PROGRAMA DE PROYECTOS DE I+D EN COLABORACIÓN

UNIÓN EUROPEA
Fondo Europeo de Desarrollo Regional
Una manera de hacer Europa

GENERALITAT VALENCIANA

IVACE
INSTITUTO VALENCIANO DE COMPETITIVIDAD EMPRESARIAL

Cool Routing

Cool Routing

Plataforma de optimización de cálculo de rutas de reparto para vehículos eléctricos con carga refrigerada

E3.2. Interfaz de acceso a los datos del sistema

ITENE

Información del documento	
Titulo	Interfaz de acceso a los datos del sistema
Participantes	ITE (coordinador) ITENE
Descripción	Gestión de la información almacenada en la base de datos del proyecto CoolRouting
Autores	Rodríguez Álvaro, José Ángel (ITENE)
Entidad responsable	ITENE
Nivel de difusión	☐ Interno ☒ Publico ☐ Restringido
Fecha de entrega	18/05/2017

Revisión

Version	Fecha	Modificado por	Comentarios
v0.0	03/11/2016	José Ángel Rodríguez Álvaro	Versión inicial plantilla
v0.1	22/11/2016	José Ángel Rodríguez Álvaro	Ampliación de información
V0.2	06/04/2017	José Ángel Rodríguez Álvaro	Sección 4.1 y 4.2
vF0.0	18/05/2017	José Ángel Rodríguez Álvaro	Versión final
vF0.1	23/05/2017	Caterina Tormo Domènech	Actualizado con modificaciones de la segunda anualidad. Versión final

Tabla de contenidos

Índice de Figuras

Índice de Tablas

1 Términos y abreviaciones

Acrónimo	Definición
BBDD	Bases de Datos
PT	Paquete de Trabajo
PTRD	Plataforma de Recogida de Datos
JSON	JavaScript Object Notation
API	Application Programming Interface
REST	REpresentational State Transfer
HTTP	HyperText Transfer Protocol
CRUD	Create, Read, Update and Delete
URL	Uniform Resource Locator

2 Sumario

Este entregable presenta los resultados de la tarea 3.2, Interfaz de acceso a los datos del sistema, estrechamente relacionada con la tarea 3.1 (Esquema de la base de datos) del paquete de trabajo 3.

El documento se ha organizado en diferentes secciones con la finalidad de explicar la gestión de la información que realiza la aplicación mediante la interfaz de acceso a los datos del sistema que permite la lectura, creación, borrado y modificación de la información almacenada en la base de datos.

3 Introducción

El objetivo general de *Cool Routing* es conseguir una mejora en el transporte de mercancía refrigerada empleando el vehículo eléctrico, a través del desarrollo y validación de las tecnologías necesarias para la implementación de una plataforma de cálculo óptimo de rutas de reparto.

El proyecto propone 9 paquetes de trabajo a lo largo de 2 anualidades. El paquete de trabajo 3 será el encargado de recoger los datos del vehículo. Tanto los datos generados por la trazabilidad de los vehículos, las rutas generadas o la información de gestión y funcionamiento de la aplicación. Este PT recoge los datos generados por el resto de plataformas de la aplicación.

Figura 1. Plan de trabajo Cool Routing.

El presente entregable forma parte del paquete de trabajo PT3. Se centra en presentar la estructura de la Plataforma de Recogida de Datos del vehículo (PTRD), en concreto el acceso y la modificación de la información generada y el motivo por el cual se han escogido las diferentes tecnologías.

En este entregable se va a presentar la interfaz de interacción con la base de datos presentada en el entregable anterior de este mismo paquete de trabajo (entregable 3.1). Para ello se ha elegido la tecnología API REST para la interfaz de interacción con los datos de la aplicación.

Entre las características de REST destaca que es un protocolo cliente/servidor sin estado: cada petición HTTP contiene toda la información necesaria para ejecutarla, lo que permite que ni cliente ni servidor necesiten recordar ningún estado previo para satisfacerla. Las operaciones más importantes relacionadas con los datos en cualquier sistema REST y la especificación HTTP son cuatro: POST (crear), GET (leer y consultar), PUT (editar) y DELETE (eliminar). Los objetos en REST siempre se manipulan a partir de la URI, la cual nos facilita acceder a la información para su modificación o borrado.

La API REST siempre es independiente del tipo de plataforma o lenguaje, lo que ofrece una gran libertad a la hora de cambiar o probar nuevos entornos dentro del desarrollo. Con una API REST se pueden tener servidores en lenguaje: PHP, Java, Python o Node.js. Lo único que es indispensable es que las

respuestas a las peticiones se hagan siempre en el lenguaje de intercambio de información usado, normalmente XML o JSON. En nuestro prototipo el lenguaje de servidor utilizado ha sido Node.js y para el intercambio de datos se ha utilizado JSON.

La plataforma de recogida de datos es la encargada de gestionar y almacenar toda la información generada por el resto de componentes del proyecto. Consta de dos interfaces que lo comunican, **I.2** con la plataforma del Cálculo de Rutas (PTCR) e **I.3** con la aplicación móvil, que envía información recogida del vehículo.

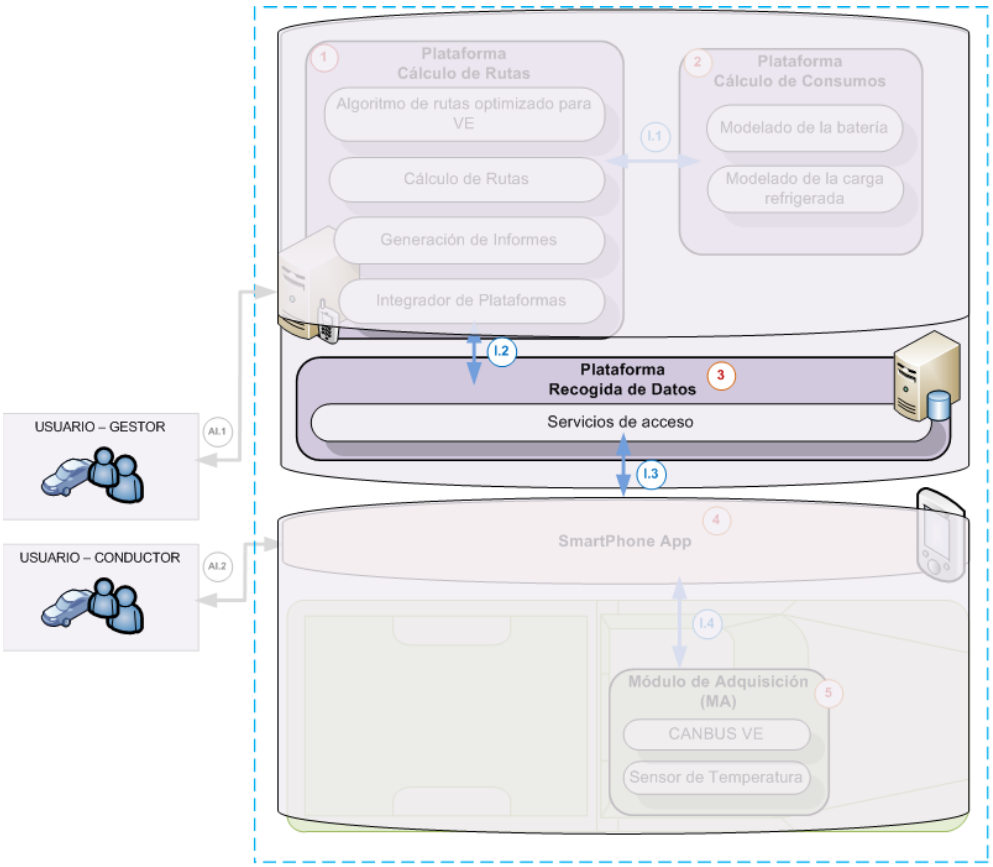

Figura 2: Arquitectura CoolRouting. Componente 3. PTRD

En resumen, el sistema cuenta con una interfaz que le permite modificar los datos de la aplicación desde cualquier dispositivo o cliente que entienda el estándar HTTP. Esto dota de una mayor facilidad y accesibilidad al acceder a los datos y nos evita problemas como el de la escalabilidad del sistema ante un mayor volumen de datos.

4 Interfaz de acceso a los datos (API)

En esta sección se trata la parte del acceso a los datos de la PTRD. La PTRD se realiza mediante lo que se conoce como interfaz de aplicación (API) que permite acceder a los datos gracias al protocolo estándar HTTP.

A cada tabla de la base de datos analizadas en el entregable 3.1, se puede acceder desde una dirección web, gracias a la tecnología REST, para realizar las 4 operaciones típicas, de creación, lectura, actualización y borrado de datos, conocidas como operaciones CRUD. El protocolo HTTP para realizar estas operaciones utiliza los siguientes métodos:

- **GET**: Operación de lectura, devuelve la información seleccionada sin modificarla.
- **POST**: Operación de creación, crea una nueva entrada con la información que se pasa en el cuerpo del mensaje.
- **PUT**: Operación de actualización, modifica toda o una parte de una entrada de información ya existente.
- **DELETE**: Operación de borrado, elimina la información seleccionada.

A continuación, se exponen los enlaces de acceso utilizados en nuestra API. En los enlaces se omite la parte inicial de la URL para una mejor compresión. Los elementos del enlace que contienen un (:) delante son los argumentos o variables y tomarán un valor concreto según el tipo de la variable al que representen, por ejemplo, (/:id) tomará el valor de un numero entero.

Operación	Enlace	Descripción
GET	'/api/v1/vehicles'	Devuelve todos los vehículos de la base de datos
GET	'/api/v1/vehicles/:license'	Devuelve el vehículo con la matrícula pasada como argumento
PUT	'/api/v1/vehicles/update/:id'	Actualiza el vehículo con el id pasado como argumento
DELETE	'/api/v1/vehicles/:id'	Elimina el vehículo con el id pasado como argumento
POST	'/api/v1/vehicles'	Crea un vehículo con los valores indicados en el cuerpo de la petición.
GET	'/api/v1/models'	Devuelve todos los modelos de la base de datos
POST	'/api/v1/models'	Crea un nuevo modelo con los valores indicados en el cuerpo de la petición.
GET	'/api/v1/nodes'	Devuelve todos los nodos de la base de datos
GET	'/api/v1/nodes/:id'	Devuelve el nodo con el id pasado como argumento
PUT	'/api/v1/nodes/update/:id'	Actualiza el nodo con el id pasado como argumento
DELETE	'/api/v1/nodes/:id'	Elimina el nodo con el id pasado como argumento
POST	'/api/v1/nodes'	Crea un nodo con los valores indicados en el cuerpo de la petición.
GET	'/api/v1/orders/:id'	Devuelve el pedido con el id pasado como argumento
GET	'/api/v1/orders'	Devuelve todos los pedidos de la base de datos
PUT	'/api/v1/orders/update/:id'	Actualiza el pedido con el id pasado como argumento
DELETE	'/api/v1/orders/:id'	Elimina el pedido con el id pasado como argumento
POST	'/api/v1/orders'	Crea un pedido con los valores indicados en el cuerpo de la petición.

GET	'/api/v1/canbus'	Devuelve todas las entradas de canbus de la BBDD
GET	'/api/v1/canbus/:id'	Devuelve el canbus con el id pasado como argumento
DELETE	'/api/v1/canbus/:id'	Actualiza el canbus con el id pasado como argumento
POST	'/api/v1/canbus'	Elimina el canbus con el id pasado como argumento
GET	'/api/v1/routes'	Devuelve todas las rutas de la base de datos
GET	'/api/v1/routes/:id'	Devuelve la ruta con el id pasado como argumento
GET	'/api/v1/routes/byvehicle/:vehicleid'	Devuelve las rutas que tiene asignadas el vehículo con el id pasado como argumento.
PUT	'/api/v1/routes/calculate/:route'	Devuelve el cálculo de la ruta óptima de la ruta a partir de los datos de la creación de la ruta.
PUT	'/api/v1/routes/validate/:route'	Marca la ruta como validada por el gestor y está lista para que el conductor la empiece.
PUT	'/api/v1/routes/start/:routeid'	Dada la lectura del canbus, inicia la ruta cambiando el estado y comprobando que se puede realizar.
PUT	'/api/v1/routes/end/:routeid'	Dada la lectura del canbus, indica que la ruta se ha finalizado comprueba que todo haya sido entregado y cambia el estado de la ruta y del vehículo.
PUT	'/api/v1/routes/confirm/:routeid'	Dada la lectura del canbus, se actualiza el estado del pedido en tiempo real.
DELETE	'/api/v1/routes/:id'	Elimina la ruta con el id pasado como argumento
POST	'/api/v1/routes'	Crea una ruta con los valores indicados en el cuerpo de la petición.
GET	'/api/v1/routes/pcc/:routeid'	Envía la información de la ruta a PTCC para actualizar los soc estimados en la ruta.
GET	'/api/v1/alerts'	Devuelve todas las alertas de la base de datos
GET	'/api/v1/alerts/:limit'	Devuelve el número de alertas indicador el valor limit en orden del más reciente al más antiguo.

Tabla 1: Lista de enlaces API REST

4.1 Funciones de la aplicación del conductor para ruta en tiempo real

Una vez la ruta ha sido creada y validada por el gestor, la inserción de los datos en tiempo real pasa a manos del conductor, que mediante la aplicación móvil realizará las siguientes llamadas al servidor. A continuación, se exponen las funciones dedicadas.

4.1.1 Empezar ruta

El conductor debe indicar mediante la aplicación que la ruta va a empezar, esta función comprueba que es posible y actualiza los campos necesarios en la base de datos.

Comprueba	• La PCR da una sola ruta como solución incorporando todos los pedidos. • Status del vehículo es disponible (200) y está en el almacén • La PTCC indica que la batería llega (>10% restante al llegar)
Hace	• Poner status vehículo a En ruta (300). • Inserción en canbus de datos iniciales. • Poner status ruta a En ejecución (300)
Llamada	/api/v1/routes/start/:routeid'
Cuerpo	**Campos** "soc","temp_int", "temp_ext", "temp_arc", "gps_lat", "gps_lng", "gps_error", "VehicleId", "RouteId", "OrderId" = 0

Posibles respuestas:

Server Status	Server Description	Response Code	Response description
200	OK	-	Entrada del canbus
500	Internal server error	1	DB Read - Route
500	Internal server error	2	DB Update - Route
500	Internal server error	3	DB Update- Vehicle
500	Internal server error	4	DB Create - Canbus
500	Internal server error	5	PTCC no funciona en estos momentos
400	Bad request	10	La ruta no ha sido validada por el gestor previamente.
400	Bad request	20	El vehículo no se encuentra disponible o está ya en otra ruta.
400	Bad request	30	Ruta no válida. La batería final no es suficiente para completar la ruta

Tabla 2: Tabla de respuestas de la función empezar ruta

4.1.2 **Entregar pedido**

Cuando el conductor realiza una entrega (sea satisfactoria o no) llamará a esta función indicando el motivo y actualizando los campos correspondientes en la base de datos. Esta función volverá a recalcular la ruta desde donde se encuentra el conductor en ese momento y actualizará los valores en consecuencia.

Comprueba	• Indicar pedido entregado o no desde la app.
Hace	• Actualizar status pedido a entregado (200) o no (500,510,520) • Leer canbus e insertarlo en BBDD con el nodo correspondiente. • Recalcular ruta desde el nodo actual (comprobando los status de los waypoints) para ver si hay cambios en la batería o si el orden de la ruta ha cambiado.
Llamada	/api/v1/routes/confirm/:routeid'
Cuerpo	**Campos:** “status”, “OrderId”, "soc", "temp_int", "temp_ext", "temp_arc", "gps_lat", "gps_lng", "gps_error","VehicleId", "RouteId"

Posibles respuestas:

Server Status	Server Description	Response Code	Response description
200	OK	-	None
500	Internal server error	1	DB Read - Route
500	Internal server error	2	DB Update - Order
500	Internal server error	3	DB Create - Canbus
500	Internal server error	4	GoogleAPI: No es posible calcular la matriz de distancias
500	Internal server error	5	PTCC no funciona en estos momentos
400	Bad request	10	Falta el campo OrderId en el cuerpo del mensage
400	Bad request	20	Falta el campo status en el cuerpo del mensage

Tabla 3: Tabla de respuestas de la función confirmar pedido

4.1.3 **Fin ruta**

Para finalizar la ruta, el conductor debe llamar a esta función que comprueba si todos los pedidos han sido entregados y actualiza el estado final de la ruta en consecuencia. También indica el vehículo como disponible.

Com-prueba	• El estado de los pedidos, no ha quedado ninguno en pendiente
Hace	• Poner status de ruta a finalizado con o sin inconvenientes (400, 450) • Poner vehículo como disponible otra vez (200)
Llamada	/api/v1/routes/end/:routeid'
Cuerpo	**Obligatorios:** "soc","temp_int","temp_ext","temp_arc","gps_lat","gps_lng","gps_error","VehicleId","RoutId","OrderId" = -1

Posibles respuestas:

Server Status	Server Description	Response Code	Response description
200	OK	-	None
500	Internal server error	1	DB Read - Route
500	Internal server error	2	DB Update - Route
500	Internal server error	3	DB Update- Vehicle
500	Internal server error	4	DB Create - Canbus
500	Internal server error	5	DB Update - Order

Tabla 4: Tabla de respuestas de la función terminar ruta

4.2 Seguridad de acceso a los datos

El acceso a los datos requiere además de dos características fundamentales. Por un lado, la seguridad de que la información almacenada en nuestra base de datos está convenientemente protegida y que no cualquier persona que conozca la dirección de petición pueda acceder o modificar esta información. Por otro lado, se requiere de autenticidad, se debe conocer siempre que usuario está realizando la consulta para permitirle o no el acceso a la información en base a sus credenciales.

Para conseguir estás dos características utilizamos una clave única y personal conocida como “apikey”, distinta a la contraseña normal del usuario, que añadiéndola a sus peticiones (vistas en la sección 5.0) le permite acceder o modificar la información de la base de datos.

* En este proyecto han colaborado por parte del ITE: Caterina Tormo Domènech, Christian Conca de la Asunción, Ignacio Javier Benítez Sánchez, Julio César Díaz Cabrera, Juan Carlos Rojas Mestre, Celeste Martínez Catalán, Ricardo Ridaura Belenguer, Esther Mocholí Munera, Alejandro Almela Frechina; Por parte de ITENE: Emilio González Viosca, Enrique De La Cruz Navarro, Miguel Angel Alferez Moreno, Jose Ángel Rodríguez Álvaro, Liseth Monticone, Sergio Güerri, Carla Cots Renau