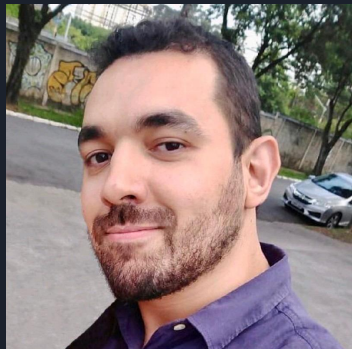




**EVOLUTIONTECH**



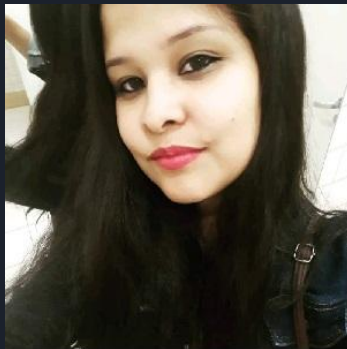
# Quem somos:



Felipe Lima



Karla Cracco



Kelly Michele



Maycon Mota



Michel Fernandes



# Visão Geral e Objetivos

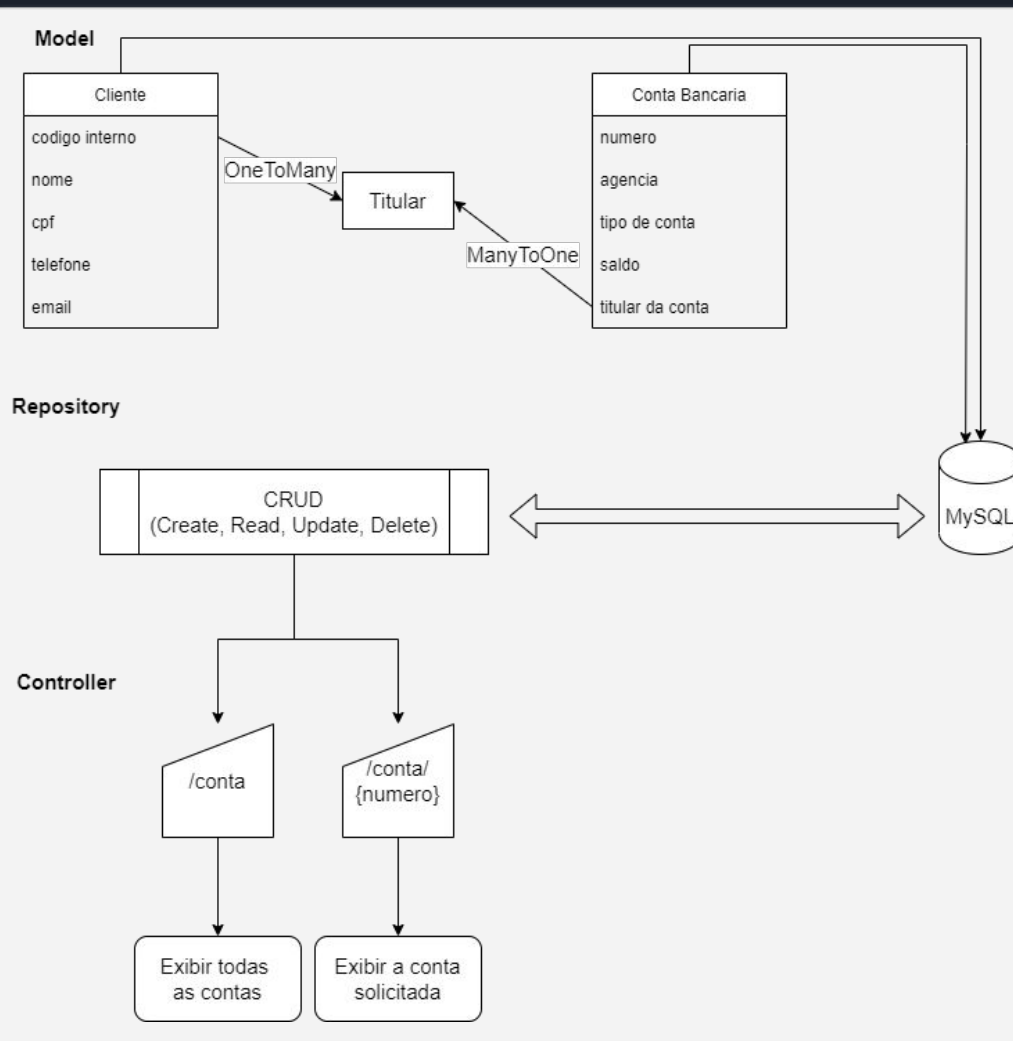
Estamos em um contexto de gerenciamento de clientes e contas bancárias da instituição.

Precisamos disponibilizar uma API que forneça dados de clientes e contas (inicialmente para consulta).

**Construir uma API Spring BOOT que possua 2 endpoints principais:**

- a. Recuperação de todas as contas bancárias
- b. Recuperação dos detalhes de 1 conta bancária (inclusive dados do seu titular)

# Projeto



# Criando variáveis e regras da Cliente

```
@Entity // Indica que esta classe será armazenada no Banco de Dados
@Table(name = "tb_cliente")
public class Cliente {

    @Id // indica que este campo é chave primario
    @GeneratedValue (strategy = GenerationType.IDENTITY) // 1,2,3.....
    private int codigo;

    @Column (name = "nome", length = 120, nullable = false)
    private String nome;

    @Column (length = 20)
    private String cpf;

    @Column (length = 20)
    private String telefone;

    @Column (length = 40, nullable = false, unique = true) // unique: não pode ter 2 emails iguais
    private String email;

    @OneToMany (mappedBy = "titular")
    @JsonIgnoreProperties("titular")
    private List<Conta> contas;
```

Information .....

Table: **tb\_cliente**

Columns:

<b>codigo</b>	int AI PK
cpf	varchar(20)
<b>email</b>	varchar(40)
nome	varchar(120)
telefone	varchar(20)

# Criando variáveis e regras da Conta

```
@Entity
@Table(name = "tb_conta")
public class Conta {

    @Id
    @GeneratedValue (strategy = GenerationType.IDENTITY)
    private int numero;

    @Column (length = 5, nullable = false)
    private int agencia;

    @Column (length = 1, nullable = false)
    private int tipoConta;

    private double saldo;

    @ManyToOne
    @JoinColumn(name = "cliente_cod")
    @JsonIgnoreProperties("contas")
    private Cliente titular;
```

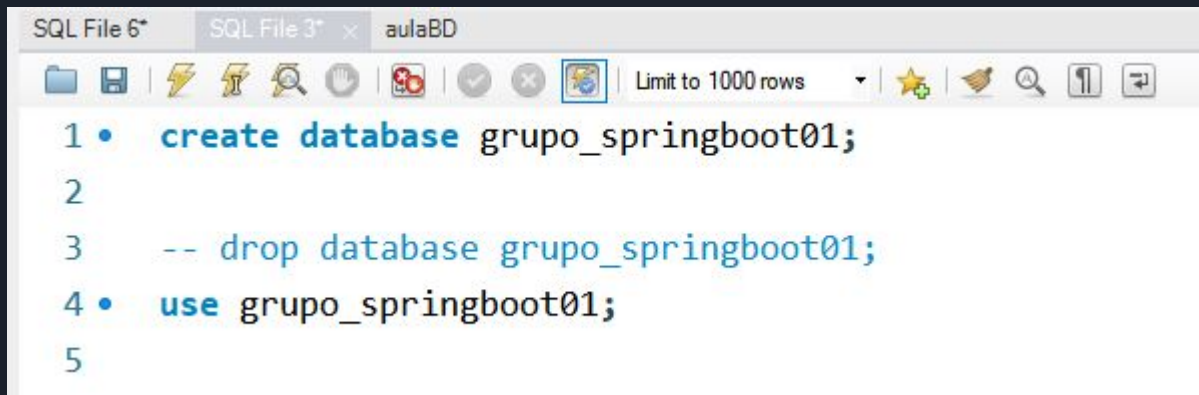
Information

Table: **tb\_conta**

Columns:

<b><u>numero</u></b>	int AI PK
agencia	int
saldo	double
tipo_conta	int
<b>cliente_cod</b>	int

# MySQL



The screenshot shows a MySQL IDE window with three tabs: 'SQL File 6\*', 'SQL File 3\*', and 'aulaBD'. The 'aulaBD' tab is active. The toolbar includes icons for file operations (folder, save, lightning bolt, copy, paste, search, hand), execution (play, stop, refresh), and viewing (limit to 1000 rows, star, brush, zoom, split, and refresh). The SQL editor contains the following code:

```
1 • create database grupo_springboot01;  
2  
3 -- drop database grupo_springboot01;  
4 • use grupo_springboot01;  
5
```

# MySQL

```
6      desc tb_cliente;
7 •  select * from tb_cliente;
8      -- Criar dados de Cliente (Felipe, Karla, Maycon, Michel e Kelly)
9 •  insert into tb_cliente values (null, "123.123.123-18", "felipaa@gmail.com", "Felipe Lima de Oliveira", "(11)12345-6789");
10 • insert into tb_cliente values (null, "987.654.321-12", "karla_cracco@hotmail.com", "Karla Rojas Cracco Lima Imperio Dalmati", "(11)9876-6789");
11 • insert into tb_cliente values (null, "654.987.321-90", "mayconmotamendes@gmail.com", "Maycon Duglas Mendes Mota", "(11)12345-6789");
12 • insert into tb_cliente values (null, "321.987.654-78", "mwesleyfernandes15071986@yahoo.com.br", "Michel Wesley Lima Fernandes", "(11)12345-6789");
13 • insert into tb_cliente values (null, "963.852.714-89", "kellymichelesaavedra@yahoo.com.br", "Kelly Michele Torrico Saavedra dos Santos", "(11)12345-6789");
14
15 • select * from tb_conta;
16      -- Criar dados de Conta (0 - Conta Corrente, 1 - Conta Poupança, 2 - Conta Investimento)
17 • insert into tb_conta values (null, 12345, 750.00, 0, 1);
18 • insert into tb_conta values (null, 10058, 1000.00, 1, 2);
19
20
```



# Codando no Controller



POSTMAN

```
16 @RestController
17 @CrossOrigin("*")
18 @RequestMapping("/conta")
19 public class ContaController {
20
21     @Autowired
22     private ContaRepo repo;
23
24     @GetMapping
25     public List<Conta> listarTodos() {
26         return (List<Conta>) repo.findAll();
27     }
28
29     @GetMapping("/numero/{numero}")
30     public ResponseEntity<Conta> buscarConta(@PathVariable int numero){
31         Conta conta = repo.findById(numero).orElse(null);
32
33         if (conta != null) {
34             return ResponseEntity.ok(conta);
35         }
36         return ResponseEntity.notFound().build();
37     }
38 }
```



GET	▼	http://localhost:8080/conta
-----	---	-----------------------------



GET	▼	http://localhost:8080/conta/numero/{numero}
-----	---	---



Obrigado!