Características de la programación reactiva:

* Crea fácilmente eventos streams o flujos de datos continuos.
* Transforma y compone flujos, con los operadores map, filter, merge, delay, forEach etc.
* Subscribir a cualquier flujo observable para realizar alguna tarea.
* Multiplataforma, porque se pueden encontrar implementaciones diferentes lenguajes como Java, JavaScript, Scala, C#, C++, Python, PHP y otros.
* Trabajan con flujos de datos continuos asíncronos.
* Contrapresión o latencia (non blocking).
* Los operadores reducen notablemente las tareas de transformaciones.
* Pueden ser creados a partir de otras fuentes como stream, listas, intervalos, rangos, etc.
* Son cancelables.
* Puede ser finitos o infinitos.
* Son inmutables.
* Concurrencia hecha más simple.
* Manejo de errores asíncrono.
* Al fallar se puede volver a reintentar.

[Reactive Streams](https://www.reactive-streams.org/)

Reactive Streams es una iniciativa para proporcionar un estándar para el procesamiento asíncrono de flujos con contrapresión no bloqueante. Abarca esfuerzos dirigidos a entornos en tiempo de ejecución (JVM y JavaScript), así como a protocolos de red.

[Webflux](https://docs.spring.io/spring-framework/reference/web/webflux.html)

Spring Webflux es un módulo web reactivo, totalmente sin bloqueo, que admite contrapresión de flujos reactivos y se ejecuta en servidores como netty, Undertow y contenedores Servlet 3.1+ Spring

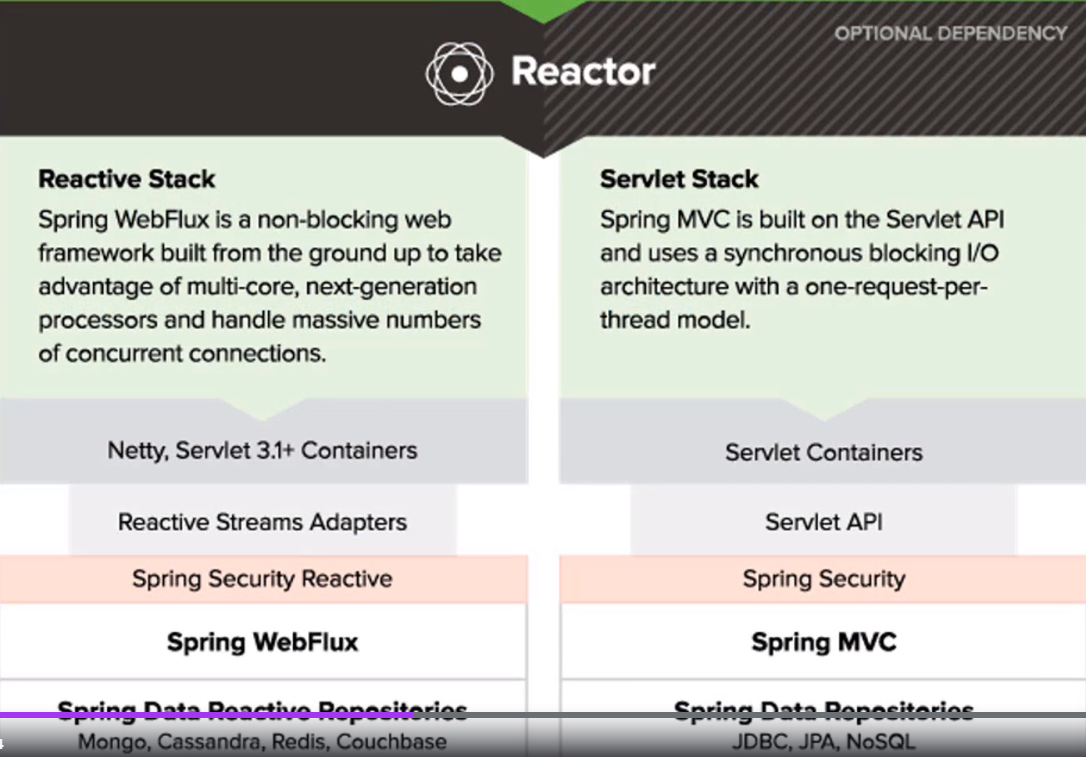
Web MVC vs Webflux

**Spring Web**: basado en la API Servlet, es síncrona y bloqueante, crea un request por hilo.

**Webflux**: nos permite crear aplicaciones web, basadas en la especificación Reactive Streams. Es totalmente no bloqueante y permite la contrapresión de flujos. Se ejecuta en servidores sin bloqueo como:

* Netty.
* Undertow.
* Tomcat (Contenedor Servlet 3.1).

Nota: **ambos conviven dentro del ecosistema de Spring Framework.**

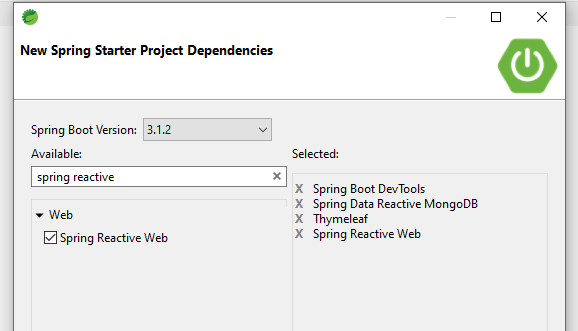


Requerimientos para trabajar un proyecto con Webflux:

* Instalar MongoDB Community Server.
* Configurar variables de entorno.
* Instalar Robot 3T o Robot Studio 3T.

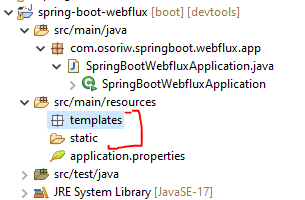
Creando proyecto con Webflux

* Crear proyecto con la opción Spring Starter Project
* Agregar las siguientes dependencias:

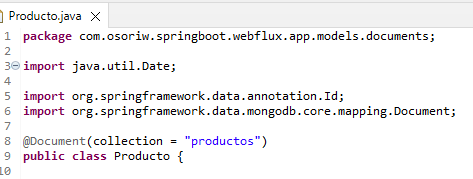


**Nota**: la dependencia Webflux, incluye por debajo Reactor.

* Crear las siguientes carpetas **templates** y **static** bajo la ruta **src/main/resources**:

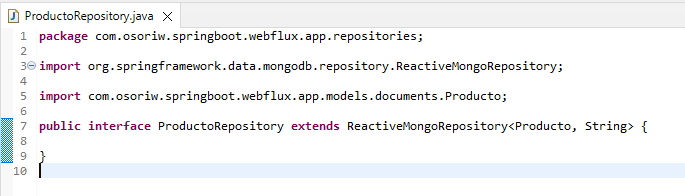


* Crear las clases del modelo de datos, como se va a trabajar con Mongo, las clases del modelo deben estar definidas de la siguiente forma:

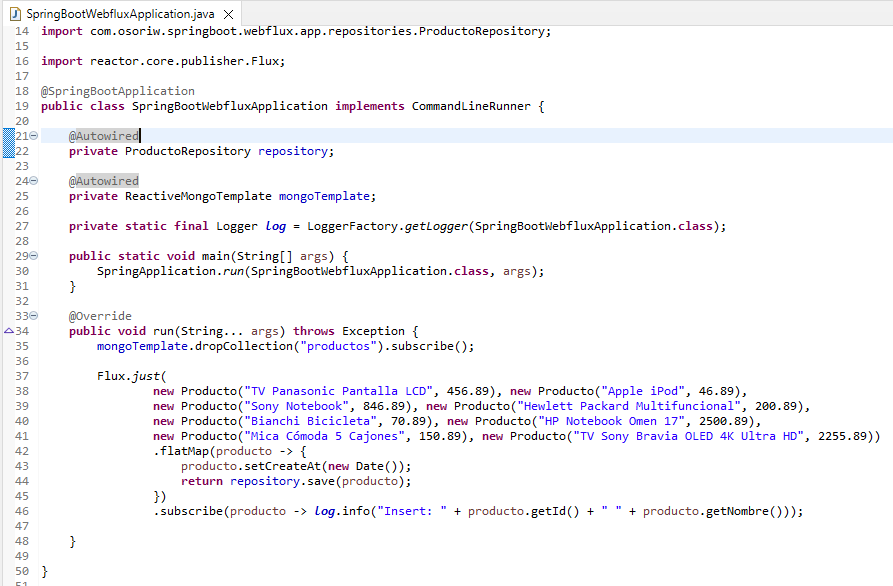


**Nota**: esta clase representa el documento que finalmente se va a guardar dentro de la colección de mongo que es un Json binario, llamado BSON.

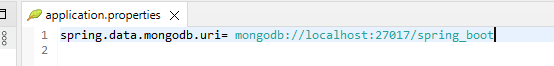
* Crear la interface repository para el acceso objeto-relacional, extendiendo de **ReactiveMongoRepository:**



* Insertar datos de prueba con la ayuda del método run de la interface CommandLineRunner, usando el siguiente esquema de ejemplo:



* Cambiando el nombre de la base de datos:



* Creando el controlador reactivo:
  + Crear un paquete para los controladores del proyecto.
  + Crear una clase Controller.
  + Implementar un endpoint para consultar la lista de productos.
  + Implementar un endpoint para consultar un producto por id.

