

# Tópicos Avançados de Programação

Genilson Medeiros

Professor do curso de Sistemas de Informação  
Mestre em Ciência e Tecnologia em Saúde  
Engenheiro de Software





# Docker



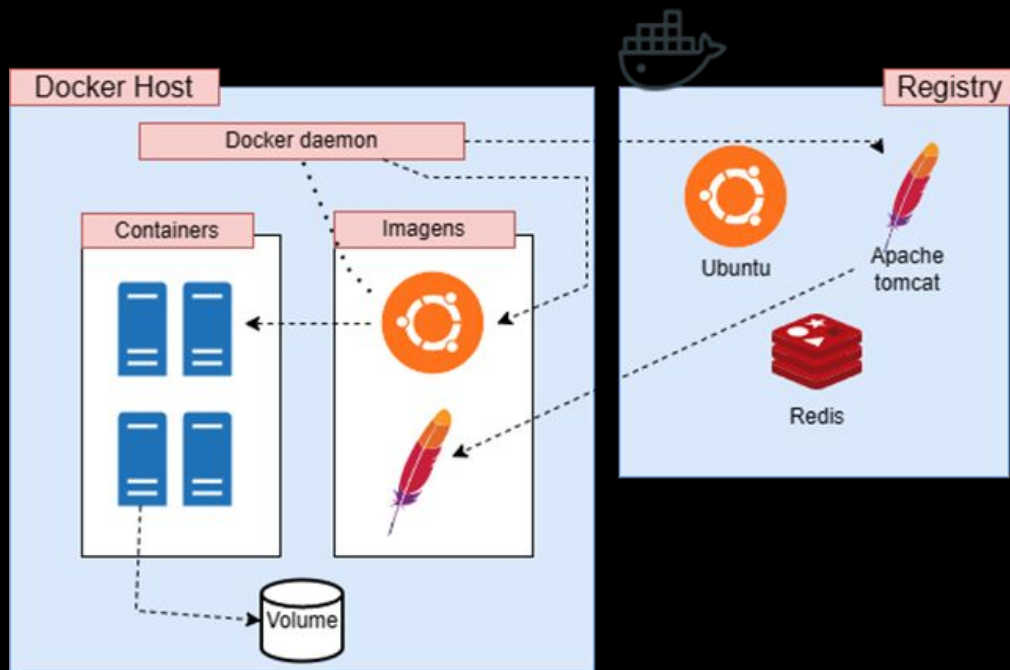
O Docker **simplifica** a complexidade na preparação de ambientes de desenvolvimento, teste e até produção. Ele elimina a necessidade de **instalar várias bibliotecas** e aplicativos, como sistemas de gerenciamento de banco de dados (SGBD), servidores web, gerenciadores de pacotes, entre outros, evitando a famosa frase: **“funciona na minha máquina”**.

# Docker

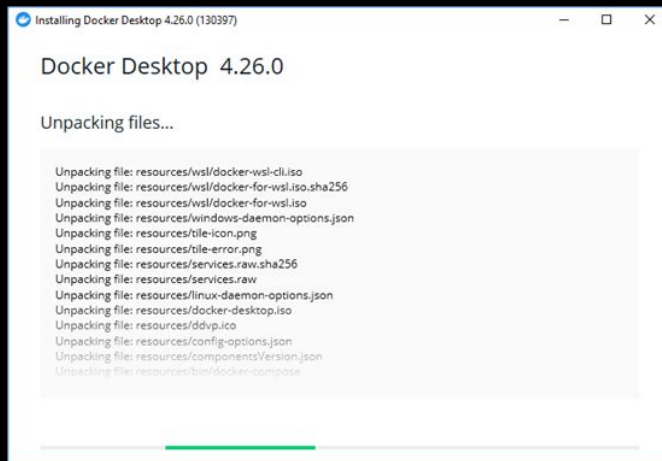
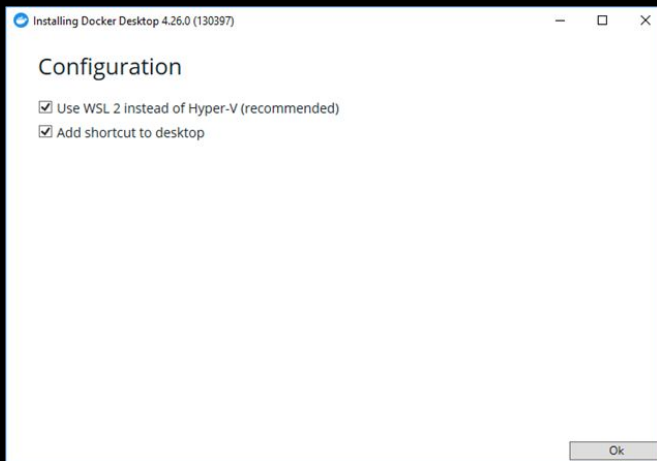


Lembre-se, Docker não é uma máquina virtual (VM). Docker é uma plataforma de virtualização de **containers**. É uma maneira de empacotar aplicativos e todas as suas **dependências** em um pacote portátil chamado **container**.

# Docker

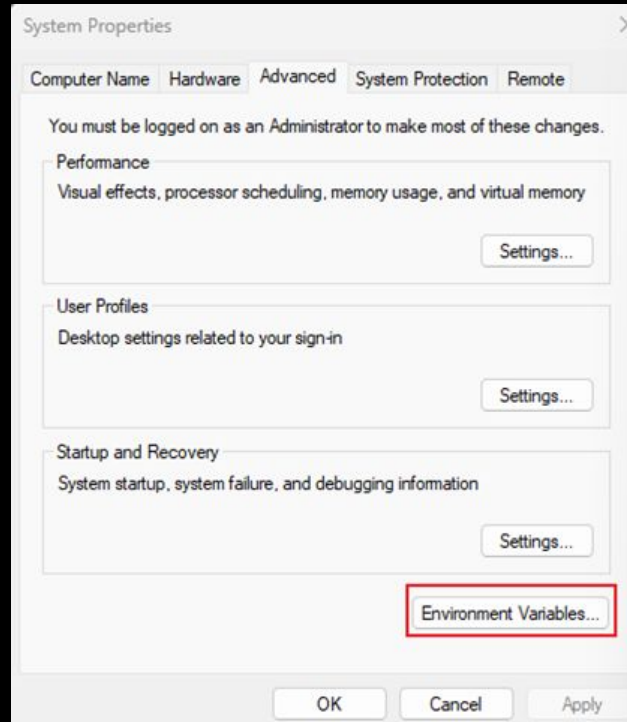


# Instalação

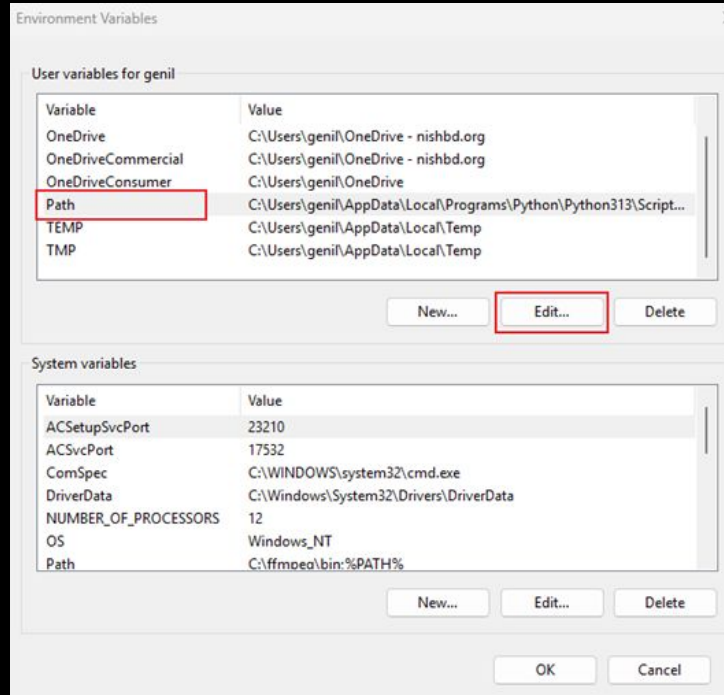


Mesmo após instalar o Docker Desktop no Windows, você pode utilizar apenas o CLI (linha de comando) para realizar todas as operações disponíveis. Também é possível usar o Docker via interface gráfica, acessível diretamente pelo Docker Desktop.

# Variáveis de ambiente

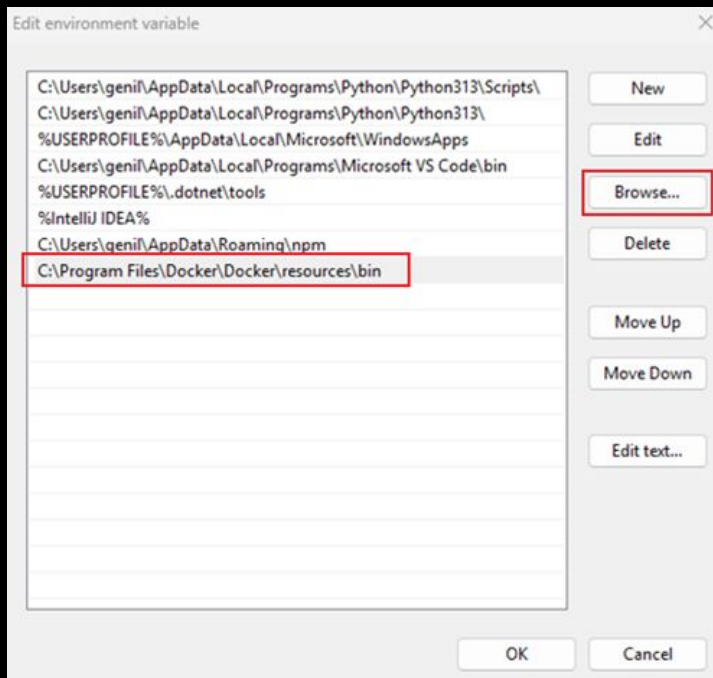


# Variáveis de ambiente





# Variáveis de ambiente



- ```
sudo apt update
```

- ```
sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

- ```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

- ```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
```

- ```
apt-cache policy docker-ce
```

- ```
sudo apt install docker-ce
```

- ```
sudo systemctl status docker
```

# Instalação no linux

Se tudo estiver ok, então você irá visualizar algo parecido com a imagem abaixo



```
genilson@genilson-Virtual-Machine: ~  
● docker.service - Docker Application Container Engine  
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset:   
   Active: active (running) since Wed 2023-12-06 14:51:21 -03; 7s ago  
TriggeredBy: ● docker.socket  
   Docs: https://docs.docker.com  
  Main PID: 5844 (dockerd)  
    Tasks: 12  
  Memory: 30.5M  
    CPU: 277ms  
   CGroup: /system.slice/docker.service  
           └─5844 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/contai
```

# API com Spring Boot Web



```
docker run -d
  --name mysql_db
  -v mysql-v
  -p 3306:3306
  -e MYSQL_ROOT_PASSWORD=root
  mysql:latest
```

# DBeaver



## DBeaver Community

Free Universal Database Tool



44,824



Follow @dbeaver\_news

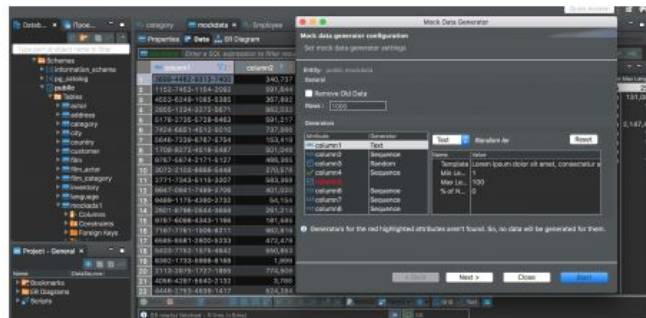
search here ...

Go

[Home](#)[About](#)[Download](#)[Documentation](#)[News](#)[Support](#)[DBeaver PRO](#)[CloudBeaver](#)[DBeaver Merch](#)[Join our team](#)

## Universal Database Tool

DBeaver Community is a free cross-platform database tool for developers, database administrators, analysts, and everyone working with data. It supports all popular SQL databases like MySQL, MariaDB, PostgreSQL, SQLite, Apache Family, and more.



# DBeaver



Connection "localhost 2" configuration

Connection settings  
MySQL connection settings

MySQL

Connection settings  
Initialization  
Shell Commands  
Client identification  
Transactions  
General  
Metadata  
Errors and timeouts  
Data Transfer  
Data Editor  
SQL Editor

Main Driver properties SSH SSL + Network configurations...

Server

Connect by: ☐ Host ☒ URL

URL: jdbc:mysql://localhost:3306/

Server Host: localhost Port: 3306

Database:

Authentication (Database Native)

Username: root

Password:  Save password

Advanced

Server Time Zone: Auto-detect

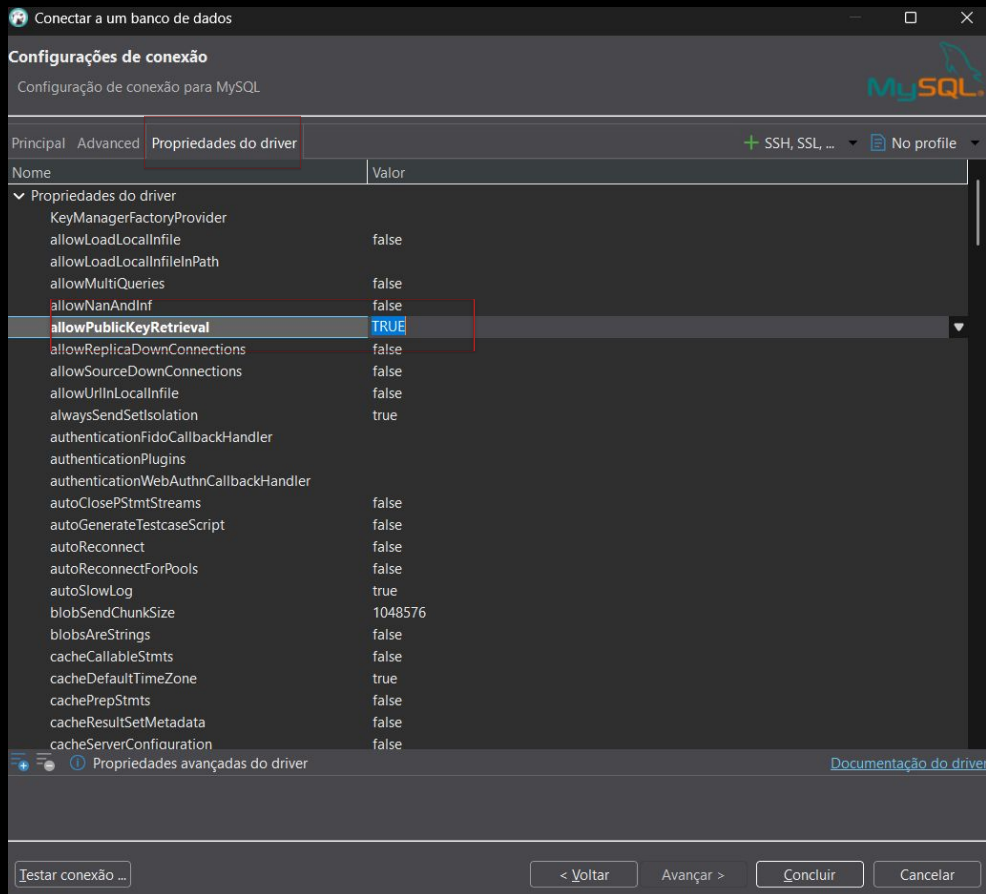
Local Client: MySQL Binaries

[You can use variables in connection parameters.](#)

Driver name: MySQL Driver Settings Driver license

Test Connection ... OK Cancel

# DBeaver



# Projeto



## Requisitos (primeira etapa):

- Criar conta
  - Nome do perfil
  - Email (login)
  - Senha
  - Foto
  - Mensagem de apresentação
- CRUD do usuário
  - O delete deve ser exclusão lógica
  - Não é possível editar o email
- Fazer upload da Foto de perfil
  - Ao fazer o upload da imagem você deve ter o cuidado para não permitir substituição da imagem com outro usuário



# Criando o projeto



## Project

- ☐ Gradle - Groovy ☐ Gradle - Kotlin ☒ Java ☐ Kotlin ☐ Groovy
- ☒ Maven

## Spring Boot

- ☐ 4.0.0 (SNAPSHOT) ☐ 4.0.0 (M1) ☐ 3.5.5 (SNAPSHOT) ☒ 3.5.4
- ☐ 3.4.9 (SNAPSHOT) ☐ 3.4.8

## Project Metadata

- Group
- Artifact
- Name
- Description
- Package name
- Packaging ☒ Jar ☐ War
- Java ☐ 24 ☐ 21 ☒ 17

## Dependencies

ADD DEPENDENCIES... CTRL + B

### Lombok DEVELOPER TOOLS

Java annotation library which helps to reduce boilerplate code.

### Spring Web WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

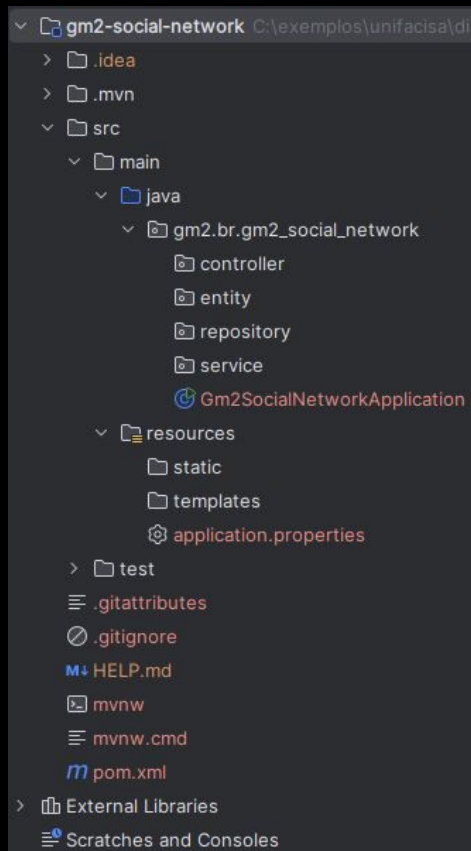
### Spring Data JPA SQL

Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

### MySQL Driver SQL

MySQL JDBC driver.

# Estrutura de pastas (inicial)



# Entidade user



```
1 package gm2.br.gm2_social_network.entity;
2
3 import gm2.br.gm2_social_network.utils.Constants;
4 import jakarta.persistence.*;
5
6 @Entity 2 usages new *
7 @Table(name = "user")
8 public class User {
9     @Id no usages
10     @GeneratedValue(strategy = GenerationType.IDENTITY)
11     private Long id;
12
13     @Column(length = Constants.GENERAL_MAX_LENGTH, nullable = false) no usages
14     private String name;
15
16     @Column(length = Constants.GENERAL_MAX_LENGTH, nullable = false, unique = true) no usages
17     private String email;
18
19     @Column(length = Constants.PHOTO_PATH, nullable = true) no usages
20     private String photo;
21
22     @Column(length = Constants.MESSAGE_LENGTH, nullable = true) no usages
23     private String message;
24 }
```

# Utils



```
1 package gm2.br.gm2_social_network.utils;
2
3 public final class Constants { 5 usages new *
4     public static final int GENERAL_MAX_LENGTH = 100; 2 usages
5     public static final int MESSAGE_LENGTH = 200; 1 usage
6     public static final int PHOTO_PATH = 200; 1 usage
7 }
```

# Repositório



```
1 package gm2.br.gm2_social_network.repository;
2
3 import gm2.br.gm2_social_network.entity.User;
4 import org.springframework.data.jpa.repository.JpaRepository;
5 import org.springframework.stereotype.Repository;
6
7 @Repository no usages new *
8 public interface UserRepository extends JpaRepository<User, Long> {
9 }
```

# Repositório



```
1  spring:
2    datasource:
3      url: jdbc:mysql://localhost/gm2_db?createDatabaseIfNotExist=true
4      username: root
5      password: root
6      driver-class-name: com.mysql.cj.jdbc.Driver
7    jpa:
8      properties:
9        hibernate:
10          dialect: org.hibernate.dialect.MySQL8Dialect
11          show_sql: true
12          format_sql: true
13      hibernate:
14        ddl-auto: update
15
```

Project structure (Left Panel):

- gm2-social-network C:\exemplo
  - .idea
  - .mvn
  - src
    - main
      - java
        - gm2.br.gm2\_social\_n
          - controller
          - entity
            - User
          - repository
          - service
          - utils
          - Gm2SocialNetwo
    - resources
      - static
      - templates
      - application.yml

# Rode o seu projeto



```
Run  Gm2SocialNetworkApplication x
[Icons: Refresh, Stop, Run, Attach, Menu]

↑ Autocommit mode: undefined/unknown
↓ Isolation level: undefined/unknown
⇌ Minimum pool size: undefined/unknown
⇌ Maximum pool size: undefined/unknown
⇌ 2025-08-16T14:09:33.889-03:00 INFO 22488 --- [main] o.h.e.t.
🖨 Hibernate:
🗑   create table user (
      id bigint not null auto_increment,
      email varchar(100) not null,
      message varchar(200),
      name varchar(100) not null,
      photo varchar(200),
      primary key (id)
    ) engine=InnoDB
```

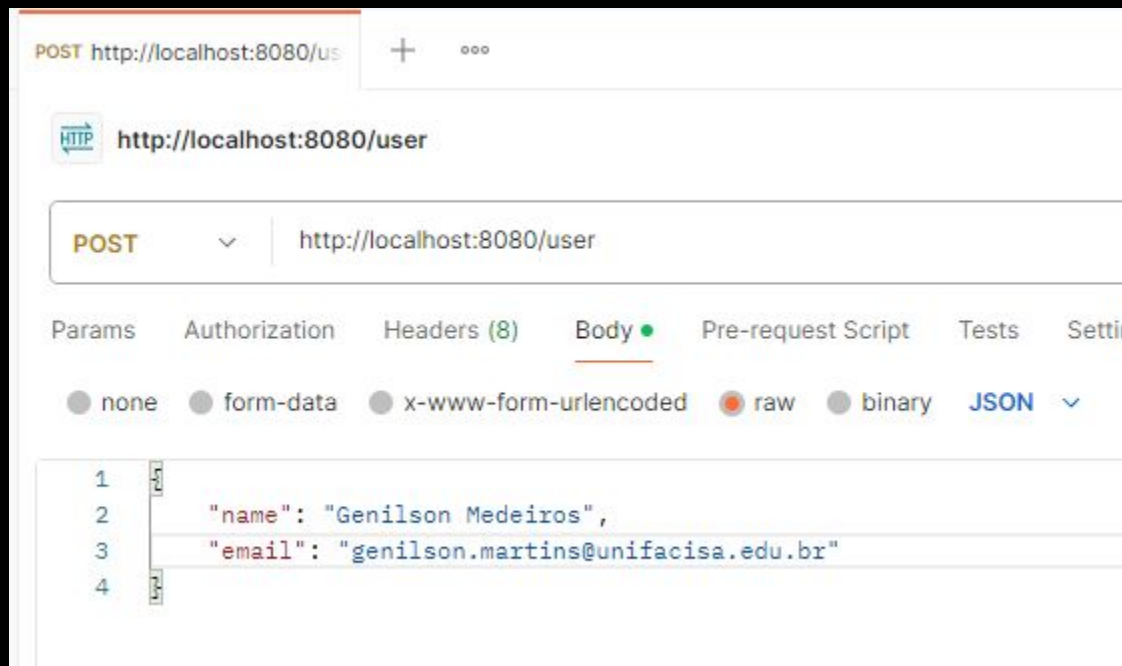
# Controller



```
12 @RestController no usages new *
13 @RequestMapping("/user")
14 public class UserController {
15
16     private final UserRepository userRepository; 2 usages
17
18     private UserController(UserRepository userRepository) { no usages new *
19         this.userRepository = userRepository;
20     }
21
22     @PostMapping() no usages new *
23     public ResponseEntity<User> register(@RequestBody User user) {
24         var userSaved = userRepository.save(user);
25         return new ResponseEntity<>(userSaved, HttpStatus.CREATED);
26     }
27 }
```



# Postman



# Erro de principiante



```
13 @RequestMapping("/user")
14 public class UserController {
15
16     private final UserRepository userRepository;
17
18     private UserController(UserRepository userRepository) {
19         this.userRepository = userRepository;
20     }
21
22     @PostMapping()
23     public ResponseEntity<User> register(@RequestBody User user) {
24         var userSaved = userRepository.save(user);
25         return new ResponseEntity<>(userSaved, HttpStatus.CREATED);
26     }
27 }
28 }
```

Threads & Variables Console

Evaluate expression (Enter) or add a watch (Ctrl+Shift+Enter)

- > this = (UserController@12087)
- ▼ user = (User@12088)
  - id = null
  - name = null
  - email = null
  - photo = null
  - message = null
- > userRepository = ({Proxy113@12089} "org.springframework.data.jpa.repository.support.SimpleJpaRepository@43689de1")

# Anotação Data do Lombok




A anotação `@Data` do Lombok em Java é uma maneira conveniente de reduzir código boilerplate, gerando automaticamente métodos como `getters`, `setters`, `equals()`, `hashCode()` e `toString()` para classes. Ela é um atalho que combina as funcionalidades das anotações `@Getter`, `@Setter`, `@EqualsAndHashCode` e `@ToString`

Na programação de computadores, código boilerplate, ou simplesmente boilerplate, são seções de código que são repetidas em vários lugares com pouca ou nenhuma variação.

# Anotação Data do Lombok



```
7  @Data 5 usages new *
8  @Entity
9  @Table(name = "user")
10 public class User {
11     @Id
12      @GeneratedValue(strategy = GenerationType.IDENTITY)
13     private Long id;
14
15     @Column(length = Constants.GENERAL_MAX_LENGTH, nullable = false)
16     private String name;
17
18     @Column(length = Constants.GENERAL_MAX_LENGTH, nullable = false, unique = true)
19     private String email;
20
21     @Column(length = Constants.PHOTO_PATH, nullable = true)
22     private String photo;
23
24     @Column(length = Constants.MESSAGE_LENGTH, nullable = true)
25     private String message;
26 }
```

# Registro de usuário



POST http://localhost:8080/user

http://localhost:8080/user

POST http://localhost:8080/user

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary JSON

```
1 {
2   "name": "Genilson Medeiros",
3   "email": "genilson.martins@unifacisa.edu.br"
4 }
```

Body Cookies Headers (5) Test Results Status:

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 1,
3   "name": "Genilson Medeiros",
4   "email": "genilson.martins@unifacisa.edu.br",
5   "photo": null,
6   "message": null
7 }
```

*Crie um Service para user*

*Crie um DTO (dica, use o Builder do lombok)*

*Finalize o CRUD*

*Não trate o erro ao tentar  
cadastrar usuários com o mesmo  
email*

*Agora é com  
você!*



# Boa prática para tratar erros?



# Classe genérica para retorno de erros



```
9 public class ResponseDTO { 3 usages new *
10     @Getter
11     private List<String> messages;
12
13     @Getter
14     private HttpStatus status;
15
16     @Getter
17     private int code;
18
19 @ public ResponseDTO(List<String> message, HttpStatus status) { no usages new *
20     this.messages = message;
21     this.status = status;
22     this.code = status.value();
23 }
24
25 @ public ResponseDTO(String message, HttpStatus status) { 1 usage new *
26     this.messages = Arrays.asList(message);
27     this.status = status;
28     this.code = status.value();
29 }
```



# Custom exception



```

1  package gm2.br.gm2_social_network.exception;
2
3  public class DuplicateEmailException extends RuntimeException {
4      public DuplicateEmailException(String msg) { super(msg); }
5  }
6

```

Project Explorer (Left):

- gm2-social-network C:\exemplos\unifacisa\dispo
  - .idea
  - .mvn
  - src
    - main
      - java
        - gm2.br.gm2\_social\_network
          - controller
          - dto
          - entity
          - exception
            - DuplicateEmailException**
          - repository
          - service
          - utils
        - Gm2SocialNetworkApplication

# @RestControllerAdvice (Aviso, recomendação ou opinião)



```
@RestControllerAdvice no usages new *
public class ApplicationAdviceController {

    ⚡ Rename usages
    @ExceptionHandler(DuplicateEmailException.class) new *
    @ResponseStatus(HttpStatus.BAD_REQUEST)
    public ResponseDTO handleDuplicateEmailException(DuplicateEmailException ex){
        String messageError = ex.getMessage();
        return new ResponseDTO(messageError, HttpStatus.BAD_REQUEST);
    }
}
```

# Recursos do Spring Data JPA



No Spring Data JPA temos um recurso chamado **Derived Query Methods**

# Recursos do Spring Data JPA



São métodos que você **declara no Repository** usando nomenclatura **padronizada**, e o Spring Data JPA gera automaticamente o SQL necessário em tempo de execução.

# Alguns exemplos



- `findBy` → busca registros.

```
java

Optional<User> findByEmail(String email);
```

- `existsBy` → verifica existência.

```
java

boolean existsByEmail(String email);
```

- `countBy` → conta registros.

```
java

long countByRole(String role);
```

- `deleteBy` → apaga registros.

```
java

void deleteByEmail(String email);
```

Combinações são possíveis:

```
java

Optional<User> findByEmailAndStatus(String email, String status);
List<User> findByAgeGreaterThan(Integer age);
```

# Tabelas para consulta

## Prefixos (o que fazer)



|                       |                                |                                             |                                                       |
|-----------------------|--------------------------------|---------------------------------------------|-------------------------------------------------------|
| <code>findBy</code>   | Busca registros                | <code>findByEmail(String email)</code>      | <code>SELECT * FROM user WHERE email = ?</code>       |
| <code>getBy</code>    | Igual ao <code>findBy</code>   | <code>getByUsername(String username)</code> | <code>SELECT * FROM user WHERE username = ?</code>    |
| <code>readBy</code>   | Igual ao <code>findBy</code>   | <code>readById(Long id)</code>              | <code>SELECT * FROM user WHERE id = ?</code>          |
| <code>queryBy</code>  | Igual ao <code>findBy</code>   | <code>queryByRole(String role)</code>       | <code>SELECT * FROM user WHERE role = ?</code>        |
| <code>existsBy</code> | Verifica existência            | <code>existsByEmail(String email)</code>    | <code>SELECT COUNT(*)...</code>                       |
| <code>countBy</code>  | Conta registros                | <code>countByRole(String role)</code>       | <code>SELECT COUNT(*) FROM user WHERE role = ?</code> |
| <code>deleteBy</code> | Deleta registros               | <code>deleteByEmail(String email)</code>    | <code>DELETE FROM user WHERE email = ?</code>         |
| <code>removeBy</code> | Igual ao <code>deleteBy</code> | <code>removeByStatus(String status)</code>  | <code>DELETE FROM user WHERE status = ?</code>        |

# Tabelas para consulta

## Operadores de Comparação

|              |                 |                                                        |                                        |
|--------------|-----------------|--------------------------------------------------------|----------------------------------------|
| Is, Equals   | Igualdade       | <code>findByEmailEquals(String email)</code>           | <code>WHERE email = ?</code>           |
| Not          | Negação         | <code>findByEmailNot(String email)</code>              | <code>WHERE email &lt;&gt; ?</code>    |
| IsNull       | Valor nulo      | <code>findByPhoneIsNull()</code>                       | <code>WHERE phone IS NULL</code>       |
| IsNotNull    | Não nulo        | <code>findByPhoneIsNotNull()</code>                    | <code>WHERE phone IS NOT NULL</code>   |
| GreaterThan  | Maior que       | <code>findByAgeGreaterThan(int age)</code>             | <code>WHERE age &gt; ?</code>          |
| LessThan     | Menor que       | <code>findByAgeLessThan(int age)</code>                | <code>WHERE age &lt; ?</code>          |
| Between      | Entre valores   | <code>findByAgeBetween(int start, int end)</code>      | <code>WHERE age BETWEEN ? AND ?</code> |
| Like         | LIKE            | <code>findByNameLike(String name)</code>               | <code>WHERE name LIKE ?</code>         |
| StartingWith | Inicia com      | <code>findByNameStartingWith(String prefix)</code>     | <code>WHERE name LIKE 'prefixX'</code> |
| EndingWith   | Termina com     | <code>findByNameEndingWith(String suffix)</code>       | <code>WHERE name LIKE '%suffix'</code> |
| Containing   | Contém          | <code>findByNameContaining(String str)</code>          | <code>WHERE name LIKE '%strX'</code>   |
| In           | Dentro de lista | <code>findByRoleIn(List&lt;String&gt; roles)</code>    | <code>WHERE role IN (...)</code>       |
| NotIn        | Fora da lista   | <code>findByRoleNotIn(List&lt;String&gt; roles)</code> | <code>WHERE role NOT IN (...)</code>   |



# Tabelas para consulta

## Operadores lógicos



|     |           |                                                                    |                                                 |
|-----|-----------|--------------------------------------------------------------------|-------------------------------------------------|
| And | E lógico  | <code>findByEmailAndStatus(String<br/>email, String status)</code> | <code>WHERE email = ? AND<br/>status = ?</code> |
| Or  | OU lógico | <code>findByEmailOrPhone(String<br/>email, String phone)</code>    | <code>WHERE email = ? OR<br/>phone = ?</code>   |



# Tabelas para consulta

## Ordenação e Limites



|            |                   |                                               |                                       |
|------------|-------------------|-----------------------------------------------|---------------------------------------|
| OrderBy    | Ordenação         | findByStatusOrderByNameAsc(<br>String status) | WHERE status = ? ORDER<br>BY name ASC |
| Top, First | Limita resultados | findTop3ByStatus(String<br>status)            | WHERE status = ? LIMIT<br>3           |

```
@Repository 3 usages new *
public interface UserRepository extends JpaRepository<User, Long> {
    boolean existsByEmail(String email); no usages new *
}
```

sql

```
select count(*) > 0 from users where email = ?
```

```
@PostMapping() no usages new *
public ResponseEntity<User> register(@RequestBody User user) {
    if (userRepository.existsByEmail(user.getEmail())) {
        throw new DuplicateEmailException(
            String.format("Já existe um usuário com o e-mail %s", user.getEmail()));
    }
    var userSaved = userRepository.save(user);
    return new ResponseEntity<>(userSaved, HttpStatus.CREATED);
}
```

# @RestControllerAdvice (Aviso, recomendação ou opinião)



POST http://localhost:8080/us + ...

http://localhost:8080/user

POST http://localhost:8080/user

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary JSON

```
1  
2  "name": "Genilson Medeiros",  
3  "email": "genilson.martins@unifacisa.edu.br"  
4
```

Body Cookies Headers (4) Test Results Status: 400 Bad Request

Pretty Raw Preview Visualize JSON

```
1  
2  "status": "BAD_REQUEST",  
3  "messages": [  
4    "Já existe um usuário com o e-mail genilson.martins@unifacisa.edu.br"  
5  ],  
6  "code": 400  
7
```