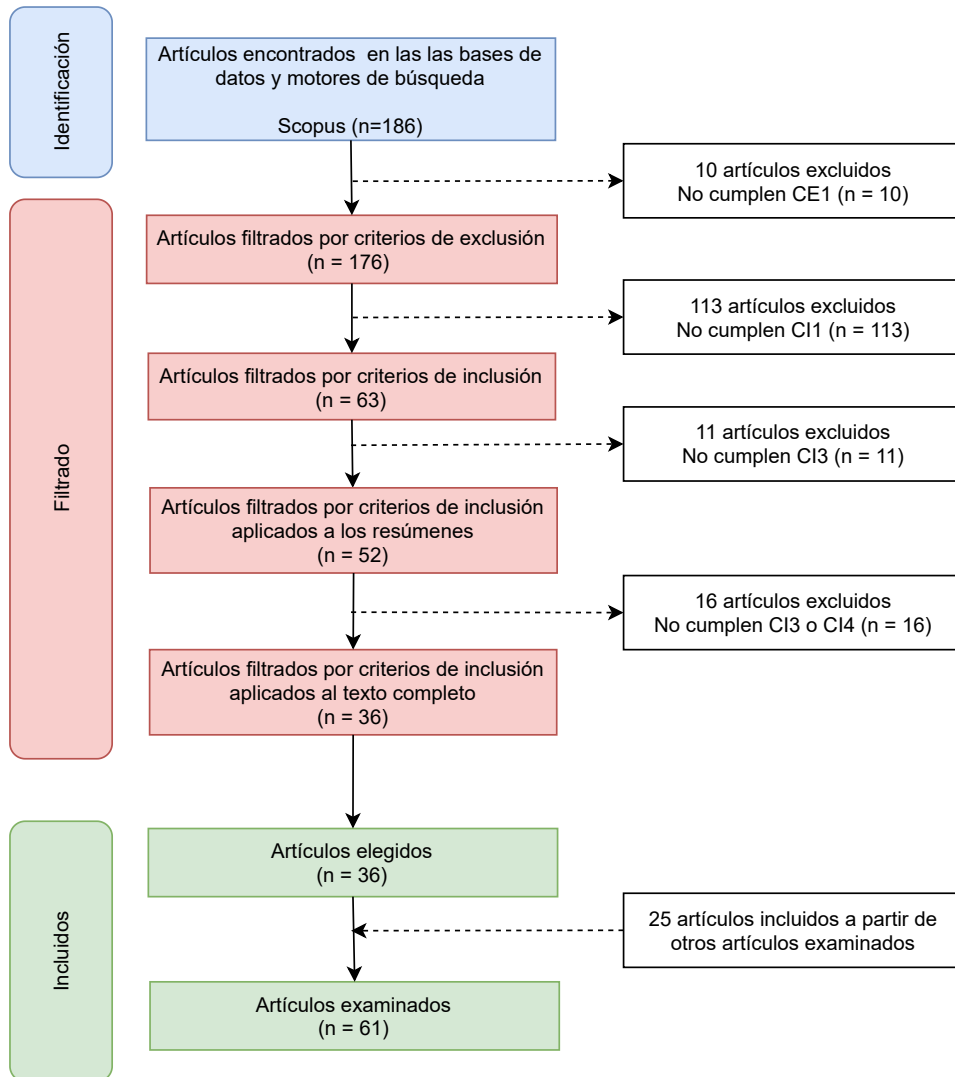


Figura 2.1: Aplicación de la metodología de la revisión sistemática de la literatura

2.3. RESULTADOS DE LA REVISIÓN SISTEMÁTICA DE LA LITERATURA 11

La figura 2.1 muestra los resultados parciales del proceso de aplicación de la metodología de la revisión sistemática de la literatura. En la parte superior se muestran los 186 artículos encontrados en *Scopus* utilizando los términos de búsqueda. El filtrado usando los criterios de exclusión (áreas de investigación) eliminó 10 artículos dejando un total de 176. A continuación, el filtrado aplicando el criterio de inclusión CI1, eliminó 113 artículos, dejando un total de 63. Posteriormente se aplicó el criterio de inclusión CI3 sobre los resúmenes lo cual eliminó 11 artículos dejando un total de 52 artículos. La referencia bibliográfica y el texto completo de cada uno de estos 52 artículos se almacenó en un gestor documental (Zotero) y se revisó cada uno de los artículos para aplicar de nuevo los criterios de inclusión CI3 y CI4 sobre los artículos eliminando 16 artículos dejando un total de 36 artículos. Cada uno de estos artículos se analizó de forma detallada intentando responder a cada una de las preguntas de investigación planteadas en la metodología. De la revisión preliminar de resúmenes y contenidos se identificaron tres artículos recientes que se centran en estudiar el estado del arte [SA21, CXC⁺20, PSR⁺13]. El trabajo de análisis detallado comenzó por estos artículos, lo que permitió añadir un total de 25 artículos adicionales de relevancia, lo que dejó el total de artículos revisados en 61. El resto de artículos se analizó por orden cronológico descendente.

2.3. Resultados de la revisión sistemática de la literatura

2.3.1. Campos de aplicación de la identificación de actividades semánticas

En relación a la pregunta de investigación Q1, *¿Cuales son los campos de aplicación de la identificación de actividades semánticas?*, de la revisión de [SA21, CXC⁺20, PSR⁺13] constatamos que en los últimos años la explotación de las trayectorias semánticas presenta un nuevo y creciente campo de investigación en la ciencia de la información geográfica, ya que permite obtener una mayor comprensión de la movilidad de usuarios y objetos. Las trayectorias sin procesar son suficientes para aplicaciones que tienen como objetivo localizar algunos usuarios u objetos en movimiento, por ejemplo, *¿dónde estaba María a las 8:00 del 8 de marzo de 2012?* o *¿qué porcentaje de trayectorias muestran una velocidad media superior a 10 km/h?*. Sin embargo, la mayoría de los análisis requieren consultas más complejas. Para ello es necesario enriquecer los datos brutos con información adicional del contexto de la aplicación. Imaginemos, por ejemplo, que queremos interpretar las trayectorias de las personas dentro de una ciudad. Utilizando información adicional de esa ciudad en concreto, por ejemplo, la cartografía y/o puntos de interés de la ciudad en cuestión, las ubicaciones GPS sin ningún contenido semántico aparente se pueden reemplazar por nombres de calles y/o con nombres de lugares de interés

como tiendas, restaurantes, museos, o con información sobre eventos en curso como conciertos. De esta manera, es posible enriquecer semánticamente cada una de esas ubicaciones GPS.

Existen numerosos trabajos de investigación que se centran en detectar actividades de personas desplazándose por una ciudad tales como estar en casa, ir de compras, hacer deporte, comer en un restaurante, etc. Entre ellos podemos encontrar [SLC⁺21, SA21, MFdAL21, LZX⁺21, FPA⁺20, CXC⁺20, JTL⁺20, PTTHGZ19, BGWSB19, BZL18, CLL18b, ZKSN17, MPLK16, PSR⁺13, YCP⁺13a]. Otros trabajos como [SGP⁺20, HZC⁺20, VSD⁺19] analizan las trayectorias de viajes en barco o avión. Por otra parte, estas y otras investigaciones van un paso más allá y además de detectar actividades, explotan la información semántica de las trayectorias permitiendo encontrar patrones de movilidad y comportamiento. Algunos de estos trabajos son por ejemplo [HZC⁺20, CXC⁺20, LZD⁺19, CCX⁺18, BZL18, CLL18b, CLL18a, WZYS18, TZDC18, MPLK16, GVD15, GLL⁺16, LCX⁺16, TZY⁺13]. Otros estudios buscan anticiparse y predecir el futuro en base a las actividades detectadas en el pasado. Por ejemplo, [GSY⁺21] predice qué destino tiene un vehículo, y [BZL18, GZLH18, WLF⁺17] predicen qué lugares de interés van a ser más visitados. Otros trabajos como [WMC21] sugieren posibles rutas o lugares de interés. [HGW⁺21] indica la relación que existe entre el tráfico en las carreteras y su funcionalidad. [CCX⁺18] intenta identificar qué riesgos laborales existen en el interior de un edificio y detecta en qué estancia se encuentra un trabajador para informarle sobre estos y así evitar accidentes de trabajo. [MPLDSE⁺20] infiere el modo de transporte de un usuario a partir de los datos recogidos de su ruta. [WYH⁺19] estudia cómo influye la demografía en las trayectorias de las personas. [WYC⁺19] busca qué lugares son más transitados. Por último, otros estudios también permiten realizar consultas sobre estas trayectorias, por ejemplo [WYC⁺19, GVD15]. Otros como por ejemplo [GVD15], definen un lenguaje de consultas para buscar patrones de movilidad.

En definitiva, en los últimos años el uso creciente de Internet, ordenadores, tabletas, móviles, etc., ha permitido que numerosos estudios comiencen a explotar la gran cantidad de datos que forman las trayectorias de los usuarios y objetos en movimiento, permitiendo en esta línea explorar un gran abanico de posibilidades y explotar un gran campo de aplicaciones, como por ejemplo, encontrar patrones de movimiento y comportamiento, predecir destinos de turistas, sugerir lugares de interés, evitar accidentes laborales, analizar rutas de barco y aviones, entre otras muchas aplicaciones.

2.3.2. Representación de las trayectorias semánticas

En relación a la pregunta planteada Q2, *¿Cómo se representan las trayectorias semánticas?*, de la revisión de [SA21, CXC⁺20, PSR⁺13] se deduce que no existe una única definición de trayectoria. Sin embargo la más común es presentada en [CXC⁺20]: una *trayectoria* se define como una colección de puntos GPS donde

2.3. RESULTADOS DE LA REVISIÓN SISTEMÁTICA DE LA LITERATURA 13

cada uno tiene asociado una marca temporal. Esta misma definición la comparten numerosos trabajos de investigación, como por ejemplo [HGW⁺21, SLC⁺21, SA21, MFdAL21, LZX⁺21, SGP⁺20, MPLDSE⁺20, PTHGZ19, BGWSB19, LZD⁺19, WYC⁺19, CCX⁺18, BZL18, JC18, CLL18b, CLL18b, GZLH18, WZYS18, SPT16, GS15, YCP⁺13a]. La mayoría de estos trabajos hacen referencia a este conjunto de datos con el término *raw trajectory*, que podría traducirse como *trayectoria sin procesar* [SA21, LZX⁺21, SGP⁺20, MPLDSE⁺20, WZYS18, SPT16, GS15, PSR⁺13].

Otra definición interesante que encontramos en la literatura es la de *proceso de enriquecimiento semántico*, el cual se puede definir como el proceso que agrega conocimiento a las trayectorias en bruto. El enriquecimiento se basa en la idea de que los datos existentes se complementan con datos adicionales, llamados anotaciones. Una anotación es cualquier dato adicional que se adjunta a una trayectoria en su conjunto o a cualquiera de sus subpartes. Por ejemplo, registrar el objetivo del viaje de una persona a París por negocios o turismo, representa una anotación, [GMSK08].

Como resultado de este proceso de enriquecimiento semántico, nos encontramos con otra definición muy frecuente en la literatura, la de *trayectoria semántica*. Este concepto lo utilizan multitud de trabajos de investigación, entre ellos, [SA21, SGP⁺20, CXC⁺20, VSD⁺19, CLL18b, CLL18a]. Todas estas investigaciones coinciden en que una trayectoria semántica está formada por un conjunto de ubicaciones, donde cada una tiene asociada una marca temporal y una o varias anotaciones que describen la información del contexto. Por ejemplo, [PSR⁺13, YCP⁺13a, Zhe15] definen una trayectoria semántica como una trayectoria enriquecida con anotaciones que aportan conocimiento semántico sobre los movimientos de objetos o usuarios. Estas anotaciones contienen muchos tipos de información contextual. Por ejemplo, la anotación “ir en bus a la escuela” describe el movimiento de un usuario que emplea como medio de transporte el bus y tiene como destino la escuela. Otros trabajos de investigación hacen referencia al mismo concepto, sin embargo, utilizan otros términos para referirse a este conjunto de datos. Por ejemplo, [MPLDSE⁺20] lo denomina *multiple-aspect trajectory* o [GVD15] emplea el término *symbolic trajectory*.

2.3.3. Técnicas de enriquecimiento semántico

En relación a la pregunta planteada Q3, *¿Qué actividades se identifican y cómo se identifican?*, tras el análisis de los resultados más sobresalientes, concluimos que, aunque es complicado extraer información semántica de los datos crudos sin procesar, existen diferentes técnicas que extraen información semántica de la secuencia de datos GPS.

La primera conclusión que se puede extraer de la revisión de la literatura es que anotar cada uno de los puntos GPS de la trayectoria no es eficiente ya que puede generar una gran cantidad de anotaciones repetitivas. Por ello, las trayectorias

pueden ser anotadas en diferentes niveles de detalle: puntos GPS individuales, episodios (partes significativas de la trayectoria), o trayectoria completa. Un episodio se define como una subsecuencia de una trayectoria, de tal manera que todos los puntos GPS cumplen con un predicado dado. El predicado se basa en las coordenadas espacio-temporales de los puntos GPS, sus anotaciones y/o sus relaciones espacio-temporales con los objetos del contexto. Por ejemplo, la trayectoria de una persona puede segmentarse en episodios definidos en función del medio de transporte, por ejemplo, una trayectoria consiste en un primer episodio “caminar” seguido de un segundo episodio “en autobús”. Algunos de los trabajos de investigación que aplican esta técnica son [HGW⁺21, SA21, SGP⁺20, HWG⁺20, CXC⁺20, JTL⁺20, dAdSBdAS20, VSD⁺19, ACG19, CCX⁺18, JC18, GZW⁺18, MPLK16, PSR⁺13, ZZM⁺11, PMPP08].

La segmentación de la trayectoria en episodios se basa en criterios que dependen de la aplicación. Un desafío para los investigadores ha sido desarrollar técnicas para la identificación de paradas y esta caracterización de los episodios de parada y movimiento depende de los requisitos de la aplicación. En [KH06] se emplean dos criterios para dividir la trayectoria de un coche en movimiento. Estos criterios son la duración del intervalo de tiempo y la velocidad. En concreto, el coche debe circular con una velocidad menor de 3 km por hora al menos durante 5 minutos, en ese caso se considera que el coche está parado. El enfoque de [AA07] utiliza un umbral temporal mayor (2 horas) para identificar los lugares importantes visitados por una persona. El supuesto es que cuanto más tiempo se pasa en un lugar, más importante se considera el lugar. Una suposición similar se utiliza en [ZZM⁺11]. Las paradas se calculan como secuencias de posiciones GPS consecutivas, de modo que si la distancia y la duración temporal está por encima de dos umbrales respectivamente, entonces se detecta una parada. *CB-SMoT* [TBKA08] propone otro enfoque. Los episodios de parada se calculan en función de la variación de la velocidad de la trayectoria. Es decir, las paradas son esos segmentos de la trayectoria en la que la velocidad es menor que la velocidad media de la trayectoria. [RTO⁺10] utiliza como criterio para detectar la paradas es el cambio de dirección. [PSR⁺13] analiza las trayectorias en barco. Cada viaje se puede segmentar en períodos de tiempo donde el barco está parado y períodos en los que se está moviendo. Los primeros se denotan como paradas, mientras que los últimos se denotan como movimientos. De esta manera, una trayectoria es una secuencia de episodios de paradas y movimientos alternados.

Otros métodos identifican paradas mediante el uso de una combinación de datos brutos, información geográfica e información de la aplicación. Por ejemplo, en el método *SMoT* [ABK⁺07] una parada se considera una posición en la que una trayectoria se mantiene durante un intervalo de tiempo mínimo y que corresponde a la posición de uno de los puntos de interés definidos por la aplicación. Cada parada está anotada por el punto de interés correspondiente. En [AS03] anotan paradas con los puntos de interés que están “cerca” de la parada, es decir, dentro de un

2.3. RESULTADOS DE LA REVISIÓN SISTEMÁTICA DE LA LITERATURA 15

radio dado que define una zona de influencia espacial alrededor de la parada. En [XDZ09] los autores utilizan un contexto geográfico para segmentar las trayectorias de acuerdo con la proximidad a los puntos de interés. Los autores definen un episodio de trayectoria como el segmento de la trayectoria cuyas posiciones están influenciadas por un punto de interés determinado. Un punto de interés influye en una posición de la trayectoria si la distancia entre la posición y el punto de interés es más pequeña que la distancia entre la posición y cualquier otro punto de interés.

Otro criterio muy popular para segmentar una trayectoria en episodios son los medios de transporte utilizados por el usuario en movimiento. Este conocimiento es importante para todas las aplicaciones de planificación y gestión del transporte público. En la literatura se proponen diferentes técnicas para esta segmentación. Por ejemplo, se identifica cada modo de transporte en función de la velocidad de forma que se detecta “caminar” si la velocidad es inferior a 5 km/h. Otra técnica utiliza la continuidad del movimiento. Por ejemplo, un autobús hace paradas mientras que un taxi no lo hace. Otras técnicas consideran las restricciones de dirección y ruta. Por ejemplo, los autobuses usan solo algunas rutas definidas mientras que un tren circula por las vías de tren. [LFK05] utiliza un modelo gaussiano basado en la velocidad. Un enfoque similar se desarrolla en [ZCL⁺10], usando velocidad y la aceleración. Los autores primero detectan las posiciones donde el movimiento cambia entre caminar y no caminar. En un segundo paso, refinan el segmento de no caminar en segmentos caracterizados por los otros modos de transporte: bicicleta, autobús y conduciendo. De esta manera, utilizan una combinación de técnicas, desde el aprendizaje supervisado hasta la toma de decisiones y agregan un paso de post-procesamiento para mejorar la precisión de la segmentación.

[SCR10] propone un método más sofisticado que se basa en información adicional sobre los puntos de interés. Los autores notan cada episodio *parada* con la lista de puntos de interés que la persona probablemente haya visitado. Un punto de interés se selecciona en función de dos criterios: deben estar a una distancia razonable desde la parada (por ejemplo, 15 minutos caminando, en este trabajo se supone que representa la posición donde la persona aparcó su coche), y la duración del episodio de parada debe cumplir con la duración especificada para una visita al punto de interés (por ejemplo, un punto de interés de un restaurante tiene asociada una duración de 1 a 2 horas, sin embargo, un punto de interés de un museo requiere al menos 1 hora para una visita). El enfoque en [YCP⁺11] utiliza también información semántica como la cartografía. Por ejemplo, usar un carril bici indica que el modo de transporte es andar en bicicleta o la ubicación de una parada de autobús puede determinar si un segmento de la trayectoria se recorre en autobús.

Una vez identificados los episodios, los trabajos utilizan los puntos de interés para anotar cada episodio con información semántica. Uno de los trabajos más completos es el presentado en [YCP⁺11], en el cual se desarrolló una plataforma informática que soporta la construcción progresiva de trayectorias semánticas a partir de datos GPS brutos. Las trayectorias sin procesar se preprocesan primero para limpiarlas

y comprimirlas. En una segunda capa se segmentan las trayectorias en paradas y movimientos que se calculan teniendo en cuenta varios criterios espacio-temporales tales como la densidad de posición, velocidad y dirección. La tercera capa es la capa de anotación semántica en función de los datos contextuales, extraídos de fuentes de la aplicación y de las fuentes geográficas. [LCX⁺16] y [Zhe15] también asignan a las ubicaciones una etiqueta en función de los puntos de interés. Sin embargo, a menudo esta técnica es ineficaz. Imaginemos, por ejemplo, un usuario que va de compras a un centro comercial que tiene en su interior muchas tiendas y restaurantes. Es muy difícil para cualquier técnica discernir si ha ido de compras o ha ido comer a un restaurante.

Una alternativa diferente para la identificación de actividades semánticas está formada por los trabajos de investigación que proponen enfoques basados en análisis de grupos (*clustering*) para la identificación de episodios. En análisis de grupos, como técnica de aprendizaje no supervisado, se utiliza en este contexto para descubrir automáticamente lugares de interés asociados con paradas en trayectorias, calculados en función de la densidad de posiciones en un área espacial durante un período de tiempo determinado. En [CCJ10], el primer paso es extraer las paradas en función de un umbral entre dos ubicaciones GPS consecutivas registradas. Es decir, cuando el GPS deja de registrar valores, si la duración entre dos puntos GPS consecutivos es mayor que un umbral, esos dos puntos se consideran como un *punto de permanencia*. [CCJ10, SZZ⁺14] y [ZZXM09] introducen tanto umbrales espaciales como temporales. Por ejemplo, dado un umbral espacial de 200 m, si el usuario permanece *caminando* en una región durante un período de tiempo superior a un umbral de 5 minutos, entonces se considera que los puntos GPS forman una *región de estancia*. Sin embargo, este método tiene la desventaja de que puede detectar de muchos *puntos de permanencia* cuando el usuario camina dentro una región pequeña durante mucho tiempo. Después de este paso, los puntos de permanencia se agrupan para representar la actividad. Para ello se emplean un algoritmo de agrupamiento SEM-CLS (*semantics-enhanced clustering*), que combina los algoritmos *OPTICS* y *K-means* e información semántica adicional. Numerosos trabajos de investigación siguen esta línea y emplean análisis de grupos para enriquecer semánticamente las trayectorias crudas. Por ejemplo, [ZFL⁺07] desarrolló *DJ-Cluster*, un algoritmo de análisis de grupos basado en densidad para detectar lugares de estancia. [LCC13] y [ZZYS13] también aplican análisis de grupos basado en densidad para encontrar grupos de puntos de permanencia.

[CXC⁺20] emplea también análisis de grupos para convertir los datos sin procesar en trayectorias semánticas. Primero, divide las trayectorias en episodios de parada y movimiento. Posteriormente enriquece las trayectorias con información semántica haciendo uso de los puntos de interés y asignándole una etiqueta semántica a cada parada. Una vez hecho esto, busca trayectorias similares, por ejemplo detecta patrones de comportamientos parecidos *casa-escuela*. [WYH⁺19] emplea aprendizaje automático y análisis de grupos para determinar cómo influye

2.3. RESULTADOS DE LA REVISIÓN SISTEMÁTICA DE LA LITERATURA 17

la demografía (y sus características) en las trayectorias de las personas. [BGWSB19] emplea análisis de grupos para buscar patrones de movilidad como por ejemplo viajar o ir de compras. [LZD⁺19] construye conjuntos de palabras a partir de puntos de interés y aplica minería de datos para detectar los patrones de movilidad más frecuentes. Esos patrones frecuentes es lo que denomina trayectorias semánticas. [CCX⁺18] también identifica grupos con comportamientos similares empleando análisis de grupos, entropías y algoritmos. [CLL18a] emplea análisis de grupos para buscar patrones de movimiento a partir de posiciones y fotos geo-etiquetadas. Por último, [ZKSN17] también emplea algoritmos y análisis de grupos. Como aspecto negativo, estos algoritmos ignoran la continuidad temporal de las trayectorias, y además fallan cuando el registro de puntos es escaso, por ejemplo, en interiores. Por otra parte, estos algoritmos presentan el inconveniente de que requieren un gran coste computacional cuando se trabaja con un número elevado de trayectorias.

Otra de las técnicas para enriquecer semánticamente las trayectorias consiste en emplear primero algoritmos estadísticos para identificar las paradas dentro de una trayectoria y a continuación cada parada asociarla a información semántica. [LZX⁺21] emplea un modelo probabilístico generativo para extraer la información semántica de las trayectorias de los vehículos y así después estudiar los patrones de movilidad. [WZYS18] extrae patrones de movilidad semántica espacio-temporal a partir trayectorias de vehículos privados. [GZW⁺18] infiere a donde se quieren desplazar los usuarios y qué intereses de viaje tienen partiendo de sus *tweets*. [WLF⁺17] también emplea distribuciones estadísticas para inferir propósitos de viaje. Y [YCP⁺13b] emplea *Hidden-Markov-Model*-(HMM). Otros trabajos de investigación, como [GSY⁺21] por ejemplo, emplean los algoritmos *t-LSE* y *TF-IDF* para enriquecer los datos crudos con información semántica. [SA21] emplea el algoritmo *IB-SMOT* para segmentar las rutas de los usuarios en paradas. Este trabajo de investigación etiqueta semánticamente los datos crudos usando información de las redes sociales. Estos datos se procesan como un conjunto de palabras (textos) mediante minería de textos.

Otros estudios emplean métodos como las ontologías para enriquecer las trayectorias crudas, es el caso de [VSD⁺19] o [SGP⁺20] que emplea la ontología *The datAcron ontology*. [ACG19] también emplea ontologías para obtener las actividades a partir de las trayectorias de los trabajadores en el interior de edificios.

Finalmente, otros estudios optan por las redes neuronales para enriquecer los datos crudos. Por ejemplo, [MPLDSE⁺20] emplea una Red Neuronal Recurrente (*Recurrent Neural Network (RNN)*). [JTL⁺20] emplea también redes neuronales para convertir las trayectorias crudas (puntos GPS) en trayectorias semánticas, entrenando el sistema con las ubicaciones y las etiquetas de los puntos de interés. [BZL18] identifica estilos de vida y también los predice a partir de puntos de interés visitados, por ejemplo, gimnasio, escuela, casa, transporte público, etc. [HZC⁺20] emplea redes neuronales, concretamente emplea la técnica *Word2vec* para entrenar al sistema con un conjunto de trayectorias. Al igual que muchos otros trabajos

de investigación parten de la información de los puntos de interés para asignar a los datos crudos información semántica. Esta tarea se puede considerar como un problema de recuperación probabilística.

2.4. Conclusiones

Tras el análisis de los diferentes artículos respecto a la pregunta Q1 planteada inicialmente, *¿Cuales son los campos de aplicación de la identificación de actividades semánticas?*, hemos visto que existen multitud de campos de aplicación: desde trabajos que se centran en detectar actividades de bajo nivel como puede ser estar parado o en movimiento, hasta estudios que identifican actividades de alto nivel como repostar o viajar en tren. Sin embargo, muchos trabajos de investigación no se detienen aquí y van un paso más allá, una vez detectadas estas actividades infieren patrones de movilidad o predicen los futuros destinos de los objetos en movimiento.

En relación con la pregunta Q2 planteada inicialmente, *¿Cómo se representan las trayectorias semánticas?*, en la mayoría de los artículos hacen referencia a dos conceptos. Por una parte, la trayectoria cruda, y por otra, la trayectoria semántica. Existen varias definiciones para ambos conceptos, sin embargo, muchas coinciden en que una trayectoria cruda consiste en una colección de puntos GPS donde cada uno tiene asociado una marca temporal. Por otro lado, muchos estudios coinciden en que trayectoria semántica está formada por un conjunto de ubicaciones donde cada una tiene asociada una marca temporal y una o varias etiquetas que describen la información del contexto.

Finalmente, respecto a la pregunta Q3, *¿Qué actividades se identifican y cómo se identifican?*, hemos visto que existen multitud de técnicas para enriquecer semánticamente las trayectorias crudas. La mayoría tienen como base la división de la trayectoria en episodios de parado/moviendo y para ello, se fundamentan en el uso de puntos de interés y regiones de interés. Siguiendo esta línea, otros trabajos de investigación basan sus técnicas en algoritmos que dividen la trayectoria atendiendo a criterios como pueden ser la velocidad o el tiempo. Por otra parte, en lugar de atender a estos criterios de división, otros estudios se decantan por emplear análisis de grupos, aprendizaje máquina, métodos estadísticos, redes neuronales u ontologías, entre otros.

Capítulo 3

Conceptos básicos

Antes de presentar las contribuciones de esta tesis es necesario describir conceptos básicos en dos áreas: la captura de datos de sensores y las técnicas de tratamiento de información geográfica.

En la primera sección de este capítulo se describen las posibilidades para recogida de datos de los usuarios mediante los sensores de dispositivos móviles en general, y de *Android* en particular. En concreto, se describen las técnicas que se utilizan para recoger datos de los sensores de los dispositivos móviles y de ubicación mediante GPS en este sistema operativo.

Dado que la información geográfica es altamente necesaria en este proyecto, en la segunda sección de este capítulo se describen las técnicas de modelado de información geográfica, de los predicados y operadores de procesamiento de este tipo de información, y de las fuentes de datos de utilidad para la detección de actividades de interés.

3.1. Captura de datos de sensores

3.1.1. Captura de datos de localización

Los dispositivos móviles actuales capturan datos de localización utilizando tres técnicas alternativas: el sistema de posicionamiento global (GPS), el posicionamiento por red móvil, y el posicionamiento por redes WiFi.

El Sistema de Posicionamiento Global (GPS) es un sistema de localización diseñado por el Departamento de Defensa de los Estados Unidos con fines militares para proporcionar estimaciones precisas de posición, velocidad y tiempo. El sistema está operativo desde 1995 y utiliza una constelación de 24 satélites para determinar por triangulación la altitud, longitud y latitud de cualquier objeto en la superficie terrestre. La parte activa del sistema se descompone en varios componentes: i) Un

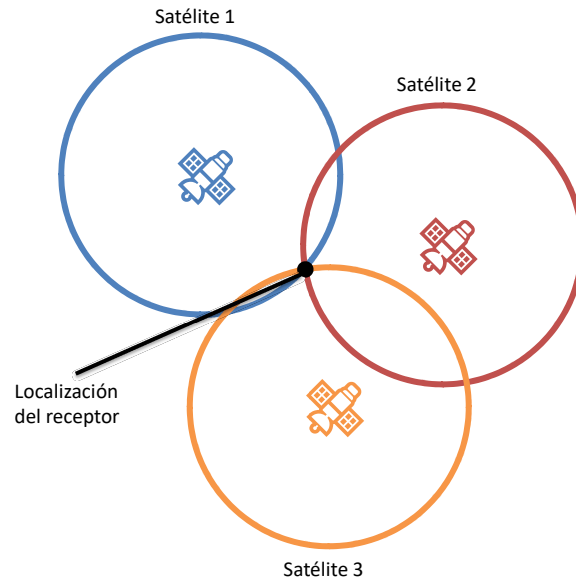


Figura 3.1: Proceso de trilateración

conjunto de 24 satélites con una órbita de 26.560 Km de radio y un periodo de 12 h, ii) cinco estaciones monitoras encargadas de mantener en órbita los satélites y supervisar su correcto funcionamiento, iii) tres antenas terrestres que envían a los satélites las señales que deben transmitir. La parte pasiva del sistema consiste en las antenas y receptores pasivos de los usuarios situados en tierra.

Para calcular la posición, el receptor GPS utiliza la señal recibida por al menos cuatro satélites diferentes. Para cada uno de los satélites visibles, el receptor calcula el tiempo que ha tardado en llegar la señal emitida por el satélite y determina una esfera dentro de la que se encuentra el receptor. La intersección de las esferas calculadas para todos los satélites visibles determina la ubicación del receptor. En la Figura 3.1 se muestra una simplificación del proceso utilizando únicamente dos dimensiones.

El requisito de disponer de cuatro satélites visibles está motivado por la baja precisión de los relojes de los dispositivos receptores. Mientras que los satélites disponen de relojes atómicos sincronizados con una estación terrestre, la precisión del reloj del receptor es baja y se desconoce si está sincronizado. Por ello, además de incluir como incógnita en las ecuaciones las coordenadas del receptor, se incluye como incógnita el tiempo que marca su reloj, lo que provoca que se necesiten cuatro señales para resolver las cuatro incógnitas.

La precisión de la ubicación GPS aumenta con el número de satélites visibles para el receptor. Además, la precisión de la ubicación GPS viene determinada

por otros factores tales como las perturbaciones en la velocidad de transmisión de la señal provocadas por la atmósfera, los errores introducidos por la recepción de señales reflejadas en lugar de la señal directa, u otras interferencias eléctricas. Sin embargo, la mayor limitación a la precisión viene dada por la interferencia introducida directamente por el Departamento de Defensa de los Estados Unidos (disponibilidad selectiva S/A).

En cuanto al posicionamiento por red móvil, las redes de telefonía móvil se basan en un conjunto de celdas controladas cada una de ellas por una estación base que contiene, entre otros dispositivos, la antena de telefonía, y que se encuentra en una posición conocida. Las estaciones base se comunican constantemente con los dispositivos móviles para determinar el momento en el que tiene que cambiar de una estación base a otra adyacente. Para ello, envían al dispositivo móvil una señal de localización (*ping*) de forma periódica. El operador de la red puede conocer la distancia del dispositivo móvil a la antena midiendo el retardo entre el envío de la señal de localización y su retorno. Realizando la intersección de los círculos resultantes de este cálculo para tres antenas, el operador de la red puede conocer la ubicación del dispositivo. El consumo de energía es mucho más bajo que en el caso del posicionamiento GPS, pero la precisión se ve comprometida en este proceso porque depende del tamaño del área cubierta por la antena del operador y la viabilidad de conseguir recibir señal de tres antenas.

La tercera alternativa es el posicionamiento por redes WiFi. En este caso, el operador del sistema operativo del dispositivo móvil (*Google* en el caso de *Android*) mantiene un registro de las redes WiFi que han detectado en el pasado los dispositivos junto con su ubicación geográfica. Dado que el alcance de una red WiFi es muy limitado (del orden de metros en oposición a los kilómetros de la red móvil), la precisión que se puede alcanzar es muy alta con un consumo de energía muy bajo. Sin embargo, la disponibilidad de redes WiFi está claramente sesgada a entornos urbanos.

La determinación de la ubicación del usuario en *Android* funciona solicitando al sistema registrarse como observador de la localización. Una vez que una aplicación se registra puede hacer tres cosas:

- Consultar a todos los proveedores de ubicación la última ubicación de usuario conocida.
- Recibir actualizaciones periódicas de la ubicación actual del usuario desde un proveedor de ubicación.
- Registrar una petición de notificación si el dispositivo entra dentro de una proximidad determinada (especificado por radio en metros) de una latitud y longitud especificada.

Si se solicita recibir actualizaciones periódicas, la frecuencia de notificación de las nuevas ubicaciones puede controlarse utilizando los parámetros *minTime* y *minDistance*:

- *minTime* indica el tiempo mínimo entre milisegundos que debe transcurrir entre notificaciones. Por ejemplo, si *minTime* tiene el valor 10.000 no se recibirán notificaciones en menos de 10 segundos desde la última recibida.
- *minDistance* indica la distancia en metros que debe existir entre las ubicaciones para recibir una notificación. Por ejemplo, si *minDistance* tiene el valor 100 no se recibirán notificaciones cuya distancia con la última recibida sea menor a 100 metros.

Las dos condiciones tienen que cumplirse simultáneamente para recibir una nueva notificación. Además, los dos parámetros pueden tener el valor 0 para recibir notificaciones tan frecuentemente como sea posible. La decisión de qué valores utilizar depende del compromiso que se quiera alcanzar entre la precisión de la ubicación y la energía consumida. Sin embargo, aunque los valores de *minTime* y *minDistance* sean elevados, hay que tener en cuenta que el receptor GPS sigue activo aunque nuestra aplicación no reciba notificaciones, por lo que el consumo de energía no se reduce a cero.

Cada localización notificada por el servicio de localización proporciona información acerca de la latitud y longitud de la ubicación (en grados decimales usando el sistema de referencia de coordenadas EPSG:4326¹), el instante temporal de la localización (en tiempo unix), el rumbo (en grados decimales), la velocidad instantánea (en metros por segundo), la altitud (en metros respecto al elipsoide WGS84 utilizado en EPSG:4326), e información respecto a la precisión de la ubicación.

La utilización de la ubicación del usuario de una aplicación móvil siempre se enfrenta al reto de obtener una precisión elevada mientras se conserva la energía del dispositivo. Esto implica la necesidad de definir un método complejo de captura de la ubicación que tenga en cuenta que:

- El receptor GPS es uno de los mayores consumidores de energía del dispositivo.
- Solicitar actualizaciones menos frecuentes de la ubicación reduce la necesidad de procesamiento pero no desactiva el receptor GPS.
- Activar el receptor GPS no implica recibir inmediatamente una ubicación. En el interior de los edificios suele ser imposible obtener una ubicación GPS.

3.1.2. Captura de datos de movimiento

La mayoría de los dispositivos *Android* incluyen sensores que son útiles si se desea monitorizar el movimiento o la posición tridimensional del dispositivo móvil. Por ejemplo, un juego puede utilizar las lecturas del acelerómetro del dispositivo para inferir gestos y movimientos complejos del usuario, como inclinación, movimiento,

¹<https://epsg.io/4326>

rotación o balanceo. Otro ejemplo puede ser una aplicación de viaje que use el sensor de campo geomagnético para informar del rumbo que se sigue. El *framework* de sensores de *Android* permite acceder a los sensores disponibles en el dispositivo y adquirir datos sin procesar del sensor. La plataforma *Android* admite tres amplias categorías de sensores:

- **Sensores de movimiento.** Estos sensores miden fuerzas de aceleración y fuerzas de rotación a lo largo de tres ejes. Esta categoría incluye acelerómetros, sensores de gravedad, giroscopios y sensores de vector giratorio.
- **Sensores ambientales.** Estos sensores miden varios parámetros ambientales, como la temperatura y presión del aire ambiente, la iluminación y la humedad. Esta categoría incluye barómetros, fotómetros y termómetros.
- **Sensores de posición.** Estos sensores miden la posición física de un dispositivo. Esta categoría incluye sensores de orientación y magnetómetros.

Para identificar los sensores que están disponibles en un dispositivo, primero se necesita obtener una referencia al servicio de sensores representado por la clase *SensorManager*. Esta clase proporciona métodos para enumerar sensores y registrar receptores de eventos de sensores. También proporciona varias constantes de sensor que se utilizan para informar la precisión del sensor, establecer velocidades de adquisición de datos y calibrar sensores.

Cada sensor concreto se representa mediante una instancia de la clase *Sensor*. Esta clase proporciona métodos que le permiten determinar las capacidades de un sensor. Por ejemplo, se pueden usar los métodos *getResolution* y *getMaximumRange* para obtener la resolución del sensor y el rango máximo de medición. También se puede usar el método *getPower* para obtener los requisitos de potencia de un sensor. Con los métodos *getVendor* y *getVersion* se puede determinar el fabricante y la versión del sensor.

Finalmente, el método *getMinDelay* que devuelve el intervalo de tiempo mínimo (en microsegundos) que un sensor puede usar para detectar datos. Si el sensor devuelve un valor distinto de cero quiere decir que es un sensor de modo *streaming* que detecta los datos a intervalos regulares. Si el método devuelve el valor cero significa que el sensor informa de los datos solo cuando hay un cambio en los parámetros que está detectando.

Para monitorizar datos de sensores, una aplicación debe registrarse como observador del sensor. El sistema *Android* notifica a la aplicación cuando ocurre lo siguiente:

- **Un sensor informa un nuevo valor.** En este caso, el sistema proporciona a la aplicación información sobre los nuevos datos que registro el sensor, la precisión de los datos, el sensor que generó los datos, y la marca de tiempo en la que se generaron los datos (en microsegundos).

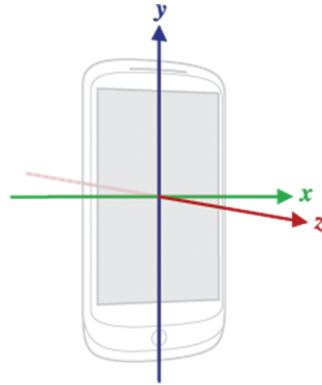


Figura 3.2: Sistema de coordenadas utilizado por el *framework* de sensores

- **La precisión de un sensor cambia.** En este caso, el sistema proporciona información del sensor que cambió y su nueva precisión del sensor.

Para comenzar a recibir datos de los sensores la aplicación debe indicar el sensor del que se quieren recibir los datos y el retraso mínimo con el que se desean recibir los datos. El retraso de datos (la inversa de la frecuencia de muestreo) controla el intervalo en el cual los eventos del sensor se envían a la aplicación. El retraso de datos predeterminado (200 milisegundos) es adecuado para monitorizar cambios típicos de orientación de pantalla. Otras alternativas predefinidas para el retraso de datos oscilan entre 60 milisegundos (orientada a modificaciones del interfaz de usuario) o 20 milisegundos (orientada a juegos). También es posible indicar un retardo de 0 microsegundos que hará que se proporcionen datos lo más rápidamente posible. El retraso que se especifica es sólo una sugerencia al sistema ya que el sistema *Android* y otras aplicaciones pueden alterar este retraso.

El *framework* de sensores utiliza un sistema de coordenadas de 3 ejes estándar para expresar los valores de los datos. El sistema de coordenadas se define en relación con la pantalla del dispositivo cuando el dispositivo se mantiene en su orientación predeterminada (ver figura 3.2). Cuando un dispositivo se mantiene en su orientación predeterminada, el eje X es horizontal y apunta hacia la derecha, el eje Y es vertical y apunta hacia arriba, y el eje Z apunta hacia el exterior de la pantalla y las coordenadas detrás de la pantalla tienen valores Z negativos. El punto más importante de este sistema de coordenadas es que los ejes no se intercambian cuando cambia la orientación de la pantalla del dispositivo; es decir, el sistema de coordenadas del sensor nunca cambia a medida que el dispositivo se mueve. Otro punto importante es que la aplicación no debe asumir que la orientación natural (por defecto) de un dispositivo es vertical. La orientación natural para muchos dispositivos es horizontal. Y el sistema de coordenadas del sensor siempre se

basa en la orientación predeterminada del dispositivo.

La plataforma *Android* proporciona varios sensores que permiten controlar el movimiento de un dispositivo. Los más relevantes de cara al objetivo de esta tesis son los siguientes:

- **Acelerómetro** (m/s^2). Mide la aceleración aplicada al dispositivo incluyendo la fuerza de la gravedad.
- **Gravedad** (unidad m/s^2). Indica la dirección y la magnitud de la gravedad.
- **Aceleración lineal** (m/s^2). Proporciona un vector tridimensional que representa la aceleración a lo largo de cada eje del dispositivo excluyendo la gravedad.
- **Giroscopio** (rad/s). Mide la velocidad de rotación alrededor de los ejes de un dispositivo.

3.2. Técnicas de análisis de información geográfica

El campo de los sistemas de información que se encarga de definir y desarrollar tecnología para el almacenamiento y procesamiento de información geográfica es el de los Sistemas de Información Geográfica (GIS, del inglés *Geographic Information System*). Este campo ha recibido mucha atención en los últimos años ya que las mejoras recientes en el hardware han hecho posible que la implementación de este tipo de sistemas sea abordable por muchas organizaciones. Además, se ha llevado a cabo un esfuerzo colaborativo para la definición de estándares por dos organismos: *International Organization for Standardization* (ISO) (mediante el ISO/TC 211 y la línea de estándares 191xx) y el *Open Geospatial Consortium* (OGC). El resultado es que el número de alternativas tecnológicas se ha incrementado notablemente.

La representación de información geográfica en un sistema de información consiste en la representación discreta de fenómenos continuos que ocurren sobre la superficie terrestre. La discretización de este espacio continuo es necesaria debido a la naturaleza propia de la representación digital de información. Esta discretización se realiza siempre en los sistemas de información siguiendo siempre un proceso de modelado similar al descrito en la Figura 3.3. En primer lugar, los elementos del mundo real se describen mediante un conjunto de abstracciones de un modelo conceptual dando lugar a un esquema conceptual de la aplicación. En el ejemplo de la Figura 3.3 una rotonda de una carretera del mundo real se describe en el modelo conceptual mediante curvas geométricas expresadas mediante funciones matemáticas. A continuación, el esquema conceptual se transforma en un esquema lógico utilizando las abstracciones de un modelo lógico que tiene en cuenta las limitaciones de los ordenadores a la hora de representar y procesar información. Por ejemplo, en la Figura 3.3, las curvas geométricas generales del modelo conceptual se

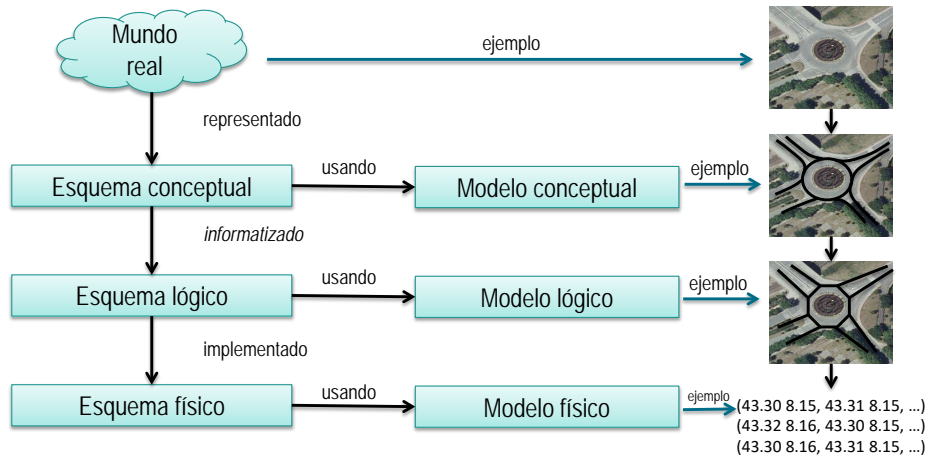


Figura 3.3: Proceso de modelado en un sistema de información

han transformado en segmentos de recta para evitar la complejidad de almacenar y procesar en un sistema gestor de base de datos ecuaciones matemáticas generales y almacenar únicamente coordenadas y segmentos. Finalmente, las soluciones tecnológicas existentes transforman el modelo lógico en una implementación concreta del mismo que denominamos modelo físico. En este modelo es donde se toman las decisiones finales que permitirán implementaciones eficientes. En el ejemplo de la Figura 3.3 se ha representado como la conversión final de las coordenadas y segmentos del modelo lógico en estructuras de datos que se almacenan en el sistema de información.

A lo largo de los años se han definido infinidad de modelos de datos conceptuales para información geográfica. Cada herramienta de desarrollo de GIS ha definido su modelo propio (ESRI ArcGIS, *Intergraph Geomedia*, etc.). De la misma forma, en el mundo de la investigación es muy habitual que para cada proyecto específico se defina su propio modelo de datos conceptual. Para lidiar con esta situación de heterogeneidad de modelos de datos conceptuales, en los últimos años se ha realizado un esfuerzo por definir estándares internacionales para información geográfica. El estándar ISO 19107: *Geographic Information – Spatial Schema* [ISO03] define una jerarquía de tipos de datos para objetos geográficos en la que se considera que la información espacial consiste en objetos que pueden ser representados mediante figuras geométricas (puntos, curvas, o superficies).

En los modelos conceptuales siempre se trabaja a nivel abstracto por lo que no hay detalles de la implementación en un ordenador. De esta forma, los objetos geográficos son conjuntos infinitos de puntos. Sin embargo, estas definiciones son

solo teóricas puesto que los ordenadores en los que finalmente se va a realizar la implementación poseen limitaciones. Por ejemplo, el espacio de almacenamiento es finito, y la precisión aritmética está limitada. Como consecuencia, los modelos conceptuales no son directamente implementables en un ordenador, y son necesarias unas transformaciones previas. De esta forma, es necesaria la transformación de un modelo conceptual a un modelo lógico. En el campo de los sistemas de información geográfica el modelo conceptual de objetos geográficos acostumbra a implementarse utilizando el modelo vectorial. El modelo vectorial utiliza tipos de datos del ordenador (por ejemplo, enteros o números en punto flotante) para definir un sistema de coordenadas sobre el que se representa la información geográfica utilizando construcciones geométricas (puntos, segmentos, etc.). Para cada tipo de datos del modelo conceptual se define una representación utilizando objetos definidos en el modelo vectorial. El modelo vectorial usado más ampliamente en la actualidad es el denominado *Simple Feature Access*, definido por el *Open Geospatial Consortium* para representar información geográfica. Este mismo estándar ha sido adoptado por ISO como ISO 19125-1:2004 [ISO04].

En la Figura 3.4 se muestran los tipos de datos definidos en el estándar y en la Figura 3.5 se pueden ver ejemplos de cada uno de los tipos de datos. El tipo de datos *Point* permite representar puntos. El tipo de datos *LineString* permite representar curvas mediante líneas formadas por segmentos de recta. El tipo de datos *Polygon* permite representar superficies con agujeros mediante polígonos formados por segmentos de recta. El tipo de datos *MultiPoint* permite representar una colección de puntos, el tipo de datos *MultiLineString* permite representar una colección de curvas. El tipo de datos *MultiPolygon* permite representar una colección de superficies. Finalmente, el tipo de datos *GeometryCollection* permite representar colecciones de objetos de cualquier tipo y el tipo de datos *Geometry* permite representar un objeto de cualquier tipo (simple o compuesto).

Una vez que la información geográfica se encuentra almacenada, es necesario definir operaciones sobre la información que permitan procesarla. Para poder definir estas operaciones es conveniente aclarar los conceptos matemáticos de dimensión, interior, borde y exterior. Si bien se podrían definir de manera matemática, nosotros los explicaremos de manera intuitiva. La dimensión de un objeto geográfico representa el número de coordenadas que harían falta, como mínimo, para indicar la posición de cualquier punto del objeto geográfico. Así, si nuestro objeto es un punto, no necesitaremos ninguna información extra para referenciar el mismo, dado que contiene un único punto. En el caso de una línea, necesitaremos una coordenada para poder referenciar los distintos puntos de la línea; por tanto, la dimensión de una línea es 1. En el caso de un polígono, necesitaremos 2 coordenadas para poder referenciar los puntos contenidos en el polígono. Por tanto, la dimensión es 2. Cuando nuestro objeto geográfico está formado por varios objetos a su vez, la dimensión será la mayor de las dimensiones de los objetos que lo conforman. El exterior de un objeto es el conjunto de puntos que no pertenecen al objeto. El

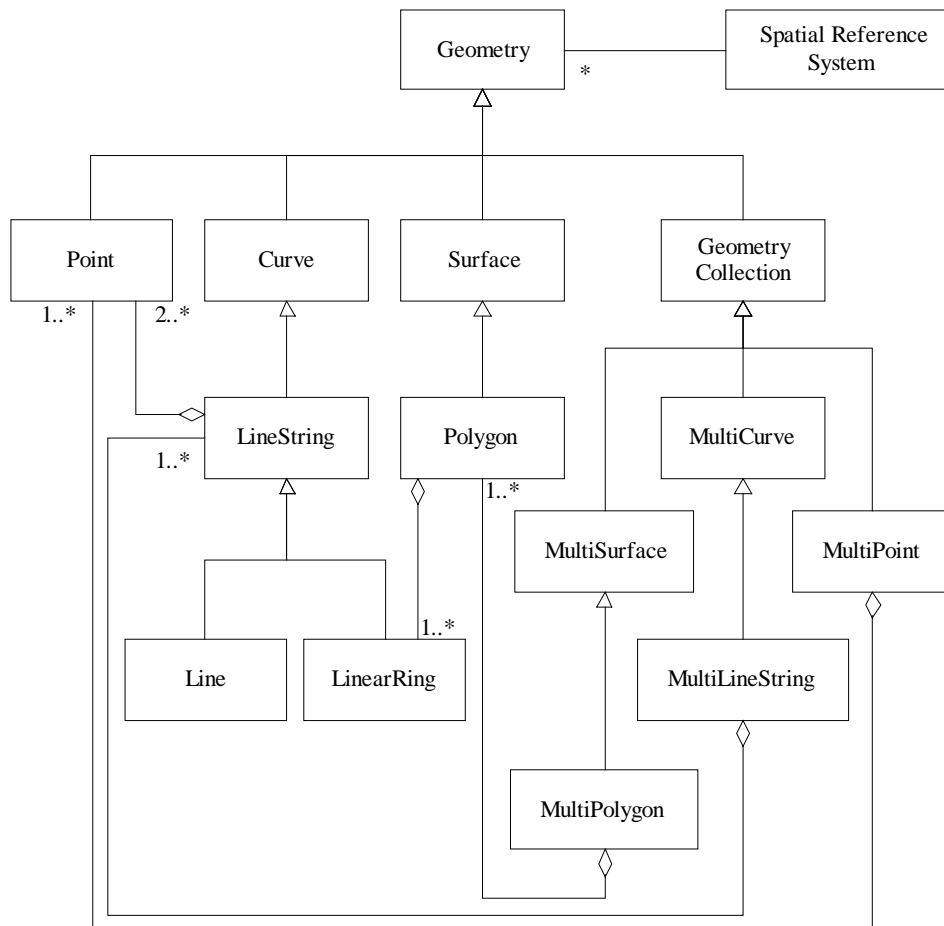


Figura 3.4: Tipos de datos de *Simple Feature Access*, extraído de [ISO04]

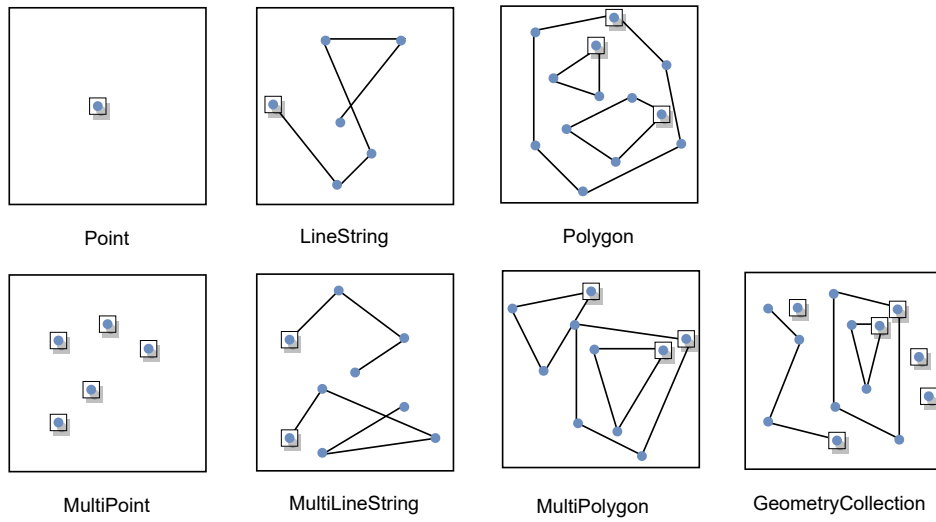


Figura 3.5: Ejemplos de valores de los tipos de datos de *Simple Feature Access*

	Borde	Interior	Exterior
	\emptyset		$U - \{\bullet\}$
			$U - \{\bullet\}$
			$U - \{\square\}$

Figura 3.6: Borde, interior y exterior de objetos geográficos

interior de un objeto es el conjunto de puntos que están dentro del objeto. El borde de un objeto son aquellos puntos en los que se pasa de estar dentro a estar fuera del objeto. La dimensión del borde siempre es exactamente una dimensión inferior a la dimensión del propio objeto. En nuestro caso particular, las operaciones que tienen más interés para nosotros son los predicados espaciales del modelo vectorial que permiten determinar las relaciones existentes entre objetos geográficos. Por ejemplo, para determinar si un objeto está totalmente solapado por otro, o para determinar si dos objetos se intersecan en algún punto. Los predicados espaciales son útiles al recuperar información de las bases de datos porque permiten establecer relaciones geográficas entre distintos elementos. Si tenemos las geometrías de los municipios de España, usando alguna de estas relaciones seremos capaces de determinar qué municipios colindan con cualquier otro.

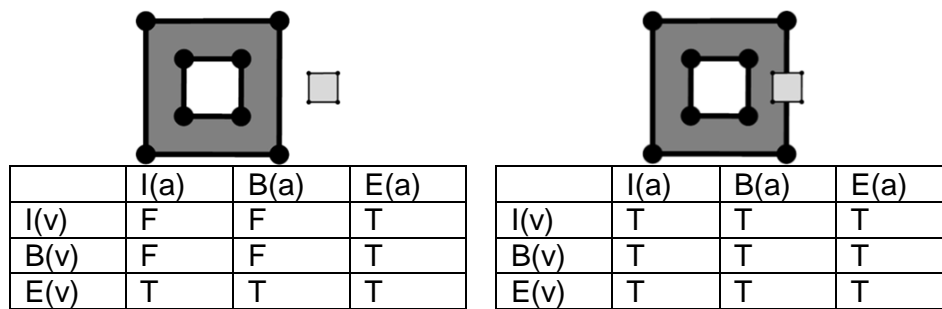


Figura 3.7: Cambios en la relación topológica de los objetos se reflejan en la matriz calculada por el *Dimensionally Extended Nine-Intersection Model (DE9IM)*

Para definir los predicados espaciales se tiene en cuenta la dimensión del resultado de realizar la intersección entre el borde, el interior y el exterior de cada objeto y se asigna un nombre a cada combinación resultante. Este modelo se denomina *Dimensionally Extended Nine-Intersection Model (DE9IM)* [CDF95, CDF96]. En la Figura 3.7 se muestra el resultado de evaluar el modelo para dos polígonos. En todos los casos se supone que el objeto a es el cuadrado pequeño de la figura, y el objeto v es el polígono grande. En la tabla se muestra la matriz que resultaría en cada caso indicando la dimensión de la intersección de los interiores, los bordes y los exteriores. Se puede ver que un cambio en la relación espacial en los objetos produce un cambio en la matriz.

En la Figura 3.8 se muestran los predicados espaciales a los que se les da nombre en la norma ISO 19125-1:2004 [ISO04] y se muestran ejemplos de las mismas para puntos, líneas y polígonos. Como podemos observar, hay muchos casos en los que una relación no es posible mediante dos tipos determinados de objetos, como por ejemplo el predicado *st_overlaps* para dos puntos. El *Dimension-Extended Nine*

























	p/p	p/l	p/po	l/l	l/po	po/po
st_disjoint						
st_touches	X					
st_within						
st_crosses	X	X	X			X
st_overlaps	X	X	X		X	
st_equals		X	X		X	

Figura 3.8: Resumen de los predicados espaciales

Intersection Model es usado en los estándares internacionales y en las bases de datos con funcionalidades geográficas como *Oracle*, *PostgreSQL* y *MySQL*.

Capítulo 4

Sistema de identificación de actividades semánticas

En esta sección presentamos nuestra propuesta para un sistema de identificación de actividades semánticas de trabajadores en movilidad que puede ser integrado en los procesos de negocio de cualquier empresa mediante la interacción con su sistema de gestión de trabajadores en movilidad (MWM, del inglés *Mobile Workforce Managment*). En la Sección 4.1 describimos la arquitectura general del sistema. A continuación describiremos cada uno de los componentes de la Figura 4.1. En la Sección 4.2 describiremos el componente de captura de datos (*SensorCollector*), en la Sección 4.3 el componente de recogida de datos (*SensorReceiver*), y en la Sección 4.4 el componente de anotación semántica (*Annotator*).

4.1. Arquitectura funcional del sistema

La Figura 4.1 muestra la arquitectura funcional del sistema. En el lado derecho de la figura se encuentra el sistema MWM que proporciona la información mínima para el funcionamiento del sistema: datos sobre los clientes de la empresa, los trabajadores y su agenda diaria. Este componente se muestra en gris porque no ha sido desarrollado por nosotros. A lo largo de los diagramas de esta sección se utilizará este color para representar los componentes externos a nuestro sistema. En el lado izquierdo de la figura se muestra el sistema que hemos desarrollado. En primer lugar, en la parte superior derecha aparece el componente de captura de datos (*SensorCollector*), el cual consiste en una aplicación móvil *Android* que tiene como objetivo recoger la información de ubicación y los datos de los sensores de cada uno de los empleados en movilidad. El componente de recogida de datos (*SensorReceiver*) tiene como objetivo recibir todos los datos de los móviles de los empleados y almacenarlos en la base de datos sin procesar (*Raw data*).

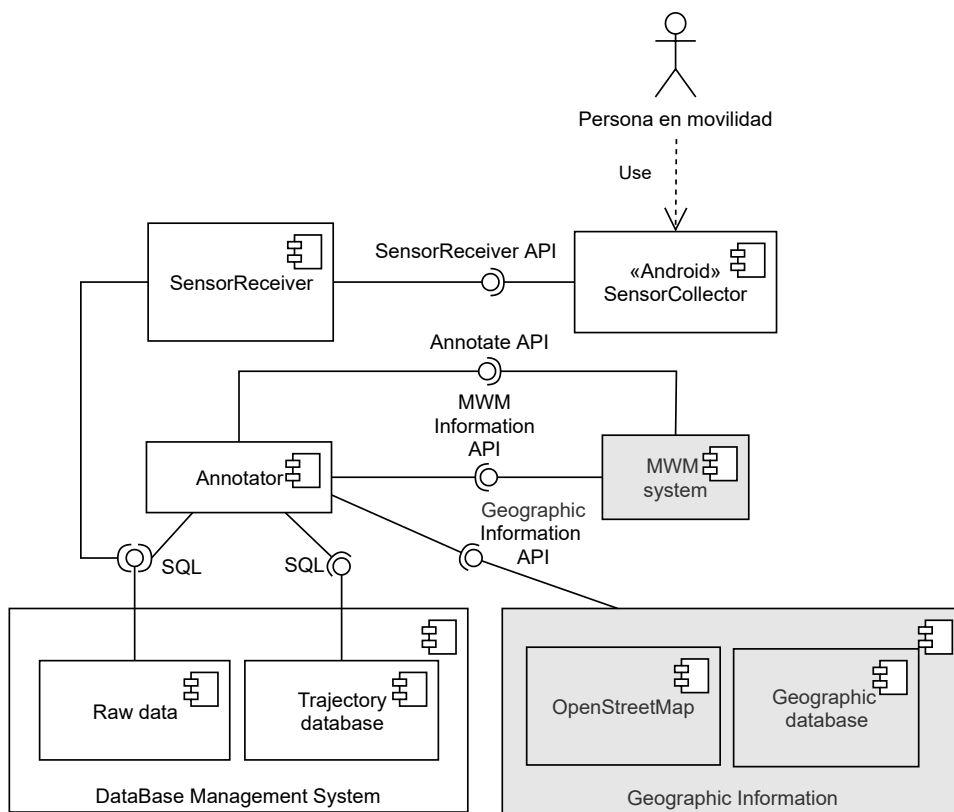


Figura 4.1: Arquitectura funcional

El componente de anotación semántica (*Annotator*) tiene como objetivo anotar semánticamente la trayectoria a partir de la información procedente de tres fuentes diferentes, el sistema *Mobile Workforce Management (MWM system)*, los datos de los sensores del dispositivo móvil (*Raw data*) y la información geográfica del dominio (*Geographic Information*). Finalmente, las actividades son almacenadas en la base de datos *Trajectory database*.

4.2. Componente de captura de datos (*SensorCollector*)

El objetivo del componente de captura de datos *SensorCollector* es recoger los datos de localización y los datos de sensores del dispositivo móvil del empleado. Para capturar los datos de localización, el componente utiliza el receptor de GPS del dispositivo móvil. Tal y como se describió en la Sección 3.1, en *Android* el desarrollador puede configurar la distancia mínima y el tiempo mínimo para obtener una nueva ubicación, por lo que la frecuencia de obtención de datos de localización es variable. Por ejemplo, un desarrollador puede indicar que un evento de localización solo debe ocurrir si la distancia desde la última localización es mayor a 10 metros y el tiempo transcurrido es mayor a 10 segundos. El desarrollador debe configurar estos valores de forma que sean lo suficientemente pequeños como para asegurar que la precisión de la localización es elevada y que permite la identificación de las actividades realizadas, pero no demasiado pequeños para evitar consumir la batería del móvil demasiado rápido. La consecuencia es que la frecuencia de recogida de datos nunca será muy alta (como máximo, del orden de varias ubicaciones por minuto). Además, los datos del sensor de ubicación solo son precisos en exteriores. Por ello, los datos de ubicación tendrán una tasa de recogida de datos variable, con una frecuencia baja, y con periodos sin datos.

Por otra parte, aunque el *framework* del sensor de *Android* admite 13 tipos de sensores¹, para la detección de actividades es suficiente con que el componente recoja los datos del sensor de aceleración lineal. El sensor de aceleración lineal proporciona un vector tridimensional que representa la aceleración en cada eje del dispositivo, sin incluir la gravedad. A pesar de que hay dispositivos que incluyen sensores con datos de más alto nivel (por ejemplo, el detector de pasos), la disponibilidad de estos sensores en los dispositivos móviles es muy baja y no se tendrán en cuenta. En algunos dispositivos no está presente el sensor de aceleración lineal, pero sus valores pueden calcularse a partir del sensor de aceleración y el sensor de gravedad.

Por todo esto, teniendo en cuenta que los datos de ubicación sólo están disponibles en exteriores, con tasa de datos variable y una baja frecuencia de recogida, y que los datos de los sensores están siempre disponibles, con tasa de datos constante y con frecuencia mucho más elevada, el subsistema de recogida de

¹https://developer.android.com/guide/topics/sensors/sensors_overview?hl=es-419

datos de ubicación debe ser independiente del subsistema de recogida de datos de sensores.

Otro punto a tener en cuenta es que la cantidad de datos que se recogen por el componente de captura de datos (*SensorCollector*) en el dispositivo móvil es muy elevada ya que se registran 5 datos diferentes cada uno con tres dimensiones en un mismo segundo. Si se transfiriesen todos y cada uno de ellos hacia *SensorReceiver* se consumiría un elevado porcentaje de los datos disponibles en la tarifa de telefonía móvil de los empleados. También es importante considerar el gasto energético, ya que el componente de captura de datos (*SensorCollector*) estará activo durante toda la jornada laboral del empleado, y por tanto existe el riesgo de que su funcionamiento consuma de forma excesiva la batería de los dispositivos de los empleados. Por todo esto, es necesario diseñar el componente de tal forma que se reduzca la cantidad de datos que se envían y que se reduzca la energía necesaria para su funcionamiento.

Por otra parte, a pesar de que la mayor parte de los empleados trabajan en un entorno urbano, existe un porcentaje significativo de los mismos que realizan su actividad en el entorno rural. En estos casos la disponibilidad de conexión móvil es baja. Por ello, el componente debe estar diseñado de manera que no dependa de que la conexión con *SensorReceiver* esté activa de forma permanente. Para ello, el componente debe poder almacenar los datos en el dispositivo móvil hasta que se disponga de conexión y se puedan enviar.

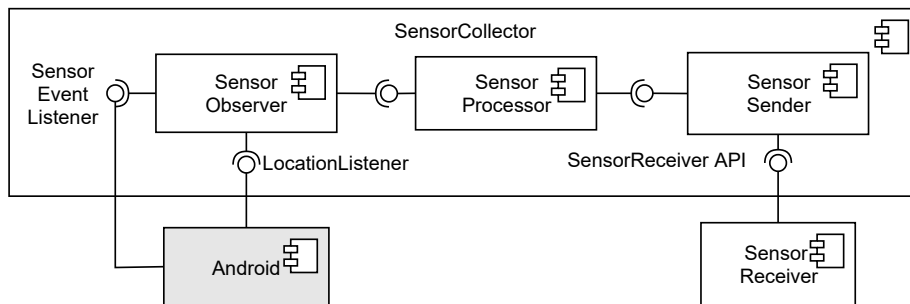


Figura 4.2: Arquitectura del componente de captura de datos (*SensorCollector*)

El componente de captura de datos (*SensorCollector*) se estructura en tres componentes de procesamiento, tal y como se muestra en la Figura 4.2. En la parte de la izquierda se muestra el componente que actúa como observador de los eventos generados por el sistema *Android* (*SensorObserver*). En la parte central se encuentra el componente que procesa los datos recogidos por los observadores y los prepara para el envío (*SensorProcessor*). En la parte de la derecha se encuentra el componente que se encarga de la comunicación con el servidor (*SensorSender*).

La frecuencia con la que se recogen tanto los datos de localización como de los

sensores se puede configurar en el componente *SensorObserver*. En la recogida de sensores, por defecto la tasa de muestreo es la más baja de las ofrecidas por *Android* (*SENSOR_DELAY_NORMAL*, 200 milisegundos aproximadamente) ya que no es necesaria una precisión muy elevada en la detección de los movimientos del móvil para la detección de actividades. En la recogida de las ubicaciones, por defecto se registra una nueva localización si la distancia desde la última posición es mayor a 10 metros y el tiempo transcurrido es mayor a 10 segundos. Estos parámetros puede configurarlos el desarrollador.

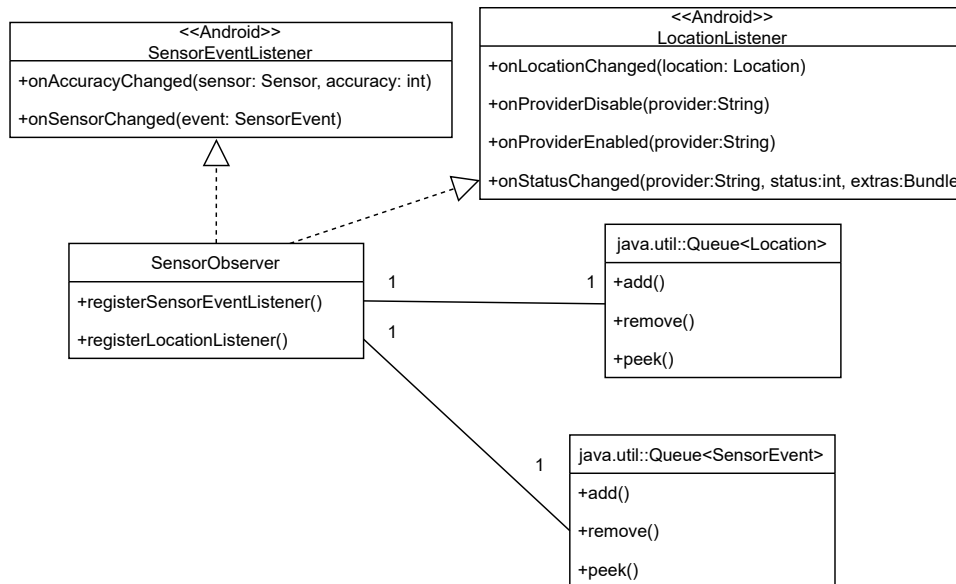


Figura 4.3: Diagrama de clases del componente de captura de datos (*SensorObserver*)

La Figura 4.3 muestra un diagrama de clases con el detalle de la implementación del componente de captura de datos (*SensorObserver*). Este componente funciona mediante dos clases que implementan el patrón *Observador* para obtener información de la ubicación y de los sensores tal y como requiere *Android*. Para recibir notificaciones de *Android* la clase *SensorObserver* implementa los interfaces *SensorEventListener* y *LocationListener*, por lo que se registra esta clase con los servicios correspondientes de *Android* y esta es notificada en caso de la existencia de nuevos datos. Los métodos de estas clases son invocados por el propio sistema operativo *Android* cada vez que hay disponible una nueva ubicación o un nuevo evento de los sensores. Como estos métodos se invocan de forma muy frecuente por el sistema operativo (varias veces por segundo en el caso de los sensores, y varias

veces por minuto en el caso de la ubicación) el procesamiento que realizan es el mínimo para evitar consumir demasiada energía y cada uno de ellos únicamente almacena en una cola del componente *SensorProcessor* los nuevos datos recibidos (*Queue<Location>* para las ubicaciones, y *Queue<SensorEvent>* para los datos de sensores).

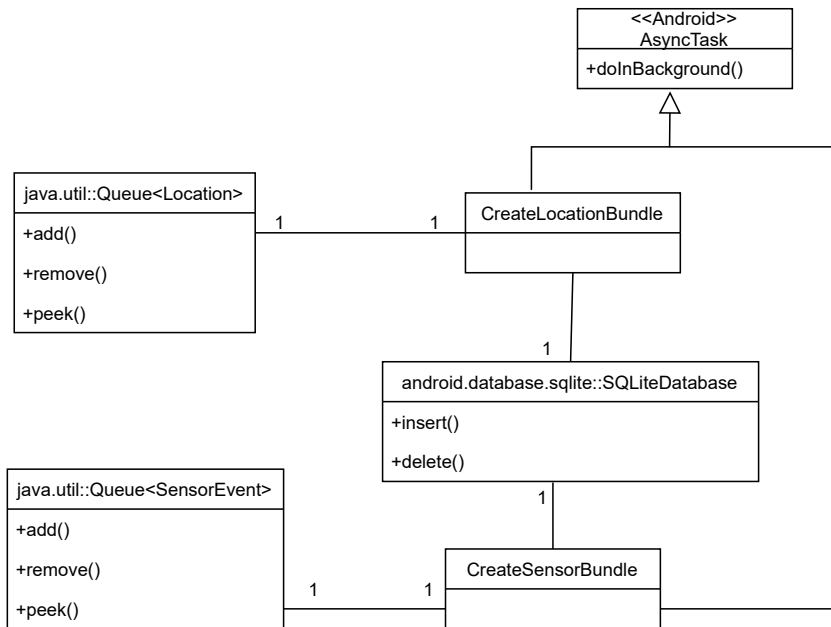


Figura 4.4: Diagrama de clases de la etapa de procesamiento de datos y preparación del envío (*SensorProcessor*)

La Figura 4.4 muestra un diagrama de clases con el detalle de la implementación del componente de procesamiento de datos y preparación del envío (*SensorProcessor*). Para preparar el envío de datos de ubicaciones basta con extraer todos los datos de las colas en forma de lista y convertir esas listas en un *array* de objetos JSON. El formato de cada objeto JSON es el que corresponde al modelo conceptual que se describe en la Figura 4.5. El proceso para convertir la cola de objetos JSON es costoso, por ello es necesario realizarlo mediante una tarea asíncrona que se ejecuta con una frecuencia baja (por defecto está configurado cada cinco minutos) para ahorrar batería del dispositivo móvil. Para realizar esta tarea es necesario crear dos clases que extiendan la funcionalidad de la clase `AsyncTask` (denominadas `CreateLocationBundle` y `CreateSensorBundle` en la Figura 4.4). En la implementación del método `doInBackground` se extraen todos los elementos de la cola correspondiente, se construye una lista, y se representa en forma de *array*

JSON.

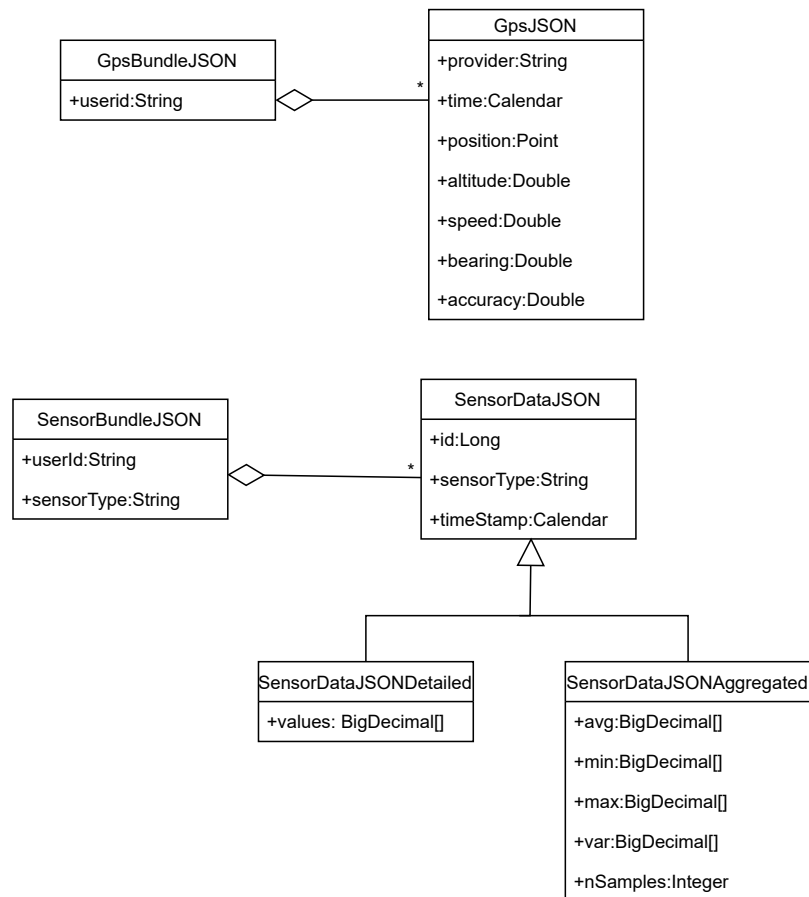


Figura 4.5: Modelo de datos de los ficheros JSON preparados para el envío

Dado que los datos de sensores son mucho más numerosos y que minimizar la cantidad de datos que se envían hacia el componente *SensorReceiver* es uno de los objetivos principales, pero ciertas aplicaciones pueden requerir datos detallados, se permiten dos opciones: en la primera opción se envían todos los datos de los sensores y en la segunda opción los datos se minimizan y se agregan. A continuación lo explicamos con más detalle. La primera opción consiste en enviar todos los eventos de los sensores que se encuentran en la cola y para cada elemento de la cola se crea un elemento de la lista. En este caso, la lista de eventos estará formada por objetos de la clase *SensorDataJSONDetailed*. En la segunda estrategia en lugar de enviar todos y cada uno de los eventos de sensores que se encuentran en la cola, se agregan

en bloques usando un intervalo temporal especificado por el desarrollador (su valor por defecto es un minuto). Dicho de otra forma, con esta estrategia se agregan todos los eventos de los sensores que se reciban en el mismo minuto y se crea, para ese minuto, un objeto que contenga el valor mínimo, el valor máximo, el valor medio, la varianza y el número de objetos que se agregaron. En este caso, el formato de este objeto JSON es el que corresponde a la clase `SensorDataJSONAggregated` descrito en la Figura 4.5. De esta manera, se logran dos importantes objetivos, primero, reducir la gran cantidad de datos capturada por los dispositivos móviles y segundo, simplificar el posterior procesamiento de la información en el componente de anotación semántica.

En ambos casos, una vez que se han representado los datos de la cola mediante un objeto en formato JSON, el siguiente paso consiste en comprimir ese objeto JSON en un fichero codificado en formato ZIP. Una vez que está comprimido, el fichero se guarda en el almacenamiento persistente del dispositivo móvil y se registra en una tabla de una base de datos *SQLite* en el dispositivo que un nuevo paquete de datos está disponible para el envío a *SensorReceiver*.

La Figura 4.5 muestra un diagrama de clases con el modelo conceptual de los ficheros JSON que contienen los datos recibidos de los dispositivos móviles. Un fichero JSON con un paquete de datos de `GPSPackageJSON` consiste en un objeto con el identificador del usuario (`userid`) y los datos propiamente dichos que se reciben como un *array* JSON de objetos. Cada uno de los objetos contiene el tipo de proveedor del que se ha obtenido la posición (`provider`), un instante de tiempo (`time`), una posición del tipo de datos geográfico *Point* (`position`), una altitud (`altitude`), una velocidad (`speed`), un rumbo (`bearing`) y la precisión de la ubicación (`accuracy`). Por otra parte, un fichero JSON con un paquete de datos de sensores (`SensorBundleJSON`) consiste en un objeto con el identificador del usuario (`userid`) y el tipo de sensor al que pertenecen los datos (`sensorType`). Permitimos dos tipos de recolectores de objetos para los datos de sensores, `SensorDataJSONAggregated` y `SensorDataJSONDetailed` dependiendo de si se decide recoger el detalle completo de los sensores o por el contrario se decide recoger los valores de los sensores y calcular las diferentes medidas de agregación en un intervalo de tiempo. Los datos propiamente dichos se reciben como un *array* JSON de objetos. En el caso de `SensorDataJSONDetailed` cada uno de los objetos contiene un instante de tiempo (`timeStamp`) y el valor del sensor en ese justo momento. En el caso de `SensorDataJSONAggregated` cada uno de los objetos contiene un instante de tiempo (`timeStamp`), el valor medio durante el minuto de los valores de los sensores (`avg`), el valor mínimo (`min`), el valor máximo (`max`), la varianza de los valores (`var`) y el número de valores que se han agregado en el minuto (`nSamples`).

La Figura 4.6 muestra un diagrama de clases con el detalle de la implementación del componente de envío de datos (*SensorSender*) al componente de recogida de datos (*SensorReceiver*). Para gestionar el proceso de envío de datos a través de

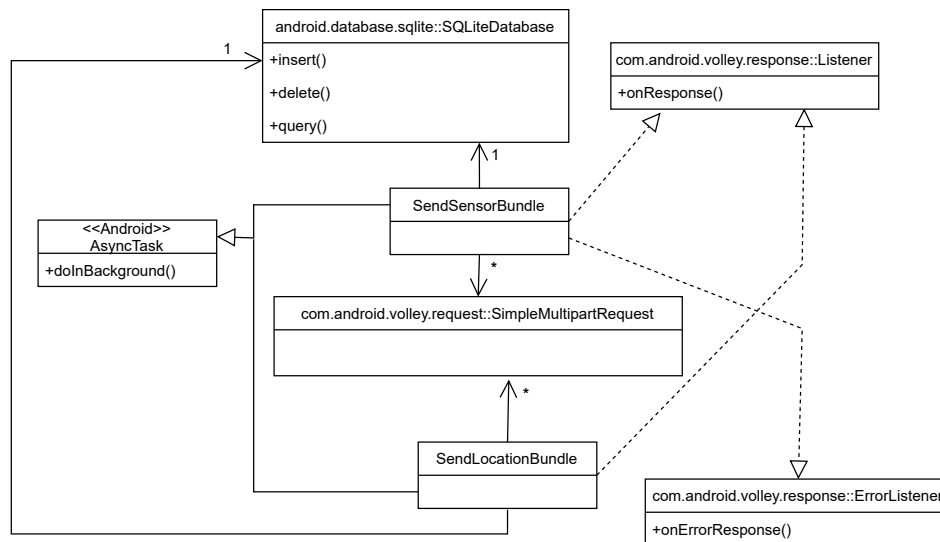


Figura 4.6: Diagrama de clases del componente de envío de datos (*SensorSender*)

la red utilizamos *Volley*², una librería desarrollada por *Google* que tiene como objetivo optimizar el envío de peticiones HTTP desde las aplicaciones *Android* hacia servidores externos. Concretamente, se ha implementado con *Volley* el envío de los datos hacia el componente *SensorReceiver*, se realiza una petición *GET* al servicio REST y se adjuntan los datos. El proceso se ejecuta de forma asíncrona con una frecuencia baja, por defecto se ha configurado cada treinta minutos, ya que estas tareas hacen uso de la red móvil y consumen bastante energía. Para implementarlo basta con crear dos clases que extiendan la funcionalidad de la clase `AsyncTask` (denominadas `SendLocationBundle` y `SendSensorBundle` en el diagrama). En la implementación del método `doInBackground` se consultan los elementos de la cola correspondiente, se construye una petición con *Volley*, y se entrega a la cola de envíos de *Volley*. Los datos se consultan de la cola almacenada en la base de datos *SQLite* y solo se borran si se recibe confirmación del componente *SensorReceiver* de que se han recibido correctamente. Para ello, estas clases deben ser observadores de las respuestas de *Volley* (`Listener` y `ErrorListener` en el diagrama) para poder ser notificadas si las peticiones se han recibido correctamente o no.

En conclusión, se ha diseñado un sistema que minimiza el gasto energético al empaquetar todos los elementos de la cola y comprimirlo en un solo paquete de datos usando el formato de archivo ZIP, de este modo a pesar de que el componente de

²<https://developer.android.com/training/volley>

captura de datos (*SensorCollector*) estará activo durante toda la jornada laboral del empleado solo se ejecuta de manera periódica. Además, este diseño permite minimizar la transferencia de datos basándose en dos técnicas, por un lado, se envían los datos comprimidos y por otro, se permite al desarrollador agregar los datos de los sensores de forma que se reduzca considerablemente el número de datos a enviar. Por último, el diseño del sistema es robusto frente a problemas de conectividad y los datos se mantienen seguros de forma persistente en una base de datos.

4.3. Componente de recogida de datos (*SensorReceiver*)

A continuación se describe el componente de recogida de datos (*SensorReceiver*). El objetivo de este componente es recoger los datos capturados por los dispositivos móviles. Para ello, debe implementar un servicio web que reciba los datos enviados por los dispositivos móviles, compruebe su validez, y en caso de que los datos sean correctos, confirme al dispositivo móvil su recepción y los almacene en la base de datos *Raw Data*. Hay que tener en cuenta que el componente de captura de datos (*SensorReceiver*) va a recibir la información de todos los dispositivos móviles de la empresa. Por ello se ha diseñado de tal forma que pueda dar servicio al mayor número posible de usuarios simultáneos.

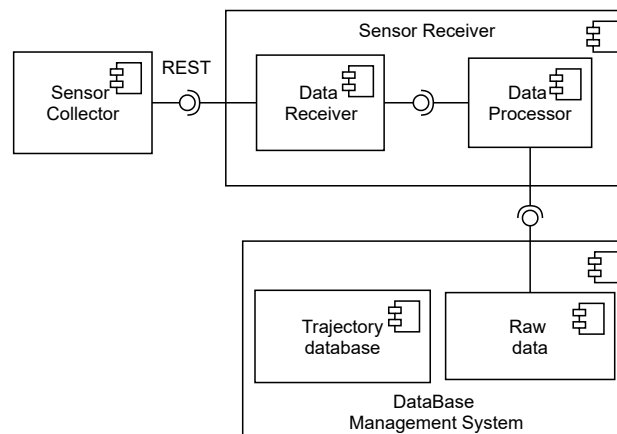


Figura 4.7: Arquitectura del componente de recogida de datos (*SensorReceiver*)

Tal y como se muestra en la Figura 4.7 el componente de captura de datos (*SensorReceiver*) se estructura en dos componentes de procesamiento. En el primer componente *Data Receiver* se reciben los datos de los dispositivos móviles, y en el

segundo componente *Data Processor* se procesan y se almacenan en la base de datos *Raw data*.

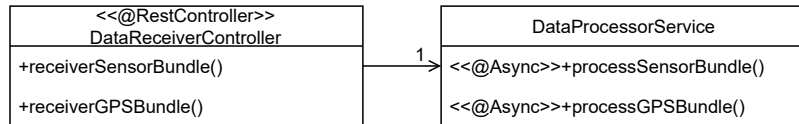


Figura 4.8: Diagrama de clases del componente de recogida de datos (*SensorReceiverr*)

Como se muestra en la Figura 4.8, el componente encargado de la recepción de los datos (*Data Receiver*) consiste en un controlador REST (*DataReceiverController*) que recibe el archivo ZIP que proviene del dispositivo móvil, y si la recepción ha sido correcta confirma inmediatamente al dispositivo móvil que los datos se han recibido y delega el procesamiento en la siguiente etapa. El componente de procesamiento de datos (*Data Processor*) implementa la clase (*DataProcessorService*). Comienza descomprimiendo el archivo ZIP recibido para extraer los datos que contiene y que están representados en formato JSON. A continuación, valida los datos y si son correctos los almacena en la base de datos (*Raw data*) y descarta el fichero ZIP y el fichero JSON.

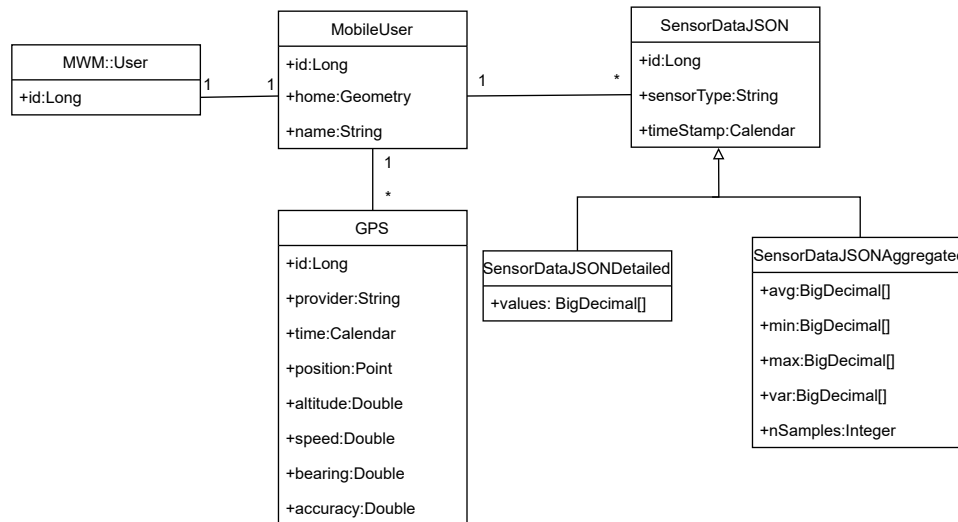


Figura 4.9: Datos de sensores y ubicaciones registrados por el dispositivo móvil

La Figura 4.9 muestra un diagrama de clases con el modelo de datos conceptual

para el almacenamiento de los datos de sensores y ubicación en la base de datos sin procesar. Para cada trabajador (representado por la clase `MobileUser`) se almacena su domicilio utilizando un tipo de datos `Geometry` de PostGIS (`home`). Esto permite almacenar el domicilio con precisión (utilizando un `Point`) o anonimizar el domicilio (usando un `Polygon`). Además, el trabajador está relacionado con el usuario del MWM. Esta separación del usuario en dos clases se realiza para aislar el módulo de anotación (`Annotator`) de las particularidades de cada implementación y sólo se asume que existe alguna tabla que identifique únicamente a los usuarios. Además, cada trabajador está relacionado con todos los valores recogidos por los sensores (`SensorDataJSON`). La información de los datos de los sensores puede ser de dos tipos y están representados por las clases `SensorDataJSONDetailed` y `SensorDataJSONAggregated`, la primera clase representa los datos de los sensores sin agregar y la segunda los datos de los sensores agregados. La clase padre (`SensorDataJSON`) almacena el tipo de sensor (`sensorType`) y el instante de tiempo en el que se recogieron los datos (`timeStamp`). La subclase `SensorDataJSONDetailed` almacena los valores de los sensores en cada instante de tiempo (`values`). La subclase `SensorDataJSONAggregated` almacena la media y varianza (`avg` y `var`) de los valores del sensor durante el intervalo de tiempo establecido, el valor mínimo y el valor máximo (`min` y `max`) registrados durante ese minuto, y el número de muestras (`nSamples`) que se agregaron en la medición. Por otra parte, se almacenan todas las ubicaciones de los trabajadores (representadas por la clase `GPS`) almacenando las variables recogidas por el GPS: el instante en el que se registra el valor (`time`), la ubicación geográfica usando el tipo de datos `Point` de `PostGIS` (`position`), la velocidad (`speed`), la altitud (`altitude`), el rumbo (`bearing`), la precisión (`accuracy`) y el proveedor de la ubicación (`provider`).

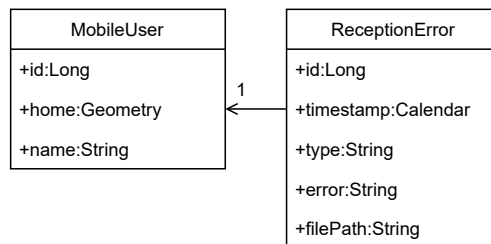


Figura 4.10: Modelo conceptual de la tabla de registro de incidencias

En caso de que alguno de los pasos de esta etapa no se pueda realizar correctamente, se registra el error en una tabla de la base de datos (*Raw data*) y se guarda el fichero ZIP para que un administrador del sistema pueda revisar que ha ocurrido y si los datos pueden recuperarse. El modelo conceptual de la tabla de incidencias se muestra en la Figura 4.10 (clase `ReceptionError`).

Para implementar la alta disponibilidad del componente de captura de datos

(*SensorReceiver*) se ha utilizado *Spring* REST. La diferencia clave entre un controlador desarrollado usando *Spring* MVC tradicional y el controlador de un servicio web REST desarrollado con *Spring* REST es que mientras que el controlador MVC tradicional se basa en la tecnología *View*, el controlador del servicio web *RESTful* simplemente devuelve el objeto y los datos del objeto se escriben directamente en la respuesta HTTP como JSON/XML. Esto hace que un servicio REST sea más apropiado para ser utilizado por los clientes al ser más ligero y más adaptado a sus necesidades. Por otra parte, desde la versión Java 7 del JDK el desarrollador no tiene que gestionar los hilos de ejecución (*threads*) de forma manual sino que la propia plataforma se encarga de gestionar grupos de hilos (*thread pools*) a los que se les puede enviar tareas para su ejecución. *Spring* amplía esta funcionalidad permitiendo que el desarrollador pueda indicar que una aplicación puede ejecutar tareas de forma asíncrona mediante la anotación `@EnableAsync`. La clase anotada con esta anotación puede configurar aspectos como el número mínimo y máximo de procesos concurrentes, así como muchos otros aspectos. Además, pueden anotarse métodos individuales para su ejecución de forma asíncrona con la anotación `@Async`. Al encontrar un método anotado de esta manera, *Spring* se ocupará de asignar su ejecución a un hilo independiente y retornar el control de forma inmediata al invocador del método. La capacidad de indicar a *Spring* que ciertos métodos se ejecutan de forma asíncrona permitirá aumentar la disponibilidad del componente de captura de datos (*SensorReceiver*) ya que evitará que el cliente del servicio REST se bloquee esperando por la respuesta del sistema central.

En conclusión, se ha diseñado un componente de captura de datos (*SensorReceiver*) que permite una alta disponibilidad permitiendo que el gran número de empleados de la empresa capturen y envíen simultáneamente los datos de los sensores a este componente. Por otra parte, el componente también permite el registro de incidencias almacenando cualquier error en la base de datos. Además guarda el paquete ZIP evitando que se pierda información.

4.4. Componente de anotación semántica (*Annotator*)

A continuación se describe el componente de anotación semántica (*Annotator*). El objetivo del componente es anotar las actividades semánticamente. Para ello, utiliza los datos recogidos por los sensores, el GPS, la información MWM y contextual. Este proceso se realiza mediante el uso de taxonomías de actividades. La descripción detallada de cada uno de los pasos para la anotación semántica se encuentra en la Sección 5. En esta sección presentamos la arquitectura general del componente para comprender el flujo de datos y el resultado.

Dado que en *Android* el marco temporal y la tasa de recogida de datos de ubicación y de datos de sensores son completamente independientes, este

componente debe encargarse de realizar la tarea de fusionar en un marco temporal homogéneo los datos de las dos fuentes de datos. Por otra parte, dado que el proceso de anotación semántica necesita información geográfica de contexto, este componente define la forma en que esta información se recupera y se pone a disposición del componente de anotación semántica. Además, dado que el proceso de etiquetado también necesita la información *Mobile Workforce Management*, este componente debe especificar la forma en la que se recibe la información del trabajador desde el MWM. Otro punto a tener a cuenta es que el proceso de anotación semántica utiliza el concepto de *taxonomía de actividades* para definir las actividades a identificar y especificar la forma de hacerlo.

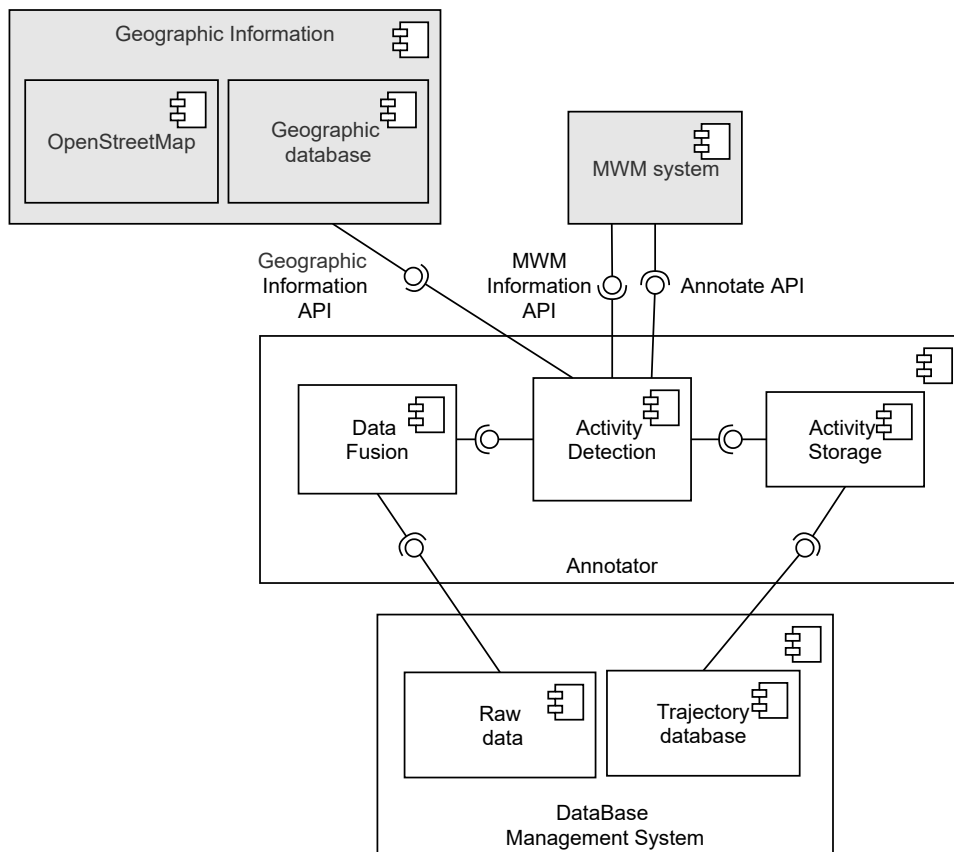


Figura 4.11: Arquitectura del componente de anotación semántica (*Annotator*)

La Figura 4.11 muestra la arquitectura del componente de anotación semántica.

El componente de anotación semántica se estructura en tres componentes de procesamiento. El componente *Data Fusion* extrae los datos de los sensores y las posiciones GPS desde la base de datos *Raw data* y a continuación los fusiona obteniendo una lista de unidades homogéneas que denominaremos segmentos. El componente *Activity Detection* recibe la lista de segmentos procedentes de *Data Fusion*, extrae la información de contexto de los componentes *Context Information* y *MWM System*, y anota los segmentos con actividades utilizando una taxonomía de actividades (este proceso se describirá con más detalle en el Capítulo 5). Finalmente, el componente *Activity Storage* agrupa los segmentos para construir actividades almacenadas en la base de datos *Trajectory database*.

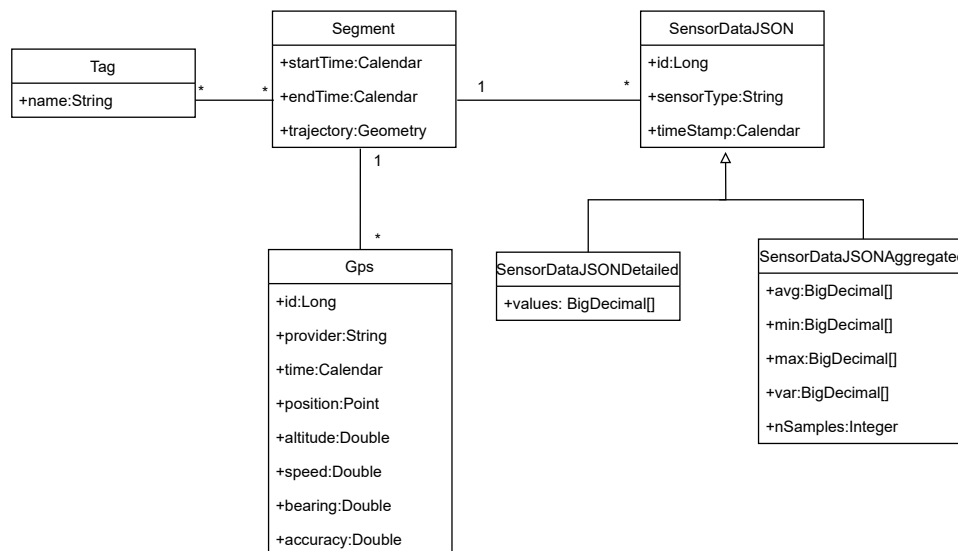


Figura 4.12: Diagrama de clases de un Segmento

El componente *Data Fusion* tiene como objetivo realizar la fusión de los datos de los sensores y las localizaciones (*Raw data*). Como resultado se obtienen los *segmentos*: unidades homogéneas que agrupan los valores de los sensores y las ubicaciones GPS. La Figura 4.12 muestra la estructura de datos que se utiliza para representar un segmento (**Segment**). Cada segmento tiene una duración arbitraria. Para ello, cada segmento almacena un tiempo inicial y un tiempo final del intervalo (**startTime** y **endTime** respectivamente), la trayectoria seguida por el usuario en ese intervalo (**trajectory**) usando un tipo de datos geográfico *Geometry* que podrá ser un *MultiPoint* (y que se utilizará en los casos en los que no se detecte movimiento dirigido) o un *LineString* (y que se utilizará en los casos en los que se detecte movimiento dirigido). Además, cada segmento está asociado con todas las ubicaciones individuales en las que el usuario se encontraba en

ese intervalo de tiempo (la colección de objetos de clase `GPS`). Por otra parte, cada segmento tiene asociado los datos de los sensores en ese intervalo de tiempo (clase `SensorDataJSON`). Por último, para dar soporte al proceso de anotación semántica, se permite que cada segmento esté asociado con un conjunto de etiquetas (representadas por la clase `Tag`). Durante el proceso de anotación estas etiquetas no representan las actividades semánticas, sino que son anotaciones para utilizar en el proceso de etiquetado y que se usan para determinar las actividades en el último paso del proceso.

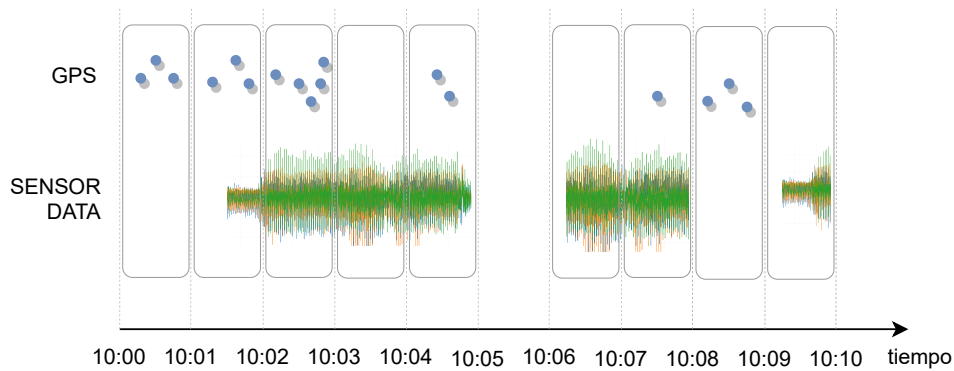


Figura 4.13: Lista de ubicaciones y datos de sensores

En la Figura 4.13 se puede ver un esquema de los datos que constituyen el punto de partida del componente *Data Fusion*. El eje horizontal representa el tiempo dividido en intervalos de un minutos. En la parte superior se muestran los datos recogidos por el GPS y en la parte inferior una gráfica con la señal de aceleración lineal recogida por el sensor del móvil. Se puede observar que el marco temporal y la tasa de muestreo de los datos de los sensores no es la misma que el marco temporal de los datos recogidos por el sensor GPS. Mientras que para los datos de sensores se recogen varios datos por segundo, para los datos GPS se recoge una ubicación cada varios segundos. En la figura también se muestran como rectángulos de bloques redondeados los *segmentos* que se utilizarán como punto de partida para el proceso de anotación.

Para llevar a cabo el algoritmo del proceso de fusión hay que tener en cuenta diferentes casuísticas. Por una parte, el componente tiene que determinar el inicio y el final del día del trabajador, como se puede ver en la Figura 4.13, en este caso particular el inicio del día lo marcan las primeras posiciones GPS y el final del día se detecta gracias a los últimos de los sensores. En el caso general, podrían ocurrir otras combinaciones que también deben estar contempladas en el algoritmo. Por otra parte, hay que tener en cuenta que es posible que no existan simultáneamente datos de los dos tipos en todos los segmentos. Por ejemplo, como se puede ver en

la Figura 4.13, en el cuarto, sexto y noveno segmentos sólo hay datos de sensores, esto puede deberse a que el trabajador se encuentra en el interior de un edificio sin cobertura GPS. Sin embargo, en el primer y en el octavo segmento sólo hay datos de GPS, esto sucede por ejemplo, cuando se ha producido un error de recepción de datos. Por último, hay que tener en cuenta que existe la posibilidad de que existan intervalos del día en los que no haya datos de ningún tipo. Por ejemplo, como se puede ver en la puede ver la Figura 4.13, entre el quinto y el sexto segmentos hay un intervalo sin datos.

Algoritmo 4.1 Obtener los segmentos a partir de ubicaciones GPS y datos de sensores

```

function MERGE(gpsData: List of GPS, sensorData: List of sensor data)
  segments  $\leftarrow$   $\emptyset$ 
  segmentWidth  $\leftarrow$  1 minute
   $\triangleright$  Se inicializa el comienzo y final de la lista de segmentos
  startTime  $\leftarrow$  minimal time of GPS record and sensor data
  endTime  $\leftarrow$  maximum time of GPS record and sensor data
   $\triangleright$  Se construye la lista completa de segmentos
  segments  $\leftarrow$  createEmptySegments(startTime, endTime, segmentWidth)
   $\triangleright$  Se asigna a cada segmento el valor correspondiente de los sensores
  for all s  $\in$  segments do
    sdInSegment  $\leftarrow$  findSensorData(sensorData, s.startTime, s.endTime)
    locInSegment  $\leftarrow$  findLocations(locationData, s.startTime, s.endTime)
    if sdInSegment.isEmpty() && locInSegment.isEmpty() then
      segmentsToBeRemoved.add(s);
    else
      s.addSensorData(sdInSegment);
      s.addLocationData(locInSegment);
    end if
  end for
  segments.remove(segmentsToBeRemoved);
end function

```

Los pasos para obtener los segmentos a partir de las ubicaciones GPS y datos de los sensores se explican en el Algoritmo 4.1. Este algoritmo se divide en varias fases, la primera tiene como objetivo obtener el inicio y el final del día del trabajador. La segunda fase tiene como objetivo asignar a cada uno de los segmentos la fecha de inicio y fin, de modo que cada segmento tenga la misma duración exacta (*segmentWidth* en el algoritmo). La siguiente fase tiene como objetivo asignar a cada segmento el valor de los sensores y de las ubicaciones geográficas en cada intervalo de tiempo. Si en un segmento no hay datos de sensores ni ubicaciones geográficas, se descarta en la última fase del algoritmo.

Una vez que el componente *Data Fusion* construye la lista de segmentos, el componente *Activity Detection* tiene como objetivo asignar una actividad a cada segmento recibido. De forma conceptual, el componente *ActivityDetection* recibe las ubicaciones GPS y los datos de los sensores mediante una lista de segmentos y utiliza esta información, junto con la información de contexto del empleado, para calcular la actividad que el empleado ha realizado en cada segmento asignándole una etiqueta (**Tag**). El detalle de este proceso se describe en el Capítulo 5. En este Capítulo simplemente mencionaremos que para la realización del proceso la información de contexto es muy importante. Esta información de contexto puede ser de dos tipos: la proporcionada por el sistema MWM (*MWM system* en la Figura 4.11) y la información geográfica de apoyo al proceso.

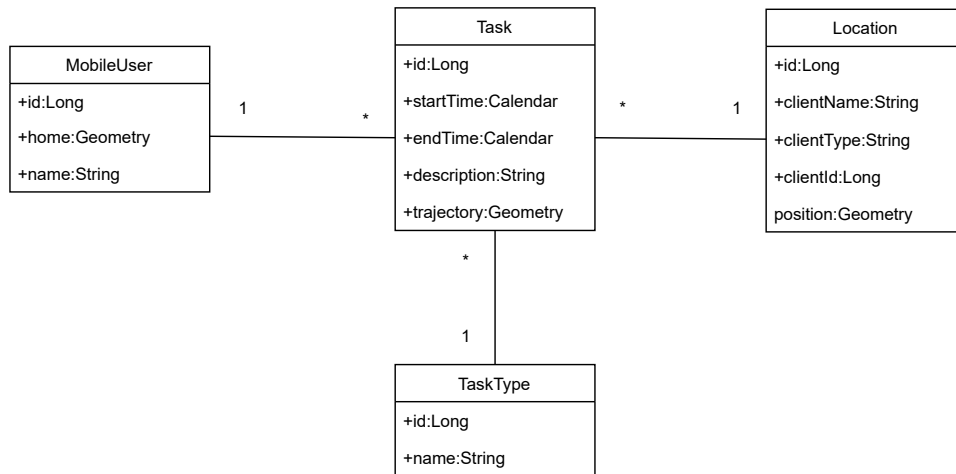


Figura 4.14: Modelo de datos para la información procedente del sistema MWM

La Figura 4.14 muestra un modelo de datos de la información mínima requerida para el proceso de anotación. En primer lugar, es necesario conocer la información de todos los trabajadores de la empresa (representada en la clase *MobileUser*). Además, debemos conocer la agenda de cada trabajador (representada por la clase *Task*). Esta clase describe, para cada tarea realizada por cada trabajador, la hora de inicio y fin programada (`startTime` y `endTime` respectivamente), el tipo de tarea (representado por la clase *TaskType*), una descripción de la tarea (`description`), y la ruta planificada previamente por el sistema MWM (`trajectory`) que el empleado debe seguir para llegar a la ubicación en la que debe realizar la tarea. La clase *Task* también está asociada con un objeto de la clase *Location* el cual describe el lugar al que el empleado tiene que desplazarse. Cada ubicación está descrita por el nombre del cliente, la posición geográfica de la ubicación y una referencia al cliente. Dado

que cada sistema MWM puede gestionar los clientes de manera distinta, en lugar de asumir que existe una tabla de clientes, se representará la asociación mediante un tipo (`clientType`) y un identificador (`clientID`). Cada implementación concreta deberá dar semántica propia a estos dos atributos.

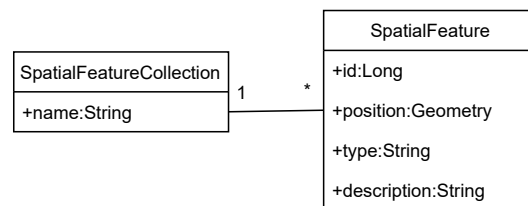


Figura 4.15: Modelo de datos para la información de contexto

La Figura 4.15 muestra el modelo de datos de la información geográfica de contexto que se puede proporcionar al proceso de anotación semántica. Esta información se considera externa al sistema de anotación, pero es necesaria en el proceso de etiquetado. Cada elemento de información geográfica del contexto se representa mediante un objeto de la clase `SpatialFeature` que contiene información geográfica del objeto (atributo `position`) mediante un atributo de tipo `Geometry` (de forma que se puedan representar objetos de cualquier tipo de datos geográfico), así como información descriptiva (un tipo utilizando el atributo `type` y una descripción utilizando el atributo `description`). Los objetos geográficos se agrupan en colecciones con nombre (la clase `SpatialFeatureCollection`) y la información del contexto consiste en un conjunto de estas colecciones. A modo de ejemplo, la información geográfica de contexto para un proceso de anotación concreto puede ser la red de carreteras representada mediante un `SpatialFeatureCollection` denominado `redDeCarreteras` que agrega todos los tramos de carreteras (cada uno un `LineString` representado mediante un `SpatialFeature`). Este mismo proceso también podría utilizar otro `SpatialFeatureCollection` denominado `gasolineras` que agrega todas las estaciones de servicio representadas mediante `SpatialFeatures` utilizando valores de tipo `Point`. En la implementación del sistema proporcionamos dos formas de recuperar la información geográfica de contexto: la primera es accediendo a una base de datos `PostGIS` local mediante consultas de SQL espacial y la segunda mediante consultas a `OpenStreetMap` utilizando el `API Overpass`. En ambos casos, el resultado es un conjunto de objetos geográficos representados mediante la clase `SpatialFeature` agregados por un objeto de la clase `SpatialFeatureCollection`.

Una vez que el componente *Activity Detection* realiza la detección de actividades, envía al componente *Activity storage* una lista de segmentos con una etiqueta (`Tag`) que describe la actividad que el empleado estaba desempeñando en ese segmento. El componente *Activity storage* es el responsable de agrupar los segmentos consecutivos

que se correspondan con la misma actividad, y almacenar en la base de datos las actividades resultantes (*Trajectory database*).

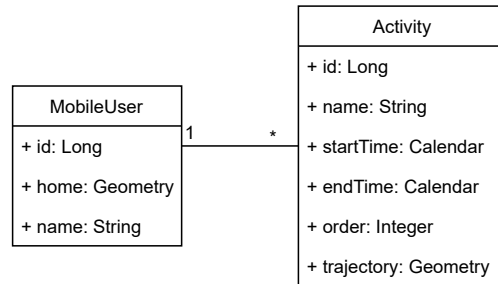


Figura 4.16: Modelo de datos para las actividades identificadas

El diagrama de clases de la Figura 4.14 muestra el modelo conceptual para el almacenamiento de las actividades que son almacenadas en la base de datos multidimensional. Cada actividad almacena el nombre de la actividad, el instante de inicio y de fin de la actividad, el orden relativo en el que se realiza la actividad con respecto a las demás actividades del mismo día, el usuario al que corresponde la actividad y la trayectoria seguida por el usuario usando un tipo de datos geográfico *Geometry* que podrá ser un *MultiPoint* o un *LineString* (en caso de que se detecte un desplazamiento).

4.5. Conclusiones

En este capítulo se ha descrito la arquitectura del sistema de identificación de actividades semánticas. Consideramos que la arquitectura descrita alcanza los requisitos planteados inicialmente y da solución a diferentes problemas. Por una parte, logra minimizar la transferencia de información gracias al diseño de un método que permite reducir la gran cantidad de datos que se recogen por el componente de captura de datos en el dispositivo móvil. De lo contrario, si nuestro sistema no presentase esta arquitectura y se transfiriesen todos y cada uno de los datos recogidos hacia sistema central, se consumiría un elevado porcentaje de los datos disponibles en la tarifa de telefonía móvil de los empleados.

Por otra parte, nuestra arquitectura logra una alta eficiencia energética en su funcionamiento. En el caso de que no se hubiese conseguido esta característica en nuestro componente de captura de datos, existiría el riesgo de que su funcionamiento consumiese de forma excesiva la batería de los dispositivos de los empleados, ya que la aplicación está activa durante toda la jornada laboral del empleado.

Otro aspecto importante logrado en nuestro componente es la robustez frente a problemas de conectividad. A pesar de que la mayor parte de los empleados trabajan

en un entorno urbano, existe un alto porcentaje de los mismos que realizan su actividad en el entorno rural. En estos casos, la disponibilidad de conexión móvil es baja, por lo tanto, el componente presenta un diseño que permite no tener conexión permanente con el sistema central. Esto se ha logrado gracias a que el diseño del componente permite almacenar los datos en el dispositivo móvil hasta que se disponga de conexión de nuevo y se puedan enviar.

Por último, este sistema logra unificar los datos procedentes de diferentes fuentes, desde la información de contexto hasta los datos de los sensores y la ubicación GPS. Esto es posible gracias a que el diseño da solución al problema de las distintas granularidades que presentan los sensores y los datos de localización, permitiendo agregarlos y fusionarlos.

En conclusión, consideramos que la arquitectura de este sistema cumple con los diferentes requisitos que harán posible la posterior identificación de actividades con un alto nivel de precisión, pero que además, lo convierten en un sistema robusto frente a fallos de conexión y que minimiza tanto el gasto energético como la transferencia de datos.

Capítulo 5

Metodología de anotación de actividades semánticas

Una vez presentada la arquitectura general de nuestra propuesta, en este capítulo nos centraremos en describir su componente fundamental: el componente de detección de actividades semánticas. Este componente debe proporcionar soluciones a tres problemas. En primer lugar, debe definir un modelo que permita representar las actividades que deben ser identificadas y la forma de identificarlas. Para definir este modelo hemos creado el concepto de *taxonomías de actividades* que se describen en la Sección 5.1, un modelo simple e intuitivo que permite representar las actividades semánticas. En segundo lugar, debe definir de forma detallada los elementos que se pueden utilizar para construir las taxonomías de actividades. En la Sección 5.2 se abarca la definición de estos elementos y se detalla el *lenguaje de especificación de patrones de actividad* que se utiliza en las taxonomías para decidir qué actividad se está realizando: un lenguaje flexible, intuitivo y muy expresivo. Y finalmente, en tercer lugar, debe definir un algoritmo para identificar actividades utilizando una taxonomía. Este algoritmo se describe en la Sección 5.3.

5.1. Taxonomía de actividades

En esta sección definiremos y describiremos en primer lugar de manera informal el concepto de taxonomía de actividades (Sección 5.1.1). A continuación describiremos la representación formal de una taxonomía de actividades en nuestro sistema (Sección 5.1.2).

5.1.1. Definición de una taxonomía de actividades

Una herramienta que se utiliza desde hace años en investigación operativa en general, y en el análisis de decisiones en particular, son los árboles de decisión. Un árbol de decisión es una representación esquemática similar a un árbol que representa un mapa de los posibles resultados que se pueden obtener. Un árbol de decisión está formado principalmente por ramas y nodos, donde cada rama se bifurca en función de los valores tomados por una o varias variables. En cada nodo intermedio se toma una decisión en función de si cumplen las condiciones especificadas en la rama asociada. Por último, cada nodo hoja representa una etiqueta de clase.

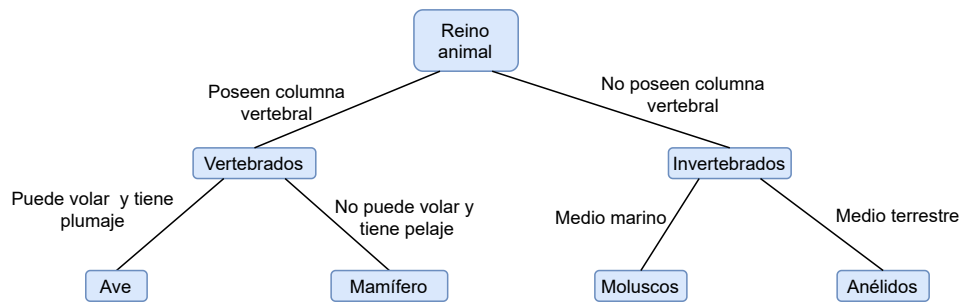


Figura 5.1: Taxonomía clasificación animales vertebrados e invertebrados

La Figura 5.1 muestra un ejemplo de árbol de decisión que se puede usar para realizar la clasificación taxonómica de un animal en una especie. El ejemplo dista mucho de ser completo y correcto, pero permite ilustrar el proceso. La entrada al nodo raíz está formada por el conjunto de todos los animales. En los nodos intermedios se toma una decisión en función de si cumple las condiciones que se indican en las ramas de salida, y en los nodos hoja se le asigna una clase. Por ejemplo, aquellos seres vivos que cumplan la condición *posee columna vertebral* se pasan al primer nodo hijo y son considerados *animales vertebrados*, por el contrario, aquellos seres vivos que no cumplan la condición, pasan al siguiente nodo y son considerados *invertebrados*. En el siguiente nivel del árbol, los animales que han sido clasificados como vertebrados y tienen plumas son clasificados como aves. Por el contrario, aquellos animales que no cumplan esta condición, pasan al siguiente nodo y se comprueba qué animales cumplen la condición *si puede volar y tienen pelaje*, en ese caso, se trata de animal que pertenece a la clase mamífero. En el subárbol izquierdo el proceso para la clasificación de animales será análogo.

Basándonos en los conceptos de árboles de decisión y de clasificación taxonómica, definimos el concepto de *taxonomía de actividades* como un árbol formado por nodos y arcos que tiene como entrada un conjunto de segmentos (tal y como se describen en la Sección 4.4) y que tiene como objetivo asignar a cada segmento una etiqueta que

identifique la actividad realizada en el segmento del mismo modo que se hace en el ejemplo de la clasificación de animales. Cada nodo de la taxonomía de actividades se corresponde con una *tarea de procesamiento* que se debe realizar sobre la secuencia de segmentos recibidos por el nodo. Por tanto, cada tarea de procesamiento recibe una secuencia de segmentos y realiza un procesamiento diferente sobre ella en función de su tipo:

- **Etiquetado.** Esta tarea de procesamiento se encarga de asignar una determinada etiqueta a todos los segmentos recibidos como entrada.
- **Decisión.** Esta tarea de procesamiento distribuye los segmentos que ha recibido como entrada a sus nodos hijo dependiendo del resultado de la evaluación del predicado asociado a los arcos que lo unen con los nodos hijo. Sólo los segmentos que cumplen el predicado se envían al nodo hijo para ser procesados.
- **Agrupar.** Esta tarea de procesamiento comprueba si entre los segmentos que recibe existen grupos de segmentos consecutivos cuya duración total supere un valor límite. A todos los segmentos que forman un grupo les asocia una etiqueta concreta. A los que no forman un grupo les asocia una etiqueta diferente. Finalmente, la tarea para permitir tomar una decisión en base al tiempo total de duración de un conjunto de segmentos (esto es, se comporta como un nodo de *Decisión*).
- **ParadoYMovimiento.** Esta tarea de procesamiento agrupa los segmentos que recibe en dos tipos: aquellos en los que la persona se encuentra parada, y aquellos en los que la persona se encuentra en movimiento. A continuación, asocia a cada segmento de cada grupo una etiqueta concreta. Finalmente, se comporta como un nodo de *Decisión*.

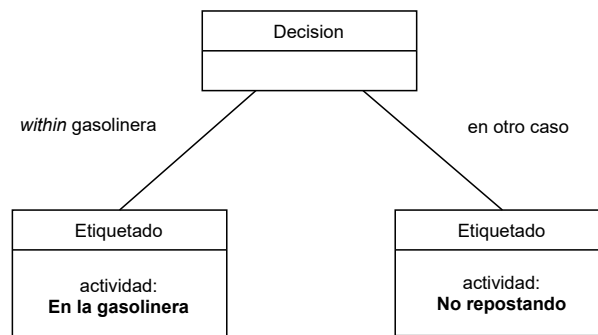


Figura 5.2: Ejemplo de taxonomía usando *Decisión* y *Etiquetado*

Vamos a ir explicando el funcionamiento de cada uno de los tipos de nodos con pequeños ejemplos ilustrativos, cada uno de ellos obviará los detalles complejos para intentar priorizar una explicación lo más didáctica posible. La Figura 5.2 muestra un ejemplo de taxonomía de actividades en la que el nodo raíz es un nodo de tipo *Decisión*, y los nodos hoja son de tipo *Etiquetar*. El nodo de tipo *Decisión* tiene la funcionalidad de distribuir los segmentos que ha recibido como entrada a sus nodos hijo dependiendo del resultado de la evaluación de los predicados asociados a los arcos que lo unen con los nodos hijo. El predicado asociado al primer arco es *within gasolinera*. Los predicados disponibles los describiremos en la Sección 5.2. Por el momento basta con decir que este predicado evalúa si las posiciones GPS asociadas al segmento cumplen el predicado espacial *st_within* respecto a la colección de objetos geográficos denominada *gasolinera* disponible en la información de contexto (esto es, una *SpatialFeatureCollection* tal y como se describió en la Sección 4.4). Los segmentos que cumplan el predicado se pasarán al nodo de la izquierda. Los segmentos que no cumplan la condición serán evaluados respecto al predicado del arco derecho, que en este caso es *en otro caso*. Este es un predicado que siempre se evalúa a cierto. Por tanto, los restantes segmentos se pasarán al nodo de la derecha. En las dos hojas de la taxonomía se encuentran nodos de tipo *Etiquetado* que son los encargados de asignar una etiqueta a cada uno de los segmentos que reciben como entrada. El resultado de este proceso es que el conjunto de segmentos inicial se dividen en dos grupos, los que cumplen el predicado *within gasolinera* se etiquetan con la actividad *En la gasolinera*, los restantes con la actividad *No repostando*.

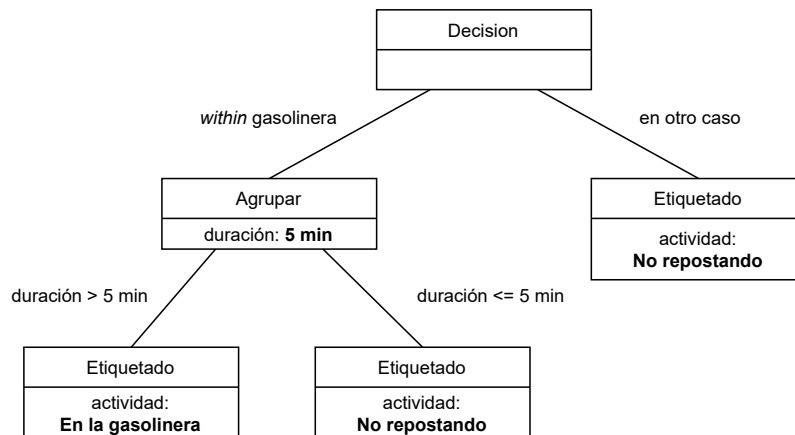


Figura 5.3: Ejemplo de taxonomía usando el nodo *Agrupar*

En ocasiones es necesario que las condiciones de los segmentos tengan una duración mínima para ser consideradas como una actividad concreta. Por ejemplo, en el caso anterior, para considerar que una persona está repostando en una

gasolinera el tiempo debe ser superior a un umbral (por ejemplo, 5 minutos). Si el tiempo es menor podría ser, por ejemplo, que la persona se ha detenido en un semáforo próximo a una gasolinera. Para identificar estas actividades hemos definido el nodo de tipo *Agrupar*. Como se puede observar en el ejemplo de la Figura 5.3, hemos modificado la taxonomía añadiendo un nodo de tipo *Agrupar* en la rama izquierda de la taxonomía. Este nodo recibe los segmentos que cumplen el predicado *within gasolinera* y evalúa si existen grupos consecutivos de al menos 5 minutos de duración (configurables) que tengan la misma etiqueta. En caso de que esto ocurra, cada grupo de segmentos se enviará hacia el nodo hijo de la izquierdo que los etiquetará como *en la gasolinera*. Los segmentos que no formen parte de un grupo consecutivo de más de 5 minutos se enviarán al nodo hijo de la derecha y se etiquetarán como *no repostando*. Por tanto, el nodo de tipo *Agrupar* nos permite identificar grupos de segmentos consecutivos con una duración mayor a una dada.

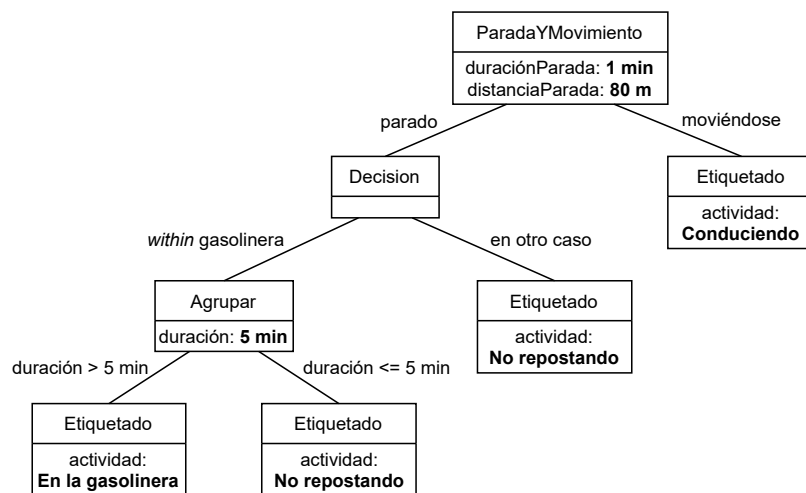


Figura 5.4: Ejemplo de taxonomía usando el nodo *ParadaYMovimiento*

Un último tipo de nodo que es necesario en la identificación de actividades semánticas es el nodo *ParadaYMovimiento*. Este tipo de nodo analiza qué segmentos de los que recibe se corresponden con una parada y cuáles con un desplazamiento significativo comprobando si la ubicación geográfica es la misma durante un periodo de tiempo mínimo. En la Figura 5.4 hemos ampliado nuestra taxonomía de ejemplo añadiendo un nodo de tipo *ParadaYMovimiento* en la raíz. Este nodo analiza todos los segmentos que recibe y los considera una parada si superan el umbral de tiempo y no sobrepasan el umbral de distancia de desplazamiento que se indican (en el ejemplo, más de un 1 minuto de tiempo y menos de 80 metros de distancia). Los segmentos que cumplen el criterio se envían al nodo de la izquierda y se etiquetan como en el ejemplo anterior. Los segmentos que no cumplen el criterio se envían al

nodo de la derecha y se etiquetan como *Conduciendo*. Por tanto, el tipo de nodo *ParadaYMovimiento* permite identificar en una trayectoria los episodios de parada y movimiento que son tan comunes en todos los trabajos del estado del arte.

5.1.2. Representación formal de una taxonomía de actividades

Una vez definidas y presentadas de manera informal las taxonomías de actividades, en esta sección describiremos de manera formal como se representa una taxonomía de actividades de actividades.

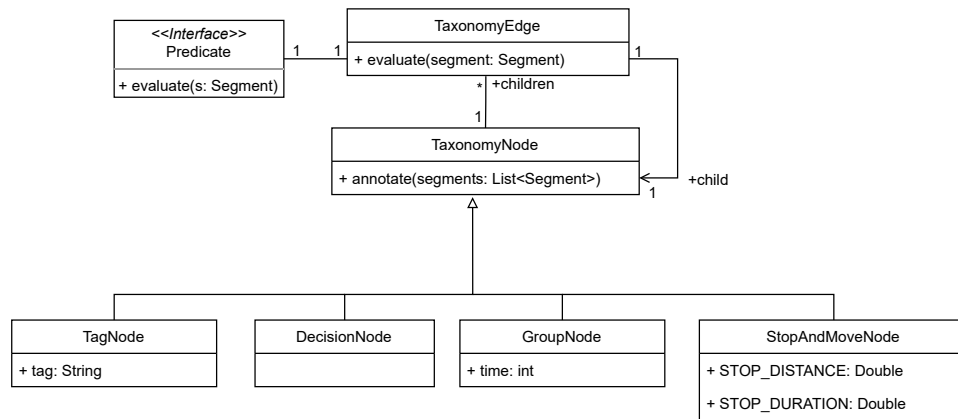


Figura 5.5: Diagrama de clases de una taxonomía de actividades

En la Figura 5.5 se muestra el diagrama de clases que representa una taxonomía de actividades. La clase central es *TaxonomyNode*, que describe un nodo de la taxonomía. Esta clase tiene un método llamado *annotate*, el cual recibe una lista de segmentos a los que le atribuye una etiqueta que describe la actividad realizada por el empleado. La clase *TaxonomyNode* tiene una lista de objetos pertenecientes a la clase *TaxonomyEdge*, esta representa cada uno de los arcos de salida del nodo. Cada arco (*TaxonomyEdge*) ofrece un método *evaluate*, el cual recibe un segmento y determina si cumple la condición o no asociada al arco. Esta condición está representada en el diagrama de la Figura 5.5 por la clase *Predicate*, la cual se describe detalladamente en la siguiente Sección 5.2. Por último, la clase *TaxonomyEdge* está asociada con un nuevo objeto de la clase *TaxonomyNode*, esta representa el nodo hijo que se alcanza recorriendo el arco y al que se le pasarán todos los segmentos que cumplan la condición.

La clase *TaxonomyNode* se especializa en cuatro clases diferentes, una para cada tipo de nodo descrito en la Sección 5.1.1. El nodo *Etiquetar* se representa mediante la clase *TagNode*, el nodo *Decisión* mediante la clase *DecisionNode*,

el nodo *Agrupar* mediante la clase `GroupNode` y el nodo *ParadaYMovimiento* mediante la clase `StopAndMoveNode`. Los algoritmos 5.1, 5.2, 5.3 y 5.4 muestran respectivamente los pseudocódigos de los algoritmos *TagNode*, *DecisionNode*, *GroupNode* y *StopAndMoveNode*. A continuación vamos a realizar una breve descripción de cada uno de ellos.

Algoritmo 5.1 Algoritmo de `TagNode`

```

function TAG(segmentos: Lista de segmentos)
  for all segmento  $\in$  segmentos do
    asignarEtiqueta(segmento, tag)
  end for
end function

```

El algoritmo 5.1, que corresponde con la clase `TagNode`, es el más sencillo. Simplemente asigna a cada segmento que recibe la etiqueta indicada en el atributo `tag` de la clase.

Algoritmo 5.2 Algoritmo de `DecisionNode`

```

function DECISION(segmentos: Lista de segmentos)
  for all arco  $\in$  nodo.getArcos() do
    listaSegmentosCierto  $\leftarrow$  arco.evaluar(segmentos)
    arco.nodoHijo.anotar(listaSegmentosCierto)
    segmentos.removeAll(listaSegmentosCierto)
  end for
end function

```

El algoritmo 5.2, que corresponde con la clase `DecisionNode`, recibe una lista de segmentos y evalúa, por orden, el predicado asociado a cada uno de los arcos. La lista de segmentos que evalúan como cierto con respecto al predicado de un arco se envía al nodo hijo de ese arco para que sea procesado y a continuación se eliminan de la lista de segmentos disponibles. De esta forma, cada segmento solo es evaluado por uno de los arcos de un nodo de tipo *Decisión*.

El algoritmo 5.3, que se corresponde con la clase `GroupNode`, es un poco más complejo que los dos algoritmos descritos anteriormente. Este algoritmo consta de dos pasos. En el primero se crean grupos a partir de los segmentos recibidos por parámetro. Para ello, se recorre la lista de segmentos buscando grupos cuyo tamaño superen la duración mínima indicada por el atributo `time`. En caso de formar un grupo, se les asigna la etiqueta indicada en el atributo `trueTag`. En caso de no formar un grupo, se les asigna la etiqueta indicada en el atributo `falseTag`. En la descripción del algoritmo se deja especificado como comentario que el último grupo que se intenta montar tendría que ser evaluado, pero no se especifica en el algoritmo por razones de espacio y claridad. El segundo paso consiste en realizar la misma

Algoritmo 5.3 Algoritmo de GroupNode

```

function GROUP(segmentos: Pila de segmentos)
  grupoActual ← segmentos.pop()
  posibleGrupo ← nueva lista
  posibleGrupo.add(grupoActual)
  while segmentos.empty() == false do
    segmentoActual ← segmentos.pop()
    if segmentoActual.fechaInicial == grupoActual.fechaFin then
      ▷ Es continuación del anterior
      grupoActual.fechaFin ← segmentoActual.fechaFin
      posibleGrupo.add(segmentoActual)
    else
      ▷ No es continuación del anterior
      esGrupo ← grupoActual.duration() > time
      for all segmento ∈ segmentosActuales do
        if esGrupo then
          segmento.addTag(trueTag)
        else
          segmento.addTag(falseTag)
        end if
      end for
      posibleGrupo.removeAll();
      posibleGrupo.add(segmentoActual)
      grupoActual ← segmentoActual
    end if
  end while
  ▷ Comprobar si el último posibleGrupo podría ser un grupo
  ▷ Realizar las operaciones de decisión igual que en el algoritmo 5.2
end function

```

evaluación de los segmentos que se realiza en el nodo `DecisionNode` y descrito en el algoritmo 5.2, para distribuir los segmentos entre los nodos referenciados por los arcos del nodo. En el algoritmo este paso se deja comentado por motivos de espacio y claridad.

Algoritmo 5.4 Algoritmo de StopAndMoveNode

```

function ANOTARPARADOYMOVIMIENTO(segmentos: Lista de segmentos)
  todasPosiciones  $\leftarrow$  recuperarPosicionesGPS(segmentos)
  puntosParado, posiblesPuntosParado  $\leftarrow$   $\emptyset$ 
  puntosMovimiento, posiblesPuntosMovimiento  $\leftarrow$   $\emptyset$ 
  centroCluster  $\leftarrow$  primera posición GPS
  for all posicionGPS  $\in$  todasPosiciones do
    if distancia(posicionGPS, centroCluster) < distanciaParada then
      posiblesPuntosParado.add(posicionGPS)
    else
       $\triangleright$  El punto está demasiado lejos del anterior
      if duracion(posiblesPuntosParado) > paradaDuracion then
        if esMovimientoValido(posiblesPuntosMovimiento) then
          puntosMovimiento.addAll(posiblesPuntosMovimiento)
        end if
        posiblesPuntosMovimiento  $\leftarrow$   $\emptyset$ 
        puntosParado.addAll(posiblesPuntosParado)
      else
         $\triangleright$  Duración parada demasiado corta
        posiblesPuntosMovimiento.addAll(posiblesPuntosParado)
      end if
    end if
    potencialesPuntosParado.removeAll()
    centroCluster  $\leftarrow$  posicionGPS
    posiblesPuntosParado.add(posicionGPS)
  end for
  puntosParado  $\leftarrow$  potencialesPuntosParado
  puntosMovimiento  $\leftarrow$  potencialesPuntosMovimiento
   $\triangleright$  Etiquetar segmentos en función de si sus posiciones GPS corresponden con
  parado y movimiento
   $\triangleright$  Realizar las operaciones de decisión igual que en el algoritmo 5.2
end function

```

Por último, el algoritmo 5.4, que se corresponde con la clase `StopAndMoveNode`, es el más complejo de los cuatro algoritmos descritos. De manera resumida y conceptual, el primer paso consiste en inicializar la lista de posiciones GPS, así como las listas de *parados*, *parados potenciales*, *movimientos* y *movimientos potenciales*.

A continuación, se inicializa el centro del clúster con la primera posición de la lista de posiciones GPS. Posteriormente, comienza el proceso de análisis de cada una de las posiciones para determinar cuáles de ellas pertenecen a *parados* y cuales a *movimientos*. Esto se realiza de la siguiente manera, se recorren cada uno de los puntos GPS, comparando la distancia que existe entre cada punto GPS y el clúster, aquellos puntos que están a una distancia inferior al umbral establecido, se añaden a la lista de *parados potenciales*, por el contrario, aquellos puntos que superen ese umbral, se añaden a la lista de *movimientos potenciales*. Cada vez que se detecta el inicio de un desplazamiento, se pasan los puntos registrados hasta el momento como *potenciales parados* a la lista de *parados*, comprobando previamente que la duración de la parada supere el umbral establecido. Por el contrario, si la duración del intervalo formada por los puntos *parados potenciales* es demasiado corta, esa lista de puntos pasan a la lista de *movimientos*. Una vez recorridas todas las posiciones GPS, los últimos *parados potenciales* pasan *parados*, y los últimos *movimientos potenciales* a *movimientos*. A continuación, se analizan los parados y los movimientos para asignarles una localización según la ubicación GPS registrada, de manera que, a cada *parado* se le asigne la ubicación geográfica usando el tipo de datos *Point* de *PostGIS* y a cada grupo de puntos que formen un movimiento se les asigna la trayectoria recorrida con el tipo de datos *LineString*. Por último, se evalúa el predicado asociado a los arcos del nodo y se asigna el valor a los segmentos agrupados. Nótese que este último paso consta de las mismas operaciones realizadas en el algoritmo 5.2.

5.2. Lenguaje de especificación de patrones de actividad

Una vez presentada la estructura y funcionamiento de la taxonomía de actividades de nuestra propuesta, en este capítulo nos centraremos en describir el *lenguaje de especificación de patrones* empleado en la definición de las diferentes condiciones asociadas a los arcos de las taxonomías. Este lenguaje debe ser muy expresivo y que tenga en cuenta la información de las diversas fuentes de datos, de tal manera que, coteje no sólo los datos del sensor sin procesar sino también los datos almacenados en el sistema MWM, y de la información geográfica del contexto relacionada con el dominio.

Como se comentó en la sección anterior, cada arco de la taxonomía de actividades está asociado con un objeto de la clase *Predicate*, el cual representa el patrón de actividad que se anota en el arco. La expresividad de la taxonomía de actividades se corresponde con la cantidad de predicados que se permiten utilizar en los arcos. Por ello, el número de predicados que se necesitarán es más elevado que el número de nodos de procesamiento. El funcionamiento de cualquier clase que implemente la interfaz *Predicate* es sencillo: recibe un segmento para evaluar y debe devolver

cierto, falso o desconocido en función de si los valores del segmento cumplen el predicado.

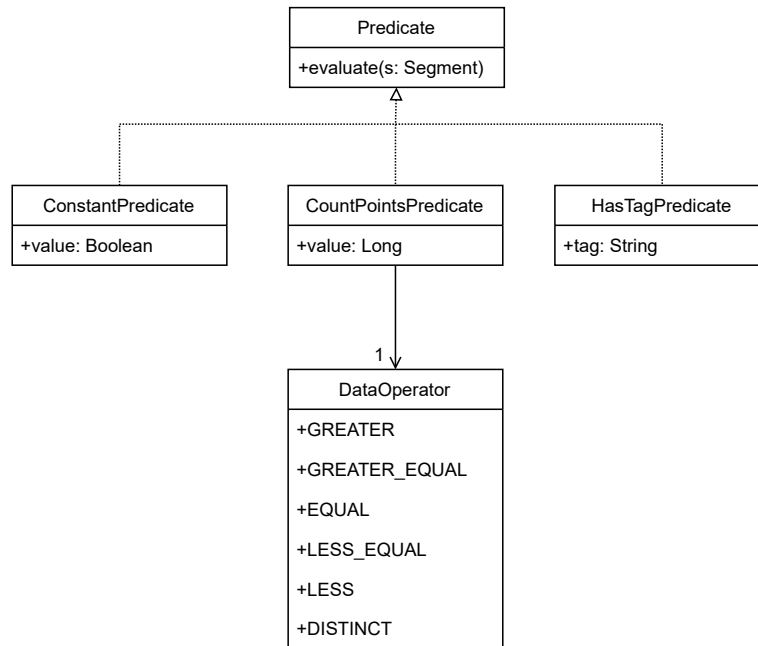


Figura 5.6: Predicados simples

A continuación vamos a explicar los diferentes tipos de predicados que se pueden definir. Comenzaremos con los tres predicados más simples, se muestran en la Figura 5.6.

- **Valor constante (ConstantPredicate).** Devuelve siempre el valor indicado en el atributo `value` (verdadero, falso o indefinido) independientemente de los valores del segmento. Es útil como predicado asociado a nodo de procesamiento tipo decisión. Por ejemplo, para implementar un arco de la taxonomía etiquetado como *en otro caso* se utilizaría el predicado de valor constante con el valor *true* para que todos los segmentos que lleguen a esa rama cumplan el predicado y sean procesados.
- **Contar puntos (CountPointsPredicate).** Este predicado cuenta el número de ubicaciones GPS asociadas al segmento y comprueba si cumple la condición indicada por el `DataOperator` asociado al predicado y el valor indicado en `value`. Por ejemplo, este predicado se puede utilizar para determinar si en un segmento tiene ubicaciones GPS asociadas instanciando el predicado con el

operador `GREATER` y el valor 0. En este caso, devolvería cierto si el número de puntos GPS supera el umbral indicado.

- Contiene etiqueta (HasTagPredicate).** Este predicado devuelve cierto si la etiqueta indicada en el atributo `tag` se encuentra entre las que están asociadas al segmento. Este predicado es útil para discriminar entre los segmentos etiquetados por nodos de procesamiento como el de *Agrupar* o el de *Parada Y Movimiento*.

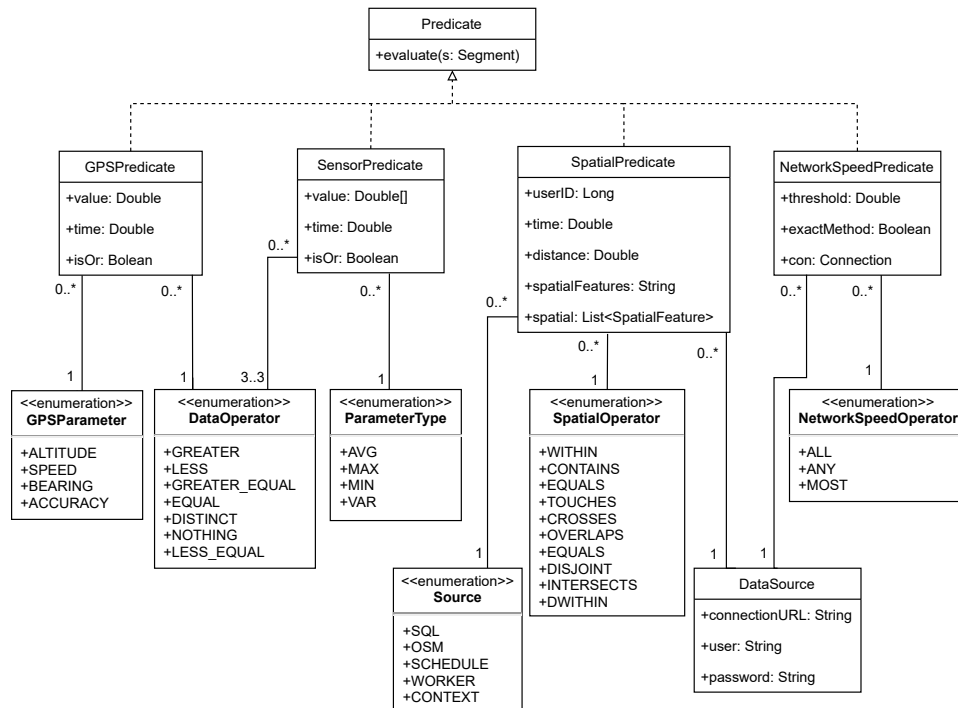


Figura 5.7: Predicados complejos

La Figura 5.7 muestra cuatro predicados que realizan comparaciones más complejas:

- Valores GPS (GpsPredicate).** Devuelve cierto si alguno de los valores de la propiedad indicada en el atributo `type` de las ubicaciones GPS asociadas al segmento satisfacen la condición indicada en el operador `DataOperator` asociado y el valor indicado en `value`. Por ejemplo, este predicado se puede utilizar para determinar si la velocidad en un segmento es mayor que 10 km/h.

- **Valores de sensor (SensorPredicate).** Devuelve cierto si los valores de la propiedad indicada en el atributo `type` de los datos de sensores asociados al segmento satisfacen la condición indicada con el operador de comparación `DataOperator` asociado y el valor indicado en `value`. El atributo `isOr` se puede utilizar para indicar si todas las dimensiones del valor del sensor deben cumplir la condición (`isOr = false`) o si basta con que alguna cumpla la condición (`isOr = true`).
- **Comparación espacial (SpatialPredicate).** Evalúa si alguna de las ubicaciones GPS asociadas al segmento satisface la relación espacial indicada mediante un `SpatialOperator` respecto a la información geográfica del contexto. La información geográfica de contexto se indica mediante su nombre en el atributo `spatialFeatureCollection` tal y como se describió en la Sección 4.4. En el caso del operador `DWITHIN`, que comprueba que la distancia sea menor que un umbral, se indica el umbral mediante el atributo `value`. Por ejemplo, este predicado se puede utilizar para comprobar si la ubicación registrada por el GPS coincide con la ubicación de la empresa del cliente donde el trabajador debe acudir a realizar una inspección.
- **Velocidad en red de carreteras (NetworkSpeedPredicate).** Devuelve cierto si la velocidad de las ubicaciones GPS del segmento supera de un umbral indicado en el atributo `threshold` a la velocidad máxima del tramo correspondiente de la red de carreteras. La velocidad máxima para cada tramo de red de carreteras se extrae de *OpenStreetMap* (directamente si está presente en el tramo, o indirectamente por el tipo de carretera si no está presente).

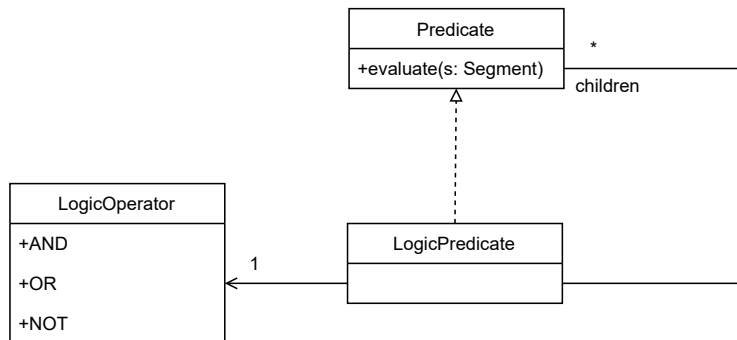


Figura 5.8: Predicado de combinación lógica para formar predicados complejos

Finalmente, a Figura 5.8 muestra un predicado que permite construir expresiones complejas mediante operadores lógicos. Para ello, se indican los predicados a

combinar mediante una colección de predicados hijo, y se indica el operador lógico con el que se quieren combinar (*and*, *or* o *not*).

Por lo tanto, concluimos que esta forma de modelar los predicados nos ha permitido obtener un lenguaje de especificación de patrones, muy expresivo y versátil, permitiendo definir taxonomías complejas y con la capacidad de detectar actividades con un alto nivel de abstracción semántica.

5.3. Anotación de actividades

Ahora que se ha explicado la arquitectura del componente de etiquetado y la definición completa de una taxonomía, vamos a explicar el proceso completo para etiquetar las actividades de los empleados de una empresa cualquiera.

El primer paso es obtener todos los trabajadores de la empresa. Entonces, para cada trabajador, se recupera la taxonomía de actividad correspondiente al tipo de trabajador. Después de eso, los datos sin procesar se recuperan para el trabajador. Los datos del sensor consisten en una colección de objetos **SensorData**, en los que cada uno de ellos representa el valor de la media, la varianza, además del valor mínimo y máximo de los sensores registrados en ese un instante de tiempo. Del mismo modo, los datos de la ubicación consisten en una colección de objetos GPS, cada uno de ellos presenta una marca de tiempo y las variables GPS.

Dado que los datos pueden no estar alineados en el sentido de que los tiempos de inicio y fin de los intervalos no son necesariamente los mismos para todos los tipos de sensores y las marcas de tiempo de ubicación no coinciden con ningún intervalo de tiempo, es necesario un paso en el algoritmo que consiga homogeneizar los intervalos de tiempo. El resultado es una colección de objetos **Segment**, cada uno de ellos consiste en un intervalo de tiempo, la varianza, media, el valor máximo, mínimo de los sensores y un conjunto de ubicaciones GPS. Este proceso se describió en la Sección 4.4.

En el paso siguiente, la taxonomía de la actividad procesa y etiqueta cada uno de los segmentos recibidos. El proceso comienza en la raíz de la taxonomía. Dependiendo del tipo de nodo de procesamiento el segmento se procesará con un algoritmo u otro. Por ejemplo, el nodo de procesamiento de tipo *ParadaYMovimiento* aplicará el correspondiente algoritmo y etiquetará los segmentos con las etiquetas correspondientes a parado y movimiento. Una vez procesados todos los segmentos, se evaluará el predicado asociado a cada una de sus aristas para cada segmento de manera que, si la evaluación del predicado devuelve cierto, se pasará el segmento a ese nodo hijo. Todos los segmentos que no se evalúan como verdaderos se evalúan en la siguiente arista de la taxonomía. Los segmentos que permanecen después de la evaluación de todas las aristas de la taxonomía son etiquetados con la etiqueta *No identificada*.

De esta forma, los segmentos van descendiendo por la taxonomía desde la raíz hasta los nodos hoja. En los nodos hoja se espera que se encuentre un nodo de

procesamiento de tipo etiquetado (**TagNode**) que asocie al segmento una etiqueta correspondiente a una de las actividades que puede realizar un trabajador (por ejemplo, parado, activo, conduciendo despacio, caminando, conduciendo rápido).

Al terminar el procesamiento de todos los segmentos por la taxonomía de actividades, deben convertirse los segmentos en actividades. Para ello, se recorre la colección completa de segmentos teniendo en cuenta solo aquellas etiquetas que se corresponden a una de las actividades definidas para la empresa. Todos los segmentos contiguos en el tiempo se agregan en un único segmento, y aquellos segmentos con la etiqueta *No identificada* serán agregados a la actividad anterior detectada.

El resultado final del proceso es una lista de segmentos agrupados en los grupos de máxima extensión posible que sólo conservan las etiquetas de las actividades. El último paso del proceso consiste en convertir la lista de segmentos etiquetada en actividades que puedan ser almacenadas en la base de datos de trayectorias semánticas. Cada actividad almacena el nombre de la actividad, el instante de inicio y de fin de la actividad, el orden relativo en el que se realiza la actividad con respecto a las demás actividades del mismo día, el usuario al que corresponde la actividad, y la trayectoria seguida por el usuario usando un tipo de datos geográfico *Geometry* que podrá ser un *MultiPoint* (en caso de que se detecte que hay movimiento) o un *LineString* (en caso de que se detecte que hay movimiento). El algoritmo 5.5 muestra el pseudocódigo del algoritmo de etiquetado semántico.

Algoritmo 5.5 Algoritmo Annotator

function ANOTAR(posiciones: Lista de gps, datosSensores: Lista de datos sensor, tareas: Lista de tareas, localizaciones: Map, idTrabajador: Long, nodoRaiz: Nodo Taxonomia)

▷ *Paso 1: Crear segmentos unificados de tamaño fijo*

segmentos ← fusionar(posiciones, sensores)

▷ *Paso 2: Anotar segmentos - véase algoritmo 3*

nodoRaiz.anotar(segmentos)

▷ *Paso 3: Agrupar segmentos etiquetados*

segmentosAgrupados ← agrupar(segmentos)

▷ *Paso 4: Convertir a actividades*

actividades ← convertirAactividades(segmentosAgrupados, idTrabajador)

end function

5.4. Conclusiones

En conclusión, en este capítulo hemos descrito un método que permite detectar las actividades de los trabajadores en movilidad. Esta metodología ha dado solución a los tres problemas planeados inicialmente. En primer lugar, hemos definido un

modelo que permite representar de una manera sencilla e intuitiva las actividades que deben ser identificadas. En segundo lugar, hemos descrito los elementos necesarios para construir las taxonomías de actividades, empleando un lenguaje de especificación de patrones muy expresivo y versátil para expresar patrones de actividades y que coteja la información procedente de diversas fuentes de información, los datos de los sensores sin procesar, los datos almacenados en el sistema MWM y de la información geográfica del contexto relacionada con el dominio. Además, el concepto de actividades definido es más poderoso que los árboles de decisión, permitiendo además de tomar decisiones, procesar datos en cada uno de sus nodos, pero no solo eso, este concepto de taxonomías de actividades permite expresar la toma de decisiones de manera más intuitiva y coloquial que definiendo las reglas empleando consultas SQL. Por último, hemos detallado tanto los algoritmos necesarios para la definición de las taxonomías como los algoritmos necesarios para la detección de las actividades semánticas.

Capítulo 6

Evaluación del sistema

En esta sección presentamos la evaluación del sistema de identificación de actividades semánticas. En primer lugar, en la Sección 6.1, se realizan dos evaluaciones experimentales de la metodología utilizando datos recogidos por voluntarios de nuestro grupo de investigación. En segundo lugar, en la Sección 6.2, realizamos dos pruebas de concepto aplicando la metodología a dos empresas e identificando actividades de interés para su día a día.

6.1. Evaluación experimental

En esta sección presentamos la evaluación experimental que hemos realizado de la metodología de identificación de actividades en un entorno controlado utilizando datos recogidos por voluntarios de nuestro grupo de investigación. En la Sección 6.1.1 describimos el conjunto de datos de prueba, en las secciones 6.1.2 y 6.1.3 exponemos las taxonomías definidas y los resultados obtenidos en los experimentos con *ParadoYMovimiento* y *Decisión* respectivamente.

6.1.1. Conjuntos de datos

Para la realización de los experimentos descritos se han implementado y se han utilizado las aplicaciones y componentes descritos en el Capítulo 4. Sin embargo, las pruebas descritas en las Secciones 6.1.2 y 6.1.3 han necesitado a mayores un conjunto de datos previo para poder realizarse. La recogida de estos datos abarcó un período desde octubre del 2019 hasta diciembre del 2019. Los datos fueron recogidos por 19 personas que trabajaban en nuestro grupo de investigación a la que se le proporcionó una aplicación *Android* construida a partir del componente de captura de datos (*SensorCollector*, ver Sección 4.2) a la que se le añadió funcionalidad para registrar manualmente qué actividad estaba realizando en cada momento. Cada

usuario iniciaba el registro de datos de los sensores y puntos GPS indicando a través de la aplicación *Android* la actividad que iban a realizar en cada momento, y una vez finalizada la actividad el usuario también registraba el final de dicha actividad a través de la aplicación *Android*. Esto nos permite evaluar la calidad de las actividades detectadas por nuestra taxonomía de actividades. Las actividades que se consideraron como objetivo para identificar en el experimento fueron las que siguen:

- **Conduciendo.** Desplazarse en un vehículo.
- **Andando.** Desplazarse caminando a otra ubicación.
- **Activo.** La persona que porta el móvil se está moviendo, pero no necesariamente es caminando y además no existe un desplazamiento a otra ubicación. Por ejemplo, está hablando por teléfono.
- **Inactivo.** La persona permanece inmóvil o ha dejado el dispositivo móvil apoyado en alguna superficie estática, por ejemplo, una mesa.

Los datos generados por los dispositivos móviles fueron recogidos por un componente de recogida de datos (*SensorReceiver*, ver Sección 4.3) implementado en Java y desplegado en un servidor del grupo de investigación. La base de datos *Raw data* se implementó en *PostgreSQL* y la Tabla 6.1 muestra, para cada tipo de actividad, el tiempo total de duración de la actividad, el número de sesiones de recogida de datos, el número de datos recogidos del sensor de aceleración lineal, el número de ubicaciones GPS y el porcentaje que el tipo de actividad representa sobre el total.

Actividad	Tiempo	Actividades	Sensores	GPS	Porcentaje
Conduciendo	42h15'07"	159	14.570.261	12.598	20,92 %
Andando	27h34'34"	217	4.854.293	5.399	13,65 %
Activo	50h13'03"	108	9.206.442	2.132	24,86 %
Inactivo	81h54'59"	163	7.395.359	2.486	40,56 %
Total	201h57'44"	647	36.026.355	17.446	100,00 %

Tabla 6.1: Descripción de los datos recogidos en el experimento

En la Tabla 6.2 se muestra una matriz de confusión con los resultados de la ejecución del proceso de identificación de actividades del trabajo de investigación [GGRFBL20]. Las columnas representan las actividades detectadas por el sistema y las filas las actividades reales que indicaron los usuarios que estaban haciendo. En el cruce se encuentra el número de datos que identificó el sistema con la actividad de la columna y que realmente se correspondían con la actividad de la fila. Por

Actividad real	Actividad detectada				Recuperación
	Conduciendo	Andando	Activo	Inactivo	
Conduciendo	3.453	1.015	948	259	60,85 %
Andando	1.714	5.675	3.772	1.341	45,39 %
Activo	9.14	1.163	8.676	3.026	62,97 %
Inactivo	90	486	610	16.787	93,40 %
Precisión	55,96 %	68,05 %	61,94 %	78,40 %	

Tabla 6.2: Resultados de la identificación preliminar de actividades realizadas en el trabajo de investigación [GGRFBL20]

ejemplo, el sistema identificó 16.787 datos con la actividad *Inactivo* que realmente correspondían a la actividad *Inactivo* (esto es, aciertos), pero identificó 3.026 datos con la actividad *Activo* que realmente correspondían a la actividad *Inactivo* (esto es, fallos). Utilizaremos estos resultados como punto de partida para comparar los resultados de nuestros algoritmos.

6.1.2. Experimento con *Parado Y Movimiento*



Figura 6.1: Taxonomía de actividades para el experimento con *Parado Y Movimiento*

En el primer experimento realizado definimos la taxonomía de la Figura 6.1.

Como se puede observar el nodo raíz es de tipo *ParadaYMovimiento*, de esta manera somos capaces de discernir en qué intervalos de tiempo el usuario está moviéndose y en cuáles se encuentra parado. Para discernir los segmentos en los que el usuario está moviéndose o parado utilizamos un predicado *HasTagPredicate*, devuelve cierto si la etiqueta *moviéndose/parado* indicada en el atributo `tag` se encuentra entre las que están asociadas al segmento. Para los segmentos que cumplen esta condición utilizamos otro nodo de tipo *Decisión* para determinar si está conduciendo o andando. Consideramos que está conduciendo si la velocidad es mayor que 3 m/s (esto es, 10 km/h) durante al menos la mitad del segmento y consideramos que está andando en otro caso (de nuevo, utilizamos un predicado de tipo *GPSPredicate*). Volviendo a la raíz de la taxonomía, en caso de que el usuario no esté moviéndose (esto es, los segmentos del arco derecho de la raíz), utilizamos un nodo de tipo *Decisión* para determinar si el usuario está activo o inactivo. Consideramos que está activo si el valor del sensor de aceleración lineal es menor que 0.2 m/s^2 en todos los ejes durante al menos el 95% del tiempo del segmento. Para evaluar esta condición utilizamos un predicado de tipo *SensorPredicate*.

Actividad real	Actividad detectada				Recuperación
	Conduciendo	Andando	Activo	Inactivo	
<i>Conduciendo</i>	1952	212	332	6	78.02 %
<i>Andando</i>	5	831	556	151	53.86 %
<i>Activo</i>	1	520	1762	220	70.40 %
<i>Inactivo</i>	0	533	300	3757	81.85 %
Precisión	99,69 %	39,65 %	59,73 %	90,88 %	

Tabla 6.3: Resultados del experimento con *ParadoYMovimiento*

La Tabla 6.3 muestra una matriz de confusión con los resultados de la ejecución del proceso de identificación de actividades en el experimento con *ParadoYMovimiento*. Las columnas representan las actividades detectadas por el proceso y las filas las actividades reales que indicaron los usuarios que estaban haciendo. En el cruce se encuentra el número de segmentos que nuestra taxonomía identificó con la actividad de la columna y que realmente se correspondían con la actividad de la fila. Por ejemplo, nuestra taxonomía identificó 1.952 segmentos con la actividad *Conduciendo* que realmente correspondían a la actividad *Conduciendo* (esto es, aciertos), pero identificó 5 segmentos con la actividad *Conduciendo* que realmente correspondían a la actividad *Andando* (esto es, fallos).

A la derecha de la tabla hemos calculado el factor de recuperación (*recall*), esto es, el número de segmentos de la actividad real que nuestra taxonomía ha conseguido identificar sobre el total de segmentos de esa actividad. Por ejemplo, nuestra

taxonomía ha identificado correctamente el 78,02 % de los segmentos de la actividad *Conduciendo*. En la última fila hemos calculado el factor de precisión (*precision*), esto es, el número de los segmentos que nuestra taxonomía identifica correctamente de cada actividad concreta sobre el número de segmentos que identifica como esa actividad. Por ejemplo, nuestra taxonomía identifica correctamente el 99,69 % de los segmentos que identifica como *Conduciendo*.

Los resultados de la Tabla 6.3 no fueron los esperados, por ejemplo, el sistema confunde un total de 66 segmentos con activo cuando realmente el empleado estaba conduciendo o andando. Además comparando los datos con otros trabajos de investigación como por ejemplo [GGRFBL20], concluimos que teníamos que hacer modificaciones en nuestra taxonomía para mejorar los resultados obtenidos. Por ejemplo, en [GGRFBL20] la precisión en la identificación de la actividad activo es de un 62.97 %, sin embargo, la nuestra es del 59,73 %. Tras la comparación, otro resultado que queríamos mejorar era el porcentaje de la identificación de la actividad *Andando*, donde en [GGRFBL20] es de un 45.39 % y el nuestro apenas llegaba al 39,65 %. Otro dato que también teníamos que mejorar era nuestro porcentaje en la identificación de la actividad *Inactivo*.

Tras un exhaustivo estudio del conjunto de datos recogido y los correspondientes resultados, concluimos que una de las causas de estos malos resultados era que el GPS es muy impreciso en el interior de los edificios, lo que provocaba que la precisión de las ubicaciones detectadas en estas ocasiones eran muy inexactas. El sistema identificaba dos puntos GPS consecutivos muy distantes (más de 100 metros de distancia) cuando en realidad el empleado permanecía en la misma localización (dentro del edificio). Para solucionar este problema, el umbral de distancia que utilizamos para distinguir parado-moviéndose tiene que ser menos estricto (es decir, considerar que alguien está parado aunque la distancia entre puntos GPS sea muy grande). Pero entonces aumentan los problemas de identificación en el caso de moviéndose porque comenzamos a considerar como parados casos en los que realmente la persona se movía. Por tanto, utilizando el nodo *ParadoYMovimiento* nos encontrábamos en un punto muerto y por ello, decidimos hacer un segundo experimento en el que en lugar de emplear un nodo *ParadoYMovimiento* en la raíz de la taxonomía, usamos un nodo *Decisión*. En la siguiente sección 6.1.3 describimos en detalle este segundo experimento y los resultados obtenidos.

6.1.3. Experimento con *Decisión*

En el segundo experimento hemos definido la taxonomía de la Figura 6.2. Como se puede observar el nodo raíz es de tipo *Decisión*, de esta manera somos capaces de discernir en qué intervalos de tiempo el usuario está moviéndose y en cuáles se encuentra parado. El nodo de tipo *Decisión* simplemente distribuye los segmentos que ha recibido como entrada a sus nodos hijo dependiendo del resultado de la evaluación de los predicados asociados a los arcos que lo unen con los nodos hijo, de manera que, se considera que el usuario está moviéndose en aquellos segmentos

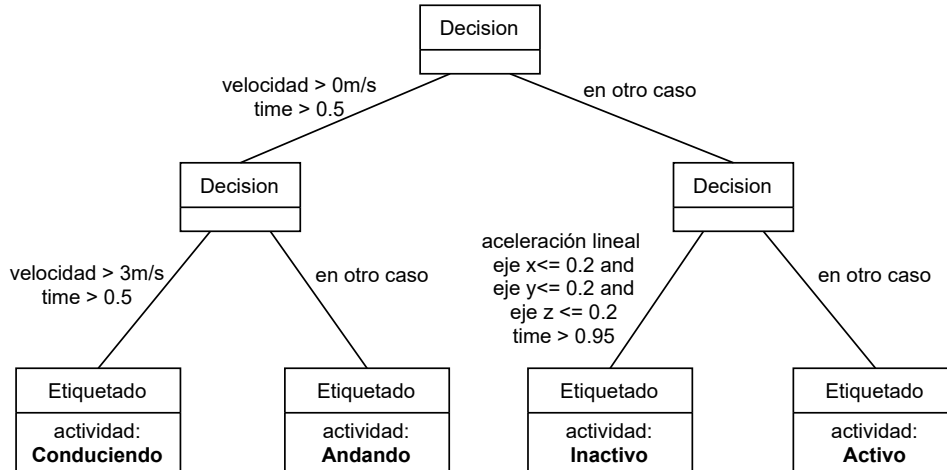


Figura 6.2: Taxonomía de actividades para el experimento con *Decisión*

donde la velocidad del segmento es mayor que 0 m/s durante al menos la mitad del segmento, por el contrario, aquellos segmentos que no cumplan este predicado, se considera que el usuario se encuentra parado. Una vez hecho esto, los segmentos en los que el empleado está moviéndose pasan a otro nodo hijo de tipo *Decisión* para determinar si está conduciendo o andando. El segmento se etiqueta con la actividad conduciendo si la velocidad del segmento es mayor que 3 m/s (esto es, 10 km/h) durante al menos la mitad del segmento y consideramos que está andando en otro caso (de nuevo, utilizamos un predicado de tipo *GPSPredicate*). Volviendo a la raíz de la taxonomía, en caso de que el usuario no esté moviéndose (esto es, los segmentos del arco derecho de la raíz), utilizamos un nodo de tipo *Decisión* para determinar si el usuario está activo o inactivo. Consideramos que está activo si el valor del sensor de aceleración lineal es menor que 0.2 m/s^2 en todos los ejes durante al menos el 95% del tiempo del segmento. Para evaluar esta condición utilizamos un predicado de tipo *SensorPredicate*.

La Tabla 6.4 muestra una matriz de confusión con los resultados de la ejecución del proceso de identificación de actividades. La representación de la matriz es análoga al caso anterior.

De los datos de la tabla se puede calcular que nuestra taxonomía de actividad tiene una exactitud (*accuracy*) del 84,26% calculado como el número total de segmentos identificados correctamente dividido por el número total de segmentos). Parte de los errores detectados se deben a la utilización de un tamaño de segmento de un minuto, lo que provoca que se produzcan errores si el inicio o el fin de una actividad real no coincide con el inicio o el fin de los segmentos.

El sistema presenta un porcentaje de precisión del 98,95% y un porcentaje de

Actividad real	Actividad detectada				Recuperación
	Conduciendo	Andando	Activo	Inactivo	
Conduciendo	2.077	174	245	6	83,01 %
Andando	7	1.009	381	151	65.19 %
Activo	4	62	2.117	320	84.58 %
Inactivo	11	119	275	4.193	91.19 %
Precisión	98,95 %	73,97 %	70,15 %	89,79 %	

Tabla 6.4: Resultados del experimento con *Decisión*

recuperación del 83,01 % detectando la actividad *Conduciendo*. Falla en aquellos casos en los que la velocidad es baja, y por lo tanto, la confunde con otras actividades como *Andando* (un 12,76 % de las veces). El sistema también falla en la detección de *Conduciendo* y lo confunde con *Inactivo* o *Activo* en un porcentaje total de 8,12 % y de 0,13 % respectivamente. Este caso ocurre cuando el usuario se encuentra en un atasco, por lo tanto, la velocidad es muy baja y el valor medio de los sensores no lo es, por lo que, el sistema lo etiqueta como *Activo*.

Nuestro sistema presenta un porcentaje de precisión del 73.97 % y un porcentaje de recuperación del 75,18 % en la detección de la actividad *Andando*. Esta es la actividad que más problemas presenta. El sistema confunde la actividad *Andando* con *Activo* un 12,62 % de las veces y con *Inactivo* un 3,23 % de las veces. Esto puede ocurrir por varias causas. Por ejemplo, cuando el usuario se entretiene hablando con un conocido, los valores de los sensores pueden ser altos porque se está moviendo y gesticulando, sin embargo, la velocidad es nula porque no existe un desplazamiento. También puede ocurrir cuando el usuario se encuentra esperando para cruzar un paso de cebra, esto es debido a que no existe un desplazamiento y el usuario permanece inmóvil, por lo que, la velocidad es nula. En estos casos, nuestro sistema detecta *Activo* o *Inactivo* en función del movimiento del dispositivo móvil.

En el caso de la detección de la actividad *Activo*, el sistema presenta un porcentaje de precisión del 70,15 % y un porcentaje de recuperación del 84,58 %. El sistema confunde *Activo* con *Andando* en un porcentaje de 4,55 %. Esto puede deberse a que algunos segmentos de la actividad *Activo* están formados por puntos GPS recogidos en el interior de instalaciones y/o edificios, por lo tanto, los datos son imprecisos debido a la inexactitud del GPS y en ocasiones se detecta movimiento (y por tanto velocidad) cuando realmente no ha ocurrido. Por otra parte, el sistema falla en un porcentaje total de 6,85 % en la detección de *Activo* y lo confunde con *Inactivo*. Esto puede deberse a que el umbral para determinar si un dispositivo está *Activo* o *Inactivo* se ha decidido de forma empírica sin realizar un estudio detallado de la actividad de los usuarios.

Por último, el sistema presenta un porcentaje precisión del 89,79% y un porcentaje de recuperación del 91,19% en la detección de la actividad *Inactivo*. El sistema confunde esta actividad con otras en aquellos segmentos que están formados por puntos GPS recogidos en interiores de edificios. Como se explicaba más arriba, la precisión del GPS es muy baja en estas ocasiones y por lo tanto, la velocidad detectada es errónea en la mayor parte de estos puntos GPS, ocasionando errores en la detección de actividades.

En conclusión, como se puede observar tras el análisis de los resultados obtenidos, con esta pequeña modificación en el segundo experimento, hemos conseguido mejorar los datos del anterior experimento descrito en la Sección 6.1.2. Además también se han mejorado los porcentajes en la identificación de todas las actividades en comparación con el trabajo de investigación [GGRFBL20]. Si comparamos las tablas 6.2 y 6.4, podemos comprobar que los porcentajes de precisión han sido mejorados y que las taxonomías tienen mejores resultados en la detección de la mayoría de las actividades, presentando una diferencia entre los resultados obtenidos en la detección de la actividad *Conduciendo* de un 38.37%, para la actividad *Andando* de un 28.58%, en el caso de la actividad *Activo* de un 7.18%. Sin embargo, cabe señalar, por último, que el porcentaje de precisión en la detección de la actividad *Inactivo*, presenta una diferencia entre ambos resultados de un 3.61%, siendo mejor la del trabajo [GGRFBL20]. Por otra parte, podemos concluir que tras la comparación de los porcentajes de recuperación, los resultados obtenidos en estos porcentajes son mejores los de la Tabla 6.4 para la mayoría de las actividades, presentando una diferencia entre los resultados para las actividades *Conduciendo*, *Activo* e *Inactivo*, del 27.05%, 22.63%, 12.79% respectivamente. Sin embargo, cabe destacar, que en el caso de la detección de la actividad *Caminando*, la diferencia de ambos porcentajes es de un 2.86%, siendo mejor el resultado obtenido en el trabajo [GGRFBL20].

6.1.4. Conclusiones

Con estos experimentos hemos demostrado que el concepto de *taxonomía de actividades* se puede utilizar para identificar actividades semánticas. Se puede ver que la definición de una taxonomía es una tarea sencilla y que se puede realizar de forma mucho más intuitiva que si hubiera que hacerlo mediante reglas expresadas en un lenguaje de consulta (por ejemplo, SQL). Además, no es necesario realizar un entrenamiento del sistema, por lo que no es necesario recoger un conjunto de datos de prueba alterando la actividad diaria de la empresa. En estas pruebas concretas no hemos alcanzado una identificación perfecta de las actividades *Andando* y *Activo*. Sin embargo, creemos que es posible mejorarlo con una taxonomía de actividades más compleja en la que se combine en los primeros niveles un nodo de *ParadoYMovimiento* junto con varios nodos de *Decisión* utilizando predicados que evalúen los valores del GPS (esto es, *GPSPredicate*) y de los sensores del móvil (esto es, *SensorPredicate*). De forma intuitiva, siendo muy estrictos con el umbral

de distancia en el nodo *ParadoYMovimiento* de la raíz identificaremos claramente todos los movimiento. Sin embargo, tendremos muchos errores en los casos en los que se está andando despacio y la distancia recorrida entre puntos GPS consecutivos es pequeña. Utilizando un nodo *Decisión* en este caso para discriminar entre los casos en los que realmente hay movimiento (porque el punto GPS indica velocidad o porque la actividad de los sensores es elevada) podríamos distinguir entre casos de la actividad *Andando* y las actividades *Activo/Inactivo*.

6.2. Prueba de concepto

En esta sección presentamos una prueba de concepto realizada sobre el sistema dentro del marco del proyecto GEMA del Laboratorio de Bases de Datos. En la Sección 6.2.1 se describe la taxonomía definida y resultados obtenidos para la empresa GESUGA y en la Sección 6.2.2 para la empresa MAYORES.

El proyecto GEMA tenía como objetivo principal el desarrollo de tecnologías software para la gestión de trabajos en movilidad, es decir, la gestión de trabajos que los empleados realizan desplazándose a diferentes destinos marcados por una agenda que organiza su jornada. El proyecto GEMA abordaba este objetivo principal desde cuatro ámbitos de actuación:

- El desarrollo de algoritmos y tecnologías de planificación inteligente.
- El análisis de la actividad realizada por las personas y el etiquetado automático de actividades y trayectorias.
- El desarrollo de tecnologías para la representación y almacenamiento eficiente de trayectorias.
- La aplicación de tecnologías de desarrollo automatizado de software que permitan generar aplicaciones de gestión de la movilidad.

El proyecto, planteado y ejecutado por cuatro empresas de distintos sectores de actividad (GESUGA, Mayores, TAPREGA y Enxenio), y por el centro tecnológico CITIC (Centro de Investigación y Desarrollo en Tecnologías de la Información) de la Universidade da Coruña, tenía como objetivo realizar aportaciones y desarrollar tecnología que avanzase el estado del arte actual en la gestión de trabajos en movilidad. En concreto, el proyecto planteaba realizar aportaciones en los siguientes objetivos de investigación:

- **Objetivo científico 1:** Planificación inteligente
- **Objetivo científico 2:** Etiquetado semántico de trayectorias
- **Objetivo científico 3:** Representación y almacenamiento eficiente de trayectorias

- **Objetivo científico 4:** Desarrollo automatizado del software

Los tres primeros objetivos científicos se centraban en el desarrollo de tecnologías que necesitarían una aplicación avanzada de Gestión de Trabajadores Móviles (*Mobile Workforce Management*) que permitiesen la planificación, seguimiento y el almacenamiento y explotación de los trabajos diarios de cada trabajador. La investigación llevada a cabo para alcanzar los tres primeros objetivos científicos se planteó en el marco de tres objetivos tecnológicos planteados por tres de las empresas del consorcio:

- **Objetivo tecnológico 1:** Planificación de rutas y seguimiento de actividades de conducción en carretera (GESUGA).
- **Objetivo tecnológico 2:** Planificación de agendas y seguimiento de actividades de conducción en ciudad (TAPREGA).
- **Objetivo tecnológico 3:** Planificación de horarios y gestión de actividades en interiores (Mayores).

Nuestro sistema ha formado parte del resultado de investigación del objetivo tecnológico 2. A continuación vamos a presentar cómo se utilizó nuestro sistema en dos empresas del consorcio, GESUGA y Mayores. En las siguientes secciones se realizará una breve descripción de estas dos empresas, a continuación se detallarán las actividades identificadas, el proceso de identificación de las mismas y los resultados obtenidos.

6.2.1. Identificación de actividades en GESUGA

GESUGA, Gestora de Subproductos de la comunidad autónoma de Galicia, es una empresa fundada en el año 2005 que centra su área de negocio en la gestión integral de los subproductos cárnicos no destinados al consumo humano. Esta empresa tiene su base y oficinas Centrales en Cerceda (A Coruña) pero cuenta, además, con otras tres plantas en Galicia lo que le permite gestionar eficientemente su actividad a lo largo de toda la Comunidad.

La actividad principal de GESUGA consiste, mayoritariamente, en la recogida y en el transporte de los distintos subproductos o residuos desde las explotaciones o puntos de recogida hasta un punto de tratamiento, para su valoración y posterior destrucción o comercialización según el posible destino de cada producto. Para hacer frente a esta actividad, GESUGA dispone de un total de 3 plantas y una flota de 36 vehículos que gestionan y realizan alrededor de unas 500 recogidas por día, con el aliciente de que entre un aviso y su recogida no debe excederse la horquilla crítica de tiempo de recogida límite establecido para cada especie.

Diariamente, GESUGA recibe todos los avisos de animales muertos que se han producido en toda Galicia. Desde el departamento de rutas se gestionan estos avisos,

principalmente de forma manual, para que cada uno quede asignado a alguna de las rutas fijas que GESUGA tiene preestablecidas. Al día siguiente, si surge alguna incidencia o surgen nuevos avisos también deben ser gestionados sobre la marcha intentando asignarlos a alguna de las rutas más próximas y que vayan a pasar cerca del nuevo punto de recogida, ya que si el nuevo aviso surge de una explotación por la que el conductor ya ha pasado seguramente no es rentable que vuelva hacia atrás.

En ese momento, al inicio de cada jornada, cada operador tiene asignada una ruta, con las diferentes recogidas a realizar con uno de los camiones. Además, todo operador debe seguir un protocolo de actuación y revisar que antes de empezar la ruta lleve consigo un *smartphone* ya cargado con los detalles de la ruta y las recogidas de la jornada. La aplicación que tenían permitía realizar un registro periódico de la posición del trabajador. Sin embargo, presentaba ciertas carencias al no disponer de un sistema central que permitiese explotar los datos, visualizar la posición de cada vehículo en tiempo real, conocer retrasos en las rutas, visualizar el estado general sobre un mapa, etc.

Como se puede comprobar, GESUGA necesitaba una solución MWM que le permitiese gestionar de una forma automática y más eficiente las recogidas a realizar diariamente, de modo que así se redujese el consumo de combustible de sus camiones al mismo tiempo que acortase el tiempo de duración de las rutas, lo que permitía que se pudiese atender mejor la demanda y realizar un mayor número de recogidas por día. También, se buscaba ahorrar costes de gestión al evitar tener que planificar manualmente en qué ruta se incluía cada uno de los avisos, desarrollando un sistema automatizado que incluyese algoritmos de Investigación Operativa para ofrecer la solución óptima a cada problema de organización de rutas. Además, las necesidades de GESUGA implicaban que el sistema pudiese replanificar durante el día teniendo en cuenta los nuevos avisos y la posición actual de cada vehículo.

GESUGA ya disponía de una aplicación móvil básica de apoyo al trabajo que debían realizar sus camioneros en cada explotación ganadera o matadero (para cubrir datos asociados a recogidas, generar justificantes de entrega, recogida de firmas, etc.). Por ello, en esta empresa se integró en su aplicación móvil nuestro componente de captura de datos (*SensorCollector*), y se integraron el componente *SensorReceiver* y el componente de segmentación y etiquetado de trayectorias para dar cobertura a las tareas de detección e identificación automática de actividades (*Annotator*).

En lo relativo al modelado y caracterización de las actividades, invertimos un tiempo importante en tratar de identificar y consensuar el conjunto de actividades consideradas de relevancia para la empresa, pues de ello dependería después también la relevancia de los estudios de análisis y explotación posteriores. Concretamente, se definieron las siguientes:

- **CF-RP: Conducción fluida en ruta planificada.** El empleado conduce un camión desde una de las plantas de GESUGA o puntos de recogida a otro. Además, su velocidad se adecúa a la recomendada por las normas viales y su

ubicación no difiere de la ruta planificada.

- **CL-RP: Conducción lenta en ruta planificada.** El empleado conduce un camión desde una de sus plantas de GESUGA o puntos de recogida a otro y su ubicación no difiere de la ruta planificada. Sin embargo, circula a una velocidad inferior a la aconsejada por las señales viales.
- **CF-FR: Conducción fluida fuera de ruta planificada.** El empleado circula a una velocidad adecuada por la red vial. Sin embargo, el trayecto seguido difiere demasiado de la ruta planificada.
- **CL-FR: Conducción lenta fuera de ruta planificada.** El empleado circula por la red vial a una velocidad inferior a la aconsejada por las normas viales y además su ubicación difiere demasiado de la ruta planificada.
- **DES: Descanso en la conducción.** El conductor realiza una parada en el trayecto de camino al punto de recogida (cliente) o hacia la factoría. Por ley, estos descansos deben realizarse cada cierto tiempo y deben tener una duración mínima de 15 minutos.
- **PAR: Parada corta.** Los empleados pueden realizar sus descansos en la conducción en cualquier lugar, ya sea en la ubicación de un cliente, en la factoría, o en una ruta (planificada o no). Cada uno de estos descansos debe durar menos de 15 minutos. Por lo tanto, cualquier parada menor de 15 minutos se etiqueta como parada corta.
- **TR-CL: Trabajo en ubicación de cliente.** El conductor está realizando alguna actividad distinta de la conducción en el punto de recogida (cliente).
- **TR-GE: Trabajo en instalación de GESUGA.** El conductor está realizando alguna actividad distinta de la conducción en las instalaciones de GESUGA.
- **NOID: Actividad no identificada.** El empleado está realizando alguna actividad que no puede ser identificada.
- **SINAC: Sin actividad.** El empleado no está realizando ninguna actividad.

En la Figura 6.3 se muestra la taxonomía que permite identificar las actividades de GESUGA. Como se puede observar en la Figura 6.3 y en el listado de las actividades definidas, el sistema debe identificar actividades tanto en el interior como en el exterior de las instalaciones, por ejemplo, es necesario detectar la actividad *TR-GE: Trabajo en instalación de GESUGA*. Sin embargo, como pudimos deducir de los experimentos realizados en las secciones 6.1.2 y 6.1.3, la información del GPS en interiores es de muy mala calidad. Si el sistema no detectaba correctamente las actividades desarrolladas en el interior de una instalación, suponía

un obstáculo en la detección del resto de actividades de más complejas ya que nuestras taxonomías parten de la identificación de actividades muy básicas y de bajo nivel de abstracción semántico para detectar actividades con un alto nivel de abstracción. Considerando esta limitación y estudiando diferentes posibilidades, concluimos que la mejor solución a este problema era una alternativa mixta.

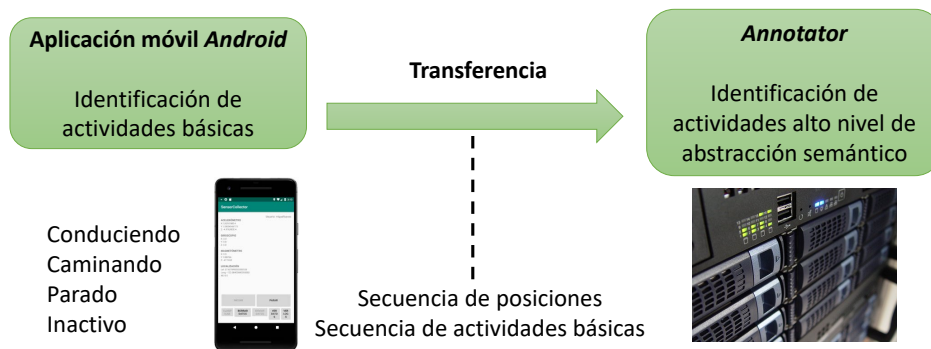


Figura 6.4: Esquema de funcionamiento de la solución mixta

Esta solución mixta consiste en aprovechar y mejorar el trabajo realizado en [GGRFBL20] y extraer la identificación de actividades básicas de las taxonomías, de manera que, en una primera fase una aplicación *Android* nativa realizó un primer trabajo de identificación de actividades básicas y posteriormente, nuestras taxonomías realizaban el trabajo de identificación de actividades más complejas y con un alto nivel de abstracción semántica.

Vamos a explicar con un poco más de detalle esta alternativa mixta, en primer lugar, una aplicación *Android* nativa recogía la información básica de cada trabajador e identificaba las actividades más básicas tanto en entornos exteriores como interiores, por ejemplo *caminando*. A continuación, nuestro componente *SensorReceiver* recibía del móvil la secuencia de ubicaciones y los actividades de bajo nivel de abstracción identificadas para su posterior procesado y conversión a actividades de un alto nivel de abstracción, tal y como se explicó en la Sección 5.3.

Esta aplicación *Android* nativa utiliza aprendizaje máquina que, en base a los datos de GPS y los sensores del dispositivo, detecta las siguientes actividades básicas (descritas en la Sección 6.1.1):

- Conduciendo
- Andando
- Activo
- Inactivo

Esta aplicación móvil *Android* se desarrolló en el CITIC y recopila datos que provienen de cuatro sensores diferentes: acelerómetro, giroscopio, magnetómetro y GPS. Se seleccionó el acelerómetro y giroscopio porque son los más usados en la literatura y los que muestran mejores resultados. También se agregaron el magnetómetro y el GPS porque podrían ser útiles en este problema. De hecho, el GPS debería ser fundamental para diferenciar las actividades realizadas al poder detectar la velocidad de movimiento del usuario que porta el *smartphone*. Además se guardan los datos del acelerómetro, el giroscopio y el magnetómetro con sus valores triaxiales. En el caso del GPS, se almacenan los incrementos del dispositivo en latitud, longitud y altitud, así como el rumbo, la velocidad y la precisión de las mediciones recopiladas. Además, para obtener valores útiles del acelerómetro que no se vean afectados por la orientación del teléfono se utiliza el sensor de gravedad para descontarla del valor del acelerómetro.

El modelo de detección basado en aprendizaje máquina lo han entrenado el alumnado e investigadores del CITIC, por lo que no ha sido necesario interrumpir las actividades de los trabajadores de las empresas.

En definitiva, esta solución mixta da solución al problema planteado y nos permitió identificar actividades desarrolladas tanto en el interior como en el exterior de edificios e instalaciones, sin tener que modificar nuestra arquitectura ni métodos para la identificación de actividades, logrando alcanzar nuestro objetivo principal, detectar de una manera simple y eficaz actividades de alto nivel de abstracción semántico sin la necesidad de un entrenamiento previo del sistema por parte de los empleados de las empresas, evitando los inconvenientes que causa:

- Dificulta el trabajo de los empleados.
- La calidad de los datos podría no ser suficiente para entrenar el algoritmo de aprendizaje máquina.

Además, esta solución mixta nos permite mantener la definición de nuestras taxonomías y el proceso de anotación. A continuación vamos a describir la taxonomía definida para la identificación de actividades en GESUGA.

Como se puede observar en la Figura 6.3, el primer paso en el procesamiento de la taxonomía de GESUGA consiste en detectar si el empleado que conduce el camión se encuentra activo, andando, conduciendo o inactivo, esto se realiza gracias a un nodo de tipo *Decision* y un predicado de tipo *HasTagPredicate*, definidos en la Sección 5.2, por ejemplo, se puede discriminar si un segmento corresponde a la actividad básica *activo o andando* utilizando el predicado de tipo *HasTagPredicate* con los valores *ACTIVE* y *WALKING* en el atributo *tag*. En segundo lugar, dependiendo de si el camionero está activo, andando, conduciendo o inactivo, se realizan diversas comprobaciones, de tal manera que, en caso de detectar que el camionero está activo o andando se utiliza la ubicación recuperada por el GPS para determinar en dónde se encuentra. O bien está en una instalación de un cliente, o bien en una de las instalaciones propias de GESUGA; o bien en otro lugar. En caso de

encontrarse en otro lugar se utiliza la duración de la parada para determinar si es una parada que legalmente se puede considerar un descanso o por el contrario es sólo una parada corta. En caso de que el empleado esté conduciendo es necesario utilizar la información de las rutas planificadas para determinar si está circulando por una ruta o no. Una vez que se determina esta información, se utiliza la red de carreteras y la velocidad recogida por el dispositivo GPS para determinar si la conducción está siendo fluida en la red de carreteras o está siendo lenta. En el caso de que el empleado esté inactivo, se etiqueta como actividad no identificada. Esto ocurre en los casos en los que el empleado ha dejado el dispositivo móvil en una superficie estática y por lo tanto, no está portando el teléfono móvil. Por último, en cualquier otro caso, el empleado no está desempeñando ninguna actividad del catálogo de actividades.

Empleado	TOTAL	GES.1: Conducción fluida en ruta planificada		GES.2: Conducción lenta en ruta planificada		GES.3: Conducción fluida fuera de ruta planificada		GES.4: Conducción lenta fuera de ruta planificada		GES.5: Descanso en la conducción		GES.6: Parada corta		GES.7: Trabajo en ubicación de cliente		GES.8: Trabajo en territorio de GESUGA		GES.9: Actividad no identificada	
		Duración	Distancia	Duración	Distancia	Duración	Distancia	Duración	Distancia	Duración	Distancia	Duración	Distancia	Duración	Distancia	Duración	Distancia	Duración	Distancia
ABRAHAM SOTELO RODRIGUEZ	39h 06'	835.80Km	13h 43'	563.52Km	-	-	05h 06'	272.28Km	-	-	02h 39'	-	02h 05'	-	12h 43'	-	02h 50'	-	-
ADRIAN GÓMEZ	66h 45'	2259.87Km	19h 46'	1945.52Km	00h 05'	6.58Km	07h 07'	307.77Km	-	-	03h 49'	-	00h 44'	-	33h 19'	-	01h 54'	-	-
ALBERTO DURAN DOMINGUEZ	73h 40'	1150.95Km	18h 30'	947.20Km	-	-	04h 24'	203.76Km	-	-	04h 34'	-	02h 36'	-	26h 26'	-	17h 11'	-	-
CRISTIAN CABADO DE LA FUENTE	77h 03'	707.73Km	14h 17'	516.73Km	00h 08'	0.97Km	05h 31'	182.74Km	00h 09'	7.29Km	02h 22'	-	00h 45'	-	43h 39'	-	10h 13'	-	-
DIEGO PREGO VARELA	06h 51'	228.92Km	02h 06'	131.07Km	-	-	01h 29'	97.85Km	-	-	-	-	00h 03'	-	02h 03'	-	01h 10'	-	-
DOMINGO VILLAVERDE	79h 22'	709.45Km	13h 53'	588.88Km	-	-	02h 51'	120.58Km	-	-	03h 01'	-	00h 43'	-	39h 22'	-	19h 32'	-	-

Figura 6.5: Interfaz de explotación de las actividades identificadas en GESUGA

En la Figura 6.5 se muestra la interfaz de explotación de datos. Esta interfaz permite realizar diferentes consultas sobre el histórico de actividades de los empleados en movilidad. Por ejemplo, permite recuperar las actividades con sus correspondientes trayectorias efectuadas en una jornada laboral o rango de fechas por los diferentes agentes en movilidad (camioneros). Otro ejemplo de consulta interesante que se puede realizar es recuperar el detalle de tiempos dedicados a cada actividad de interés.

En resumen, por una parte la interfaz de explotación de datos permite consultar el histórico de actividades, permitiendo obtener la información de etiquetado semántico de las trayectorias realizadas en una determinada jornada laboral o rango de fechas por los diferentes agentes en movilidad (los camioneros que

efectúan las recogidas de residuos). Además, permite recuperar el detalle de los tiempos dedicados a cada una de las actividades de interés (tiempos de trabajo en ubicaciones de cliente, en planta, descansos realizados, horas de conducción, etc.). Por otra parte, esta interfaz permite realizar consultas gráficas sobre el histórico de visitas, permitiendo así consultas filtrando por fecha, empleado o cliente, las rutas planificadas, orden de visitas, y detalle de las mismas. Por último, esta interfaz también permite consultar las diferentes estadísticas de las actividades, por ejemplo, en Figura 6.5 se muestra la obtención de una vista global agregada por actividades en un rango de fechas.

6.2.2. Identificación de actividades en Mayores

En MAYORES era relevante hacer un seguimiento de los movimientos de las auxiliares (cuidadoras) para poder analizarlos buscando minimizar sus desplazamientos para que pudiesen dedicar el mayor tiempo posible al cuidado de los MAYORES, lo que también implica estar con ellos, darles conversación, etc. También era relevante hacer un seguimiento de los movimientos de los residentes para poder localizarlos rápidamente en caso de necesidad, detectar problemas con residentes que deambulan por la residencia, etc. El problema que se planteó es que la tecnología de GPS no es utilizable en el interior de edificios ya que la señal de los satélites no alcanza el interior de los edificios a no ser que las personas estuvieran cerca de las ventanas, por lo que tuvimos que empezar por elegir una tecnología adecuada. Finalmente, elegimos *Bluetooth* (en concreto, *BLE: Bluetooth Low Energy*, para maximizar la duración de las baterías de los *beacons*) y una estrategia de colocación de *beacons* y antenas que se ilustra en la figura que sigue y se describe a continuación.

Como se observa en la figura:

- Cada persona (sea auxiliar o residente) portaba un *beacon* (una pulsera). Estos *beacons* simplemente emitían una señal *Bluetooth*.
- Se colocaron receptores en distintas zonas de la residencia. Inicialmente, fue suficiente con uno en cada habitación y en cada baño. En zonas más grandes (comedores, áreas comunes, pasillos) podía haber uno o más de estos receptores.
- Cada auxiliar, además de un *beacon*, también portaba un móvil que actuaba como receptor *Bluetooth*, de forma que era capaz de detectar a los usuarios más próximos.

Los receptores colocados en las distintas salas escaneaban continuamente el espectro *Bluetooth* y cuando detectaban un *beacon* en su proximidad, lo comunicaban por WiFi a un servidor que almacenaba las posiciones registradas de cada *beacon*. Cada registro consistía en un identificador de *beacon*, un identificador



Figura 6.6: Colocación de receptores y *beacons Bluetooth* en Mayores

de receptor, y un instante temporal. Finalmente, una tarea periódica recogía las posiciones identificadas cada día y las asociaba a los residentes y los auxiliares. Todos estos datos se almacenaron de forma histórica y podían ser consultados posteriormente por la aplicación de explotación de datos. De esta forma podía obtenerse de forma sencilla información tan relevante para un familiar como el tiempo que habían pasado cuidando a la persona mayor, o cuánto tiempo habían invertido en su aseo personal, o darle de comer, o si se habían parado a charlar con él en el pasillo. Este tipo de información pudo y puede ayudar a dar buena fe de la calidad del servicio ofrecido.

En MAYORES no sólo pretendíamos identificar las actividades que realizaban las auxiliares (nombre de las trabajadoras que realizan las actividades de cuidado de los MAYORES) sino que también se quiso identificar las actividades de los residentes y muy especialmente aquellos momentos en donde había interacción entre un residente y una auxiliar, pues esa actividad significaba que existía un apoyo, ayuda o simplemente un ratito de interacción humana. Consideramos que la calidad del servicio que se ofrecía en MAYORES descansaba precisamente sobre esos momentos de interacción, por ello era de vital importancia la detección automática para permitir la medición del tiempo dedicado a esas interacciones. Así, la identificación y caracterización de las actividades relevantes para MAYORES que se realizó dio como resultado las siguientes actividades:

Residentes

- *RES - Baño.* El residente se encuentra en el baño sin asistencia.
- *RES - Habitación.* El residente se encuentra en la habitación sin asistencia.
- *RES - Alimentación.* El residente se encuentra en el comedor sin asistencia.
- *RES - En zonas comunes.* El residente está en una zona común sin asistencia.
- *RES - En habitación incorrecta.* El residente está en otra habitación.
- *RES - Deambular.* El residente está realizando movimientos no específicos (caminar, pasear *sin rumbo*) sin la presencia de una auxiliar.

Auxiliares

- *AUX - Habitación.* La auxiliar se encuentra en una habitación sin presencia de un residente.
- *AUX - Comedor.* La auxiliar se encuentra en el comedor sin presencia de residente.
- *AUX - Trabajo en sala común.* La auxiliar se encuentra activa en la sala común sin interactuar con residentes.
- *AUX - Trabajo indeterminado.* La auxiliar se encuentra activa sin interactuar con residentes, pero no se encuentra en una sala específica.
- *AUX - Descanso.* La auxiliar se encuentra realizando su periodo de descanso.

Interacciones

- *INT - Baño.* Ayuda en la higiene personal.
- *INT - Cama.* Ayuda a levantarse o acostarse
- *INT - Traslado.* Acompañar (o llevar en silla) a un residente a una ubicación
- *INT - Alimentación.* Asistir en la comida a un residente; puede ser en el comedor, habitación o áreas comunes.
- *INT - Asistencia individual.* Asistencia no específica (p.ej. charla en el pasillo)
- *INT - Asistencia en grupo.* La auxiliar Interacciona con un grupo (p.ej. en la sala común o en el comedor)

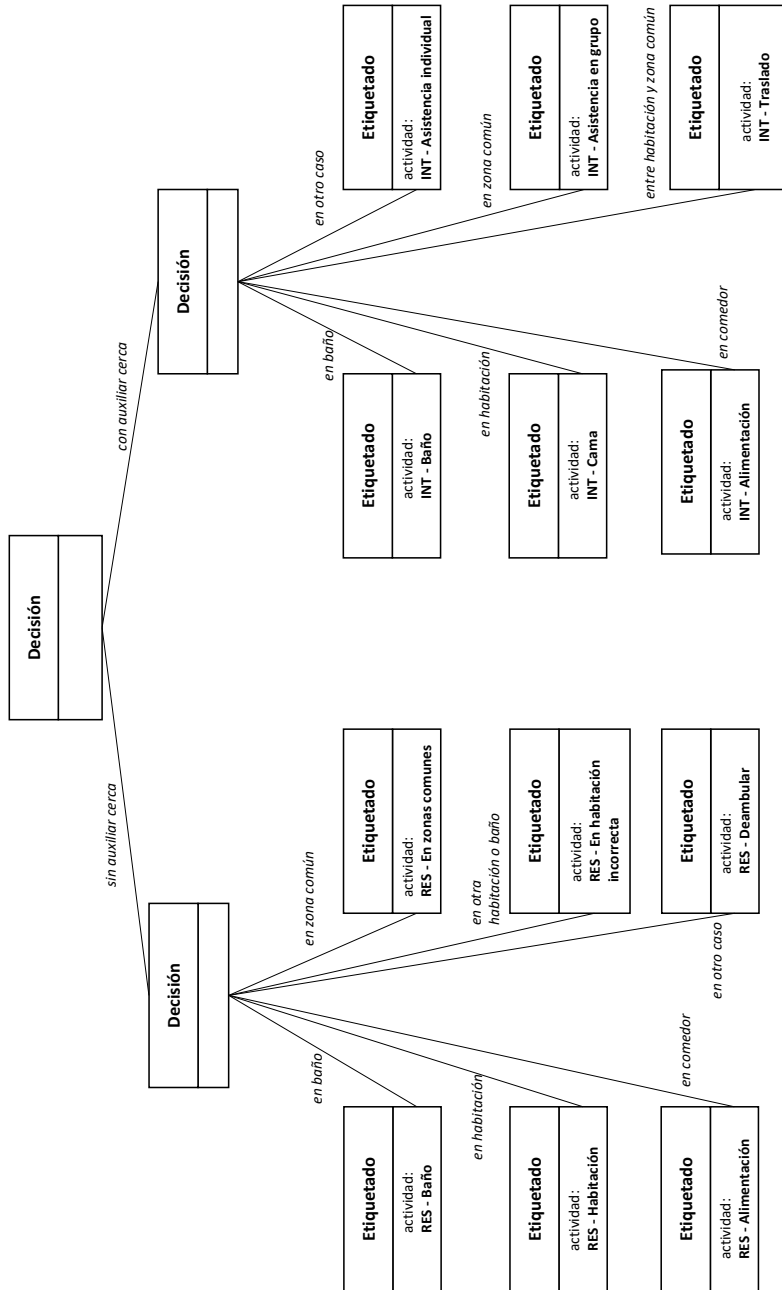


Figura 6.7: Taxonomía de actividades para residentes en Mayores

En la Figura 6.7 se muestra la taxonomía que permitió identificar las actividades de MAYORES relativas a los residentes. En el primer nodo de procesamiento se determina si el residente tiene una auxiliar cerca. En caso negativo, se comprueba su ubicación y dependiendo del lugar donde se encuentre dentro de las instalaciones, las actividades detectadas podían ser, *RES - Baño* si se encontraba en el baño de su habitación, *RES - Habitación* si estaba en su habitación, *RES - Alimentación* si se encontraba en el comedor, *RES - En zonas comunes* si se estaba en una zona común, *RES - En habitación incorrecta* si se encontraba en otra habitación o el baño, y por último *RES - Deambular* si se estaba en otra ubicación de las instalaciones. Por otro lado, cuando el residente tenía una auxiliar cerca, se comprobaba su ubicación, y dependiendo de su localización en el interior de las instalaciones, las actividades a realizar podían ser, respectivamente, si estaba en el baño de su habitación, *INT - Baño*, si se encontraba en su habitación, *INT - Cama*, si estaba en el comedor, *INT - Alimentación*, si se encontraba entre una habitación y una zona común, *INT - Traslado*, si estaba en una zona común, *INT - Asistencia en grupo*, y por último, si se encontraba en cualquier otra ubicación, la actividad detectada era *INT - Asistencia individual*.

En la Figura 6.8 se muestra la taxonomía que permitió identificar las actividades de MAYORES relativas a las auxiliares. En el primer nodo de procesamiento se determina si la auxiliar tenía un usuario cerca. En caso negativo, si la auxiliar se encontraba inactiva, entonces estaba en un descanso. Por el contrario, si la auxiliar se encontraba activa, se comprobaba su ubicación y dependiendo del lugar donde se encontrara dentro de las instalaciones, la actividad detectada era *AUX - Habitación* si se encontraba en una habitación, *AUX - Comedor* si estaba en el comedor, *AUX - Trabajo en sala común* si se encontraba en una zona común, y por último, *AUX - Trabajo indeterminado* si se encontraba en otra localización de las instalaciones. Por otro lado, cuando la auxiliar tenía un residente cerca, se comprobaba la ubicación de la auxiliar, y dependiendo de su localización en el interior de las instalaciones, las actividades a realizar podían ser, respectivamente, si estaba en el baño, *INT - Baño*, si se encontraba en una habitación, *INT - Cama*, si estaba en el comedor, *INT - Alimentación*, si se encontraba entre una habitación y una zona común, *INT - Traslado*, si estaba en una zona común, *INT - Asistencia en grupo*, y por último, si se encontraba en cualquier otra ubicación, la actividad detectada es *INT - Asistencia individual*.

Desafortunadamente, debido a la situación causada por la COVID-19, y especialmente tratándose de una residencia de mayores, la implantación del sistema de posicionamiento no pudo llevarse a cabo. Se realizó una prueba de concepto, para validar la tecnología, instalando 5 sensores (en una habitación, un baño, una sala común, un comedor y la recepción) y 4 *beacons* (2 para auxiliares y 2 para residentes). Se verificó que la captura de datos funcionaba correctamente. También se verificó que el componente de integración de datos desde el sistema de posicionamiento a la base de datos espacial funcionó correctamente. Sin embargo,

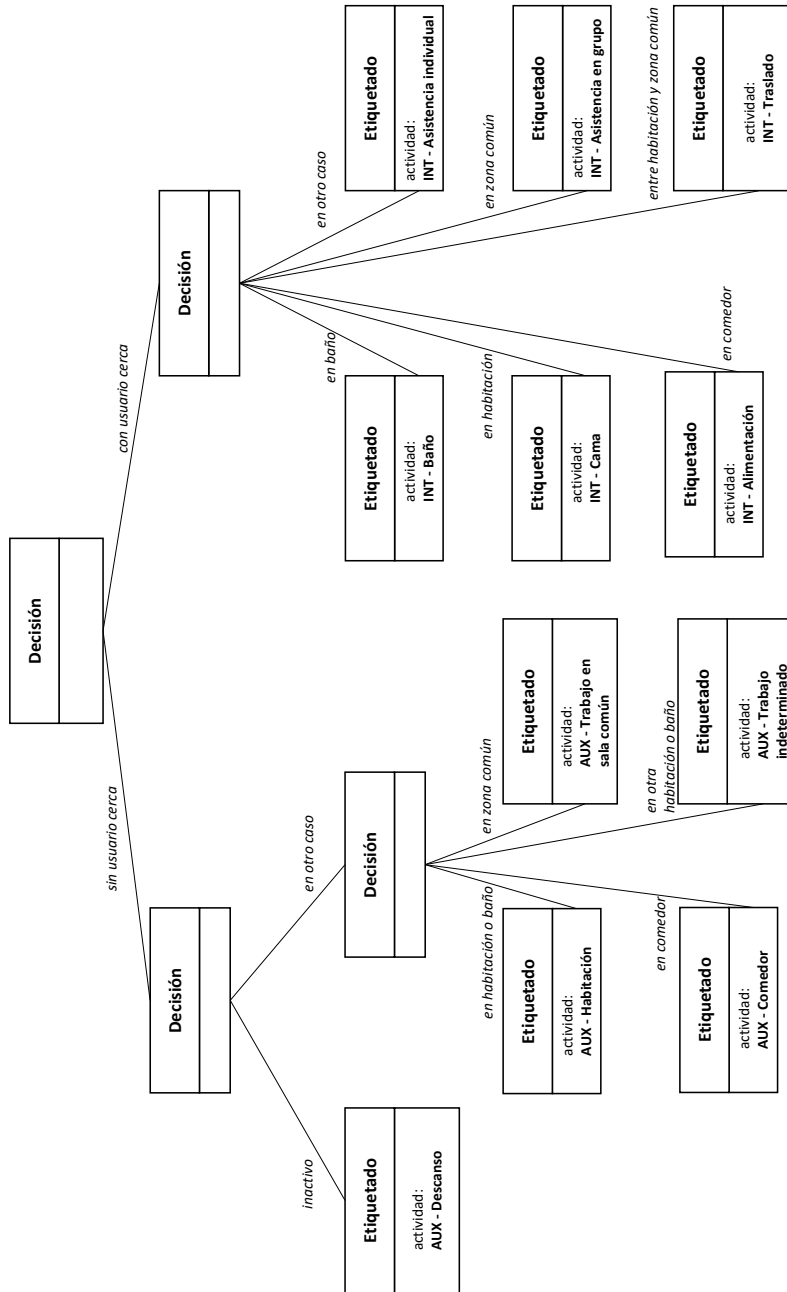


Figura 6.8: Taxonomía de actividades para auxiliares en Mayores

no se pudo utilizar ya que el personal y los residentes dejaron de portar los *beacons* y el sistema quedó desatendido. Por supuesto, las limitaciones impuestas por la pandemia impidieron que personal externo a la residencia pudiese entrar para tratar de solventar ningún problema del sistema.

Para continuar con el resto del sistema, y más concretamente con el componente de detección de actividades (*Annotator*), se construyó un simulador de datos. Con este simulador se generaron datos de posicionamiento para 103 personas (69 residentes y 34 auxiliares) *obtenidas* por 198 sensores distribuidos a lo largo de la residencia, para un total de 2000 días. Hay que destacar que el simulador consideró variabilidad en el comportamiento, especialmente de los residentes, considerando diversas características, entre ellas el grado de dependencia y de deambulación, frecuencia de uso del baño y tiempo dedicado a las comidas. Con esta información, se completó la implementación del componente de detección de tareas y se validó utilizando estos datos de prueba.

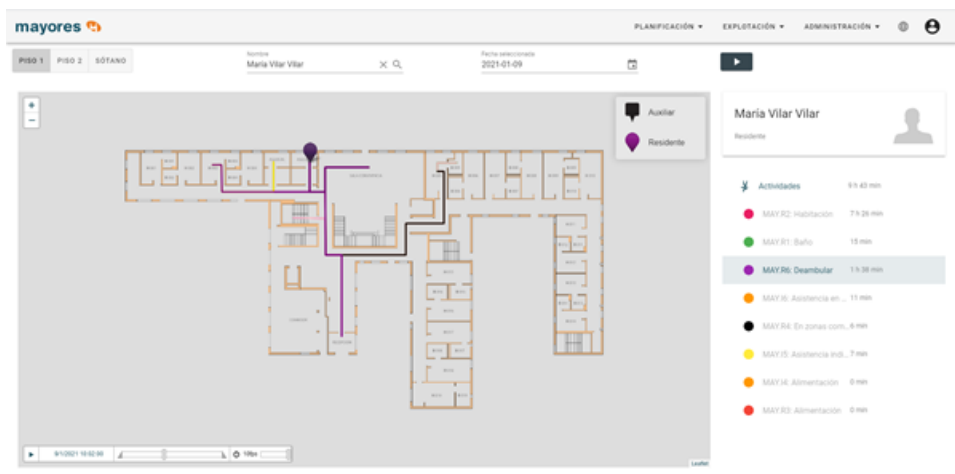


Figura 6.9: Interfaz de explotación de las actividades identificadas en Mayores

Finalmente, una vez identificadas las actividades realizadas por residentes y auxiliares, en el proyecto se implementó un interfaz de usuario para explotar los datos. La Figura 6.9 muestra el recorrido realizado por una persona, en este caso un residente, durante un día. Como se puede observar, a la derecha se informa de las actividades detectadas y la duración de las mismas. Cabe señalar, en primer lugar que la interfaz de explotación de datos permite consultar el histórico de actividades, permitiendo obtener la información de etiquetado semántico de las trayectorias realizadas en una determinada jornada laboral o rango de fechas por las diferentes auxiliares y residentes. Además, también permite recuperar el

detalle de los tiempos dedicados a cada una de las actividades de interés (tiempos de trabajo en las distintas localizaciones de la residencia, descansos realizados, asistencias individuales, etc.). En segundo lugar, esta interfaz permitía también realizar consultas gráficas filtrando por fecha, auxiliar o residentes y obtener así las rutas en el interior de las instalaciones, orden de las tareas realizadas, y detalle de las mismas.

6.3. Conclusiones

En conclusión, en este capítulo hemos descrito las diferentes pruebas experimentales realizadas a lo largo del trabajo de investigación de esta tesis y podemos considerar que en líneas generales los resultados de la evaluación experimental han sido óptimos. Sin embargo, cabe destacar, que el mayor obstáculo que nos hemos encontrado tras las pruebas realizadas ha sido distinguir con un alto grado de precisión las actividades andando y activo.

Por otra parte, concluimos que hemos sido capaces de mejorar los resultados obtenidos en otro trabajo de investigación del estado del arte [GGRFBL20]¹ al identificar actividades básicas como andando, conduciendo, activo e inactivo. Sin embargo, estos resultados obtenidos de la comparación de ambas pruebas eran los esperados porque el trabajo [GGRFBL20] únicamente emplea *Support vector machine* (SVM).

Por último, cabe señalar que hemos realizado dos pruebas de concepto en dos empresas reales con metodologías MWM y podemos concluir que los resultados han sido satisfactorios, permitiendo a ambas empresas explotar los datos recogidos, consultar el histórico de actividades, obtener la información de etiquetado semántico de las trayectorias realizadas en una determinada jornada laboral o rango de fechas, entre otras funcionalidades que involucran la explotación masiva de datos. Es decir, cumplir con todos los objetivos y funcionalidades planteadas al inicio de estas dos pruebas conceptos, y obteniendo resultados óptimos.

¹Los datos están disponibles en <https://lbd.udc.es/research/real-life-HAR-dataset/>

Capítulo 7

Conclusiones

En este capítulo se resumen los resultados obtenidos en esta tesis. En la Sección 7.1 se presentan nuestras conclusiones, detallando las aportaciones principales del trabajo realizado. Finalmente, en la Sección 7.2, se describen las líneas de trabajo futuro que abre esta tesis.

7.1. Conclusiones y aportaciones principales

Como ya se detalló Introducción (Sección 1), al inicio de esta tesis se pretendía explorar un campo de investigación en auge y que ofrece un gran abanico de posibilidades, la identificación de actividades semánticas en entornos MWM. Concretamente los objetivos que pretendíamos alcanzar con el trabajo de esta tesis eran los siguientes:

- Conseguir una arquitectura de un sistema completo para la identificación de actividades semánticas de trabajadores en movilidad que pudiese ser integrado en los procesos de negocio de cualquier empresa mediante la interacción con su sistema MWM (*Mobile Workforce Managent*).
- Definir una metodología de identificación de actividades semánticas.

Respecto al primer objetivo, consideramos que hemos propuesto una arquitectura que cumple con todos los requisitos iniciales y que incluye desde la aplicación móvil de recogida de datos (*SensorCollector*), tal y como se describe en la Sección 4, hasta la identificación de las actividades (*Annotator*), pasando por el almacenamiento de los datos crudos y la interacción con el MWM.

Además, concretamente, esta arquitectura propuesta logra alcanzar los siguientes objetivos:

- Minimizar la transferencia de información gracias al diseño de un método que permite reducir la gran cantidad de datos que se recogen por el componente de captura de datos en el dispositivo móvil.
- Alta eficiencia energética en su funcionamiento.
- Robustez frente a problemas de conectividad. La arquitectura construida presenta un diseño que permite no tener conexión permanente con el sistema central.
- Unificar los datos procedentes de diferentes fuentes, desde la información de contexto hasta los datos de los sensores y la ubicación GPS. El diseño de la arquitectura permite dar solución al problema de las distintas granularidades que presentan los sensores y los datos de localización, permitiendo agregarlos y fusionarlos.

Respecto al segundo objetivo, definir una metodología de identificación de actividades semánticas, consideramos que hemos propuesto un componente de detección de actividades semánticas que alcanza tres objetivos fundamentales planteados a lo largo del trabajo de esta tesis:

- En primer lugar, se ha definido un modelo que permite representar las actividades que deben ser identificadas y que aporta la forma de identificarlas. Para ello, hemos creado el concepto de taxonomías de actividades, una manera intuitiva y sencilla de identificar actividades de empleados en entornos MWM.
- En segundo lugar, se ha definido de forma detallada los elementos necesarios para construir las taxonomías de actividades. En particular, hemos definido un *lenguaje de especificación de patrones de actividad* que se utiliza en las taxonomías para decidir qué actividad se está realizando.
- En tercer lugar, se ha diseñado un proceso para la detección de actividades semánticas en un alto nivel de abstracción semántico y con un grado de precisión elevado en comparación con otros trabajos del estado del arte.

Por último, para finalizar el trabajo de esta tesis se han realizado diferentes experimentos en entornos reales. Concretamente, podríamos agruparlas y resumirlas como siguen:

- Dos experimentos para la detección de actividades básicas en entornos de exteriores.
- Dos pruebas de concepto para la detección de actividades de alto nivel de abstracción semántica en dos empresas reales que pertenecen a entornos MWM.

Respecto a los dos primeros experimentos, consideramos que hemos demostrado que es posible la identificación de actividades básicas en exteriores mediante nuestra propuesta, siendo los resultados de la evaluación experimental satisfactorios e incluso mejorando los resultados de otro trabajo de investigación del estado del arte, [GGRFBL20].

Respecto a las dos pruebas de concepto realizadas en dos empresas reales y que emplean sistemas MWM, concluimos que los resultados han sido satisfactorios, permitiendo a ambas empresas explotar los datos recogidos, consultar el histórico de actividades, obtener la información de etiquetado semántico de las trayectorias realizadas en una determinada jornada laboral o rango de fechas por las diferentes, etc. Es decir, cumplir con todos los objetivos y funcionalidades planteadas al inicio de estas dos pruebas conceptos, y obteniendo resultados satisfactorios.

En conclusión, consideramos, que esta tesis ha cumplido con los requisitos iniciales, ha logrado los objetivos planteados y ha sabido dar solución a los diferentes problemas del contexto, presentado una nueva metodología mediante el concepto de taxonomías de actividades que la convierten en genérica, flexible y extensible. Esta metodología permite convertir los datos crudos recogidos por los sensores de dispositivos móviles en trayectorias anotadas con actividades semánticas de alto nivel de abstracción. Además, la taxonomía de actividades describe los valores esperados para cada una de las variables que se recogen en el sistema utilizando predicados definidos en un *lenguaje de especificación de patrones*, muy expresivo y que tiene en cuenta no sólo los datos de los sensores sin procesar sino también los datos almacenados en el sistema *MWM*, y de la información geográfica del contexto relacionada con el dominio. Finalmente, cabe señalar que se trata de un sistema que puede ser fácilmente integrado en sistemas MWM y en el flujo de trabajo de cualquier empresa.

7.2. Líneas de trabajo futuro

En esta sección vamos a introducir algunas de las líneas de investigación abiertas en el trabajo de esta tesis. Estos retos han surgido a lo largo del desarrollo de esta tesis, concretamente la mayor parte aparecieron cuando se realizaron las pruebas experimentales y de concepto.

En primer lugar, contemplamos mejorar y ampliar los tipos nodos de procesamiento así como los predicados que se puedan definir en las taxonomías de actividades. Por una parte, uno de nuestros objetivos es modificar el nodo *ParadoYMovimiento* para mejorar la precisión en la detección de actividades básicas como parado y movimiento. Por otra parte, otro de nuestros objetivos es incluir nuevos nodos de procesamiento y nuevos tipos de predicados para poder expresar taxonomías más complejas. Por ejemplo, en la actualidad, una taxonomía de actividades sólo puede utilizar datos de un empleado en el procesamiento y no se permite comparar su actividad con la del resto de empleados del sistema.

Por otro lado, las pruebas de concepto en ambas empresas también abrieron nuevos retos de investigación. En concreto, tenemos como objetivo buscar alternativas y testearlas en entornos MWM reales para determinar cuál es la mejor forma para la identificación de actividades en interiores.

Además, otro reto de investigación que ha surgido a la hora de realizar las pruebas de investigación es poder comparar nuestra propuesta con más alternativas. En concreto nuestro objetivo es tomar como punto de partida otras investigaciones que publiquen tanto el método como el conjunto de datos utilizado, y así poder contestar a las siguientes preguntas, *¿somos capaces de identificar sus actividades?*, *¿cómo se compara nuestra recuperación y precisión con la suya?*, *¿qué método requiere menos esfuerzo para construir el sistema?*.

Por otra parte, otros trabajos de investigación que nos planteamos en un futuro es ampliar las pruebas experimentales en entornos reales. Nuestro objetivo es obtener el porcentaje de recuperación y precisión en pruebas con empresas reales y que además estas necesiten la detección de otras actividades semánticas que conlleven la definición de taxonomías más complejas.

Por último, también consideramos proteger la propiedad intelectual del resultado y buscar posibilidades para su comercialización, lo que culminaría con nuestro trabajo de investigación y la posibilidad de explotar por fin toda la información que engloban los datos crudos de las trayectorias y permitir así la retroalimentación de los sistemas MWM, sin duda alguna un campo en auge en los últimos años.

Bibliografía

- [AA07] N. Andrienko and G. Andrienko. Designing visual analytics methods for massive collections of movement data. *Cartographica*, 42(2):117–138, 2007. Cited By :73.
- [ABK⁺07] L. O. Alvares, V. Bogorny, B. Kuijpers, J. A. F. De Macedo, B. Moelans, and A. Vaisman. A model for enriching trajectories with semantic geographical information. In *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*, pages 162–169, 2007. Cited By :243.
- [ACG19] M. Arslan, C. Cruz, and D. Ginhac. Semantic trajectory insights for worker safety in dynamic environments. *Automation in Construction*, 106, 2019. Cited By :9.
- [AS03] D. Ashbrook and T. Starner. Using gps to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing*, 7(5):275–286, 2003. Cited By :790.
- [BGWSB19] I. Ben-Gal, S. Weinstock, G. Singer, and N. Bambos. Clustering users by their mobility behavioral patterns. *ACM Transactions on Knowledge Discovery from Data*, 13(4), 2019. Cited By :1.
- [BZL18] E. Ben Zion and B. Lerner. Identifying and predicting social lifestyles in people’s trajectories by neural networks. *EPJ Data Science*, 7(1), 2018. Cited By :7.
- [CCJ10] X. Cao, G. Cong, and C. S. Jensen. Mining significant semantic locations from gps data. *Proceedings of the VLDB Endowment*, 3(1):1009–1020, 2010. Cited By :270.
- [CCX⁺18] Y. Cao, Z. Cai, F. Xue, T. Li, and Z. Ding. Semantic trajectory based behavior generation for groups identification. *KSI Transactions on Internet and Information Systems*, 12(12):5782–5799, 2018. Cited By :3.

- [CDF95] Eliseo Clementini and Paolino Di Felice. A comparison of methods for representing topological relationships. *Inf. Sci.*, 3(3):149–178, May 1995.
- [CDF96] Eliseo Clementini and Paolino Di Felice. A model for representing topological relationships between complex geometric features in spatial databases. *Inf. Sci.*, 90(1–4):121–136, April 1996.
- [CLL18a] G. Cai, K. Lee, and I. Lee. Mining mobility patterns from geotagged photos through semantic trajectory clustering. *Cybernetics and Systems*, 49(4):234–256, 2018. Cited By :4.
- [CLL18b] G. Cai, K. Lee, and I. Lee. Mining semantic trajectory patterns from geo-tagged data. *Journal of Computer Science and Technology*, 33(4):849–862, 2018. Cited By :7.
- [CXC⁺20] Y. Cao, F. Xue, Y. Chi, Z. Ding, L. Guo, Z. Cai, and H. Tang. Effective spatio-temporal semantic trajectory generation for similar pattern group identification. *International Journal of Machine Learning and Cybernetics*, 11(2):287–300, 2020. Cited By :5.
- [dAdSBdAS20] D. R. de Almeida, C. de Souza Baptista, F. G. de Andrade, and A. Soares. A survey on big data for trajectory analytics. *ISPRS International Journal of Geo-Information*, 9(2), 2020. Cited By :1.
- [FPA⁺20] C. A. Ferrero, L. M. Petry, L. O. Alvares, C. L. da Silva, W. Zalewski, and V. Bogorny. Mastermovelets: discovering heterogeneous movelets for multiple aspect trajectory classification. *Data Mining and Knowledge Discovery*, 34(3):652–680, 2020. Cited By :1.
- [Fri02] Billie Friedland. Book reviews: Conducting research literature reviews: From paper to the internet, by arlene fink (1998), sage publications, 265 pages, \$29.95 paper. *Teacher Education and Special Education*, 25(2):210–210, 2002.
- [GGRFBL20] Daniel Garcia-Gonzalez, Daniel Rivero, Enrique Fernandez-Blanco, and Miguel R. Luaces. A public domain dataset for real-life human activity recognition using smartphone sensors. *Sensors*, 20(8), 2020.
- [GLL⁺16] C. Guan, X. Lu, X. Li, E. Chen, W. Zhou, and H. Xiong. Discovery of college students in financial hardship. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, volume 2016-January, pages 141–150, 2016. Cited By :17.

- [GMSK08] Baris Guc, Michael May, Yucel Saygin, and Christine Kopp. Semantic annotation of gps trajectories. *11th AGILE International Conference on Geographic Information Science*, 01 2008.
- [GP17] F. J. García-Peñalvo. Revisión sistemática de literatura en los Trabajos de Final de Máster y en las Tesis Doctorales, March 2017.
- [GS15] A. Y. Grinberger and N. Shoval. A temporal-contextual analysis of urban dynamics using location-based data. *International Journal of Geographical Information Science*, 29(11):1969–1987, 2015. Cited By :11.
- [GSY⁺21] Z. Gui, Y. Sun, L. Yang, D. Peng, F. Li, H. Wu, C. Guo, W. Guo, and J. Gong. Lsi-lstm: An attention-aware lstm for real-time driving destination prediction by considering location semantics and location importance of trajectory points. *Neurocomputing*, 440:72–88, 2021.
- [GVD15] R. H. Güting, F. Valdés, and M. L. Damiani. Symbolic trajectories. *ACM Transactions on Spatial Algorithms and Systems*, 1(2), 2015. Cited By :40.
- [GZLH18] X. Guo, R. Zhang, X. Liu, and J. Huai. Human mobility semantics analysis: a probabilistic and scalable approach. *GeoInformatica*, 22(3):507–539, 2018.
- [GZW⁺18] L. Guo, D. Zhang, Y. Wang, H. Wu, B. Cui, and K. . Tan. Co2: Inferring personal interests from raw footprints by connecting the offline world with the online world. *ACM Transactions on Information Systems*, 36(3), 2018. Cited By :5.
- [HGW⁺21] S. Hu, S. Gao, L. Wu, Y. Xu, Z. Zhang, H. Cui, and X. Gong. Urban function classification at road segment level using taxi trajectory data: A graph convolutional neural network approach. *Computers, Environment and Urban Systems*, 87, 2021.
- [HWG⁺20] L. Huang, Y. Wen, W. Guo, X. Zhu, C. Zhou, F. Zhang, and M. Zhu. Mobility pattern analysis of ship trajectories based on semantic transformation and topic model. *Ocean Engineering*, 201, 2020. Cited By :5.
- [HZC⁺20] Z. Huang, Y. Zhao, W. Chen, S. Gao, K. Yu, W. Xu, M. Tang, M. Zhu, and M. Xu. A natural-language-based visual query approach of uncertain human trajectories. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1256–1266, 2020. Cited By :2.

- [ISO03] ISO. *Geographic information – Spatial schema*. International Organization for Standardization (TC 211), 2003.
- [ISO04] ISO. *Geographic information — Simple feature access — Part 1: Common architecture*. International Organization for Standardization (TC 211), 2004.
- [JC18] M. Jin and C. Claramunt. A semantic model for human mobility in an urban region. *Journal on Data Semantics*, 7(3):171–187, 2018. Cited By :4.
- [JTL⁺20] C. Jin, T. Tao, X. Luo, Z. Liu, and M. Wu. S2n2: An interpretive semantic structure attention neural network for trajectory classification. *IEEE Access*, 8:58763–58773, 2020.
- [KC07] Barbara Ann Kitchenham and Stuart Charters. Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report, 07 2007.
- [KH06] John Krumm and Eric Horvitz. Predestination: Inferring destinations from partial trajectories. In *Proceedings of the 8th International Conference on Ubiquitous Computing*, UbiComp’06, page 243–260, Berlin, Heidelberg, 2006. Springer-Verlag.
- [LCC13] M. Lv, L. Chen, and G. Chen. Mining user similarity based on routine activities. *Information Sciences*, 236:17–32, 2013. Cited By :46.
- [LCX⁺16] M. Lv, L. Chen, Z. Xu, Y. Li, and G. Chen. The discovery of personally semantic places based on trajectory data mining. *Neurocomputing*, 173:1142–1153, 2016. Cited By :47.
- [LFK05] L. Liao, D. Fox, and H. Kautz. Location-based activity recognition using relational markov networks. In *IJCAI International Joint Conference on Artificial Intelligence*, pages 773–778, 2005. Cited By :257.
- [LSLB⁺11] Antonio Hidalgo Landa, Istvan Szabo, Liam Le Brun, Ian Owen, and Graham Fletcher. An evidence-based approach to scoping reviews. *Electronic Journal of Information Systems Evaluation*, 14(1):pp46–52, 2011.
- [LZD⁺19] Z. Lin, Q. Zeng, H. Duan, C. Liu, and F. Lu. A semantic user distance metric using gps trajectory data. *IEEE Access*, 7:30185–30196, 2019. Cited By :2.

- [LZX⁺21] J. Li, F. Zeng, Z. Xiao, Z. Zheng, H. Jiang, and Z. Li. Social relationship inference over private vehicle mobility data. *IEEE Transactions on Vehicular Technology*, 2021.
- [MFdAL21] D. Minatel, V. Ferreira, and A. de Andrade Lopes. Local-entity resolution for building location-based social networks by using stay points. *Theoretical Computer Science*, 851:62–76, 2021.
- [MPLDSE⁺20] L. May Petry, C. Leite Da Silva, A. Esuli, C. Renso, and V. Bogorny. Marc: a robust method for multiple-aspect trajectory classification via space, time, and semantic embeddings. *International Journal of Geographical Information Science*, 34(7):1428–1450, 2020.
- [MPLK16] P. Mazumdar, B. K. Patra, R. Lock, and S. B. Korra. An approach to compute user similarity for gps applications. *Knowledge-Based Systems*, 113:125–142, 2016. Cited By :14.
- [PMPP08] Ruggero Gaetano Pensa, Anna Monreale, Fabio Pinelli, and Dino Pedreschi. Pattern-preserving k-anonymization of sequences and its application to mobility data mining. In *International Workshop on Privacy in Location-Based Applications PiLBA'08*, volume 397, pages 44–60. CEUR-WS. org, 2008.
- [Pou16] Jacob Poushter. Smartphone ownership and internet usage continues to climb in emerging economies, 2016. Recuperado de: <https://www.pewresearch.org/global/2016/02/22/smartphone-ownership-and-internet-usage-continues-to-climb-in-emerging-economies/> [03/10/2021].
- [PSR⁺13] C. Parent, S. Spaccapietra, C. Renso, G. Andrienko, N. Andrienko, V. Bogorny, M. L. Damiani, A. Gkoulalas-Divanis, J. Macedo, N. Pelekis, Y. Theodoridis, and Z. Yan. Semantic trajectories modeling and analysis. *ACM Computing Surveys*, 45(4), 2013. Cited By :308.
- [PTTHGZ19] R. Pérez-Torres, C. Torres-Huitzil, and H. Galeana-Zapién. A spatio-temporal approach to individual mobility modeling in on-device cognitive computing platforms. *Sensors (Switzerland)*, 19(18), 2019.
- [RTO⁺10] J. A. M. R. Rocha, V. C. Times, G. Oliveira, L. O. Alvares, and V. Bogorny. Db-smot: A direction-based spatio-temporal clustering method. In *2010 IEEE International Conference on Intelligent Systems, IS 2010 - Proceedings*, pages 114–119, 2010. Cited By :85.

- [SA21] T. Sakouhi and J. Akaichi. Dynamic and multi-source semantic annotation of raw mobility data using geographic and social media data. *Pervasive and Mobile Computing*, 71, 2021.
- [SCR10] L. Spinsanti, F. Celli, and C. Renso. Where you stop is who you are: Understanding people’s activities by places visited. In *CEUR Workshop Proceedings*, volume 678, pages 38–52, 2010. Cited By :10.
- [SGP+20] G. M. Santipantakis, A. Glenis, K. Patroumpas, A. Vlachou, C. Doulkeridis, G. A. Vouros, N. Pelekis, and Y. Theodoridis. Spartan: Semantic integration of big spatio-temporal data from streaming and archival sources. *Future Generation Computer Systems*, 110:540–555, 2020.
- [Shu11] Jonathan J Shuster. Cochrane handbook for systematic reviews for interventions, version 5.1. 0, published 3/2011. julian pt higgins and sally green, editors, 2011.
- [SLC+21] H. Shi, Y. Li, H. Cao, X. Zhou, C. Zhang, and V. Kostakos. Semantics-aware hidden markov model for human mobility. *IEEE Transactions on Knowledge and Data Engineering*, 33(3):1183–1194, 2021.
- [SPT16] S. Sideridis, N. Pelekis, and Y. Theodoridis. On querying and mining semantic-aware mobility timelines. *International Journal of Data Science and Analytics*, 2(1-2):29–44, 2016. Cited By :3.
- [SZZ+14] H. Su, K. Zheng, K. Zeng, J. Huang, and X. Zhou. Stmaker-a system to make sense of trajectory data. In *Proceedings of the VLDB Endowment*, volume 7, pages 1701–1704, 2014. Cited By :28.
- [TBKA08] A. Tietbohl, V. Bogorny, B. Kuijpers, and L. O. Alvares. A clustering-based approach for discovering interesting places in trajectories. In *Proceedings of the ACM Symposium on Applied Computing*, pages 863–868, 2008. Cited By :285.
- [TZDC18] L. Tang, Y. Zhao, Z. Duan, and J. Chen. Efficient similarity search for travel behavior. *IEEE Access*, 6:68760–68772, 2018. Cited By :3.
- [TZY+13] L. . Tang, Y. Zheng, J. Yuan, J. Han, A. Leung, W. . Peng, and T. . Porta. A framework of traveling companion discovery on trajectory data streams. *ACM Transactions on Intelligent Systems and Technology*, 5(1), 2013. Cited By :62.

- [VSD⁺19] G. A. Vouros, G. M. Santipantakis, C. Doukeridis, A. Vlachou, G. Andrienko, N. Andrienko, G. Fuchs, J. M. Cordero Garcia, and M. G. Martinez. The datacron ontology for the specification of semantic trajectories: Specification of semantic trajectories for data transformations supporting visual analytics. *Journal on Data Semantics*, 8(4):235–262, 2019. Cited By :2.
- [WLF⁺17] P. Wang, G. Liu, Y. Fu, Y. Zhou, and J. Li. Spotting trip purposes from taxi trajectories: A general probabilistic model. *ACM Transactions on Intelligent Systems and Technology*, 9(3), 2017. Cited By :9.
- [WMC21] S. Wang, G. Mei, and S. Cuomo. A generic paradigm for mining human mobility patterns based on the gps trajectory data using complex network analysis. *Concurrency Computation*, 33(4), 2021.
- [WYC⁺19] Z. Wang, Y. Yuan, L. Chang, X. Sun, and X. Luo. A graph-based visual query method for massive human trajectory data. *IEEE Access*, 7:160879–160888, 2019. Cited By :1.
- [WYH⁺19] L. Wu, L. Yang, Z. Huang, Y. Wang, Y. Chai, X. Peng, and Y. Liu. Inferring demographics from human trajectories and geographical context. *Computers, Environment and Urban Systems*, 77, 2019. Cited By :9.
- [WZYS18] C. Wan, Y. Zhu, J. Yu, and Y. Shen. Smopat: Mining semantic mobility patterns from trajectories of private vehicles. *Information Sciences*, 429:12–25, 2018. Cited By :19.
- [XDZ09] K. Xie, K. Deng, and X. Zhou. From trajectories to activities: A spatio-temporal join approach. In *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*, pages 25–32, 2009. Cited By :44.
- [YCP⁺11] Z. Yan, D. Chakraborty, C. Parent, S. Spaccapietra, and K. Aberer. Semitri: A framework for semantic annotation of heterogeneous trajectories. In *ACM International Conference Proceeding Series*, pages 259–270, 2011. Cited By :124.
- [YCP⁺13a] Z. Yan, D. Chakraborty, C. Parent, S. Spaccapietra, and K. Aberer. Semantic trajectories: Mobility data computation and annotation. *ACM Transactions on Intelligent Systems and Technology*, 4(3), 2013. Cited By :139.
- [YCP⁺13b] Z. Yan, D. Chakraborty, C. Parent, S. Spaccapietra, and K. Aberer. Semantic trajectories: Mobility data computation and annotation. *Computer Communication Review*, 43(3), 2013. Cited By :1.

- [ZCL⁺10] Y. Zheng, Y. Chen, Q. Li, X. Xie, and W. . Ma. Understanding transportation modes based on gps data for web applications. *ACM Transactions on the Web*, 4(1), 2010. Cited By :299.
- [ZFL⁺07] C. Zhou, D. Frankowski, P. Ludford, S. Shekhar, and L. Terveen. Discovering personally meaningful places: An interactive clustering approach. *ACM Transactions on Information Systems*, 25(3), 2007. Cited By :150.
- [Zhe15] Y. Zheng. Trajectory data mining: An overview. *ACM Transactions on Intelligent Systems and Technology*, 6(3), 2015. Cited By :886.
- [ZKSN17] A. Zafar, M. Kamran, S. A. Shad, and W. Nisar. A robust missing data-recovering technique for mobility data mining. *Applied Artificial Intelligence*, 31(5-6):425–438, 2017. Cited By :5.
- [ZZM⁺11] Y. Zheng, L. Zhang, Z. Ma, X. Xie, and W. . Ma. Recommending friends and locations based on individual location history. *ACM Transactions on the Web*, 5(1), 2011. Cited By :366.
- [ZZXM09] Y. Zheng, L. Zhang, X. Xie, and W. . Ma. Mining interesting locations and travel sequences from gps trajectories. In *WWW'09 - Proceedings of the 18th International World Wide Web Conference*, pages 791–800, 2009. Cited By :1324.
- [ZZYS13] K. Zheng, Y. Zheng, N. J. Yuan, and S. Shang. On discovery of gathering patterns from trajectories. In *Proceedings - International Conference on Data Engineering*, pages 242–253, 2013. Cited By :173.

