



Análise e Desenvolvimento de Sistemas

Prof. MSc Marcelo Tomio Hama

## AULA 5

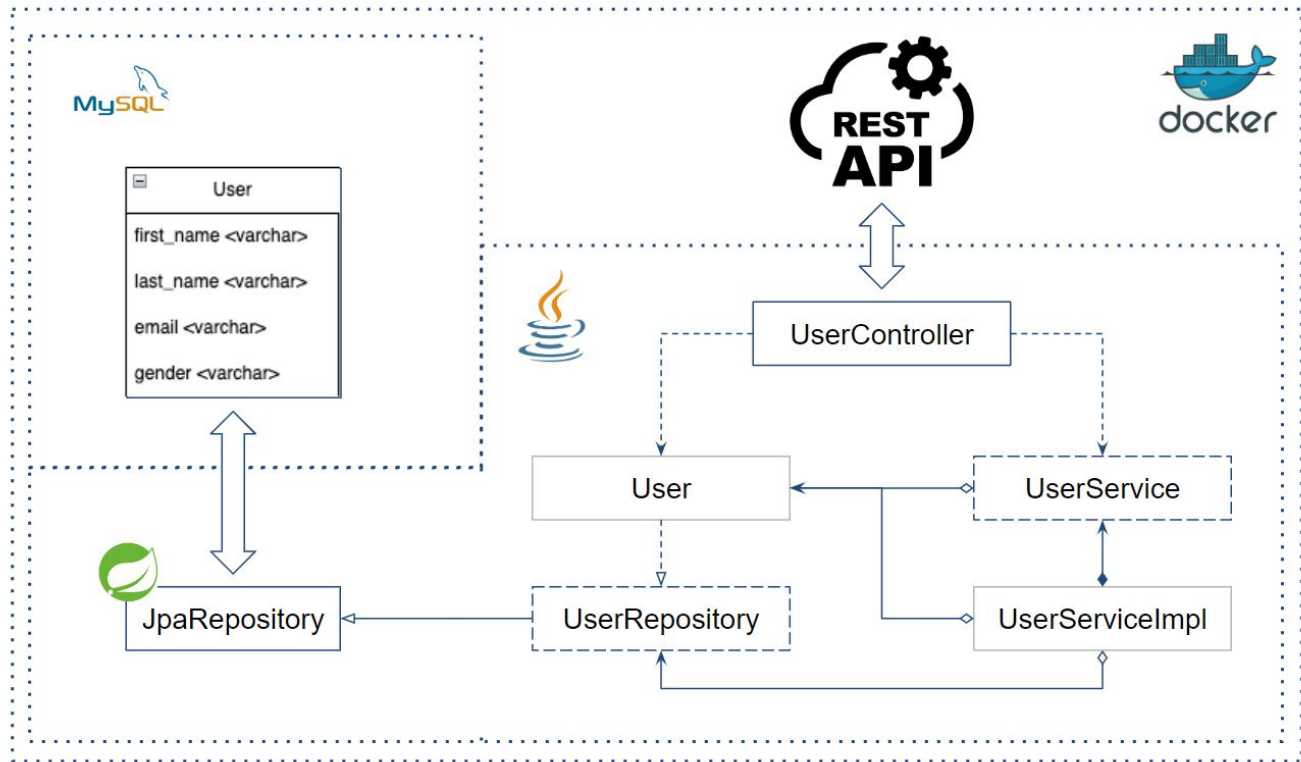
# API Restful com Spring Data JPA - Implementação do CRUD, Paginação de resultados, Manipulação de Exceções.

## PARTE I

### OBJETIVOS

1. Revisar e aplicar os conhecimentos obtidos da aula anterior
2. Implementar um controller completamente do zero, com métodos CRUD mapeados em URLs customizadas;
3. Entender o conceito de paginação e ordenamento e implementá-los;
4. Manipular exceções e tratar casos específicos.

# Arquitetura do Nosso Case



Trata-se de uma pequena aplicação dockerizada que executa sob um container docker instalado e publicado localmente.

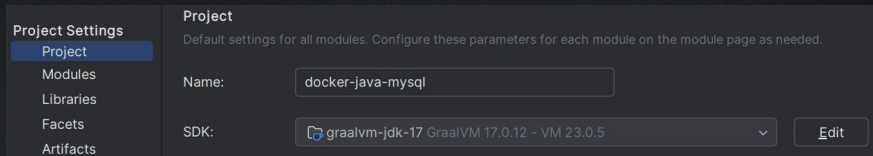
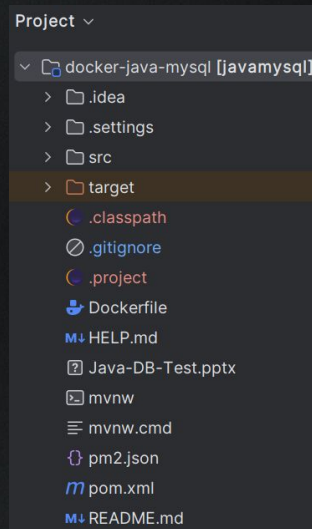
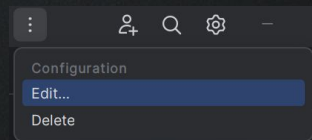
A aplicação faz uso das funções mínimas do JPA, do MySQL (executado em outro container), e que publica APIs para consumo local.

O JPA (Java Persistence API) facilita o desenvolvimento de tecnologias de acesso a dados.

# Hands-On Laboratorial | Ambiente

## Passo-a-Passo Geral (Windows 10)

1. Faça a instalação do IntelliJ IDEA
  - a. Download: <https://www.jetbrains.com/pt-br/idea/download/?section=windows>
  - b. Configure a licença gratuita: <https://www.jetbrains.com/shop/eform/students>
2. Faça a instalação do Docker Desktop;
  - a. Download: <https://www.docker.com/products/docker-desktop/>
  - b. Crie sua conta em <https://app.docker.com/> e faça o login no Docker Desktop;
3. Faça o clone do repositório:  
<https://github.com/marcelohama/docker-java-mysql>
4. Crie o projeto no IntelliJ IDEA
  - a. File > New -> Project from existing sources, selecione o diretório raiz do repositório clonado “docker-java-mysql”;
  - b. No ícone “engrenagem” vá em Project Structure e depois em Project, e configure o SDK para usar o graalvm-jdk-17;
  - c. Nos “3 pontinhos”, clique em edit para criar uma config de build, e use a seguinte linha para run: “*-Dmaven.test.skip=true package*”





# Hands-On Laboratorial | MySQL

## Subindo um Container MySQL e Criando Tabela

1. Abra o terminal do IntelliJ IDEA, local ao diretório raiz do projeto
2. Com o dashboard do Docker Desktop aberto, execute a linha para criar uma rede para ponte entre containers/aplicações:

```
$ docker network create --driver bridge javamysql-network
```

3. Crie um container mysql com imagem da comunidade:

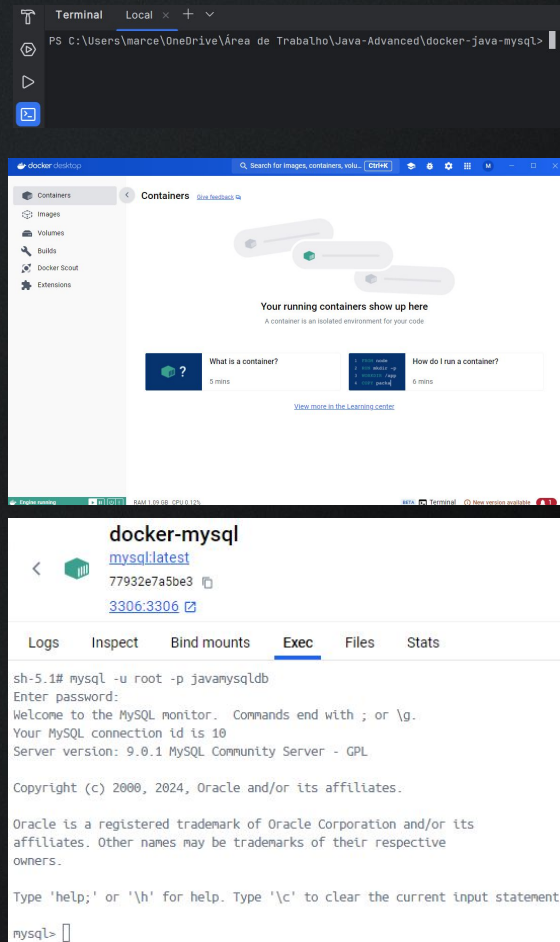
```
$ docker container run -p 3306:3306 --name docker-mysql --network  
javamysql-network -e MYSQL_ROOT_PASSWORD=1q2w3e4r5t -e  
MYSQL_DATABASE=javamysqldb -d mysql:latest
```

4. Entre no bash do servidor MySQL, clicando no container MySQL visível no Docker Desktop e depois em exec, e entre no CLI do servidor MySQL com o root no schema criado:

```
$ mysql -u root -p javamysqldb
```

5. Dentro do CLI, execute a query para criar uma tabela USERS:

```
$ CREATE TABLE users ( id int NOT NULL,  
first_name varchar(255) NOT NULL,  
last_name varchar(255),  
email varchar(255),  
gender varchar(255)  
);
```



# Hands-On Laboratorial | MySQL

## Criando Registros no Banco

1. Insira os registros no banco de dados:

```
$ INSERT INTO users (id, first_name, last_name, gender, email) VALUES (1, 'donatello', 'Ninja Turtle', 'male', 'donatello@fiap.com.br');
```

```
$ INSERT INTO users (id, first_name, last_name, gender, email) VALUES (2, 'leonardo', 'Ninja Turtle', 'male', 'leonardo@fiap.com.br');
```

```
$ INSERT INTO users (id, first_name, last_name, gender, email) VALUES (3, 'michelangelo', 'Ninja Turtle', 'male', 'michelangelo@fiap.com.br');
```

```
$ INSERT INTO users (id, first_name, last_name, gender, email) VALUES (4, 'rafael', 'Ninja Turtle', 'male', 'rafael@fiap.com.br');
```

2. Faça um teste de leitura dos registros criados:

```
$ SELECT * FROM users;
```

```
mysql> SELECT * FROM users;
```

id	first_name	last_name	email	gender
1	donatello	Ninja Turtle	donatello@fiap.com.br	male
2	leonardo	Ninja Turtle	leonardo@fiap.com.br	male
3	michelangelo	Ninja Turtle	michelangelo@fiap.com.br	male
4	rafael	Ninja Turtle	rafael@fiap.com.br	male

4 rows in set (0.00 sec)













1. Execute o build clicando no botão verde “play”;
2. Faça a construção do container com o pacote `jar` spring boot criado:

### 3. Execute o container criado:

```
. _ _ _ _ _  
/\ / _ _ _' _ _ _ _ _(_)_ _ _ _ _ \\\ \  
( ( )_ _ _ | ' _ | ' _ | ' _ V _ ' | _ _ _ _ _  
\V _ _ _| |_) | | | | | | | (_ | | ) ) )  
'_ | _ _ _ | . _ _ | | _ | | _ _ , / / / /  
=====|_|=====/_ _ _/_/_/_/  
  
:: Spring Boot ::                (v3.0.1)
```

```
2024-08-11T18:42:42.726Z INFO 1 --- [main] br.com.javamysql.JavamysqlApplication: Starting JavamysqlApplication v0.0.1-SNAPSHOT on localhost with PID 1 (/javamysql-0.0.1-SNAPSHOT.jar started by root in /)
```

Name	Image	Status	Port(s)	CPU (%)	Last started	Actions		
 <a href="#">docker-1</a> 77932e7a	<a href="#">mysql:lates</a>	Running	<a href="#">3306:3306</a> 	0.72%	37 minutes			
 <a href="#">java-bac</a> 7c9886dc	<a href="#">javamysql-l</a>	Running	<a href="#">8080:8080</a> 	0.38%	3 minutes a			



# Hands-On Laboratorial | Backend

## Simulando o Backend JAVA na rede

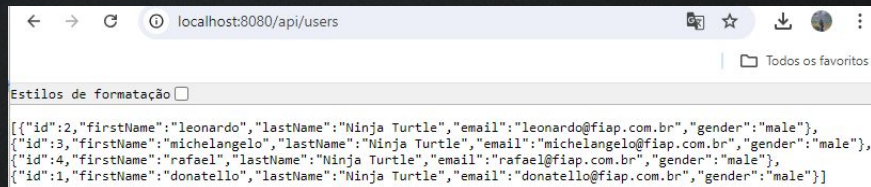
Acesse <http://localhost:8080/api/users>

## Opcional: Teste as chamadas com o Postman

1. Download: <https://www.postman.com/downloads/>
2. GET, POST, PUT, DELETE

## Opcional: Exponha seu Serviço na rede pública

1. Download do ngrok: <https://ngrok.com/download> e crie uma conta em <https://dashboard.ngrok.com>
2. Crie seu token em <https://dashboard.ngrok.com/tunnels/auth/tokens>
3. Abra o ngrok e configure o token:  
`$ ngrok authtoken <NGROK_AUTHTOKEN>`
4. Execute o ngrok no terminal do IntelliJ IDEA:  
`$ ./ngrok http 8080`
5. Em um dispositivo diferente, abra a URL:  
`<forwarding_url>/api/users`

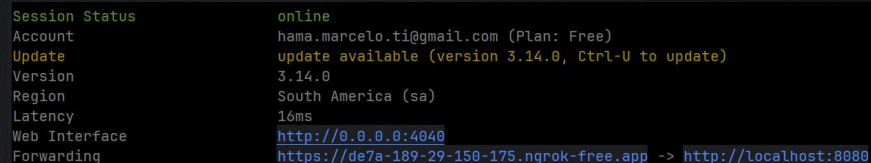


```

[[{"id":2,"firstName":"leonardo","lastName":"Ninja Turtle","email":"leonardo@fiap.com.br","gender":"male"},
{"id":3,"firstName":"michelangelo","lastName":"Ninja Turtle","email":"michelangelo@fiap.com.br","gender":"male"},
{"id":4,"firstName":"rafael","lastName":"Ninja Turtle","email":"rafael@fiap.com.br","gender":"male"},
{"id":1,"firstName":"donatello","lastName":"Ninja Turtle","email":"donatello@fiap.com.br","gender":"male"}]]

```

ID	Description	Owner	Metadata	Created
cr_4lAyJA	credential for 'hama.marcelo.ti@gmail.com'	hama.marcelo.ti@gmail.com	0 bytes	5y ago
cr_wBCJuT	Tunnel Authtoken for 'hama.marcelo.ti@gmail.com'	hama.marcelo.ti@gmail.com	0 bytes	2m ago



```

Session Status      online
Account             hama.marcelo.ti@gmail.com (Plan: Free)
Update              update available (version 3.14.0, Ctrl-U to update)
Version             3.14.0
Region              South America (sa)
Latency              16ms
Web Interface        http://0.0.0.0:4040
Forwarding           https://de7a-189-29-150-175.ngrok-free.app -> http://localhost:8080

```



# API Restful | Conceitos

HTTP Method	CRUD	Collection Resource (e.g. /users)	Single Resource (e.g. /users/123)
POST	Create	201 (Created), 'Location' header with link to /users/{id} containing new ID	Avoid using POST on a single resource
GET	Read	200 (OK), list of users. Use pagination, sorting, and filtering to navigate big lists	200 (OK), single user. 404 (Not Found), if ID not found or invalid
PUT	Update/Replace	405 (Method not allowed), unless you want to update every resource in the entire collection of resource	200 (OK) or 204 (No Content). Use 404 (Not Found), if ID is not found or invalid
PATCH	Partial Update/Modify	405 (Method not allowed), unless you want to modify the collection itself	200 (OK) or 204 (No Content). Use 404 (Not Found), if ID is not found or invalid
DELETE	Delete	405 (Method not allowed), unless you want to delete the whole collection — use with caution	200 (OK). 404 (Not Found), if ID not found or invalid

# Hands-On Laboratorial

Considere a seguinte tabela MySQL

```
$ CREATE TABLE addresses (  
  id int NOT NULL,  
  street_name varchar(128) NOT NULL,  
  neighborhood varchar(128),  
  state varchar(2)  
);
```

Implemente:

1. A classe entidade que representa esta tabela;
  - Use Lombok
2. A interface repository que extenda de JpaRepository, para manipular persistências;
3. A classe controller, para criar, obter, atualizar, e apagar registros de ADDRESSES;
  - Disponibilize os endpoints no formato “<domínio>/addresses/{param}”
4. Crie um endpoint em “<domínio>/dummy/{qty}”, que cria <qty> registros com campos quaisquer para testes, populando o seu banco de dados;
5. Trate as exceções, para ao menos 1 de cada família de respostas (2xx, 3xx, 4xx, e 5xx);
6. Publique seu código no github.

# Paginação de Resultados

## Conceito do Paginação

A paginação costuma ser útil quando temos um grande conjunto de dados e queremos apresentá-lo ao usuário em partes menores. Além disso, muitas vezes precisamos classificar esses dados por alguns critérios durante a paginação.

A paginação pode ser feita de diversas formas, e é comum que o offset e o tamanho da página sejam parâmetros do serviço.



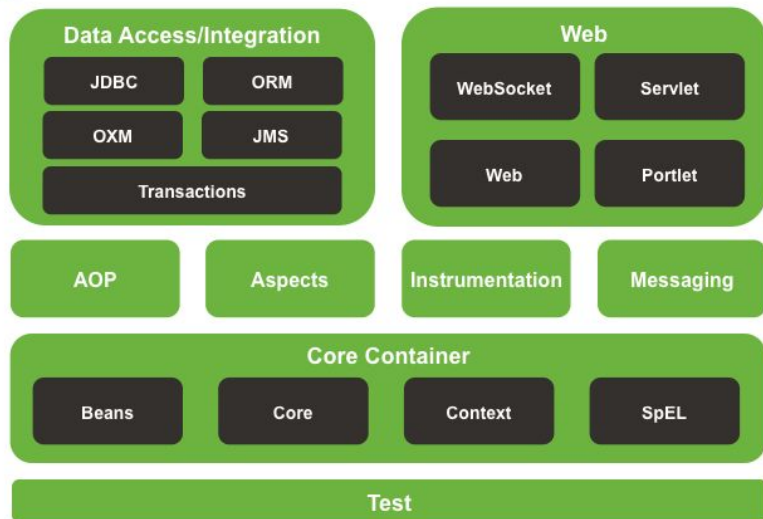
```
},
"pageable": {
  "sort": {
    "empty": false,
    "unsorted": false,
    "sorted": true
  },
  "offset": 0,
  "pageNumber": 0,
  "pageSize": 10,
  "paged": true,
  "unpaged": false
},
"totalPages": 1,
"totalElements": 5,
"last": true,
"size": 10,
"number": 0,
"sort": {
  "empty": false,
  "unsorted": false,
  "sorted": true
},
"numberOfElements": 5,
"first": true,
"empty": false
```



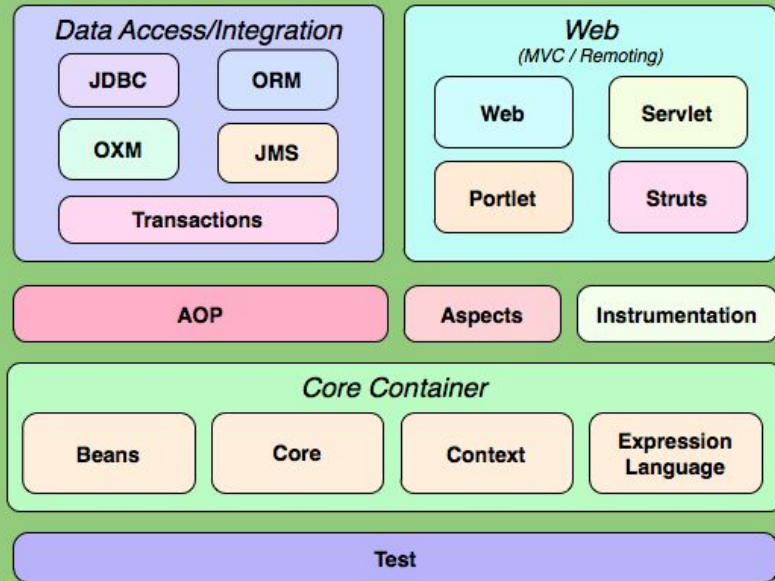
# Framework Spring



## Spring Framework Runtime



## Spring Framework Runtime



Execução do Spring Framework. Fonte:

<https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/spring-introduction.html>

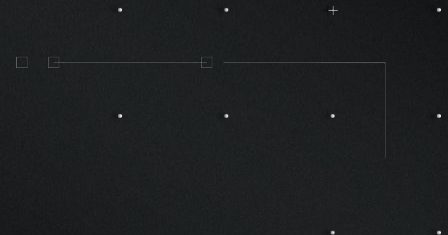
# Referências

## Email do Professor

[profmarcelo.hama@fiap.com.br](mailto:profmarcelo.hama@fiap.com.br)

## Bibliografias/Sites

- <https://docs.spring.io/spring-data/rest/reference/paging-and-sorting.html>  
<https://docs.spring.io/spring-data/rest/docs/2.0.0.M1/reference/html/paging-chapter.html>  
<https://fabiano-eprogramar.medium.com/api-rest-com-pagina%C3%A7%C3%A3o-usando-spring-data-e-query-9eddb29c9223>





*"O que sabemos é uma gota, o que ignoramos é um oceano."*

Isaac Newton

**FIM**

---