**JHipster**

**Instalación, creación y modificación de una app con JHipster**

**Creado**: 3 Junio

**Última modificación**: 3 Junio

Como un punto separado del TFG se ha tenido en cuenta JHipster. En paralelo con la instalación de Hadoop y Hive, se ha instalado y probado JHipster para explorar las opciones que ofrece:

**INSTALACIÓN:**

**Paso 1.** Se ha comprobado que Java está correctamente instalado en el equipo.

**¿Cómo?**

**1.1:** Abriendo una terminal y tecleando *java -version*. Java está **MAL** instalado en el equipo si lo que sale en pantalla es algo del estilo:

“*OpenJDK Runtime Environment (IcedTea 2.5.2) (7u65-2.5.2-3~14.04)*

*OpenJDK Client VM (build 24.65-b04, mixed mode, sharing)”*

**1.2:** Para instalar correctamente Java, se han seguido los pasos indicados en el tutorial de wikihow.com [1]

**Paso 2.** Se ha instalado JHipster en el equipo.

**¿Cómo?**

**2.1:** Una vez que Java estaba bien configurado en el equipo, se ha instalado JHipster siguiendo el tutorial de instalación local con Yarn de la página oficial de JHipster [2].

**CREACIÓN DE UNA APLICACIÓN CON JHIPSTER:**

**Paso 1.** Se ha creado una aplicación base con JHipster.

**¿Cómo?**

**1.1:** Para ello, se ha seguido el tutorial disponible en el apartado de creación de una aplicación con JHipster [3]. A continuación se da una breve descripción de las opciones de creación de una app con JHipster y se indica la elegida para crear la aplicación de prueba.

1. Which *type* of application would you like to create?
   1. **Monolithic application: Aplicación monolítica; es un modelo de aplicación local, de un solo nodo, fácil de desarrollar y usar.**
   2. Microservice application: Aplicación microservicio; en una arquitectura de microservicios, la aplicación con JHipster sería un servicio.
   3. Microservice gateway: Gateway de microservicios; en una arquitectura de microservicios, es el servidor que enruta y provee seguridad a las peticiones.
   4. JHipster UAA server [BETA]: en una arquitectura de microservicios, es un servidor de autenticación OAuth2 que provee seguridad a los microservicios.
2. What is the base name of your application?
   1. **“myFirstJHipsterApp*”.*** Se trata de darle un nombre a la aplicación.
3. Would you like to install other generators from the JHipster Marketplace? (y/N)
   1. Y (JHipster Marketplace [4] es un portal desde donde se pueden instalar módulos adicionales, desarrollados por terceros).
   2. **N - no instalar módulos adicionales.**
4. What is your default Java package name?
   1. Se trata de darle un nombre para el paquete de Java por defecto. Este valor lo almacenará Yeoman para que la próxima vez que se lance el generador, el último valor se establezca por defecto. **“com.mycompany.myapp”.**
5. Do you want to use the JHipster Registry to configure, monitor and scale your application? JHipster Registry es una herramienta gratuita y de código abierto que permite gestionar la aplicación cuando está lanzada. Es obligatorio cuando se usa una arquitectura de microservicios.
   1. Yes - Dejar que el JHipster Registry [5] configure, monitorice y escale la aplicación.
   2. **No - No permitir que JHipster Registry configure, monitorice y escale la aplicación.**
6. Which *type* of authentication would you like to use?
   1. **JWT authentication (stateless, with a token) - Autenticación con JWT.**
   2. HTTP Session Authentication (stateful, default Spring Security mechanism) - Autenticación con sesiones sobre HTTP. Es la manera clásica de autenticación y se puede usar con Spring Social [6], que permite usar login con terceros (Google, Facebook, Twitter).
   3. OAuth2 Authentication (stateless, with an OAuth2 server implementation) - Autenticación mediante OAuth 2.0.
7. Which *type* of database would you like to use?
   1. **SQL (H2, MySQL, MariaDB, PostgreSQL, Oracle, MSSQL) - Una base de datos SQL, que será accesible a través de Spring Data JPA [7]**
   2. MongoDB
   3. Cassandra
8. Which *production* database would you like to use? - Se trata de la base de datos que se va a usar en producción.
   1. **MySQL**
   2. MariaDB
   3. PostgreSQL
   4. Oracle
   5. Microsoft SQL Server
9. Which *development* database would you like to use? - Se trata de la base de datos que se va a usar en desarrollo.
   1. **H2 with disk-based persistence (es la manera más rápida de configurar JHipster, pero los datos se borrarán cuando se reinicie el servidor).**
   2. H2 with in-memory persistence (Está aún en BETA y no funciona para windows) es una buena manera que la anterior puesto que los datos no se perderán al reiniciar el servidor.
   3. MySQL - La misma usada para producción. Es un poco más complicado de configurar pero es más eficiente.
10. Do you want to use Hibernate [8] 2nd level cache? (Hibernate es el proveedor de JPA (Java Persistence API) que usa JHipster) Por razones de rendimiento, es altamente recomendable el uso de una caché. No obstante, para la prueba de concepto con JHipster no será necesario.
    1. Yes, with ehcache (local cache, for a single node) - Usar la cache de Hibernate de nivel 2 (cache local, para un nodo singular).
    2. Yes, with HazelCast (distributed cache, for multiple nodes) - Usar la caché de HazelCast (distribuida, para múltiples nodos).
    3. **No - No usar caché para la base de datos.**
11. Would you like to use Maven or Gradle?
    1. Maven - Usar Maven para compilado y dependencias
    2. **Gradle - Usar Gradle para compilado y dependencias**
12. Which other technologies would you like to use? (Press <space> to select, <a> to toggle all, <i> to inverse selection) - Aquellos servicios o tecnologías extras que se pueden añadir sobre JHipster
    1. **◯ Social login (Google, Facebook, Twitter) - Opción únicamente disponible si se ha seleccionado una base de datos SQL o MongoDB. Añade soporte Spring Social para que los usuarios puedan loguearse con sus cuentas de Google, Facebook o Twitter.**
    2. ◯ Search engine using Elasticsearch [9] (Elasticsearch es un motor de búsqueda distribuida RESTful. Se integra con Spring Data Elasticsearch [10])
    3. ◯ Clustered HTTP sessions using Hazelcast - Si se ha elegido la opción de sesiones sobre HTTP en la pregunta 6 y además se ha elegido una aplicación clusterizada, esta opción se debe marcar puesto que es la que permite replicación de las sesiones en un cluster.
    4. ◯ WebSockets using Spring Websocket. (WebSockets mediante Spring Websocket [11]).
    5. ◯ [BETA] Asynchronous messages using Apache Kafka [12] (Apache Kafka como broker de mensajería publicador / suscriptor).
13. Which *Framework* would you like to use for the client?
    1. **AngularJS 1.x**
    2. AngularJS 4 (BETA) - No se recomienda para producción.
14. Would you like to use the LibSass stylesheet preprocessor for your CSS? (Utiliza Node-Sass [13] para diseñar CSS de manera simplificada)
    1. **Y utilizar LibSass**
    2. N no utilizarlo
15. Would you like to enable translation support with Angular Translate? (Por defecto JHipster ofrece gran soporte para internalización, pero esta opción añade algunos aspectos extra y es un poco más difícil de gestionar).
    1. Y (habilitar)
    2. **N (no habilitar)**
16. Besides JUnit and Karma, which testing frameworks would you like to use? (Press <space> to select, <a> to toggle all, <i> to inverse selection)
    1. ◯ Gatling (Tests de rendimiento)
    2. ◯ Cucumber (Tests de comportamiento)
    3. ◯ Protractor (Integración de tests para Angular)

**MODIFICACIÓN DE LA APLICACIÓN CREADA:**

**Paso 1.** Se ha modificado la aplicación anterior añadiéndole otro esquema de base de datos.

**¿Cómo?**

**1.1:** Se ha creado un fichero .jh con el esquema deseado en la raíz del proyecto (en este caso, un esquema de base de datos que gestiona libros, editoriales y librerías):

**FICHERO *books.jh*:**

*entity Libro {*

*nombreLibro String required,*

*anioCreacion Integer,*

*anioPublicacion Integer*

*}*

*entity Autor {*

*nombreAutor String required,*

*edad Integer*

*}*

*// an ignored comment*

*/\*\* not an ignored comment \*/*

*entity Libreria {*

*direccion String required,*

*codigoPostal String,*

*ciudad Ciudad,*

*provincia String*

*}*

*entity Editorial {*

*nombreEditorial String required*

*}*

*enum Ciudad {*

*HUESCA, ZARAGOZA, TERUEL*

*}*

*relationship ManyToMany {*

*Libreria{contiene} to Libro,*

*Editorial{vende} to Libreria*

*}*

*relationship OneToMany {*

*Autor{pertenece} to Editorial,*

*Autor{escribe} to Libro*

*}*

*// Set pagination options*

*paginate Autor with infinite-scroll*

*paginate Libro with pagination*

*dto \* with mapstruct*

*// Set service options to all except few*

*service all with serviceImpl except Ciudad*

*// Set an angular suffix*

*angularSuffix \* with mySuffix*

**1.2:** Se ha ejecutado en la consola el comando *jhipster import-jdl books.jh*

**1.3:** Se ha indicado la opción *a* para indicar que se sobreescriban todos los conflictos con el esquema de base de datos ya existente previamente.

**PARA COMPLETAMENTE CAMBIAR EL ESQUEMA DE LA BASE DE DATOS:**

0. Suponemos que en la carpeta <raíz del proyecto> existe un fichero llamado *schema.jh* coon el esquema de la base de datos en jdl.

1**.** Acceder al fichero *<raíz del proyecto>/src/main/resources/config/liquibase/master.xml*

1.1 Borrar todas las líneas que contienen el tag include MENOS esta:

<include file="classpath:config/liquibase/changelog/00000000000000\_initial\_schema.xml" relativeToChangelogFile="false"/>

2.Acceder a la carpeta *<raíz del proyecto>/src/main/resources/config/liquibase/changelog*

*2*.1 Borrar todos los ficheros menos este: *00000000000000\_initial\_schema.xml*

3. Acceder a la carpeta *<raíz del proyecto>/.jhipster* y borrar todos los ficheros

**Lo anterior es la parte del servidor. A continuación, hay que borrar lo que no nos interesa del frontend:**

4. Acceder a *<raíz del proyecto>/src/main/webapp/app/entities* y borrar todas las carpetas (no ficheros) de allí.

5. Acceder a <raíz del proyecto>/src/main/webapp/app/layouts/navbar/navbar.html y borrar del navbar las entidades previamente existentes (aparecen en un dropdown con los tags <li ...>).

6. Borrar todos las lineas referentes a los scripts de las entidades de nuestro proyecto desde el fichero index.html localizado en *<raíz del proyecto>/src/main/webapp/index.html*. Son líneas del tipo:

<script src="app/entities/coches/coches.state.js"></script>

**Lo anterior ha sido la parte del frontend. A continuación hay que ejecutar lo siguiente:**

Desde la consola, dentro de <raíz del proyecto> ejecutar: ./gradlew clean

6. Modificamos el contenido del fichero schema.jh y cambiamos el esquema de la base de datos.

7. Ejecutamos *jhipster import-jdl schema.jh* y veremos como el esquema de la base de datos se configura según nuestras necesidades.

[1]: Instalación Java: <http://www.wikihow.com/Install-Oracle-Java-JDK-on-Ubuntu-Linux>

[2]: Instalación JHipster: <https://jhipster.github.io/installation/>

[3]: Creación de una aplicación con JHipster: <https://jhipster.github.io/creating-an-app/>

[4]: JHipster Marketplace: [https://jhipster.github.io/modules/marketplace/#/list](https://jhipster.github.io/modules/marketplace/" \l "/list)

[5]: JHipster Registry: <https://jhipster.github.io/jhipster-registry/>

[6]: Spring Social: <http://projects.spring.io/spring-social/>

[7]: Spring Data JPA: <https://docs.spring.io/spring-data/jpa/docs/current/reference/html/>

[8]: Hibernate: <http://hibernate.org/>

[9]: Elasticsearch: <https://github.com/elastic/elasticsearch>

[10]: Spring Data Elasticsearch: <https://github.com/elastic/elasticsearch>

[11]: Spring WebSockets: <https://docs.spring.io/spring/docs/current/spring-framework-reference/html/websocket.html>

[12]: JHipster con Apache Kafka: <https://jhipster.github.io/using-kafka/>

[13]: Node-sass: <https://www.npmjs.com/package/node-sass>