# Desafío literalura

Este proyecto, llamado "Desafío Literalura", fue realizado como parte de la formación en el programa Oracle Next Education (Oracle ONE). Es una aplicación de consola en Java que permite a los usuarios buscar libros por título, listar libros y autores registrados, listar autores vivos en un determinado año, listar libros por idioma, ver los 10 libros más descargados y obtener estadísticas.

## Descripción del proyecto

El proyecto utiliza Java como lenguaje de programación, Spring Boot como framework y Maven para la gestión de dependencias. La estructura del proyecto sigue la estructura estándar de Maven y está organizado en varios paquetes, incluyendo modelos, repositorios y servicios.

La aplicación consume la API de Gutendex para obtener datos de libros y autores, y luego convierte estos datos en objetos de modelo que se pueden manipular dentro de la aplicación. Los datos de los libros y autores se almacenan en una base de datos, lo que permite a los usuarios realizar búsquedas y obtener estadísticas.

La comunicación con la aplicación se realiza a través de la consola, con el siguiente menú:

1 - Buscar libros por título

2 - Listar libros registrados

3 - Listar autores registrados

4 - Listar autores vivos en un determinado año

5 - Listar libros por idioma

6 - Top 10 libros más descargados

7 - Obtener estadísiticas

0 - Salir

## Organización del código

El proyecto sigue la estructura estándar de Maven y está organizado en varios paquetes:

- `com.alura.literalura.models`: Contiene las clases de modelo para Datos, Libros, DatosLibros, Autores y DatosAutores.

- Datos.java: Record que hace la conexión entre la lista de libros (la cual está contenida en el array "results") del Json y la aplicación.

- DatosLibros.java: Record que hace la conexión entre los datos de los libros del Json y las variables de la aplicación.

- DatosAutores.java: Record que hace la conexión entre los datos de los autores de cada libro del Json (los autores y sus datos están dentro de otro array) y la aplicación.

- Libros.java: Clase que define los atributos de los libros y crea su respectiva tabla en la base de datos. Tiene conexión ManyToOne con el modelo Autores.

- Autores.java: Clase que define los atributos de los autores y crea su respectiva tabla en la base de datos. Tiene relación OneToMany con el modelo Libros.

- `com.alura.literalura.repositorio`: Contiene las interfaces de repositorio para interactuar con la base de datos.

- IAutoresRepository.java: Repositorio para hacer consultas y busquedas relacionadas con los autores en la base de datos.

- ILibrosRepository.java: Repositorio para hacer consultas y busquedas relacionadas con los libros en la base de datos.

- `com.alura.literalura.service`: Contiene las clases de servicio para consumir la API de Gutendex y convertir los datos.

- ConsumoApi.java: Se encarga de obtener los datos de la API por medio del link proporcionado, y devuelve los datos como Json.

- ConvierteDatos.java: Se encarga de convertir el Json en una clase (modelo) genérica.

- IConvierteDatos.java: Esta interfaz se encarga de definir el tipo de dato genérico.

- `com.alura.literalura.principal`: Contiene la clase Principal que maneja la lógica principal de la aplicación.

- Principal.java: Es la clase principal de la aplicación que maneja la interacción con el usuario y la ejecución de las funcionalidades, y cada opción mostrada en el menú se realiza con propio método interno, igual definido en principal.

Asimismo, contamos con:

- Las dependencias necesarias:

- Jackson Dataformat

- Jackson Core

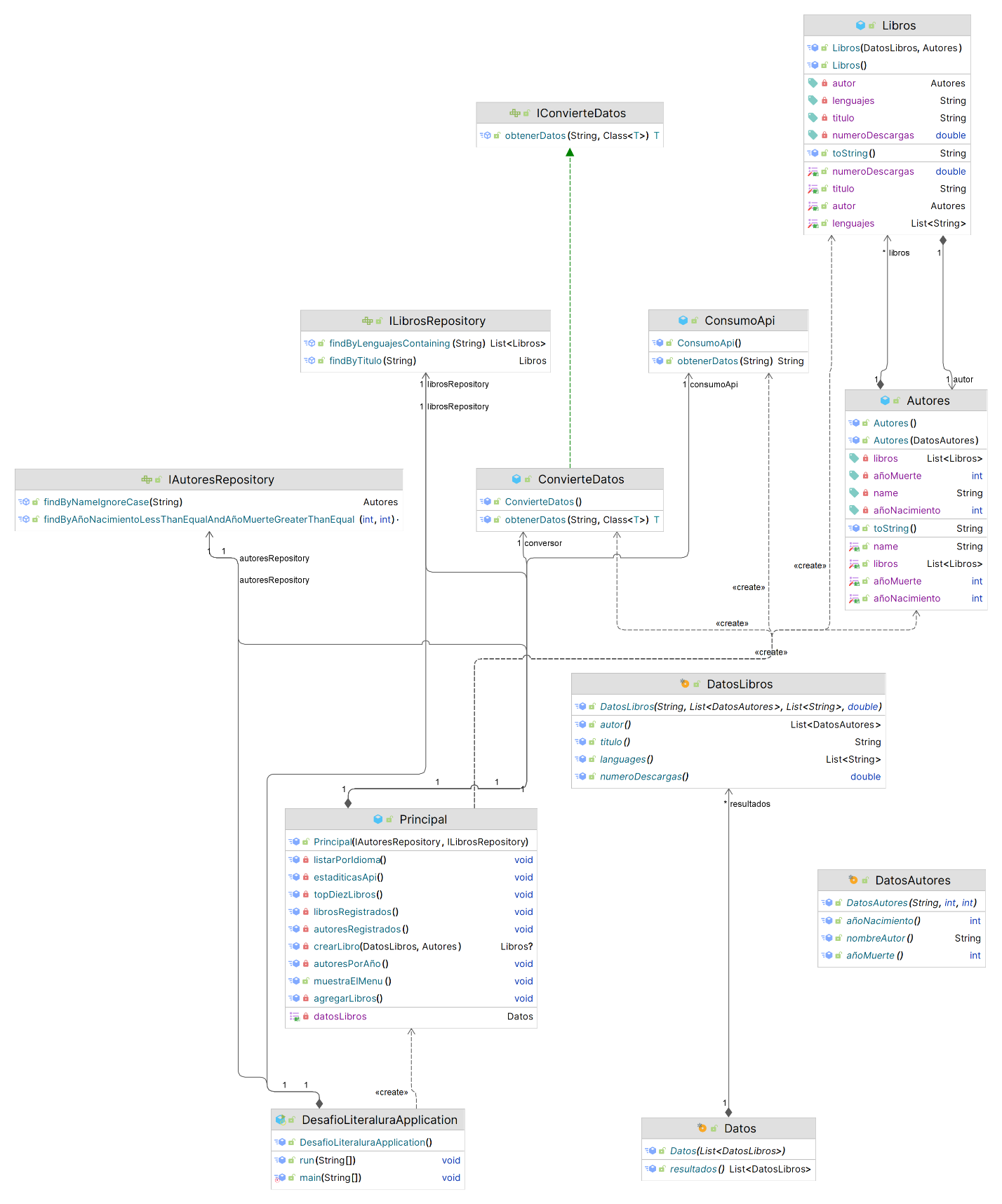
- Spring Data JPA

- PostgreSQL Driver

- La clase de arranque `LiteraluraApplication`, encargada iniciar la aplicación, crear el contexto de Spring y ejecuta la lógica principal de la aplicación a través de la clase Principal.

- La declaración y configuración de la base de datos en el archivo `application.properties`

## Organización del código de manera gráfica



## Como usar la aplicación

\*\*Clonar el repositorio\*\*

Para empezar, necesitas clonar el repositorio en tu máquina local.

O si lo prefieres, puedes presionar el botón verde `code<>` del repositorio y descargar los archivos como ZIP

1. Configurar la base de datos

La aplicación utiliza una base de datos PostgreSQL. Deberás crear una base de datos local y configurar las credenciales de la base de datos en el archivo src/main/resources/application.properties. Reemplaza ${DB\_HOST}, ${DB\_NAME\_LITERALURA} ${DB\_USER} y ${DB\_PASSWORD} con tus propios valores (Puede ser en el mismo archivo properties, o creando tus propias variables de entorno).

- DB\_HOST: el nombre de tu host (si la base de datos está en tu computadora, pon localhost).

- DB\_NAME\_LITERALURA: el nombre de tu base de datos.

- DB\_USER: el nombre del usuario de la base datos (por default, postgres pondrá el nombre de "postgre").

- DB\_PASSWORD: la contraseña que pusiste para postgreSQL

2. Compilar y ejecutar la aplicación

La aplicación utiliza Maven para la gestión de dependencias. Puedes compilar y ejecutar la aplicación utilizando los siguientes comandos en tu terminal: <pre>cd literalura mvn clean install mvn spring-boot:run </pre> Esto compilará la aplicación y la iniciará.

Asimismo, si prefieres ejecutarla desde un IDE como IntelliJ por ejemplo, solo tienes que abrir el proyecto y correr la clase LiteraluraApplication.

3. Interactuar con la aplicación

Una vez que la aplicación esté en ejecución, puedes interactuar con ella a través de la consola. Se te presentará un menú con varias opciones para buscar libros, listar libros y autores registrados, listar autores vivos en un determinado año, listar libros por idioma, ver los 10 libros más descargados y obtener estadísticas. Simplemente sigue las instrucciones en pantalla para interactuar con la aplicación.

## Tecnologías utilizadas

- Java SE17

- Maven

- Spring Boot V3.3.0

- Para el desarrollo del código, se usó el IDE IntelliJ

- Para la base de datos, se usó PostgreSQL

## Estado del proyecto

Finalizado.

## Licencias

MIT License.