MAXCT

ENTREGABLE R1

"DISEÑO TÉCNICO DEL SISTEMA: CTPATH"

11 de Mayo, 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 contexto

El proyecto en el que se enmarca este documento tiene la tarea principal de crear **dos aplicaciones**: un programa de apoyo para la **gestión semafórica** en una ciudad (HITUL) y un **planificador de rutas inteligente**, denominado CTPATH, que tenga en cuenta distancias y contaminación. La primera aplicación está orientada a ayudar al **centro de control de tráfico** sugiriendo ciclos de semáforos adaptados a la forma y tráfico de la ciudad, mientras que la segunda está orientada a los **ciudadanos**, para que el tráfico final resultante sea fluido y con conciencia ecológica.

En este documento describimos los componentes principales de **CTPATH**, el planificador de rutas inteligente, así como las funcionalidades de que dispondrá una vez completado. El objetivo final **no es hacer una aplicación comercial**, dado que se trata de un trabajo de investigación y las condiciones socio-económicas de la financiación del proyecto no están dirigidas a tal fin, sino más bien se trata de asentar las **bases científicas y tecnológicas** para que este tipo de aplicación sea útil en ciudades andaluzas como Málaga y sirva de ejemplo (su desarrollo, construcción y filosofía de uso) para inspirar futuros trabajos similares.

La **migración** hacia la ciudad [Uni13], el **progreso** económico y otros factores aumentan parque automotor, el número de desplazamientos y la probabilidad de sufrir un **atasco** [Tns11]. Estos son cada vez más frecuentes y se repiten normalmente en horarios similares durante el día (horas punta). Una de las consecuencias de la **congestión** de las vías dentro de la ciudad es que los tiempos de viaje aumentan considerablemente lo cual deriva en más estrés y ansiedad para los ciudadanos al ver que no llegan a tiempo a su destino. Otra consecuencia, que no es siempre percibida correctamente, es el aumento de las **emisiones** de gases de efecto invernadero [GLF13]. Cuanto más tiempo se encuentre un vehículo en un atasco debido a que la ruta que ha escogido no es la más adecuada, más gases tales como CO, CO_2, hidrocarburos y óxidos de nitrógeno se emitirán a la atmósfera. Los efectos a corto plazo de estas emisiones sobre la salud humana comprenden enfermedades del sistema respiratorio y cardiovascular [HJ+08], lo que no solo representa gastos médicos sino también **pérdidas** de días de trabajo. A largo plazo, producen un efecto invernadero que puede tener consecuencias para el clima global. Esta situación justifica el interés en planificadores inteligentes como CTPATH.

2. Descripción de CTPATH

CTPATH es un planificador de rutas **inteligente y holístico** para entornos urbanos.Es **inteligente** por el tipo de tecnología y algoritmos computacionales usados (inspirados en la naturaleza, entre otros) y es **holístico** porque analiza todo el contexto del viaje dinámicamente, no únicamente al usuario ni únicamente la ciudad, como otros competidores. Por tanto, su objetivo es proporcionar rutas a los conductores de vehículos en ciudad (el prototipo funcionará con mapas y datos de la **ciudad de Málaga** en concreto). Si bien ya existe software que proporciona rutas (google maps [GM15], navegadores GPS como Tom-Tom [TT15] o iGO [INS15]), hay varias diferencias clave en este caso, en el apartado de su inteligencia y en el apartado de la integralidad con la que evalúa dichas rutas atendiendo a varios criterios posiblemente contrapuestos. Algunas de las **características** de CTPATH que lo hacen único son las siguientes:

- Se proporciona como **código abierto**, comprensible, extensible y usable por otros investigadores, agencias o empresas en el futuro.
- Obtiene información de **sensores** repartidos por la ciudad que pueden conectarse de forma directa (3G) o indirecta (Wi-Fi) con un servidor central para enviar datos actualizados que permiten estimar el estado del tráfico.
- Es **sensible al estado de tráfico** de la ciudad, usando tanto información previa proporcionada por parte el área de movilidad del ayuntamiento como información recolectada a partir de los sensores situados en las calles y de los usuarios de la propia aplicación.
- CTPATH introduce conceptos de **gamificación** [DS+11][ZC11], mostrando un ranking de los usuarios más ecológicos o los que más usan la aplicación.
- Gracias a la información recogida de cada conductor durante el uso de la aplicación, podrá crear **perfiles de conducción** para cada usuario que permite mejorar las predicciones de contaminación producida y tiempo/distancia de tránsito de cada usuario particular, generando de esta forma rutas con información **específica** para cada usuario.

Estas características son muy desafiantes en el plano **científico**: permiten **conocer** la ciudad tanto offline como online (hacer modelos matemáticos de la ciudad en el futuro), requieren **inteligencia** en la determinación de rutas (problemas matemáticos e informáticos computacionalmente duros), manejan **objetivos contrapuestos** (toma de decisiones multi-criterio) y se basan en **tecnología moderna** aunque de bajo coste (smartphones, Raspberry Pi/Arduino, 3G, Wi-Fi, servidores multinúcleo con posibilidades de gamificación).

Asimismo, este trabajo es difícil porque requiere **interaccionar** con el conductor, así que debe ser visualmente atractivo y admitir la lectura de rutas para no provocar distracciones al volante. Debe tener una interfaz intuitiva al mismo tiempo que un núcleo interior muy sofisticado. La construcción de CTPATH además requiere de un **montaje tecnológico** en nuestros laboratorios considerable, con numerosas aplicaciones de ayuda para diseñar, implementar y testar el planificador antes de liberar el prototipo final. Asimismo, las constantes noticias y técnicas nuevas en el dominio exige que el equipo de trabajo se **actualice** con interacciones con otros investigadores internacionales en seminarios y conferencias para traer al proyecto ideas competitivas y actualizadas.

3. Funcionalidades de CTPATH

En esta sección se definen de forma pormenorizada las principales funcionalidades del sistema propuesto que ofrecerá a los usuarios rutas eficientes que minimizarán el tiempo de trayecto, la contaminación que se genera o la distancia que se recorre. Estas rutas serán calculadas por un sistema inteligente central y se podrán consultar a través de un dispositivo móvil o un ordenador de sobremesa. Los usuarios que dispongan de la aplicación en su dispositivo móvil podrán obtener una ruta personalizada calculada teniendo en cuenta su perfil de conducción. Además, se alimenta al sistema central con información del estado actual de las carreteras y del estilo de conducción del usuario. La **Figura 1** resume el sistema CTPATH, mostrando las funciones fundamentales del mismo, así como sus principales componentes.

Hemos dividido las funciones de CTPATH en **cinco grupos** distintos, según sean para (i) la aplicación móvil del **usuario** final, (ii) la aplicación móvil del **operario** de CTPATH, (iii) la aplicación **web**, (iv) los **sensores** o (v) el sistema **central**. Incluimos un identificador de funcionalidad y su explicación en las siguientes cinco secciones, una por grupo.

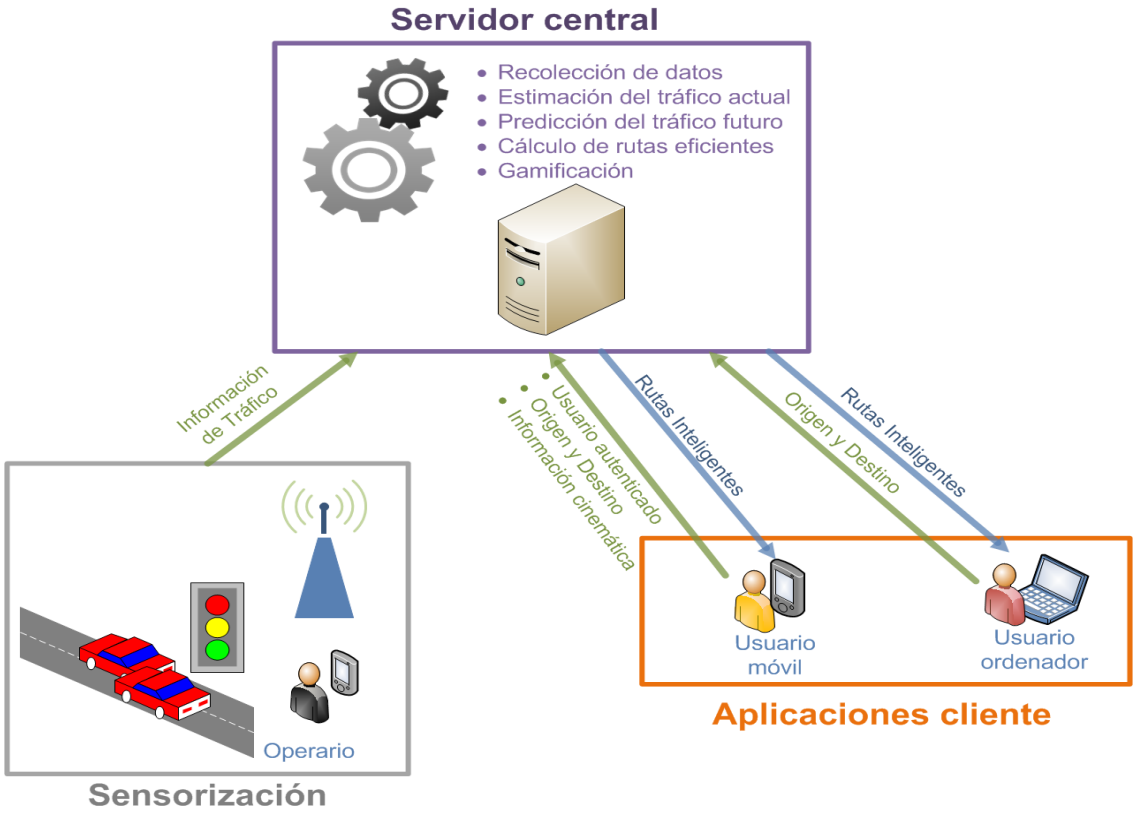

Figura 1. Diagrama de componentes de CTPATH

3.1 Funcionalidades de la aplicación móvil del usuario final

- **RFM001:** El usuario puede indicar en un mapa el punto de origen y destino de un trayecto que desea realizar en su vehículo. Opcionalmente puede indicar el tipo de vehículo (coche, en nuestro prototipo) que posee de entre varios mostrados. Los tipos de coches concretos que se definirán están por determinar y dependen de un análisis de los factores que caracterizan el modelo de contaminación que usaremos para los cálculos, así como de la hora de comienzo del trayecto. Con esta información, la aplicación, a la orden del usuario, solicitará a un servicio Web que calcule varias rutas alternativas que mostrará al usuario y dibujará en un mapa. Se mostrarán al menos tres rutas: la menos contaminante, la más corta en tiempo y una de duración y contaminación ponderadas.

- **RFM002:** Para cada ruta propuesta, la aplicación mostrará la contaminación estimada que produce el coche en dicho trayecto con indicación de la cantidad de CO, CO_2, NO_x, hidrocarburos y partículas emitidas durante el mismo.

- **RFM003:** El usuario de la aplicación deberá autenticarse en la misma con usuario y contraseña para obtener el servicio. Esto requiere gestión interna compleja e interacciones entre elementos de la aplicación.

- **RFM004:** La estimación para el trayecto del usuario se basará en información recogida mediante la aplicación móvil sobre el estilo de conducción de usuario, su coche habitual y el estado estimado del tráfico en la ciudad.

- **RFM005:** La aplicación móvil mostrará al usuario indicaciones para llegar a su destino e indicará su posición en el mapa gracias al GPS (si el dispositivo tiene uno y está activado).

- **RFM006:** Cuando el usuario utilice la aplicación del móvil en modo navegación, los datos recogidos por el móvil se transmitirán al servidor central de CTPATH, para hacer estimaciones futuras del estado del tráfico y del perfil de conducción del usuario.

- **RFM007:** La aplicación permitirá a sus usuarios crear una cuenta en CTPATH para poder acceder a los servicios de rutas personalizadas mencionadas en RFM003.

- **RFM008:** Los usuarios podrán consultar desde la aplicación móvil un ranking con los usuarios "más verdes" (los que menos contaminación emiten por distancia recorrida) y los que más usan la aplicación. Esta es la semilla de la idea de gamificación, que podrá extenderse después.

- **RFM009:** Los usuarios podrán consultar un resumen de las rutas que han consultado en el sistema, la cantidad de gases contaminantes emitidos y la distancia recorrida.

3.2 Funcionalidades de la aplicación web (calculador de rutas)

- **RFW001:** El usuario puede indicar en un mapa el punto de origen y destino de un trayecto que desea realizar en coche. Opcionalmente puede indicar el tipo de coche que posee de entre varios mostrados y la hora de comienzo del trayecto. Con esta

información, la aplicación, a la orden del usuario, calculará varias rutas alternativas que mostrará al usuario y dibujará en el mapa. Se mostrarán al menos tres rutas: la menos contaminante, la más corta en tiempo y una de duración y contaminación intermedias.

- **RFW002:** Para cada ruta propuesta, la aplicación mostrará la contaminación estimada que produce el coche en dicho trayecto con indicación de la cantidad de CO, CO_2, NO_x, hidrocarburos y partículas emitidas durante el mismo.

- **RFW003:** Los usuarios podrán autenticarse en el sistema para obtener rutas personalizadas de acuerdo a su perfil de conducción.

- **RFW004:** La aplicación permitirá a sus usuarios crear una cuenta en CTPATH para poder acceder a las rutas personalizadas mencionadas en RFW003.

- **RFW005:** Los usuarios podrán consultar un ranking con los usuarios "más verdes" (los que menos contaminación emiten por distancia recorrida), los que más distancia recorren con la aplicación móvil activa y los que más usan la aplicación.

- **RFW006:** Los usuarios registrados podrán consultar un resumen de las rutas que han calculado en el sistema, la cantidad de gases contaminantes emitidos y la distancia recorrida.

3.3 Funcionalidades del sistema central

- **RFC001:** El sistema central recolectará los datos recogidos de la aplicación móvil relativa a la posición y aceleración de los usuarios de la misma.

- **RFC002:** El sistema central recolectará los datos recogidos por los sensores inalámbricos repartidos por la ciudad.

- **RFC003:** Con los datos recogidos, el sistema central estimará el estado del tráfico.

- **RFC004:** Con la información histórica del estado del tráfico, realizará predicciones acerca del estado del tráfico en un futuro cercano usando teoría de series temporales y técnicas de aprendizaje máquina (Machine Learning).

- **RFC005:** La información sobre posición y aceleración de los usuarios autenticados permitirá al sistema realizar un perfil de movilidad de los mismos con el que realizar estimaciones más precisas de duración y contaminación en los trayectos solicitados por estos usuarios.

- **RFC006:** La información sobre el estado actual del tráfico se hará pública a través de la plataforma facilitada por FIWARE [VS+11] si los habilitadores genéricos que ésta tiene lo permiten, o medios equivalentes que promocionen el acceso público a la aplicación.

3.4 Funcionalidades de los sensores inalámbricos

- **RFS001:** Varios sensores inalámbricos repartidos por la ciudad estimarán el número de coches que pasan por su cercanía capturando las direcciones (direcciones del protocolo de acceso al medio o MAC) de los dispositivos inalámbricos.

- **RFS002**: La información recogida se podrá transmitir al servidor central mediante una conexión a Internet o, si no es posible, se almacenará localmente para su posterior recuperación por parte de un operario de CTPATH.

3.5 Funcionalidades de la aplicación móvil para operarios de CTPATH

- **RFO001:** La aplicación móvil de los operarios buscará sensores inalámbricos en su rango de de cobertura inalámbrica (acción *sniffer*) y cuando detecte uno solicitará que le transmita toda la información recolectada, almacenada internamente y no transmitida al sistema central.

- **RFO002:** Cuando la aplicación posea información procedente de uno o varios sensores, la transmitirá al sistema central a través de su conexión a Internet, típicamente utilizando 3G.

4. Hardware necesario

Dentro de CTPATH dispondremos de diferentes sistemas **hardware**. Por un lado, las **máquinas fijas** internas del sistema que realizan el cómputo inteligente de rutas (y maneja perfiles de usuarios, gamificación, cuentas, etc.). Por otro lado están los **dispositivos móviles**

del usuario u operario. Finalmente, tenemos los **sensores** de captación de flujos vehiculares que se ubicarán dentro de la ciudad.

La máquina en la que se ejecutará la aplicación (**servidor externo**) a la que los usuarios pueden solicitar y visualizar las rutas estimamos que puede ser equivalente en un Intel Xeon E3-1220 v3 de cuatro núcleos (ver Figura 2). La gestión de cuentas, petición de históricos, etc. se hace también en este servidor.

Figura 2. Máquinas HP con Intel Xeon E3-1220 v3 (izq.) y Dell con Intel Xeon E5-2670 v3 (der.)

Para acometer el cálculo intensivo que implica la optimización de rutas, emisiones y tiempos de viaje, utilizaremos un multiprocesador compuesto de dos Intel Xeon E5-2670 v3 de 12 núcleos, que permitirá el cómputo de hasta 48 rutas diferentes en paralelo lo que se ajusta perfectamente a las necesidades del proyecto. Esta es la máquina (**servidor interno**) que pretendemos adquirir en el proyecto, o una equivalente en cuanto a potencia computacional, ya que hay que dar servicio a muchos usuarios y para cada uno hacer cálculos complejos y que requieren un tiempo considerable (ver Figura 2). Configuraciones de precio competitivo similares pueden conseguirse en Dell (casi sin competidores dentro de los rangos de nuestro presupuesto).

Los **sensores** se desarrollarán sobre placas Raspberry Pi 2 [FW+13][HK+14] (ver Figura 3). Para la detección del flujo de vehículos se equipará el sensor con un interfaz Bluetooth ASUS USB-BT400 y una interfaz WiFi Alfa con capacidad para trabajar en modo monitor. Para la comunicación del sensor con el servidor se instalará otra interfaz WiFi, una TP LINK TL-WN725N, y opcionalmente, un módem Huawei (o similar) que ofrece tecnología 3G.

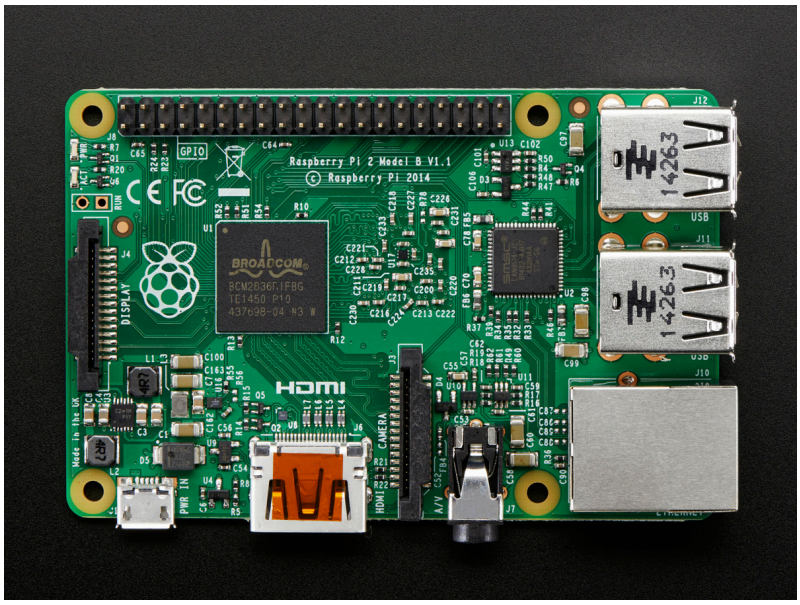

Figura 3. Placa Raspberry Pi 2

También usaremos como posible sensor un Kit de desarrollo Arduino [Kua14][Mic11] compuesto por la placa del microcontrolador ATMega 2560 R3, módulo Bluetooth, módulo WiFi con soporte para tarjeta SD y sensores de temperatura, humedad, etc. (ver Figura 4). Arduino es una plataforma de hardware abierto inicialmente desarrollado en el *Interaction Design Institute Ivrea* en el norte de Italia. Actualmente se ofrecen varios modelos de placas de desarrollo tales como Arduino Uno, Leonardo, Due, Yun, Tre, Zero, Mega, etc. Esta plataforma será evaluada como posible dispositivo a ser distribuido por la ciudad para captación de flujos vehiculares.

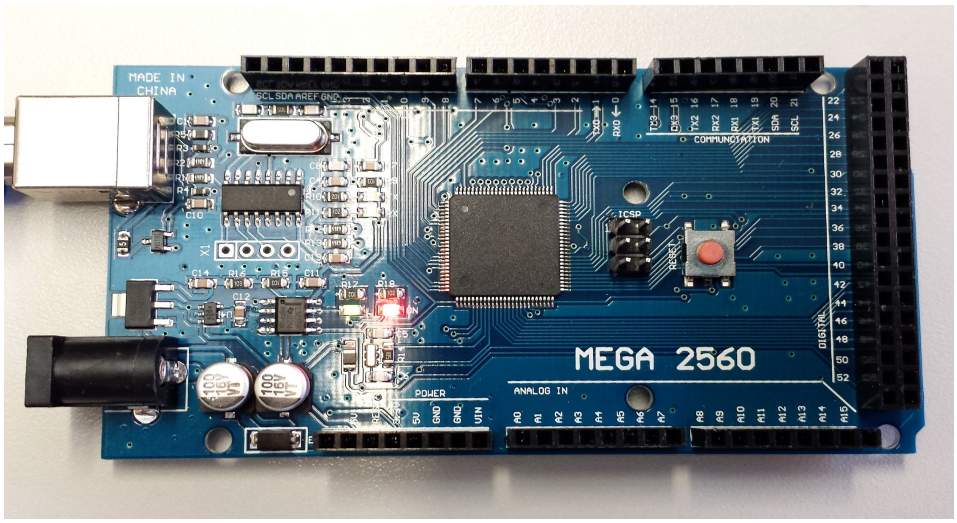

Figura 4. Arduino MEGA2560

Para las pruebas con las aplicaciones móviles utilizaremos un teléfono móvil Samsung Galaxy Alpha (SM-G850Y) con sistema operativo Android 5.0 (desde mayo de 2015, según Samsung). Este dispositivo se caracteriza por la cantidad y calidad de sus sensores. También utilizaremos un dispositivo iOS, en particular un iPad Mini 3 con 128 GB y conexión WiFI y celular (ver Figura 5). Finalmente, es posible que evaluemos otros terminales móviles, ya que pretendemos testear la aplicación en dispositivos populares en manos de los ciudadanos.

Figura 5. Samsung Galaxy Alpha (izq.) y iPad Mini 3 (der.)

5. Arquitectura de CTPATH

La arquitectura de la aplicación incorpora los componentes que describimos a continuación. En las **Figuras 6 y 7** se puede ver un esquema de la interrelación entre componentes. Para la siguiente descripción vamos a organizar los distintos componentes software de acuerdo a su ubicación en el hardware necesario en CTPATH.

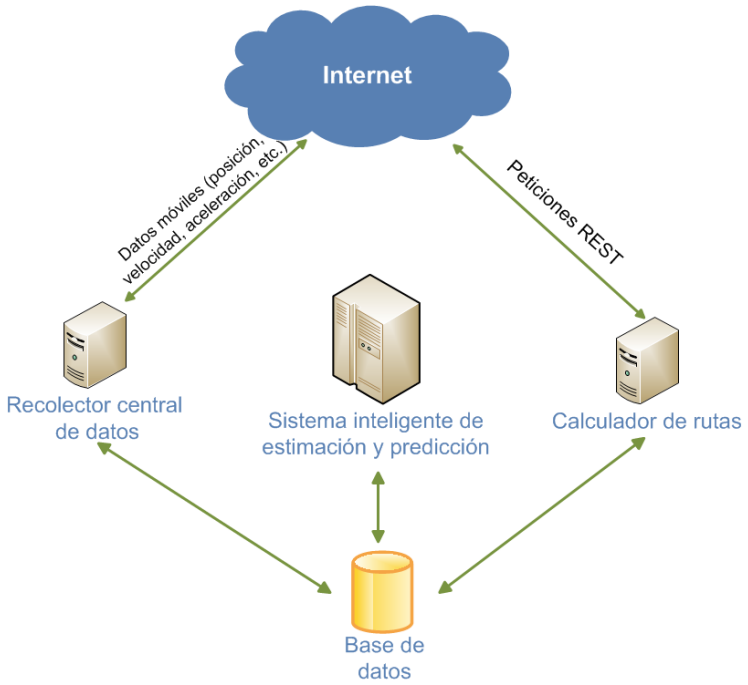

Figura 6. Arquitectura del sistema central.

Figura 7. Arquitectura del sistema de comunicación.

5.1 Componentes en el servidor central

- **Recolector central de datos**. Este componente será el encargado de tomar los datos de las aplicaciones móviles que corren en los dispositivos de los usuarios y de los sensores repartidos por la ciudad. Todos los datos recibidos serán almacenados en una base de datos local. Este componente será un servicio Web implementado según el paradigma REST [RR07]. Los datos que recibirá de las aplicaciones móviles de los usuarios finales serán su posición en un instante dado, su velocidad y su aceleración.

- **Sistema inteligente de estimación y predicción**. Este componente tomará los datos almacenados en la base de datos por el **recolector de datos** y los procesará para estimar: 1) el tráfico existente en cada calle de la ciudad, 2) el tiempo estimado en atravesar cada calle de la ciudad, 3) el tráfico en el futuro cercano (varios minutos), 4) el perfil de conducción de los usuarios registrados (cómo aceleran, cuál es su velocidad de

crucero, etc.). Una vez realizadas estas estimaciones las almacenará en la base de datos y servirán de entrada para el **calculador de rutas**.

- **Calculador de rutas**. Este componente ofrece una aplicación web para que el usuario pueda determinar la ruta más corta/ecológica de un punto a otro y un servicio web REST para ofrecer la misma funcionalidad a la aplicación móvil del usuario final. Para calcular el tiempo y la contaminación hará uso de las estimaciones y predicciones de tráfico y perfil de usuarios creados por el **sistema inteligente de estimación y predicción**. Este componente también deberá ofrecer la posibilidad (tanto en el servicio web como en la aplicación web) de crear una cuenta de usuario en CTPATH, que podrá utilizarse para obtener rutas personalizadas de acuerdo al perfil de conducción de cada usuario. Asimismo, proporcionará información sobre los rankings de usuario de acuerdo a los tres criterios de gamificación mencionados en los requisitos RFW005 y RFM008.

5.2 Componentes en el lado de los usuarios de CTPATH

- **Aplicación móvil de usuario**. Esta aplicación ofrece al usuario la posibilidad de escoger la ruta más corta/ecológica entre un origen y destino, indicando paso a paso por dónde debe ir. Para ello solicitará la ruta al **calculador de rutas** a través de su servicio web REST. Ofrecerá información sobre los usuario que más usan la aplicación y los más “verdes”. La aplicación también recogerá información sobre posición, velocidad y aceleración del dispositivo y la mandará al **recolector central de datos** para la creación de un perfil de usuario y estimar estado del tráfico. Se implementará una versión para Android y, opcionalmente, para iOS.
- **Aplicación Web para el usuario**. Esta aplicación ofrece al usuario la posibilidad de escoger la ruta más corta/ecológica entre un origen y destino. Para ello solicitará la ruta al **calculador de rutas** a través de su servicio web REST. Ofrecerá información sobre los usuario que más usan la aplicación y los más “verdes”.

5.3 Componente en los sensores

- **Aplicación de sensorización**. Este componente recogerá información sobre las direcciones hardware de todas las interfaces Wi-Fi y Bluetooth que observe y la mandará al **recolector central de datos**. Esta aplicación puede hacer llegar la información al recolector de dos formas: 1) usando un conexión de datos 3G gracias a una tarjeta 3G incorporada en el sensor, y 2) mediante recolección de datos por parte

de un operario que porta una aplicación en su dispositivo móvil para conectarse por Wi-Fi al sensor y envía la información al **recolector central de datos** a través de una conexión a Internet.

5.4 Componente en el dispositivo móvil del operario de CTPATH

- **Aplicación de recolección de datos móvil**. Este componente toma los datos almacenados en un sensor cercano mediante una conexión Wi-Fi directa con él y los transfiere a través de una conexión a Internet al **recolector central de datos**. La aplicación está pensada para utilizarse en el caso de sensores que no tienen conexión 3G y cuyos datos deben ser recolectados por parte de un operario que visita el sensor. Si en el momento de la recogida de datos el dispositivo del operario no tiene conexión a Internet, se almacenarán localmente en dicho dispositivo para su entrega posterior al **recolector central de datos**.

5.5 Sistema de comunicación

Uno de los ejes fundamentales sobre los que se asienta este proyecto es la plataforma de comunicación urbana. Así, se definen distintos escenarios de comunicación que permiten satisfacer las distintas necesidades del sistema, a la vez que facilita la interoperatibilidad con distintos tipos de usuarios. Los escenarios que se definen en nuestro sistema son los siguientes:

- **Sensores equipados con interfaces de comunicación celular (3G)**: Los sensores que tengan capacidad de comunicarse empleando tecnologías celulares (3G) enviarán directamente la información recolectada al **recolector de datos**. Esta información actualizará en tiempo real la información del servidor que será utilizada para el cálculo inteligente de las rutas.
- **Sensores equipados con interfaces de comunicación inalámbrica (Wi-Fi)**: Los sensores que se comuniquen únicamente empleando comunicaciones inalámbricas de corto alcance (Wi-Fi) almacenarán la información en un unidad interna. Con una frecuencia dada, un operario equipado con un dispositivo móvil con 3G se acercará al dispositivo, tras autenticar al usuario el sensor transmitirá los datos recolectados al dispositivo móvil, que los retransmitirá directamente al servidor. Esta información actualizará los datos almacenados en el servidor y será utilizada para mejorar las estimaciones empleadas para el cálculo de las rutas.

- **Usuarios equipados con dispositivos móviles con interfaces de comunicación celular (3G)**: Los usuarios que accedan a la aplicación a través de un dispositivo móvil 3G tendrán la posibilidad de poder navegar a través del mapa y recibir las instrucciones para alcanzar su destino. Además, estos usuarios podrán interactuar directamente con el servidor en tiempo real para enviar su información cinemática (posición, velocidad instantánea, aceleración, etc.) que será utilizada para inferir el estado del tráfico actual y un perfil de conducción. De este modo este usuario se comporta como un sensor móvil del sistema, mejorando tanto la granularidad de la información del tráfico como la precisión de la estimación.

6. Tecnologías y herramientas software empleadas

Esta sección es **altamente técnica** y muestra las decisiones adoptadas en el proyecto después de una fase inicial de estudio. En la Tabla 1 se resumen las principales tecnologías y paquetes software externos utilizados para desarrollar los componentes anteriores. En las siguientes secciones detallamos estas tecnologías y herramientas.

Tabla 1. Descripción resumida de la tecnología/software y su uso en CTPATH.

Nombre	**Descripción y su utilización dentro de CTPATH**
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. Tendremos 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.
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 que generamos.
SUMO	Una herramienta de simulación de tráfico, libre y abierta [KE+12]. 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. Seguramente usaremos 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.

6.1 Una nueva infraestructura software para el desarrollo de aplicaciones

En esta sección se describe la infraestructura utilizada para el desarrollo de CTPATH (que también se utilizará para el resto de desarrollos del proyecto completo). 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 1: **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 8). 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 CTPATH.

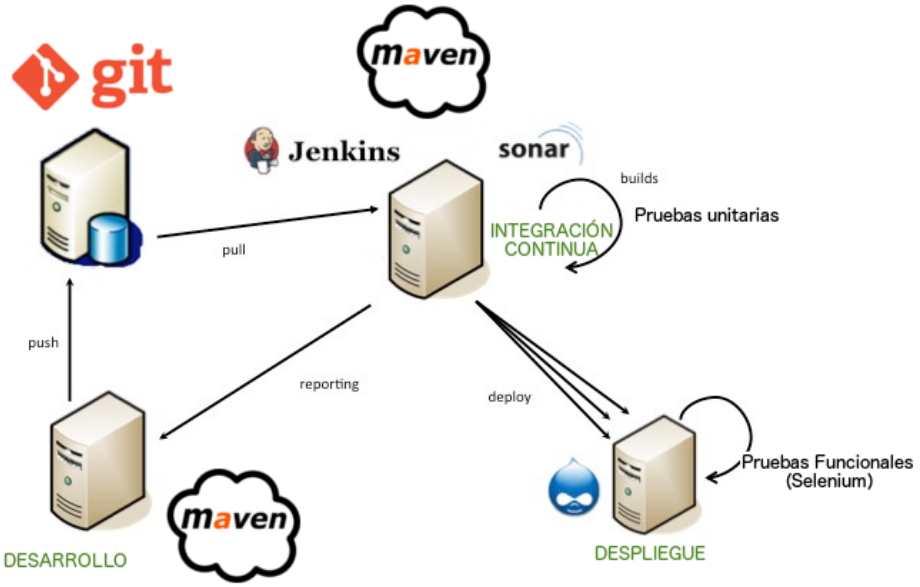

Figura 8. La interacción entre las cuatro herramientas principales de nuestra infraestructura de desarrollo de software.

6.1.1 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**).

6.1.1.1 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 4).

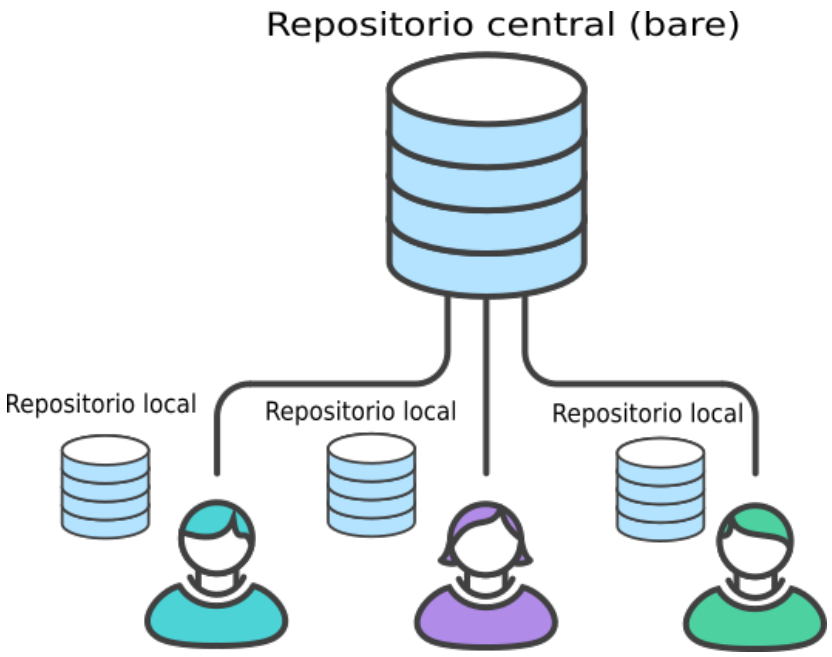

Figura 9. Organización de repositorios Git en nuestro proyecto.

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.

6.1.1.2 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 CTPATH 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 CTPATH.

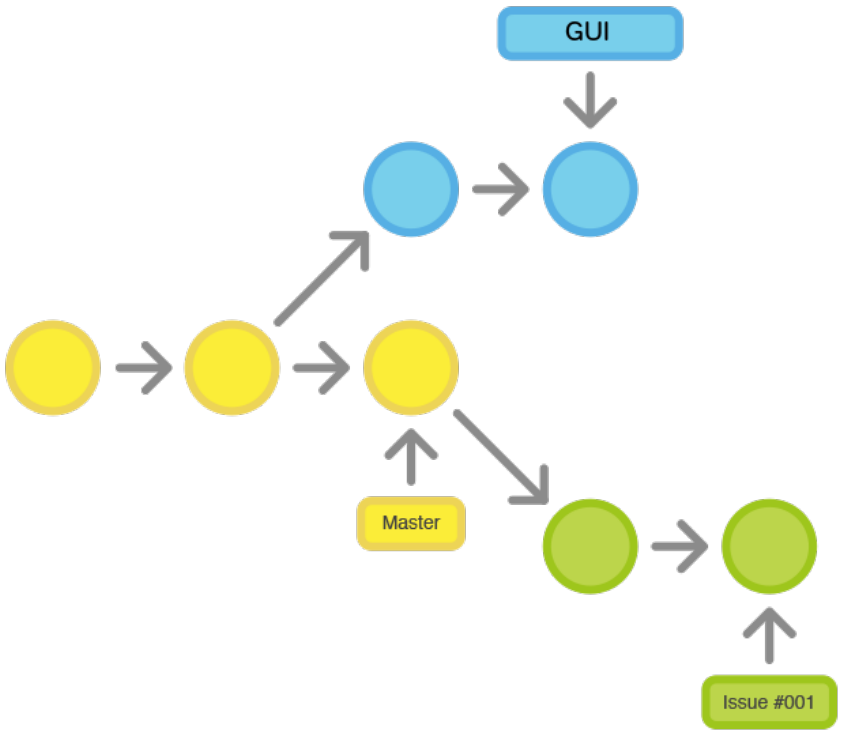

Figura 10. Rama característica de flujo de trabajo en CTPATH.

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 en el repositorio central.

Por último, también utilizaremos ramas para preparar los **prototipos** S3 y S4. 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.

6.1.2 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 todo esto, desde hace muchos años se han propuesto 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 desarrollo 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.

6.1.3 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 provocadas por 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 11. Algunos ejemplos de tareas utilizadas en nuestro servidor Jenkins.

Se definirán al menos dos tareas en Jenkins (ver Figura 11):

- 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).

6.1.4 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 12), 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.

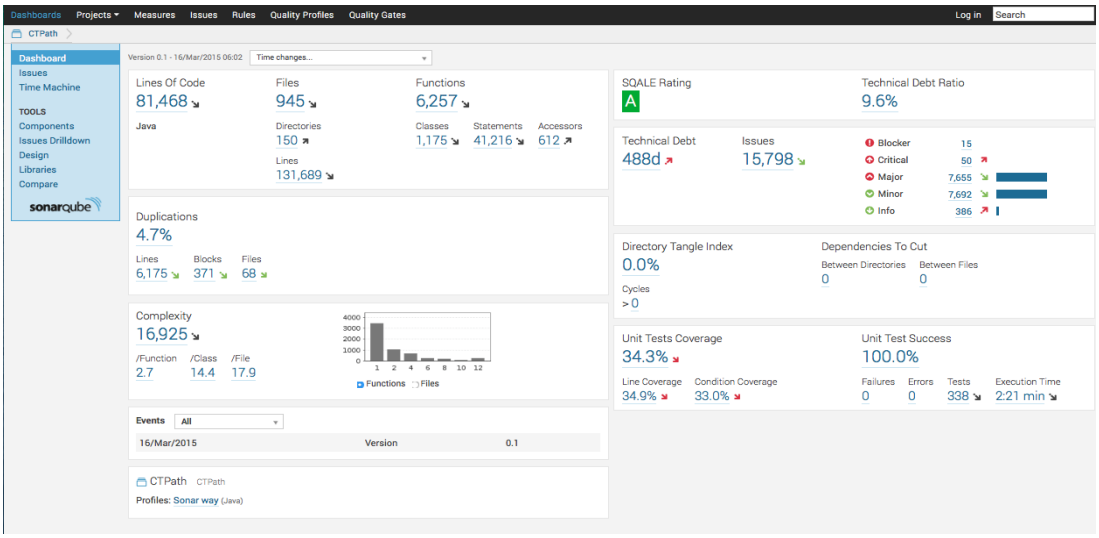

Figura 12. Medidas globales para un proyecto en SonarQube.

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).

6.1.5 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.

6.1.6 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 puede acceder fácilmente al estado de las tareas de desarrollo y participar en cada una (ver Figura 13).

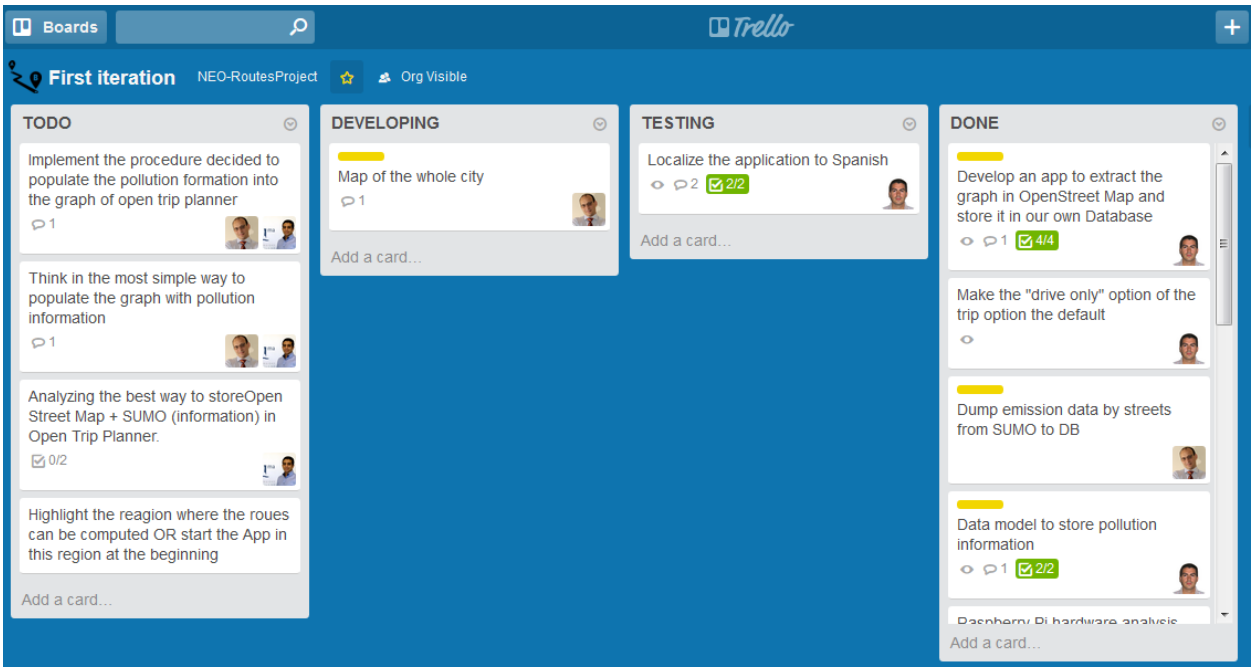

Figura 13. Captura de pantalla del tablero de Trello para la aplicación de CTPATH.

6.2 Tecnologías

El **servidor central** de la aplicación se desarrollará usando el lenguaje Java y tecnologías asociadas. Las razones por las que se ha escogido la plataforma Java son: 1) el amplio conocimiento de este lenguaje y sus bibliotecas por parte del equipo de desarrollo y 2) la existencia de una aplicación web open source cuya estructura software e interfaz gráfica facilitan que nos centremos en el apartado inteligente propuesto en el proyecto: **OpenTripPlanner** (http://www.opentripplanner.org). De esta forma, podemos centrarnos en las contribuciones de este proyecto frente a otros competidores (e.g. información de contaminación y perfiles personalizados de conducción, aproximación holística, gamificación....) en lugar de emplear tiempo en la construcción de una infraestructura software básica que ya exista.

Para la implementación de las aplicaciones que se ejecutan en el lado del **cliente (usuarios y operarios de CTPATH)** usaremos distinta tecnología dependiendo del tipo de cliente:

- Para la aplicación web que se ejecuta en el navegador optaremos por **HTML5** junto con **javascript** en el lado del cliente. OpenTripPlanner ya proporciona un cliente con estas características que tendremos que modificar para añadir nuestras contribuciones.

- Para la aplicación en dispositivos Android, emplearemos el SDK oficial de **Android**, que usa Java como lenguaje base.
- Para la aplicación en dispositivos iOS, emplearemos la API de **iOS 8** con el lenguaje Swift.

Por último, para las aplicaciones en los dispositivos sensores (Raspberry PI y Arduino) usaremos las plataformas y lenguajes de desarrollo propios de cada una de ellas. Para **Raspberry Pi** se desarrollará empleando herramientas Linux compatibles con la plataforma hardware que utilizarán principalmente C++ y Python. Para Arduino utilizaremos el entorno de desarrollo **ARDUINO 1.6.3** para escribir los programas en C++ que serán luego compilados y volcados en la placa de desarrollo.

En lo que sigue daremos una breve descripción de cada una de estas tecnologías.

6.2.1 En el servidor

6.2.1.1 Servicios Web REST

Los servicios web son aplicaciones accesibles a través de Internet pensadas para ser consumidas por programas de ordenador, en lugar de personas. Por este motivo, los servicios Web ofrecen una API en la que las distintas operaciones son transformadas en distintos tipos de mensajes y los argumentos son documentos XML o JSON (típicamente) que se envían desde el cliente al servidor. Igualmente las respuestas del servidor, cuando contienen resultados se devuelven como documentos XML o JSON. En el caso particular del paradigma REST (*Representational State Transfer*), el servicio define recursos identificados con una URI y se usan los métodos propios del protocolo HTTP (normalmente, GET, POST, PUT y DELETE) para especificar la operación a realizar con dichos recursos (leer, insertar, modificar y eliminar).

6.2.1.2 JPA

Java Persistence API (JPA), es una API que permite realizar una traslación entre objetos Java y filas en las tablas de una base de datos relacional (objeto-relacional). En la programación orientada a objetos (POO), el concepto de objeto y clase son la esencia del paradigma y todo el desarrollo de las aplicacion OO gira en torno a ellos. Por otro lado, en el mundo de las bases de datos relacionales, son las relaciones (tablas) y las tuplas (filas) de datos los que poseen protagonismo. Cuando en una aplicación programada siguiendo el

paradigma OO se hace necesario almacenar información en una base de datos relacional, se hace necesario transformar los objetos en filas para almacenarlos en las tablas. JPA permite al desarrollador indicar cómo se va a realizar esta transformación, ocultando los detalles relacionados con las consultas SQL y la ubicación de la base de datos. El uso de esta tecnología permite reducir el tiempo de desarrollo, centrando el esfuerzo en la elaboración del modelo de datos y su uso en la aplicación, más que en los detalles del almacenamiento de estos datos.

6.2.2 En el cliente Web

6.2.2.1 HTML5

HTML es un lenguaje de marcado que permite especificar el contenido de las páginas Web (ya sean estáticas o dinámicas). En combinación con CSS y Javascript, permite implementar auténticas aplicaciones en el lado del cliente (navegador web) que pueden comunicarse con un servidor mediante peticiones AJAX para actualizar la información. Toda aplicación Web que deba ser accedida por un usuario humano debe devolver las respuesta en algún lenguaje de marcado (normalmente HTML o XHTML). La ventaja de HTML5 frente a sus versiones anteriores o XHTML es que su Document Object Model (DOM) es estándar, por lo que el código Javascript se simplifica al no tener que considerar los distintos DOM que distintos navegadores generaban hasta la fecha. Además, en la versión 5, HTML ha redoblado los esfuerzos por realizar una separación clara entre contenido (especificado en HTML) y formato (especificado usando hojas de estilo en cascada o CSS).

6.2.2.1 CSS

Las hojas de estilo en cascada (*Cascade Style Sheets*) permiten especificar el formato que tendrán las páginas en HTML5. Aspectos como el color del texto, su tamaño, la ubicación y forma de los menús, etc. estarán controladas mediante hojas de estilo en cascada. Esta separación permite al desarrollador de la aplicación centrarse en la funcionalidad mientras que el diseñador gráfico puede trabajar de manera casi aislada en el aspecto visual de la misma.

6.2.2.3 JavaScript

Es un lenguaje de programación interpretado para el que casi todos los navegadores ofrecen intérpretes en la actualidad. El código javascript acompaña a páginas HTML y las dotan

de acción, convirtiéndose en páginas activas. Al poseer el navegador posibilidad de ejecutar código en la máquina del cliente se distribuye la carga computacional entre el cliente y el servidor, además de reducir el ancho de banda requerido para las aplicaciones Web y aumentando el rendimiento de la aplicación, a la vez que se mejora la experiencia del usuario.

6.2.3 En los clientes móviles

6.2.3.1 Android

Se trata de un sistema operativo originalmente diseñado para dispositivos móviles (pero hoy en día se ejecuta en televisores incluso). De acuerdo al último estudio de Kantar Worldpanel [KWP15], el sistema operativo Android se encuentra instalado en el 87.6 % de los dispositivos móviles en España. Esto incluye teléfonos inteligentes, tabletas y relojes. Debido a su gran penetración en el mercado, no podemos obviarlo a la hora de realizar una versión para dispositivos móviles de nuestra aplicación. Las aplicaciones Android se desarrollan en Java, aunque luego son compiladas a un lenguaje intermedio que entiende la máquina virtual Dalvik (diferente de la máquina virtual Java).

6.2.3.2 iOS

Este sistema operativo acompaña siempre a los dispositivos móviles de la compañía Apple. Es el segundo más usado en de los dispositivos móviles (8.7% en España) y por eso lo consideramos en el proyecto en segundo lugar. Las aplicaciones para iOS se puede programar en Objective-C (superconjunto de C) o, desde hace algunos meses, en el nuevo lenguaje Swift, más próximo en sintaxis a Java y C#.

6.2.4 En los sensores

Los sensores desplegados en las principales vías de la ciudad captarán el flujo vehicular. Los vehículos serán detectados a partir de los dispositivos con capacidad de comunicación inalámbrica que hayan dentro del mismo (Bluetooth y Wi-Fi) estén utilizando o no el sistema CTPATH. Se tendrán en cuenta tanto los dispositivos del propio vehículo como los

de los usuarios que viajan en el mismo. Las herramientas software que se utilizan dependen de la naturaleza hardware que se emplee, Raspberry Pi o Arduino, del sensor.

6.2.4.1 Raspberry Pi

La Raspberry Pi 2 emplea **Raspbian**, un sistema operativo basado en Debian (Linux) que está optimizado para este hardware específico. Así, las herramientas software que se utilizarán tendrán que ser compatibles con este sistema operativo.

Se va a desarrollar una aplicación empleando C++ y lenguajes de scripting como Python. La aplicación estará basada en herramientas de auditoría y análisis de redes inalámbricas como son tcpdump, tshark, iwconfig, airmon-ng, etc. para la detección de los dispositivos Wi-Fi y hcitool para descubrir los dispositivos Bluetooth. La aplicación almacenará la información del flujo de tráfico y la transmitirá al servidor a través del interfaz 3G o a un operario empleando la interfaz Wi-Fi según se le indique.

6.2.4.2 Arduino

Para el desarrollo en Arduino utilizaremos el entorno de desarrollo ARDUINO 1.6.3 que consiste en el editor de textos y el compilador C++, gcc. Además serán necesarias las bibliotecas de desarrollo destinadas a comunicarse con los dispositivos tales como SoftwareSerial, Wi-Fi, etc. La comunicación de los datos obtenidos podrá realizarse mediante el uso de Internet o bien ser almacenados y recolectados al estar en las proximidades del punto de sensorización.

7. Primer modelo entidad-relación

En la siguiente figura mostramos el primer modelo entidad-relación (E-R) de la aplicación CTPATH. En ella se pueden observar las entidades necesarias para implementar los requisitos y las relaciones existentes entre éstas. Este es el modelo inicial E-R que podría sufrir alguna mejora en los dos prototipos de la aplicación. A continuación detallaremos estas entidades y relaciones.

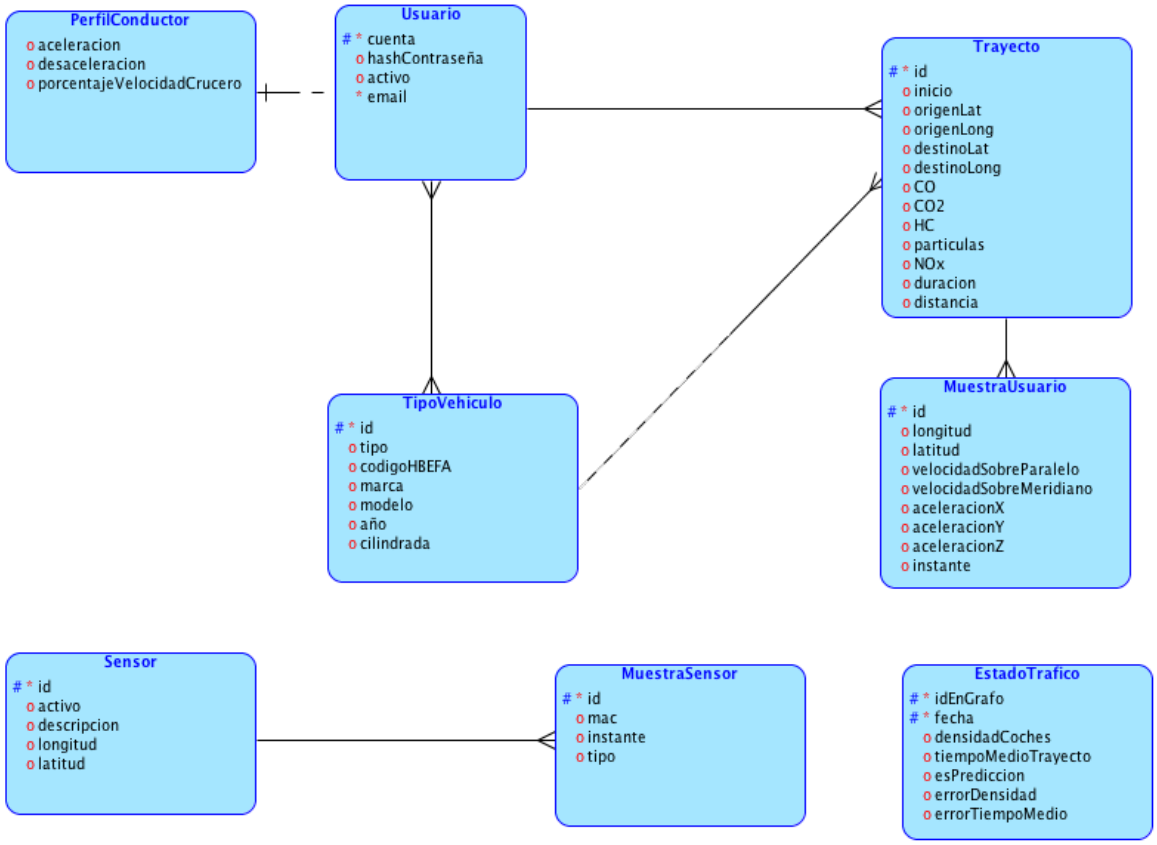

Figura 14. Modelo entidad-relación para la aplicación CTPATH.

La entidad **Usuario** representa a los usuarios registrados de la aplicación. Esta entidad almacena la información mínima necesaria para la gestión de la cuenta y restauración de la contraseña.

Cada usuario registrado que utilice la aplicación móvil enviará información al sistema acerca de su posición, velocidad y aceleración. La entidad **MuestraUsuario** almacena dicha información. Las muestras recibidas se agruparán de acuerdo al trayecto realizado por el usuario en ese momento determinado. La entidad **Trayecto** representa este itinerario y asociado a él existirán muchas muestras. El usuario puede realizar muchos trayectos, de ahí la relación uno a mucho entre **Usuario** y **Trayecto**. Para cada trayecto, el usuario empleará un vehículo de los que ha registrado en el sistema y esto se representa mediante la relación muchos-a-uno entre **Trayecto** y **TipoVehiculo**. Aunque no se indica en el modelo E-R, el tipo de vehículo escogido por el usuario para un trayecto debe ser uno de los que están asociados al usuario. El usuario podrá gestionar sus tipos de vehículos, añadiendo nuevos tipos si lo desea. El atributo tipo de

TipoVehiculo permitirá a la aplicación seleccionar aplicar los coeficientes correctos al modelo de estimación de polución generada.

Por último, el sistema creará para cada usuario registrado un perfil de conductor, representado por la entidad **PerfilConductor**. Este perfil permitirá a CTPATH realizar una estimación más precisa sobre el tiempo necesario por un usuario para realizar un trayecto y la polución emitida a la atmósfera. El perfil se actualizará de acuerdo a los datos recogidos en **MuestraUsuario**.

Los sensores ubicados en la ciudad están representados mediante la entidad **Sensor**. Estos sensores generarán datos acerca de los dispositivos Wi-Fi detectados. Esa información viene representada mediante la entidad **MuestraSensor**.

Por último, CTPATH estimará el estado del tráfico basándose en las mediciones de los sensores y de los propios usuarios registrados. Estas estimación están representadas por la entidad **EstadoTrafico** y consistirán en información sobre la densidad de coches y el tiempo medio requerido para atravesar una calle junto con su error asociado. Además de las estimaciones, el sistema realizará predicciones usando técnicas de análisis de series temporales. La información de esta entidad se planea almacenar también en la plataforma de soporte a datos abierto FIWARE.

8. Prototipo de la interfaz gráfica de CTPATH

En esta sección mostramos los el diseño de la interfaz gráfica de usuario (GUI) para usar las funcionalidades finales de CTPATH. En particular, en la Figura 15 se muestra el prototipo para la interfaz gráfica de la aplicación **web** y en la Figura 16 el prototipo para la interfaz gráfica de la aplicación **móvil**.
Entre todas las funcionalidades de CTPATH destaca la obtención de tres rutas diferentes para ir desde un origen a un destino dados por el usuario: la ruta más ecológica (la que menos contaminación produce), la ruta más rápida y una ruta intermedia en la que se pondera la contaminación y el tiempo de ruta según indique el usuario (por defecto estarán al 50% la contaminación y al 50% tiempo de ruta).

Una vez que se accede a la aplicación web el usuario puede autenticarse o no para usar los servicios de ruta. Si el usuario se autentica se le mostrará información sobre su perfil, como las estadísticas de uso y rankings. Tras este paso, el usuario va a establecer un punto de origen, un punto de destino y el porcentaje para ponderar la contaminación en la ruta intermedia que

irá del 0% a 100%. La ponderación del tiempo de ruta será siempre 100% menos la ponderación elegida para la contaminación.

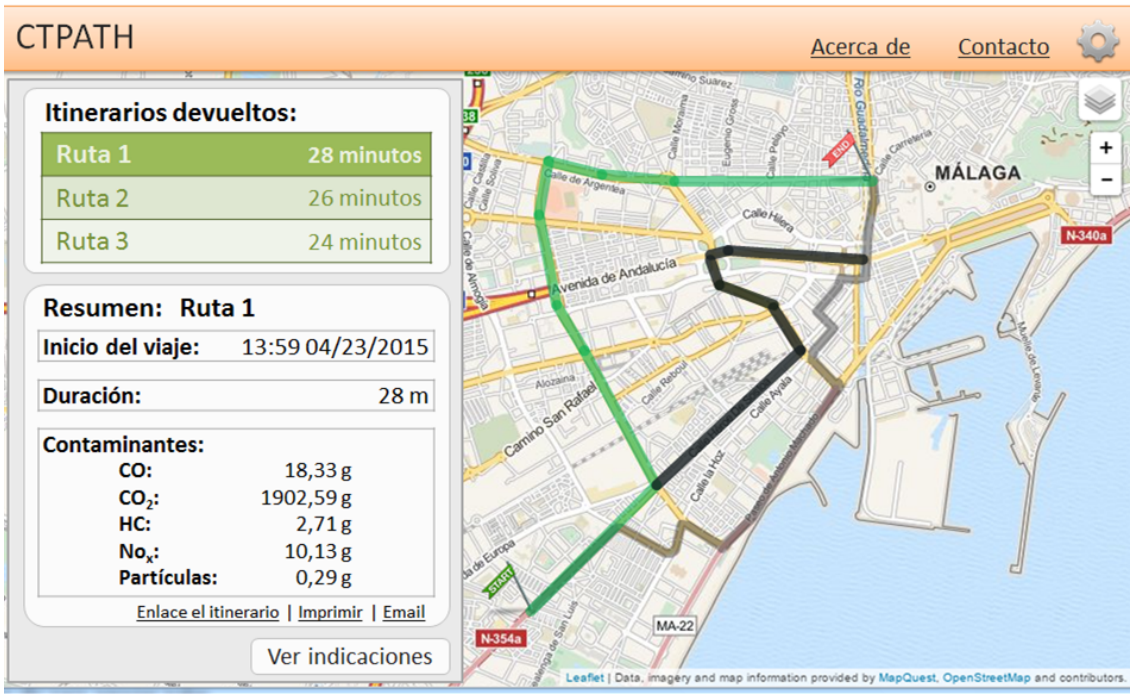

Figura 15. Prototipo de la GUI de CTPATH (aplicación web).

Podemos observar que al establecer un punto de origen y destino, la aplicación calcula varias rutas y las muestra al usuario. La ruta indicada en color verde es la ruta que menos contaminación produce. La más oscura es la ruta que menos tiempo requiere. Por último, la ruta gris es una ruta con contaminación y duración ponderadas según el usuario. Cuando el usuario selecciona una de las rutas en el panel de la izquierda, aparece justo debajo información adicional, indicando la duración del viaje y la cantidad de gases y sustancias contaminantes emitidas durante la misma. El botón “Ver indicaciones” del panel inferior izquierdo permite al usuario ver paso a paso las indicaciones a seguir para llegar desde el origen hasta el destino.

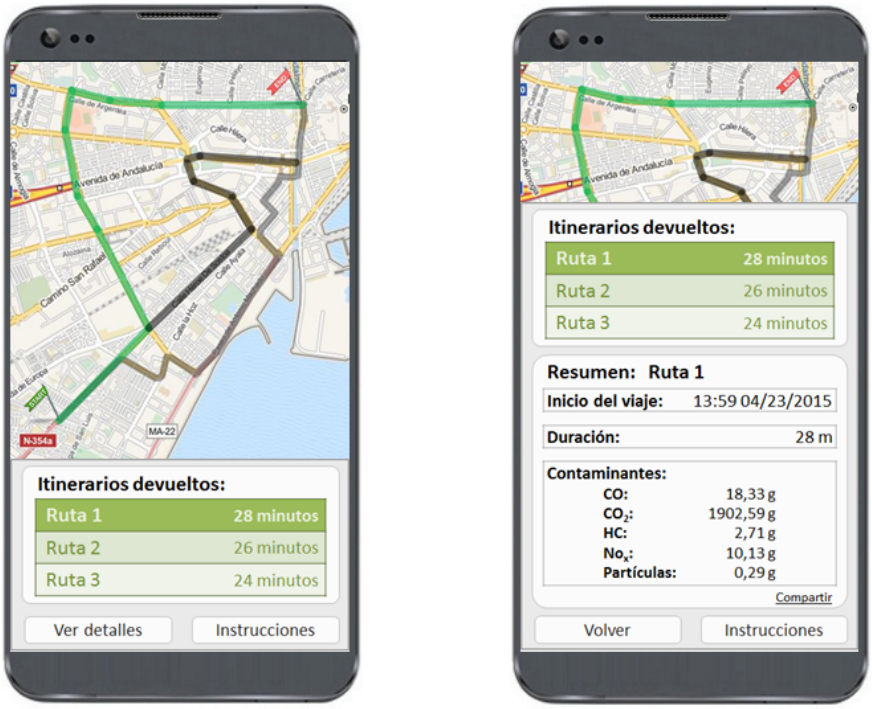

Figura 16. Prototipo de la aplicación móvil de CTPATH.

En el caso de la aplicación móvil la pantalla de inicio muestra directamente la información de perfil del usuario, como puede ser la distancia total recorrida, la distancia recorrida la última semana, los ránkings, etc. Desde esa pantalla se pasa a la función principal, el cálculo de las rutas. Así pues, su funcionamiento es parecido a la aplicación web, lo que cambia es la forma de ver las rutas devueltas y que la información de las instrucciones, además de mostrarse por pantalla, se irá leyendo a medida que se vaya por la ruta.

Según se muestra en la Figura 16, en la pantalla principal, debido al reducido tamaño del móvil, se mostrará únicamente las rutas en el mapa junto con el panel de selección de rutas. Ahí podemos encontrar también los botones para ver las instrucciones paso a paso para llegar al destino y empezar la ruta. Los detalles de cada ruta (por ejemplo, emisiones) se podrán consultar al pulsar el botón “Ver detalles”, que provoca que aparezca el panel de detalles de ruta, como puede observarse en la imagen.

9. Referencias

[DS+11] Deterding, S, Sicart, M., Nacke, L., O'Hara, K., and Dixon, D., “Gamification. using game-design elements in non-gaming contexts,” CHI'11 Extended Abstracts on Human Factors in Computing Systems, pp. 2425,2428, ACM, 2011

[FW+13] Fung, P.T., White, D.R., Jouet, S., Singer, J., Pezaros, D.P., “The Glasgow Raspberry Pi Cloud: A Scale Model for Cloud Computing Infrastructures," Distributed Computing Systems Workshops (ICDCSW), 2013 IEEE 33rd International Conference on, pp.108,112, 8-11 July 2013.

[GLF13] Cristina Guerreiro, Frank de Leeuw, and Valentin Foltescu, “Air quality in Europe 2013 report,” Tech. Rep., European Environment Agency, 2013.

[GM15] Google Maps. (Abril 2015). https://www.google.es/maps

[HJ+08] Ole Hertel, Steen Solvang Jensen, Martin Hvidberg et al, “Assessing the Impacts of Traffic Air Pollution on Human Exposure and Health,” in Road Pricing, the Economy and the Environment, Advances in Spatial Science, pp. 277–299. Springer Berlin Heidelberg, 2008.

[HK+14] Hajdarevic, K., Konjicija, S., Subasi, A. "A low energy APRS-IS client-server infrastructure implementation using Raspberry Pi," Telecommunications Forum Telfor (TELFOR), 2014 22nd , vol., no., pp.296,299, 25-27 Nov. 2014

[INS15] iGO Navigation - Global Navigation Software. (Abril 2015). http://www.igonavigation.com/

[KE+12] Krajzewicz D., Erdmann J., Behrisch M., and Bieker L., “Recent Development and Applications of SUMO–Simulation of Urban MObility,” Int. J. Adv. Syst. Meas., vol. 5, no. 3, pp. 128–138, 2012.

[Kua14] Y. Kuang, “Communication between PLC and Arduino Based on Modbus Protocol,” in Instrumentation and Measurement, Computer, Communication and Control (IMCCC), 2014 Fourth International Conference on, 2014, pp. 370–373.

[KWP15] Kantar Worldpanel, estudio de uso de sistemas operativos móviles en España (febrero de 2015). http://www.kantarworldpanel.com/global/smartphone-os-market-share

[Mic11] M. Michael, “Arduino Cookbook.” California, EE. UU.: O’Reilly Media, 2011.

[RR07] Leonard Richardson and Sam Ruby, “RESTful Web Services”, O’Reilly Media, 2007

[Tns13] TNS Opinion & Social, “Attitudes of Europeans towards urban mobility,” Tech. Rep. June, 2013.

[TT15] TomTom. (Abril 2015) http://www.tomtom.com

[Uni11] United Nations, “World Urbanization Prospects The 2011 Revision,” Tech. Rep., 2011.

[VS+11] Vazquez, A.G.,Soria-Rodriguez, P., Bisson, P., Gidoin, D., Trabelsi, S., and Serme, G., “FI-WARE security: future internet security core”, pp. 144-152, Springer, 2011

[ZC11] Zichermann, G., and Cunningham, C., “Gamification by design: Implementing game mechanics in web and mobile apps”, O'Reilly Media, Inc., 2011