****

**IES Maestre de Calatrava**

Paseo de la Universidad,1

13005-Ciudad Real

**F.P. INFORMÁTICA**

**CURSO 2018-19**

**Marzo - Diciembre**

**Convocatoria: Ordinaria**

**Técnico Superior en**

**Desarrollo de Aplicaciones Multiplataforma**

MEROTrack

**Nombre Alumno: Alberto Menchén Ruiz**

**DNI: 20616193M**

**Todos los apartados son obligatorios**

Se recomienda que el trabajo a realizar se desarrolle con arreglo al siguiente guion:

* Portada (**ANEXO** I)
* Índice de contenidos paginado.

**APARTADO I:** (Contenidos vinculados a los módulos de la especialidad de FOL)

1. **Definición de la Idea de Negocio** (Factor diferenciador y Utilidad del mismo)
2. **Análisis del Entorno de la Empresa.**
3. **DAFO y CAME**
4. **Viabilidad Económica y Técnica.**
5. **Trámites de puesta en marcha e impuestos a los que estás sujeto.**
6. **Política de prevención de Riesgos laborales en la Empresa**. (Riesgos expuestos-medidas preventivas adoptadas)

APARTADO II:

# TITULO

MEROTrack (Menchén-Romero-Transportes/Tracker)

# INTRODUCCIÓN (Español)

MEROTrack es una aplicación web dinámica que permite, desde un panel de control en la web, mantener un seguimiento mediante geolocalización una flota de camiones y otros vehículos de transporte.

El tema de este proyecto fue elegido por la cercanía con el mundo del transporte, ya que toda mi familia se dedica en mayor o menor medida a esta actividad.

También fue elegido por la necesidad del transporte en el mundo moderno. Sin importar el producto, todo aquello que se produce ha de ser transportado. Las materias primas han de llevarse a las fábricas, y a su vez, los productos que estas fábricas generan han de ser distribuidos bien directamente al cliente, o a las tiendas donde se vayan a vender. Aunque no lo ‘veamos’, el transporte está detrás de todo aquello que consumimos y compramos.

Por tanto, una de las primeras necesidades que cubre MEROTrack es la gestión de viajes y repartos, haciendo especial hincapié en la optimización de rutas, a fin de ahorrar al máximo posible el gasto en combustible y más importante aún, tiempo, a la empresa (y al cliente).

La otra necesidad a cubrir es el control de la posición de vehículos, por dos principales factores:

* **El robo de vehículos, o su mercancía.** El cual es más común de lo que se piensa, al mantener geolocalizado el mismo, hay más posibilidades de recuperar el vehículo tras su robo.
* **Fraude.** Algunos conductores hacen ver como que trabajan o hacen más kilómetros de lo que realmente hacen, alargando los tiempos de descanso en enorme medida. Mantener un control en tiempo real, e histórico, de la posición de su vehículo ayuda en la evaluación del rendimiento de nuestros conductores.

# INTRO (English)

*MEROTrack is a dynamic webapp that allows, thanks to its web-based dashboard, keep track of a cargo vehicle fleet.*

*This topic has been chosen because of close to my family transportation is, due to the fact that almost all my entire family works in this field.*

*It has also been chosen because of how important the transport field is nowadays. No matter the product, everything once it’s produced must be carried around. From raw materials, which must be delivered to factories; to finished products, which must be delivered to the customer. Even if we don’t usually get to see the backstage, transportation is always behind every single thing we buy and consume.*

*Because of this, one of the first needs that MEROTrack must fulfill, is the management of trips and deliveries, with special interest in route and delivery optimization, achieving economical and time savings to the deliverer and the deliveree.*

*The second, and equally important neccessity to cover, is the management of the delivery vehicles themselves, for two main reasons:*

* ***Theft of the vehicle or its cargo.*** *More common than what it’s usually though, thanks to the integrated geotracker that sends its position in real-time, we can easily recover the stolen goods.*
* ***Fraud****. Some drivers pretend to work more time, or drive more miles than what they really do, by knowing where they are, we can effectively measure their actual performance.*

# RECURSOS MATERIALES Y PERSONALES PARA LA REALIZACIÓN DEL PROYECTO

A nivel personal, el recurso a aportar obviamente ha sido el tiempo, el cual estimo en unas 150 horas.

El módulo hardware encargado de la geolocalización de los vehículos, el rastreador GPS, está compuesto de una controladora programable Arduino modelo UNO v3, al cual se ha acoplado una *shield* marca DFRobotcon en el chip Sim908, el cual permite acceder a la red telefónica e internet mediante una tarjeta SIM, y obtener posiciones GPS mediante una antena integrada.

Debido a que el proyecto no se va a llevar a producción, y no hay una carga de trabajo importante, todo el software se ejecuta en un ordenador personal, y no se ha montado ningún servidor dedicado.

Se ha optado principalmente por software de código abierto para evitar costes y problemas de licencias. Algunos programas, librerías o frameworks utilizados son:

* BD PostgreSQL
* Backend en Spring BOOT
  + Spring Initializr para la inicialización del proyecto backend
  + Spring Data REST para exponer rutas del modelo al cliente
  + Spring Data JPA para el acceso a BD
  + Spring Security para la securización del API
  + Spring Web
  + Gradle para gestión de dependencias Java
  + Lombok
* Frontend en Angular 7
  + Angular4-Hal para interactuar con el backend
  + Nebular, componentes basados en bootstrap
  + Ngx-Admin como principal plantilla de diseño e interfaz
  + PrimeNG, algunos componentes
  + Leaflet como componente para visualización de mapas
    - Leaflet Routing Machine
    - Leaflet Control Geocoder

Los entornos y utlidades de desarrollo empleadas son, principalmente:

* Arduino IDE
* IntelliJ IDEA
* VSCode
* Angular CLI
* Docker
* Postman
* PgAdmin

# PRESUPUESTO ECONÓMICO

No se ha invertido una gran cantidad de recursos en la realización del proyecto.

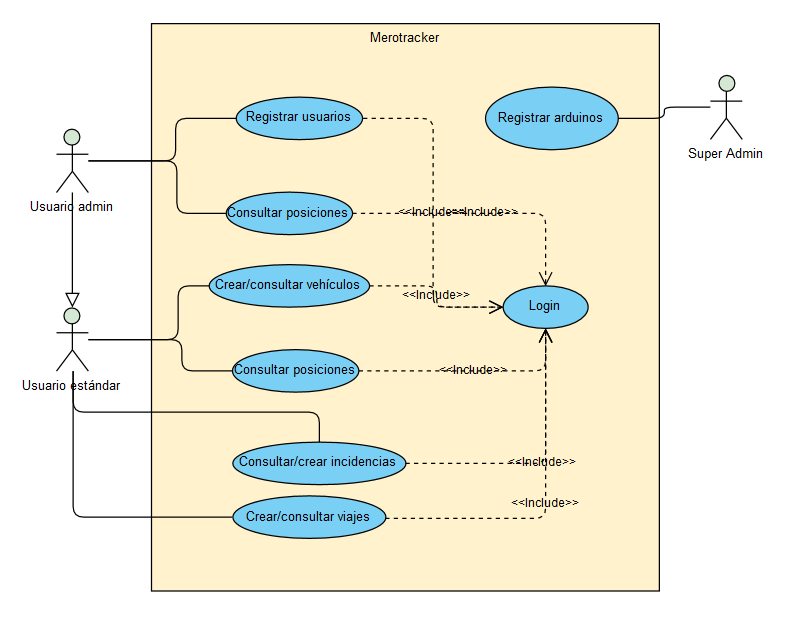
Debido a que no se ha llevado a producción, no ha sido necesario invertir en un servidor dedicado.

El resto de materiales estaban ya comprados antes del inicio del desarrollo. El portátil (~1000€ aprox), la arduino (~40€), y la shield SIM908 (~90€ en el momento de su compra, ahora 40€)

Los servicios de mapas son provistos por Mapbox; sin embargo, debido a que no se ha alcanzado el límite mensual de consultas, no ha supuesto ningún coste.

# FASES Y SECUENCIACIÓN DE LAS ACTIVIDADES DEL PROYECTO

## DIAGRAMA CASOS DE USO



Existen dos principales roles de uso, siendo el ‘Super admin’ el administrador total del sistema, que será el único que tenga acceso (directamente de la BD) a los identificadores privados y públicos de cada vehículo almacenado en la BD. Cabe destacar que aunque se puedan crear vehículos en el panel web, estos no enviarán ninguna posición al ser necesario registrar previamente la Arduino que se encarga de la geolocalización.

El usuario ‘estándar’ es aquel usuario que puede acceder a los menús de creación/edición/eliminación de viajes, vehículos e incidencias; sin embargo, no puede acceder a la vista de posiciones ni a la de gestión de usuarios por motivos de seguridad.

El administrador, junto a las acciones a las que puede acceder el usuario estándar, sí que puede consultar la vista de posiciones (tanto la vista de flota, que muestra la última posición conocida de cada vehículo, como la vista histórica, que muestra las posiciones de un vehículo concreto en un rango de fechas). Y también puede acceder al panel de gestión de usuarios.

## DIAGRAMA DE CLASES UML

## Spoiler: No hay

### FRONTEND

El paradigma de ‘programación’ que sigue Angular, si bien usa clases ‘tradicionales’ en algunas partes, la inmensa mayoría del código se realiza mediante *Componentes*, los cuales están declarados por *Módulos,* y a los que se pueden inyectar *Servicios*.

Los componentes a su vez implementan Templates, Styles, y pueden implementar a su vez Directivas de entrada/salida.

Además de eso, un componente puede estar formado por varios archivos: su código, su template HTML, su hoja de estilos independiente…

Angular no sigue una metodología de programación que de pie a un Diagrama UML, debido a su complejidad, el hecho de que puede hacer uso de infinidad de librerías y componentes de terceros (en este proyecto se ha usado Nebular, PrimeNg y Leaflet) y el hecho de que muchos de los componentes son inyectados directamente con clases propias de Angular como el HTTPClient, por poner un ejemplo.

### BACKEND

Spring, pese a ser un framework de Java, tampoco hace un uso ‘tradicional’ de las clases. Gracias a las anotaciones (Java 1.5+), Spring genera en tiempo de compilación gran parte de la implementación y se encarga automáticamente de la inyección de dependencias y la configuración general de la aplicación. Esta forma de desarrollar tampoco daría pie a un diagrama UML al uso.

Aunque no haya una relación directa entre las múltiples partes de la aplicación Spring durante el desarrollo, el proyecto mantiene la siguiente estructura:

* Modelo: Los POJOS, los cuales contienen los datos de las columnas de las entidades de la BD y su mapeo JPA
* Repositorio: Permiten el acceso a datos, y exponen rutas REST de forma automática.
* Controllers: Para operaciones específicas, sobrescriben algunas rutas REST.
* Specifications: Contienen filtros de búsqueda para las consultas a la BD.
* Security: Configuraciones varias de seguridad y filtros de autorización/autenticación.

### ARDUINO

El programa que controla la Arduino, el rastreador, está escrito en C, no C++, el cual directamente no tiene soporte para clases.

## DIAGRAMA E/R DE LA BASE DE DATOS

METER AQUÍ EL DIAGRAMAAAAAAAAAAAAAAAAAAAAAAAAAA

## INTERFACES GRÁFICAS

## DESARROLLO DEL CÓDIGO E INTEGRACIÓN CON LA BASE DE DATOS

### PRUEBAS DE SOFTWARE