

# Attendance System



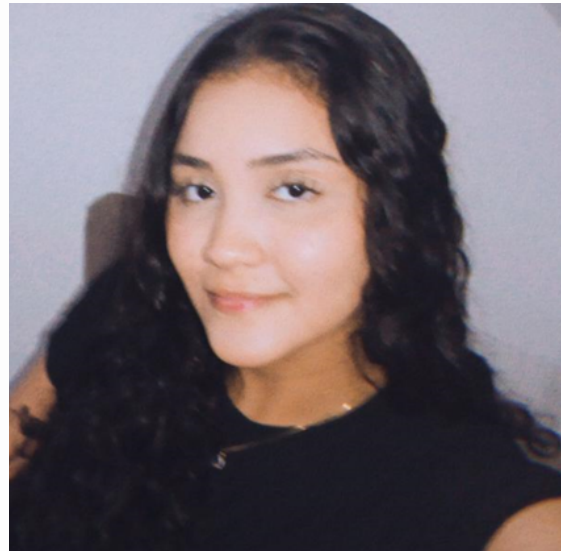
# Equipe



**André Filipe**  
Desenvolvedor



**Ayrton**  
Desenvolvedor



**Amanda**  
Documentador



**Ariano**  
Documentador



**Estephani**  
Scrum Master



**Manuela**  
Gerente

# Sumário


▶▶▶	Introdução	04
▶▶▶	EAP	05
▶▶▶	Custos e investimentos	07
▶▶▶	Cronograma	08
▶▶▶	Trello	09
▶▶▶	Repositório	10
▶▶▶	Casos de uso	11
▶▶▶	Protótipo	14
▶▶▶	Extratos do código	15
▶▶▶	Demonstração de uso	16
▶▶▶	Lições aprendidas	17
▶▶▶	Referências	18
▶▶▶	Agradecimentos	19



# Introdução ao Attendance System

O Attendance System foi desenvolvido para simplificar a gestão e acompanhamento da frequência de estudantes em instituições de ensino. Ele permite registrar, monitorar e organizar a presença de forma prática e eficiente, garantindo transparência e facilidade no processo educacional.

Seu objetivo é simplificar o trabalho de professores e administradores ao oferecer uma ferramenta centralizada para o controle da frequência, além de possibilitar que os alunos acompanhem seu próprio histórico de presenças. Com isso, o sistema contribui diretamente para uma gestão educacional mais eficiente e conectada.



# Estrutura Analítica do Projeto

## Nível 1

### Planejamento e Gerenciamento

Definição de escopo e configuração do ambiente.

### Desenvolvimento Técnico

Codificação, integração com PostgreSQL e APIs REST.

### Qualidade e Validação

Testes de funcionalidade, conexão e APIs.

### Implantação e Suporte

Hospedagem no Railway, documentação e suporte inicial.

# Estrutura Analítica do Projeto

## Nível 2

### Planejamento

Definição do escopo, configuração do ambiente e banco de dados.

### Desenvolvimento

Criação de entidades, APIs REST e integração com banco.

### Testes e Validação

Testes unitários, integração e validação de rotas.

### Implantação e Documentação

Hospedagem no Railway, documentação e acesso público.

# Custos e investimentos

Categoria de Custo	Descrição	Data de pagamento	Responsável	Observação	Valor
Recursos Humanos					
Salário da equipe	Custos com salários dos desenvolvedores, documentadores, Scrum Master, etc.	Mensal	Gerente de Projeto	Estimativa mensal, incluindo benefícios.	R\$ 15.000,00
Equipamentos					
Equipamentos para desenvolvimento	Compra de equipamentos de hardware, como servidores, etc.	inicial	Desenvolvedor		R\$ 3.500,00
Material de Escritório	Custos com materiais de escritório (papel, caneta, etc.)	Mensal	Todos os membros		R\$ 150,00
Infraestrutura					
Servidor Railway	Custo com a hospedagem do sistema no Railway	Mensal	Gerente de TI		R\$ 200,00
Banco de Dados PostgreSQL	PostgreSQL no Railway ou serviço externo	Mensal	Gerente de TI		R\$ 400,00
Licenças e Ferramentas					
Licença do Spring Boot (open-source)	Utilização do framework Spring Boot (open-source)	Não se aplica	Desenvolvedor	Não há custo, pois é open-source.	R\$ -
IDE e Ferramentas de Desenvolvimento	Licença de IDE (Ex: IntelliJ, Eclipse) e ferramentas adicionais de desenvolvimento	inicial	Desenvolvedor	Licenças para as ferramentas usadas.	R\$ 500,00
Treinamento e Capacitação					
Capacitação e cursos	Capacitação da equipe sobre o uso do Spring Boot, PostgreSQL, e boas práticas de desenvolvimento	inicial	Gerente de Projeto	Cursos para a equipe.	R\$ 800,00
Total de custos:					R\$ 20.550,00

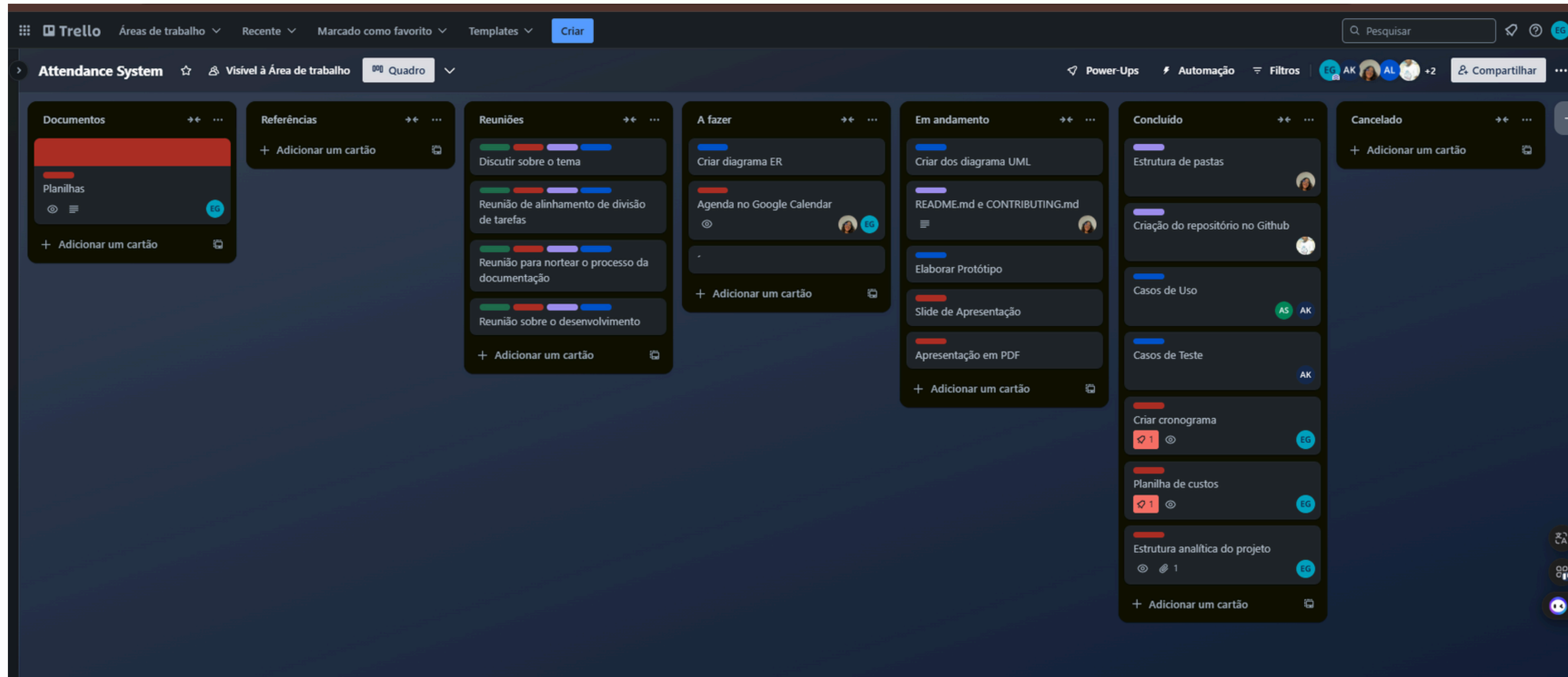


# Cronograma com Gantt

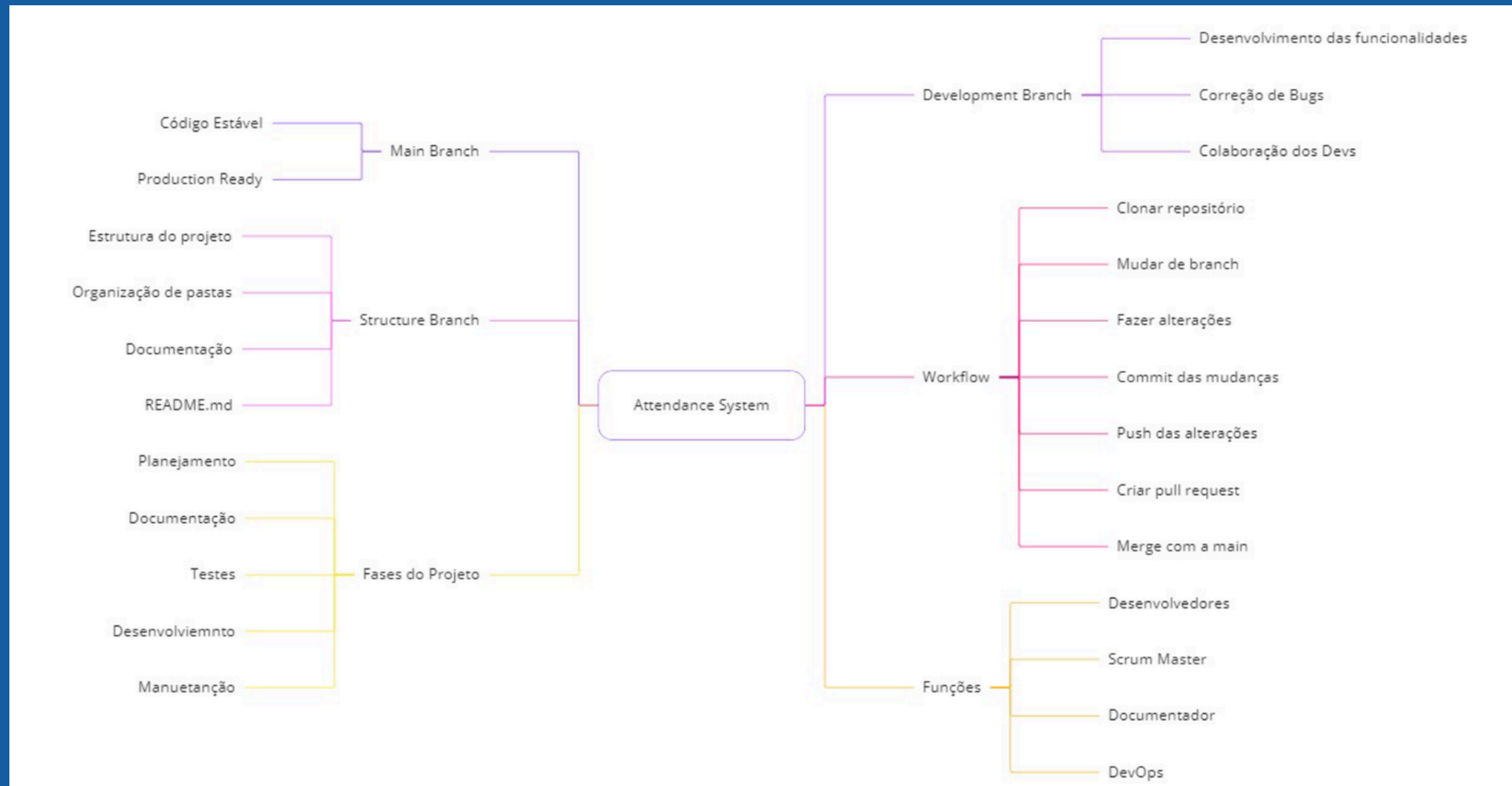
Etapas	11/10 - 17/10	18/10 - 24/10	25/10 - 31/10	01/11 - 07/11	08/11 - 14/11	15/11 - 21/11	22/11 - 28/11	29/11 - 05/12	06/dez	Responsável
Definição do tema										Equipe
Pesquisa										Analista de dados e negócio
Indexadores										Scrum Master, Analista de dados e negócio
Leitura e extração										Analista de dados e negócio
Prototipação										Gerente do Projeto e Designer
Desenvolvimento										Desenvolvedor
Escrita/ Documentação										Documentadores
Construção dos slides										Scrum Master
Repositório Git										Gerente de configuração
Validação com time e ensaio										Equipe
Apresentação										Equipe
Revisão geral										Equipe
Relatório										Analista de dados e negócio



# Trello



# Fluxo de trabalho no repositório



# Casos de Uso

01

UC001 – Realizar Login no Sistema

Atores: Administrador

Objetivo: Permitir que o usuário entre no sistema com credenciais válidas.

- Fluxo Principal: Usuário informa credenciais -> Sistema valida -> Concede acesso.
- Fluxos Alternativos: Exibe erro em caso de credenciais inválidas ou conta bloqueada.
- Fluxo Opcional: Usuário pode solicitar redefinição de senha por e-mail.
- Exceções: Erros de banco de dados ou detecção de ataques de força bruta.

02

UC002 – Registrar Estudante

Atores: Administrador

Objetivo: Registrar novos estudantes no sistema.

- Fluxo Principal: Administrador preenche dados -> Sistema valida e registra.
- Fluxos Alternativos: Dados inválidos ou matrícula duplicada geram mensagens de erro.
- Fluxo Opcional: Inclusão de dados de um responsável pelo estudante.
- Exceções: Falhas no banco de dados registradas em log.

# Casos de Uso



UC003 – Registrar Presença via Reconhecimento Facial

Atores: Aluno, Sistema

Objetivo: Registrar automaticamente a presença do aluno usando reconhecimento facial.

- Fluxo Principal: Captura e validação da imagem -> Presença registrada no banco.
- Fluxos Alternativos: Imagem de baixa qualidade, falha na câmera.
- Fluxo Opcional: Registro manual da presença se o reconhecimento falhar.
- Exceções: Detecção de tentativas de falsificação (spoofing).



UC004 – Consultar Frequência do Estudante

Atores: Aluno, Professor, Administrador

Objetivo: Permitir a consulta do histórico de frequência de um estudante.

- Fluxo Principal: Busca por nome/matricula -> Filtros aplicados -> Relatório gerado.
- Fluxos Alternativos: Aluno não encontrado ou sem registros no período selecionado.
- Fluxo Opcional: Geração de gráficos de evolução da frequência.
- Exceções: Falha no banco de dados registrada em log.

# Casos de Uso



UC005 – Enviar Notificação para Responsável em Caso de Falta

Atores: Sistema, Responsável

Objetivo: Notificar o responsável por faltas injustificadas do aluno.

- Fluxo Principal: Falta detectada -> Dados do responsável buscados -> Notificação enviada.
- Fluxos Alternativos: Dados inválidos ou falha no envio resultam em log de erro e tentativas posteriores.
- Fluxo Opcional: Personalização da mensagem enviada.
- Exceções: Limite de tentativas definido para entrega da notificação.

# Protótipo

### CADASTRO DE ALUNO

Nome:

Email

Responsável

Telefone do responsável:

Telefone:

Data de nascimento


REGISTRAR FOTO

Foto registrada com sucesso!

Erro ao registrar foto

FAZER CADASTRO

### INFORMAÇÕES DO ALUNO



André Almeida

FALTAS SEMANAIS: 2

FALTAS TOTAIS: 25

dezembro de 2024

DOM.

SEG.

TER.

QUA.

QUI.

SEX.

SÁB.

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

CONTATOS:


RESPONSÁVEL: RICARDO ALMEIDA

Nº TELEFONE: (XX) XXXXX-XXXX

### RECONHECIMENTO FACIAL

REALIZAR RECONHECIMENTO

### RECONHECIMENTO FACIAL




André Almeida

TENTAR NOVAMENTE

REALIZAR OUTRO RECONHECIMENTO


### RECONHECIMENTO FACIAL





Aluno não reconhecido!


TENTAR NOVAMENTE


REALIZAR OUTRO RECONHECIMENTO


CADASTRO


INFO. ALUNO


ID FACIAL


CADASTRO


INFO. ALUNO


ID FACIAL


CADASTRO


INFO. ALUNO


ID FACIAL


CADASTRO

INFO. ALUNO

ID FACIAL

CADASTRO

INFO. ALUNO

ID FACIAL



# Extratos do código

```
1 export type Student = {
2   id: number;
3   name: string;
4   email: string;
5   password: string;
6   phone: string;
7   studentImage?: string;
8   birth: string;
9   guardians: Guardian[];
10  absences?: Absence[];
11 }
12 export type StudentRequestDTO = {
13   name: string;
14   email: string;
15   password: string;
16   phone: string;
17   studentImage?: string;
18   birth: string;
19   guardians: GuardianRequestDTO[];
20 }
21 export type ImageUploadDTO = {
22   file: any;
23 }
24 export interface Guardian {
25   id?: number;
26   name: string;
27   phone: string;
28 }
29 export type GuardianRequestDTO = {
30   name: string;
31   phone: string;
32 }
33 export interface Absence {
34   id?: number;
35   date: string;
36   justification?: string;
37 }
```

```
1 addStudent(newStudent: StudentRequestDTO) {
2   console.log(newStudent);
3   this.attedanceService.addStudent(newStudent).subscribe({
4     next: (student) => {
5       this.students.push(student);
6       if (this.studentImage) {
7         this.uploadImage(this.studentImage, student.id);
8       }
9     },
10    error: (error) => {
11      console.error('Error adding student', error);
12    }
13  });
14 }
```

```
1 @PostMapping
2 public ResponseEntity<StudentResponseDTO> create(@RequestBody StudentRequestDTO studentRequestDTO){
3   return ResponseEntity.status(HttpStatus.CREATED).body(studentService.create(studentRequestDTO));
4 }
5 @GetMapping
6 public ResponseEntity<List<StudentResponseDTO>> findAll(){
7   return ResponseEntity.ok(studentService.findAll());
8 }
9 @GetMapping("/{id}")
10 public ResponseEntity<StudentResponseDTO> findById(@PathVariable("id") Long id){
11   return ResponseEntity.status(HttpStatus.OK).body(studentService.findById(id));
12 }
13 @PutMapping("/{id}")
14 public ResponseEntity<StudentResponseDTO> update(@PathVariable("id") Long id, @RequestBody StudentUpdateDTO student){
15   return ResponseEntity.status(HttpStatus.OK).body(studentService.update(id, student));
16 }
```



# Demonstração de uso



# Lições aprendidas

## Documentação Essencial:

- A utilização de Swagger para documentar as APIs facilitou o entendimento e a integração.

## Configuração Flexível do Banco de Dados:

- A configuração para banco local e Railway possibilitou testes locais e produção sem dificuldades.

## Importância dos Testes:

- Testes de integração e validação das rotas ajudaram a identificar e corrigir falhas rapidamente.
- 

# Referências

## Tecnologias Utilizadas:

- [Spring Boot](#)
- [PostgreSQL](#)
- [Railway](#)

## Ferramentas de Documentação:

- [Swagger](#)
- [GitHub - Attendance System](#): Repositório do projeto.
- [Trello](#)

**OBRIGADO!**