# **Sistema de Estágio**

## **Membros:**

Abner Castanho Cardoso - **RA:**

Gabriel Phelipe Bernardo Coronado - **RA:**

Guilherme Tófoli da Silva - **RA:** 24098313-2

Matheus Henrique Coronado Elias - **RA:** 24154884-2

### 

### 

### 

### 

### **1. Visão Geral: Arquitetura em Camadas + Microsserviços**

● **Frontend (Camada de Apresentação):** Aplicação Web responsiva (pode ser desenvolvida com React, Angular ou Vue.js).  
  
Permite acesso aos usuários via navegador: alunos, gestores, RH, instituições.  
  
Comunicação com a API por meio de chamadas RESTful ou GraphQL.

● **Backend (Camada de Lógica de Negócio):**API construída em Node.js, Spring Boot (Java), Django (Python) ou ASP.NET Core (C#).

Dividida em **microsserviços**, cada um responsável por um domínio funcional, por exemplo:

* Autenticação e Autorização
* Gestão de Usuários
* Gestão de Instituições
* Gestão de Estágios
* Notificações (e-mail)

Cada serviço se comunica via API Gateway e utiliza mensagens assíncronas quando necessário (por exemplo, via RabbitMQ ou Kafka).

● **Banco de Dados (Camada de Persistência):**Banco de dados relacional (como PostgreSQL ou MySQL) para garantir integridade e consistência dos dados.  
  
Redis ou MongoDB pode ser usado como apoio para cache ou armazenamento temporário de formulários ou logs.

#### **1.2** **Serviços de Suporte: Serviço de Notificações**: envio automatizado de e-mails via fila. **Serviço de Logs e Auditoria**: para rastrear ações no sistema. **Serviço de Segurança**: controle de acesso, criptografia, autenticação (JWT + OAuth2).

### **2. Tecnologias Recomendadas**

| **Camada** | **Tecnologia Sugerida** |
| --- | --- |
| Frontend | React.js + Tailwind + Axios |
| Backend (API) | Node.js com Express / NestJS |
| Banco de Dados | PostgreSQL (relacional) |
| Autenticação | JWT + OAuth2 |
| Mensageria (opcional) | RabbitMQ / Kafka |
| Cache (opcional) | Redis |
| Hospedagem | Docker + Kubernetes (ou Docker Compose no MVP) |
| Deploy/Infra | AWS / Azure / Render / Railway |

### **3. Segurança**

* Criptografia de senhas com bcrypt.
* Controle de acesso por níveis de permissão.
* Autenticação com tokens JWT.
* HTTPS obrigatório.
* Política de acesso restrita aos dados sensíveis (como CPF, e-mail, documentos).

### **4. Escalabilidade e Manutenção**

* Microsserviços permitem escalar partes específicas (por exemplo, o serviço de notificações).
* Isolamento de falhas.
* Facilidade para adicionar novas funcionalidades.
* CI/CD pode ser integrado com GitHub Actions ou GitLab CI.