Práctica 1 - PTI

Java Servlets

Arnau Yepes Huguet

Natalia Dai

Grupo 1

19/02/2023

**Índice**

[**1. -Introducción**](#_3ba7otabqmjx) **2**

[**2. -New Rental**](#_8hzfko10hyyx) **3**

[**3. -List Rentals**](#_s477e7au4vbd) **4**

[**4. -Configuración SSL/TLS**](#_slhudp3t8xx0) **6**

[**5. -Dockerization**](#_42amrznusima) **7**

[**6. -Extra: Pros & Cons Java Servlet vs Node.js**](#_6fpy76yuvvhs) **7**

# **1. -Introducción**

En esta sesión desarrollaremos una aplicación web sencilla de alquiler de coches con servlets de Java con la ayuda de Tomcat (servidor web), implementada con ficheros HTML, JSON y por supuesto, JAVA.

Una vez instalado y configurado todo lo necesario para hacer la práctica según el enunciado, iniciamos el servidor de Apache Tomcat (que lo tuvimos que instalar) con “./bin/startup.sh &” y accedemos a <http://localhost:8080/>, donde tendremos que programar las funcionalidades que la aplicación ha de tener:

* En la sección de “New rental” tener un formulario sobre los parámetros del coche que se desee alquilar y guardar estos datos de la petición. Esta parte está implementada en el fichero de “CarRentalNew.java”
* En la sección de “List rentals” poder entrar solamente si se introducen los credenciales correctos, y obtener una lista de todos los coches que se han alquilado con sus parámetros. Esta parte está implementada en el fichero de “CarRentalList.java”

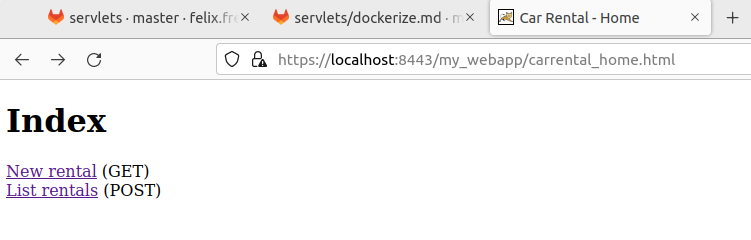


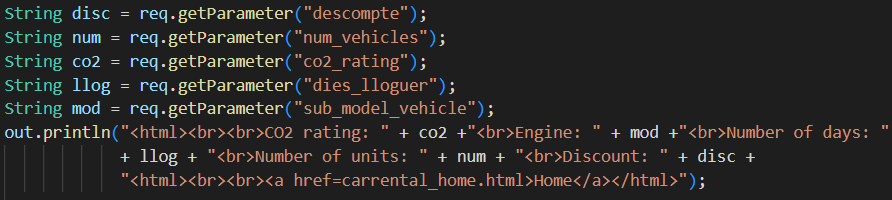
Figura 1 - Índice de la página principal

# 

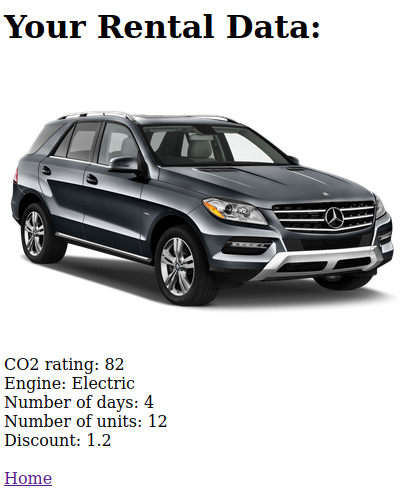
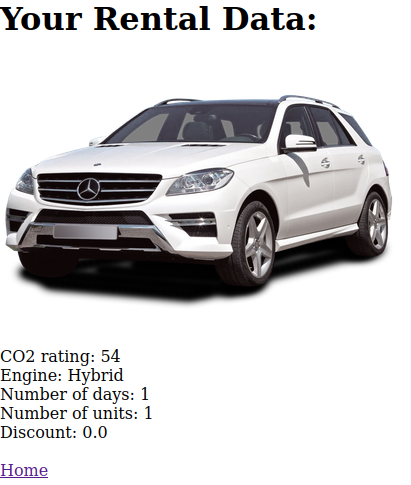
# 

# **2. -New Rental**

Al entrar en esta página y rellenar los datos que corresponden en cada caso según el cliente, mostramos en la página los datos introducidos:

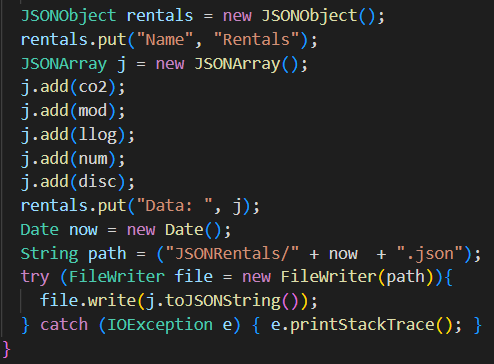


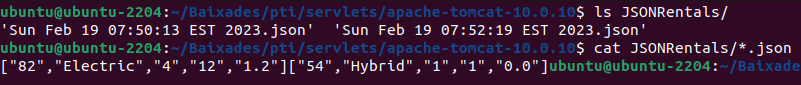




Figuras 2,3,4,5 - New Rental datos introducidos

Al tener los datos que queremos, lo guardamos en un fichero JSON que creamos dentro de la carpeta de “JSONRentals”, donde guardamos todos los JSONArrays que creamos. En este caso, al tener dos coches alquilados, saldrían:





Figuras 6, 7 - Uso de JSON para guardar datos

# **3. -List Rentals**

Al entrar en esta página nos encontramos con este formulario para poder hacer login y ver la información almacenada en los ficheros JSON sobre los coches alquilados.

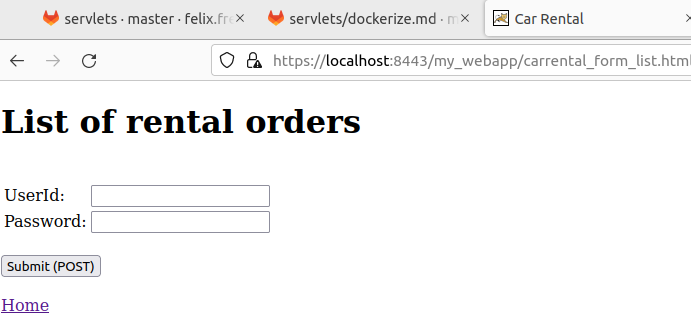


FIgura 8 - Login de la página List Rentals

Con la entrada de datos de “UserId” y “Password” comprobamos los credenciales. En este caso hemos puesto que se ha de acceder con el usuario “admin” y contraseña “123”. En caso de ser incorrecto el usuario no tendrá acceso a los datos, como en este ejemplo, que se accede con usuario “a” y contraseña “a”:

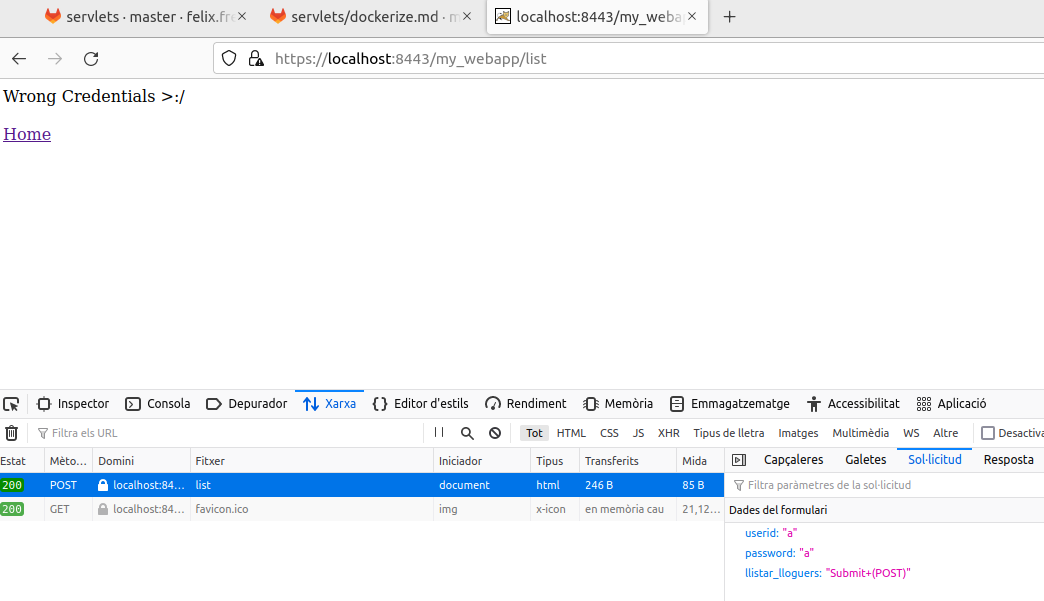
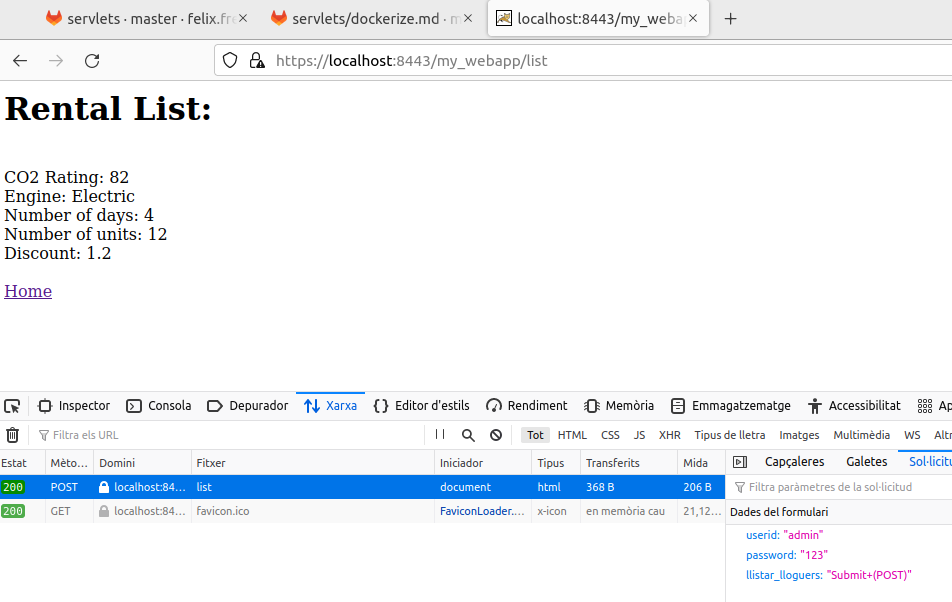


Figura 9 - Credenciales incorrectas

En caso de que el valor de ambos campos, usuario y contraseña, sean los indicados, se podrá ver una lista de los coches alquilados con cada una de sus características pasando por todos los JSONArrays creados:





Figuras 10, 11 - Implementación y resultado de Rental List

# **4. -Configuración SSL/TLS**

En este apartado, creamos una llave privada para el servidor y un certificado firmado por sí mismo, para poder establecer comunicación por https.

Ejecutando la comanda que se nos proporciona en la práctica (“sudo keytool -genkey -alias tomcat -keyalg RSA”) le ponemos una contraseña y después de configurar todo según el enunciado, intentamos acceder a <https://localhost:8443> y vemos que la conexión no es privada, por lo que sabemos que está bien configurada.



Figura 12 - conexión a <https://localhost:8443>

# **5. -Dockerization**

En este apartado, hemos hecho un docker de nuestra aplicación web con la ayuda del Dockerfile que se nos proporciona (el contenido) y unas comandas (“docker build -f Dockerfile -t carrental .” y “docker run --name carrental -d -p 8080:8080 -p 8443:8443 carrental”). Ahora con el Dockerfile y la carpeta “my\_webapp” construimos el tarball para la entrega.

# **6. -Extra: Pros & Cons Java Servlet vs Node.js**

En esta práctica estamos aplicando Java Servlet pero hay otras alternativas con funcionalidades parecidas, como es el caso de Node.js.

Aunque puedan ser similares, Node.js es un entorno de código abierto, usado normalmente para aplicaciones Javascript, y el Java Servlet es una clase de Java que se usa para ampliar las capacidades de un servidor.

Los algunos **pros** sobre el uso de Java Servlet sobre Node.js serían:

* Java es un lenguaje de programación más comúnmente usado, por tanto tiene una comunidad (programadores y organizaciones) más extensa, pudiendo dar más soporte y documentación que Node.
* Java es más fácil de entender y de usar que Node (y al tener más soporte, las dudas son más fáciles o más rápidas de resolver).
* Java tiene IDEs (herramientas para ayudar a los desarrolladores a producir código) más complejos y son más eficientes, por tanto ayuda más al programador.
* Java puede trabajar concurrentemente mientras Node no puede.
* Java cuenta con Eclipse, NetBeans e Intellij, tres herramientas de primer nivel muy bien integradas con depuradores, servidores y descompiladores.

Los algunos **pros** sobre el uso de Node.js sobre Java Servlet serían:

* El código de Node es más compacto y por tanto, podría resultar ser más limpio y más fácil de leer que el código en Java.
* Node muestra un rendimiento casi un 20% más rápido que Java
* Node tiene una I/O asíncrona.