SpringBoot:

Group ID: Es como el apellido de una familia de proyectos en programación. Ayuda a organizarlos y distinguirlos dentro de un sistema. (Organizar proyectos)

Administrador de dependencias: Es como un asistente que se encarga de traer las herramientas necesarias para que tu proyecto funcione, evitando que tengas que buscarlas manualmente.

Inyección de dependencias: Es como si tuvieras una tienda y en lugar de ir cada día a buscar los productos, alguien los coloca automáticamente en tus estantes. Así, tu tienda funciona sin que tú te preocupes por dónde conseguir cada cosa.

Jerarquía de carpetas: Es la organización estructurada de directorios dentro de un sistema.

Siguiendo el modelo de árbol:

- **Raíz (Root)**: La carpeta principal que contiene todo.
- **Subcarpetas**: Dentro de la raíz, hay divisiones lógicas (ej. Documentos, Imágenes, Código).
- **Más niveles**: Dentro de cada subcarpeta, se pueden anidar más carpetas según necesidad.
- **Ejemplo:**

Nivel de elementos directamente

Se refiere a la ubicación o profundidad de archivos dentro de una estructura. Un archivo puede estar:

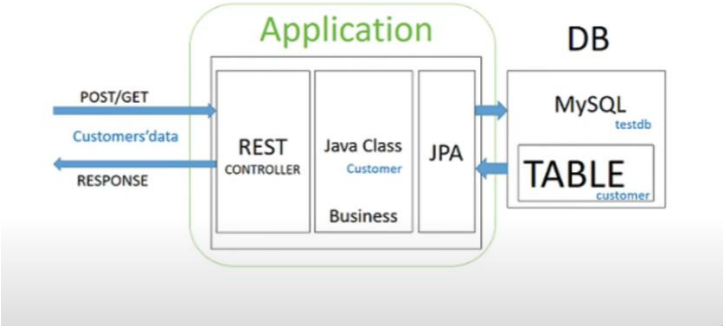
- **Nivel raíz**: Directamente accesible desde la primera capa del sistema.

- **Nivel intermedio**: Dentro de una subcarpeta, accesible con más navegación.
- **Nivel profundo**: Enterrado en múltiples niveles de carpetas, requiriendo caminos específicos para acceso

Controlador: Es una clase que maneja las solicitudes http entrantes y define como deben ser procesadas para generar una respuesta.

Spring es un proceso de backen, hablamos de servidores: un servidor es un elemento en la red que recibe solicitudes para dar respuestas, para eso usamos controladores para manejar los requerimientos que entran a nuestro servidor.

Ejemplo:



Modelo Vista Controlador (MVC): Es un patrón de diseño que contiene tres componentes principales:

- Modelo (Model): Gestiona los datos de la aplicación y la lógica de negocio.
- Vista (View): Encargada de presentar los datos al usuario.
- Controlador (Controller): Procesa las solicitudes del usuario, interactúa con el modelo y selección la vista adecuada para renderizar.

El objetivo de MVC es facilitar el mantenimiento, la escalabilidad y la reutilización del código.

Arquitectura de Spring MVC:

- DispatcherServlet: Es un punto de entrada para todas las solicitudes en una aplicación de "spring", todas las solicitudes que se van hacer a nuestro servidor primero pasan por el "DispatcherServlet", se encarga de enrutar las solicitudes a los controladores adecuados.
- Controladores: Manejan las solicitudes, procesan los datos gracias a los servicios o repositorios y luego selecciona la vista adecuada para devolver el dato al cliente.

- Modelo: Contiene datos y lógica de negocios esto incluye los “beens” y los “dao”…
- Beans: Son clases que representan datos. Ejemplo, si tienes una aplicación para gestionar libros, un “BookBean” podrían tener atributos como titulo, autor y precio. Los Benas suelen tener métodos get y set para acceder y modificar sus valores.
- Dao: Son clases que se encargan de interactuar con la base de datos. En lugar de que cada parte del código haga consultas a la base de datos directamente, el DAO maneja todo eso.

- Vista: Es la parte de la interfaz de usuario como: jsp, thymeleaf
- View Resolve: Determinan que archivo de vista debe renderizarse con base en el nombre devuelto por el controlador.
- Handler Mapping: Mapean las solicitudes a los controladores correctos.

Flujo de Trabajo de MVC:

- El cliente envía una solicitud
- DispatcherServlet recibe la solicitud
- El controlador procesa la solicitud
- El controlador selecciona una vista
- El view resolver localiza la vista
- Se genera una respuesta

Anotaciones de un Controlador:

RequesMapping: Es el método con el que el controlador va a manejar las solicitudes, recibe las solicitudes de http y lo hace capturando los datos desde una url, que seria el cuerpo de esa solicitud y luego le devuelve una respuesta al elemento que hace la solicitud.

Y esta respuesta que manda ese controlador puede ser un archivo de tipo:

- .json
- .xml
- .html

Método Handler (Métodos http) : Un Handler es una función o clase encargada de manejar solicitudes HTTP en un servidor. Cada solicitud (request) llega con un método, que define la acción a realizar.

Método:

- Get: Se usa para solicitar información sin modificar datos. Ejemplo: obtener una lista de usuarios desde una API.

- Post: Envía datos al servidor para crear un nuevo recurso. Ejemplo: registrar un usuario en la base de datos.
- Delete: Borra un recurso existente. Ejemplo: eliminar un producto de un catálogo.
- Put: Actualiza un recurso existente, reemplazándolo completamente. Ejemplo: modificar un perfil de usuario.

Requests (solicitudes http): Una **request** es la petición que un cliente (como un navegador o una aplicación) envía a un servidor para obtener o modificar información. Cada request tiene varios elementos clave:

Componentes de una request:

- **URL**: La dirección del recurso al que se accede (https://api.ejemplo.com/usuarios).
- **Método HTTP**: Define la acción que se quiere realizar (GET, POST, DELETE, PUT).
- **Headers**: Contienen información sobre la request, como el tipo de contenido (Content-Type: application/json).
- **Body**: Solo en métodos como POST o PUT, contiene los datos que se envían al servidor.

Ejemplo de una request POST en formato JSON:

Para enviar una solicitud al servidor para crear un usuario:

```
POST /usuarios HTTP/1.1
Host: api.ejemplo.com
Content-Type: application/json

{
   "nombre": "Carlos",
   "email": "carlos@email.com"
}
```

Esto indica al servidor que debe crear un usuario con los datos proporcionados que en el ejemplo son el nombre y el correo.

Mapeo:

Inyección de dependencias:

Thymeleaf Motor de Plantillas Dinamicas en SpringBoot:

Rest Controller: Es una anotación que se utiliza para definir un controlador que maneja solicitudes http y que produce respuestas directamente en un formateo: .JSON, .XML ..

También se utilizan para mapear Solicitudes http, vamos a poder definir solicitudes entrantes están basadas en criterios como los métodos get, pot, put, deleate.

Método HandlerRest:

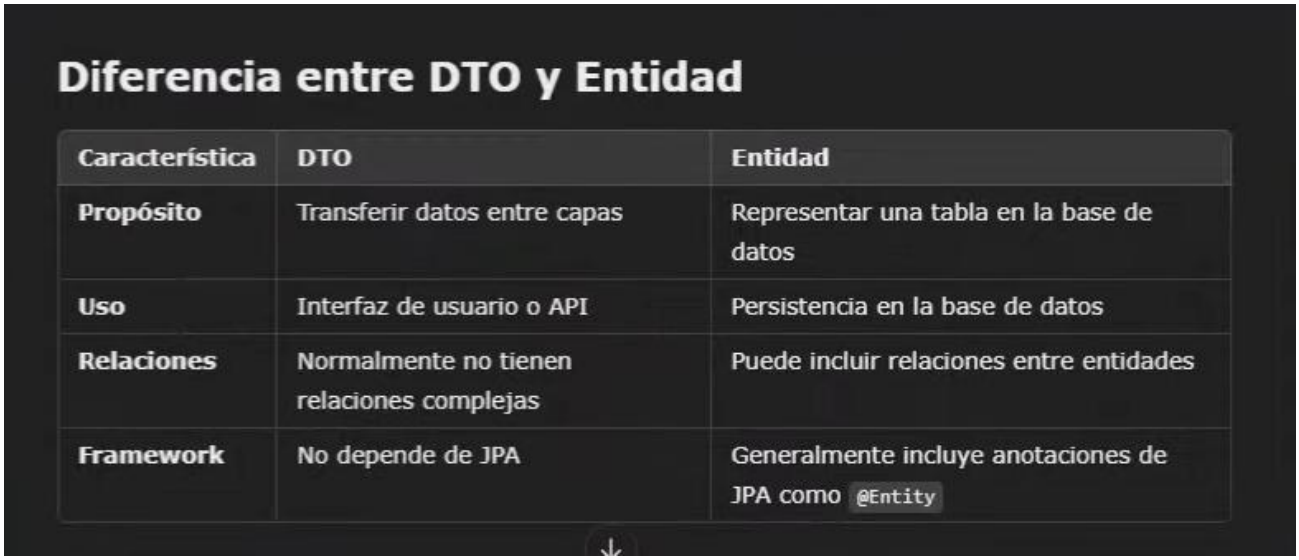
API:

API_REST:

Objeto Model en Spring: Es un parámetro que se usa en los métodos de los controladores, spring lo que hace es que inyecta automáticamente al objeto model cuando se define en la declaración de un método.

Clase (DTO) Data Transfer Object: Es una clase diseñada para transferir datos entre diferentes capas de una aplicación, por ejemplo: La capa de controlador y la de servicio, esta clase DTO contiene una lógica de negocio y su propósito es actuar como un contenedor para nuestros datos, para que se usan:

- Separación de Preocupaciones: Es una funcionalidad, característica o una especie de requisito que se debe implementar en una aplicación, por ejemplo: El manejo de la lógica de un negocio / proyecto.. Eso es una preocupación.
- Manejo de Persistencia de datos:
- Seguridad:
- Diferencia entre DTO Y Entidad



Diferencia entre DTO y Entidad

Característica	DTO	Entidad
Propósito	Transferir datos entre capas	Representar una tabla en la base de datos
Uso	Interfaz de usuario o API	Persistencia en la base de datos
Relaciones	Normalmente no tienen relaciones complejas	Puede incluir relaciones entre entidades
Framework	No depende de JPA	Generalmente incluye anotaciones de JPA como `@Entity`