SISTEMA DE AYUDA A LA TOMA DE DECISIONES POR LA CONTAMINACIÓN DEL AIRE

29/11/2019

Práctica 3

Andrés Gavín Murillo 716358

Eduardo Gimeno Soriano 721615

Sergio Álvarez Peiro 740241

Grupo 2-6

Sistemas de Información

Ingeniería Informática

Universidad de Zaragoza

ÍNDICE

Herramientas utilizadas

Capa de control

Para el desarrollo de esta capa se ha utilizado el framework Spring [1], el cual permite el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java. Para facilitar aún más el desarrollo y el uso de este framework se ha utilizado el IDE Spring boot suite [2], eclipse con Spring directamente integrado.

En asignaturas que ya habíamos cursado previamente, como proyecto software, ya habíamos utilizado tanto Spring como el IDE, utilizando el paradigma REST, por tanto, no requeriría demasiado tiempo de adaptación o aprendizaje. Además éramos conscientes de que enmascaraba el uso de servlets y demás tecnologías de bajo nivel, facilitando el desarrollo promoviendo un nivel de desarrollo más alto.

Capa de vista

Para el desarrollo de esta capa se ha utilizado, como en la práctica 0, Bootstrap Studio [3], y en mayor medida Brackets [4], el cual es un editor de código fuente con un enfoque principal en el desarrollo web. Creado por Adobe Systems, es un software gratuito y de código abierto licenciado bajo la Licencia MIT, y actualmente Adobe y otros desarrolladores de código abierto lo mantienen en GitHub.

También se ha utilizado la API de Google maps para el lenguaje Javascript [5] para la gestión del mapa. Ofrece operaciones de topo tipo para obtener, manipular, guardar y cargar rutas, añadir capas KML, etc.

Para establecer en qué zonas se divide la ciudad se ha utilizado My Maps [6] de Google, la cual permite definir gráficamente polígonos sobre un mapa y obtener ficheros KML.

Otras herramientas

Para monitorizar la base de datos mientras se han realizado distintas pruebas, se ha utilizado como en la herramienta anterior DataGrip [7].

Para el despliegue se ha utilizado una máquina virtual en VirtualBox, la cual contiene un Debian 10 en el que se ha desplegado un contenedor Docker con la base de datos y otro con Tomcat, el cual contiene la aplicación web.

Metodología y distribución de trabajo

Capa de control
Cada miembro del grupo se ha encargado de implementar los siguientes controladores:

- UsuarioControl: Sergio
- RutaControl: Andrés
- ZonaControl y DiscurreControl: Eduardo

Una vez implementados cada miembro ha hecho pruebas del respectivo controlador que ha desarrollado.

Capa de vista
Las tareas a realizar para desarrollar esta capa se han distribuido de la siguiente manera:

- Lógica de las rutas: Andrés y Sergio
- Creación de zonas: Eduardo
- Lógica de las zonas: Andrés y Sergio
- Lógica de usuario: Andrés y Eduardo
- Guardado y cargado de rutas: Andrés y Sergio

Las pruebas se han realizado conjuntamente entre los tres miembros del grupo.

Ajustes de la práctica 2

En la primera entrega no se implementó el patrón VO/DAO correctamente, ya que a las funciones que guardaban los datos en la base de datos se les proporcionaba directamente como argumentos los campos que conforman el tipo de dato que debería ser cada entidad a gestionar. Por ejemplo la función para guardar un nuevo usuario en la base de datos recibía como argumentos email, nombre, apellidos y contraseña.

Con las modificaciones realizadas existen ficheros java que implementan el modelo VO y otros que implementan el modelo DAO. Siguiendo el ejemplo anterior la función para guardar un nuevo usuario en la base de datos se encuentra en el fichero UsuarioDAO.java y recibe como argumento un tipo de dato UsuarioVO. Este tipo de dato contiene los campos de la entidad a gestionar, en este caso email, nombre, apellidos y contraseña.

Las rutas en la base de datos finalmente no se guardan con un fichero KML, se guarda el objeto devuelto por el servicio de directions de la API de Google maps, que contiene la información de la ruta, ya que resultaba más sencillo cargarla en el mapa.

Capa de control

Desarrollada utilizando el framework Spring, mencionado en el apartado *Herramientas utilizadas,* ofrece una interfaz a la capa de vista, las operaciones están mapeadas por URLs.

Dentro de esta capa existen cuatro controladores, uno para cada entidad que se manejan en el sistema. El controlador UsuarioControl ofrece las operaciones referentes a la entidad Usuario, estas operaciones hacen referencia al registro y login de usuarios, edición de datos personales y contraseña y borrado de usuarios.

El controlador RutaControl ofrece operaciones referentes a la entidad Ruta, estas operaciones hacen referencia al guardado, recuperación y borrado de rutas de un usuario.

El controlador ZonaControl ofrece una operación referente a la entidad Zona, esta operación hace referencia a la recuperación de las zonas guardadas en la base de datos.

El controlador DiscurreControl ofrece operaciones referentes a la entidad Discurre, estas operaciones hacen referencia al guardado de las zonas por las que transcurre una ruta que haya sido guardada por un usuario, su recuperación y borrado.

Todas las operaciones mencionadas anteriormente producen un JSON, el cual es enviado a quién haya realizado la respectiva petición.

Cada controlador hace uso de las capas VO y DAO para recuperar datos o realizar persistencia de los mismos en la base de datos.

Se hace uso de las anotaciones de Spring para publicar las distintas operaciones de cada controlador mediante una URL, establecer los parámetros necesarios que necesita una llamada a cada operación y declarar cada controlador como un controlador REST. También se utiliza la notación Produces de JAX-WS para declarar que produce una llamada a cada operación.

En cuanto a documentación sobre esta capa se ha generado con Javadoc. Además también se ha intentado generar con Swagger, pero finalmente no ha sido posible debido a problemas de configuración, aunque las anotaciones figuran en el código.

Capa de vista

Esta capa está compuesta por los ficheros html, javascript, css, ficheros auxiliares que contienen iconos, imágenes, etc y un fichero xml descriptor de implementación.

Esta capa de vista contiene cinco páginas:

- index: página de inicio, contiene el login y el registro del usuario.
- principal: página principal del sistema, contiene el mapa y en ella el usuario puede buscar las rutas que desee y guardarlas si lo desea. También puede visualizar las distintas zonas en las que está dividida la ciudad y los niveles de contaminación generales.
- perfil: contiene el perfil del usuario, en ella puede modificar sus datos personales y contraseña, eliminar su cuenta, consultar su historial de rutas y cargar en el mapa una ruta si así lo desea.
- ayuda: contiene preguntas frecuentes y un formulario de contacto.
- contraseña: permite al usuario recuperar su contraseña si la ha olvidado.
- ruta: similar a la página principal, pero esta página es utilizada para mostrar una ruta cargada por el usuario con los niveles de contaminación actualizados.

Los ficheros javascript contienen la lógica necesaria para esta vista:

- index: se realizan peticiones hacia la capa de control referentes al registro y login del usuario.
- principal: se gestiona el mapa, se realizan peticiones a la API de Google maps para obtener las rutas desde el origen al destino introducidos por el usuario. Se realizan los cálculos de contaminación para estas rutas viendo por que zonas pasan y realizando peticiones a la capa de control para obtener el valor de dióxido de carbono de cada zona, mostrando este valor junto con la distancia y tiempo. También se realizan peticiones para obtener las distintas zonas en la que está dividida la ciudad y cargarlas en el mapa.
- perfil: se realizan peticiones hacia la capa de control referentes a la edición de datos personales y contraseña de un usuario, así como el borrado de la cuenta y carga de rutas guardadas por el usuario previamente.
- ruta: se realizan peticiones hacia la capa de control referentes a la obtención de niveles de contaminación actualizados para la ruta que ha sido cargada en el mapa por el usuario.
- common: contiene lógica común que utilizan varias páginas, como cerrar sesión y gestión de la cookie del usuario.
- md5: contiene lógica para cifrado, utilizado para el cifrado de las contraseñas.

Como se menciona anteriormente para la gestión del mapa se utiliza la API de Google maps para obtener tres rutas como máximo desde un punto origen a un punto destino, para saber si dados una serie de puntos que conforman una ruta, pasa por una de las zonas definidas previamente, de la cual se conocen sus vértices. Cada zona es una capa KML, la cual se carga en el mapa una vez obtenida de la base de datos, las rutas, sin embargo, se cargan sobre el mapa base.

La lógica referente al mapa se ha desarrollado de forma autónoma al resto del sistema, exceptuando el guardado de rutas y la recuperación de las mismas y las zonas, ya que es la parte más compleja del sistema y para realizar pruebas básicas, como obtener una ruta dados un origen y un destino hubiera requerido mucho tiempo utilizando el sistema entero.

Una vez esta lógica ha funcionado correctamente, se ha añadido junto con el resto del sistema.

Pruebas

Capa de control
Para realizar pruebas sobre esta capa, se ejecutaba el servidor sobre el puerto 8083 y a través del navegador se realizaban peticiones mediante URLs. La base de datos se monitorizaba utilizando DataGrip para ver que los datos se guardaban correctamente. Además se comprobaba el contenido de los JSON devueltos por las distintas peticiones.

Capa de vista
Las pruebas sobre esta capa se realizaban sobre el propio navegador, utilizando el editor Brackets, el cual provee algunos aspectos que facilitan este tipo de pruebas ya que está orientado a desarrollo web. Con estas pruebas se comprobaba la lógica de la web que no requería de interacción con la capa de control, es decir, gestión del mapa, obtención de rutas, etc.

Pruebas de integración
Una vez se realizaban las pruebas anteriores, se realizaban pruebas de integración del sistema completo, también para comprobar la lógica que incluía ambas capas, como puede ser la carga de rutas de un usuario. Estas pruebas requerían desplegar el sistema completo.

Dificultades encontradas

Las principales dificultades encontradas han surgido en la capa de vista, ya que era la primera vez que utilizabamos una tecnología como la de Google maps, además de Javascript. Una vez obtenida una ruta dados un origen y un destino, obtener las distintas zonas por las que transcurre no ha sido fácil, aunque las zonas las hemos definido nosotros, hemos necesitado obtener los vértices de cada una de ellas para utilizar una operación de la API de Google maps que dados dos puntos cardinales (NE y SW) construya esa zona y nos diga si un tercer punto cardinal, pasado también como parámetro, está dentro de esa zona.

Otra dificultad encontrada ha sido el guardado de las rutas de un usuario. Guardarlas en formato KML nos ha generado diversos problemas, sobre todo a la hora de cargarla posteriormente. Finalmente, hemos optado por guardar una ruta como el objeto devuelto por el servicio directions de la API de Google maps.

Al definir nosotros las zonas en las que está dividida la ciudad, nos ha supuesto por un lado facilidades y por otro dificultades. Por un lado, nos permite conocer bien los límites de cada zona, por otro, estas no coinciden con las del ayuntamiento, por tanto, no podemos disponer de los mismos datos. Finalmente, hemos considerado que nos resultaba más crucial conocer los puntos cardinales limítrofes de cada zona para realizar los cálculos de contaminación de cada ruta. Aunque se nos presenta como inconveniente que tengamos que actualizar los datos de contaminación nosotros.

Esfuerzos invertidos

Eduardo		
Fecha	Tarea	Duración
8/11	Ajustes de práctica 2 aplicando el patrón DAO y VO	1h
8/11	Creación y configuración de proyecto Spring, y unión de P2 con P1	2h
15/11	Capa de control: ZonaControl y DiscurreControl	2h
16/11	Capa de control: pruebas	1h
25/11	Lógica de usuario (html y javascript)	2'5h
26/11	Creación de zonas, obtención de ficheros kml y carga en la base de datos	3h
27/11	Gestión de las zonas y ayuda a los otros dos miembros del grupo en este aspecto	4h
28/11	Memoria	5,5h

Andrés		
Fecha	Tarea	Duración
8/11	Ajustes de práctica 2 aplicando el patrón DAO y VO	3h
8/11	Creación y configuración de proyecto Spring, y unión de P2 con P1	2h
15/11	Capa de control: RutaControl	2h
16/11	Capa de control: pruebas, empaquetado Docker y despliegue	3h
17/11	Lógica de usuario y ruta (html y javascript)	4h
22/11	Lógica de rutas, zonas y guardado/carga de rutas	3,5h
25/11	Capa de control: modificaciones	1,5h
28/11	Capa de control: modificaciones, lógica de rutas, zonas y guardado/carga de rutas	10h

Sergio		
Fecha	Tarea	Duración
8/11	Ajustes de práctica 2 aplicando el patrón DAO y VO	1h
8/11	Creación y configuración de proyecto Spring, y unión de P2 con P1	2h
15/11	Capa de control: UsuarioControl	2h
16/11	Capa de control: pruebas	2h
26/11	Lógica de rutas, zonas y guardado/carga de rutas	6h
27/11	Lógica de rutas, zonas y guardado/carga de rutas	5h
28/11	Lógica de rutas, zonas y guardado/carga de rutas	4h

Bibliografía y referencias

[1] *Spring framework*
https://spring.io/

[2] *Spring tools suite*
https://spring.io/tools

[3] *Bootstrap Studio*
https://bootstrapstudio.io/

[4] *Brackets*
http://brackets.io/

[5] *Javascript Google maps API*
https://developers.google.com/maps/documentation/javascript/tutorial

[6] *MyMaps*
https://www.google.com/intl/es_ES/maps/about/mymaps/

[7] *DataGrip*
https://www.jetbrains.com/datagrip/?gclid=CjwKCAiA_f3uBRAmEiwAzPuaM7Bl2dXtzHqxPnKoyDvZXtAcMP8pp9fUkLf7L1byy2llJPsJ_UucRhoC3cIQAvD_BwE