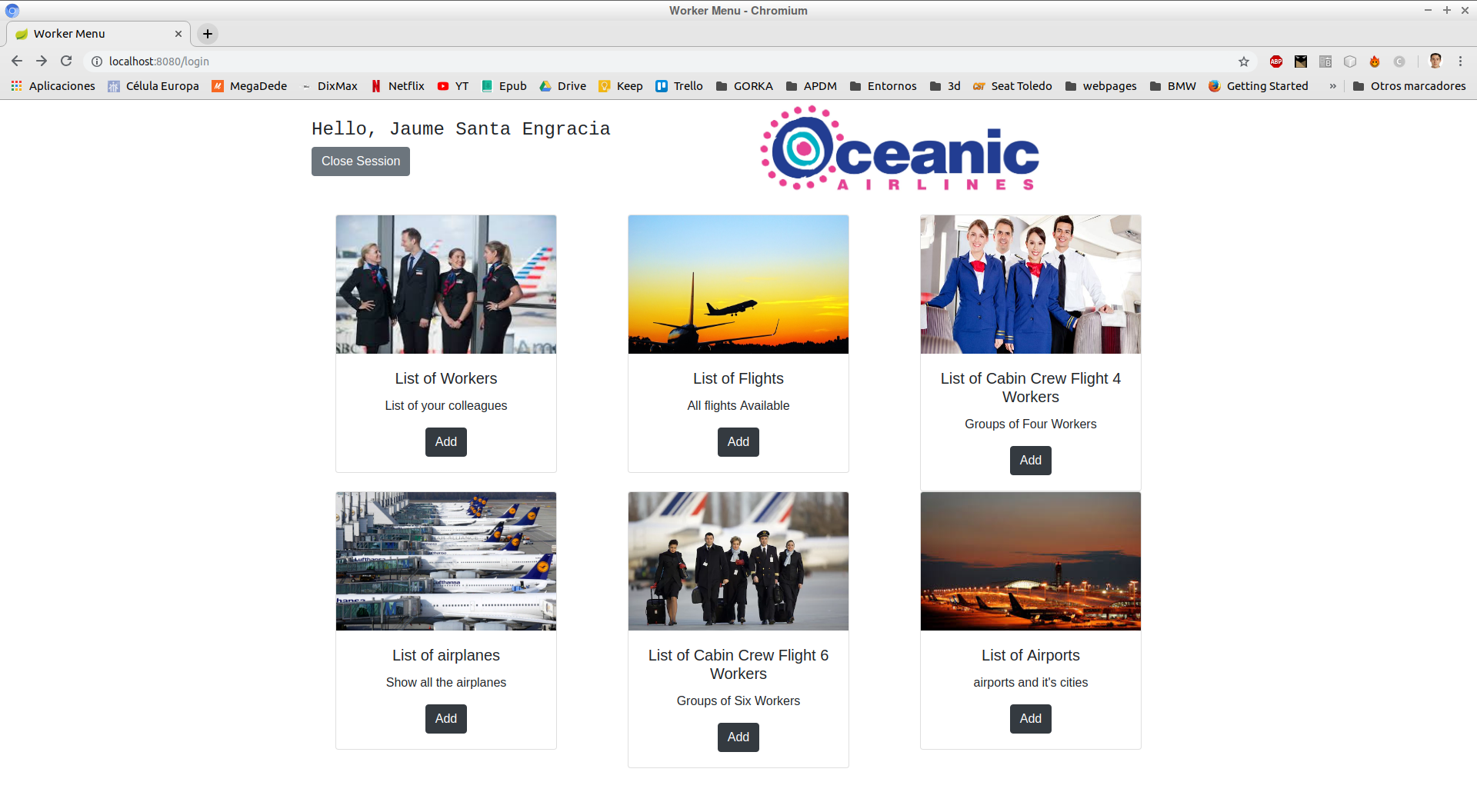
TRABAJO DE FIN DE GRADO



Antonio Enrique García Escribano 2°H

GESTIÓN DE AEROLÍNEA DESPLEGADA EN WEB

Como dice el título, he decidido crear esta solución para una compañía aérea con las ventajas que supone centralizar el programa en un servidor:

VENTAJAS:

* No es necesario usar recursos locales como el procesador, RAM, Disco duro…
* No estamos atados a que sistema operativo tiene el cliente
* Sin problemas con drivers, DLLs, ajustes gráficos, Etc
* Más fácil de actualizar

DESVENTAJAS:

* La seguridad está comprometida si el cliente está infectado
* La seguridad está comprometida si la conexión está hackeada
* Problemas gráficos si el ordenador cliente es demasiado antiguo o no está actualizado
* Imposible conectar si no hay internet

**SOLUCIONES QUE APORTA ESTA APLICACIÓN**

El objetivo es aportar una solución web a todas las partes de la aerolínea:

* Pasajeros
* Trabajadores
  + Administrativos
  + Tripulantes de cabina

Esta aplicación se va a componer de un sistema de inicio de sesión para controlar que tipo de usuario es el que está en la página. Si es un pasajero, controlaremos que las páginas que pueda visitar sean para comprar billetes, ver precios y disponibilidad de vuelos, editar reservas, etc.

Si es un piloto o una azafata, podrá iniciar sesión para ver cuál es la lista de vuelos en los que tendrá que trabajar en un futuro, comprar billetes más baratos o ver información de los companeros.

Si es alguien de la gestión administrativa de la aerolínea, podrá crear vuelos, dar de alta aviones nuevos que se incorporen a la flota, crear grupos de trabajo para asociar a un vuelo, podrá crear menús de comida, crear perfiles de trabajadores y editarla si es necesario, etc.

**¿QUÉ LENGUAJE Y POR QUÉ?**

He visto dos posibles soluciones a esto, la primera fue utilizar ASP .NET con C# y MVC, muchas posibilidades para desarrollar esta solución rápidamente. Pero las circunstancias de utilizar el ordenador de forma remota y los problemas que me dio Visual Studio, no he podido resolverlos, así que he acabado tomando la siguiente decisión, cambiar a JAVA con Spring.

Decisión muy interesante ya que me toca explorar caminos nuevos, como instalar MySQL en Linux, conectar manualmente la BBDD al proyecto y desarrollar una aplicación log in en Java.

Al final desarrollaré el proyecto en una máquina virtual Lubuntu con Eclipse, MySQL en Linux y usando MySQL WorkBench como entorno gráfico. También usaré GIT como sistema de control de versiones, para las dependencias usaré Maven, utilizando las etiquetas XML me ahorrará incorporar JARs y más archivos al proyecto.

Para hacer más atractivo el apartado visual, he anadido Bootstrap a las páginas JSP, además de aplicar estilos muy correctos y de forma muy rápida, es capaz de hacer una página Responsive (adaptable a cualquier resolución) con unas simples etiquetas. Por último Spring ahorra muchísimas líneas de código cuando está incluido en el proyecto, añadiré algunos ejemplos más adelante.

**PRIMEROS PASOS**

Cuando me propuse aplicar tecnologías o soluciones que no había utilizado nunca, todo empieza por dar pequenos pasos primero. Me tocó instalar y configurar el gestor de bases datos (Workbench), crear unas tablas sencillas para probar las primeras inserciones y los primeros listados. También tocó crear las primeras conexiones con la BBDD, y probar a mostrar datos en web.

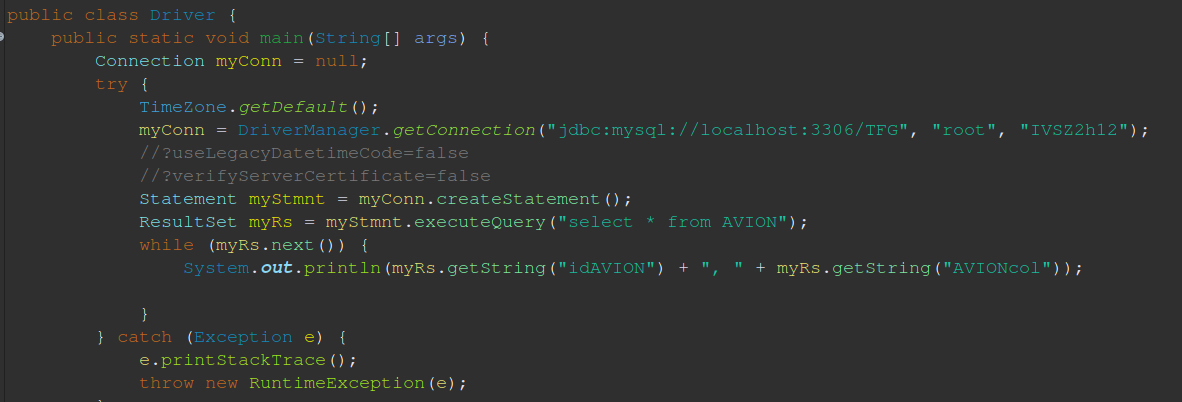
Una vez conseguido eso, incorporé bootstrap al proyecto, mucho más rápido de lo que pensaba. Después tocó crear la base de datos completa, que ha ido evolucionando conforme he visto que ciertos cambios eran necesarios.

En total existen catorce tablas en el proyecto, mientras iba mapeando en mi cabeza como tenía que crear la solución al problema, las tablas se han ido sumando y al final ha quedado una estructura lo suficientemente compleja e interesante para ser desglosada.

**TEST DE SELECCIÓN**

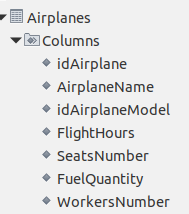
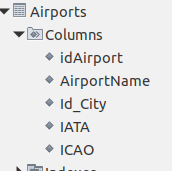
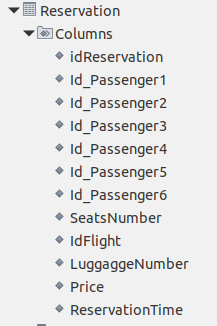
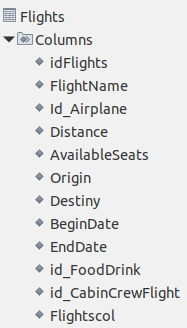
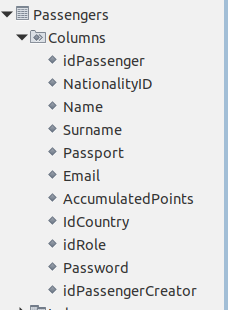
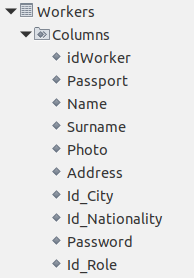
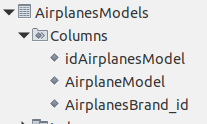
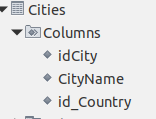
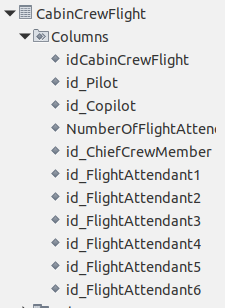
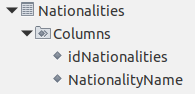
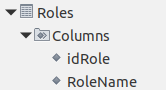
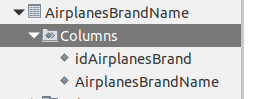
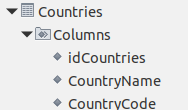
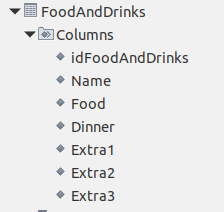
Lo primero de todo, antes de crear la base de datos definitiva, creé una BBDD muy sencilla con una tabla, para comprobar que el string de conexión, el puerto, todos los elementos de los que depende interactuar con la Base de Datos funcionaba.

Creé un pequeño archivo con una función de conexión que mostró por consola los datos de la tabla.



**LA BASE DE DATOS**

Como ya he dicho, tenemos catorce tablas, que son:

* Airplanes:
  + La tabla en la que tenemos registrados los aviones de la flota
  + 
* Airports:
  + Aeropuertos disponibles para realizar un vuelo
  + 
* Reservation:
  + Reserva de un vuelo hecha por un pasajero
  + 
* Flights:
  + Tabla de vuelos, en las cuales está relacionado un grupo de trabajo, un avión, un origen y un destino.
  + 
* Passengers
  + Los pasajeros que tienen un perfil en el sistema
  + 
* Workers
  + Trabajadores de la compañía, pueden ser de administración, azafatas, jefas de azafatas, pilotos o copilotos
  + 
* Airplanes Models
  + Tabla de modelos de aviones, que está registrada con una marca
  + 
* Cities:
  + Ciudades del mundo, los aeropuertos están vinculados con ciudades de esta tabla
  + 
* CabinCrewFlight
  + Grupos de trabajo, en los cuales van asociados los IDs de trabajadores, estos grupos luego serán asociados en la tabla Flights, pueden variar de 4 a 6 azafatas dependiendo del modelo del avión.
  + 
* Nationalities
  + Nacionalidades de trabajadores o de pasajeros.
  + 
* Roles
  + Roles que van asociados a los trabajadores.
  + 
* Airplanes Brand Name
  + Marcas de aviones que van asociados con los modelos.
  + 
* Countries
  + Países que van asociados con las ciudades.
  + 
* Food And Drinks
  + Menús que van asociados con los vuelos.
  + 

El cliente de SQL es SQL Workbench, un entorno al que hay que acostumbrarse porque no funciona como SQL Server, sin embargo, es 100% funcional para crear y mantener una estructura de bases de datos en Linux. La creación de tablas, actualización de registros o inserciones muestran siempre el script sql que van a ejecutar pudiendo modificarlo incluso antes de ser lanzado.

El periodo de aprendizaje de esta nueva plataforma, el desarrollo de la base datos, sujeta siempre a cambios para no dejar de mejorarla, tomó bastante tiempo de desarrollo, sin embargo, ahora es un elemento sólido del proyecto que casi no necesita ser modificado.

Como ejemplos interesantes, las ciudades que contiene la base de datos son más de 12800, cuadrar tantas inserciones y relacionarlas con los países fue toda una aventura, teniendo que relacionar el id del país con esa ciudad antes de hacer la inserción.

**ESTRUCTURA**

El proyecto está programado sobre JAVA y Spring como framework de desarrollo, Spring facilita la vida a la hora de desarrollar un proyecto desplegado en Web, a continuación voy a explicar la estructura y cómo funciona el proyecto.

**LOS ARCHIVOS**

El proyecto está dividido en distintos paquetes:

* Connection: almacena los archivos de conexión.
* Controller: almacena todas las funciones de controlador y tratamiento de los datos que recibe la página de los formularios.
* Demo: archivo necesario para iniciar el proyecto.
* Model: contiene los modelos que corresponden con la BBDD.
* Repository: lugar que almacena las consultas de inserción, selección, actualización y borrado con los que funciona la aplicación. Cada modelo tiene su propio repositorio.
* Service: el service actúa de interfaz entre el repositorio y el controlador, un buen punto para mejorar sería crearlo como una interfaz. Cada modelo tiene su propio servicio.

**SPRING**

Spring relaciona todas las clases y atributos mediante etiquetas que anadimos a las clases, a continuación vamos a comentar las etiquetas que están incluidas en el proyecto:

* @component: La encontramos en las clases de los modelos.
* @Repository: La utilizamos en los repostiorios, donde almacenamos las consultas SQL.
* @Service: La utilizamos en el Service, que usamos de interfaz entre el repositorio y el controlador.
* @Autowired: La utilizamos como la propia palabra indica en inglés para buscar dentro del proyecto un objeto que corresponda con el declarado
* @Qualifier: Para declarar un objeto que tiene una etiqueta de Spring con un nombre concreto.
* @Controller: para identificar las clases que son controladores
* @SessionAttributes: para cargar en sesión los objetos que correspondan en nombre, en nuestro proyecto, trabajador en sesión
* @PostMapping y GetMapping: para indicar de qué forma mandamos los parámetros o accedemos al enlace.

**ARCHIVOS DE CONEXIÓN**

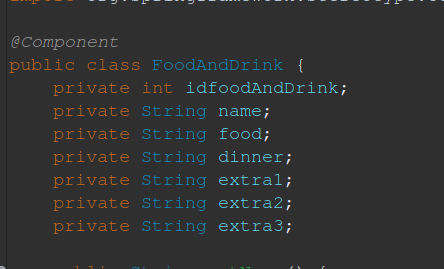
En el package conexión, podemos ver que tenemos tres clases:

* Connection Manager, que es una interfaz con métodos vacíos.
* Connection SQL: que implementa la interfaz anterior, en esta extendemos las funciones que tenía:
  + open:
    - Función que abre la conexión con la BBDD.
  + executeSql:
    - función que ejecuta la consulta que nosotros hemos introducido.
  + Close:
    - Función que cierra la consulta después de haber ejecutado la sentencia SQL.
* Driver: fichero de prueba que testeaba la conexión con la BBDD.

**EJEMPLO BASADO EN INSERCIÓN**

Para entender el funcionamiento de la página, vamos a hacer un recorrido desde la página que ve el usuario hasta el repositorio en el que se hace la inserción.

**PÁGINA JSP**

Para empezar la página en la que hacemos el formulario de inserción la tenemos que tener relacionada con nuestras clases, es decir, nuestro formulario tendrá un nombre que se corresponderá con el de una clase y a su vez los campos de los formularios serán los atributos de esa clase.

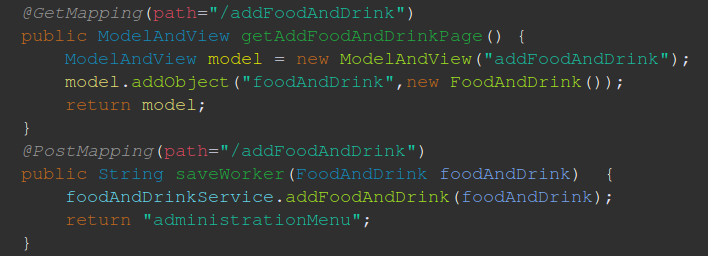


Tenemos la etiqueta action, sobre el enlace que actuará cuando enviemos el formulario, modelAttribute es el nombre de la clase con la que queremos relacionar nuestro formulario, si vamos al primer input del formulario podemos ver que tiene etiqueta name y path, de esta forma relacionamos ese input con un atributo del formulario.

Cuando cliquemos el botón submit actuará sobre la dirección indicada en la etiqueta action. Esto está indicado en el controlador de la aplicación, el lugar donde se concentra casi toda la programación de la aplicación.

**EL CONTROLADOR**

Cada acción sobre una página o un formulario de la aplicación está dividido en funciones, en nuestro caso vamos a buscar las funciones que responde a la petición de addFoodAndDrink, porque accedemos a esta misma dirección por post y por get.

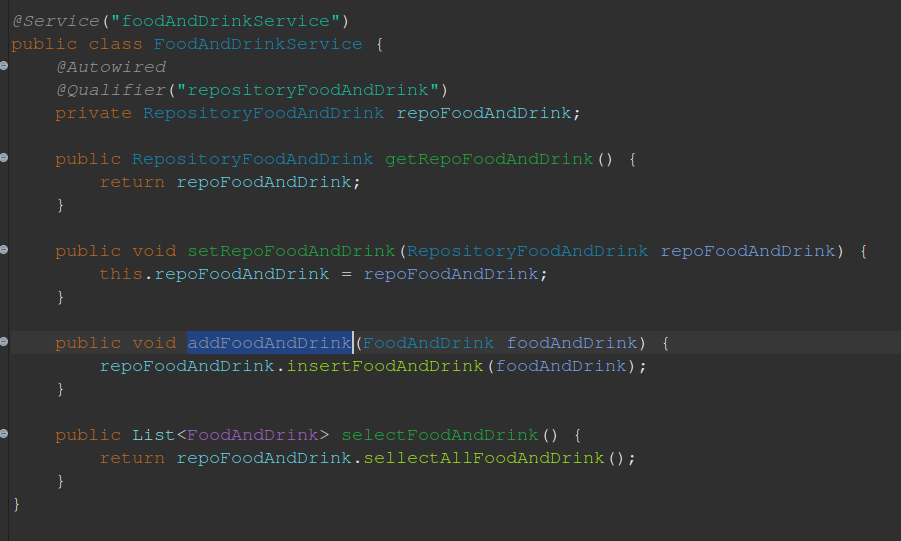


El primer método es para acceder a la página, busca un modelo que coincida con el nombre de una de las páginas que tenemos, además añade un objeto, que es el que tiene la etiqueta de modelAttribute, añade e instancia ese objeto, esos parámetros como ya hemos dicho, coincidirán con ese objeto.

El post, es el momento en el que nosotros añadimos ese objeto a nuestra base de datos, espera recibir un objeto de la clase FoodAndDrink, de esta manera no tenemos que hacer un set de cada atributo, como están ya relacionados como ya hemos dicho.

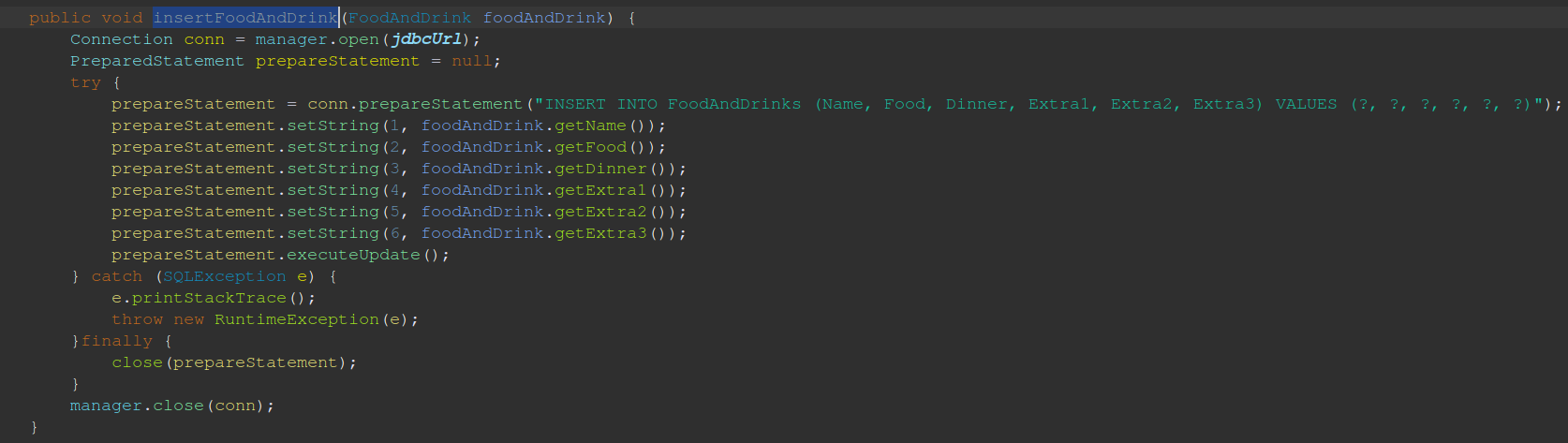
Finalmente nos devolverá al menú de inicio.

**EL SERVICE**



Esto hace de interfaz entre el controlador, donde se concentra la mayor cantidad de programación en la aplicación, y el repositorio.

**EL REPOSITORIO**



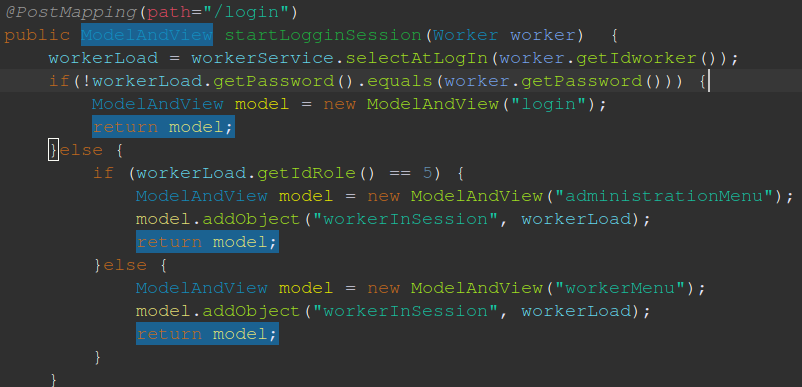
Aquí tenemos la función que relaciona cada elemento del objeto que le pasamos y relacionamos con la consulta.

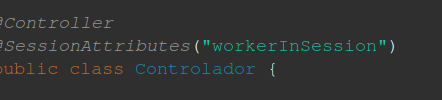
Con tan pocos pasos, somos capaces de relacionar, manejar y pasar los parámetros de una función a otra, esto es la clave de las páginas desplegadas en web, una página web, con toda la potencia de Java detrás.

**EL LOGIN**

El login es una de las partes más interesantes de la aplicación permite diferenciar entre un trabajador cualquiera y los de administración, dando solamente potestad para entrar a ciertas páginas a los de administración.

Lo que hacemos en la página de login es pasar los datos que nosotros insertamos al SessionAttribute, a ese objeto que llamamos workerInSession.





Comparamos el id que introduce el usuario, lo busca en la base de datos y le pregunta por su role, si es cinco entra en administración, sino entra a la página de trabajadores.