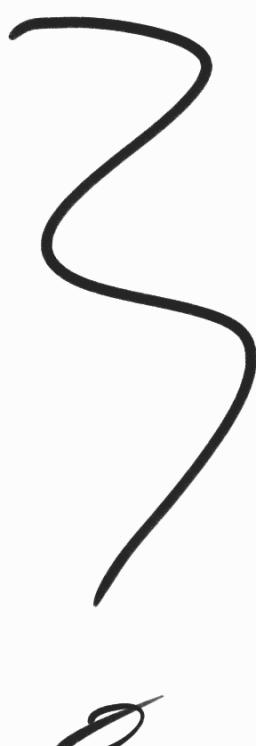
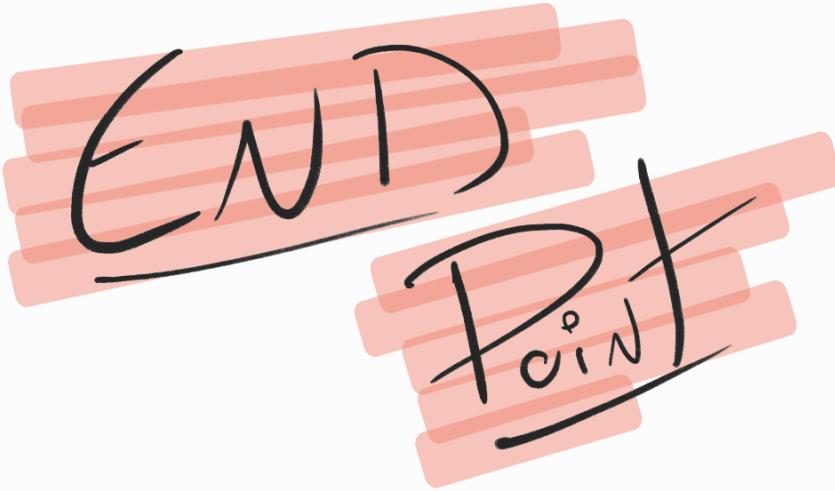


Projeto: Mini-Blog criando usuários, autenticação, postagens, comentários

estruturação do projeto → Tabelas: - Tag  
- Post  
- User  
- Comment

Cada tabela contendo Id único,  
incrementado automaticamente pelo JPA





Endpoint é um URL que permite o acesso a um serviço específico pelo aplicação cliente.

Configuração a porta 8080 no arquivo application.properties

Packages Necessários:

- Models
- ENUMS
- Controllens
- Service

# Models

## Classes de Entidade:

- User

- Tag

- Post

- Comment



Utilizando as anotações **@Entity** e  
**@Table** para definir as entidades e suas  
respectivas tabelas e os relacionamentos  
Ex.: **Many-to-one** entre Post e User.

# Controllers

- UserController
- TagController
- PostController
- CommentController



Utilizando @RestController e

@RequestMapping

Mapear classes  
como controllers  
Rest.

Definir os caminhos  
dos End-Points

Ex.: No UserController, defino o caminho "User"  
e crie os métodos para as operações

CRUD

# Services

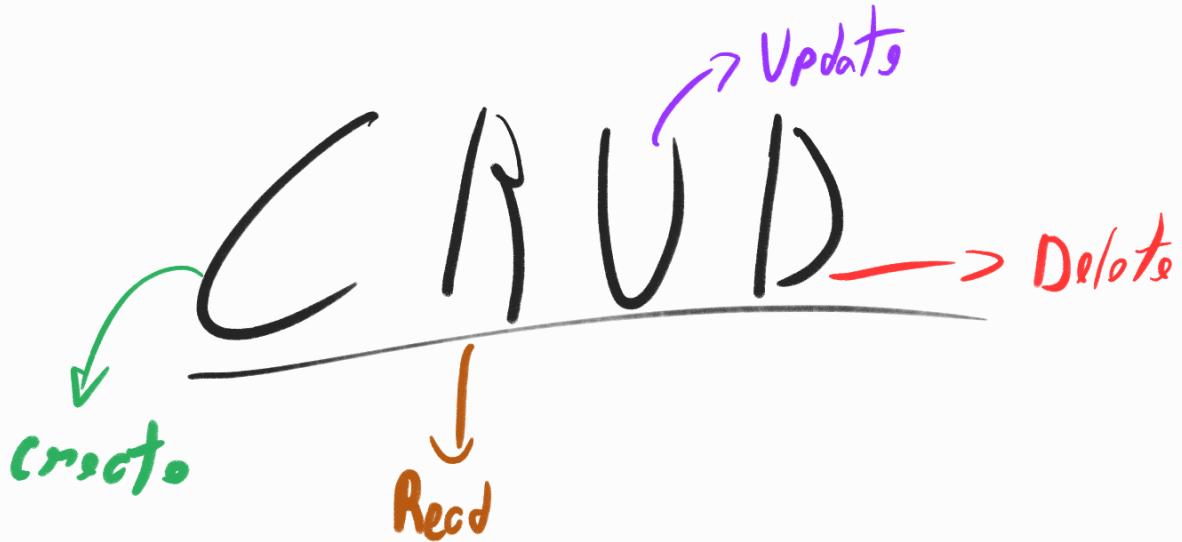
- UserService
- TagService
- PostService
- CommentService

Interfaces

→ Definindo os contratos para as operações de entidades.



Implemente-os de acordo com o padrão (IMP)  
utilizando as anotações `@Service` e `@Autowired` para injetar as dependências dos repositórios.



Usa **Controller** → método **Save** para criar novo usuário

utilizando a anotação **@PostMapping** para mapear o endpoint

"**/Save**" e a anotação **@RequestBody** para receber os dados do usuário na requisição.

Condição: Verificação da existência de Banco de Dados.

Caso Exista, retornamos um erro. Caso contrário, criamos um novo usuário e salvamos no Banco de Dados do Usuário.



# CRUD

Para leituras, criamos métodos getAll e getBy<sub>Id</sub>

@GetMapping ↑ mapear os endpoints

@PathVariable ↗

Recabem os parâmetros do requisício

GetAll → retorna uma lista de todos os Usuários do Banco de Dados

GetBy<sub>Id</sub> → retorna um usuário específico baseado no <sub>Id</sub>



# CRUD

Ponto Atualizações, Criar ou o método Update

@PostMapping → recebemos o ID do usuário na requisição

Condigo: Verificamos se o usuário existe. Caso exista, realizamos a Atualização.



# CRUD

Para exclusão, usamos o método Delete

@DeleteMapping → Recebe-se o ID do usuário.

Condição: Verificamos a existência da usuário. Caso exista, removemos do Banco de Dados.

