# Bases de datos avanzadas

# Práctica 1: GIS

Juan Casado Ballesteros

Gino Cocolo



# Índice

[Introducción 3](#_Toc33437967)

[Áreas FIR/UIR 3](#_Toc33437968)

[Áreas TMA 3](#_Toc33437969)

[Rutas de los aviones 3](#_Toc33437970)

[Manejo de datos espaciales 3](#_Toc33437971)

[Visualización de datos espaciales 4](#_Toc33437972)

[Trabajo extra 4](#_Toc33437973)

[Diseño de la base de datos 5](#_Toc33437974)

[Tablas y columnas 5](#_Toc33437975)

[Inserción de datos 5](#_Toc33437976)

[Cálculos realizados en la base de datos 5](#_Toc33437977)

[Acceso a la base de datos 6](#_Toc33437978)

[Realización de las consultas 6](#_Toc33437979)

[Interfaz REST 6](#_Toc33437980)

[Servidor 6](#_Toc33437981)

[Diseño de la Interfaz web 7](#_Toc33437982)

[Open Layers 7](#_Toc33437983)

[React 7](#_Toc33437984)

[Conexión con la base de datos 8](#_Toc33437985)

[Acceso a la interfaz 8](#_Toc33437986)

[Otras formas de visualizar Los datos 9](#_Toc33437987)

[QGIS 9](#_Toc33437988)

[Despliegue de la aplicación 10](#_Toc33437989)

[Docker 10](#_Toc33437990)

## Introducción

Nuestro objetivo en esta práctica es el de crear una aplicación capaz de calcular el coste de las rutas de los aviones en función de la trayectoria recorrida.

Para lograrlo deberemos manejar los siguientes tipos de datos de datos:

#### Áreas FIR/UIR

Cubren diversas regiones de la Península y de Canarias. Todo avión dentro de estas áreas deberá pagar en función de la siguiente ecuación. Un avión solo puede ser cobrado por estar en una de estas áreas a la vez.

Cabe destacar que los aviones pagarán más cuanta más distancia recorran en estas áreas.

PRECIO = [51,08 si Península o 43,73 si Canarias]\*KM/100\*(TONELADAS\_METRICAS/50) ^ 0,5

#### Áreas TMA

Cubren regiones próximas a los aeropuertos. Estas áreas son en realidad volúmenes con altura máxima en los 20.000 pies. Un avión solo puede estar ser cobrado por estar en una de estas áreas a la vez, si además está en un área FIR/UIR será cobrado como si solo estuviera en TMA.

Cabe destacar que no importa la distancia que se recorra dentro de las áreas TMA.

PRECIO = [18,72 o 16,84 o 14,04 según aeropuerto]\*(TONELADAS\_METRICAS/50) ^ 0,7

#### Rutas de los aviones

Nos importa la ruta que han recorrido, en concreto la lista de puntos por los que han pasado para poder calcular dentro de qué área estaban, así como la distancia recorrida dentro de cada una de ellas. Adicionalmente nos importa también la altura en las que estaban en cada uno de los puntos.

También necesitaremos conocer el avión que realizó la ruta para poder saber sus MTWO o peso máximo autorizado en el despegue en toneladas métricas.

### Manejo de datos espaciales

Como podemos ver muchos de los datos que manejaremos tiene referencias espaciales, en concreto los puntos que definen las áreas y los puntos que definen la trayectoria de los aviones.

Utilizaremos por tanto una base de datos capaz de manejar datos con referencias espaciales para que nos ayude a almacenar los datos, así como a obtener las relaciones entre ellos. En nuestro caso utilizaremos la base de datos Postgre con su extensión GIS postgis.

La ventaja de utilizar un sistema GIS recae en tres pilares fundamentales:

* **Estandarización**: según hemos estado trabajando con la base de datos y la hemos ido integrando con otras partes de nuestra aplicación hemos comprobado la sencillez de hacer esto debido a que todos los componentes eran capaces de entender los formatos en que estaban expresadas las referencias espaciales.
* **Adaptación**: Una herramienta GIS está ya preparada para trabajar con referencias espaciales de forma óptima sin que debamos de hacer ninguna configuración adicional sobre ella. En la documentación de postgis hemos encontrado que una técnica recurrente que utilizan son los árboles quad-tree para indexar los datos que contiene referencias espaciales. Este tipo de índices utilizan nodos con cuatro hijos que subdividen regiones de espacio un cuarto menor que las que su padre cubría.

También es digno de ser mencionado lo sencillo y cómodo que resulta almacenar los datos espaciales en postgis utilizando el tipo geom para las columnas de las tablas.

* **Reutilización**: la mayoría de las operaciones que hemos deseado realizar sobre los datos espaciales estaban ya implementadas en postgis. Adicionalmente explorando la documentación hemos encontrado otras operaciones que, aunque no nos han hecho falta para completar la práctica nos han parecido de gran utilidad.

### Visualización de datos espaciales

Adicionalmente desarrollaremos una aplicación web que nos permita manejar y visualizar los datos desde una interfaz cómoda. Dicha aplicación consistirá en una pequeña interfaz creada con React sobre la que incorporamos las librerías de Open Layers y Turf.js para visualizar los datos que provienen de la base de datos sobre un mapa.

Los mapas que utilizamos provienen de Open Street Maps, un proveedor de mapas gratuito cuyos mapas son editables por cualquier persona. Explorando este proveedor de información hemos descubierto que no solo proporciona mapas si no que también proporciona otros muchos tipos de datos como edificios, puntos de interés en las ciudades o carreteras.

### Trabajo extra

* Hemos probado otras herramientas GIS, en concreto QGIS, comprobando que se integraban adecuadamente con nuestra base de datos.
* Todas las partes de la aplicación han sido desarrolladas dentro de contenedores Docker. Esto nos ha permitido trabajar sin tener problemas de instalación de las distintas herramientas y sin tener problemas de versiones o dependencias.
* Hemos desarrollado un servidor web que crea una interfaz REST frente a la base de datos. Dicho servidor es un Tomcat que sirve una aplicación JAVA que mediante Sprint crea la interfaz REST. El uso de este servidor nos ha facilitado en gran medida el trabajo pues es un intermediario que simplifica el acceso a los datos almacenados en la base de datos desde la interfaz web.

## Diseño de la base de datos

Introducción

### Tablas y columnas

Explicación de las decisiones de diseño tomadas

### Inserción de datos

Formato de los datos para ser introducidos

### Cálculos realizados en la base de datos

Cómo se realizan, conversiones de datos…

## Acceso a la base de datos

Introducción

### Realización de las consultas

JDBC u otros, almacenamiento de las consultas en xml…

### Interfaz REST

Sprint

### Servidor

Tomcat y war

## Diseño de la Interfaz web

Para crear la interfaz web hemos decidido utilizar Open Layers. Esta es una librería capaz de visualizar mapas en una página web, así como de manejar datos espaciales. Para tratar los datos espaciales hemos también instalado y probado Turf.js. No obstante, debido a que Open Layers proporciona por si sola todas las características de las que requería nuestra aplicación finalmente Turf.js no ha sido utilizada.

Hemos comprobado que las características que proporciona Open Layers aunque excelentes para crear mapas web interactivos necesitan de apoyo externo para poder crear una interfaz. Es por ello por lo que hemos decidido utilizar también React.

### Open Layers

Open Layers nos ha permitido visualizar sobre un mapa obtenido de OSM los datos almacenados en la base de datos. Adicionalmente hemos profundizado algo más en el uso de esta librería permitiendo a los usuarios interactuar con los datos que se les mostraban.

Las acciones que pueden hacer son reducidas, no obstante, nos han permitido entender un poco mejor como funciona la librería por dentro. Dichas acciones consisten en proporcionar información a los usuarios sobre los elementos que seleccionan en el mapa, así como visualizar punto a punto las trayectorias de los aviones pudiendo desplazar estos con un slider.

Respecto a cómo utilizar Open Layers el concepto principal es que cada elemento debe pertenecer a una capa (layer) pero a su ver cada elemento es una capa en si mismo. De este modo se crea una estructura de capas en forma de árbol a través de la cual se propagan los estilos, las interacciones y la configuración de la capa. Podemos por ejemplo aplicar un estilo para que todos los elementos dentro de una capa sean de color rojo y posteriormente sobre una rama de árbol decir que a partir de allí sean de color azul. Esto permite crear estilos base que sean modificados según los elementos se hacen más y más concretos.

Frente a otras librerías similares como Leaflet hemos observado que Open Layers requiere de mayor trabajo por parte del usuario para poder crear y manejar las capas. Hay distintos tipos de capas como las vectoriales, las raster o las XYZ. Estas últimas son para visualizar datos en formato Tile que generalmente son extraídos desde un servidor web, este tipo de capa es el utilizado para mostrar los datos de OSM. Por el contrario, en Leaflet solo debemos añadir una capa nueva y la librería se encarga de configurar cada elemento dentro de ella para que se visualice adecuadamente.

### React

Para crear la interfaz hemos decidido utilizar React pues esta librería es capaz de trabajar junto a cualquier otra librería escrita en JavaScript a diferencia de otras que solo permiten utilizar librerías desarrolladas especialmente para trabajar con ellas. No obstante hacer que React y Open Layers pudieran trabajar juntos no ha sido tarea fácil pues se deben sincronizar las actualizaciones que ambas librerías realizan sobre el DOM. No obstante, se ha logrado que componentes de ambas librerías puedan cooperar entre ellos sin limitación alguna.

La ventaja de utilizar React frete a no haberlo echo es la capacidad de poder reutilizar componentes de forma sencilla, así como de poder modificar la interfaz de forma dinámica según se utiliza y sin necesidad de actualizar la página entera.

Esto puede apreciarse especialmente en que los componentes utilizados en la barra lateral son los mismos que los utilizados en el pop up que aparece cuando el usuario selecciona los elementos de mapa. Esto nos ha permitido en definitiva crear la aplicación más rápidamente.

### Conexión con la base de datos

La conexión con la base de datos desde la interfaz no se hace de forma directa si no que se realiza a través del servidor REST que hemos creado encima de ella. Esto es de gran utilizad por múltiples razones:

* Las peticiones que se pueden realizar sobre la base de datos están más controladas.
* No es necesario conocer las consultas SQL concretas que se realizan sobre la base de datos, su estructura es opaca desde la aplicación web.
* Tener un servidor escrito en un lenguaje de programación completo permite realizar acciones más complejas que las peticiones SQL por si solas como formatear los datos o pre procesarlos antes de enviarlos a quien los solicite.

### Acceso a la interfaz

React proporciona un servidor por defecto muy útil para desarrollar la página web pero que recomiendan no debe ser utilizado en producción. Entre otras cosas esto es así pues en caso de que se produzca algún error del que en condiciones normales JavaScript podría recuperar la aplicación fallará por completo alertando al desarrollador de la existencia del problema.

Para evitar esto la aplicación ha sido transformada en un único archivo html como si de una página no desarrollada con React se tratara. Posteriormente dicho archivo es proporcionado a un servidor NGINX que se encargará de servirlo como un documento estático a todos los usuarios que tengan acceso a él y escriban su dirección en el navegador.

## Otras formas de visualizar Los datos

En nuestro caso hemos decidido crear una interfaz web para visualizar los datos y mostrar el coste de las rutas de los aviones, no obstante, esto podría no haberse realizado. Otra opción para acabar obteniendo estos mismos resultados o muy similares hubiera sido no haber creado ningún tipo de interfaz gráfica y haber hecho una sencilla aplicación de consola. No obstante, existen herramientas que ya proporcionan una interfaz gráfica preparada para visualizar datos espaciales cuya configuración es casi inmediata.

### QGIS

En nuestro caso hemos decidido utilizar QGIS como ejemplo de una de estas herramientas. QGIS es capaz de conectarse directamente a la base de datos y extraer de ella toda la información visualizándola posteriormente sobre un mapa interactivo.

En QGIS podemos incluso visualizar los mismos mapas de OSM que han sido utilizados en la interfaz web.

## Despliegue de la aplicación

Uno de los problemas recurrente a la hora de trabajar con bases de datos, servidores y en especial con Java son la instalación y la existencia de múltiples versiones del mismo programa. Es habitual tener una JDK configurada para un conjunto de aplicaciones pero que al descargar una nueva requiera una versión distinta que acaba sobrescribiendo o reemplazando las que ha existía. También es habitual que distintas versiones de la misma base de datos tengan problemas de compatibilidad con ciertas herramientas y de que no todas las versiones estén disponibles para todos los sistemas operativos.

Para solucionar todos estos problemas hemos decidido utilizar contenedores en los que poder encapsular versiones concretas y compatibles de todos los elementos que utilizamos.

### Docker

Hemos utilizado Docker para construir todas las partes que conforman nuestra aplicación. Casa una de ellas estará encapsulada en su propio contenedor, aisladas entre ellas y aisladas también de la máquina en la que se ejecuten con solo los puertos necesarios como forma de acceso a los contenedores.

Esto nos permite automatizar el proceso de construcción de la aplicación con garantías de que esta podrá funcionar en cualquier máquina en la que sea ejecutada.

Como apunte adicional comentaremos la forma en el que en contenedor que contiene la base de datos Postgre+postgis se construye. Para hacerlo utilizamos la versión 12 de Postgre son su extensión postgis compatible ambas descargadas del repositorio **kartoza** especializado en aplicaciones GIS. Posteriormente solo debemos configurar la base de datos, el usuario y la contraseña con la que deseamos acceder a ella. Finalmente indicaremos el puerto desde el que deseamos acceder a la base de datos.

Hemos creado también un volumen, lo cual es algo opcional, para que la base de datos guarde la información que contiene aunque el contenedor finalice su ejecución.