



Análise e Desenvolvimento de Sistemas

Prof. MSc Marcelo Tomio Hama

## AULA 2

# Introdução ao Spring Framework e Spring Boot (Frameworks x Bibliotecas) Lombok, Spring Boot dev tools, Spring Web. Verbos HTTP, Request, Response, HTTP Status Code

### OBJETIVOS

1. Entender pra que serve o Lombok e revisar o uso do Git
2. Entender o que é o Spring Framework e o Spring Boot
3. Ter uma visão geral das bibliotecas do Spring
4. Executar um projeto assistido, com o Spring Web + JPA
5. Revisar os verbos e status HTTP
6. Testar Requests/Responses com uma API RestFul

```
<dependencies>
...
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
  <version>1.18.30</version>
  <scope>provided</scope>
</dependency>
...
</dependencies>
```

# Sobre o Lombok

## É um “automatizador” de preenchimento de código

Quando se tem código padronizado, como por exemplo listas extensas de GET/SET para serem escritos, o lombok pode ser usado como forma de preencher de forma automática a implementação.

Documentação técnica:

<https://www.oracle.com/corporate/features/project-lombok.html>



```
void doSomething(Object arg) {  
    if(arg == null) {  
        throw new NullPointerException();  
    }  
    // ...  
}
```

```
void doSomething(@NonNull Object arg) {  
    // ...  
}
```

Exemplos de uso do Lombok. Fonte:

<https://medium.com/javarevisited/all-the-16-lombok-annotations-explained-in-a-4-minute-article-926f71934ec6>



# Casos de uso do Lombok

```
public class FullName {  
    private String firstName;  
    private String lastName;  
  
    public String getFirstName() {  
        return firstName;  
    }  
    public void setFirstName(String firstName) {  
        this.firstName = firstName;  
    }  
    public String getLastName() {  
        return lastName;  
    }  
    public void setLastName(String lastName) {  
        this.lastName = lastName;  
    }  
}
```

```
@Getter  
@Setter  
public class FullName {  
    private String firstName;  
    private String lastName;  
}
```

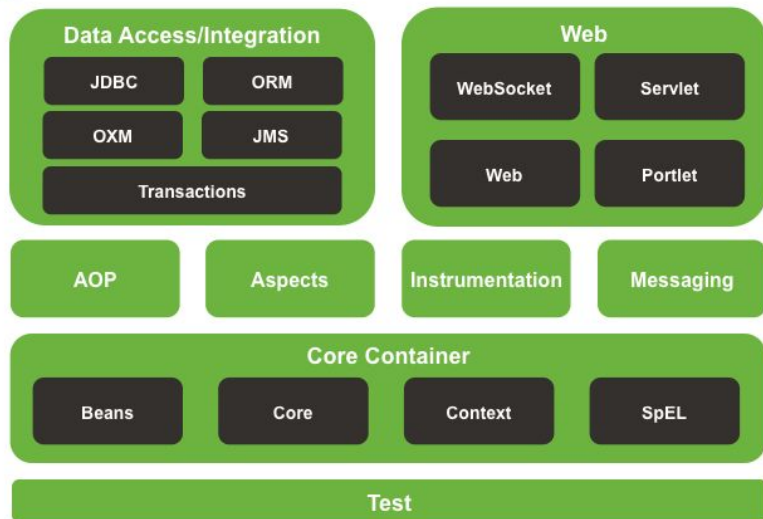
Exemplos de uso do Lombok. Fonte:

<https://medium.com/javarevisited/all-the-16-lombok-annotations-explained-in-a-4-minute-article-926f71934ec6>

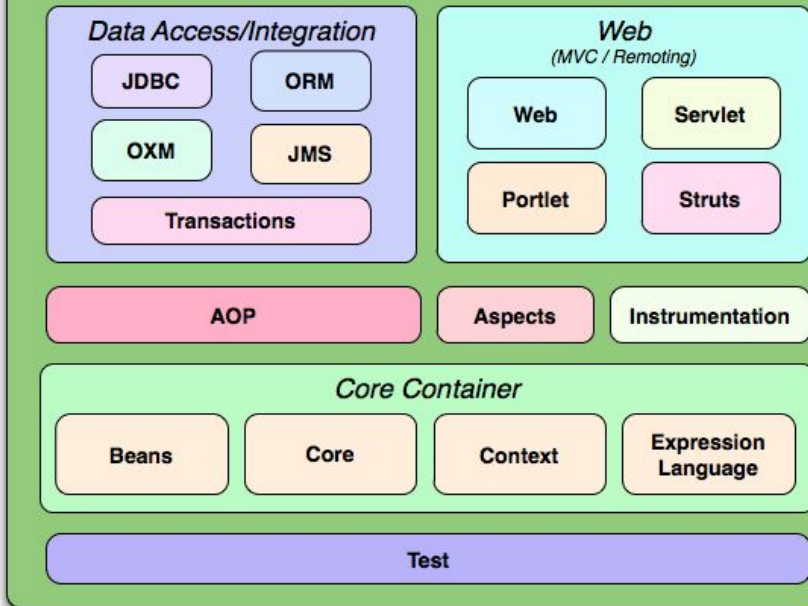
# Framework Spring



## Spring Framework Runtime



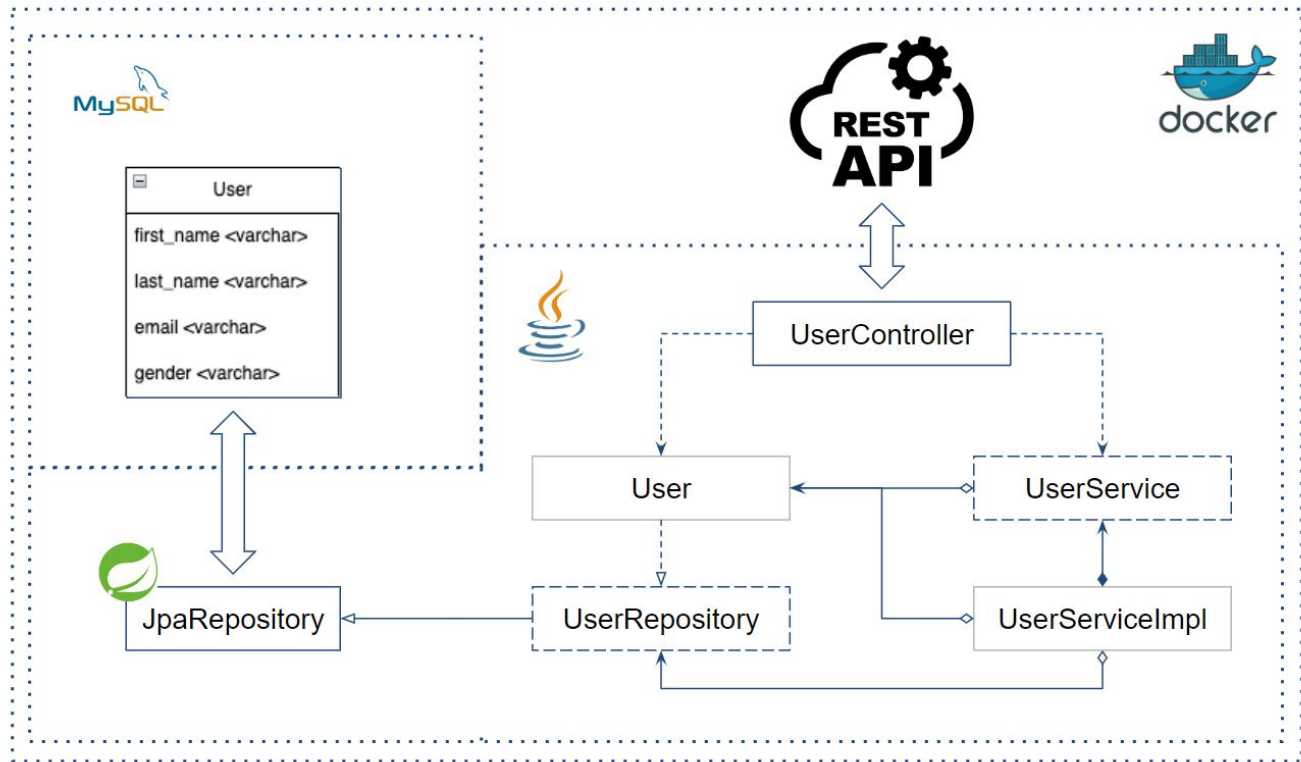
## Spring Framework Runtime



Execução do Spring Framework. Fonte:

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

# 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.

# Obtenhas as Ferramentas para o Ambiente

## IntelliJ IDEA

<https://www.jetbrains.com/idea/download/?section=windows>

Faça a instalação dos módulos do git conforme for solicitado

Faça a instalação do JDK conforme for solicitado (use JDK 17)

## Docker Desktop

<https://www.docker.com/products/docker-desktop/>

Crie sua conta caso não a possua e faça o login para operar o Docker daemon

## Faça o pull do projeto

<https://github.com/marcelohama/docker-java-mysql>

Crie como um projeto Maven a partir deste código

## Lombok (.jar)

<https://projectlombok.org/download>

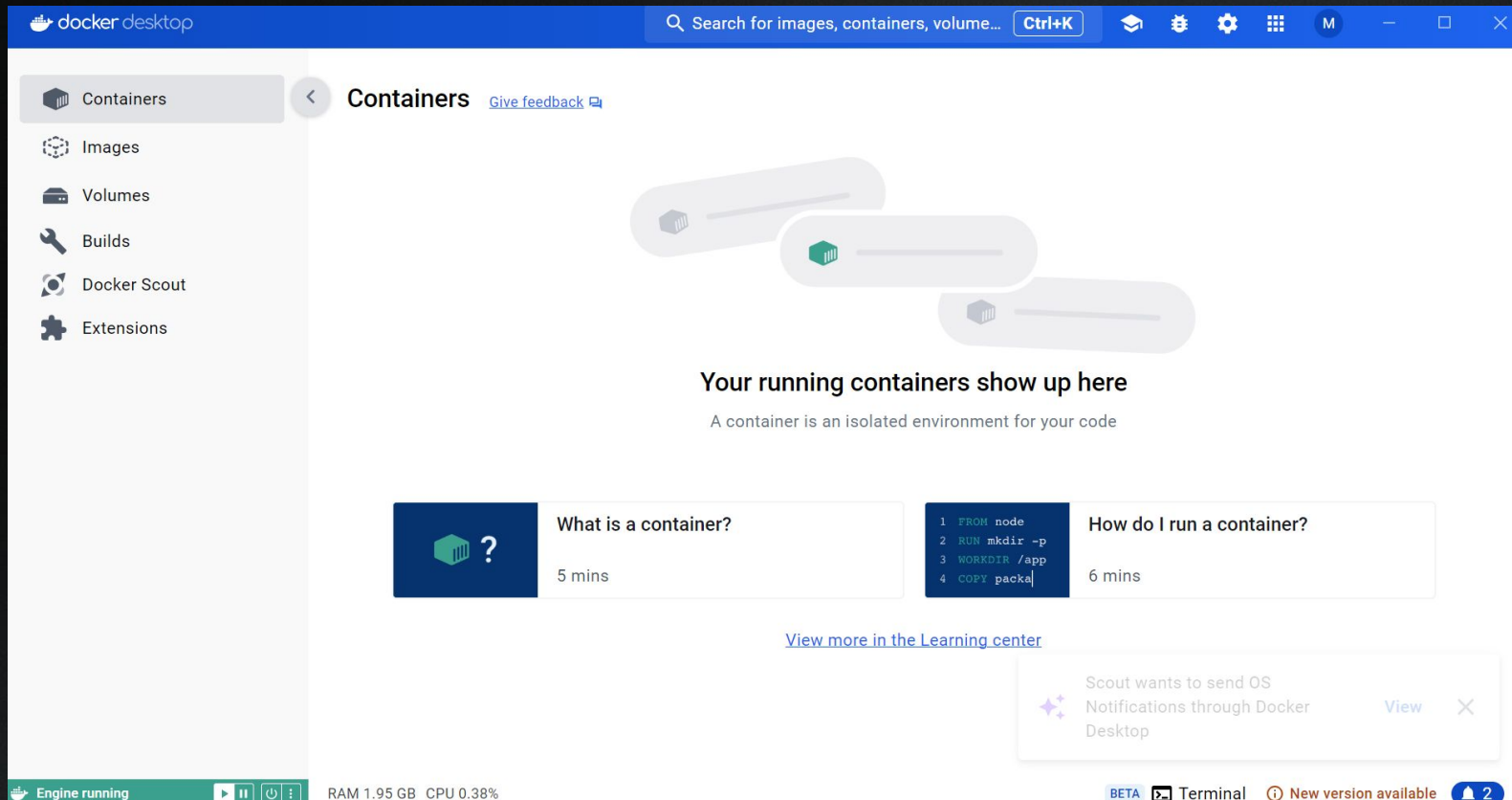
## Ngrok

<https://ngrok.com/download>





# Docker Desktop



The screenshot shows the Docker Desktop application window. The title bar is blue with the Docker logo and the text "docker desktop". Below the title bar is a search bar with the placeholder text "Search for images, containers, volume..." and a "Ctrl+K" button. The left sidebar contains a list of navigation items: Containers (selected), Images, Volumes, Builds, Docker Scout, and Extensions. The main content area is titled "Containers" and features a large graphic of three overlapping rounded rectangles, each containing a Docker logo. Below this graphic, the text reads "Your running containers show up here" and "A container is an isolated environment for your code". At the bottom of the main content area, there are two cards: "What is a container?" with a question mark icon and a "5 mins" duration, and "How do I run a container?" with a code snippet and a "6 mins" duration. The code snippet is: 

```
1 FROM node
2 RUN mkdir -p
3 WORKDIR /app
4 COPY packa
```

 Below these cards is a link "View more in the Learning center". In the bottom right corner, there is a notification from Docker Scout: "Scout wants to send OS Notifications through Docker Desktop" with a "View" button and a close icon. The bottom status bar shows "Engine running" with a play button icon, "RAM 1.95 GB CPU 0.38%", a "BETA" label, a "Terminal" button, a "New version available" notification, and a "2" badge.

docker desktop

Search for images, containers, volume... Ctrl+K

Containers Give feedback

Images

Volumes

Builds

Docker Scout

Extensions

Your running containers show up here

A container is an isolated environment for your code

What is a container?

5 mins

How do I run a container?

```
1 FROM node
2 RUN mkdir -p
3 WORKDIR /app
4 COPY packa
```


6 mins





[View more in the Learning center](#)




Scout wants to send OS Notifications through Docker Desktop View X



Engine running RAM 1.95 GB CPU 0.38% BETA Terminal New version available 2


# Clone do Repositório



 docker-java-mysql Public

 Pin  Unwatch **2**  Fork **0**  Star **0**




 main  1 Branch  0 Tags

 Add file  Code

 About






 **Marcelo Tomio Hama** add ppt 909e874 · 2 months ago  6 Commits

src	arrange files	9 months ago
target	arrange files	9 months ago
.gitignore	first commit	9 months ago
Dockerfile	arrange files	9 months ago
HELP.md	arrange files	9 months ago
Java-DB-Test.pptx	add ppt	2 months ago
README.md	fixes in meta	9 months ago
mvnw	arrange files	9 months ago
mvnw.cmd	arrange files	9 months ago
pm2.json	arrange files	9 months ago
pom.xml	arrange files	9 months ago

 **README**  

## javamysql-backend

This is a sample project of a Dockerized MySQL accessed by a simple backend with a CRUD functionality using Java SpringBoot technology.

 Readme  Activity  0 stars  2 watching  0 forks

**Releases**

No releases published  
[Create a new release](#)

**Packages**

No packages published  
[Publish your first package](#)

**Languages**

Java 97.1%

Dockerfile 2.9%

**Suggested workflows**

Based on your tech stack

# Hands-On Laboratorial

## Subindo um Container MySQL

```
$ docker network create --driver bridge javamysql-network  
$ docker container run -p 3306:3306 --name docker-mysql --network javamysql-network -e  
MYSQL_ROOT_PASSWORD=1q2w3e4r5t -e MYSQL_DATABASE=javamysqldb -d mysql:latest  
$ docker container exec -it docker-mysql bash
```

## Inserção de Registros

```
$ mysql -u root -p javamysqldb  
$ CREATE TABLE users ( id int NOT NULL,  
  first_name varchar(255) NOT NULL,  
  last_name varchar(255),  
  email varchar(255),  
  gender varchar(255)  
);  
$ INSERT INTO users (id, first_name, last_name, gender, email) VALUES (1, 'Teste', 'Teste', 'male',  
'teste@gmail.com');  
$ SELECT * FROM users;
```

# Hands-On Laboratorial

## Container do Backend JAVA

```
$ mvn -Dmaven.test.skip=true package
```

A linha de comando pode ser usada no build

```
$ docker build -t javamysql-backend:latest .
```

```
$ docker run -p 8080:8080 --name java-backend javamysql-backend:latest
```

Se necessário use para liberar a porta da aplicação (Windows)

```
$ netstat -ano | findstr :<porta>
```

```
$ taskkill /pid <PID> /F
```

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

## Simulando o Backend JAVA na rede

O Ngrok expõe serviços que estão sendo executados localmente para a Internet.

```
$ docker run -it -e NGROK_AUTH_TOKEN=<token> ngrok/ngrok http 8080
```

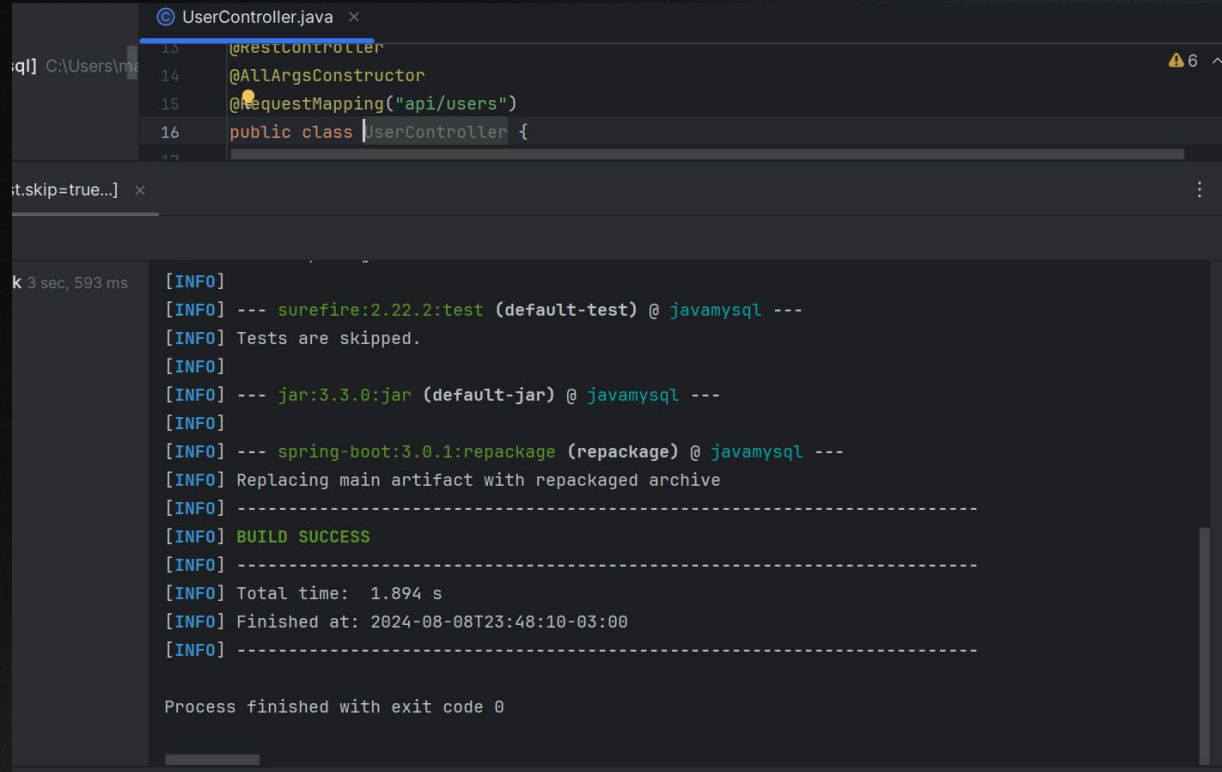
```
$ ngrok http 8080
```

— Acesse <forwarding\_url>/api/users





# Hands-On Laboratorial



The screenshot shows an IDE with two main panels. The top panel displays the code for `UserController.java`, which includes annotations like `@RestController`, `@AllArgsConstructor`, and `@RequestMapping("api/users")`, followed by the start of a `UserController` class. The bottom panel is a terminal window showing the output of a Maven build. The build process includes running tests (which are skipped), packaging a JAR, and repackaging the Spring Boot artifact. The final output indicates a successful build.

```
UserController.java x
13 @RestController
14 @AllArgsConstructor
15 @RequestMapping("api/users")
16 public class UserController {
17

it.skip=true...] x

k 3 sec, 593 ms [INFO]
[INFO] --- surefire:2.22.2:test (default-test) @ javamysql ---
[INFO] Tests are skipped.
[INFO]
[INFO] --- jar:3.3.0:jar (default-jar) @ javamysql ---
[INFO]
[INFO] --- spring-boot:3.0.1:repackage (repackage) @ javamysql ---
[INFO] Replacing main artifact with repackaged archive
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 1.894 s
[INFO] Finished at: 2024-08-08T23:48:10-03:00
[INFO] -----

Process finished with exit code 0
```

# Hands-On Laboratorial

```
(use "git add <file>..." to include in what will be committed)
.idea/
```

nothing added to commit but untracked files present (use "git add" to track)

```
C:\Users\marce\Desktop\Java-Advanced\docker-java-mysql>docker build -t javamysql-backend:latest .
```

```
[+] Building 2.6s (8/8) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile              0.0s
=> => transferring dockerfile: 208B                             0.0s
=> WARN: ConsistentInstructionCasing: Command 'expose' should match the cas 0.0s
=> [internal] load metadata for docker.io/library/openjdk:17-jdk-alpine 1.4s
=> [auth] library/openjdk:pull token for registry-1.docker.io     0.0s
=> [internal] load .dockerignore                                 0.0s
=> => transferring context: 2B                                    0.0s
=> [internal] load build context                                0.8s
=> => transferring context: 44.41MB                               0.8s
=> CACHED [1/2] FROM docker.io/library/openjdk:17-jdk-alpine@sha256:4b6abae 0.0s
=> [2/2] ADD target/javamysql-0.0.1-SNAPSHOT.jar javamysql-0.0.1-SNAPSHOT.j 0.2s
=> exporting to image                                           0.1s
=> => exporting layers                                           0.1s
=> => writing image sha256:eef9b6a4dbdc5258f62897d6c730bb826b90b0a5e3231e5c 0.0s
=> => naming to docker.io/library/javamysql-backend:latest     0.0s
```

```
1 warning found (use --debug to expand):
- ConsistentInstructionCasing: Command 'expose' should match the case of the comma
nd majority (uppercase) (line 3)
```

```
C:\Users\marce\Desktop\Java-Advanced\docker-java-mysql>
```

[illegible]



# Hands-On Laboratorial

## Edite, recompile, e republique os métodos em UserController.java

- Para devolver uma resposta com HTTP 5XX
- Para devolver uma resposta com HTTP 4XX
- Para devolver uma resposta com HTTP 3XX

## Tarefa para Casa...

Crie uma imagem de VMWare (ou sistema similar) e hospede seu projeto nela, para que seja possível abrir o ambiente de forma remota ou em laboratório.





# Referências

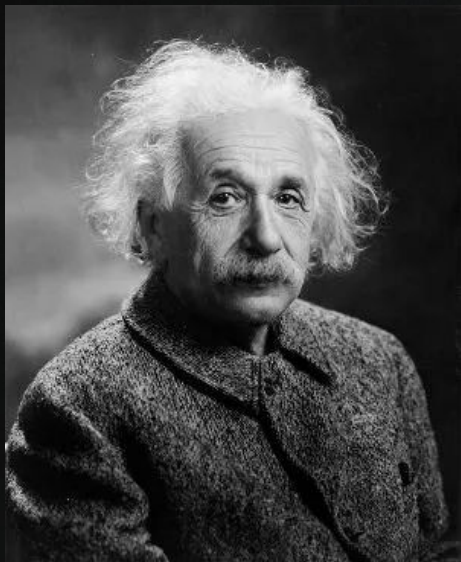
## Email do Professor

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

## Bibliografias/Sites

- <https://www.fiap.com.br/graduacao/tecnologo/analise-e-desenvolvimento-de-sistemas/>
- <https://www.jetbrains.com/idea/download/>
- <https://docs.spring.io>





*"A mente que se abre a uma  
nova ideia jamais voltará ao seu  
tamanho original"*

Albert Einstein

**FIM**

---