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

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

E3.1. Esquema base de datos

ITENE

Información del documento	
Titulo	Esquema base de datos
Participantes	ITE (coordinador) ITENE
Description	Estructura de 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	15/12/2016

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
v0.3	28/11/2016	Caterina Tormo Domènech	Revisión y comentarios
v0.4	01/12/2016	José Ángel Rodríguez Álvaro Rubén Ponce Tortajada	Añadida sección 5 y ampliación de introducción
v0.5	07/12/2016	Caterina Tormo Domènech	Revisión y comentarios
vF0.0	14/12/2016	José Ángel Rodríguez Álvaro	Versión final
vF0.1	15/12/2016	Christian Conca de la Asunción	Revisión y comentarios
vF0.3	09/01/2016	José Ángel Rodríguez Álvaro	Añadidas funcionalidades
vF0.4	20/02/2017	José Ángel Rodríguez Álvaro	Añadidos cambios en la plataforma
vF0.5	06/04/2017	José Ángel Rodríguez Álvaro	Versión definitiva
vF0.6	23/05/2017	Caterina Tormo Domènech	Actualización en base a la segunda anualidad. Versión definitiva

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
SQL	Structured Query Language
NoSQL	Non-structured Query Language
JSON	JavaScript Object Notation
PTCC	Plataforma de Cálculo de Consumos
API	Application Programming Interface
REST	REpresentational State Transfer
HTTP	HyperText Transfer Protocol
BSD	Berkeley Software Distribution

2 Sumario

Este entregable presenta los resultados de la tarea 3.1, Esquema de base de datos, que pertenece al paquete de trabajo 3.

El documento se ha organizado en diferentes secciones con la finalidad de explicar la estructura de los datos que forman la solución software mediante una base de datos relacional.

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 generados por el resto plataformas del proyecto. Tanto los datos generados por la trazabilidad de los vehículos, las rutas óptimas o la información para la gestión y funcionamiento 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 la estructura de la información almacenada en la base de datos y el motivo por el cual se han escogido las diferentes tecnologías implicadas.

El esquema de base de datos escogido consiste en un híbrido entre una base de datos relacional clásica (SQL) y el de una base de datos NoSQL, tratando de obtener lo mejor de las dos tecnologías. Por un lado, la base de datos relacional se encarga de la información más persistente, la que modificaremos pocas veces. Las ventajas de este tipo de BBDD es que tienen una estructura y unas relaciones entre los datos bien definidas. Por otro lado, la necesidad de una mayor velocidad de computación y una mayor versatilidad en la estructura de los datos, nos lleva a elegir un formato de almacenamiento NoSQL, el cual utilizaremos para almacenar los datos de la optimización de rutas.

Con este propósito se ha elegido **PostgreSQL** como sistema de gestión de bases de datos. Ya que, aunque está diseñado para una gestión de bases de datos objeto-relacional, también admite el uso de objetos JSON que nos permite cierta libertad en la estructura de los datos.

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará al resto y el sistema continuará funcionando. Su desarrollo comenzó hace más de 16 años, y durante este tiempo, estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema. Está distribuido bajo licencia BSD y con su código fuente disponible libremente.

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.

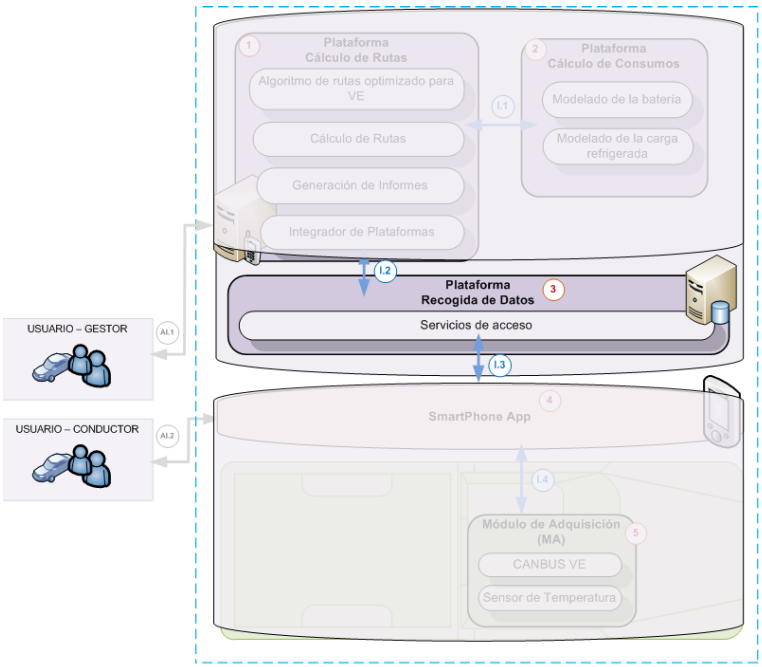

Figura 2: Arquitectura CoolRouting. Componente 3. PTRD

Por tanto, es vital la flexibilidad a la hora de acceder a ella y la velocidad en la respuesta de sus operaciones.

En cuanto a la tecnología para el acceso a los datos de la aplicación se ha escogido la tecnología API REST. Es decir, el programa 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 simpleza y accesibilidad al acceder a los datos y nos evita problemas como el de la escalabilidad del sistema ante un mayor volumen de datos. Está tecnología de acceso a los datos se describirá ampliamente en el entregable 3.2.

4 Estructura de la base de datos

4.1 Análisis de los casos de uso

En base a los casos de uso definidos en E1.2. se ha realizado un análisis de los específicos del componente 4.

Cuando los gestores (AI1) decidan realizar una planificación de ruta para su reparto de mercancía a través de CoolRouting, ésta se llevará a cabo a través de una interfaz amigable dónde introducir la in-formación para poder obtener las pautas necesarias para ejecutarlo.

INTERFACES			CÁLCULO DE RUTA	SERVICIOS EN RUTA		SEGURIDAD
			CUCR0100 Cálculo de Ruta	CUSR0200 Servicios en Ruta		CUS0300 Seguridad
			CUCR0101	CUSR0201	CUSR0202	CUS0301
I.1	PTCR	PTCC	X		X	
I.2	**PTCR**	**PTRD**	**X**	**X**	**X**	
I.3	**PTRD**	**Smartphone**		**X**	**X**	**X**
I.4	Smartphone	MA		X	X	
AI.1	Usuario Gestor	PTCR	X			X
AI.2	Usuario Conductor	Smartphone		X	X	X

Tabla 1. Interfaces vs Casos de Uso

En resumen, la plataforma de recogida de datos interviene en todos los casos de uso, cálculo de rutas, servicios en ruta y seguridad. Puesto que por la arquitectura del sistema es la encargada de comunicar la plataforma con el sistema del vehículo (App). Con lo que los casos de uso involucrados son CUCR0101, CUSR0201, CUSR0202 y CUS0301.

En base a los casos de uso analizados la base de datos se estructura en 8 tablas:

- **Nodes**: hace referencia a la información relativa a las localizaciones de los lugares de interés de nuestra aplicación; pueden ser tanto ubicaciones de clientes como de almacenes.
- **Orders**: encargada de recoger la información relativa a los pedidos de los clientes.
- **Routes**: encargada de almacenar la información relativa a las rutas.
- **Models**: almacena los modelos de vehículo admitidos en nuestro sistema, más concretamente algunas de sus características como capacidad o batería máxima.
- **Users**: esta tabla es la encargada de almacenar los datos de los usuarios registrados que tendrán acceso a la aplicación, así como los atributos cifrados para su acceso.
- **Vehicles**: encargada de almacenar la información relativa a los vehículos
- **Canbus**: almacena los datos recogidos por el módulo de adquisición del vehículo, más la posición gps, más la temperatura del arcón y el vehículo, en un momento concreto.
- **Alerts**: almacena las alertas de las acciones realizadas por un usuario en la aplicación.

4.2 Esquema global de la base de datos

Las diferentes relaciones entre tablas pueden variar según cada caso. Por un lado, puede ser una relación opcional u obligatoria, esto implica la necesidad o no de que cada registro de esa tabla esté relacionado con otra tabla. La representación de este tipo de relación se muestra mediante un circulo blanco en el extremo cuando es opcional y con 1 o 2 líneas perpendiculares cuando es obligatoria. Además, también distinguiremos entre el número de elementos relacionados entre las tablas pudiendo ser:

- **Relaciones 1 a 1**: Cada registro de la tabla A se relaciona con un único registro de la tabla B y cada registro de la tabla B sólo se relaciona con un elemento de la tabla A.
- **Relaciones 1 a muchos:** Cada registro de una tabla A, puede estar enlazado con más de un registro de otra tabla B. En cambio, cada registro de la tabla B sólo puede estar enlazado a un registro de la tabla A.
- **Relaciones muchos a muchos:** Cada registro de la tabla A se le pueden asociar varios registros de la tabla B y cada registro de la tabla B puede estar relacionado con más de un registro de la tabla A.

Representaremos el extremo único con una solo línea en la tabla y el extremo muchos cuando la línea se divide en 3. Como ejemplo, la relación entre la tabla “Vehicles” y “Models” es 1 a muchos ya que un vehículo solo puede ser de un modelo, pero puede haber muchos vehículos de ese modelo.

A continuación, en la figura 3 se muestran las tablas con sus atributos y las relaciones entre ellas, esta información es la que nos permite estructurar la información de nuestra base de datos.

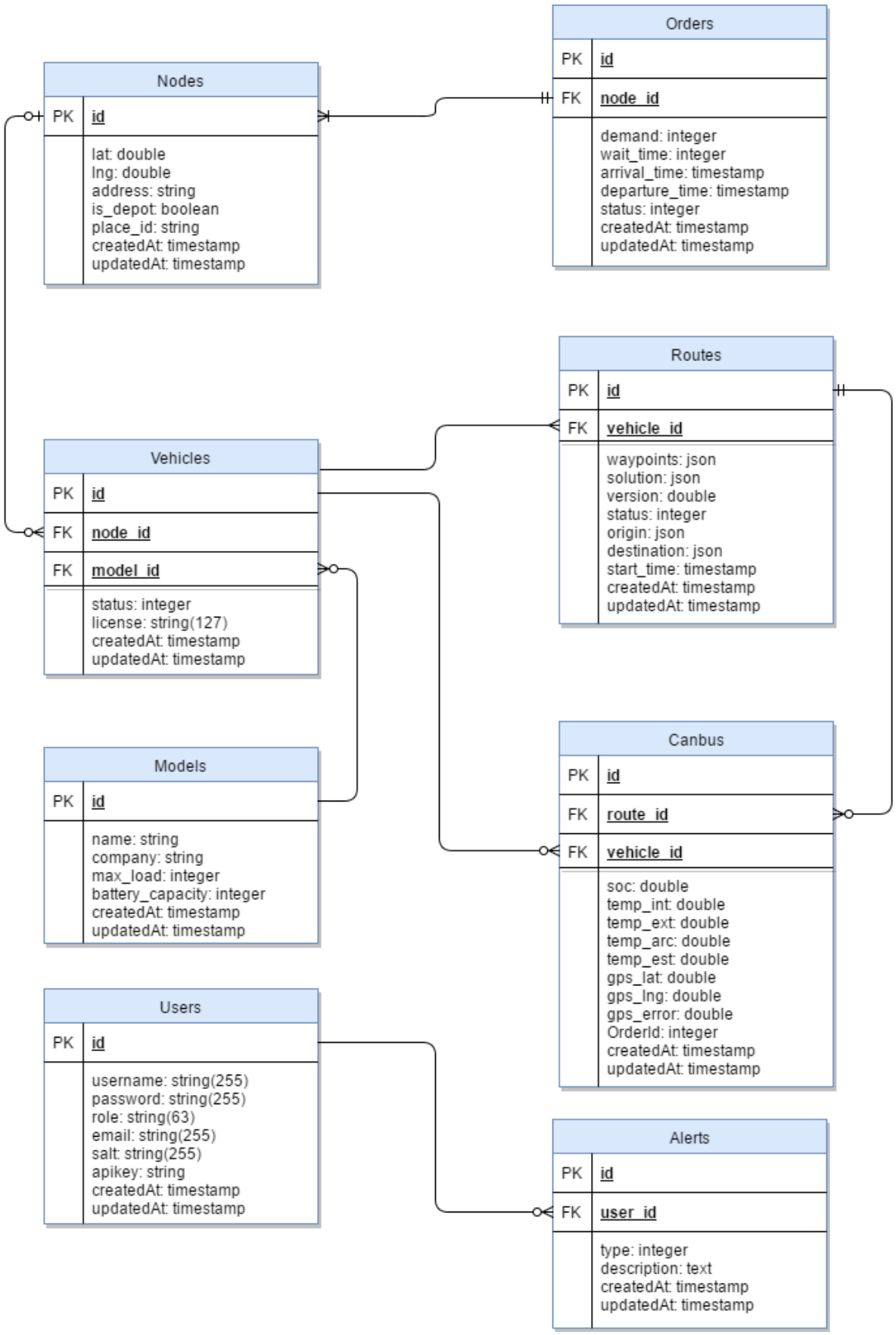

Figura 3: Esquema de la BBDD

4.3 Especificación de los campos de la BBDD

En esta subsección se detallan todos los campos (o tablas) de la base de datos, indicando su tipo y una breve descripción para indicar para que sirven. Estas tablas están formadas por atributos que admiten un valor de un tipo determinado, como por ejemplo numérico (integer) o texto (string). En el caso de algunos atributos se ha optado por una representación de un código numérico formado por 3 cifras que representa el estado de ese objeto en particular. No se han incluido los campos que comparten todas las tablas ("createdAt" y "updatedAt") que indican la fecha y hora en que se crea o actualiza respectivamente, una entrada concreta en esa tabla de la BBDD.

4.3.1 Especificación de la tabla "Nodes"

Esta tabla es la encargada de recoger la información relativa a las localizaciones de los lugares de interés de la aplicación; pueden ser tanto clientes como almacenes.

Variable	Tipo	Descripción
id	Integer	Identificador del nodo
lat	Long	Coordenada de latitud
lng	Long	Coordenada de longitud
is_depot	Boolean	Indica si es cliente o almacén
address	String	Nombre de la dirección del lugar
google_id	String	Localizador de google del lugar

Tabla 1: Especificación de la tabla Nodes.

4.3.2 Especificación de la tabla "Orders"

Esta tabla es la encargada de recoger la información relativa a los pedidos de los clientes.

Variable	Tipo	Descripción
id	Integer	Identificador del pedido
NodeId	Integer	Clave ajena a la tabla Nodes que indica la localización del pedido
demand	integer	Cantidad necesaria de cajas para transportar el pedido
arrival_time	timestamp	Tiempo de inicio de la ventana horaria para la entrega
departure_time	timestamp	Tiempo de fin de la ventana horaria para la entrega.
wait_time	integer	Tiempo de espera aproximada en la entrega (segundos)
status	Integer	Estado del pedido

Tabla 2: Especificación de la tabla Orders.

El campo status podrá adoptar los siguientes valores en base al estado del pedido al cliente:

Estados ruta posibles	Descripción
100	Pendiente de asignación
150	Pedido reservado para una ruta concreta
200	Entregado correctamente
500	Error en la entrega
510	No se ha podido entregar porque el cliente no estaba disponible
550	No se ha podido entregar por problema en la ruta

Tabla 3: Especificación del código de estados de los pedidos

4.3.3 Especificación de la tabla "Vehicles"

Esta tabla es la encargada de almacenar la información relativa a los vehículos.

Variable	Tipo	Descripción
id	Integer	Identificador del vehículo
node_id	Integer	Clave ajena a la tabla Nodes que indica el lugar donde se encuentra el vehículo.
model_id	Integer	Clave ajena a la tabla Model que indica el modelo del vehículo seleccionado.
license	String	Indica la matrícula del vehículo
status	integer	Código de 3 dígitos que indica el estado actual del vehículo

Tabla 4: Especificación de la tabla Vehicles

El campo status puede adquirir los siguientes valores según el estado del vehículo al que se refiere:

Estados vehículo posibles	Descripción
200	Vehículo disponible para asignar a una ruta
300	Vehículo en ruta
400	Vehículo no disponible

Tabla 5: Especificación del código de estados del vehículo

4.3.4 Especificación de la tabla “Models”

Esta tabla es la encargada de almacenar los modelos de vehículo admitidos en nuestro sistema, así como su capacidad máxima.

Variable	Tipo	Descripción
id	Integer	Identificador del modelo
name	String	Nombre específico del modelo
company	String	Nombre de la marca a la que pertenece el vehículo
max_load	Integer	Capacidad espacial máxima del vehículo en número de cajas
battery_capacity	Integer	Capacidad máxima de la batería (dada por el fabricante)

Tabla 6: Especificación de la tabla Models.

4.3.5 Especificación de la tabla “Users”

Esta tabla es la encargada de los datos de los usuarios registrados que tendrán acceso a la aplicación, así como los atributos cifrados para su acceso.

Variable	Tipo	Descripción
id	Integer	Identificador del usuario
username	String	Nombre de acceso al sistema del usuario
password	String	Contraseña cifrada para el acceso del usuario
role	String	Rol del usuario; limita las opciones de acceso que tendrá el usuario en la aplicación
email	String	Correo electrónico con el que se registró el usuario
salt	String	Código de seguridad para el cifrado de la contraseña
apikey	String	Código para autentificar la identidad del usuario que realiza las peticiones a la API

Tabla 7: Especificación de la tabla Users

4.3.6 Especificación de la tabla “Alerts”

Esta tabla es la encargada de almacenar las alertas de las acciones realizadas por un usuario en la aplicación, de está forma queda un registro de sus acciones y puede mostrarse al resto de usuarios.

Variable	Tipo	Descripción
id	Integer	Identificador del registro Alerts
type	Integer	Indica el tipo de alerta
description	Text	Descripción de la alerta.
User_id	integer	Clave ajena a la tabla “Users” para la asignación de la Alerta a un usuario concreto.

Tabla 8: Especificación de la tabla Alerts

4.3.7 Especificación de la tabla “Routes”

Esta tabla es la encargada de almacenar la información relativa a las rutas, tanto los datos de la ruta como los resultados del algoritmo de enrutado.

Variable	Tipo	Descripción
id	Integer	Identificador de la ruta
origin	Integer	Clave ajena a la tabla Nodes que indica el lugar de posición inicial de la ruta(debe ser un almacén)
vehicle_id	Integer	Clave ajena a la tabla Vehicles que indica el vehículo que va a realizar la ruta
waypoints	Json	Información inicial y sin orden de los clientes que tiene que servir esta ruta.
solution	Json	Información después de calcular la ruta optimizada con el orden que debe seguir el vehículo para cumplir todas las restricciones.
version	Double	Indica la versión correspondiente del algoritmo de enrutado utilizado cuando se calculó la solución.
status	integer	Código de 3 dígitos que indica el estado actual de la ruta
start_time	Date	Hora planeada de inicio de la ruta

Tabla 9: Especificación de la tabla Routes

El campo status podrá adoptar los siguientes valores en base al estado de la ruta en cada momento:

Estados ruta posibles	Descripción
100	Condiciones de ruta insertadas, ningún cálculo realizado
150	Ruta optimada validada, a falta de ser aceptada por el gestor
200	Ruta validada por el gestor y lista para ser empezada
300	Ejecución de la ruta en marcha
400	Ruta finalizada con éxito
450	Ruta finalizada con alguna entrega errónea o sin realizar
500	Ruta cancelada por el gestor
510	Ruta cancelada por imposibilidad de realizarse en una sola ruta (PCR)
520	Ruta cancelada por insuficiencia de batería (PTCC)
550	Ruta cancelada por problema en el vehículo

Tabla 10: Especificación del código de estados de la ruta

4.3.8 Especificación de la tabla “Canbus”

Esta tabla es la encargada de almacenar los datos recogidos por el módulo de adquisición del vehículo y posición GPS, en un momento concreto.

Variable	Tipo	Descripción
id	Integer	Identificador del registro Canbus
soc	Integer	Porcentaje de la batería actual del vehículo [0-100]
temp_int	double	Temperatura en el interior del vehículo
temp_ext	double	Temperatura en el exterior del vehículo (ambiente)
temp_arc	double	Temperatura en el interior del arcón
temp_des	double	Temperatura deseada en el arcón
gps_lat	double	Coordenada gps de latitud del vehículo
gps_lng	double	Coordenada gps de longitud del vehículo.
gps_error	double	Margen de error que puede tomar el gps
VehicleId	Integer	Clave ajena a la tabla “Vehicles” para la asignación de la entrada de canbus a un vehículo concreto.
OrderId	Integer	Id del pedido que ha entregado en el caso de que lo haya (opcional)

Tabla 11: Especificación de la tabla Canbus

5 Conclusiones

En conclusión, tenemos un sistema de gestión de la información para nuestra aplicación robusto, seguro y flexible. Por un lado, la base de datos utiliza tecnología PostgreSQL cuyo uso está muy extendido en todo el mundo y por tanto su funcionamiento está muy optimizado y comprobado. El sistema es seguro, porque el acceso a los datos de nuestra aplicación requiere de autorización para cada una de las consultas, la cual podemos revocar o limitar para ciertos usuarios en cualquier momento y de forma sencilla.

A destacar también la flexibilidad debido al tipo de desarrollo realizado, que separa el cliente del servidor lo que permite modificar una u otra parte independientemente, modificar una no afecta a la otra siempre que se respete la estructura de la comunicación entre ambas. Sin olvidarnos de la escalabilidad que ofrece esta arquitectura ya que puede crecer el tamaño de la base de datos sin afectar a como se accede a estos.

* En este proyecto han colaborado por parte del ITE: Christian Conca, Caterina Tormo; Por parte de ITENE: Enrique De La Cruz Navarro, Jorge León Bello, Miguel Angel Alferez Moreno, Vicente Villagrasa Blanco, Ruben Ponce Tortajada, Jose Angel Rodriguez Alvaro