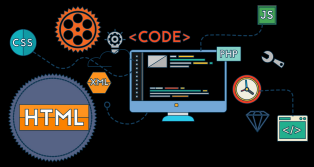
Tópicos Avançados de Programação

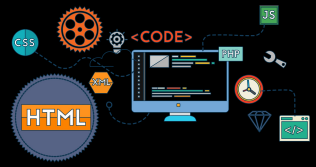
Genilson Medeiros

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


<CODE>
JS
CSS
PHP
HTML

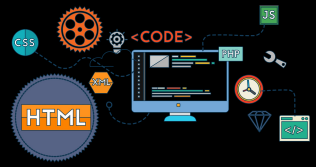

docker

Docker



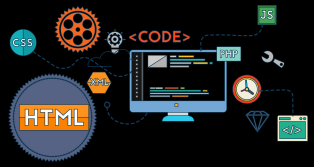
O Docker simplifica a complexidade na preparação de ambientes de desenvolvimento, teste e até produção. Ele elimina a necessidade de instalar várias bibliotecas e aplicativos, como sistemas de gerenciamento de banco de dados (SGBD), servidores web, gerenciadores de pacotes, entre outros, evitando a famosa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

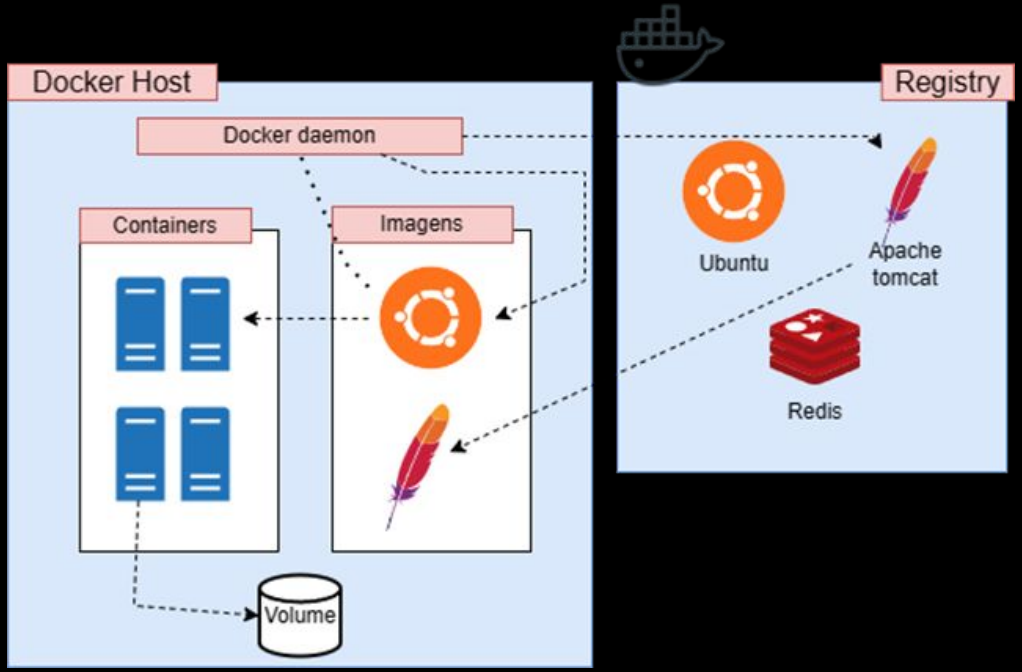
Docker



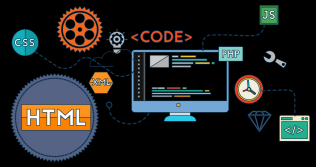
Lembre-se, Docker não é uma máquina virtual (VM). Docker é uma plataforma de virtualização de containeres. É uma maneira de empacotar aplicativos e todas as suas dependências em um pacote portátil chamado container.

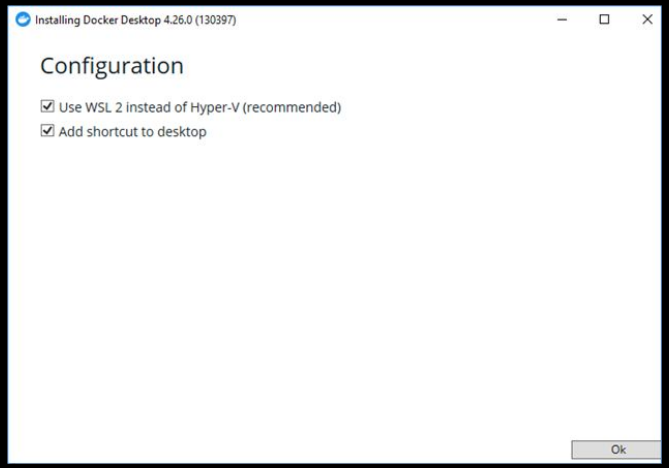
Docker

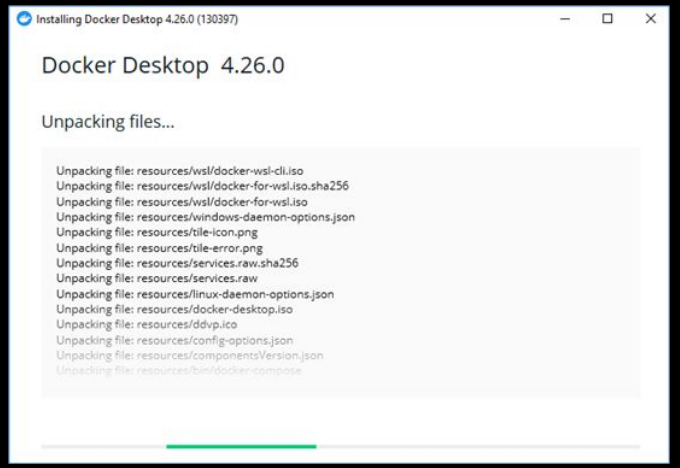

CSS
<CODE>
JS
PHP
HTML


Docker Host
Docker daemon
Containers
Imagens
Volume
Registry
Ubuntu
Apache tomcat
Redis

Instalação

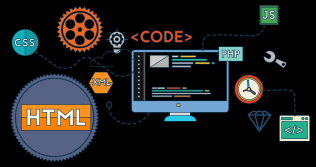


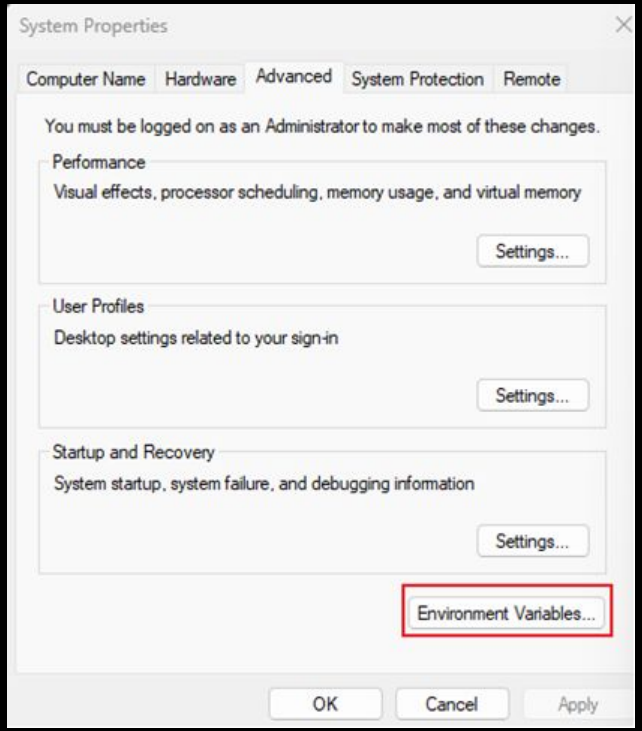




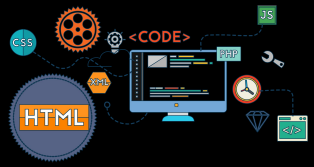
Mesmo após instalar o Docker Desktop no Windows, você pode utilizar apenas o CLI (linha de comando) para realizar todas as operações disponíveis. Também é possível usar o Docker via interface gráfica, acessível diretamente pelo Docker Desktop.

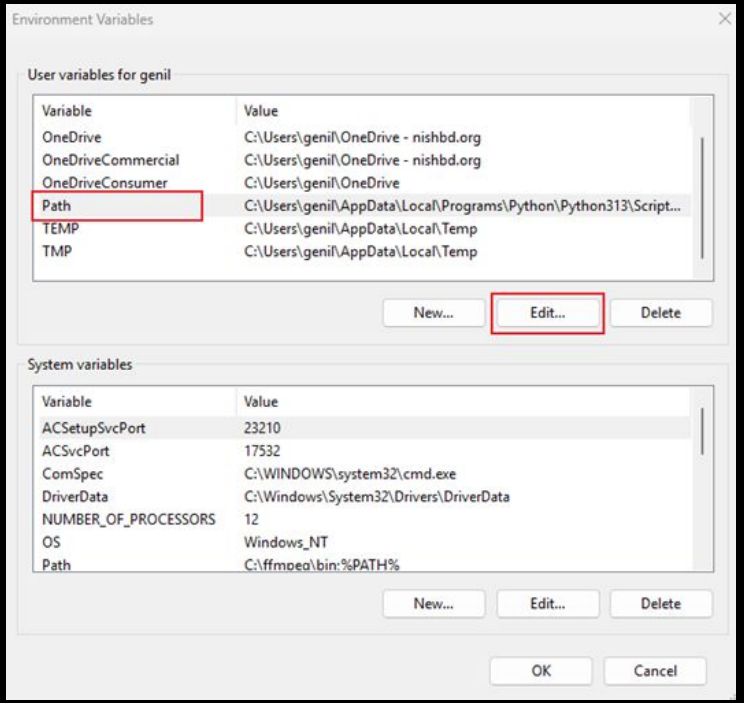
Variáveis de ambiente



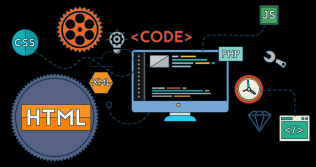


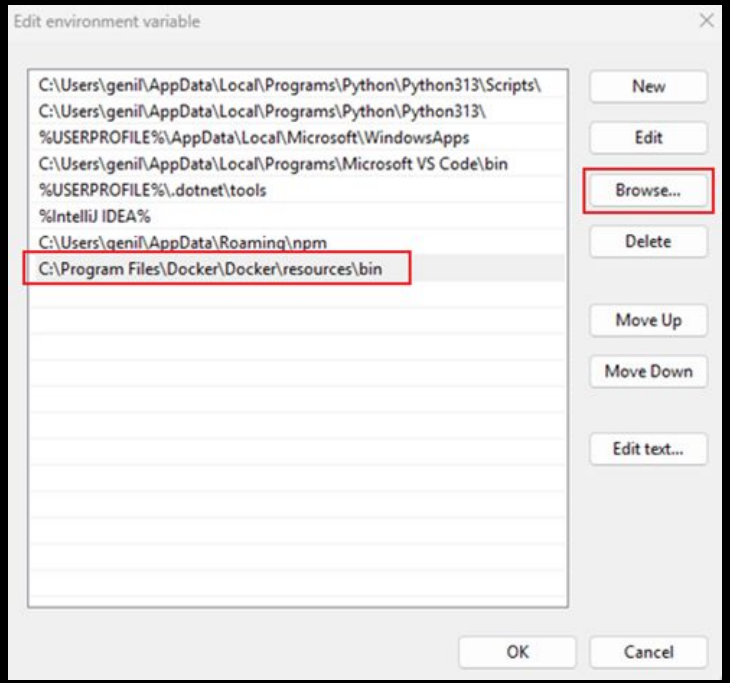
Variáveis de ambiente



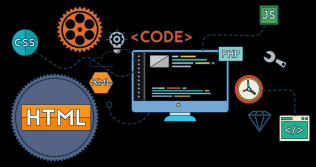


Variáveis de ambiente





Instalação no linux



- Atualizar a lista de pacotes:

  sudo apt update

- Instalar alguns pacotes que são pré-requisitos

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

- Adiciona a chave GPG do repositório oficial do repositório Docker

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

- Adiciona o repositório Docker para o APT

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

- Certificar que será instalado a partir do repositório Docker e não do repositório padrão do Ubuntu

  apt-cache policy docker-ce

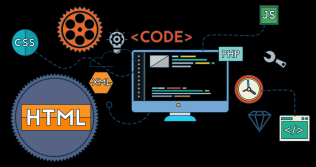
- Instalação do Docker

  sudo apt install docker-ce

- Inicializar o Docker

  sudo systemctl status docker

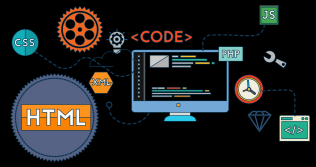
Instalação no linux



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

Star 44,824 Follow @dbeaver_news

search here … Go

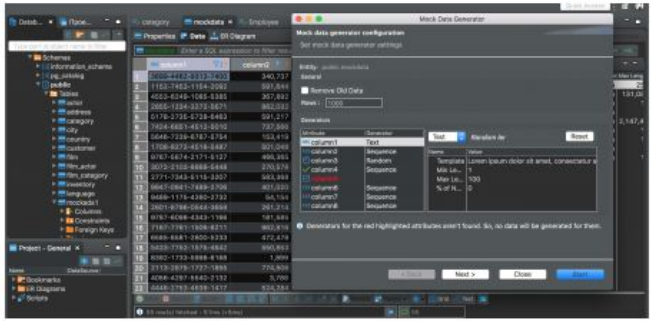
DBeaver Community

Free Universal Database Tool

Home | About | Download | Documentation | News | Support | DBeaver PRO | CloudBeaver | DBeaver Merch | Join our team

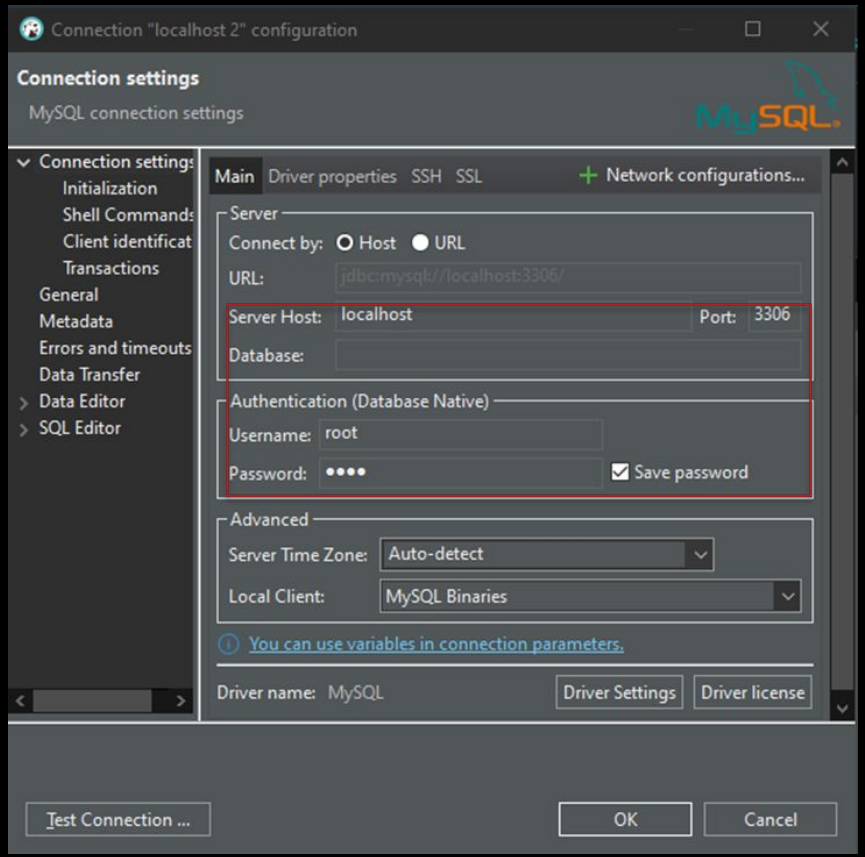
Universal Database Tool

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.



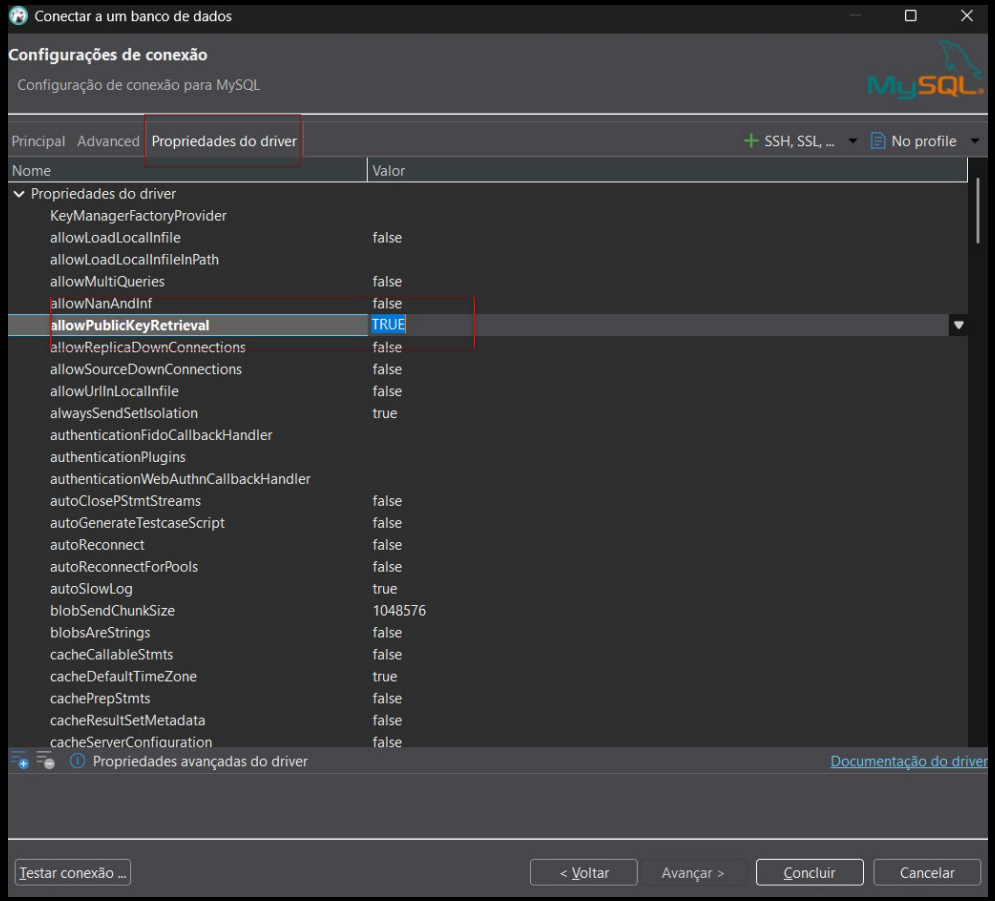
DBeaver


CSS
<CODE>
JS
HTML

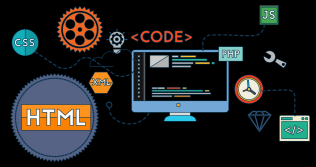

Connection "localhost 2" configuration
Connection settings
MySQL connection settings
MySQL
Connection settings
Initialization
Shell Commands
Client identificat
Transactions
General
Metadata
Errors and timeouts
Data Transfer
Data Editor
SQL Editor
Main
Driver properties
SSH
SSL
Network configurations...
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


<CODE>
HTML

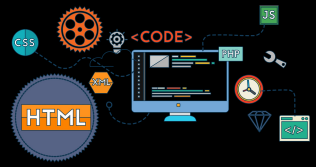

Conectar a um banco de dados
Configurações de conexão
Configuração de conexão para MySQL
MySQL
Principal
Advanced
Propriedades do driver
SSH, SSL, ...
No profile
Nome
Valor
Propriedades do driver
KeyManagerFactoryProvider
allowLoadLocalInfile false
allowLoadLocalInfileInPath
allowMultiQueries false
allowNanAndInf false
allowPublicKeyRetrieval TRUE
allowReplicaDownConnections false
allowSourceDownConnections false
allowUrlInLocalInfile false
alwaysSendSetIsolation true
authenticationFidoCallbackHandler
authenticationPlugins
authenticationWebAuthnCallbackHandler
autoClosePStmtStreams false
autoGenerateTestcaseScript false
autoReconnect false
autoReconnectForPools false
autoSlowLog true
blobSendChunkSize 1048576
blobsAreStrings false
cacheCallableStmts false
cacheDefaultTimeZone true
cachePrepStmts false
cacheResultSetMetadata false
cacheServerConfiguration false
Propriedades avançadas do driver
Documentação do driver
Testar conexão ...
< Voltar
Avançar >
Concluir
Cancelar

Acessando via CLI




```
PS C:\Users\       > docker exec -it mysql_bd mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 9.4.0 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Acessando via CLI




```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| my_db              |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.001 sec)

mysql> use my_db
Database changed
mysql>
```

Visualizar e selecionar o database

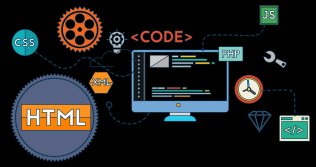
```
mysql> show tables;
+----------------+
| Tables_in_my_db |
+----------------+
| user           |
+----------------+
1 row in set (0.001 sec)

mysql>
```

Visualizar as tabelas

```
mysql> select * from user;
+----+----------------+----------+
| id | name           | password |
+----+----------------+----------+
|  1 | Maria da Silva | maria123 |
|  2 | Jo◆o Batista   | joao1234 |
+----+----------------+----------+
2 rows in set (0.001 sec)

mysql>
```

Realizar consulta

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

spring initializr

Project
- ○ Gradle - Groovy
- ○ Gradle - Kotlin
- ● Maven

Language
- ● Java
- ○ Kotlin
- ○ Groovy

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	gm2.br
Artifact	gm2-social-network
Name	gm2-social-network
Description	Rede social GM2
Package name	gm2.br.gm2-social-network
Packaging	● Jar ○ War
Java	○ 24 ○ 21 ● 17

Dependencies — ADD DEPENDENCIES... CTRL + B

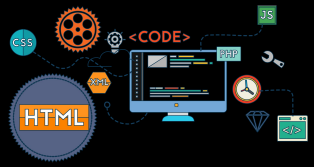
Lombok DEVELOPER TOOLS
Java annotation library which helps to reduce boilerplate code.

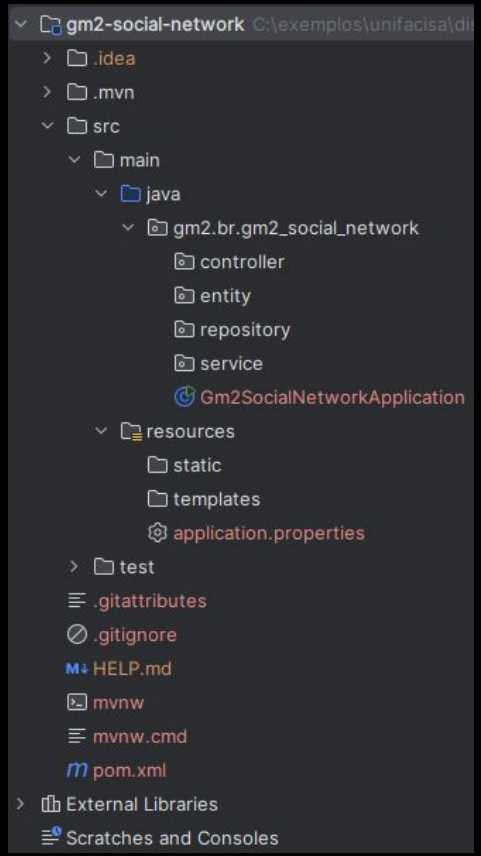
Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

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)

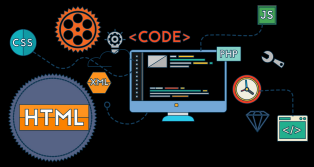




Entidade user

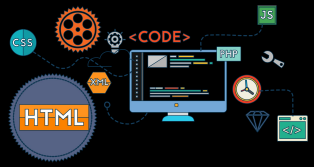

```java
package gm2.br.gm2_social_network.entity;

import gm2.br.gm2_social_network.utils.Constants;
import jakarta.persistence.*;

@Entity  2 usages  new *
@Table(name = "user")
public class User {
    @Id  no usages
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(length = Constants.GENERAL_MAX_LENGTH, nullable = false)  no usages
    private String name;

    @Column(length = Constants.GENERAL_MAX_LENGTH, nullable = false, unique = true)  no usages
    private String email;

    @Column(length = Constants.PHOTO_PATH, nullable = true)  no usages
    private String photo;

    @Column(length = Constants.MESSAGE_LENGTH, nullable = true)  no usages
    private String message;
}
```

Utils



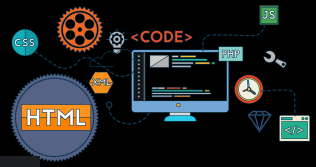

```java
package gm2.br.gm2_social_network.utils;

public final class Constants {  5 usages  new *
    public static final int GENERAL_MAX_LENGTH = 100;  2 usages
    public static final int MESSAGE_LENGTH = 200;  1 usage
    public static final int PHOTO_PATH = 200;  1 usage
}
```

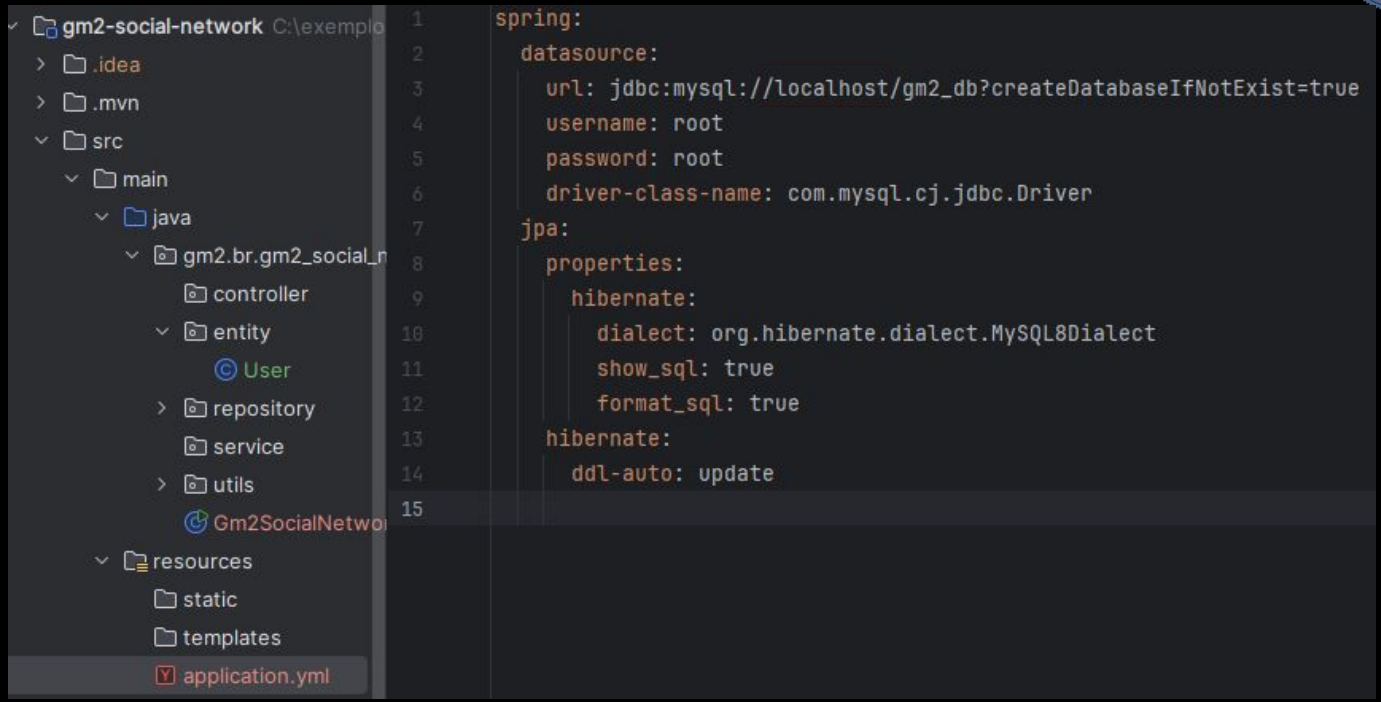
Repositório




```java
package gm2.br.gm2_social_network.repository;

import gm2.br.gm2_social_network.entity.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository  no usages  new *
public interface UserRepository extends JpaRepository<User, Long> {
}
```

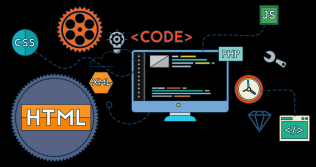
Repositório





[livro_docker_pratico/backend-app/src/main/resources/application.yml at main · genilsonmm/livro_docker_pratico · GitHub]

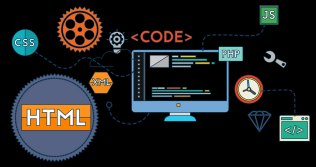
Rode o seu projeto



Run Gm2SocialNetworkApplication

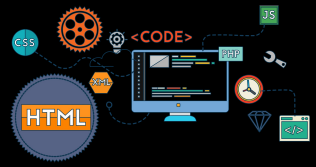
```
    Autocommit mode: undefined/unknown
    Isolation level: undefined/unknown
    Minimum pool size: undefined/unknown
    Maximum pool size: undefined/unknown
2025-08-16T14:09:33.889-03:00  INFO 22488 --- [          main] o.h.e.t.
Hibernate:
    create table user (
        id bigint not null auto_increment,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

Controller




```java
@RestController  no usages  new *
@RequestMapping("/user")
public class UserController {

    private final UserRepository userRepository;  2 usages

    private UserController(UserRepository userRepository) {  no usages  new *
        this.userRepository = userRepository;
    }

    @PostMapping()  no usages  new *
    public ResponseEntity<User> register(@RequestBody User user) {
        var userSaved = userRepository.save(user);
        return new ResponseEntity<>(userSaved, HttpStatus.CREATED);
    }
}
```

Postman



POST http://localhost:8080/us

http://localhost:8080/user

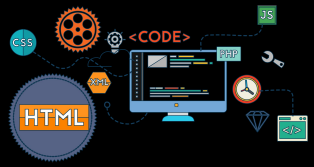
POST http://localhost:8080/user

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

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

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

Erro de principiante



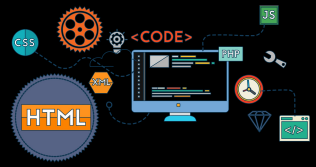

```java
@RequestMapping("/user")
public class UserController {

    private final UserRepository userRepository;  2 usages        userRepository: "org.springframework.dat

    private UserController(UserRepository userRepository) {  no usages  new *
        this.userRepository = userRepository;
    }

    @PostMapping()  no usages  new *
    public ResponseEntity<User> register(@RequestBody User user) {  user: User@12088
        var userSaved = userRepository.save(user);  user: User@12088    userRepository: "org.springframe
        return new ResponseEntity<>(userSaved, HttpStatus.CREATED);
    }
}
```

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



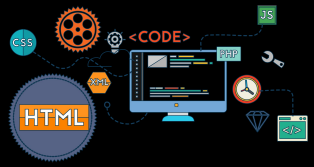
A anotação @Data do Lombok em Java é uma maneira conveniente de reduzir código boilerplate, gerando automaticamente métodos como getters, setters, equals(), hashCode() e toString() para classes. Ela é um atalho que combina as funcionalidades das anotações @Getter, @Setter, @EqualsAndHashCode e @ToString

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


```java
@Data  5 usages  new *
@Entity
@Table(name = "user")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(length = Constants.GENERAL_MAX_LENGTH, nullable = false)
    private String name;

    @Column(length = Constants.GENERAL_MAX_LENGTH, nullable = false, unique = true)
    private String email;

    @Column(length = Constants.PHOTO_PATH, nullable = true)
    private String photo;

    @Column(length = Constants.MESSAGE_LENGTH, nullable = true)
    private String message;
}
```

Registro de usuário



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 (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

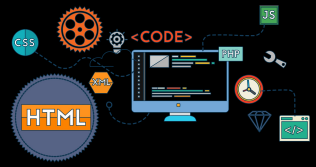
Não trate o erro ao tentar cadastrar usuários com o mesmo email

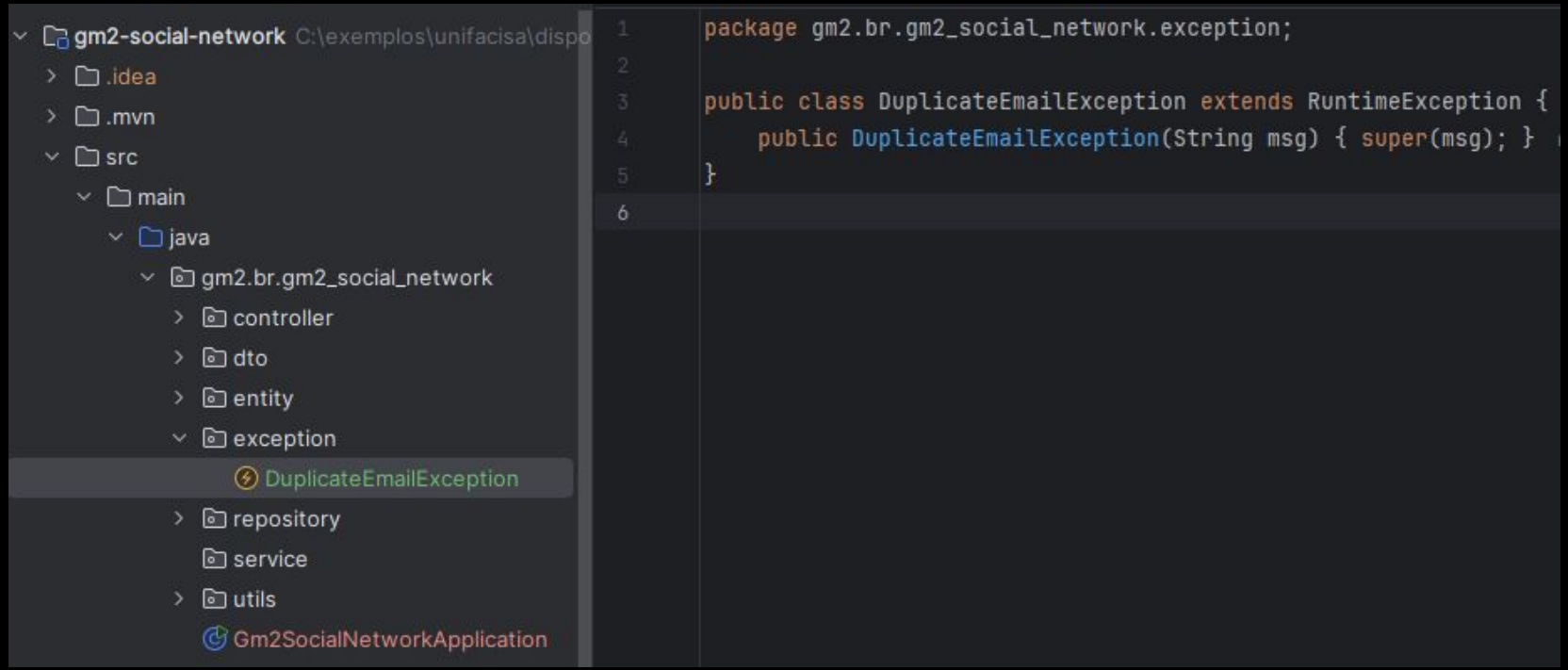
Boa prática para tratar erros?

Classe genérica para retorno de erros

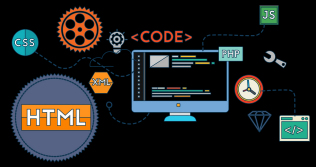

```java
public class ResponseDTO {  3 usages  new *
    @Getter
    private List<String> messages;

    @Getter
    private HttpStatus status;

    @Getter
    private int code;

    public ResponseDTO(List<String> message, HttpStatus status) {  no usages  new
        this.messages = message;
        this.status = status;
        this.code = status.value();
    }

    public ResponseDTO(String message, HttpStatus status) {  1 usage  new *
        this.messages = Arrays.asList(message);
        this.status = status;
        this.code = status.value();
    }
```

Custom exception



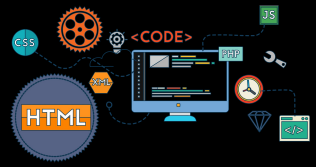


@RestControllerAdvice (Aviso, recomendação ou opinião)



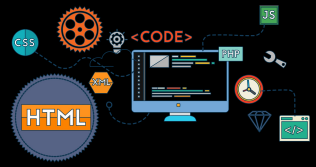

```java
@RestControllerAdvice  no usages  new *
public class ApplicationAdviceController {

    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



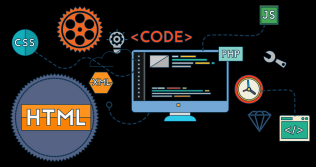
No Spring Data JPA temos um recurso chamado Derived Query Methods

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);
```

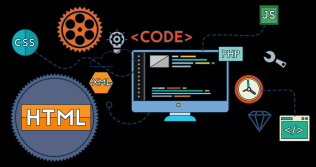
- `deleteBy` → apaga registros.

```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)

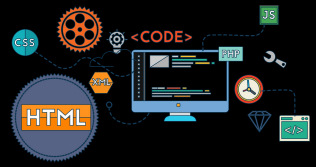
`findBy`	Busca registros	`findByEmail(String email)`	`SELECT * FROM user WHERE email = ?`
`getBy`	Igual ao `findBy`	`getByUsername(String username)`	`SELECT * FROM user WHERE username = ?`
`readBy`	Igual ao `findBy`	`readById(Long id)`	`SELECT * FROM user WHERE id = ?`
`queryBy`	Igual ao `findBy`	`queryByRole(String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

Tabelas para consulta

Operadores de Comparação

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

Tabelas para consulta

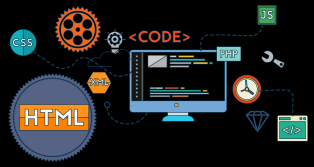


Operadores lógicos

`And`	E lógico	`findByEmailAndStatus(String email, String status)`	`WHERE email = ? AND status = ?`
`Or`	OU lógico	`findByEmailOrPhone(String email, String phone)`	`WHERE email = ? OR phone = ?`

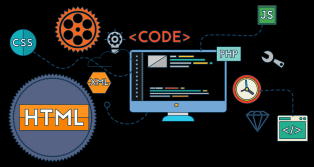
Tabelas para consulta

Ordenação e Limites



`OrderBy`	Ordenação	`findByStatusOrderByNameAsc(String status)`	`WHERE status = ? ORDER BY name ASC`
`Top`, `First`	Limita resultados	`findTop3ByStatus(String status)`	`WHERE status = ? LIMIT 3`

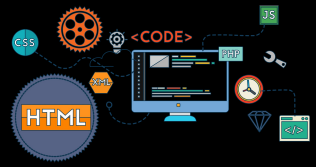
Verificar se usuário já existe




```java
@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 = ?
```

Antes de salvar o usuário…




```
@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)