

Task-API

Projeto Back - End

 Professor: João Ferreira

EQUIPE

Iann Vitor Souza de Lima - Scrum Master

Luciano Tomé Ferreira Correia Chaves - Desenvolvedor

Mario Antônio Carvalho Fragoso filho - Documentador

José Vinícius Alves Teixeira - Gerente de configuração

Yuki Peterson William Da Silva Araujo - Desenvolvedor II

Marcus Antônio Teti Cavalcanti Gomes - Documentador II

Agenda

OUTUBRO

DIA 16/10 -
ATRIBUIÇÃO DE
TAREFAS
6 até 7:45pm

NOVEMBRO

DIA 23/10 -
PLANEJAMENTO
7 até 8:30pm

DIA 22/10 -
REUNIÃO COM O
GRUPO EM GERAL II
6:30 até 8:15pm

DIA 29/11 -
DOCUMENTAÇÃO
7:30 até 9pm

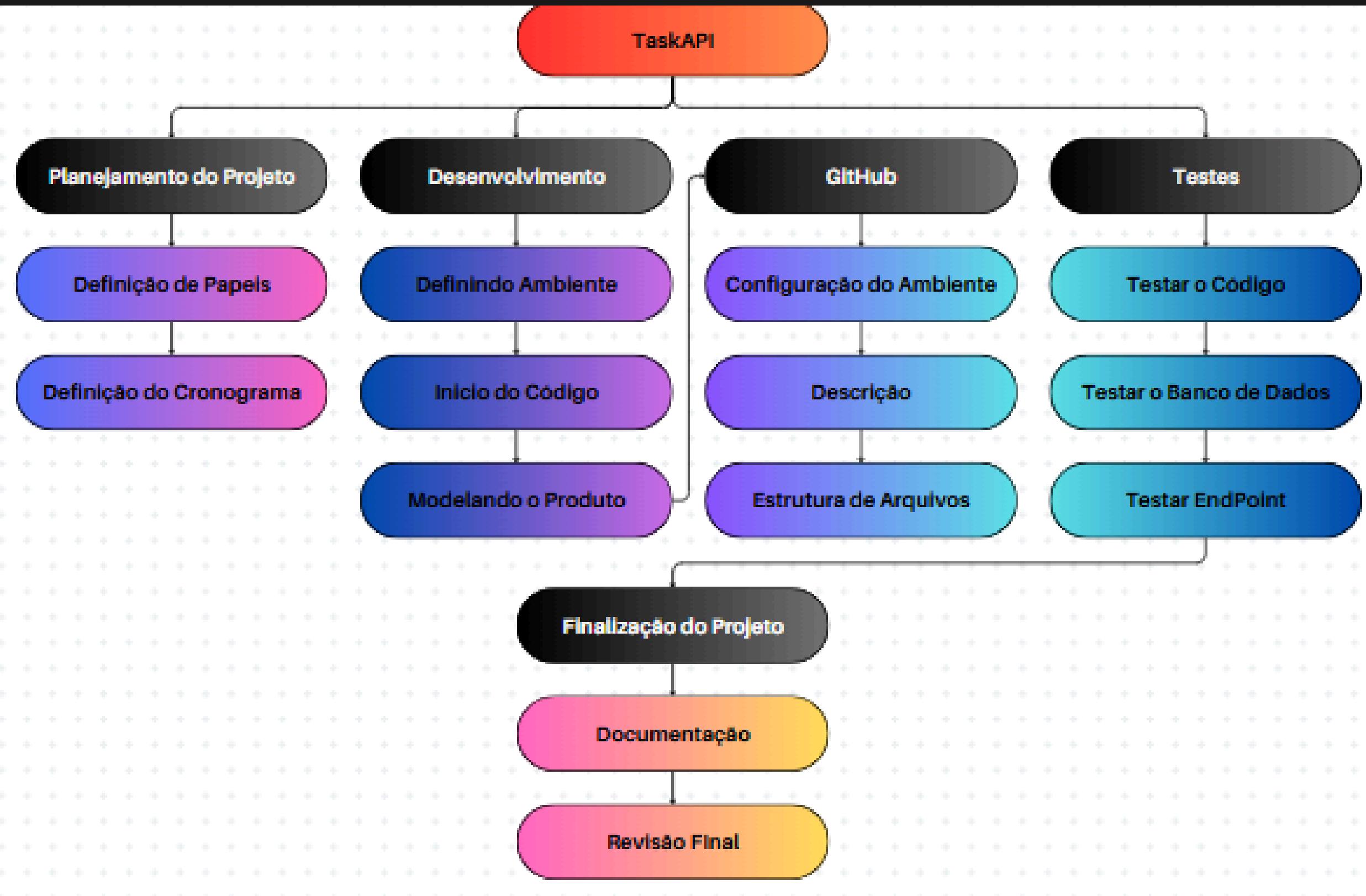
DEZEMBRO

DIA 1/11 -
FINALIZAÇÃO DO
7:15 até 8:45pm

DIA 5/11 - REUNIÃO
PARA RESOLVER
ERRO DO CODIGO
7:30 até 9:15pm

DIA 6/12
FINALIZAÇÃO II
7 até 8:30pm

Estrutura Análitica do Projeto - EAP



Atribuição de Tarefas

- **Iann Vitor** → **Scrum Master, Criação dos Slides, Elaborar EAP, Elaboração de Planilhas**
- **Luciano Tomé** → **Desenvolvedor, Criador do Ambiente Virtual**
- **José Vinicius** → **Gerente de Configuração, Elaboração do Github, Organização do Github**
- **Yuki Peterson** → **Criação dos Slides, Criação do Trello**
- **Marcus Teti** → **Documentador, Diagramas**
- **Mario Antônio** → **Documentador II, Casos de uso, Fluxograma**

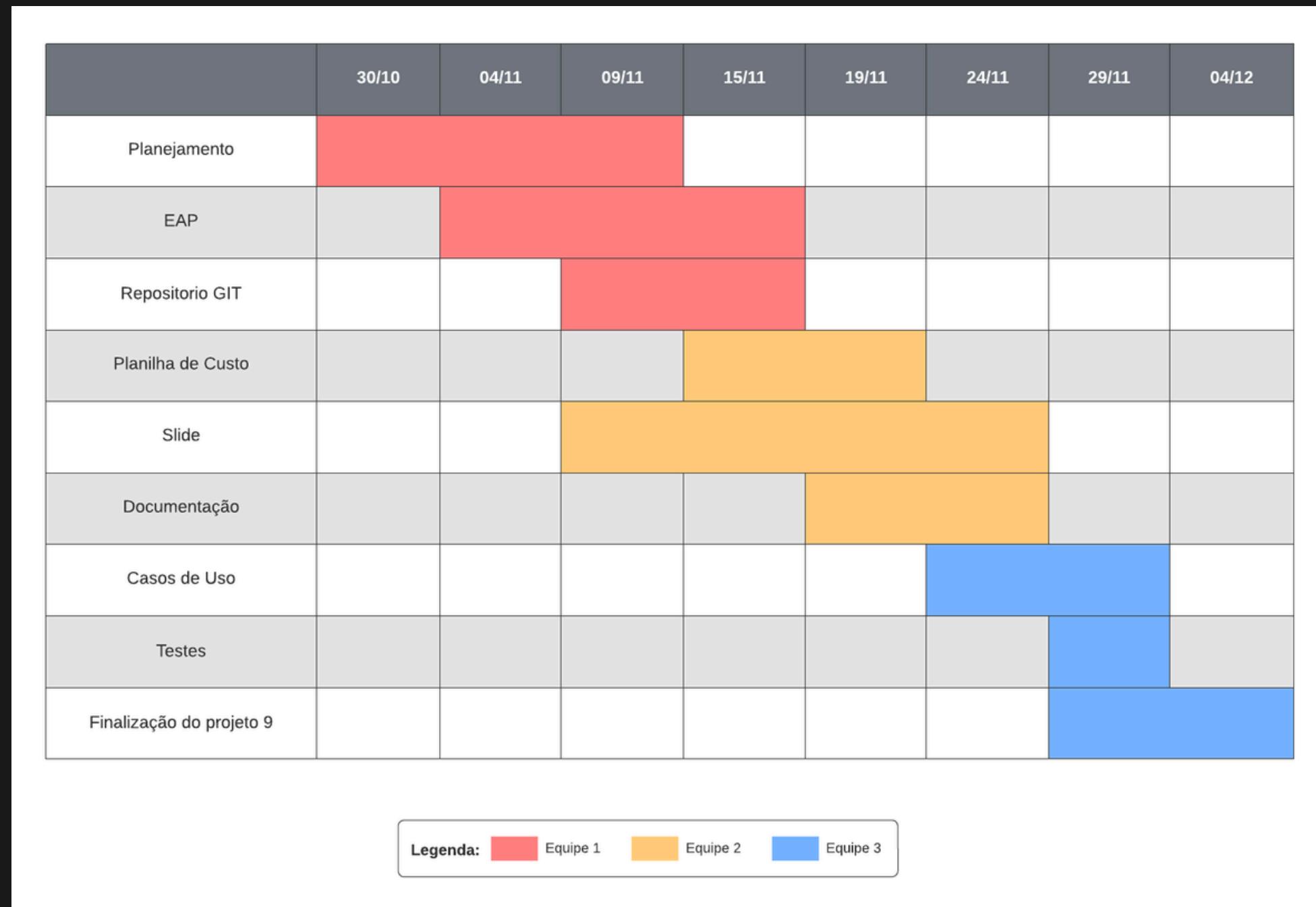
Custos e Investimentos

Categoria	Item	Investimento Estimado	Descrição
Ferramentas	Equipamentos	\$15000	Compra de Equipamentos como PC's, notebooks e etc
Capacitação	Cursos	\$1000	Treinamento e capacitação para os funcionários
Marketing e Divulgação	Campanha digital + Marca Digital	\$2500	Investimento em anúncios e na criação de uma identidade digital
Total		\$18.500	

Custos e Investimentos II

Categoria	Item	Custo Estimado	Descrição
Desenvolvimento	Ferramentas de desenvolvimento	Gratuito	IDE's, Frameworks como VSCode e Express.js são gratuitos
Documentação	Criação de Diagramas	Gratuito	As ferramentas usadas são totalmente gratuitas (Draw.io, Canva)
Equipe	2 Devs	\$6000	Custo estimado para contratar devs
Teste	Ferramenta de Teste	Gratuito	Ferramentas como Postman e ThunderClient
Total		\$6000	

Gráfico de Gantt



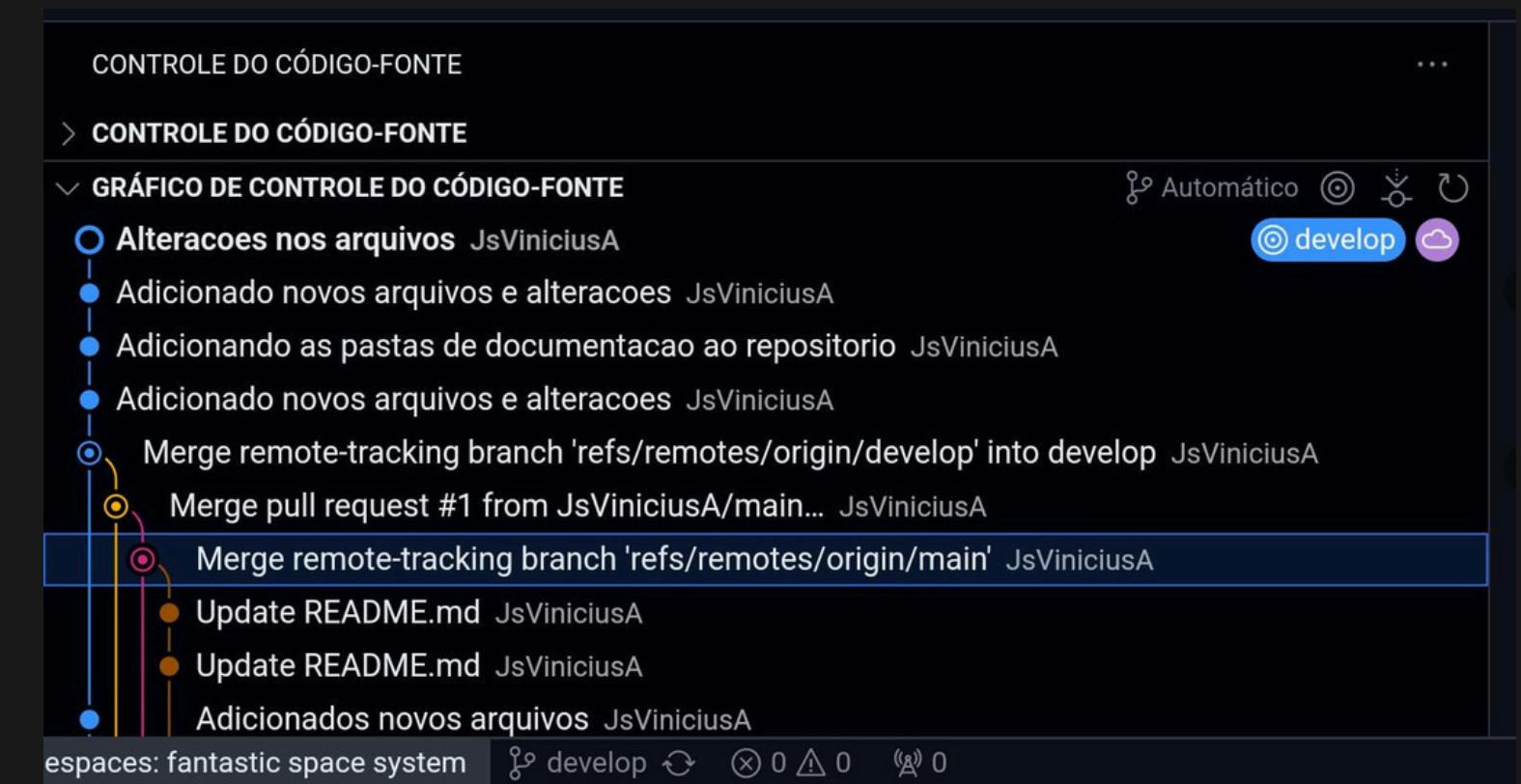
Resumo do Trello

The screenshot shows a Trello board titled "Projeto Back End". The board has the following structure:

- TODO**: Contains the card "+ Adicionar um cartão".
- PLANNED**: Contains the card "+ Adicionar um cartão".
- DOING**: Contains the card "+ Adicionar um cartão".
- WAITING**: Contains the card "Criação dos slides" with 1 item and a due date of 12 de dez.
- DONE**: Contains several cards:
 - Criação do mapa mental
 - Criação da tabela e colunas das categorias
 - Criação da tabela e colunas das tarefas
 - Criar tabela e colunas de usuários
 - Filtros de tarefas e busca que serão implementados
 - Base de dados da API(Criação do database da API)
 - Definição do Scrum Master - Iann
 - Definição do gerente de configuração - José Vinícius
 - Criação do repositório
 - Métodos do CRUD de usuários
- REFERENCES**: Contains cards for various topics:
 - Cargos
 - Desenvolvimento - Essencial
 - Desenvolvimento - Personalizado
 - Github
 - Banco de dados
 - Materiais para a apresentação

At the top right, there are buttons for "Power-Ups", "Automação", "Filtros", and "Compartilhar". Below the board, there are user icons for JV, LC, MC, and B.

Gerente de Configuração - Fluxo de Trabalho



The screenshot shows a GitHub search results page with the query 'is:pr is:closed'. The results list two closed pull requests:

- Merge para a main #2 by JsViniciusA was merged last week
- Merge para da main para a develop #1 by JsViniciusA was merged last month

Below the results, there are filters for 'Open' (0) and 'Closed' (2), and dropdowns for 'Author', 'Label', and 'Projects'.

The screenshot shows a GitHub commit history for a repository. It includes a header for 'JsViniciusA authored last week' and sections for 'Add files via upload' (by JsViniciusA last week) and 'Commits on Nov 27, 2024'. The commits listed are:

- atualização (by iamaplahebetboy committed 2 weeks ago)
- Alteracoes nos arquivos (by JsViniciusA committed 2 weeks ago)

At the bottom, it says 'Commits on Nov 25, 2024'.

Gerente de Configuração

Tutorial de Fluxo de Trabalho com Git

Fluxo de Trabalho do Projeto

1. Clonar o Repositório:

Baixe uma cópia do repositório localmente com o comando:

```
git clone https://github.com/JsViniciusA/TasksApi.git
```

2. Criar uma Nova Branch:

Sempre crie uma branch para cada funcionalidade ou correção:

```
git checkout -b nome-da-branch
```

3. Adicionar Alterações ao Stage:

Adicione os arquivos modificados ao stage:

```
git add .
```

4. Criar um Commit:

Registre as mudanças no histórico do Git com uma mensagem clara:

```
git commit -m "Descrição do que foi feito"
```

5. Atualizar com a Branch Principal (opcional):

Antes de enviar, atualize sua branch com as últimas mudanças da branch principal, para ter certeza de que não haverá conflitos nos arquivos:

```
git pull origin main
```

6. Enviar Alterações para o Re却itório Remoto:

Faça o push das mudanças para a branch remota:

```
git push origin nome-da-branch
```

7. Abrir um Pull Request:

Após o push, abra um Pull Request (PR) no GitHub para integrar as mudanças na branch principal. Um dos colaboradores irá analisar suas mudanças e se forem aceitas, irá mesclar elas.

Casos de Uso

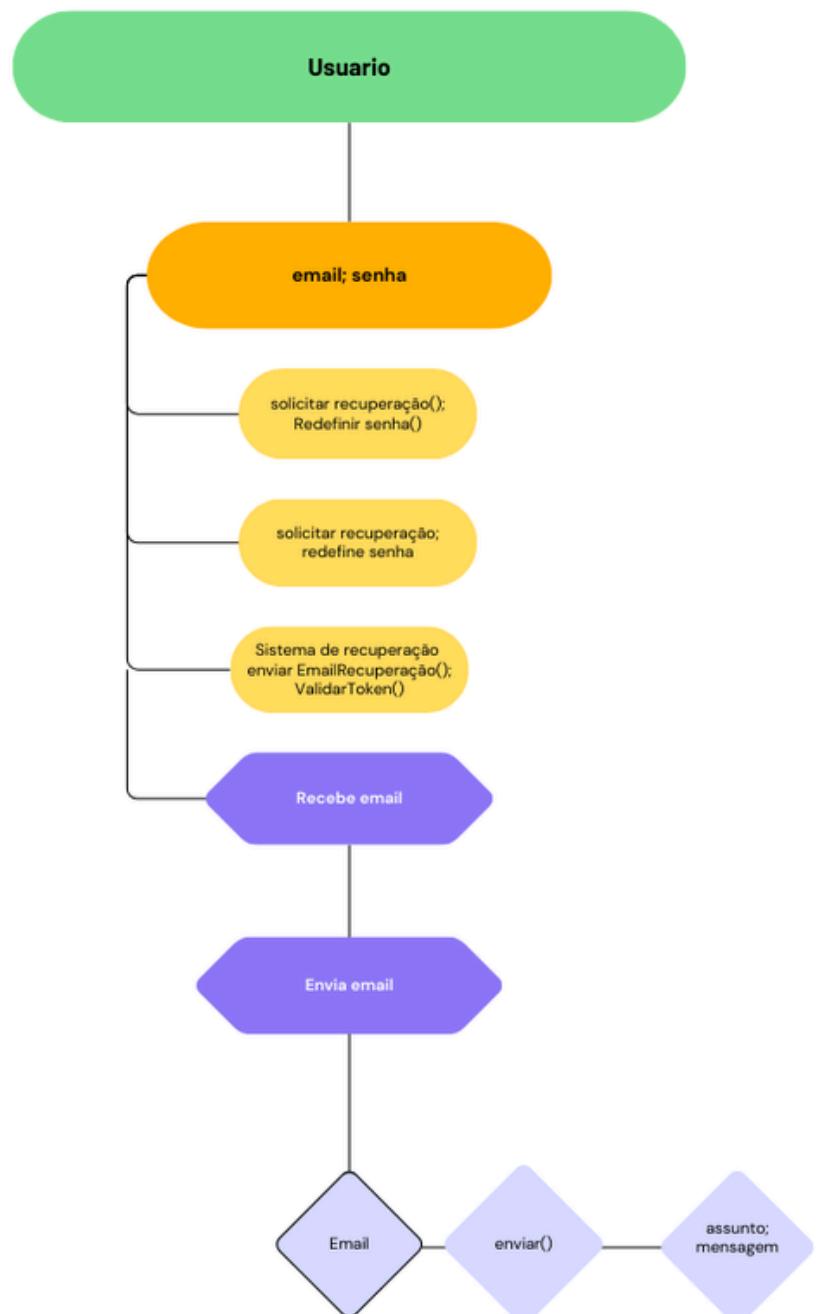
Seção	Descrição
ID	UC001
Título	Realizar Cadastro no Sistema
Autor Principal	Usuário 1
Objetivo	Permitir que o usuário crie uma conta para acessar o sistema.
Pré-condição	O usuário deve fornecer informações obrigatórias como e-mail e senha.
Pós-condição	O sistema registra os dados do usuário e permite que ele faça login.
Fluxo Principal	<ol style="list-style-type: none"> O usuário acessa a página de cadastro. Preenche os campos obrigatórios (e-mail, senha, nome). Confirma o cadastro clicando em "Cadastrar". O sistema salva os dados e exibe uma mensagem de sucesso.
Fluxo Alternativo	<ol style="list-style-type: none"> Se o e-mail informado já estiver registrado, o sistema exibe uma mensagem: "E-mail já cadastrado". O usuário pode tentar outro e-mail ou recuperar a conta existente.
Fluxo de Exceção	<ol style="list-style-type: none"> Caso o sistema detecte dados inválidos (e.g., senha fraca), exibe uma mensagem de erro. O cadastro não é concluído até que os dados sejam corrigidos.

Seção	Descrição
ID	UC002
Título	Recuperar Senha
Autor Principal	Usuário 2
Objetivo	Permitir que o usuário recupere o acesso à conta em caso de esquecimento da senha.
Pré-condição	O usuário deve informar um e-mail válido registrado no sistema.
Pós-condição	O sistema envia um e-mail com instruções para redefinição da senha.
Fluxo Principal	<ol style="list-style-type: none"> O usuário acessa a página de recuperação de senha. Insere o e-mail registrado na conta. Clica no botão "Recuperar Senha". O sistema envia um e-mail com um link de redefinição de senha. O usuário acessa o link, insere uma nova senha e confirma. O sistema salva a nova senha e exibe uma mensagem de sucesso.
Fluxo Alternativo	<ol style="list-style-type: none"> Se o e-mail informado não estiver registrado, o sistema exibe a mensagem: "E-mail não encontrado". O usuário pode tentar outro e-mail ou criar uma nova conta.
Fluxo de Exceção	<ol style="list-style-type: none"> Caso o sistema detecte problemas ao enviar o e-mail, exibe a mensagem: "Erro ao enviar e-mail. Tente novamente mais tarde". O processo de recuperação de senha é interrompido até que o problema seja resolvido.

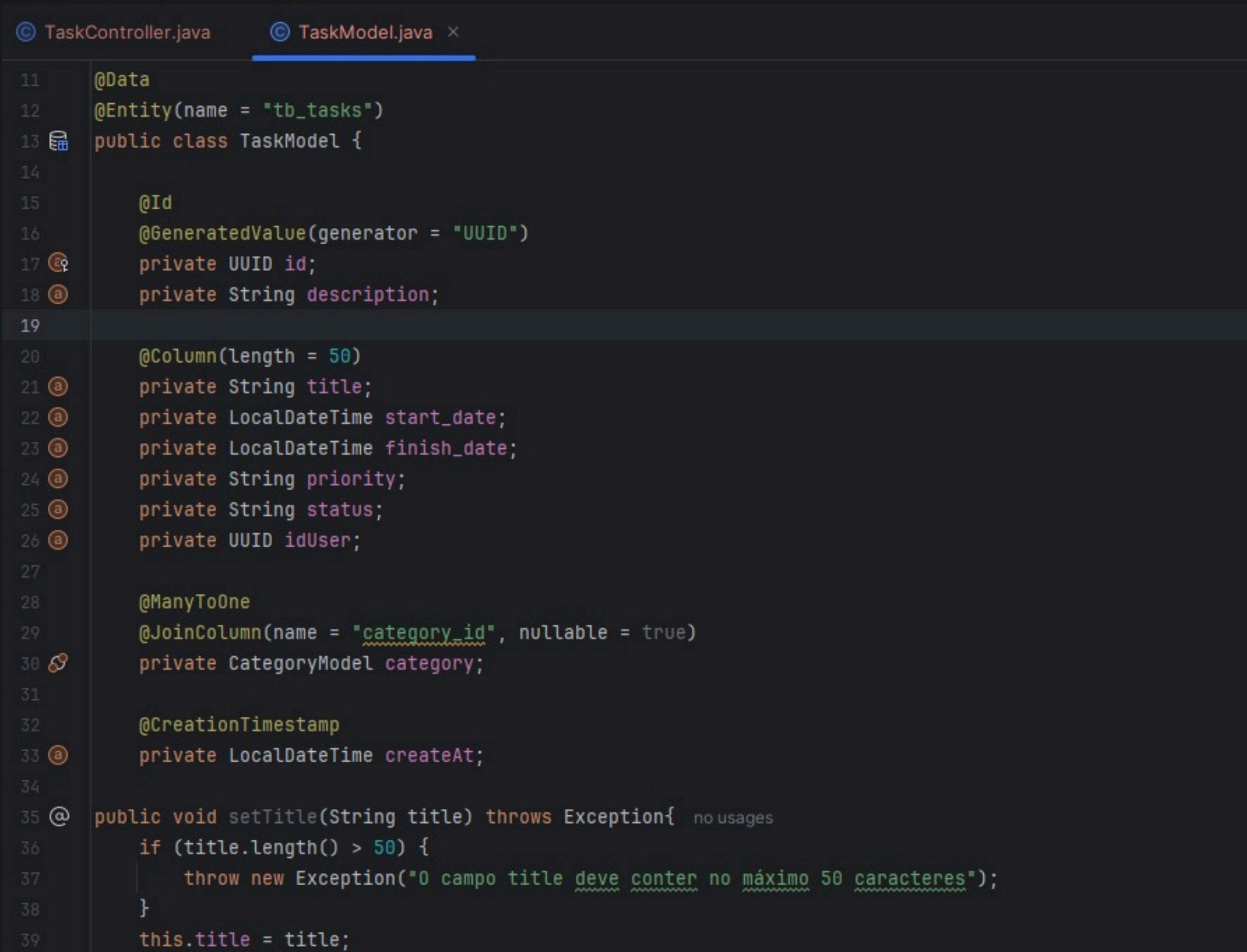
Seção	Descrição
ID	UC003
Título	Alterar dados do perfil
Autor Principal	Usuário 3
Objetivo	Permitir que o usuário atualize suas informações pessoais no sistema.
Pré-condição	O usuário deve estar logado no sistema.
Pós-condição	O sistema salva as informações atualizadas e mantém o perfil do usuário atualizado.
Fluxo Principal	<ol style="list-style-type: none"> O usuário acessa a página de "Editar Perfil". Altera os campos desejados (e.g., nome, e-mail, telefone). Clica no botão "Salvar Alterações". O sistema valida os dados, salva as alterações e exibe uma mensagem de sucesso.
Fluxo Alternativo	<ol style="list-style-type: none"> Se o usuário tentar alterar o e-mail para um já registrado, o sistema exibe a mensagem: "E-mail já utilizado". O usuário pode tentar outro e-mail ou manter o atual.
Fluxo de Exceção	<ol style="list-style-type: none"> Caso o sistema detecte dados inválidos (e.g., telefone em formato incorreto), exibe uma mensagem de erro. As alterações não são salvas até que os dados sejam corrigidos.



Diagrama UML para o Processo de Redefinição



Código (Modelo de tarefa)



The screenshot shows a code editor with two tabs: `TaskController.java` and `TaskModel.java`. The `TaskModel.java` tab is active, displaying the following Java code:

```
11  @Data
12  @Entity(name = "tb_tasks")
13  public class TaskModel {
14
15      @Id
16      @GeneratedValue(generator = "UUID")
17      private UUID id;
18      private String description;
19
20      @Column(length = 50)
21      private String title;
22      private LocalDateTime start_date;
23      private LocalDateTime finish_date;
24      private String priority;
25      private String status;
26      private UUID idUser;
27
28      @ManyToOne
29      @JoinColumn(name = "category_id", nullable = true)
30      private CategoryModel category;
31
32      @CreationTimestamp
33      private LocalDateTime createdAt;
34
35      public void setTitle(String title) throws Exception{
36          if (title.length() > 50) {
37              throw new Exception("O campo title deve conter no máximo 50 caracteres");
38          }
39          this.title = title;
}
```

Código (Controlador da tarefa)

```
© TaskController.java × © TaskModel.java
20  public class TaskController {
84      public ResponseEntity update(@RequestBody TaskModel taskModel, HttpServletRequest request, @PathVariable UUID id) {
110     }
111
112     @Operation(
113         summary = "Listar tarefas por categoria",
114         description = "Esta rota retorna todas as tarefas vinculadas a uma categoria específica, do usuário autenticado."
115     )
116     @ApiResponse(responseCode = "200", description = "Lista de tarefas retornada com sucesso")
117     @ApiResponse(responseCode = "404", description = "Categoria não encontrada ou não pertence ao usuário")
118     @ApiResponse(responseCode = "500", description = "Erro interno do servidor")
119     @GetMapping("/{category}/{categoryId}")
120     public ResponseEntity listTasksByCategory(@PathVariable UUID categoryId, HttpServletRequest request) {
121         var idUser = request.getAttribute("idUser");
122
123         var category = categoryRepository.findById(categoryId).orElse(null);
124         if (category == null || !category.getUser().getId().equals(idUser)) {
125             return ResponseEntity.status(HttpStatus.NOT_FOUND).body("Categoria não encontrada ou não pertence ao usuário.");
126         }
127
128         var tasks = taskRepository.findByCategoryId(categoryId);
129         return ResponseEntity.status(HttpStatus.OK).body(tasks);
130     }
131
132
133 }
134 }
```

Códigos (Módelo de Usuário)

```
© UserController.java    © UserModel.java ×  
1 package br.com.thiagomota.todolist.user;  
2  
3 > import ...  
11  
12  
13     @Data  
14     @Entity(name = "tb_users")  
15     public class UserModel {  
16  
17         @Id  
18         @GeneratedValue(generator = "UUID")  
19         private UUID id;  
20  
21         @Column(unique = true)  
22         private String username;  
23         private String name;  
24         private String password;  
25  
26         @CreationTimestamp  
27         private LocalDateTime createdAt;  
28 }
```

Código (Controlador do Usuário)

```
UserController.java x UserModel.java : 4 1 63 ^ v  
19 public class UserController {  
42  
43     @Operation(summary = "Listar usuários", description = "Esta rota serve para listar todos os usuários cadastrados.")  
44     @ApiResponses({  
45         @ApiResponse(responseCode = "200", description = "Usuários listados com sucesso"),  
46         @ApiResponse(responseCode = "500", description = "Erro interno do servidor")  
47     })  
48     @GetMapping("/{list}")  
49     public ResponseEntity findAllUsers() {  
50         var users = this.userRepository.findAll();  
51         return ResponseEntity.status(HttpStatus.OK).body(users);  
52     }  
53  
54     @Operation(summary = "Buscar usuário por ID", description = "Esta rota retorna os dados de um usuário específico pelo seu ID.")  
55     @ApiResponses({  
56         @ApiResponse(responseCode = "200", description = "Usuário encontrado"),  
57         @ApiResponse(responseCode = "404", description = "Usuário não encontrado"),  
58         @ApiResponse(responseCode = "500", description = "Erro interno do servidor")  
59     })  
60     @GetMapping("/{list}/{id}")  
61     public ResponseEntity findOneUser(@PathVariable UUID id) {  
62         UserModel user = this.userRepository.findById(id).orElse(null);  
63         if (user != null) {  
64             return ResponseEntity.status(HttpStatus.OK).body(user);  
65         }  
66         return ResponseEntity.status(HttpStatus.NOT_FOUND).body("Usuário não encontrado.");  
67     }  
68  
69     @Operation(summary = "Login do usuário", description = "Esta rota valida as credenciais do usuário para autenticação.")  
70     @ApiResponses({
```

```
UserController.java x UserModel.java : 4 1 63 ^ v  
19 public class UserController {  
67  
68  
69     @Operation(summary = "Login do usuário", description = "Esta rota valida as credenciais do usuário para autenticação.")  
70     @ApiResponses({  
71         @ApiResponse(responseCode = "200", description = "Login bem-sucedido"),  
72         @ApiResponse(responseCode = "401", description = "Senha incorreta"),  
73         @ApiResponse(responseCode = "404", description = "Usuário não encontrado"),  
74         @ApiResponse(responseCode = "500", description = "Erro interno do servidor")  
75     })  
76     @PostMapping("/{login}")  
77     public ResponseEntity login(@RequestBody LoginUserDto loginRequest) {  
78         UserModel user = this.userRepository.findByUsername(loginRequest.username());  
79         if (user == null) {  
80             return ResponseEntity.status(HttpStatus.NOT_FOUND).body("Usuário não encontrado.");  
81         }  
82         boolean passwordMatches = BCrypt.verifyer()  
83             .verify(loginRequest.password().toCharArray(), user.getPassword())  
84             .verified;  
85         if (passwordMatches) {  
86             return ResponseEntity.status(HttpStatus.OK).body("Login bem-sucedido.");  
87         } else {  
88             return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body("Senha incorreta.");  
89         }  
90     }  
91  
92 }
```

Código (Rotas de Usuario)

```
© UserController.java x © UserModel.java
15  @Tag(name = "Gerenciamento de Usuários", description = "Rotas para gerenciar Usuário")
16
17  @RestController
18  @RequestMapping("/users")
19  public class UserController {
20
21      @Autowired
22      private UserRepository userRepository;
23
24      @Operation(summary = "Criar usuário", description = "Esta rota serve para criar um novo usuário.")
25      @ApiResponses({
26          @ApiResponse(responseCode = "201", description = "Usuário criado com sucesso"),
27          @ApiResponse(responseCode = "400", description = "Usuário já existente"),
28          @ApiResponse(responseCode = "500", description = "Erro interno do servidor")
29      })
30      @PostMapping("/register")
31      public ResponseEntity<UserModel> create(@RequestBody UserModel userModel) {
32          var user = this.userRepository.findByUsername(userModel.getUsername());
33          if (user != null) {
34              return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("Usuário já existente.");
35          }
36          var passwordHashred = BCrypt.withDefaults()
37              .hashToString( cost: 12, userModel.getPassword().toCharArray());
38          userModel.setPassword(passwordHashred);
39          var userCreated = this.userRepository.save(userModel);
40          return ResponseEntity.status(HttpStatus.CREATED).body(userCreated);
41      }
}
```

Códigos(Controlador de categoria)

```
CategoryController.java
public class CategoryController {
    ...
    @Operation(
        new = {
            @Summary("Adicionar tarefa à categoria"),
            @Description("Esta rota vincula uma tarefa a uma categoria específica.")
        },
        @ApiResponse(responseCode = "200", description = "Tarefa adicionada à categoria com sucesso"),
        @ApiResponse(responseCode = "404", description = "Categoria ou tarefa não encontrada, ou não pertencem ao usuário"),
        @ApiResponse(responseCode = "500", description = "Erro interno do servidor")
    )
    @PostMapping("/{categoryId}/tasks/{taskId}")
    public ResponseEntity addTaskToCategory(@PathVariable UUID categoryId, @PathVariable UUID taskId, @RequestParam UUID userId) {
        var category = categoryRepository.findById(categoryId).orElse(null);
        if (category == null || !category.getUser().getId().equals(userId)) {
            return ResponseEntity.status(HttpStatus.NOT_FOUND).body("Categoria não encontrada ou não pertence ao usuário.");
        }

        var task = taskRepository.findById(taskId).orElse(null);
        if (task == null || !task.getIdUser().equals(userId)) {
            return ResponseEntity.status(HttpStatus.NOT_FOUND).body("Tarefa não encontrada ou não pertence ao usuário.");
        }

        task.setCategory(category);
        taskRepository.save(task);
        return ResponseEntity.status(HttpStatus.OK).body("Tarefa adicionada à categoria.");
    }
}

CategoryController.java
public class CategoryController {
    ...
    @Operation(
        new = {
            @Summary("Remover tarefa da categoria"),
            @Description("Esta rota desvincula uma tarefa de uma categoria específica.")
        },
        @ApiResponse(responseCode = "200", description = "Tarefa removida da categoria com sucesso"),
        @ApiResponse(responseCode = "404", description = "Categoria ou tarefa não encontrada, ou não pertencem ao usuário"),
        @ApiResponse(responseCode = "500", description = "Erro interno do servidor")
    )
    @DeleteMapping("/{categoryId}/tasks/{taskId}")
    public ResponseEntity removeTaskFromCategory(@PathVariable UUID categoryId, @PathVariable UUID taskId, @RequestParam UUID userId) {
        var category = categoryRepository.findById(categoryId).orElse(null);
        if (category == null || !category.getUser().getId().equals(userId)) {
            return ResponseEntity.status(HttpStatus.NOT_FOUND).body("Categoria não encontrada ou não pertence ao usuário.");
        }

        var task = taskRepository.findById(taskId).orElse(null);
        if (task == null || !task.getIdUser().equals(userId) || task.getCategory() == null || !task.getCategory().getId().equals(categoryId)) {
            return ResponseEntity.status(HttpStatus.NOT_FOUND).body("Tarefa não encontrada ou não pertence à categoria.");
        }

        task.setCategory(null);
        taskRepository.save(task);
        return ResponseEntity.status(HttpStatus.OK).body("Tarefa removida da categoria.");
    }
}
```

Códigos(Rotas de categorias)

© CategoryController.java × © CategoryModel.java © CategoryDto.java

```
21 public class CategoryController {  
22     )  
23     @ApiResponse(responseCode = "200", description = "Categoria encontrada com sucesso")  
24     @ApiResponse(responseCode = "404", description = "Categoria não encontrada ou não pertence ao usuário")  
25     @ApiResponse(responseCode = "500", description = "Erro interno do servidor")  
26     @GetMapping("*/{id}")  
27     public ResponseEntity getCategory(@PathVariable UUID id, @RequestParam UUID userId) {  
28         CategoryModel category = categoryRepository.findById(id).orElse(null);  
29         if (category == null || !category.getUser().getId().equals(userId)) {  
30             return ResponseEntity.status(HttpStatus.NOT_FOUND).body("Categoria não encontrada.");  
31         }  
32         return ResponseEntity.status(HttpStatus.OK).body(category);  
33     }  
34  
35     @Operation( new *  
36         summary = "Atualizar categoria",  
37         description = "Esta rota atualiza as informações de uma categoria específica."  
38     )  
39     @ApiResponse(responseCode = "200", description = "Categoria atualizada com sucesso")  
40     @ApiResponse(responseCode = "404", description = "Categoria não encontrada ou não pertence ao usuário")  
41     @ApiResponse(responseCode = "500", description = "Erro interno do servidor")  
42     @PutMapping("*/{id}")  
43     public ResponseEntity updateCategory(@PathVariable UUID id, @RequestParam UUID userId, @RequestBody CategoryModel updatedCategory) {  
44         CategoryModel category = categoryRepository.findById(id).orElse(null);  
45         if (category == null || !category.getUser().getId().equals(userId)) {  
46             return ResponseEntity.status(HttpStatus.NOT_FOUND).body("Categoria não encontrada.");  
47         }  
48     }  
49 }
```

Códigos(Modelo de Categoria)

```
© CategoryController.java   © CategoryModel.java ×  © CategoryDto.java

11
12     @Data new *
13     @NoArgsConstructor
14     @Entity
15     @Table(name = "categories")
16     public class CategoryModel {
17
18         @Id
19         @GeneratedValue(strategy = GenerationType.AUTO)
20         private UUID id;
21
22         @Column(nullable = false)
23         private String name;
24
25         @ManyToOne
26         @JoinColumn(name = "user_id", nullable = false)
27         private UserModel user;
28
29         @Column(nullable = true)
30         @OneToMany(mappedBy = "category", cascade = CascadeType.ALL, orphanRemoval = true)
31         private List<TaskModel> tasks;
32     }
33
```

Protótipo

Gerenciamento de Tarefas Rotas para gerenciar tarefas ▼

Gerenciamento de Usuários Rotas para gerenciar Usuário ^

POST [/users/register](#) Criar usuário 🔒 ^

Esta rota serve para criar um novo usuário.

Parameters Try it out Reset

No parameters

Request body required application/json ▼

[Example Value](#) [Schema](#)

```
{  
  "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",  
  "username": "string",  
  "name": "string",  
  "password": "string",  
  "createdAt": "2024-12-13T01:25:11.496Z"  
}
```

Protótipo

POST /tasks/ Criar tarefa

Esta rota serve para criar uma nova tarefa vinculada ao usuário autenticado.

Parameters

No parameters

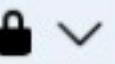
Request body required

application/json

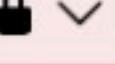
```
{  
  "description": "trabalho back",  
  "title": "em progresso",  
  "start_date": "2024-12-13T01:17:32.960Z",  
  "finish_date": "2024-12-13T01:17:32.960Z",  
  "priority": "10",  
  "status": "aguardando",  
  "idUser": "3fa85f64-5717-4562-b3fc-2c963f66afa6",  
}  
Execute
```

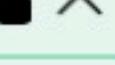
Protótipo

Gerenciamento de Categorias Rotas para gerenciar categorias e vincular tarefas ^

GET /categories/{id} Buscar categoria  ▾

PUT /categories/{id} Atualizar categoria  ▾

DELETE /categories/{id} Deletar categoria  ▾

POST /categories/{userId} Criar categoria  ▲

Inserir um pouquinho de texto

Esta rota cria uma nova categoria vinculada a um usuário específico.

Parameters Try it out

Name	Description
userId * required string(\$uuid) (query)	userId

Protótipo

Servers

http://localhost:8080 - Generated server url

Authorize 

Gerenciamento de Tarefas

Gerenciamento de Usuários

POST /users/register Criar usuário

POST /users/login Login do usuário

GET /users/list Listar usuários

GET /users/list/{id} Buscar usuário por ID

Gerenciamento de Categorias Rotas para gerenciar categorias e vincular tarefas

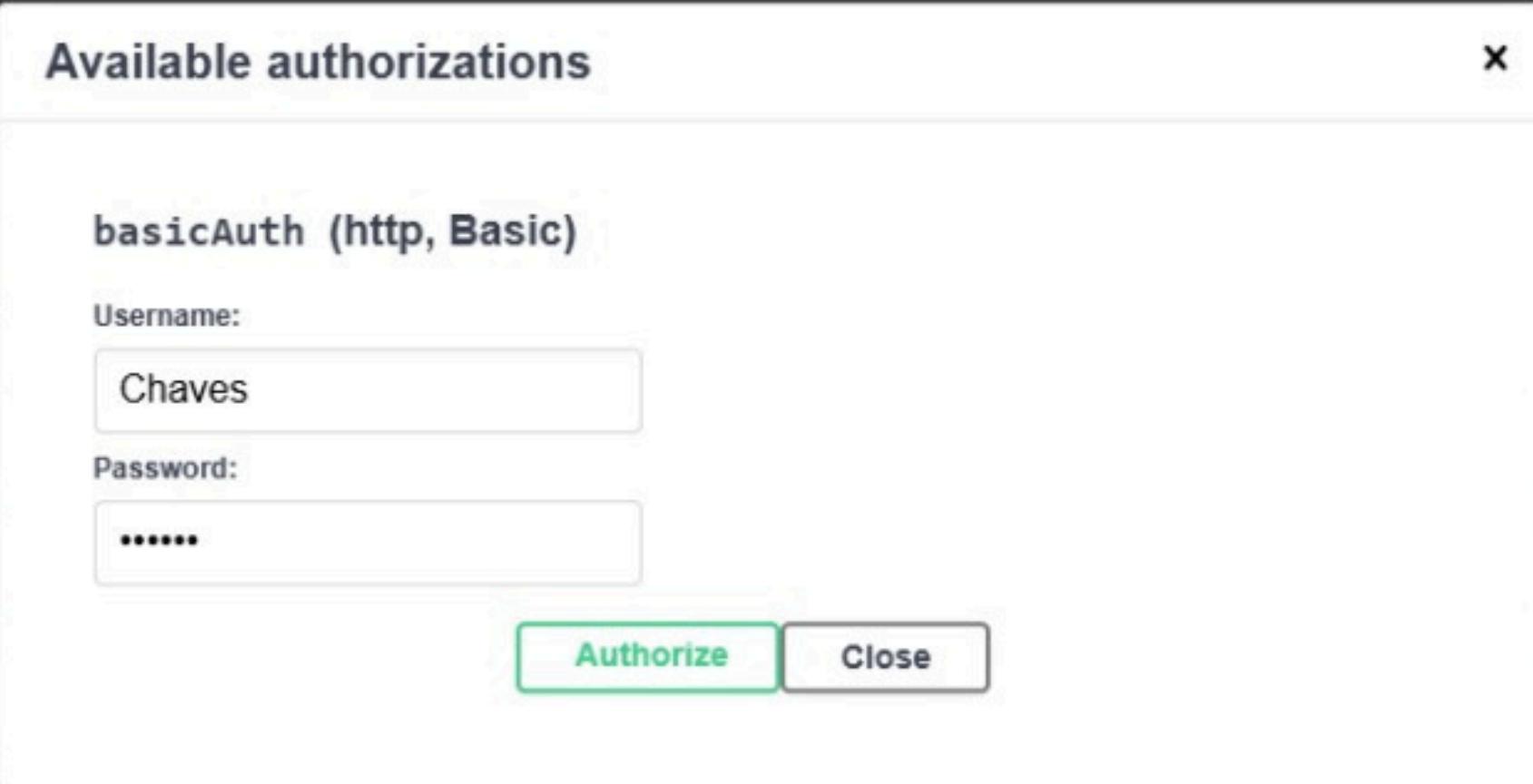
Available authorizations

basicAuth (http, Basic)

Username: Chaves

Password: *****

Authorize Close



Lições Aprendidas

- **Não deixar as coisas para ultima hora**
- **Trabalhar sempre em equipe**
- **Escutar seus colegas de equipe**
- **Aprender a lidar com o desconhecido**

Agradecimentos

Comunidade de Desenvolvedores

Professor João Ferreira

Todos os envolvidos no projeto

Referencias

ChatGPT

Youtube

Comunidade da Uninassau (vulgo MD)

GitHub

FIM