

---

**Bernardo Alvim, Joaquim Vilela e Marcos Alberto**

# **SISTEMA DEMOEDA ESTUDANTIL**

Laboratório de Projetos de Software

---

# Frontend Development

## REACT

- Alta performance com DOM virtual
- Reutilização de componentes
- Fácil integração com APIs
- Permite criação de UI moderna, responsiva e fluida



# JAVASCRIPT

- Compatível com todos os navegadores
  - Permite criar interfaces dinâmicas e reativas
  - Sinergia total com React
  - Permite realizar validações e feedback ao usuário em tempo real

# AXIOS

- Facilita requisições ao backend
  - Tratamento claro de erros e respostas
  - Melhor legibilidade e organização das chamadas API do que fetch nativo

```
"container">
  <div class="row">
    <div class="col-md-6 col-lg-8"> <!--
      <nav id="nav" role="navigation">
        <ul>
          <li><a href="index.html">Home</a>
          <li><a href="home-events.html">Home Events</a>
          <li><a href="multi-col-menu.html">Multi Col Menu</a>
          <li class="has-children"> <a href="#">More Options</a>
            <ul>
              <li><a href="tall-button-height.html">Tall Button Height</a>
              <li><a href="image-logo.html">Image Logo</a>
              <li class="active"><a href="#">Variable Width</a>
            </ul>
          </li>
          <li class="has-children"> <a href="#">More Options</a>
            <ul>
              <li><a href="variable-width.html">Variable Width</a>
            </ul>
          </li>
        </ul>
      </nav>
    </div>
  </div>

```

# Backend Development

## SPRING BOOT

- Estrutura pronta para CRUDs, autenticação e segurança
- Configuração simplificada e produtiva
- Suporte nativo para APIs REST
- Integrado ao Spring Security para login/autenticação
- Ótimo desempenho para sistemas escaláveis



# JAVA

- Forte tipagem e segurança em produção
- Confiável para sistemas escaláveis e críticos
- Ecossistema rico e bem documentado
- Fácil manutenção e alta legibilidade

## PADRÃO REPOSITORY

- Abstrai o acesso ao banco de dados
- Centraliza todas as operações de persistência (consultar, salvar, atualizar e remover dados).
- Isola a lógica de acesso aos dados da lógica de negócio

```
    resp_iter = self.stub.GetStatuses()
    statuses = {}
    for data in resp_iter:
        status = Status(
            status_id=data.id, name=data.name,
            description=data.description)
        statuses[status.name] = status
    return statuses
```

# Repository, Service, Controller, Models

Com a divisão em Controller, Service e Repository, o sistema fica organizado, fácil de manter e pronto para crescer

Proporciona uma base sólida para o gerenciamento da moeda virtual, distribuição de recompensas e integração entre usuários e parceiros.



# REPOSITORY

- Abstração de queries SQL
- Reduz erros e agiliza desenvolvimento

# SERVICE

- Separação clara da lógica de negócio
- Código mais limpo e reutilizável

# CONTROLLER

- Organização das rotas e entradas do sistema
- Mantém camada de controle objetiva

# MODELS (ENTIDADES)

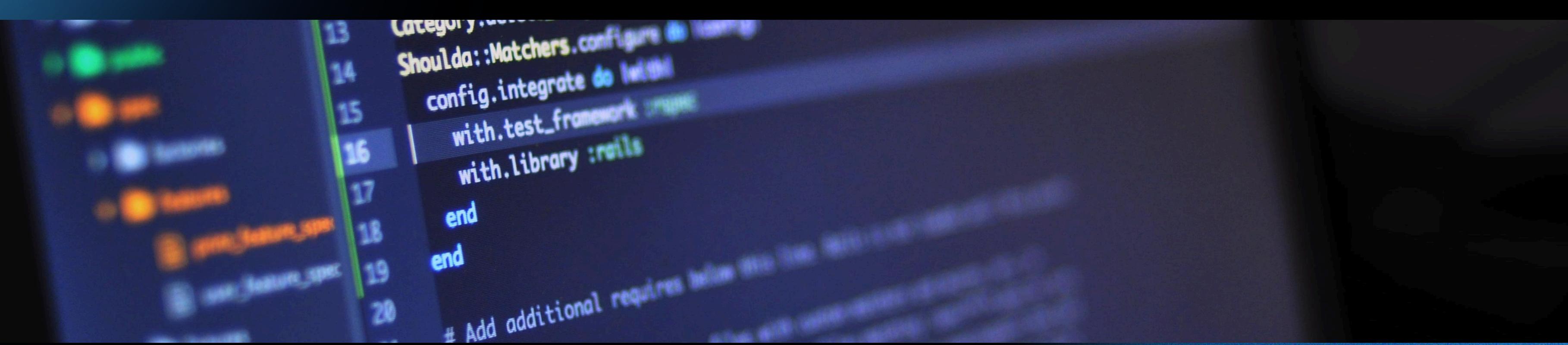
- Organização clara dos dados
- Facilita manipulação e persistência



# MYSQL

- Alta performance para operações transacionais (CRUD)
- Estrutura relacional ideal para manter integridade dos dados (ex.: alunos, professores, moedas, transações)
- Integração nativa com Spring Data JPA, facilitando persistência e consultas

# BANCO DE DADOS



# Conclusão

## Integração das Tecnologias

### React + JavaScript + Axios

- UI rápida e moderna
- Consumo eficiente da API

### Spring Boot + Java

- Backend seguro e escalável
- Regras de negócio e APIs REST

### Arquitetura (Models / Repository / Service / Controller)

- Código organizado
- Fácil manutenção e evolução

### MySQL

- Dados consistentes
- Relacionamentos e histórico seguro

```
17 string sInput;
18 int iLength, iN;
19 double dblTemp;
20 bool again = true;
21 while (again) {
22     iN = -1;
23     again = false;
24     getline(cin, sInput);
25     system("cls");
26     stringstream(sInput) >> dblTemp;
27     iLength = sInput.length();
28     if (iLength < 4) {
29         again = true;
30         continue;
31     } else if (sInput[iLength - 3] != '.') {
32         again = true;
33         continue;
34     } else if (++iN < iLength) {
35         if (sInput[iN]) {
36             again = true;
37             continue;
38         }
39     }
40 }
```

OB  
RIGADO