**REQUERMIENTO TÉCNICOS:**

* LENGUAGE\_BACKEND

JAVA => Versión 8.

* FRAMEWORK\_BACKEND

SPRINGBOOT => Versión 2.6.4

* IDE

ECLIPSE IDE => Versión 2023-03 (4.27.0)

VISUAL STUDIO CODE => Versión 1.77.3

* BASE\_DE\_DATOS

ORACLE SQL DEVELOPER => Versión 22.2.1.234

* FRAMEWORK\_FRONTEND

BOOTSTRAP => Versión 4.0.0

* GIT

Git => Versión 2.37.2.windows.2

**DEPENDENCIAS:**

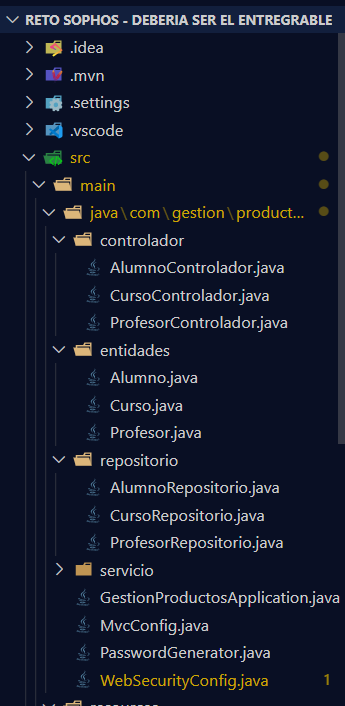
Para la inicialización del proyecto contemplo el uso de las siguientes dependencias para el proyecto en Spring Boot:

* Spring Boot DevTools -> Provides fast application restarts, LiveReload, and configurations for enhanced development experience.
* Validation - > Bean Validation with Hibernate validator.
* Spring Data JPA -> Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.
* Sprig Data JDBC -> Persist data in SQL stores with plain JDBC using Spring Data.
* Thyme leaf -> A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.
* Thymeleaf-extras-springsecurity5 -> A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.
* Spring Web -> Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

**PROYECTO:**

El proyecto se trabajó orientado en un MVC = Modelo – Vista – Controlador = Model – View – Controller

**CONTROLADORES**



En el cual se trabajaron los siguientes, 3 controladores, uno para Alumno, Curso y Profesor, en los cuales se configuraron para cada uno sus respectivos **@RequestMapping** para acceder a las rutas “url” y designarles una función en específico para controlar la información respecto a los verbos **REST(GET, POST,PUT,DELTE),** marcando también como **@Controller**, y una anotación **@Autowired**, que se encarga de inyectar automáticamente una instancia del servicio de cursos en la clase controlador cuando se necesita de la siguiente manera:

Texto

Descripción generada automáticamente

**ENTIDADES – MODELOS**

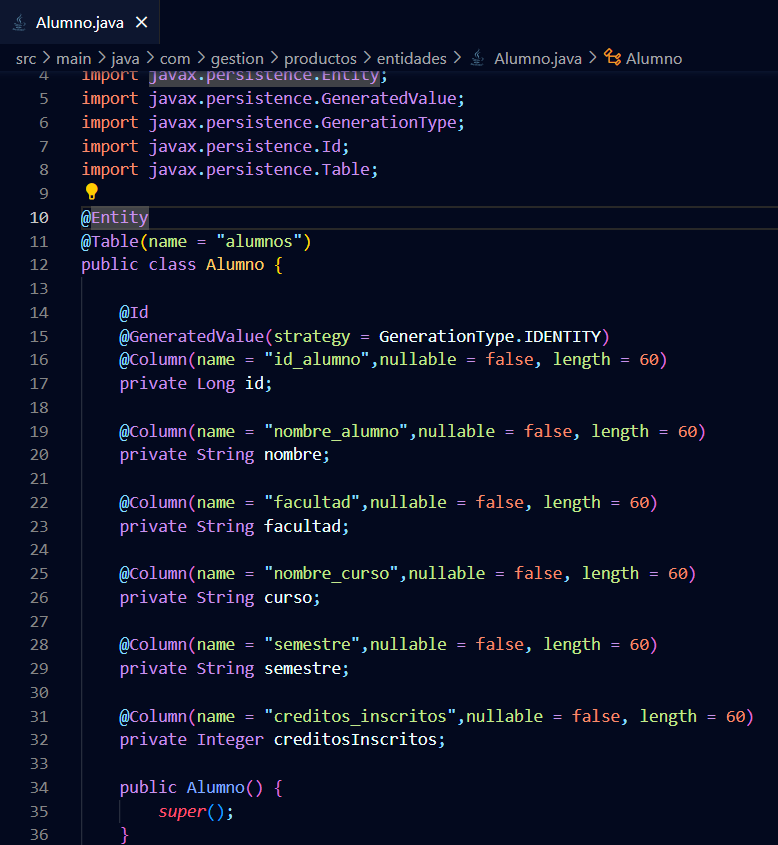
En cuanto a los modelos se realizó lo siguiente, se crearon 3 modelos, ya que era necesario 1 para Alumnos, Cursos y Profesores.

Texto

Descripción generada automáticamente con confianza media

Como la carpeta de Modelo-Entidades es una carpeta en la cual se inicializa una clase con la que se toma como estructuración de las tablas de la base de datos por medio HIBERNATE se debe configurar los atributos de la siguiente manera:

Se le designa como tipo **@Entity** para trabajar con la Base de Datos deseada, y se le añade **@Table(name = “ ”)** para que tome los atributos, el constructor de la clase y pueda crear las respectivas tablas en la base de datos.



Donde al primer atributo que será el “Id” se le designa como **@Id** para que sea tomado como “Primary Key” de la tabla, seguidamente de agrega un **@GeneratedValue(strategy = GenerationType.IDENTITY)** para autoincrementar el valor del “Id”, seguidamente de la configuración de cada atributo con el **@Column()** con el cual se puede designar los nombres de las columnas y algunas características al gusto del creador o petición del cliente.

Texto

Descripción generada automáticamente

Sin olvidar la puesta de los debidos **constructores** y los **getters and setters**.

**REPOSITORIOS**

Para lo que son los repositorios de códigos deben crease interfaces que hereden de una clase predeterminada llamada (**JpaRepository<,>)**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Aquí dentro de la interfaz se deben colocar los **@Query** que se vayan a trabajar, pero cada solicitud debe ser escrita como si se estuviera manejando los objetos creados dentro del proyecto, no funciona si la consulta se hace en las denominaciones con las que se manejan las instancias en la base de datos, pues si es trabajo de esa manera no da ningún resultado y genera error, es por esto por lo que el **@Query** debe ser escrito como objetos así:

Texto

Descripción generada automáticamente

**SERVICIOS**

En cuanto a los servicios, en esta sección se introducen las funciones que le dan estructuración a un CRUD = Create – Read – Update – Delete

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

@Service es una anotación que se utiliza para marcar una clase como un componente de servicio en una aplicación. Escribiendo las siguientes que dan las estructura al CRUD de la siguiente manera:

Texto

Descripción generada automáticamente

**METODOS ADICIONALES**

**Texto

Descripción generada automáticamente**

En la clase “GestionProductosApplication” se tiene el método MAIN principal.

Texto, Escala de tiempo

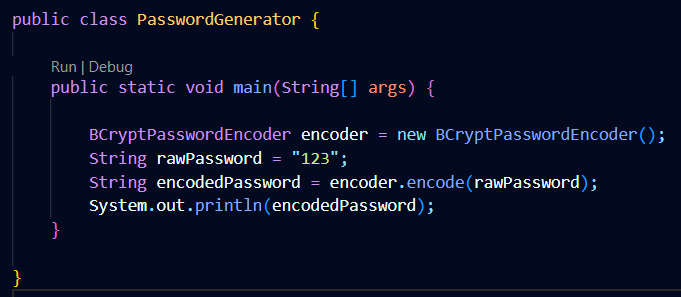
Descripción generada automáticamente

En cuanto a la clase **MvcConfig** se debe configurar para que implemente las características de una clase predeterminada llamada **WebMvcConfigurer** que le permitirá utilizar funciones de seguridad para una dirección “url” especifica que se le indique, y hacer una validación de usuario.

Texto

Descripción generada automáticamente

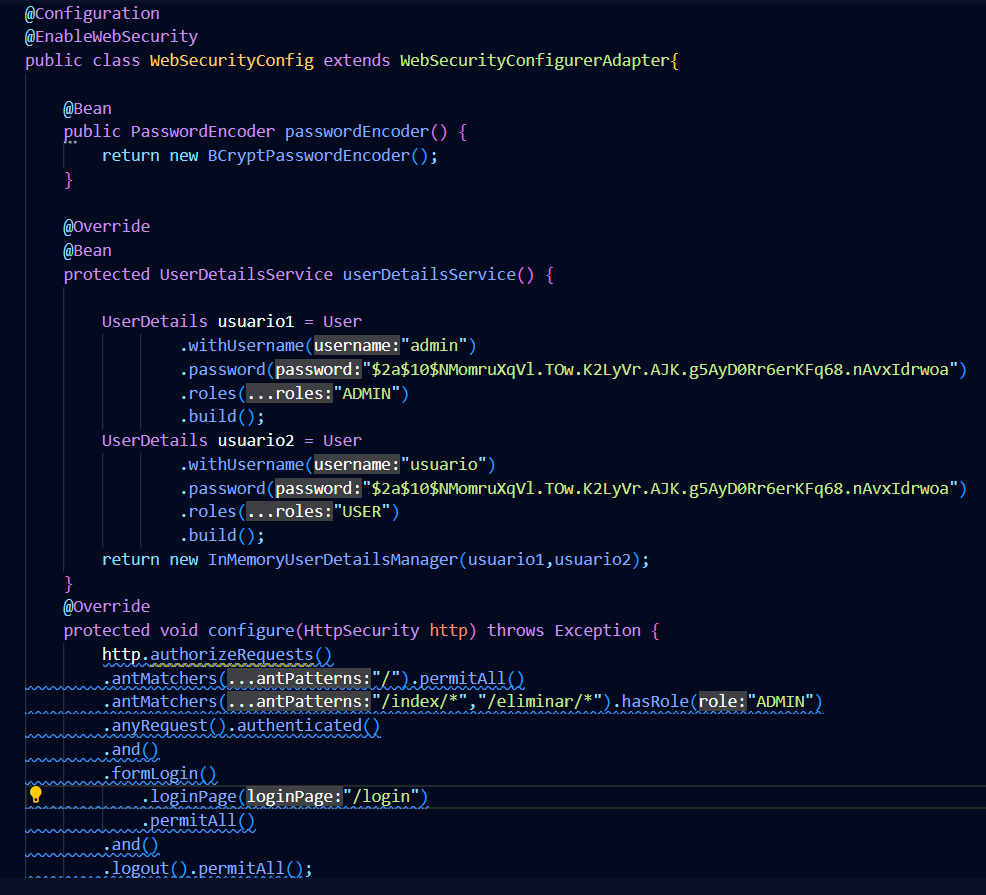
En la clase **PasswordGenerator** Se configura un objeto por medio del cual se encripta la contraseña de caracteres en una **String** que nosotros le indiquemos, en este caso “**123**”.



Para la clase **WebSecurityConfig** se tiene que una clase predeterminada llamada **WebSecurityConfigurerAdapter** que le permite a la clase que nosotros creamos acceder a todos los componentes de ella, pudiendo configurar el **password** de los **“roles”** que nosotros le indiquemos, en este caso hay dos, el **“rol”** de **“admin**” y el rol de **“usuario”**, cada uno tiene facultad de hacer acciones dentro del aplicativo, pero el **“admin”** tiene mas **“privilegios”** que el **“usuario”.**

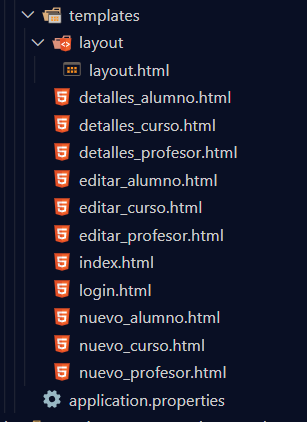
La anotación **@Configuration** se utiliza junto con otras anotaciones como **@Bean** para configurar y definir los **beans** en una aplicación de Spring.

**@Bean** es una anotación que se utiliza en un método que registra el objeto devuelto por ese método como un **bean** en el contenedor de Spring.



**VISTA**

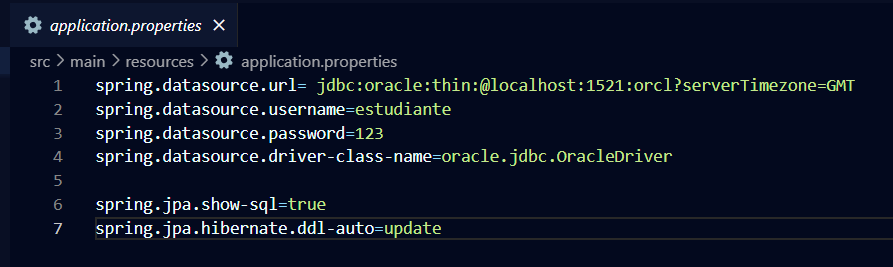
Para la vista se trabajaron varios archivos.html para dar funcionalidad los verbos (GET, PUT, POST, DELETE) determinados en el controlador, también el archivo con **layout** en el patrón de arquitectura Modelo-Vista-Controlador, se utiliza para definir cómo se visualiza la Vista en la aplicación web. En otras palabras, el **layout** define la estructura y el estilo general de la página que se muestra al usuario. Denominado los siguientes:



En el archivo **index.html**, se definió la vista principal, y los respetivos **nuevos, detalles y editar** de cada objeto, y un archivo llamado **login.html** que muestra una vista que estructurada con la **autenticaciones, roles y seguridad** de la clase **WebSecurityConfig.**

**BASE DE DATOS**

Para la correcta configuración de la Base de Datos, es importante configurar la url, username, password y el driver que esta debe manejar.

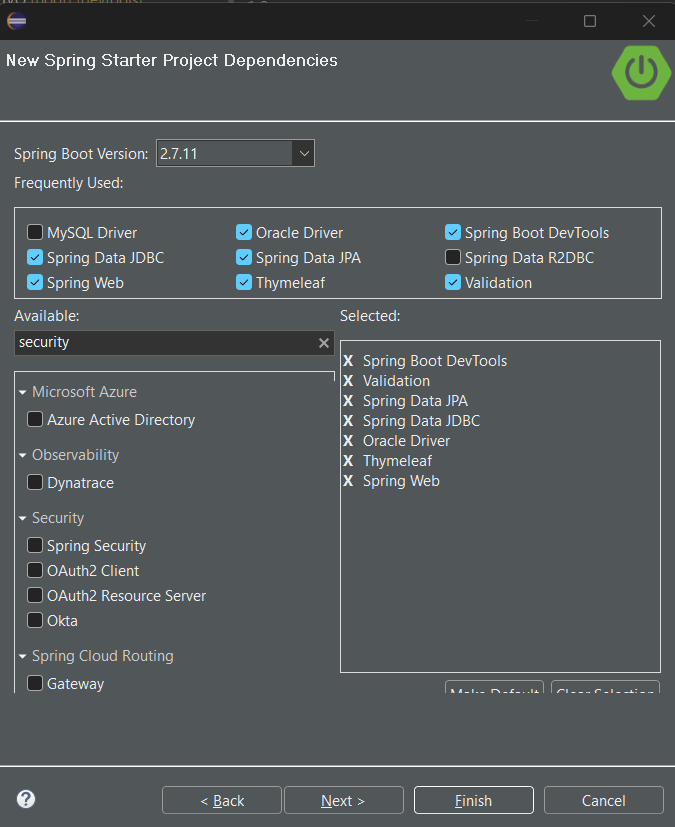


En cuanto a los **spring.jpa.show-sql = true** nos mostrará las sentencias **SQL** por consola del **IDE** que se ejecuten y se reflejen en la base de datos.

En cuanto al **spring.jpa.hibernate.ddl – auto = create** ; proporciona la facultad de permitir que el archivo controller ejecute y cree en la base de datos los parámetros de objetos en sentencias y estructuras **SQL** en la **base de datos Oracle SQL**, para el correcto funcionamiento del aplicativo es necesario iniciar esta configuración con “**create**” y después de ejecutado 1 vez cambiar el “**create**” por un “**update**” y así las tablas ya no se volverán a crear o generar error si no que se que se actualizaran cuando se haga un cambio en el aplicativo.

**INFORMACION ADICIONAL**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico, Teams

Descripción generada automáticamente

Captura de pantalla de un celular

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente