INSTITUTO FEDERAL
Triângulo Mineiro
Campus Uberlândia Centro

Projeto Backend Microsserviços e NoSQL

Primeiros Passos com MongoDB

Prof. Lucas Montanheiro
lucasmontanheiro@iftm.edu.br

Comandos para subir o MongoDB

- Copiar ubuntu-mongodb-remoto.zip do //10.10.90.130
- Extrair do zip
- Acrescentar VM no VirtualBox
- Configurar placa de rede como: Host-Only
- Usuário e senha: aluno
- Ao logar use os comandos:
  - sudo su
  - docker start mongodb
- Baixar Mongo Compass e colocar:
  - mongodb://192.168.56.101:27017/

Crie um novo Projeto

- **Project:** Maven
- **Language:** Java
- **Spring Boot:** 3.*.* (preferencia 3.1.*)
- **Group:** com.iftm
- **Artifact:** start-example
- **Name:** startexample
- **Description:** Primeiro exemplo de Spring Boot com MongoDB
- **Package name:** com.iftm.startexample
- **Packaging:** Jar
- **Java:** 17

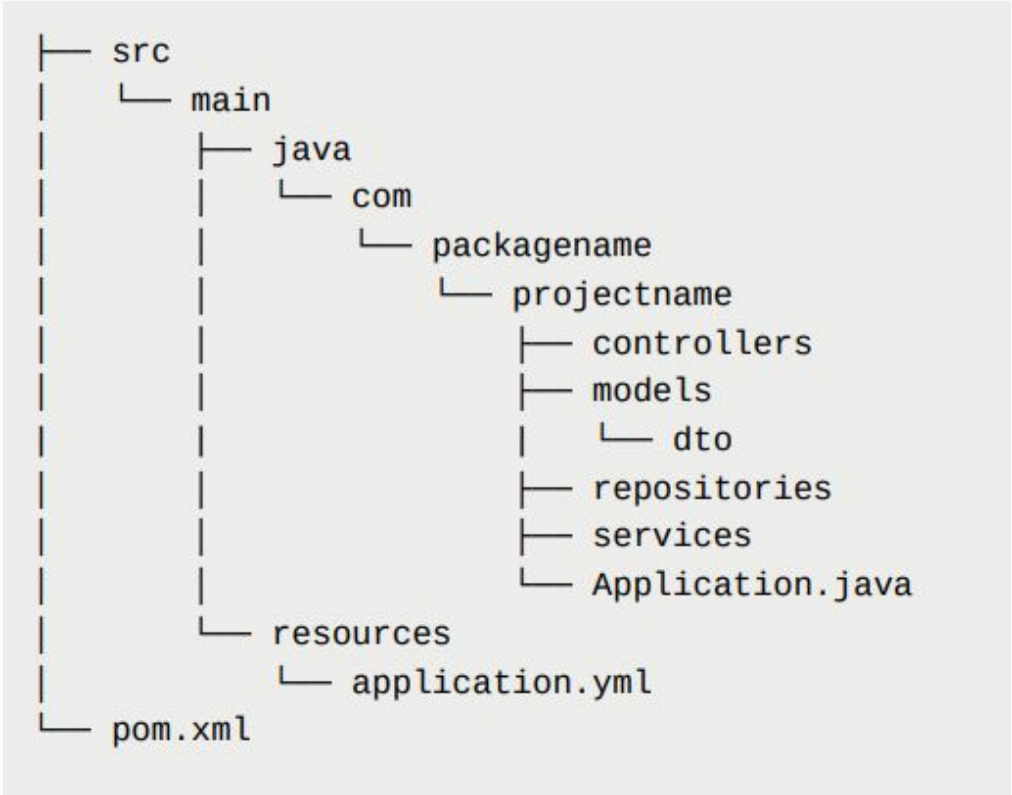
Adicione as Dependências Necessárias

- Spring Data MongoDB;
- Spring Web.

Estrutura de Dependências do POM:

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-mongodb</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>
```

Crie a Estrutura de Pastas



Padrão Maven

Implementação da Classe Principal (main)

```
package com.iftm.startexample;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.data.mongodb.repository.config.EnableMongoRepositories;

@SpringBootApplication
@EnableMongoRepositories
public class StartExampleApplication {

    public static void main(String[] args) {
        SpringApplication.run(StartExampleApplication.class, args);
    }
}
```

@EnableMongoRepositories, faz a importação do Spring Boot Data com MongoDB.

Implementação da Classe User

Local:

```
└── src
    └── main
        └── java
            └── com
                └── packagename
                    └── projectname
                        └── models
                            └── User.java
```

Implementação da Classe User

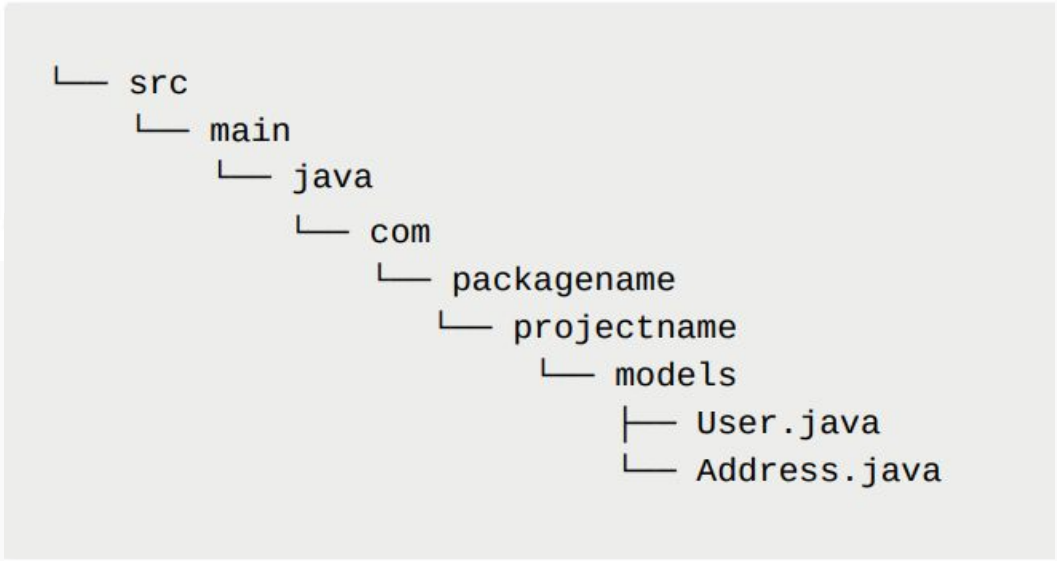
Código:

```
@Document(collection = "user")
public class User {

    @Id //@MongoId ou @MongoId(targetType = FieldType.OBJECT_ID) ou @BsonId
    private ObjectId id;
    @Field("name")
    private String name;
    private int age;
    private Address address;

    public User() {
    }

    public User(String name, int age, Address address) {
        this.name = name;
        this.age = age;
        this.address = address;
    }
    // Getters and Setters
    // equals e hashCode
    // toStrint
}
```

Podem gerar Getters e Setters ou utilizar o Lombok

Implementação da Classe Address

Local:



Implementação da Classe Address

src > main > java > com > iftm > startexample > models > Address.java > {} com.iftm.startexample.models

```java
package com.iftm.startexample.models;

import java.util.Objects;

public class Address {
    private String street;
    private int number;

    public Address() {
    }

    public Address(String street, int number) {
        this.street = street;
        this.number = number;
    }
```

Implementação da Classe UserDTO

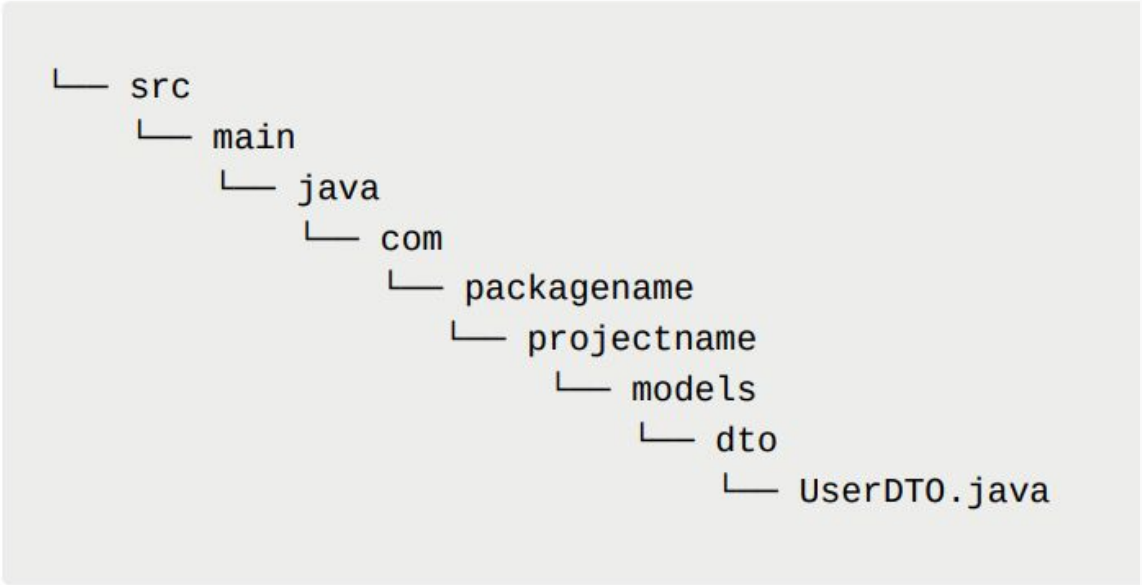
Código:

```
@Document(collection = "user")
public class User {

    @Id //@MongoId ou @MongoId(targetType = FieldType.OBJECT_ID) ou @BsonId
    private ObjectId id;
    @Field("name")
    private String name;
    private int age;
    private Address address;

    public User() {
    }

    public User(String name, int age, Address address) {
        this.name = name;
        this.age = age;
        this.address = address;
    }
    // Getters and Setters
    // equals e hashCode
    // toStrint
}
```

Podem gerar Getters e Setters ou utilizar o Lombok

Implementação da Classe UserDTO

Local:



Implementação da Classe UserDTO

Código:

```
public class UserDTO implements Serializable {
    private String id;
    private String name;
    private int age;
    private Address address;

    public UserDTO() {
    }

    public UserDTO(User user) {
        this.id = user.getId().toString();
        this.name = user.getName();
        this.age = user.getAge();
        this.address = user.getAddress();
    }

    public UserDTO(String name, int age) {
        this.name = name;
        this.age = age;
    }
    // Getters and Setters
    // equals e hashCode
    // toStrint
}
```

Podem gerar Getters e Setters ou utilizar o Lombok

Implementação da Interface UserRepository

Local:

```
└── src
    └── main
        └── java
            └── com
                └── packagename
                    └── projectname
                        └── repositories
                            └── UserRepository.java
```

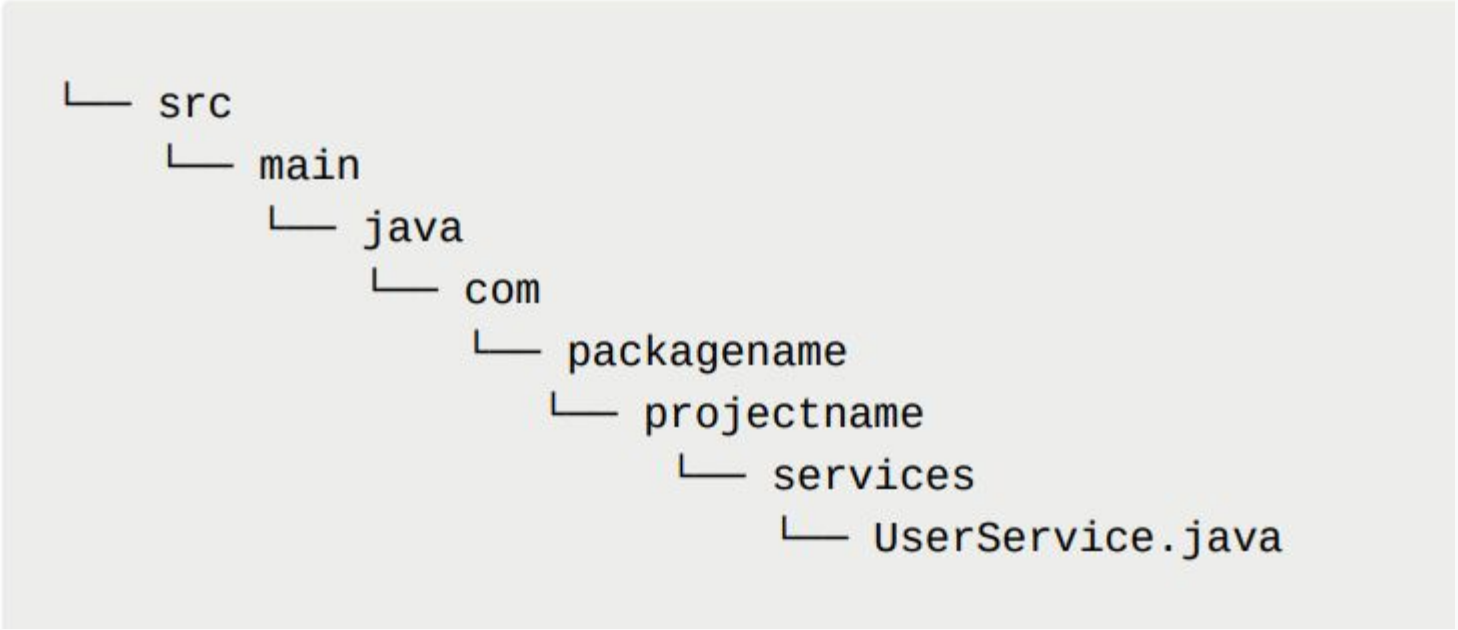
Implementação da Interface UserRepository

Código:

```
package com.iftm.startexample.repositories;

import com.iftm.startexample.models.User;
import org.bson.types.ObjectId;
import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface UserRepository extends MongoRepository<User, ObjectId> {
}
```

Implementação da Classe UserService

Local:



Implementação da Classe UserService

Código:

```
public class UserService {

    @Autowired
    private UserRepository repository;

    public ResponseEntity<List<UserDTO>> findAll() {
        var dbUsers = repository.findAll();
        if(dbUsers.isEmpty())
            return ResponseEntity.notFound().build();

        var userDtos = dbUsers.stream().map(user -> {
            var userDTO = new UserDTO(user);
            return userDTO;
        }).collect(Collectors.toList());

        return ResponseEntity.ok(userDtos);
    }

    public ResponseEntity<UserDTO> findById(ObjectId id) {
        if(id == null)
            return ResponseEntity.badRequest().build();
        var dbUser = repository.findById(id);
```

Implementação da Classe UserService

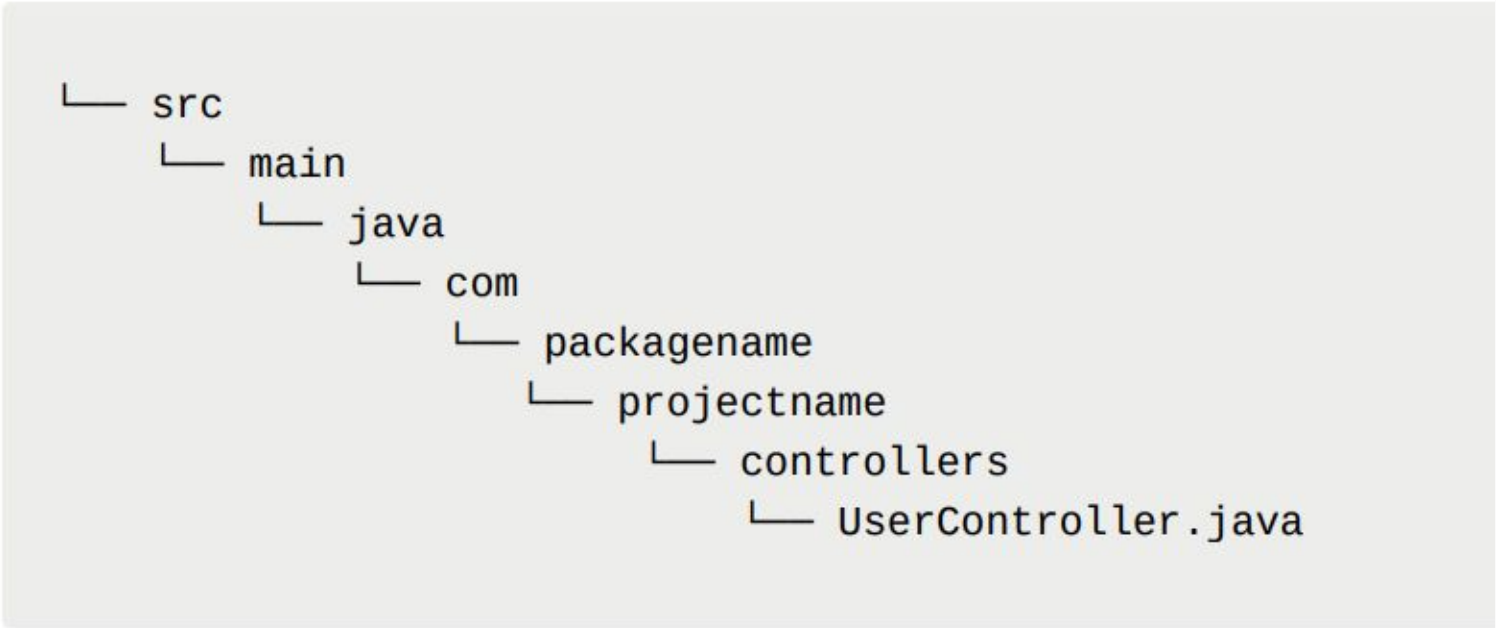
```
        if(dbUser.isEmpty())
            return ResponseEntity.notFound().build();
        return ResponseEntity.ok(new UserDTO( dbUser.get()));
    }

    public ResponseEntity<UserDTO> save(User user) {
        //validar user
        if(user.getName().isBlank() || user.getAge() <= 0)
            return ResponseEntity.badRequest().build();
        user.setId(ObjectId.get());
        return ResponseEntity.ok(new UserDTO(repository.save(user)));
    }

    public ResponseEntity<UserDTO> update(UserDTO user) {
        // validar user
        if(user.getId() == null)
            return ResponseEntity.badRequest().build();

        var objectId = new ObjectId(user.getId());
        var dbUser = repository.findById(objectId);
        if(dbUser.isEmpty())
            return ResponseEntity.notFound().build();
        // atualizar
        var dbUserObj = dbUser.get();
        dbUserObj.setName(user.getName());
        dbUserObj.setAge(user.getAge());
        return ResponseEntity.ok(new UserDTO(repository.save(dbUserObj)));
    }
```

Implementação da Classe UserService

```
    public ResponseEntity<?> delete(ObjectId id) {
        if(id != null)
            return ResponseEntity.badRequest().build();
        repository.deleteById(id);

        var dbUser = repository.findById(id);
        if(dbUser.isPresent())
            return ResponseEntity.status(HttpStatus.NOT_MODIFIED).build();
        return ResponseEntity.ok().build();
    }
}
```

Implementação da Classe UserController

Local:



Implementação da Classe UserController

Código:

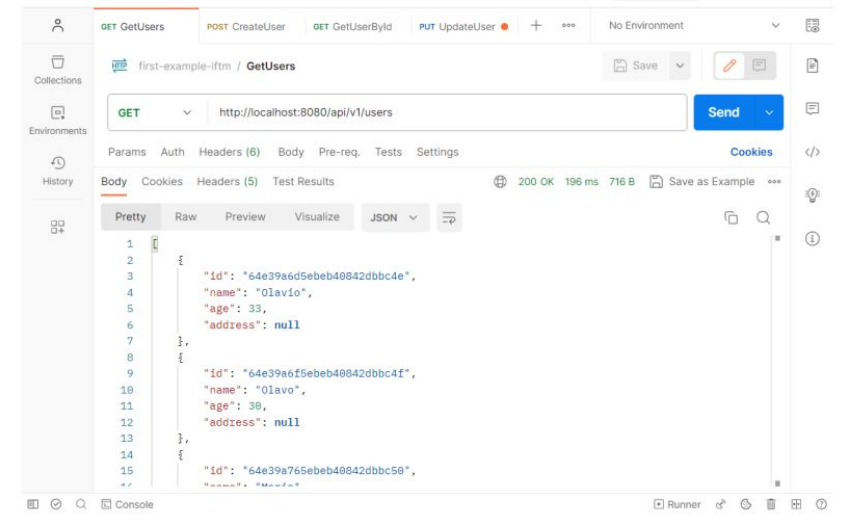
```
@RestController
@RequestMapping("api/v1/users")
public class UserController {

    @Autowired
    private UserService service;

    @GetMapping
    public ResponseEntity<List<UserDTO>> findAll() {
        return service.findAll();
    }

    @GetMapping("/id/{id}")
    public ResponseEntity<UserDTO> findById(@PathVariable("id") ObjectId id) {
        return service.findById(id);
    }

    @PostMapping
    public ResponseEntity<UserDTO> create(@RequestBody User user) {
        return service.save(user);
    }
```

Implementação da Classe UserController

```
    @PutMapping
    public ResponseEntity<UserDTO> update(@RequestBody UserDTO user) {
        return service.update(user);
    }

    @DeleteMapping("/id/{id}")
    public ResponseEntity<?> delete(@PathVariable("id") ObjectId id) {
        return service.delete(id);
    }
}
```

Objetivo de Hoje

- Terminar a implementação da aplicação e executar os testes da aplicação com o Postman, disponível no projeto.
- https://github.com/Montanheiro/projeto-ms-java-mongodb-iftm2024



Objetivo de Hoje

- Terminar a implementação da aplicação e executar os testes da aplicação com o Postman, disponível no projeto.
- Rotas a serem testadas:
  - GET /users
  - GET /users/id/[ID AQUI]
  - POST /users
  - PUT /users
  - DELETE /users/id/[ID AQUI]