SEMINÁRIO DE TECNOLOGIAS

LAB 03 - SPRINT 03

Gabriel Henrique Mota Rodrigues
Matheus Vinicius Mota Rodrigues
Pedro Luis Gonçalves

01

ARQUITETURA

02

PERSISTÊNCIA

03

BENEFÍCIOS

ARQUITETURA MVC

ARQUITETURA MVC

A arquitetura MVC (Model-View-Controller) é um padrão de design usado para estruturar aplicações de forma modular, separando suas responsabilidades em três componentes principais:

- **Model:** Responsável por gerenciar os dados, regras de negócio e a lógica da aplicação.
- **View:** Cuida da apresentação da informação ao usuário, recebendo os dados do model e exibindo de forma adequada.
- **Controller:** Atua como intermediário entre o Model e a View. Recebe as requisições do usuário (geralmente por meio da interface de usuário) e decide o que fazer.

Controller:

```java
public class AlunoController {

    @Autowired
    private AlunoService alunoService;

    @GetMapping
    public List<Aluno> getAllAlunos() {
        return alunoService.getAllAlunos();
    }

    @GetMapping("/{id}")
    public Optional<Aluno> getAlunoById(@PathVariable Long id) {
        return alunoService.getAlunoById(id);
    }

    @PostMapping
    public Aluno createAluno(@RequestBody Aluno aluno) {
        return alunoService.saveAluno(aluno);
    }

    @PutMapping("/{id}")
    public Aluno updateAluno(@PathVariable Long id, @RequestBody Aluno aluno) {
        aluno.setId(id);
        return alunoService.saveAluno(aluno);
    }

    @DeleteMapping("/{id}")
    public void deleteAluno(@PathVariable Long id) {
        alunoService.deleteAluno(id);
    }

}
```

Model:

```java
public class Aluno implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nome;
    private String email;
    private String cpf;
    private String rg;
    private String endereco;

    @ManyToOne
    @JoinColumn(name = "instituicao_id")
    private Instituicao instituicao;

    private String curso;
    private int saldo;

    @OneToMany(mappedBy = "aluno", cascade = CascadeType.ALL)
    private List<Transacao> extrato = new ArrayList<>();

    public Aluno() {
    }

    public Aluno(String nome, String email, String cpf, String rg, String endereco,
Instituicao instituicao,
                 String curso) {
        this.nome = nome;
        this.email = email;
        this.cpf = cpf;
        this.rg = rg;
        this.endereco = endereco;
        this.instituicao = instituicao;
        this.curso = curso;
        this.saldo = 0;
    }
}
```

PERSISTÊNCIA

ORM com Spring Data JPA

ORM (Object-Relational Mapping) é uma técnica de programação que permite a conversão entre dados de uma base de dados relacional e objetos de programação orientada a objetos.

- Mapeia classes a tabelas de banco de dados e vice-versa;
- Facilitando a manipulação dos dados de forma mais intuitiva e menos propenso a erros.

Repositórios

Interfaces que estendem JpaRepository para realizar operações de CRUD

```java
@Repository
public interface EmpresaParceiraRepository extends
JpaRepository<EmpresaParceira, Long> {



}
```

Serviços

Os serviços contêm a lógica de negócios e utilizam os repositórios para interagir com o banco de dados.

```java
public class EmpresaParceiraService {

    @Autowired
       private EmpresaParceiraRepository empresaParceiraRepository;

       public List<EmpresaParceira> getAllEmpresas() {
           return empresaParceiraRepository.findAll();
       }

       public Optional<EmpresaParceira> getEmpresaById(Long id) {
           return empresaParceiraRepository.findById(id);
       }

       public EmpresaParceira saveEmpresa(EmpresaParceira empresaParceira) {
           return empresaParceiraRepository.save(empresaParceira);
       }

       public void deleteEmpresa(Long id) {
           empresaParceiraRepository.deleteById(id);
       }

}
```

BENEFÍCIOS

Benefícios do ORM

ABSTRAÇÃO

Permite interagir com o banco de dados usando objetos e classes da linguagem de programação

PRODUTIVIDADE

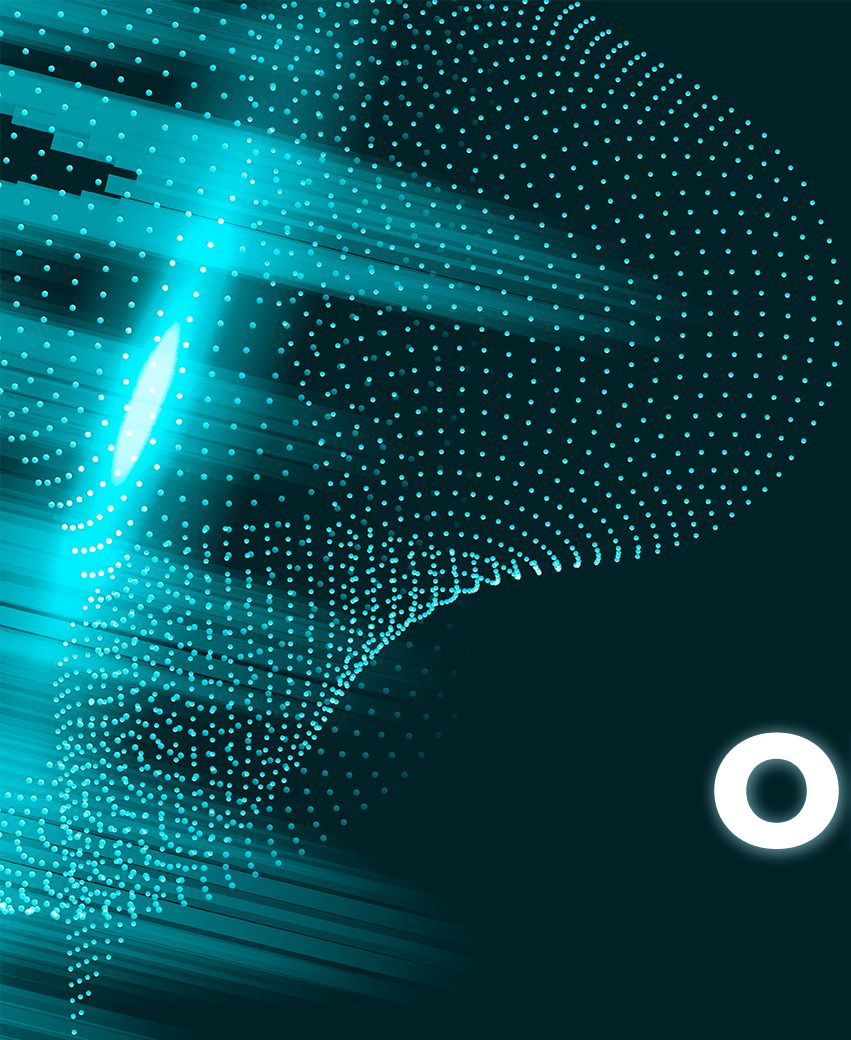
Um ORM simplifica o trabalho com bancos de dados

MANUTENÇÃO E EVOLUÇÃO

Alterações na estrutura de dados podem ser feitas no código da aplicação, e o ORM sincroniza essas mudanças com o banco de dados.

PORTABILIDADE

Usando um ORM, a aplicação se torna menos dependente de um banco de dados específico.



OBRIGADO!