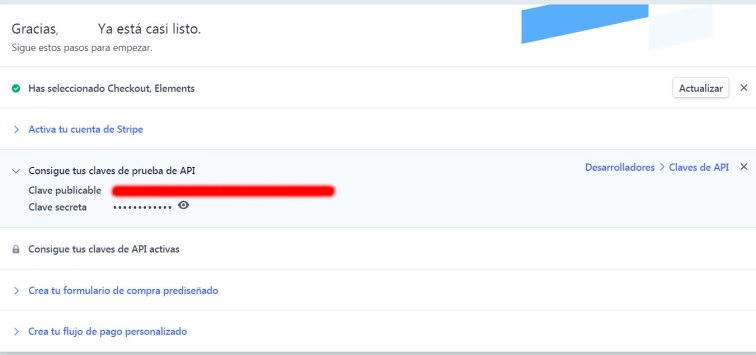
SPRING BOOT + ANGULAR +STRIPE

# PAGOS CON STRIPE EN SPRING BOOT Y ANGULAR 8. CAPÍTULO 2: CREANDO EL BACKEND

**Cuenta en Stripe:**

Lo primero que tenemos que hacer es crear una cuenta en Stripe. Para ello vamos al [sitio web](https://stripe.com/es) y si no la tenemos creada vamos al registro (botón “Empezar Ahora”). Nos pedirá el nombre, un correo y una contraseña.

Una vez que la tengamos iniciamos sesión. En la pantalla de inicio -dashboard- nos interesan las 2 claves de prueba (pública y secreta). Pulsamos sobre “Consigue tus claves de prueba de API” y nos las mostrará.



**Proyecto Spring Boot:**

Vamos a la página de [Spring Initializr](https://start.spring.io/) y creamos un proyecto (yo lo ha llamado TutorialStripe).  Añadimos las siguientes dependencias:

* JPA
* MySQL
* WEB

Le damos a descargar, lo descomprimimos  y lo importamos con Intellij.

**pom.xml:**

Vamos a necesitar dos dependencias más: commons-lang3 y la de Stripe. El archivo pom.xml nos quedará así:

|  |  |
| --- | --- |
|  | <?xml version="1.0" encoding="UTF-8"?> |
|  | <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" |
|  | xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"> |
|  | <modelVersion>4.0.0</modelVersion> |
|  | <parent> |
|  | <groupId>org.springframework.boot</groupId> |
|  | <artifactId>spring-boot-starter-parent</artifactId> |
|  | <version>2.1.8.RELEASE</version> |
|  | <relativePath/> <!-- lookup parent from repository --> |
|  | </parent> |
|  | <groupId>com.cavanosa</groupId> |
|  | <artifactId>TutorialStripe</artifactId> |
|  | <version>0.0.1-SNAPSHOT</version> |
|  | <name>TutorialStripe</name> |
|  | <description>Demo project for Spring Boot</description> |
|  |  |
|  | <properties> |
|  | <java.version>1.8</java.version> |
|  | </properties> |
|  |  |
|  | <dependencies> |
|  | <dependency> |
|  | <groupId>org.springframework.boot</groupId> |
|  | <artifactId>spring-boot-starter-data-jpa</artifactId> |
|  | </dependency> |
|  | <dependency> |
|  | <groupId>org.springframework.boot</groupId> |
|  | <artifactId>spring-boot-starter-web</artifactId> |
|  | </dependency> |
|  |  |
|  | <dependency> |
|  | <groupId>mysql</groupId> |
|  | <artifactId>mysql-connector-java</artifactId> |
|  | <scope>runtime</scope> |
|  | </dependency> |
|  | <dependency> |
|  | <groupId>org.springframework.boot</groupId> |
|  | <artifactId>spring-boot-starter-test</artifactId> |
|  | <scope>test</scope> |
|  | </dependency> |
|  | <!-- https://mvnrepository.com/artifact/org.apache.commons/commons-lang3 --> |
|  | <dependency> |
|  | <groupId>org.apache.commons</groupId> |
|  | <artifactId>commons-lang3</artifactId> |
|  | <version>3.4</version> |
|  | </dependency> |
|  | <!-- https://mvnrepository.com/artifact/com.stripe/stripe-java --> |
|  | <dependency> |
|  | <groupId>com.stripe</groupId> |
|  | <artifactId>stripe-java</artifactId> |
|  | <version>12.0.0</version> |
|  | </dependency> |
|  | </dependencies> |
|  |  |
|  | <build> |
|  | <plugins> |
|  | <plugin> |
|  | <groupId>org.springframework.boot</groupId> |
|  | <artifactId>spring-boot-maven-plugin</artifactId> |
|  | </plugin> |
|  | </plugins> |
|  | </build> |
|  |  |
|  | </project> |

[**view raw**](https://gist.github.com/cavanosa/a1d0616e645cf624c5cd99c3eb75e083/raw/66962a00f999984ba72bdf9d15edba1f5165e756/pom.xml)[**pom.xml**](https://gist.github.com/cavanosa/a1d0616e645cf624c5cd99c3eb75e083#file-pom-xml) hosted with  by [**GitHub**](https://github.com/)

**application.properties:**

En este archivo configuramos la conexión a la base de datos y ponemos las claves de Stripe:

|  |  |
| --- | --- |
|  | spring.datasource.url = jdbc:mysql://localhost:3306/tutorial\_stripe?useSSL=false&serverTimezone=UTC&useLegacyDateTimeCode=false |
|  |  |
|  | # nombre de usuario y contraseña |
|  | spring.datasource.username = username |
|  | spring.datasource.password = password |
|  |  |
|  | # mostrar sentencias SQL en la consola |
|  | spring.jpa.show-sql = true |
|  |  |
|  | # actualizar base de datos y crear entidades |
|  | spring.jpa.hibernate.ddl-auto = update |
|  |  |
|  | # hibernate genera SQL optimizado |
|  | spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect |
|  |  |
|  | #stripe keys |
|  | stripe.key.public=tu clave pública |
|  | stripe.key.secret=tu clave secreta |

[**view raw**](https://gist.github.com/cavanosa/a71647b67d988cea37c8a3044c637967/raw/dc601ec9fe32b952a7073f9e3068c2aa710533da/application.properties)[**application.properties**](https://gist.github.com/cavanosa/a71647b67d988cea37c8a3044c637967#file-application-properties) hosted with  by [**GitHub**](https://github.com/)

En este tutorial llega con poner sólo la clave secreta pues la pública se va a enviar desde el lado del cliente.

**Base de datos:**

Nos queda crear la base de datos MySQL. En el application.properties hemos indicado que su nombre es tutorial\_stripe:

CREATE DATABASE tutorial\_stripe CHARACTER SET utf8 COLLATE utf8\_general\_ci;

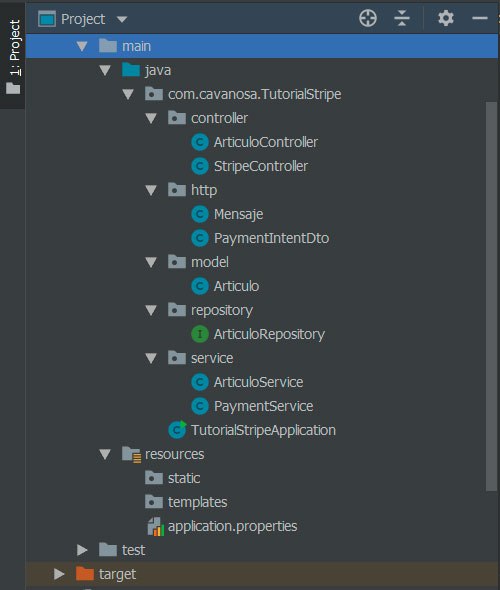
Una vez que hayamos creado la base de datos volvemos a editor, Intellij en nuestro caso; si todo está OK, al ejecutar el proyecto no nos lanzará ningún error.

**Paquetes del proyecto:**

Crearemos 5 paquetes:

* **controller**: donde irán las clases ArticuloController y StripeController
* **http**: otras dos clases, Mensaje y PaymentIntentDto
* **model**: la clase Articulo
* **repository**: la interfaz ArticuloRepository
* **service**: las clases ArticuloService y PaymentService

Este es el árbol de archivos una vez que tengamos todas las clases y paquetes:



En el siguiente capítulo haremos el CRUD de la clase Articulo.