Informe PTI

Lab 2

Autores

Marc Cabezas

Antonio Arellano

# 

[**1. Familiarización con el entorno**](#_zbsrztttoz71) **3**

[**2. Caso práctico**](#_eqrvn97mp2sg) **4**

[2.1.CarRentalNew.java](#_jtws06r9179f) 4

[2.2 CarRentalList.java](#_ef0fft55jxqr) 6

[**3.Conclusiones**](#_mbgfccx2owvk) **9**

[3.1 Conclusiones de la familiarización con el entorno](#_3e7zl5nzfwxi) 10

[3.2 Conclusiones del caso práctico](#_e8qfp4soriq8) 10

[3.3 Conclusiones finales](#_jor0xhirxwt5) 10

# 

# 

# 1. Familiarización con el entorno

Se ha configurado la máquina, instalando Tomcat 7 y Java para poder ejecutar correctamente los archivos html y .java.

Se han ejecutado estos comandos:

*sudo* ***add****-apt-repository* ***pp****a:webupd8team/java*

*sudo apt-****get******update***

*sudo apt-****get*** *install oracle-java8-installer*

*sudo apt-****get*** *install oracle-java8-****set****-default*

*sudo* ***ln*** *-****s*** */usr/lib/jvm/java-8-oracle /usr/lib/jvm/default-java*

*sudo apt-get* ***update***

*sudo apt-****get******install*** *tomcat7*

*sudo apt-****get******install*** *tomcat7-docs tomcat7-admin*

Una vez estaba todo instalado se han realizado pruebas para comprobar que todo se había instalado correctamente.

Se han realizado más pruebas con varios ejemplos:

|  |
| --- |
|  |
| Figura 1. Página de inicio de Tomcat 7. |

|  |
| --- |
| /Users/MarcCaps/Downloads/Screenshot from 2017-02-25 20-37-04.png |
| Figura 2. Ejemplo de mensaje en Tomcat7 usando HTML. |

|  |
| --- |
| /Users/MarcCaps/Downloads/Screenshot from 2017-02-25 20-41-44.png |
| Figura 3. Ejemplo de mensaje en Tomcat7 usando Java. |

# 2. Caso práctico

En esta sección se explicarán las soluciones adoptadas para los problemas propuestos, luego éstas soluciones nos ayudarán a elaborar un conclusión.

## 2.1.CarRentalNew.java

Se nos pedía crear un formulario en el cual pudiésemos añadir un nuevo alquiler de coche a una supuesta lista de alquileres de vehículos, en él debían aparecer campos como el modelo del auto, el tipo de motor de éste, el número de días de alquiler, el número de unidades a alquilar, y la base de descuento aplicada al precio final del alquiler. El front-end de este formulario ya se nos facilitaba a modo de ejemplo (figura 5), hemos optado por utilizarlo y crear la clase “CarRentalNew.java” el cual implementa la lógica del formulario. Pudiera ser que en el momento de introducir los datos cometamos errores de tipo, como por ejemplo introducir un número negativo de días que alquilamos un vehículo o 0, unidades o números negativos en el descuento, en estos casos hemos optado por capturar los errores en la clase e informar al usuario del error volviendo a generar el formulario y añadiendo un mensaje de error, ésta vez desde la clase, así podemos probar también cómo generar código HTML desde Java (figura 6). En caso de que el alquiler se añada correctamente, se ha generado por la clase CarRentalNew.java una vista que nos informa de eso y nos da todos los detalles sobre el alquiler que acabamos de añadir (figura 7). Vamos a mostrar a continuación una serie de capturas, las cuales reflejan las diferentes vistas del ejercicio.

|  |
| --- |
| creenshot from 2017-02-19 15:55:14.png |
| Figura 5. Formulario de registro de alquiler. |

|  |
| --- |
| creenshot from 2017-02-19 16:00:28.png |
| Figura 6. Formulario con mensaje de error. |

|  |
| --- |
| creenshot from 2017-02-19 15:55:30.png |
| Figura 7. Vista que se obtiene cuando se ha añadido correctamente un alquiler. |

## 2.2 CarRentalList.java

También se nos pedía implementar una lista con todos los alquileres registrados en el formulario anterior, se nos proporcionaba un formulario de login (figura 8), ya que se supone que esa lista sólo la puede ver el administrador, por lo que si el login es satisfactorio, se nos mostrará la lista con los alquileres registrados (figura 9). Hemos utilizado el formulario que se nos proporcionaba, y creado la clase “CarRentalList.java” la cual implementa tanto la lógica detrás del formulario como la lógica de mostrar la lista, e igual que en el punto 2.1, si se producía algún error en la introducción de los datos en el formulario, se vuelve a generar el formulario con Java más un mensaje de error (figura 10). Como ya se ha dicho, “CarRentalList.java” nos genera la lista en HTML desde Java. Vamos a mostrar a continuación una serie de capturas, las cuales reflejan las diferentes vistas del ejercicio.

|  |
| --- |
|  |
| Figura 8. Formulario de inicio de sesión. |

|  |
| --- |
|  |
| Figura 9. Listado de alquileres registrados. |

|  |
| --- |
|  |
| Figura 10. Formulario con mensaje de error. |

# 3.Conclusiones

La estructura de las conclusiones va a ser parecida a la del informe anterior salvo por el matiz que ahora tenemos otra tecnología con la que comparar resultados e impresiones. Si bien, este factor, el de tener una tecnología con que comparar, no va alterar mucho la estructura, pero sí su contenido, ya que se intentarán remarcar algunas diferencias entre las dos tecnologías. Se vuelven a estructurar de la siguiente manera, primero comentaremos las conclusiones que hemos sacado del punto 1 (Familiarización con el entorno), luego las del punto 2 (Caso práctico) y finalmente a partir de las conclusiones anteriores, intentaremos sacar una conclusión definitiva de las tecnologías con las que hemos trabajado, así como puntos a favor y en contra o posibles escenarios en donde éstas tecnologías serían de gran utilidad.

## 3.1 Conclusiones de la familiarización con el entorno

El deploy del entorno no ha sido difícil, la instalación de los paquetes no ha dado problemas, y se ha conseguido hacer en un tiempo considerablemente corto. En la prueba de los ejemplos no ha habido ningún problema. Destacar que con ésta tecnología la estructura de carpetas está centralizada, cosa que con la tecnología del informe pasado (Apache 2 y CGI) no era así. Como punto negativo la puesta a punto del entorno de desarrollo. Como se comentará en las conclusiones del caso práctico, lo que más problemas nos ha dado ha sido preparar el entorno de desarrollo, ya sea porque se han utilizado IDEs y se ha tenido que adaptar el desarrollo a éstos (configurando el servidor para que éstos funcionen) o por problemas con las librerías.

## 3.2 Conclusiones del caso práctico

No ha sido difícil entender los lenguajes que se utilizan en el caso práctico ya que los dos componentes del grupo habíamos utilizado Java antes, pero comparándolo con Python, Java es más complejo a nivel de cómo estructurar los archivos de código. Para el programador novel es más amigable Python ya que no ha de molestarse donde están contenidas las clases, en cambio en Java ha de tener presente la organización de los packages y la importación de librerías puede ser molesta sin esos conceptos claros.

Para almacenar la lista de alquileres, ésta vez se ha utilizado un archivo JSON en vez del fichero CSV que utilizabamos en la anterior práctica, si bien éste hecho tiene ventajas e inconvenientes. Como ventaja principal encontramos que java puede tratar como un objeto éste tipo de archivos, por lo que se pueden elaborar programas más complejos y amigables para el programador, por otro lado no hemos conseguido añadir alquileres sin antes cargarlos todos en memoria, con CSV no hacía falta leer para escribir, en cambio con JSON hemos de leer todo el archivo JSON, añadirle al objeto el nuevo alquiler y guardar todo el objeto en el mismo archivo, quizás haya otra manera más eficiente de hacerlo, pero nuestro grupo no la ha encontrado, por lo que quizás lo que se gana en legibilidad del código se pierde en eficiencia.

## 3.3 Conclusiones finales

Basándonos en las conclusiones anteriores y el informe de la semana pasada, las tecnologías de ésta semana no tienen un deployment complejo, pero se encuentran dificultades a la hora de instalar un entorno de desarrollo amigable. Es una solución óptima cuando se requiera complejidad, pero no rapidez, lo que sí que es cierto, es que es una solución para proyectos grandes por la potencia que tiene el lenguaje, en este caso Java. Se ha hablado del problema de leer datos de una archivo JSON y de que quizás a nivel de escalabilidad ésta opción no es la más óptima por cómo trata el lenguaje éste tipo de archivos, una solución a ésto podría ser buscar alguna librería que nos permitiera escribir en un archivo JSON sin tener que leerlo entero, o hacer uso de bases de datos. La inclusión de bases de datos con Tomcat es bastante sencilla y parece la solución más elegante al problema. Para el problema del entorno de desarrollo, se recomienda el uso de algún IDE que gestione el tema de lanzar el servidor cuando se le requiera y poder probar así las clases aún en fase de desarrollo. Se puede concluir que ésta tecnología presenta posibilidades altamente interesantes por la potencia del lenguaje Java, ésta gran potencia conlleva que su uso no sea trivial, pero tampoco es de una complejidad extrema, por lo que se recomienda su uso cuando sepamos que el proyecto puede llegar a crecer y necesitemos herramientas que respondan a la escalabilidad del proyecto.