MAXCT

ENTREGABLE R2

"DISEÑO TÉCNICO DEL SISTEMA: HITUL"

27 de Marzo, 2015

F.E.D.E.R.

JUNTA DE ANDALUCIA
CONSEJERÍA DE FOMENTO Y VIVIENDA

UNIVERSIDAD DE MÁLAGA

NEO

Movilidad Inteligente: Wifi, Rutas y Contaminación
Proyecto I+D+i Ene-Oct, 2015. N° GGI3003IDII. OTRI-UMA # 8.06/5.47.4356.

Contenido

1. Introducción y descripción

HITUL es el nombre del **paquete software** entregado en este proyecto que se encarga de **la planificación óptima de los semáforos** a nivel de ciudad. Esta herramienta tiene como objetivo apoyar las decisiones de un agente técnico de tráfico que trabaja en el **centro de control de tráfico de la ciudad**. Esta ayuda que proporciona nuestro sistema se basa en el análisis de los **mapas** y la **historia** previa del tráfico en la ciudad, así como en la inclusión de **conocimiento experto** en él y la posibilidad de tomar en consideración cualquier otra información **dinámica** que llegue al paquete.

HITUL es el acrónimo de "**Holistic Intelligence for Traffic Urban Lights**". En este sistema se tiene la intención de generar automáticamente **ciclos óptimos de programas** de semáforos. La planificación óptima de los semáforos, y en particular los programas óptimos de los ciclos de luces semafóricas, es una tarea crucial en ciudades actuales, lo que lleva a beneficios potenciales en términos de consumo de energía, gestión del flujo de tráfico, seguridad peatonal y cuestiones ambientales. A diferencia de las semáforos descentralizados auto-regulados, o los sistemas tradicionales basados en cola, el uso de algoritmos **metaheurísticos** [BLS13] [GK03] para encontrar ciclos de semáforos apropiados es todavía una cuestión abierta. La capacidad de las metaheurísticas y **técnicas inspiradas en la naturaleza** [AB+09] [LMH11] para resolver problemas muy grandes con muchas restricciones y en tiempo de ejecución competitivo los hacen únicos para construir HITUL.

HITUL será desarrollado por el grupo de la UMA que ya dispone de una amplia experiencia científica en la resolución de problemas complejos y en la realización de trabajos sobre **movilidad inteligente** [GOA13] [TGA12] [SA14]. Este sistema se dirige **a toda la ciudad de Málaga** como un primer paso para mostrar la potencia de las soluciones que es capaz de generar. Por lo tanto, **la interacción** con los gestores de tráfico en esta ciudad es muy importante, suponiendo mucho tiempo en nuestra carga de trabajo semanal. Se está interactuando con ellos para una mejor definición de las prioridades en las ciudades modernas para conocer los requisitos del paquete software, y obtener información real de la ciudad para probar nuestros **prototipos** sobre escenarios realistas. Si bien esta colaboración UMA-ciudad ya está establecida (tras muchos meses de reuniones con los gestores municipales) será de carácter **dinámico**, lo que significa que la interacción con los representantes de la ciudad **dará forma y posiblemente cambiará** muchas de las decisiones tomadas en estos documentos intermedios afectando al prototipo final y estable que se desplegará al término del proyecto.

En este proyecto, nos centramos en una **extensa zona urbana** de la ciudad de Málaga (España) como caso de estudio. Este es un objetivo **muy difícil** que necesita grandes cantidades de información, grandes recursos computacionales para el análisis de la ciudad y profundos conocimientos tecnológicos para completar el trabajo. En resumen, esto equivale a necesidades constantes de lectura, viajes para el aprendizaje científico, pensar en los usuarios a la hora del desarrollo, una gestión precisa y una intensa difusión de los resultados. También se planea tomar estas decisiones para facilitar la **reproducción** de nuestro trabajo en otras ciudades españolas y extranjeras, ampliando la versatilidad del sistema.

Se utilizará el conocido simulador de tráfico urbano **SUMO** (Simulator of Urban Mobility) [BB+11][KE+12], que proporciona información constante y realista sobre flujo de vehículos (tales como velocidad, consumo de combustible, emisiones, tiempo de viaje, etc.), dando lugar a configuraciones de escenarios realistas de acuerdo con patrones concretos de movilidad. SUMO requiere información sobre el tráfico y las calles a ser estudiadas. La red topográfica manejada por SUMO contiene información sobre la estructura de los mapas: nodos y aristas. La red debe ser lo más fiable posible para obtener una **simulación realista** para analizar. Esta es una diferencia fundamental de nuestro proyecto con otros: no sólo la gran **dimensión** de las zonas consideradas (los trabajos previos no tratan más allá de una calle o una esquina mientras nosotros planeamos abordar una ciudad completa), sino también por el nivel de **detalle** y lo realista de nuestros estudios (mapas reales, uso de las reglas de conducción, análisis individualizados de vehículos y perfiles de conductores, tiempos, contaminación, ...).

Este documento describe las **funcionalidades del sistema HITUL** y la mayoría de las cuestiones técnicas pertinentes para su diseño. Este informe servirá también como guía y lista de verificación durante la vida del proyecto, aunque probablemente sufrirá algunos cambios a medida que avancemos en el proyecto e interaccionemos con los gestores de tráfico.

1.1 Partes y contenidos del entregable

Este documento se compone de seis partes (además de la presente introducción):

- **Perspectiva del sistema:** en esta sección se definen los principales objetivos de HITUL a alto nivel, dando una visión global del sistema propuesto.
- **Funcionalidades del sistema**: en esta parte, se hace una descripción detallada de todas las funcionalidades que se están desarrollando en el paquete de software HITUL.
- **Descripción de la arquitectura**: aquí, se detallarán los diferentes componentes que se planean utilizar para el diseño y la ejecución real de HITUL.
- **Resumen de tecnologías:** esta sección describe la infraestructura interna usada en la UMA para el desarrollo de nuestro producto.
- **Modelo de datos conceptual**: esta parte define las principales colecciones de datos de este sistema y las relaciones entre ellas.
- **Prototipo del sistema**: por último, se mostrará un primer prototipo conceptual de la interfaz de la aplicación y algunos casos de uso comunes.

1.2 Relaciones con los otros hitos del proyecto

Este informe está relacionado con otros documentos de la siguiente manera (véase también la Tabla 1):

- ***"R3: Guía de entregables"***: este documento describe las principales aportaciones de cada informe, incluyendo al actual.

- ***"R5: Gestión y tareas de HITUL"***: en este informe se definen las tareas y otras acciones adicionales necesarias para el desarrollo del sistema HITUL.
- ***"R6: Competidores y estado del arte"***: en este trabajo se incluye una descripción detallada y las características de herramientas y proyectos similares al propuesto.
- ***"R7: Ventajas sobre los competidores"****:* en este informe se muestran las características incluidas en nuestro sistema HITUL que no han sido consideradas por los competidores y porqué son interesantes.
- ***"R8: Análisis del hardware necesario"****:* este documento estudiará los requisitos de la plataforma hardware necesaria para desarrollar y desplegar el sistema HITUL.
- ***"R9: Análisis del software necesario"****:* este informe analizará la plataforma software interna utilizada para desarrollar y desplegar el sistema descrito en esta propuesta.
- ***"R10: Informe Final"****:* este informe incluirá una descripción completa del sistema HITUL implementado y nuestras experiencias probando este sistema.
- ***"S1: Prototipo #1 HITUL"****:* este es el primero de dos prototipos que serán implementados del sistema descrito en este documento.
- ***"S2: Prototipo #2 HITUL"****:* de forma similar a la anterior, esta entrega representa el segundo (y último) prototipo de sistema HITUL en el que se llevarán a cabo todas las funciones requeridas que se detallan en el documento actual.

Tabla 1. ***Entregables relacionados con este (R2)****.*

R1. Diseño técnico del sistema: CTPATH	**-**	R7. Ventajas sobre los competidores	**X**
R2. Diseño técnico del sistema: HITUL	**-**	S3. Prototipo #1 CTPATH	**-**
R3. Guía de entregables	**X**	R8. Análisis del hardware necesario	**X**
R4. Gestión y tareas de CTPATH	**-**	R9. Análisis del software necesario	**X**
R5. Gestión y tareas de HITUL	**X**	S2. Prototipo #2 HITUL	**X**
S1. Prototipo #1 HITUL	**X**	S4. Prototipo #2 CTPATH	**-**
R6. Competidores y estado del arte	**X**	R10. Informe Final	**X**

1.3 Acrónimos y abreviaturas

Aquí, se muestra (en la Tabla 2) los principales acrónimos y abreviaturas utilizados en este documento y su significado en el dominio HITUL. El objetivo es facilitar la lectura de este informe.

Tabla 2. *Acrónimos con sus descripciones.*

Acrónimo	**Descripción**
EA	Algoritmo Evolutivo (*Evolutionary Algorithm*)

DBMS	Sistema Gestor de Base de Datos (*Database Management System*)
DE	Evolución Diferencial (*Differential Evolution*)
GA	Algoritmo Genético (*Genetic Algorithm*)
GUI	Interfaz Gráfico de Usuario (*Graphic User Interface*)
HITUL	Holistic Intelligence Traffic Urban Lights
MCDM	Toma de Decisión Multi-Criterio (*Multi-Criteria Decision Making*)
MOEA/D	Algoritmo Evolutivo Multi-objetivo basado en Descomposición (*Multi-objective Evolutionary Algorithm Based on Decomposition*)
NSGA-II	Algoritmo Genético basado en Clasificación No-Dominada II (*Non-dominated Sorting Genetic Algorithm-II*)
OSM	OpenStreetMap
PSO	Optimización basada en Cúmulo de Partículas (*Particle Swarm Optimization*)
SA	Enfriamiento Simulado (*Simulated Annealing*)
SCM	Sistema de Gestión de Código Fuente (*Source Code Management system*)
SUMO	Simulator of Urban Mobility
TCC	Centro de Control de Tráfico (*Traffic Control Center*)
TL	Semáforo (*Traffic Light*)
TLJ	Intersección semafórica (*Traffic Light Junction*)
TLP	Plan Semafórico (*Traffic Light Plan*)
XML	Lenguaje de Marcas Extensible (*eXtensible Markup Language*)
XSD	Documentos esquema XML (*XML Schema Definitions*)

2. Perspectiva del sistema

Este sistema está pensado como una **herramienta de apoyo para la toma de decisiones** respecto a la planificación de la red de semáforos de la ciudad, una de las maneras para lidiar con la congestión y los atascos de tráfico. HITUL es una aplicación preparada para usar potentes recursos de computación (por ejemplo, redes, servidores, aplicaciones...) y diseñada con técnicas de optimización y simulación robustas (como metaheurísticas mono- y

multi-objetivo, SUMO, etc.) para apoyar a los gestores de control de tráfico ante preguntas similares a las siguientes:

- ¿Cómo de buena es la planificación de los ciclos semafóricos que estamos utilizando para las condiciones actuales del tráfico?
- ¿Cómo mejorar el flujo de tráfico en un área específica que es propensa a los atascos?
- ¿Cuál es el efecto de la configuración manual de un solo semáforo en la red de carreteras en general?

Esto sólo es posible debido al **tratamiento numérico de los datos** y la flexibilidad proporcionada por el amplio conjunto de **funcionalidades** de HITUL, que permiten llevar a cabo la búsqueda de TLPs óptimos de acuerdo a diferentes criterios y para diferentes condiciones en la red vial. La búsqueda de TLPs óptimos se puede adaptar para mejorar la comodidad de los viajeros, la sostenibilidad de la ciudad y la implementación de políticas específicas de control de tráfico. La Figura 1 ilustra esta relación entre nuestro sistema HITUL propuesto y los componentes externos o actores en el sistema.

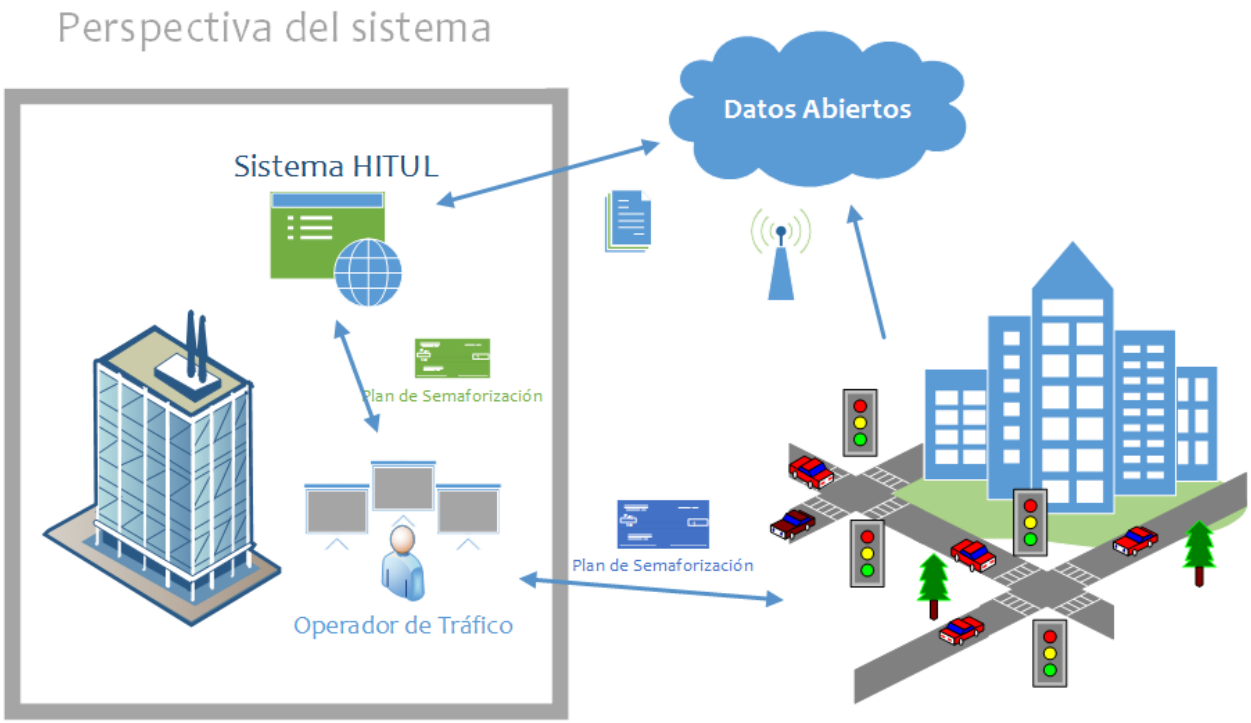

Figura 1. ***Perspectiva del Sistema HITUL***.

En el contexto de un área metropolitana, el sistema HITUL se aprovecha de los datos abiertos de tráfico y servicios de información para proporcionar su funcionalidad. Esas colecciones de datos están disponibles para todo el mundo sin restricciones de Copyright o derechos de licencia. Un ejemplo es **OpenStreetMap** [HW08], un repositorio libre de información geográfica que cuenta con el apoyo de más de 1,6 millones de usuarios en todo el

mundo, que contribuyen rellenando formularios manualmente. HITUL utiliza también información proporcionada **públicamente** por el centro de control de tráfico (TCC) de la ciudad, como, por ejemplo, la intensidad del tráfico de lugares específicos de la ciudad. Por último, también se podría usar la **colaboración abierta** (*crowd-sourcing*) para la recolección de datos de tráfico en tiempo real a través de teléfonos inteligentes. Dicha información será capturada por una aplicación diferente llamada CTPATH que también será desarrollada en este proyecto.

El sistema HITUL no sólo consume sino que también contribuye a la iniciativa **Open Data**, lo que permite que otras comunidades ciudadanas o grupos de desarrolladores colaboren en proyectos I+D+i similares. Con este objetivo, se emplearán herramientas automatizadas para transformar los datos generados por nuestro sistema en información significativa para ayudar en la gestión del tráfico en la ciudad. Por lo tanto, podríamos compartir (después del permiso adecuado y adecuación de los datos) los planes semafóricos resultantes, definiciones matemáticas del problema de optimización, etc. Las actividades de realización de este proceso se describen con más detalle en la Sección 4.2.

En resumen, el oficial de control de tráfico **tendrá un sistema asistido por ordenador potente** para tomar decisiones informadas sobre los efectos de la planificación de los semáforos seleccionados. Estos planes podrían ser completados por el TCC o por nuevos planes calculados por HITUL. La capacidad de ofrecer planes semafóricos optimizados, planificaciones alternativas o permitir un análisis comparativo con los planes actuales utilizados por el TCC y estudiar sus efectos, son algunos de los elementos que conforman a nuestro sistema con una interesante herramienta. En la siguiente sección, veremos una descripción de las funcionalidades de HITUL.

3. Funcionalidad del sistema

En esta sección se definen las principales funcionalidades del sistema propuesto que van desde la creación TLPs óptimos de acuerdo a diferentes criterios y aplicables a diferentes condiciones de la ciudad, hasta la visualización y exportación de la información generada, así como la información directamente analizada a partir de los datos de tráfico real. La Figura 2 muestra las funciones previstas para el gestor del control de tráfico que toma decisiones con respecto a los problemas de tráfico diario.

Describimos cada una de ellas en las siguientes subsecciones, dando sus requisitos y sus principales características.

SF1. Minimizar el tiempo de espera de los vehículos

Esta función permite generar TLPs óptimos para **reducir el tiempo de viaje** de los conductores debido a la adecuada configuración de los ciclos en los semáforos en la ciudad. Esta es una característica de alta prioridad, ya que evita que los conductores perder tiempo y llegar tarde a su destino deseado.

Figura 2. *Funcionalidades del Sistema HITUL.*

Como efecto secundario, esto también **ayudará a reducir el consumo** y las emisiones de CO2, ya que reducirá el tiempo que el vehículo está detenido debido a que el semáforo esté en rojo. Esta función requiere que el operador seleccione un área y un perfil de tráfico produciendo un conjunto de planes fijos para las semáforos pertenecientes a ese área.

SF2. Minimizar el número de paradas

Esta funcionalidad permite generar TLPs óptimos para facilitar un **flujo de tráfico continuo** en la dirección principal. Minimizar el número de paradas se puede lograr de muchas maneras, algunas incluso contra intuitivas, y el objetivo de HITUL es proporcionar planes de minimización de las paradas de una manera **automática**. Estos planes podrían ser muy diferentes a lo que un ser humano pudiera aportar. Además, las **ondas verdes** son formas intuitivas bien conocidas de proporcionar flujos continuos de tráfico, así que vamos a analizar si aparecen en los planes de HITUL además de tratar de promoverlas en el caso de que no sean generadas automáticamente. HITUL puede hacerlo gracias al análisis detallado y amplio del tráfico de la ciudad a gran escala proporcionando planes más inteligentes al gestor del control de tráfico.

Esta importante característica es muy importante en otros aspectos del tráfico, ya que además de reducir el consumo energético y las emisiones producidas por las paradas, permite que el oficial de control de tráfico establezca **políticas de control complejas** en la red de avenidas, calles y carreteras.

Esta función requiere que se especifique los viales principales y direcciones con las cargas más pesadas para conseguir este tratamiento **preferencial**.

SF3. Reducir la emisión de gases

El sistema generará planes para los semáforos que ayuden a **reducir el impacto ecológico**. Las funcionalidades antes mencionadas **contribuyen indirectamente** a este objetivo mediante la reducción del número de paradas o tiempos de espera de los vehículos. Sin embargo, esta función se centra en la reducción de la intensidad de tráfico en esos viales donde el nivel de emisiones contaminantes es alto. Por lo tanto, el objetivo de reducir las emisiones de gases será **incluido explícitamente en el modelado matemático** del problema a ser resuelto por HITUL, lo que contrasta con tener ese beneficio como resultado del uso de otra meta para el modelado matemático (como el número de paradas, tiempos, etc.).

Esta funcionalidad será efectiva si la **información actual de emisiones** por áreas/caminos/calles está disponible. Estamos en contacto con el ayuntamiento de Málaga para obtener datos similares. Tenemos otras maneras de conseguirlo, como medir el número y tipo de vehículos que circulen por las calles u otros factores. Además, ya estamos planeando una medición (limitada) de estos aspectos en la ciudad, y estamos investigando diferentes plataformas hardware (Raspberry Pi o Arduino) que pueden proporcionar información sobre la contaminación a un bajo costo (gases, partículas, etc.).

SF4. Combinando Distintos Objetivos

Las características de HITUL incluyen la capacidad de generar planes óptimos basados en **diferentes objetivos al mismo tiempo**. Esta funcionalidad pretende buscar TLPs más estables de acuerdo a múltiples criterios, posiblemente en **conflicto**, en vez de satisfacer un único requisito. La optimización multi-objetivo y la toma de decisiones multi-criterio son cuestiones importantes y candentes en la investigación que aún no han sido explotadas en este tipo de aplicaciones. En este proyecto, se tiene la intención de utilizar muchas técnicas punteras en este campo, con el objetivo de tratar aspectos no tratados por los competidores.

Esta funcionalidad requiere seleccionar los objetivos que van a ser considerados en el proceso de optimización. El usuario entonces realizará un proceso de **toma de decisiones multi-criterio** (**MCDM**). Esa es una manera novedosa, reconocida y eficiente de tomar decisiones más inteligentes para la planificación de una ciudad, encontrándose también en muchos otros ámbitos de trabajo. Para esta función, nuestra experiencia en algoritmos

evolutivos multi-objetivo y en su aplicación a muchos problemas en el pasado será muy valiosa [FCA12][ML+11][TB+12].

SF5. Considerar diferentes perfiles de tráfico

El comportamiento del tráfico por carretera cambia según la hora del día, el día en sí mismo, y el momento del año. Nuestro sistema generará TLPs óptimos para adaptarse a diferentes **perfiles** de tráfico de la red considerándolos de forma explícita (otro resultado adicional de HITUL) y para extraer información de ellos para crear a su vez nuevos planes para los semáforos.

Esta funcionalidad requiere la selección de un perfil de tráfico de entre las **opciones disponibles en una lista**: día laborable, hora punta en un día laborable, sábado, domingo, hora punta del sábado, u horas de regreso a la ciudad el domingo. En general, HITUL considerará tanto escenarios habituales como aquellos excepcionales definidos por expertos de la ciudad de Málaga o por el uso de las clasificaciones internacionales clásicas de tipo de día u hora pico o por pura intuición y sentido común.

Si procedemos correctamente con el segundo objetivo de **monitorizar partes de la ciudad** (para esto el *Urban Lab* de Málaga será probablemente decisivo) para obtener la información realista y actual sobre el número de vehículos que circulan y la cantidad de tráfico, dicha información se compartirá libremente como **datos abiertos**. Entonces, como resultado, esta funcionalidad produce TLP óptimos de acuerdo con los perfiles de tráfico especificados y medidos dinámicamente.

SF6. Planes personalizados para zonas urbanas especiales

Esta funcionalidad permite seleccionar una **zona** (barrio, región, distrito, ...), en lugar de toda la ciudad, indicando que semáforos serán el objeto de la optimización. Las señales semafóricas del resto de la ciudad se configurarán de acuerdo con un TLP proporcionado por el gestor de tráfico del TCC o por nuestro propio sistema.

Esta es una característica muy práctica para el operador, ya que permite generar **TLP personalizados** de acuerdo a las políticas o requisitos de control de tráfico temporales. Por lo tanto, se requiere la especificación de una zona, además del criterio de optimización y el perfil de tráfico. La Figura 3 ilustra los distritos de Málaga, que será nuestra primera opción de división de la ciudad en diferentes regiones.

En nuestras reuniones, ya en curso, con la **TCC de la ciudad de Málaga**, llegamos al acuerdo de que sus expertos nos proporcionarían sus **preferencias** en los objetivos a optimizar, perfiles de tráfico y barrios de la ciudad por dónde empezar. Ya que planeamos hacer un software razonablemente general, vamos a incluir en HITUL una manera fácil de definir las diferentes regiones para que pueda utilizarse en otras ciudades (uso futuro de la herramienta).

Figura 3. *Los diez distritos Málaga.*

SF7. Generar mapas especializados

Esta funcionalidad permite **visualizar** la localización de los semáforos en la ciudad con la información sobre el plan óptimo propuesto. Esta representación visual facilita la selección y consulta de los TLPs para cada intersección de la ciudad.

Esta característica de HITUL no es secundaria, ya que la mayoría de los administradores del tráfico de la ciudad están buscando herramientas de visualización que les ayude a **tomar decisiones rápidas y prever los problemas**. Hay dos partes en esta visualización, una es parte de HITUL y la otra es uno de los valores añadidos de este proyecto. Por un lado, la interfaz de HITUL incorporará una capacidad de visualización limitada, para comprender y utilizar mejor los TLPs de esta herramienta. Por otro lado, podemos tomar los datos abiertos recogidos de la ciudad (y también los resultados de HITUL) y hacer una información visual más detallada que informe a los gestores municipales e incluso de otros usuarios (ciudadanos o empresas).

La generación de un mapa con información sobre TLP e informaciones relacionadas requiere una solución (un plan semafórico optimizado por HITUL) con el fin de **visualizar** la información correspondiente. Esta característica está muy relacionada con las principales funciones de nuestro sistema: la búsqueda de TLPs óptimos (SF1 a SF6) y la capacidad de

ofrecer un análisis de los resultados *online* (mientras se optimiza) y *offline* (después de la optimización). De nuevo, esto es una característica distintiva de otras herramientas ya que permite a los expertos **aprender** nuevas políticas para los semáforos procedentes de una solución generada por ordenador pero que es fácilmente comprensible para un ser humano, no un conjunto de símbolos, números o datos sin significado intuitivo.

SF8. Comparación con otros TLPs

El sistema permitirá comparar los planes generados con un TLP base proporcionado por un **experto** en un formato estándar (por ejemplo, XML). Los archivos XSD para definir planes semafóricos están disponibles al público en la página web de SUMO (http://sumo-sim.org/xsd/tllogic_file.xsd), lo que permitirá la interoperabilidad de nuestro sistema con otros planes proporcionados por otras fuentes (como el TCC) o los calculado previamente por HITUL, siempre que sigan el formato HITUL esperado para TLPs.

El sistema permitirá cargar el plan base y representará de forma visual las **principales diferencias** para que el oficial pueda identificar fácilmente los semáforos que más se diferencian con relación al plan base proporcionado.

SF9. Exportar planes de semáforos óptimos

El sistema facilitará la integración con otros sistemas TCC a través de la capacidad de exportar mediante el formato estándar XML. Esta funcionalidad requiere la selección de un TLP óptimo previamente generado para su exportación.

SF10. Analizar la información de tráfico

Esta aplicación software también **permitirá analizar** los datos generados durante el proceso de optimización para obtener datos enriquecidos.

- Semáforos críticos.
- Agrupación de TLs para mejorar la optimización.
- El análisis de correlación de los objetivos.

Esta función requiere un TLP óptimo previamente generado. Este tipo análisis de resultados hará uso de la representación mediante mapas especializados descrita en SF7.

4. Descripción de la arquitectura

En esta sección se ofrece un resumen técnico de nuestro sistema HITUL. Nuestra aplicación se estructura en **tres capas** que agrupan componentes mutuamente relacionados. En primer lugar, ***front-end*** que actúa como servidor para proporcionar la aplicación de una sola

página (SPA) que encarga de mostrar la interfaz para la optimización del tráfico y también las interfaces de acceso a los datos abiertos generados por nuestro paquete software.

Además, el sistema cuenta con un ***back-end,* un servidor numérico,** que ejecuta el procesamiento de datos y las herramientas de optimización en una plataforma de desarrollo similar a Java EE, ofreciendo un alto nivel de disponibilidad, fiabilidad y escalabilidad. Basándonos en nuestra experiencia en este campo, los mejores algoritmos con los que empezar a buscar soluciones optimizados son técnicas basadas en inteligencia colectiva y bio-inspiradas como PSO [Cle10], DE [Cha08], EA [GK03], y algunas versiones multi-objetivo basados en mecanismos de control de la diversidad (*crowding)* y probablemente mediante otros mecanismos adicionales como archivos de soluciones no dominadas cuya moto de optimización esté regulado por NSGA-II [DP+02] o MOEA/D [ZL07].

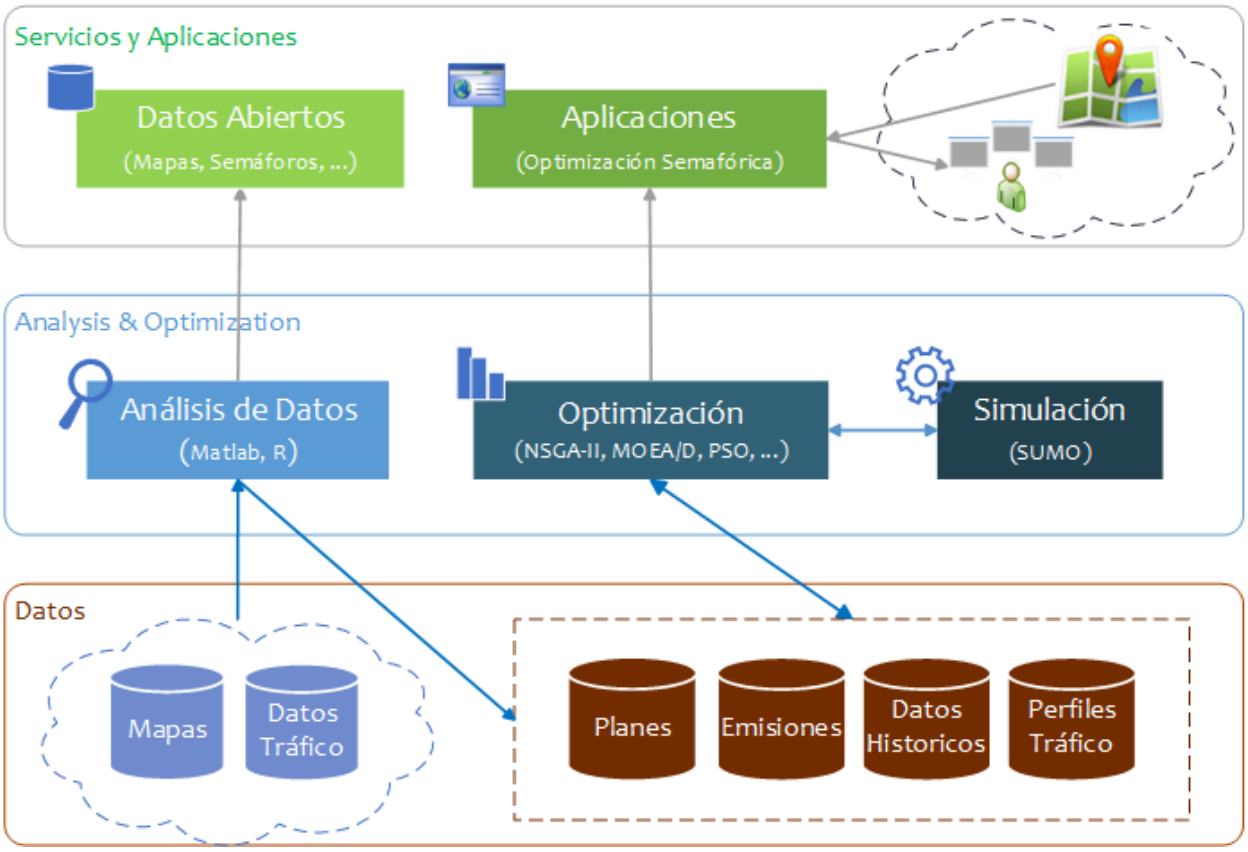

Figura 4. *Introducción a la arquitectura de sistema HITUL.*

Por último, nuestra aplicación comprende un tercer nivel, **un servidor de base de datos**, que contiene las colecciones de datos y las interfaces para manejarlas. Probablemente se utilice una base de datos relacional, ya que no tenemos razones pertinentes para usar un almacenamiento de datos y acceso no relacional.

La Figura 4 resume los principales componentes en cada capa, así como su interacción con otros componentes en nuestro paquete software. La arquitectura permitirá un **despliegue**

flexible del servicio HITUL. La utilización de un esquema cliente/servidor permite tanto el acceso remoto como un **acercamiento local**, donde los usuarios se encuentren en una sala y los servidores computacionales reales estén en un centro de computación o una sala fría, con los datos siempre circulando dentro de las instalaciones y respetando las medidas de seguridad estándar del TCC. Esto es altamente dependiente de la ciudad, y nuestras decisiones han sido tomadas para facilitar la aplicación en nuevos escenarios en diferentes ciudades.

En los siguientes apartados se describe detalladamente los componentes dentro de cada capa: **almacenamiento de datos (capa inferior), análisis/optimización (capa intermedia), y servicios/aplicaciones (capa superior).**

4.1 Almacenamiento de datos

Esta capa contiene todos los datos utilizados en nuestra aplicación. Es importante separar el almacenamiento de **datos** de la **manipulación** de los mismos y de los **servicios** a los usuarios finales. Sólo de esta manera el sistema de software permitirá ser actualizado con éxito, cambiándolo y ampliándolo en el futuro. La minimización del acoplamiento y la **abstracción de la implementación** son dos de las decisiones básicas que se están tomando aquí para un desarrollo de alta calidad en HITUL. Se pueden distinguir **dos conjuntos de datos**: (i) los datos **externos** compuestos por la información obtenida a partir de fuentes externas de este paquete software y (ii) la información procesada generada **dentro** de HITUL. En las siguientes subsecciones se describirán estas colecciones de datos, y ya en la parte final, se detallará nuestro sistema de base de datos. Este análisis se completa con el modelo de datos conceptual presentado en la sección 6.

4.1.1 Colecciones de datos externos

Nuestro objetivo final es conseguir un sistema que se pueda utilizar en un escenario real (Málaga) y por lo tanto un aspecto muy importante es el modelo utilizado por nuestro motor de optimización. Con el fin de obtener una representación precisa de la ciudad, nuestro primer objetivo es reunir información realista es mediante el uso de varias fuentes tales como:

- **Datos abiertos públicos***:* Málaga está comprometida con la iniciativa global que tiene como objetivo poner disponible para todos los ciudadanos y empresas el conjunto de datos e informaciones en poder de las administraciones. Entonces, un número bastante grande de colecciones de datos están disponibles al público, y se utilizarán como entrada para nuestro sistema. También tenemos planes para reunir información de **otras** webs y proyectos **de datos abiertos**. Algunos ejemplos de este tipo de datos, que nuestro sistema utilizará, son:

    - Mapas de Málaga de **OpenStreetMap** [HW08, OSM10].
    - Datos abiertos proporcionados en la web del Ayuntamiento de Málaga [Mal15a], como la localización de semáforos acústicos.

  - Intensidad del tráfico, disponible en el área de movilidad del Centro de Control de Tráfico de Málaga [TCC15].
  - Información de distritos del callejero de Málaga [Mal15b].

- **Datos de la ciudad TCC**: Los gestores de tráfico de Málaga han mostrado su interés en este proyecto, ya que puede ser muy útil para la ciudad y especialmente para el TCC. De hecho, apoyaron la propuesta de este proyecto, y nuestro equipo tiene reuniones periódicas con los miembros de dicho organismo municipal. Se plantean compartir información con nosotros complementando los datos públicos con algunos documentos específicos sobre las localizaciones de los TLs, información detallada de la intensidad de tráfico, etc.

- **Información recogida por CTPATH**: el sistema CTPATH tiene como objetivo trabajar como una aplicación móvil para los conductores. Esta plataforma también se utilizará para recopilar información sobre las rutas de los conductores y las condiciones de tráfico (como por ejemplo el tiempo para atravesar una calle). Esta información se puede utilizar para completar y corregir los datos históricos obtenidos de sitios web de datos abiertos y conseguir un escenario más realista, dinámico e interesante.

Todos los datos se dividen en dos colecciones principales:

- **Mapas**: esta colección incluye todos los datos relacionados con información estática sobre la ciudad, como calles, carriles, distritos o localizaciones TL.

- **Datos de tráfico**: esta colección incluye toda la información recogida sobre el tráfico, los patrones de conductores, rutas, etc.

4.1.2 Colecciones de datos internos

Nuestro sistema (en la capa intermedia) también genera un amplio conjunto de datos que pueden ser utilizados por los elementos de capa superior. Las principales colecciones de datos son:

- **Planes semafóricos**: este es el resultado más destacado de nuestro proyecto y contiene toda la información para configurar los TLs de acuerdo con algunos criterios. Estos datos son generados por el componente de optimización en la capa de Análisis y Optimización.

- **Tasas de emisión**: esta es una colección de datos sobre las emisiones de contaminantes como el CO2, NOx, CO, etc., que recogemos en la capa de Análisis y Optimización, cuando se simula el flujo de tráfico en la ciudad utilizando diferentes perfiles de tráfico. La tasa de emisiones es útil para proporcionar la ruta menos contaminante para un conductor (sistema CTPATH). Esta información puede calcularse basándose en el tráfico, cuanto más realista sea el tráfico mejores serán los índices

calculados. También podría venir de la (limitada) monitorización que planeamos hacer en la ciudad.

- **Datos históricos**: esto es el histórico de optimizaciones realizadas en diferentes mapas, perfiles de tráfico, estrategias de resolución con diferentes parámetros o configuraciones. Estos datos se analizan después para extraer información útil sobre el rendimiento de las técnicas de resolución utilizadas. Tenemos que crear mejores algoritmos (más "inteligentes"), y por lo tanto necesitamos retroalimentación sobre su comportamiento más un meta-nivel de re-diseño de técnicas.

- **Perfiles de tráfico**: en esta colección se almacenan los perfiles de flujo de tráfico identificados en el análisis de los datos de intensidad de tráfico y los datos recogidos por el sistema CTPATH.

4.1.3 Sistema de base de datos

Por último, esta parte describe el sistema de gestión usado para **controlar y manipular** todos los conjuntos de datos anteriores.

Nuestro plan inicial es usar una base de datos **MySQL** para almacenar todas las colecciones que se describen en las secciones anteriores de acuerdo con el modelo de datos presentado en la sección 6. Además, tenemos la intención de utilizar este sistema de gestión de base de datos (DBMS) para almacenar los datos de nuestra infraestructura software para el desarrollo. MySQL es una de las **bases de datos de código abierto** más utilizadas. Incluye el conjunto más completo de características avanzadas, herramientas de gestión y apoyo técnico para lograr los más altos niveles de escalabilidad, seguridad, confiabilidad y tiempo de actividad. Además, permite reduce el riesgo, el coste y la complejidad en el desarrollo, implementación y administración de aplicaciones críticas.

Después de un **análisis inicial** de esta base de datos se concluyó que cumple todos los requisitos de nuestro sistema. En el futuro planeamos para analizar el rendimiento de esta plataforma para discutir la posibilidad de migrar a otro DBMS (como PostgreSQL). Los resultados de este análisis serán reportados en el documento R9.

4.2 Análisis y optimización

El objetivo principal de esta capa es la manipulación de datos y la generación de nuevo conocimiento a partir del ya existente. Esta capa se compone de dos componentes principales: (i) el componente relacionado con el análisis y procesamiento de datos y (ii) el motor de optimización (resolutor).

4.2.1 Análisis de datos y procesamiento

En la sección anterior, se explicaron todos los conjuntos de datos utilizados en este sistema. Algunos de ellos serán generados por nuestro propio sistema y otros serán externos, obtenidos de diferentes sitios de datos abiertos o de socios del proyecto. Sin embargo, hay varios problemas con este tipo de datos externos: cada fuente utiliza una **representación** diferente para los datos (no hay formatos estándar que se utilicen en la mayoría de los casos), hay errores, datos incompletos, información desactualizada, ... Por lo tanto, para utilizar estos datos se necesitan algunas fases de pre-procesamiento y análisis. Algunas de las actividades realizadas por este componente de análisis de datos son:

- **Corregir** los datos**:** eliminando los errores y completando los datos.
- Convertir a **formatos** utilizados por nuestro sistema (XSD a disposición del público en la página web de SUMO).
- **Combinar** la información obtenida de diferentes fuentes.
- **Procesar los datos** de colecciones externas para generar nuevos datos procesados, tales como, perfiles diferenciados de tráfico (hora punta en días laborables, horario valle en días laborables, días no laborables, días festivos, ...).

Con el fin de completar las actividades anteriores, planeamos utilizar algunas herramientas especializadas, bibliotecas externas y desarrollos internos:

- Los **mapas** de **OSM** se convertirán al formato requerido por el sistema (archivos SUMO XML) utilizando la herramienta Convertir de SUMO. Más tarde, se necesitarán algunos ajustes adicionales tanto manuales como automáticos para corregir y actualizar los mapas con los datos de otras fuentes.

- Para la ubicación de los **TLs** en un mapa tendremos que usar algunas bibliotecas de programación adicionales (conversión de coordenadas) y desarrollar algunos programas internos (por lo general en Java o JavaScript para evitar problemas de compatibilidad).

- El análisis de los datos resultantes de nuestro componente de optimización se llevará a cabo con un amplio conjunto de herramientas (R, Matlab, Weka y desarrollos propios en Java) para obtener **conocimiento adicional** (flujos de tráfico realista, perfiles de tráfico, detección de los TLs más influyentes en la zona....).

Aunque el objetivo principal es desarrollar el paquete software HITUL, los resultados obtenidos por nuestro sistema, tales como los programas de ciclos optimizados de acuerdo a diferentes criterios, la mejora de la información resultante del tratamiento de los datos del mundo real, los flujos de tráfico previsto..., se pueden utilizar para extraer información adicional muy útil que nos permite ofrecer varios servicios adicionales para el ciudadano. Por ejemplo:

- Determinar la significancia estadística de la efectividad de los planes actuales.

- Detectar cuestiones ambientales/contextuales en los TLPs que afectan al tráfico.
- Analizar la robustez de los programas de ciclos propuestos.
- Ofrecer diferentes alternativas de visualización de los TLPs.

4.2.2 Motor de optimización

Este es uno de los componentes más importantes en nuestro sistema, ya que encapsula las técnicas inteligentes y los algoritmos utilizados para generar los TLPs óptimos de acuerdo con los diferentes criterios considerados.

La generación óptima de un TLP es un problema muy difícil desde muchos puntos de vista, lo que explica que no haya sido abordado o sólo parcialmente en el pasado:

- El **espacio de búsqueda** (espacio de posibles soluciones a este problema) es muy grande y aumenta de manera exponencial en función del número de TLJs. Por ejemplo, una intersección simple, con 3 TLS y 8 fases representa 85^8 soluciones posibles (más de $2{,}7 \times 10^{15}$ soluciones). Si tenemos en cuenta toda una zona o toda una ciudad con decenas o incluso centenares de TLJs, es imposible analizar todas las posibles soluciones tentativas y, por lo tanto, es necesaria una manera inteligente y eficaz der visitar el espacio de búsqueda para encontrar el TPL óptimo.

- La segunda cuestión principal es cómo evaluar la **calidad** de un TLP. Este es un factor clave para diseñar las técnicas apropiadas, ya que los métodos de optimización utilizan este valor para guiar la búsqueda. Desgraciadamente, para este problema no existe una formula cerrada para calcular la calidad de un TLP concreto, ya que la modelización de los flujos de tráfico en una ciudad es muy compleja. Entonces la utilización de **simuladores** son una herramienta habitual para calcular cómo los TLPs influyen en los flujos de tráfico. En HITUL planeamos utilizar SUMO [BB+11] para calcular la influencia de nuestras soluciones en las emisiones, las distancias de viaje, tiempos, ... Pero el punto débil de la utilización de este tipo de sistemas es que son unos procesos que **consumen mucho tiempo** (segundos o incluso unos pocos minutos por simulación). Además, estos sistemas también incluyen algunos factores estocásticos y entonces, deben realizarse varias simulaciones para obtener resultados significativos, lo que provoca tiempos de simulación aún más grandes. Esto no debe ser tomado como una desventaja, de hecho, son el único camino a seguir para un sistema tan complejo como una ciudad. En su lugar, tendremos que hacer frente a estas altas necesidades computacionales en el proyecto, y de ahí la necesidad de un potente **multiprocesador** para el procesamiento numérico y las **interacciones frecuentes** con otros investigadores para estudiar cómo reducir y lidiar con todos estos desafíos.

- Por último, el sistema propuesto debe generar diferentes TLPs atendiendo a diferentes **objetivos individuales** (funcionalidades SF1-3: los tiempos de espera, emisiones, ...), o **múltiples objetivos simultáneos** (funcionalidad SF4: objetivos combinados), y además con otros **criterios adicionales** (funcionalidades SF5-6: zonas, perfiles de tráfico...),

haciendo que el proceso de optimización sea aún más difícil. Aunque el problema a resolver es el mismo, cuando se cambia el objetivo de optimizar, el aspecto del espacio de búsqueda también cambia. Esto provoca que un método específico pueda obtener soluciones muy precisas para un objetivo, pero que las soluciones encontradas por el mismo método con un objetivo diferente pueden estar muy lejos del óptimo global. Por esto, se necesitan **métodos muy robustos**, así como una estrategia para seleccionar la técnica más adecuada para cada objetivo o conjunto de objetivos. De ahí el importante reto científico que supone este proyecto.

Para hacer frente a todas estas dificultades, en este componente optimización se incluyen:

- **Representaciones y operadores** eficientes que incluyen conocimiento del problema. Éstos se pueden integrar con las técnicas inteligentes y bio-inspiradas con las que ya hemos trabajado en artículos científicos previos de nuestro grupo. Esto nos permitirá reducir el espacio de búsqueda y explorarlo de una manera eficiente. No esperamos usar las mismas técnicas que construimos en el pasado, sino que se desarrollarán nuevas estrategias para HITUL.

- Una estrategia para **seleccionar** la técnica más adecuada para cada objetivo (o conjunto de objetivos). Esto es un problema abierto en investigación avanzada, que aquí sólo planeamos como método para aconsejar sobre las mejores técnicas basándonos en la experiencia conseguida durante el proyecto.

- **Algoritmos básicos**: en primer lugar, tenemos la intención de desarrollar técnicas básicas (búsqueda aleatoria, técnicas de seguimiento del gradiente, soluciones proporcionadas por SUMO, ...) que nos permitan obtener una solución base para comparar más adelante la exactitud de los métodos más avanzados.

- **Algoritmos avanzados**: con el fin de obtener resultados más precisos, se propone la utilización de metaheurísticas [BLS13] [GK03]. Este tipo de técnicas se han aplicado con éxito a un amplio conjunto de problemas del mundo real, en los que las técnicas clásicas y enumerativas no son capaces de obtener resultados. De acuerdo con las funciones descritas en la Sección 3, tenemos la intención de optimizar los TLPs no solo acuerdo con uno de los objetivos sino también de acuerdo a un conjunto de objetivos (posiblemente con conflictos entre ellos). Por lo tanto, vamos a utilizar técnicas de optimización tanto del campo mono-objetivo como del multi-objetivo. En los estudios iniciales planeamos utilizar los siguiente métodos:
    - Metaheurísticas **mono-objetivo**: GA [Gol06], SA [KV83], DE [Cha08], y PSO [Cle10].
    - Metaheurísticas **multi-objetivo**: NSGA-II [DP+02], MOEA/D [ZL07], y MOPSO [CL02].

El **motor de optimización** hace un **uso** intensivo de la **capa de almacenamiento de datos**, recuperando datos, como los mapas y los flujos de tráfico y almacenando los TLP

resultantes y toda la información relacionada con ellos (emisiones, tiempos, distancias...) en la base de datos MySQL. Este esquema permite, por lo tanto, que estos TLPs resultantes puedan ser procesados automáticamente por otros componentes de nuestros sistemas como el componente de Análisis o la capa de Servicio y Aplicación.

Para el desarrollo de este componente se utilizará **jMetal** [DN11], un *framework* basado en Java, orientado a objetos, para la optimización multi-objetivo con metaheurísticas. La utilización de este **framework** en conjunto con algunos **desarrollos adicionales** Java propios (algoritmos, representaciones, y operadores que no estén implementados en jMetal) acelerará el desarrollo de este componente.

La infraestructura software completa que se utiliza para el desarrollo de este componente se describe más adelante en la Sección 5.

4.3 Servicios y aplicaciones

En esta capa se incluye el ***front-end*** de la aplicación que servirá de interfaz entre el usuario final y nuestro sistema, ofreciendo una **interfaz gráfica** amigable al usuario de HITUL. En la Dección 7 mostraremos un prototipo inicial de diseño para el sistema HITUL. Esta interfaz gráfica de usuario debe involucrar la mayoría de las funcionalidades descritas en la Sección 3 (el resto se incluyó en otras aplicaciones secundarias). Para desarrollar este sistema de *front-end*, vamos a utilizar las tecnologías bien conocidas de desarrollo (véase la Sección 5). También estamos estudiando el uso de algunos servicios de **representación en mapas**, como la API de Google Maps [GI13] para visualizar los mapas e información contextual en ellos.

Además del sistema de HITUL, otro resultado importante generado por los componentes de la capa de Análisis y Optimización es la información corregida y mejorada y pensamos ponerla a disposición del público en la **página web del proyecto** (utilizando tecnologías web como HTML5 [WH+14], CSS [WC+11] o JavaScript [EI11]) o en plataformas abiertas como podría ser **FIWARE** [FG15]. Este objetivo también encaja en la iniciativa ***Open Data*** y está en consonancia con los objetivos de la ciudad de Málaga en el tema de poner abiertos la mayor cantidad de datos posibles.

5. Resumen de tecnologías

Esta sección es **altamente técnica** y muestra las decisiones adoptadas en el proyecto después de una fase inicial de estudio. En la Tabla 3 se resumen las principales tecnologías y paquetes software externos utilizados para desarrollar los componentes anteriores.

Tabla 3. *Descripción de la tecnología/software y su uso en HITUL.*

Nombre	**Descripción y su utilización dentro de HITUL**
Gestión de Datos	
FIWARE	Una plataforma europea que ofrece un potente conjunto de APIs (*Application Programming Interfaces*) que facilitan el desarrollo de aplicaciones inteligentes (aún bajo consideración para los servicios de cliente/servidor).
MySQL	El sistema más popular de código abierto para la gestión de base de datos SQL. Este es el gestor de bases de datos donde planteamos almacenar la información de entidades (ver modelo conceptual de datos en la Sección 6).
Tecnologías de desarrollo	
CSS	Un lenguaje de hojas de estilo usado para describir el aspecto y el formato de un documento escrito en el lenguaje de marcado HTML5 en nuestras aplicaciones.
Eclipse	Un entorno de desarrollo integrado (IDE) utilizado principalmente para desarrollos Java. Utilizamos esta herramienta para desarrollar el código fuente de nuestro sistema.
GIT	Un sistema de control de versiones distribuidas. Tenemos una rama principal del proyecto, donde la versión de desarrollo estable estará disponible. Todas las nuevas características añadidas a las aplicaciones se desarrollarán en una propia rama diferente y se fusionarán en la rama maestra sólo cuando sean probadas y nos aseguremos de su corrección.
Google API	Una interfaz de programación de aplicaciones Google que permite la comunicación con los servicios de Google. Se utilizará la API de Google cuando se tengan que hacer frente a los servicios de Google Maps en nuestras aplicaciones web.
HTML5	La nueva versión del lenguaje de marcado para la presentación de contenidos en la World Wide Web. Se usa como tecnología central para el desarrollo de sitios web y aplicaciones web.
Java	Un lenguaje general de programación orientado a objetos, concurrente, y multi-plataforma. La mayor parte del código fuente del proyecto será Java.
Javascript	Un lenguaje de programación utilizado en los navegadores web. Nuestras aplicaciones web utilizan scripts de Javascript para ejecutar algún código en el lado del cliente para interactuar con el usuario.
Jenkins	Una herramienta que proporciona servicios de integración continua para el

	desarrollo de software. Se usa Jenkins para construir y empaquetar el código fuente desarrollado.
jMetal	Un *framework* de Java orientado a objetos dirigido a facilitar el desarrollo de metaheurísticas para resolver problemas de optimización multiobjetivo. Se utiliza para resolver varios problemas de planificación de semáforos.
Maven	Una herramienta de automatización de construcción. Utilizamos esta herramienta para describir cómo se construye software, y sus dependencias.
SonarQube	Una plataforma abierta para gestionar la calidad del código. Todos los días Jenkins ejecuta un conjunto de pruebas de Sonar en el código fuente que desarrollamos. Esta herramienta mide la calidad del código.
SUMO	Una herramienta de simulación de tráfico, libre y abierta. Nos permite evaluar los TLPs antes de implementarlos en el entorno real de la ciudad.
Análisis de datos	
Weka	Una herramienta que contiene una colección de herramientas de visualización y algoritmos para el análisis de datos y el modelado predictivo.
Matlab	Un entorno de computación numérica para la manipulación de datos, funciones de trazado y datos, o la implementación de algoritmos y scripts.
R	Un entorno de software libre para computación y elementos estadísticos. Podríamos utilizar esta herramienta para analizar estadísticamente diferentes TLPs y algoritmos.
SPSS	Un paquete de software utilizado para el análisis estadístico. Es una alternativa privada a R. Podríamos utilizar esta herramienta para analizar estadísticamente y comparar entre diferentes TLPs.
Gestión de proyectos	
Trello	Una aplicación web gratuita basada en la gestión de proyectos que facilita la colaboración. Se utilizará para complementar nuestra metodología de desarrollo ágil.
Google Drive	Un almacenamiento de archivos y servicio de sincronización. Permite a los usuarios almacenar documentos en la nube, compartir archivos y editar documentos con los colaboradores. Nos permite compartir documentos y la edición en grupo.
Wiki	Una aplicación web, que permite realizar de forma colaborativa la modificación, ampliación o eliminación de contenido e información. Utilizamos este servicio para compartir información interna sobre el proyecto.

5.1 Una nueva infraestructura software para el desarrollo de aplicaciones

En esta sección se describe la infraestructura utilizada para el desarrollo de HITUL (que también se utilizará para el resto de desarrollos del proyecto). Debido a la complejidad de las aplicaciones de este proyecto tenemos que construir primero un entorno de trabajo que, por sí mismo, representa un desafío. Para llegar a un desarrollo satisfactorio y eficiente se requiere un profundo análisis de las muchas herramientas disponibles existentes para el desarrollo y su combinación con el hardware disponible. Esta sección está dedicada a las herramientas para el desarrollo. Mientras que el resto también son complejas y muy especializadas nos centramos en la parte de desarrollo, debido a su alto impacto en la calidad y eficiencia en los que estamos progresando de un hito a otro y lo que es muy crítico en un proyecto de una duración tan baja como este.

Esta infraestructura se compone de cuatro **herramientas principales**, que se describen brevemente en la Tabla 3: **Git**, **Maven**, **Jenkins** y **Sonar**. Estas herramientas son la columna vertebral de nuestra infraestructura de desarrollo de software y en esta sección se describen cómo las utilizamos para ayudar en el ciclo de vida de desarrollo de software (ver Figura 5). Estas no son las únicas herramientas que asisten a nuestro equipo de desarrollo, otras herramientas convenientes (como IDEs, componentes de seguimiento de incidencias, etc.) también se mencionarán en la siguiente subsección, sin embargo, las herramientas de la columna vertebral son las más importantes e interactúan entre sí para construir y supervisar el paquete software de HITUL.

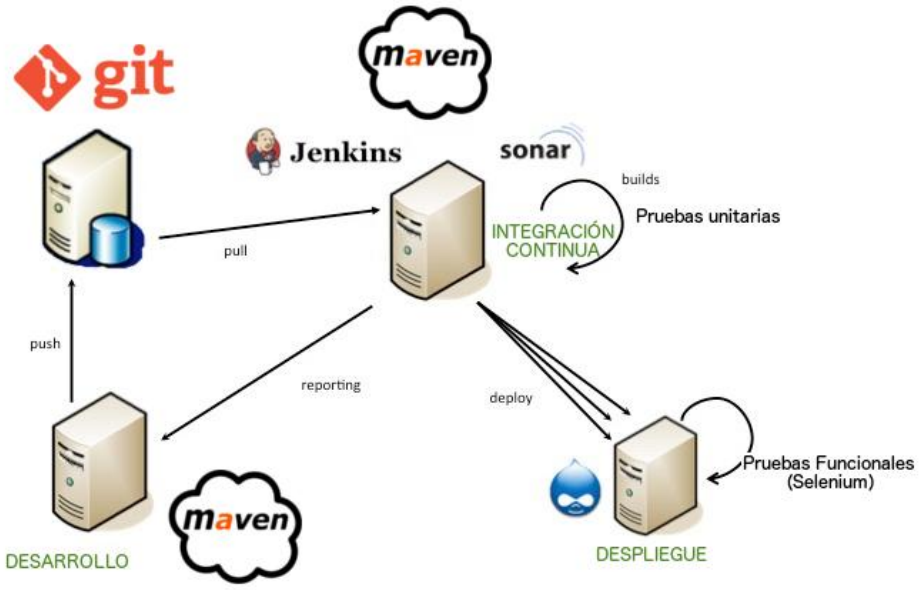

Figura 5. *La interacción entre las cuatro herramientas principales de nuestra infraestructura de desarrollo de software.*

5.1.1 Git

Git (http://git-scm.com) es una herramienta de gestión de código fuente (SCM) que realiza un seguimiento de todas las versiones del código fuente generado durante el desarrollo software. Una instantánea de todo el código fuente para el proyecto se llama **commit** en Git y se pueden identificar por su SHA-1. Conforme los programadores desarrollan el software, van añadiendo nuevos commits. Los commits se pueden organizar en **ramas**, que representan la cronología de desarrollo. Todos los commits y las ramas se almacenan en **repositorios Git**, que pueden considerarse como bases de datos de Git, donde se almacena toda la información sobre el proyecto software.

Git, a diferencia de SVN o CVS, es un SCM distribuido. Para cada proyecto de software existen varios repositorios Git, cada uno posiblemente en una máquina diferente y con su propia colección de commits y ramas. Los commits y ramas en un repositorio pueden ser "metidos" en un repositorio remoto (**operación push**) y "sacados" desde el repositorio remoto (**operación pull**).

5.1.1.1 Repositorios Git

Hay varias formas de organizar los repositorios Git (ver [CS14]), dependiendo de cuántos de ellos existen y cómo se gestionan. En nuestro caso, tenemos un equipo de desarrollo pequeño que necesita comunicar sus avances a los demás lo más rápidamente posible. Por esta razón, se utiliza un **flujo de trabajo centralizado**, donde cada desarrollador tiene su propio repositorio Git local para el proyecto, y todos ellos insertan sus cambios a un repositorio central (ver Figura 6).

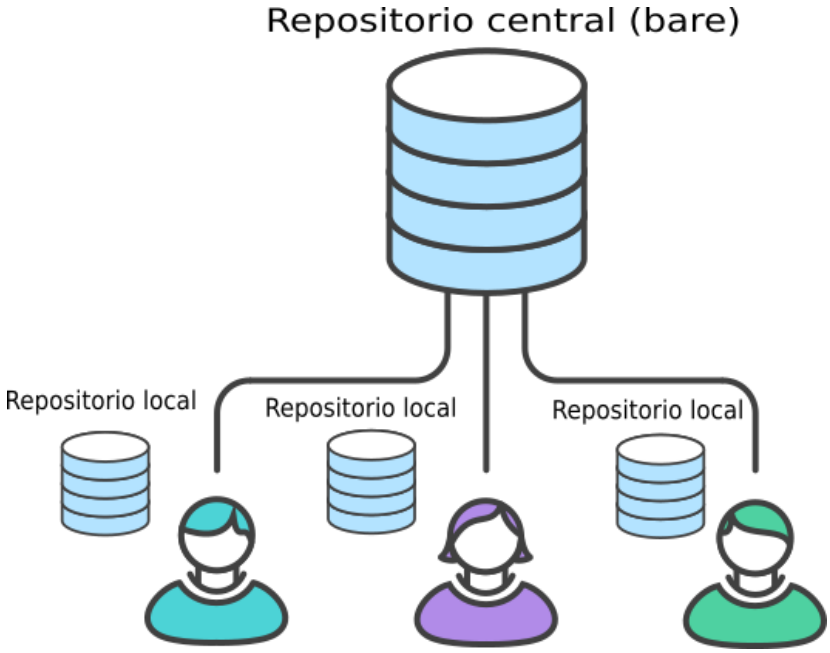

Figura 6. *Organización de repositorios Git en nuestro proyecto.*

Este repositorio **central** no tiene directorio de trabajo (es un repositorio vacío) y su único propósito es servir como eje central para que los desarrolladores se comuniquen entre sí. Los repositorios **locales** permiten a los desarrolladores trabajar con un repositorio git normal sin la necesidad de tener acceso a la máquina donde está el repositorio central, y con la posibilidad de revertir cualquier posible problema, causado por un comando git equivocado.

El repositorio central de git se desplegará en una máquina que se utilice sólo para el desarrollo. Todos los miembros del equipo de desarrollo tendrán acceso a esta máquina a través de ssh.

5.1.1.2 Ramas Git

Una de las características más destacadas de Git es su mecanismo ligero de ramificación. Este fácil esquema de ramificación fomenta el uso de la ramificación durante el desarrollo. Se han propuesto varios flujos de trabajo de ramificación (ver algunos de ellos en https://www.atlassian.com/git/tutorials/comparing-workflows/). En el desarrollo de HITUL se usará la denominada **rama característica de flujo de trabajo**. En ella se utilizará una **rama maestra** en Git para mantener la versión de desarrollo actualizada de HITUL.

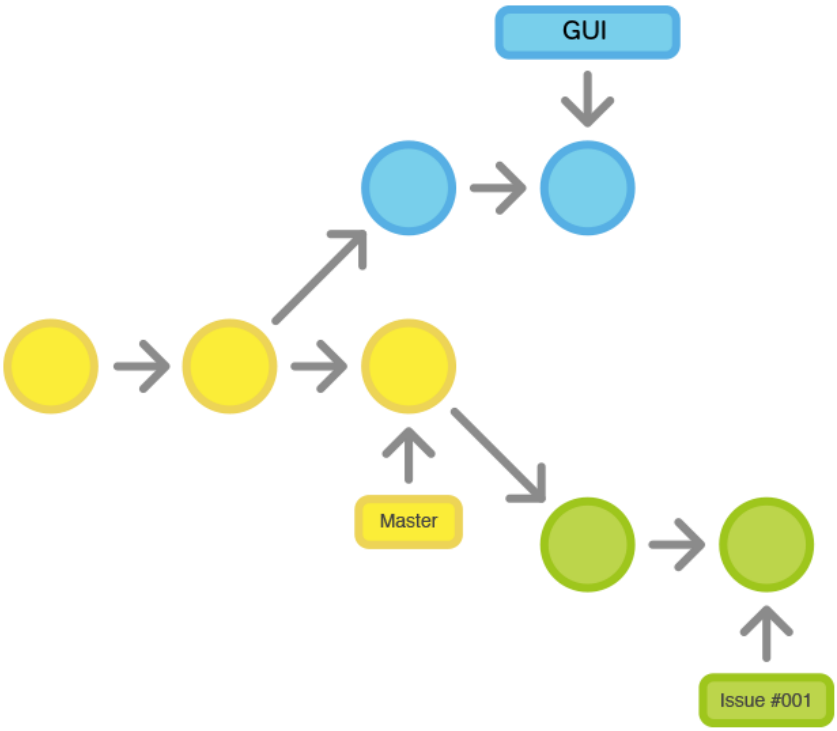

Figura 7. *Rama característica de flujo de trabajo en HITUL.*

El código en **la rama maestra** siempre será la instantánea estable y ejecutable del software. Cada vez que se añada una nueva característica al software, el desarrollador que la hizo creará una nueva rama partiendo del hilo principal actual y, a continuación, todo el desarrollo relacionado con esta función se llevará a cabo en esa rama. Cuando la función esté

probada adecuadamente, se fusionará con la rama maestra para producir una nueva versión de software estable que incluye la nueva característica. El desarrollador a cargo de la fusión comprobará primero la fusión en su repositorio local para luego insertar los commit en el repositorio central. De esta manera podemos mantener una **versión estable** en la rama principal del repositorio central.

Pero Git no sólo se utiliza para las nuevas características, las ramas también se utilizarán para **corregir errores** en el código. En este caso, una vez que se detecte un error, el desarrollador a cargo de arreglarlo creará una rama y preparará una solución en esa nueva rama. Cuando la solución esté lista y probada, se fusionará con la rama principal y, si todo está bien, será insertada al repositorio central.

Por último, también utilizaremos ramas para preparar los **prototipos** S1 y S2. Una vez que los prototipos estén listos, crearemos una rama para preparar y empaquetar este prototipo como versión liberada del desarrollo almacenado en la rama principal.

5.1.2 Maven

Maven (http://maven.apache.org) es una herramienta de software para ayudar a los desarrolladores en la construcción de software. Un software complejo con muchas dependencias de bibliotecas externas, como el que estamos desarrollando aquí, requiere también de un **proceso de construcción complejo**, que en muchos casos consta de los siguientes pasos: generar fuentes y recursos para la compilación, compilarlos, generar fuentes y recursos para las pruebas, compilar las pruebas, ejecutar las pruebas y analizar los resultados del informe, empaquetar el software, y finalmente liberar el software (incluso puede considerarse también el mantenimiento). Para cada paso, por lo general necesitaremos varias bibliotecas, por ejemplo, las dependencias del proyecto para compilar las fuentes, JUnit para las pruebas, etc.

Para gestionar toda esto, desde hace muchos años varias se han propuesta diferentes herramientas para la construcción de una configuración. **Ant** fue durante un tiempo el más popular en el mundo Java. Sin embargo, Ant no fue capaz de resolver el problema conocido como el "*Jar hell*": cuando nuestro software depende de varias bibliotecas que, la que a su vez dependen de otras bibliotecas, la gestión de archivos JAR es una tarea compleja y propensa a errores. **Maven** se desarrolló como herramienta para tener una gestión apropiada de las dependencias.

En un proyecto Maven hay un archivo clave, llamado **pom.xml**, que contiene toda la información relevante del proyecto Maven para poder construir el software con la versión adecuada de las bibliotecas. Entre otros datos, el archivo pom contiene las bibliotecas de las que nuestro proyecto depende, y la versión concreta necesitada. Cuando Maven intenta generar el proyecto, se descargará la biblioteca desde un repositorio central de Maven si es necesario. La biblioteca descargada se almacena en un repositorio Maven local para utilizarla en el futuro. De esta manera, Maven resuelve el "*Jar hell*".

En nuestro caso, vamos a usar Maven en todas las máquinas de los desarrolladores, así como en el servidor central para construir el software. Su función abarca tanto ejecutar las pruebas unitarias como empaquetar las aplicaciones.

5.1.3 Jenkins

Mientras Git y Maven pueden (y serán) utilizadas en las máquinas de los desarrolladores para realizar un seguimiento de la evolución del software y construcción del software, queremos tener un lugar **centralizado** donde todos los desarrolladores y los jefes de proyecto pueden echar un vistazo a la situación del software. Esto implica la construcción y el empaquetado de cada versión de software estable, las métricas de calidad en el código fuente, y desplegar la aplicación en un servidor para evaluar el software.

Jenkins (http://jenkins-ci.org) es una aplicación web que realiza automáticamente **tareas** relacionadas con el desarrollo de software. Estas tareas se pueden ejecutar de forma periódica o como consecuencia de un cambio en un repositorio del SMC. Se utilizará Jenkins para construir el software de la rama principal de nuestro repositorio Git central, calcular estadísticas y métricas sobre el código fuente, y finalmente desplegarlo (cuando se trata de una aplicación web) en la máquina del servidor para las pruebas del sistema, todo esto **sin intervención humana**. Los desarrolladores y jefes de proyecto pueden descargar los ejecutables del proyecto, comprobar la calidad del proyecto observando la cobertura de código obtenido por las pruebas unitarias, comprobar los resultados de las pruebas unitarias, y utilizar la aplicación para acceder a la máquina de desarrollo.

Figura 8. *Algunos ejemplos de tareas utilizadas en nuestro servidor Jenkins.*

Se definirán al menos dos tareas en Jenkins (ver Figura 8):

- Una de las tareas para la construcción del software de **la rama principal** del repositorio Git. Esta tarea se desencadena por un cambio en el repositorio Git y generará los artefactos ejecutables (Jar, War, Ear, dependiendo de la aplicación) y los informes de las pruebas.

- Una de las tareas para las **métricas de calidad del software** (la cobertura de código, detección de clones, reglas de estilo de código, etc.). Esta tarea se activará no más de una vez al día si se detecta un cambio en el repositorio Git. Las métricas de calidad se entregarán a nuestra instancia de SonarQube (discutido en la próxima sección).

5.1.4 SonarQube

SonarQube (http://www.sonarqube.org) es una aplicación web para controlar la **calidad** de proyectos software. Es un lugar central para los desarrolladores y administradores de proyectos que permite evaluar la calidad de los proyectos y tomar medidas para mejorarlo. Además de las medidas globales de calidad de software (ver Figura 9), SonarQube proporciona detalles sobre cada problema encontrado en el código fuente. El jefe de proyecto puede asignar cada uno a un desarrollador, quien tendrá la responsabilidad de solucionar el problema.

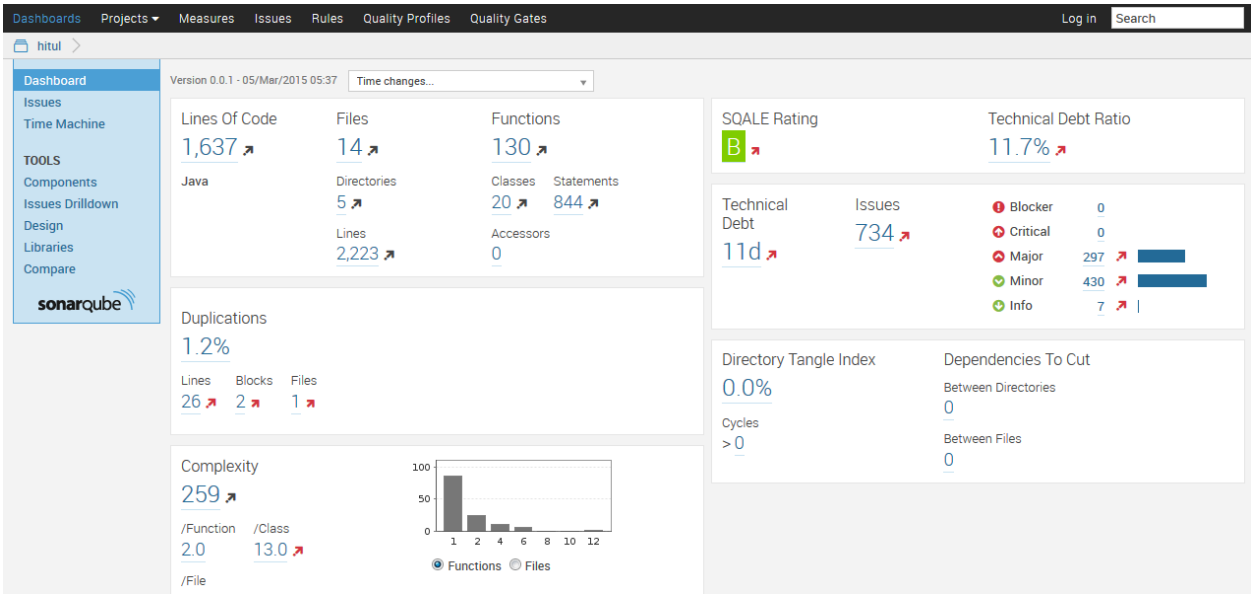

Figura 9. *Medidas globales para un proyecto en SonarQube.*

La información mostrada por Sonar viene de **tres** lugares diferentes. En primer lugar, durante la construcción de software, **Maven** ejecuta las pruebas unitarias y genera algunos informes con los resultados de estas pruebas. Estos informes son enviados a la instancia de SonarQube y son asociados al proyecto. En segundo lugar, los procesos de **Sonar** son programas Java que calculan algunas métricas en el código fuente. En nuestro caso, una de las tareas Jenkins relacionadas con la aplicación ejecuta un proceso Sonar que calcula estas métricas, incluyendo métricas de cobertura de código. En tercer lugar, cuando toda la información calculada por el proceso Sonar lanzado por Jenkins llega a la instancia de SonarQube, se ponen en marcha algunas tareas adicionales para calcular medidas adicionales. Todos los datos recogidos en cualquiera de estas tres formas se almacenan en una **base de datos** (MySQL en nuestro caso) asociada al proyecto.

SonarQube mantiene la información de toda la **historia** del proyecto. Esto permite que el director del proyecto mida la evolución de la calidad del software a lo largo del tiempo. Para facilitar este proceso, SonarQube muestra los cambios en relación a la última ejecución de las herramientas de análisis y destaca las tendencias utilizando las flechas (arriba o abajo).

5.1.5 Integrated Development Environments (IDEs)

Para el desarrollo del software se utilizarán varios IDEs en función de las necesidades personales de cada desarrollador. En particular, las IDEs que planean utilizar son: **Eclipse** (http://www.eclipse.org), **Netbeans** (https://netbeans.org), y **Xcode** (https://developer.apple.com/xcode/). Una característica común de todos ellos es que todos ellos soportan Git. Esta es de vital importancia en nuestro proyecto, ya que todo el software desarrollado se almacena en repositorios Git. Las herramientas de desarrollo Java (Eclipse y Netbeans) también soportan Maven.

5.1.6 Panel de tareas de Scrum

Vamos a seguir una metodología ágil de desarrollo de software. Vamos a utilizar una variante de **Scrum**. En Scrum, las funcionalidades del software están representados por las **historias de usuario** que se implementan de forma incremental en los **sprints**. Un sprint suele durar una o dos semanas, donde el equipo de desarrollo está enfocado a aplicar las historias de usuarios comprometidas. Al inicio de cada sprint las historias de usuario se priorizan y algunas de ellas son seleccionadas para ser implementados en el sprint. Las seleccionadas se descomponen en tareas de desarrollo. Las tareas están incluidas en el **sprint backlog**, un tablero de "por hacer" (TODO) donde los desarrolladores pueden seleccionar sus tareas favoritas que desean implementar. Cada tarea pasa por cuatro estados: por hacer, en desarrollo, en pruebas, y completada. Una vez que un desarrollador termina una tarea (lo incluye en la tabla de completadas), si selecciona una nueva desde el tablero de por hacer y repite el proceso hasta que ese tablero de tareas pendientes está vacío, y el sprint termina.

Debido a la naturaleza del equipo de investigación, donde algunos de los miembros tienen tareas docentes y de investigación, además de las tareas de este proyecto, la duración de los sprints serán flexibles y las **reuniones diarias de scrum** no serán realmente diarias, sino que se planea hacerlas cada dos o tres días.

Se usará Trello (www.trello.com) para almacenar el sprint backlog y todos los tableros de cada sprint. Con esta herramienta cada desarrollador pueden acceder fácilmente al estado de las tareas de desarrollo y participar en cada una (ver Figura 10).

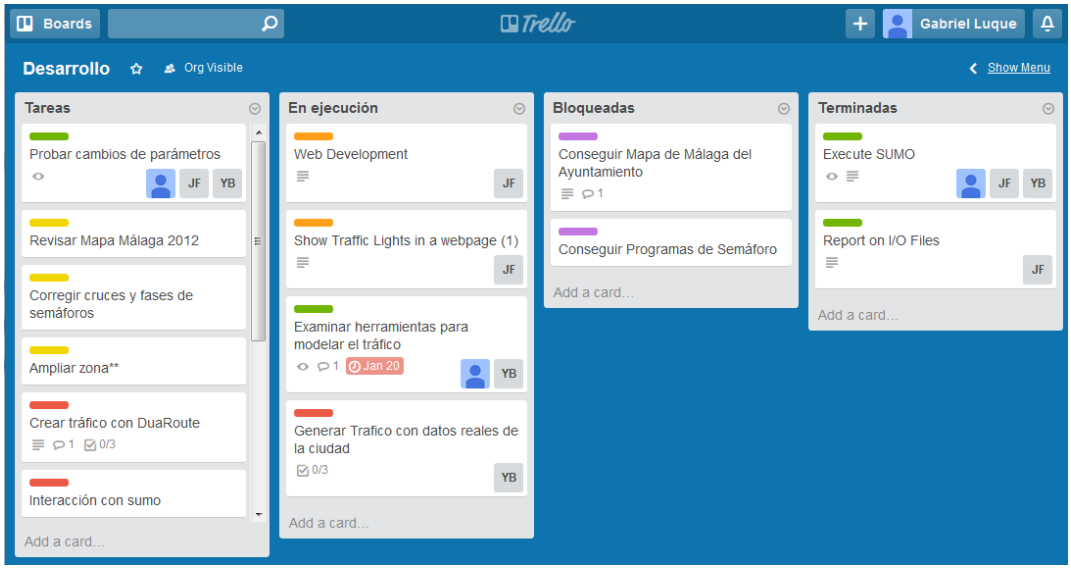

Figura 10. *Captura de pantalla del tablero de Trello para la aplicación de HITUL.*

6. Modelo de datos conceptual

Este **modelo de datos conceptual** identifica las relaciones a alto nivel entre las entidades pertinentes al sistema de HITUL. La idea de este modelo es ofrecer una forma sencilla de comunicar ideas a la amplia gama de partes interesadas, por lo tanto, no contiene ninguna información relacionada con una plataforma concreta. Este modelo cumple con los requisitos de todas las **funcionalidades** descritas en la Sección 3, sin embargo, podría sufrir modificaciones durante la vida del proyecto que sean beneficiosos para el diseño de la aplicación final. Estamos siguiendo una metodología ágil, iterativa e incremental para el desarrollo del sistema, por lo que es difícil tener un diagrama final de Entidad-Relación en esta temprana etapa del ciclo de vida de desarrollo del software. De todos modos, en la Figura 11 se muestra el modelo de datos conceptual que representa al sistema HITUL.

A continuación se describen brevemente las entidades del modelo de datos conceptual:

- **Planes Semafóricos**: Esta entidad representa un TLP, que es la entidad principal en el diagrama. Algunas características destacadas de esta entidad son el tiempo medio de espera, el número promedio de paradas en un viaje, la tasa de emisión, etc. Existe una relación directa e indirecta entre un plan de semáforo y todas las demás entidades. Un TLP está optimizado para un mapa en particular, un perfil de tráfico concreto, dando prioridad a uno o varios objetivos, y está relacionado con los tiempos de fase optimizadas de todos los semáforos que figuran en el mapa.

- **Mapa**: Esta entidad representa un mapa que está determinada por sus límites. El área que se optimiza por un TLP particular, se define por el mapa, así que se propone un TLP para todos los semáforos contenidos en los límites del mapa. Además, un mapa podría tener varios TLPs dependiendo de los objetivos seleccionados por el usuario.

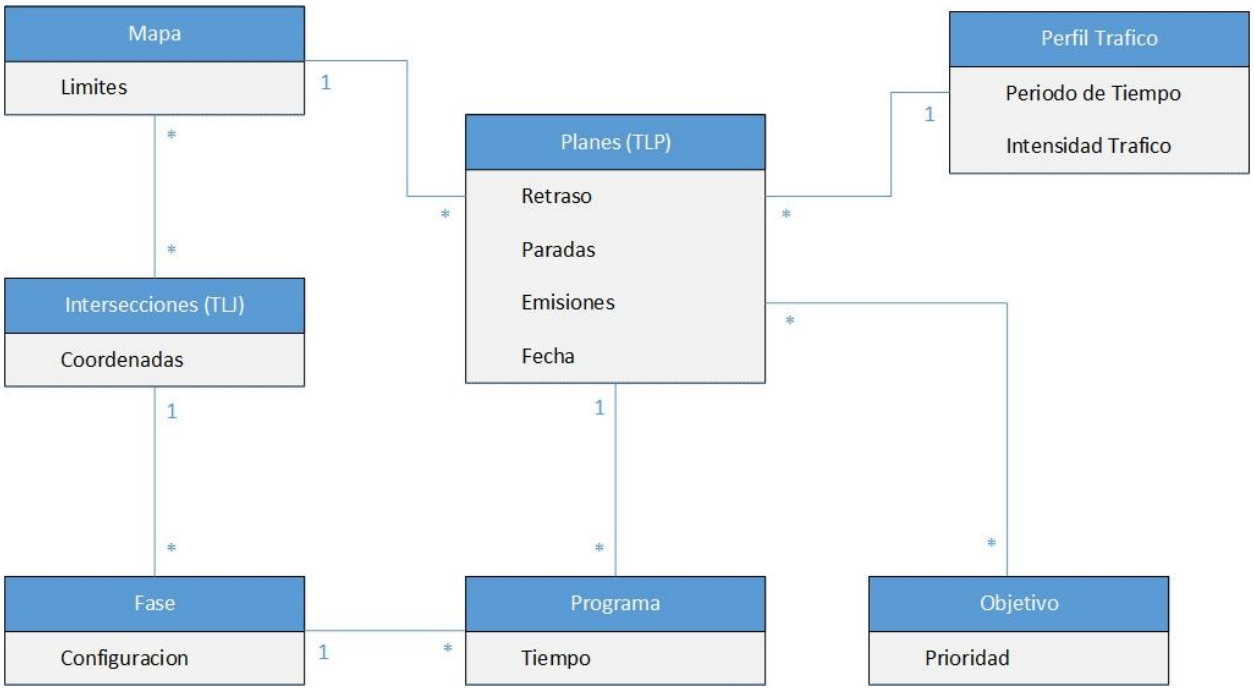

Figura 11. *Modelo de datos conceptual del sistema HITUL.*

- **Intersecciones**: Esta entidad representa una intersección regulada por semáforos (TLJ). En primer lugar, un TLJ se coloca en las coordenadas geográficas de su posición real en la ciudad, por lo que un semáforo podría estar contenido en varios mapas. Por ejemplo, un mapa de Málaga centro contiene un subconjunto de TLJs que también están en un mapa de toda la ciudad. Típicamente, un TLJ tiene diferentes semáforos, que generan varias configuraciones válidas llamadas fases.

- **Fase**: Esta entidad representa el estado de todos los semáforos que participan en un TLJ. Cada TLJ tiene varias fases, sin embargo una fase sólo pertenece a una TLJ particular. Un TLP establece el tiempo de la fase de todos los semáforos contenidos en el mapa.

- **Programa**: Esta entidad representa el período de tiempo que una fase estará activa en un TLP concreto. Un subconjunto de soluciones caracterizan un TLP en particular, siendo las soluciones relacionadas con una fase específica que pertenecen a un semáforo particular. Además, una fase se podría establecer a diferentes valores en diferentes TLPs.

- **Perfil de tráfico**: Esta entidad representa la intensidad real del flujo de tráfico en un período de tiempo, como días laborables, la hora punta en días laborables, sábados, domingos, etc. Un TLP está optimizado para un perfil de tráfico particular, es decir, el sistema HITUL establece los tiempos de fases del semáforo en función del flujo de tráfico definido por su perfil.

- **Objetivo**: Esta entidad representa el/los objetivo/s que el usuario del sistema quiera priorizar. Diferentes partes interesadas en el proyecto podrían definir diferentes objetivos como

reducir de las paradas y el tiempo de espera de los conductores, reducir los gases contaminantes emitidos, reducir el tiempo de viaje global, etc. Un TLP se genera de acuerdo con el objetivo (u objetivos) establecido por el usuario del sistema.

7. Prototipo HITUL

En esta sección se describen aspectos relacionados con la percepción del usuario y cómo las funciones y características descritas anteriormente se trasladan hacia una herramienta útil e intuitiva. La Figura 12 presenta una posible **interfaz** con los elementos más relevantes del sistema HITUL. Este prototipo considera todas las funciones descritas en la Sección 3, pero su objetivo es mostrar una interfaz gráfica para ayudar a entender este sistema y sus principales usos. Es probable que durante las próximas iteraciones de nuestro desarrollo de software incremental, sufra algunos cambios.

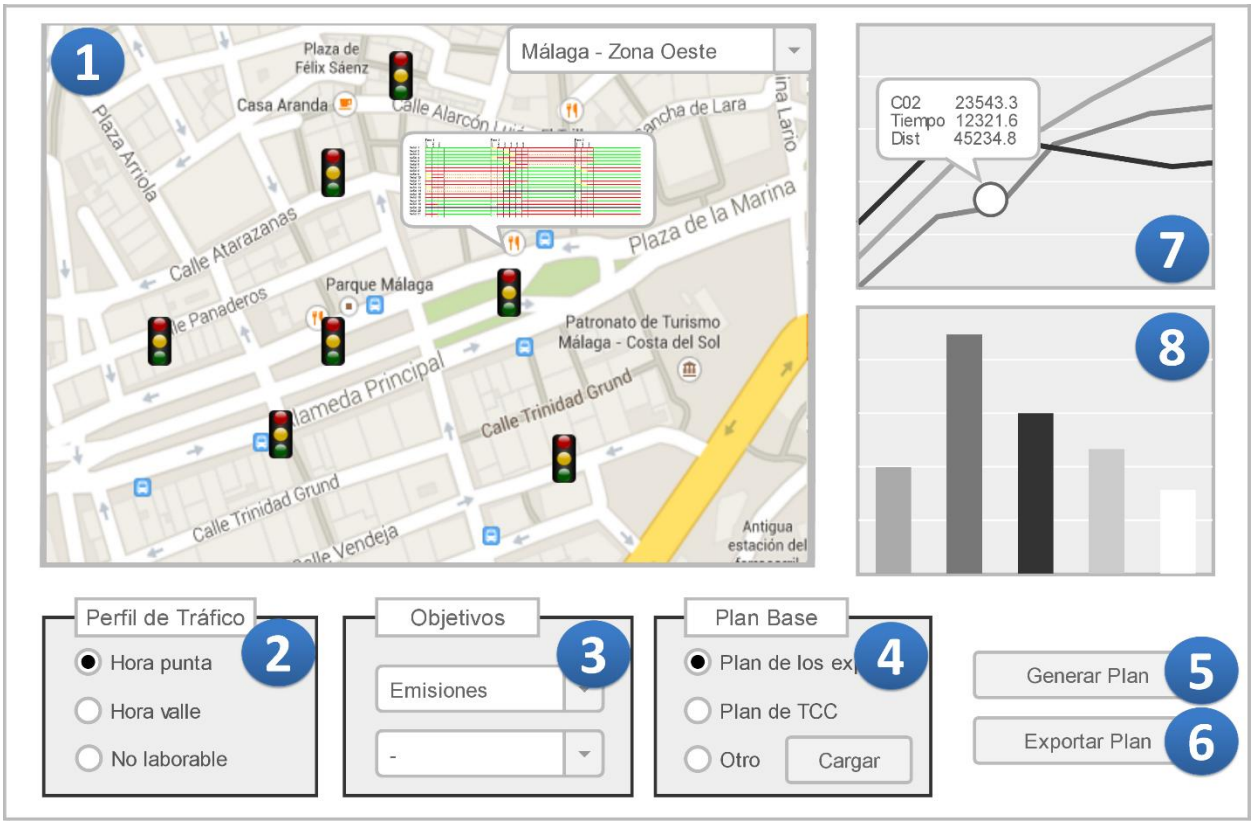

Figura 12. *Escritorio de diseño conceptual de HITUL.*

Podemos distinguir cuatro tipos de elementos:

- De configuración (elementos 1-4)
- De ejecución del sistema (elemento 5)
- De visualización de los resultados (elementos 1, 7, y 8)
- Para exportar los resultados (elemento 6)

El caso de uso más común de HITUL comienza en el elemento 1, donde el usuario final selecciona la **zona** a optimizar (funcionalidad SF6). En nuestro primer prototipo, tenemos la intención de incluir un menú desplegable para seleccionar el distrito (o la ciudad completa). En próxima versiones, también queremos considerar la posibilidad de seleccionar una zona (o incluso el TL) interaccionando directamente con el mapa.

Los próximos pasos incluyen los elementos 2, 3 y 4, donde se pueden seleccionar las diferentes **opciones**. El elemento 2 permite al usuario elegir el perfil de tráfico aplicado durante la optimización (funcionalidad SF5) mientras que el elemento 3 expone el objetivo (funcionalidades SF1-3) u objetivos (funcionalidad SF4) que serán optimizados. Finalmente, el elemento 4 define el TLP que se utiliza para como base para comparar respecto al nuevo generado (funcionalidad SF8).

Una vez que el usuario ha configurado el sistema con sus preferencias (también podría usar las opciones predefinidas), el proceso de optimización comienza cuando se presiona el botón "Generar Plan" (elemento 5). Cuando el motor de optimización está en funcionamiento, algunos elementos se actualizan:

- “Generar plan" (elemento 5) cambia su texto a "Parar optimización", para permitir al usuario detener la fase de optimización.

- El elemento 7 mostrará la curva de evolución de la calidad de los planes (si se ha seleccionado un único objetivo en el elemento 3) o el frente de Pareto generado (si se seleccionaron dos objetivos en el elemento 3).

.

El usuario puede seleccionar un TLP en particular (haciendo clic en un punto de la curva mostrada en el elemento 7) y la solución será visualizada en el mapa (elemento 1, funcionalidad SF7) y también será usada para la comparativa respecto la solución base seleccionada en el elemento 8 (funcionalidad SF8).

Por último, cuando el usuario detecta algún TLP interesante, puede exportarlo para usos futuros pulsando el botón "Exportar Plan" (elemento 6, funcionalidad SF9).

Este diseño conceptual hace hincapié en los usos más comunes de la herramienta y destaca sus elementos más sobresalientes. Además existirán las opciones habituales en las aplicaciones normales como abrir o importar un TLP, suspender/reanudar la búsqueda de nuevos planes, menú de inicio de sesión, el idioma, ayuda en línea, cierre de sesión, etc.

Referencias

[AB+09] Alba, E., Blum, C., Asasi, P., Leon, C., & Gomez, J. A. (Eds.). (2009). *Optimization techniques for solving complex problems* (Vol. 76). John Wiley & Sons.

[BB+11] Behrisch, M., Bieker, L., Erdmann, J., & Krajzewicz, D. (2011, October). Sumo-simulation of urban mobility-an overview. In *SIMUL 2011, The Third International Conference on Advances in System Simulation* (pp. 55-60).

[BLS13] Boussaïd, I., Lepagnot, J., & Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences*, *237*, 82-117.

[Cha08] Chakraborty, U.K., ed. (2008), *Advances in Differential Evolution*, Springer

[Cle10] Clerc, M. (2010). *Particle swarm optimization* (Vol. 93). John Wiley & Sons.

[CL02] Coello Coello, C. A., & Lechuga, M. S. (2002). MOPSO: A proposal for multiple objective particle swarm optimization. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on* (Vol. 2, pp. 1051-1056). IEEE.

[CS14] Scott Chacon and Ben Straub, Pro Git (chapter 5), 2nd ed., Apress 2014

[DN11] J.J. Durillo, A.J. Nebro *jMetal: a Java Framework for Multi-Objective Optimization*. Advances in Engineering Software 42 (2011) 760-771

[DP+02] Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. A. M. T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on*, *6*(2), 182-197.

[EI11] ECMA International (June 2011). *Standard ECMA-262: ECMAScript® Language Specification.* <*http://www.ecma-international.org/ecma-262/5.1/*>

[FCA12] J. Ferrer, J.F. Chicano, E. Alba, *Evolutionary Algorithms for the Multi-objective Test Data Generation Problem, Software*, Practice and Experience, 42(11):1331-1362, 2012

[FG15] FIWARE Group (2015). *FIWARE: Open APIs for Open Minds.* <http://www.fiware.org/>

[Gol06] Goldberg, D. E. (2006). *Genetic algorithms*. Pearson Education India.

[GI13] Google Inc. (2013) *Google Maps API.* <https://developers.google.com/maps/>

[GK03] Glover, F., & Kochenberger, G. A. (Eds.). (2003). *Handbook of metaheuristics*. Springer Science & Business Media.

[GOA13] Garcia-Nieto, J., Olivera, A. C., & Alba, E. (2013). Optimal cycle program of traffic lights with particle swarm optimization. *Evolutionary Computation, IEEE Transactions on*, *17*(6), 823-839.

[HW08] Haklay, Mordechai, and Patrick Weber. Openstreetmap: User-generated street maps. *Pervasive Computing*, IEEE 7.4 (2008): 12-18.

[KE+12] Krajzewicz, D., Erdmann, J., Behrisch, M., & Bieker, L. (2012). Recent development and applications of SUMO–simulation of urban mobility. *International Journal On Advances in Systems and Measurements*, *5* (3 and 4), 128-138.

[KV83] Kirkpatrick, S., & Vecchi, M. P. (1983). Optimization by simmulated annealing. *science*, *220*(4598), 671-680.

[LMH11] Lozano, M., Molina, D., & Herrera, F. (2011). Soft Computing: Special Issue on scalability of EAs and other metaheuristics for large-scale continuous optimization problems., vol. 15.

[Mal15a] Datos abiertos Ayto. Málaga. 9 Mar. 2015 <http://datosabiertos.malaga.eu/>

[Mal15b] Callejero Ayto. Málaga.. 20 Mar 2015 <http://callejero.malaga.eu/>

[ML+11] G. Molina, F. Luna, A. J. Nebro, E. Alba, *An Efficient Local Improvement Operator for the Multi-objective WSN Deployment Problem*, Engineering Optimization, 43(10):1115-1139, 2011

[OSM10] Frederik Ramm, Jochen Topf, Steve Chilton (2010) *OpenStreetMap: Using and Enhancing the Free Map of the World*, UIT Cambridge*, ISBN: 978-1-90686-011-0*

[SA14] Stolfi, D. H., & Alba, E. (2014). Red Swarm: Reducing travel times in smart cities by using bio-inspired algorithms. *Applied Soft Computing*, *24*, 181-195.

[TB+12] E-G Talbi, M. Basseur, A.J. Nebro, E. Alba, *Multi-objective Optimization Using Metaheuristics: Non-standard Algorithms*, International transactions in Operational Research, 19:283–305, 2012

[TCC15] Área de Movilidad Ayto. Málaga. 2013. 20 Mar. 2015 <http://movilidad.malaga.eu/>

[TGA12] Toutouh, J., García-Nieto, J., & Alba, E. (2012). Intelligent OLSR routing protocol optimization for VANETs. *Vehicular Technology, IEEE Transactions on*, *61*(4), 1884-1894.

[WC+11] W3C CSS Working Group (May 2011). *CSS 2010 Snapshot*. <http://www.w3.org/TR/css-2010/>

[WH+14] W3C HTML Working Group (28 October 2014). *HTML5 standard*. <http://www.w3.org/TR/html5/>

[ZL07] Zhang, Q., & Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *Evolutionary Computation, IEEE Transactions on*, *11*(6), 712-731.