MANUAL DE USUARIO

Elaborado por: Enrique Condori.

Para ejecutar el proyecto primero debes realizar los sgtes paso:

1. Clonar el repositorio

Primero, necesitas clonar el repositorio en tu máquina local. Abre una terminal o consola y ejecuta el siguiente comando:

git clone https://github.com/kikePrograMer007/SprintBootPasantia.git

Esto creará una carpeta llamada "SprintBootPasantia" en tu directorio actual.

2. Abrir el proyecto en tu IDE

2.1 Para STS4:

2.1.1 Abrir la herramienta: Abre STS4.

2.1.2 Importar proyecto:

- Ve a `File > Import`.
- En el menú de importación, selecciona `Existing Maven Projects` y haz clic en `Next`.
- En la ventana de importación, haz clic en `Browse` y navega hasta el directorio donde clonaste el repositorio `SprintBootPasantia`.
- Selecciona la carpeta del proyecto y haz clic en `Finish`.

2.2. Para Eclipse:

2.2.1 Abrir Eclipse: Abre Eclipse.

2.2.2. Importar proyecto:

- Ve a `File > Import`.
- En el menú de importación, selecciona `Maven > Existing Maven Projects` y haz clic en `Next`.
- Haz clic en `Browse` y navega hasta la carpeta donde clonaste el repositorio `SprintBootPasantia`.
- Selecciona la carpeta y haz clic en `Finish`.

3. Esperar a que Maven Descargue las Dependencias

STS4 y Eclipse detectarán automáticamente el archivo pom.xml y comenzarán a descargar las dependencias necesarias. Esto puede tardar unos minutos dependiendo de la velocidad de tu conexión a Internet.

4. Ejecutar el Proyecto

Ejecutar como aplicación Spring Boot:

- Haz clic derecho en la carpeta principal (`SpringBootPasantia`).
- Selecciona `Run As > Spring Boot App`.

El servidor Spring Boot se iniciará y deberías ver los logs en la consola de STS4 o Eclipse.

5. Verificar la Ejecución

Abre un navegador web y ve a http://localhost:8080/api/posts (o el puerto configurado en el proyecto) para verificar que la aplicación esté corriendo correctamente.

6. (en caso de errores) Resolver Problemas Comunes

- **Dependencias no resueltas:** Si algunas dependencias no se descargan correctamente, ve a `Project > Update Maven Project` para forzar la actualización.
- **Cambiar el puerto**: Si **8080** está ocupado, puedes cambiar el puerto en `application.properties` utilizando `server.port=PUERTO`.

EXPLICACIÓN DE LAS APIS CONSUMIDAS Y CONFIGURADAS

Como api publica se usó **MockAPI** , estas APIs simulan el comportamiento de un backend real.

1. API Pública Consumida:

URL Base Posts: `https://66b74e907f7b1c6d8f1b7f24.mockapi.io/api/v1/posts`

URL Base Users: `https://66b74e907f7b1c6d8f1b7f24.mockapi.io/api/v1/users`

2. API Propia Configurada:

Posts: http://localhost:8080/api/posts

Users: http://localhost:8080/api/users

3. Descripción del Desarrollo:

- Nuestra API propia fue configurada como un servicio RESTful utilizando Spring Boot. Esta API permite la gestión completa de los usuarios y comentarios en el sistema a través de operaciones CRUD (Crear, Leer, Actualizar, Eliminar).
- Esta API maneja solicitudes HTTP desde el frontend u otros servicios que requieren acceso a la información de usuarios y comentarios.

4. Ejemplo de Uso:

GET /api/posts:

- **Descripción:** Retorna una lista de posts obtenidos de la API pública.
- **Ejemplo**: http://localhost:8080/api/posts

GET /api/posts/{id}:

- **Descripción:** Retorna los detalles de un post específico, obtenido de la API pública.
- **Ejemplo:** http://localhost:8080/api/posts/1

5. (Opcional) Adicionalmente se desarrolló un ejemplo de POST, PUT y DELETE

POST /api/posts:

- **Descripción:** Permite la creación de un nuevo post en la API pública.
- **Ejemplo:** http://localhost:8080/api/posts
- **Cuerpo de la Solicitud:**

```
{
   "userId": 1,
   "title": "Mi titulo es aqui",
   "body": "body prueba"
}
```

PUT /api/posts/{id}

- **Descripción:** Actualiza los detalles de un post existente.
- **Ejemplo:** http://localhost:8080/api/posts/1
- **Cuerpo de la Solicitud:**

```
{
    "userId": 3,
     "title": "title 101 as pruebas",
    "body": "body 101 asa aass"
}
```

DELETE /api/posts/{id}

- **Descripción:** Elimina un post existente en tu sistema.
- **Ejemplo:** http://localhost:8080/api/posts/1

Característica del proyecto

Entorno de desarrollo: IDE Spring Tool Suite 4

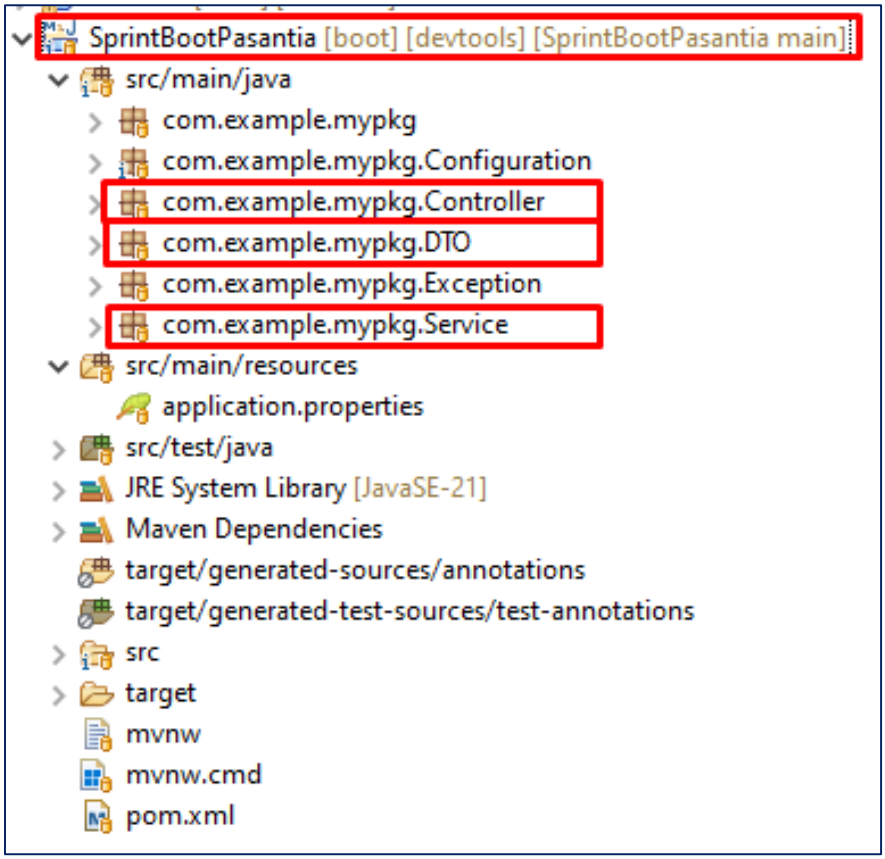
Librería: JAVA JDK 21.

Sprint Boot Version: 3.3.2

Dependencias utilizadas

- Spring Boot Starter Web
- Spring Boot Starter Validation
- Lombok
- Spring Boot DevTools

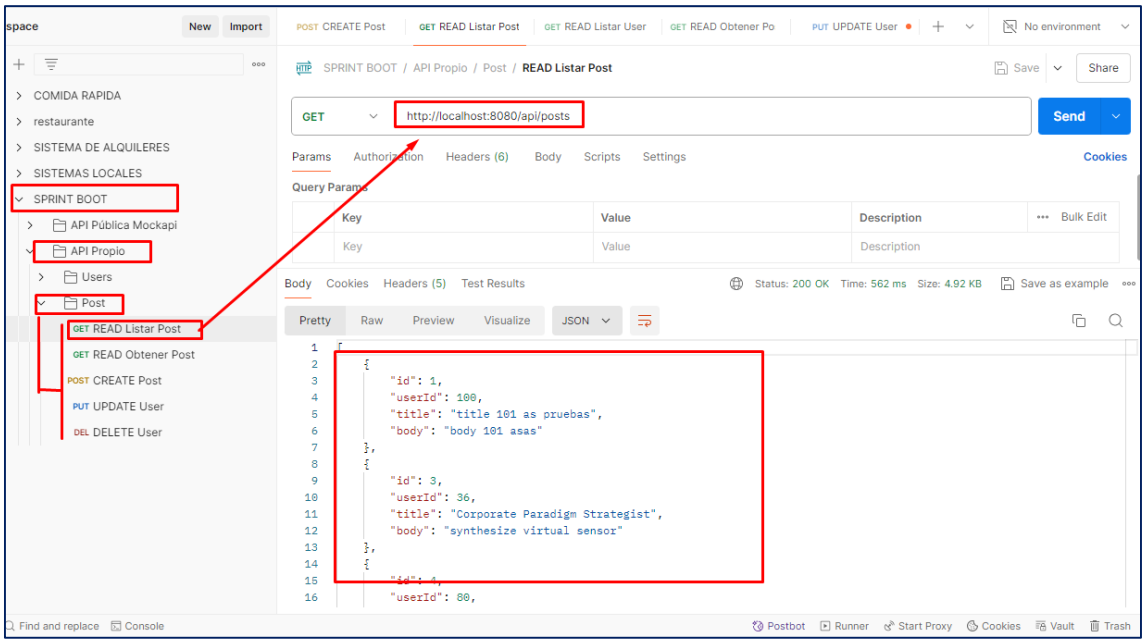
Arquitectura N Capas:

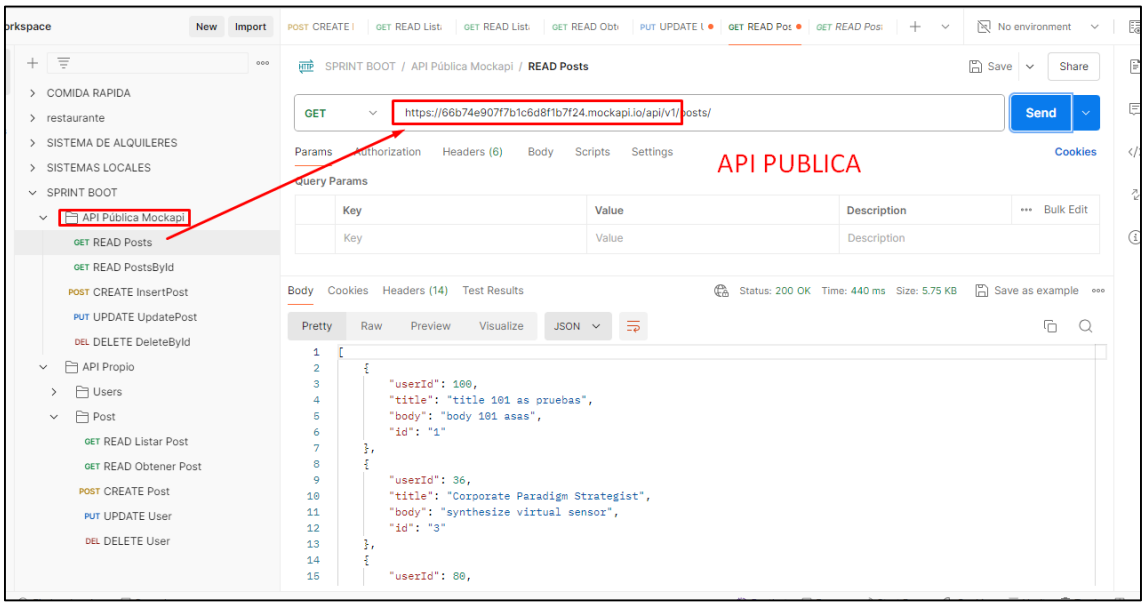


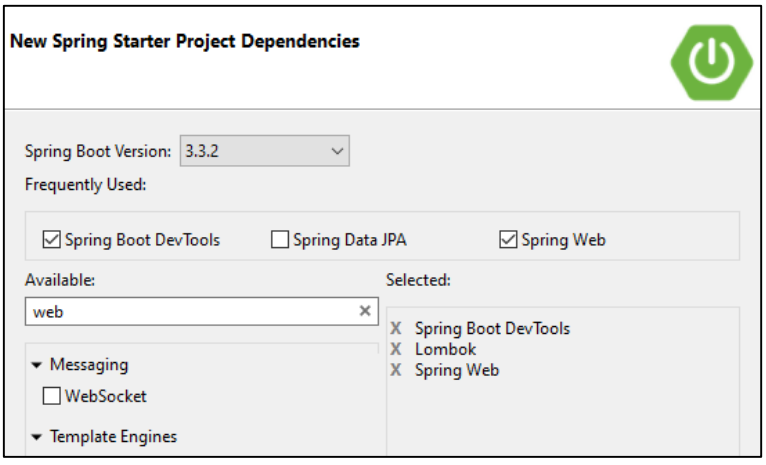
Visión general de cómo los paquetes interactúan entre sí en la arquitectura de la aplicación:

1. **Controller**: Recibe las solicitudes HTTP y usa los servicios de la capa Service para procesar las solicitudes. Los controladores utilizan objetos del paquete DTO para enviar y recibir datos.
2. **Service**: Contiene la lógica de negocio y las reglas de aplicación. Esta capa coordina las operaciones entre el controlador y la capa de persistencia o en este caso el api externo.
3. **DTO**: Los objetos DTO se utilizan para transportar datos entre la capa de presentación (Controller) y la capa de servicio (Service). Los DTOs a menudo incluyen anotaciones de validación para asegurar que los datos sean correctos antes de ser procesados.
4. **Exception**: Maneja las excepciones a nivel global y proporciona respuestas estandarizadas cuando ocurren errores. Los controladores y servicios pueden lanzar excepciones que son capturadas y gestionadas por el manejador de excepciones global.
5. **Configuration**: Proporciona configuraciones necesarias para la operación del sistema, como la configuración de seguridad, conexión a bases de datos, etc.

ANEXOS:


COMIDA RAPIDA
restaurante
SISTEMA DE ALQUILERES
SISTEMAS LOCALES
SPRINT BOOT
API Pública Mockapi
API Propio
Users
Post
GET READ Listar Post
GET READ Obtener Post
POST CREATE Post
PUT UPDATE User
DEL DELETE User
SPRINT BOOT / API Propio / Post / READ Listar Post
GET
http://localhost:8080/api/posts
Send
Params
Authorization
Headers (6)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 562 ms
Size: 4.92 KB
Save as example
Pretty
Raw
Preview
Visualize
JSON
[
{
"id": 1,
"userId": 100,
"title": "title 101 as pruebas",
"body": "body 101 asas"
},
{
"id": 3,
"userId": 36,
"title": "Corporate Paradigm Strategist",
"body": "synthesize virtual sensor"
},
{
"id": 4,
"userId": 80,


COMIDA RAPIDA
restaurante
SISTEMA DE ALQUILERES
SISTEMAS LOCALES
SPRINT BOOT
API Pública Mockapi
GET READ Posts
GET READ PostsById
POST CREATE InsertPost
PUT UPDATE UpdatePost
DEL DELETE DeleteById
API Propio
Users
Post
GET READ Listar Post
GET READ Obtener Post
POST CREATE Post
PUT UPDATE User
DEL DELETE User
SPRINT BOOT / API Pública Mockapi / READ Posts
GET
https://66b74e907f7b1c6d8f1b7f24.mockapi.io/api/v1/posts/
Send
API PUBLICA
Params
Authorization
Headers (6)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Body
Cookies
Headers (14)
Test Results
Status: 200 OK
Time: 440 ms
Size: 5.75 KB
Save as example
Pretty
Raw
Preview
Visualize
JSON
[
{
"userId": 100,
"title": "title 101 as pruebas",
"body": "body 101 asas",
"id": "1"
},
{
"userId": 36,
"title": "Corporate Paradigm Strategist",
"body": "synthesize virtual sensor",
"id": "3"
},
{
"userId": 80,


New Spring Starter Project Dependencies
Spring Boot Version: 3.3.2
Frequently Used:
Spring Boot DevTools
Spring Data JPA
Spring Web
Available:
web
Messaging
WebSocket
Template Engines
Selected:
X Spring Boot DevTools
X Lombok
X Spring Web