Laboratorios 3 y 4

Objetivos:

- ✓ Aplicar la teoría estudiada sobre API's, Microservicios y Domain Driver Design.
- ✓ Diseñar una capa de integración para un dominio funcional.
- ✓ Implementar una capa de integración para un dominio funcional.

Actividades

1- Diseño [LAB03]

Se debe diseñar una capa de integración para un dominio funcional Productos de una tienda en línea. Esta capa de integración tiene que proveer todas las interfaces necesarias para poder realizar operaciones CRUD.

A continuación, se detallan los pasos a seguir y también las consideraciones generales sobre el trabajo solicitado.

1. El diseño debe cumplir las características de un RESTFul API.
2. Las URL's debe contemplar versionado del API.
3. Se deben definir los recursos Productos y Facturas (Misma lógica que pueden seguir para la definición de todos los recursos necesarios para su TP).
4. Se deben especificar las URL's para las operaciones CRUD.
   a. Create
   b. Retrieve
   c. Update
   d. Delete

Método	Acción	URL
GET	Retrieve	/v1/productos
GET	Retrieve	/v1/productos/{id}
POST	Create	/v1/productos
PUT	Update	/v1/productos/{id}
DELETE	Delete	/v1/productos/{id}

2- Implementación [LAB04]

Se debe implementar un microservicio para el dominio funcional Productos y Facturas, el cual fue diseñado en el paso anterior. Esta implementación debe darle cuerpo a todas las URL's definidas. Este microservicio debe tener su propia base de datos.

Pasos a seguir y también las consideraciones generales

2.1. Base de Datos

Construir una base de datos con las tablas necesarias, para proporcionar persistencia.
Ejemplo: Base de Datos relacional, mediante el SGBD PostgreSQL.
Sugerencias:

- ✓ Verificar que se tenga el SGBD instalado, virtualizado o contenerizado para su uso.
- ✓ Contar con alguna herramienta de administración y desarrollo de bases de datos, como DBeaver, pgAdmin, SQL Tabs o similares compatibles con tu sistema operativo.

Material de Ayuda:

En EDUCA – Base de datos puedes encontrar un script básico que puede servirles para construir sus tablas.

2.2.- Proyecto

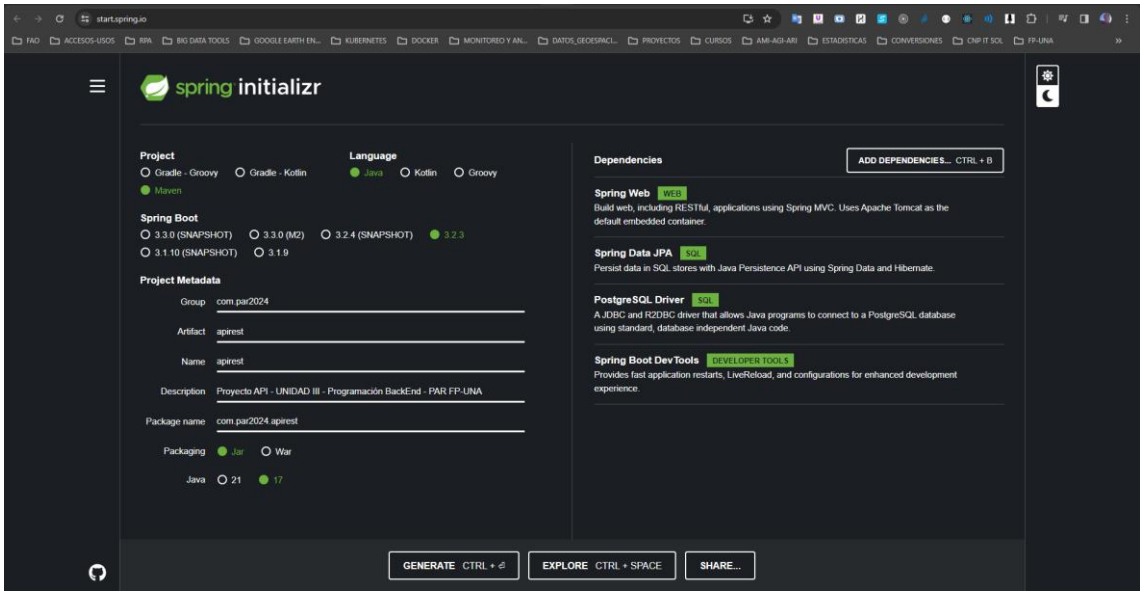
Podemos utilizar Spring inirializr: https://start.spring.io para para generar nuestro proyecto, o con su IDE de preferencia.

Nota:
Spring inirializr *es una herramienta en línea proporcionada por el equipo de Spring para generar rápidamente proyectos de Spring Boot. Permite a los desarrolladores configurar y crear proyectos de Spring Boot personalizados con facilidad, proporcionando una interfaz intuitiva para configurar las dependencias, la versión de Spring Boot y otras configuraciones del proyecto.*

Utilidades:

- ✓ *Se pueden seleccionar las dependencias específicas que necesitan para su proyecto, como Spring Web, Spring Data JPA, Spring Security, etc., y descargar un proyecto de Spring Boot listo para usar.*
- ✓ *Se pueden especificar la versión de Spring Boot, el idioma (Java o Kotlin), el tipo de empaquetado (JAR o WAR), el nombre del paquete y otras propiedades del proyecto.*
- ✓ *Además de las dependencias proporcionadas por Spring, también se pueden agregar dependencias de terceros, como bibliotecas de base de datos, bibliotecas de prueba, etc.*
- ✓ *Se puede personalizar aún más el proyecto descargado según nuestras necesidades específicas, agregando nuestra propia lógica, configuración y componentes.*
- ✓ *No requiere instalación.*
- ✓ *Se descarga el proyecto comprimido.*

Ejemplo:



Sugerencias:

- ✓ Dependencias: Revisar pom.xml
- ✓ Propiedades de la aplicación: Revisar application.properties, para definir DATA SOURCE y JPA / HIBERNATE, por ejemplo. Ya que las mismas son importantes para la conexión a la base de datos.

Consideraciones:

1. Implemente las URL's diseñadas en la actividad anterior.
2. Los paquetes a contemplar: modelos, repositorio, servicios y controladores o en su efecto:
   a. entity: Donde se definen los recursos Producto y Factura
   b. repository: Donde se realiza la implementación del acceso a datos del microservicio.
   c. service: Implementación de la lógica de negocio del microservicio. Esta lógica utiliza el acceso a datos para componer lógica de más alto nivel.
   d. rest: Lógica de exposición del servicio. Aquí es donde decidimos exponer el servicio como REST (podría haber otro paquete por ejemplo soap donde la misma lógica sea expuesta con otro mecanismo de delivery).
3. La mensajería utilizada será Json.

Material de Ayuda:

En EDUCA - Proyecto Base pueden encontrar un proyecto base, generado con Spring inirializr, configurado con las dependencias básicas, conexión a PostgreSQL, algunas interfaces y clases por capas: Modelos, Repositorio, Servicios y Controladores, una estructura para la publicación de microservicios, implementado para un dominio funcional.

Nota: aprovechemos este espacio para recordar que, en el desarrollo de software basada en el patrón de diseño, solemos encontrarnos con las capas de ***Modelos, Repositorio, Servicios y Controladores***, cada una de estas capas tiene funciones específicas y siguen el principio de responsabilidad única y desempeñan un papel específico en el diseño y la arquitectura de la aplicación.

Modelos:

- *En esta capa se definen las clases que representan los datos de la aplicación, también conocidas como entidades o modelos.*
- *Estas clases contienen los atributos y métodos necesarios para representar la estructura y el comportamiento de los datos.*
- *Los modelos suelen ser objetos simples de tipo POJO (Plain Old Java Object) o entidades JPA (Java Persistence API) en aplicaciones Java.*

Repositorio:

- *En esta capa se definen las interfaces o clases encargadas de acceder a la capa de persistencia (base de datos) para realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre los datos.*
- *Los repositorios actúan como una abstracción sobre las operaciones de acceso a datos, proporcionando métodos para interactuar con la base de datos.*
- *Se utilizan para separar la lógica de acceso a datos de la lógica de negocio de la aplicación.*
- *Los repositorios pueden implementarse utilizando tecnologías como Spring Data JPA en aplicaciones Java.*

Servicios:

- *En esta capa se definen las clases que contienen la lógica de negocio de la aplicación.*
- *Los servicios encapsulan la lógica relacionada con las operaciones que se realizan sobre los datos, como la validación, el procesamiento y la transformación.*
- *Se utilizan para mantener un código limpio y modular, separando las preocupaciones de la aplicación.*
- *Los servicios interactúan con los modelos y los repositorios para realizar operaciones sobre los datos.*
- *Pueden contener transacciones y lógica de control de flujo.*

Controladores:

- *En esta capa se definen las clases que actúan como punto de entrada de las solicitudes HTTP en una aplicación web.*
- *Los controladores manejan las solicitudes del cliente, invocan los servicios correspondientes y devuelven una respuesta al cliente.*
- *Se encargan de enrutar las solicitudes a la lógica adecuada de la aplicación y de manejar los datos de entrada y salida.*
- *Los controladores suelen estar asociados a rutas específicas de la aplicación y a métodos HTTP (GET, POST, PUT, DELETE).*
- *En aplicaciones Java, los controladores pueden implementarse utilizando frameworks como Spring MVC o Spring WebFlux.*

2.3.- Pruebas

Utilice herramientas como Postman, Insomnia, RESTClient u otra herramienta similar, de tu preferencia o que quieras explorar, para realizar las pruebas.

Realiza las siguientes pruebas y genera evidencias (capturas de pantalla) para agregar a tu informe:

GET /v1/productos
GET /v1/productos/{id}
POST /v1/productos
PUT /v1/productos/{id}
DELETE /v1/productos/{id}

GET /v1/facturas
GET /v1/facturas/{id}
POST /v1/facturas
PUT /v1/facturas/{id}
DELETE /v1/facturas/{id}

Entregas:

Lab03- Entrega solo de un documento PDF a través de la plataforma EDUCA.

Lab04- Entrega por plataforma EDUCA y al repositorio.
Dentro del repositorio creado en GitHub, con tu usuario y la descripción: **par2025**
Dentro de **laboratorios/Lab04**
Levantar la implementación realizada.
Permitir el acceso a: cnpalacios@pol.una.py.
Los códigos que no pueden ser probado, no serán considerados.

Para Lab03 y Lab04: Puedes usar alguna plantilla de laboratorios anteriores para tu informe de evidencias.