Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação

Sistema de Gestão de Tarefas Acadêmicas

Relatório de Projeto

Disciplina: SSC0124 - Análise e Projeto Orientado a Objetos
Profa. Dra. Lina Garcés

Integrantes do Grupo:
Caroline Akimi Kurosaki Ueda - 15445630
Felipe Camargo Cerri - 15451119
Nicolas de Sousa Maia - 15481857

São Carlos, 8 de dezembro de 2025

Sumário

1 Introdução

O presente trabalho tem como objetivo aplicar os conhecimentos adquiridos na disciplina de Análise e Projeto Orientado a Objetos (SSC0124) para o desenvolvimento de um sistema de software que solucione um problema real. O sistema proposto visa auxiliar estudantes universitários no gerenciamento semestral de suas tarefas acadêmicas e no acompanhamento das frequências nas disciplinas matriculadas.

A partir da lista de requisitos funcionais fornecida, o projeto abrange desde a modelagem inicial, com Diagramas de Casos de Uso e Componentes, até o detalhamento e implementação de um componente específico do sistema. O relatório a seguir documenta as etapas de análise, projeto e implementação realizadas pelo grupo.

2 Lista de Requisitos Funcionais e Casos de Uso

De acordo com os requisitos funcionais fornecidos para o sistema, organizamos os casos de uso para uma melhor modelagem do sistema como constam abaixo:

2.1 Componente de Cadastro e Autenticação

- Como Aluno, eu gostaria de realizar o meu cadastro no sistema.
- Como Aluno, eu gostaria de fazer login no sistema.
- Como Aluno, eu gostaria de fazer logout do sistema.
- Como Aluno, eu gostaria de recuperar minha senha.

2.2 Componente de Gerenciamento de Disciplinas

- Como Aluno, eu gostaria de cadastrar, editar e excluir disciplinas.
- Como Aluno, eu gostaria de associar tarefas, provas e trabalhos a uma disciplina.
- Como Aluno, eu gostaria de configurar os pesos das avaliações na disciplina.
- Como Sistema, eu gostaria de calcular as médias por grupo de atividades e a média final da disciplina.

2.3 Componente de Gerenciamento de Tarefas e Atividades

- Como Aluno, eu gostaria de criar, editar e excluir tarefas (atividades, aulas, provas, trabalhos).

- Como Aluno, eu gostaria de visualizar as tarefas, seus prazos e status.
- Como Aluno, eu gostaria de marcar uma tarefa como concluída.
- Como Aluno, eu gostaria de registrar a nota obtida em uma tarefa.
- Como Aluno, eu gostaria de definir lembretes para tarefas próximas do prazo.

2.4 Componente de Controle de Frequência

- Como Aluno, eu gostaria de registrar minha frequência em uma aula.
- Como Aluno, eu gostaria de informar o número total de aulas previstas no semestre.
- Como Aluno, eu gostaria de editar ou excluir um registro de frequência incorreto.
- Como Sistema, eu gostaria de calcular o total de faltas e a porcentagem de presença automaticamente.
- Como Sistema, eu gostaria de emitir um alerta quando a porcentagem de faltas estiver próxima do limite.
- Como Aluno, eu gostaria de configurar um limite de faltas que gera uma notificação de aviso.
- Como Aluno, eu gostaria de visualizar o relatório de frequência e de risco de reprovação por falta de cada disciplina.

2.5 Componente de Relatórios (Dashboard)

- Como Aluno, eu gostaria de visualizar minhas tarefas e aulas.
- Como Aluno, eu gostaria de filtrar tarefas por disciplina, tipo ou status.
- Como Aluno, eu gostaria de ordenar as tarefas por data ou prioridade.
- Como Aluno, eu gostaria de visualizar um resumo de pendências e conclusão de tarefas com percentual de presença por disciplina.
- Como Aluno, eu gostaria de visualizar um resumo de pendências e conclusão de tarefas com percentual de presença geral por semestre.

3 Diagrama de Casos de Uso

Abaixo apresenta-se o diagrama de Casos de Uso do Sistema de Gestão Acadêmica, modelados a partir dos Casos de Uso definidos anteriormente.

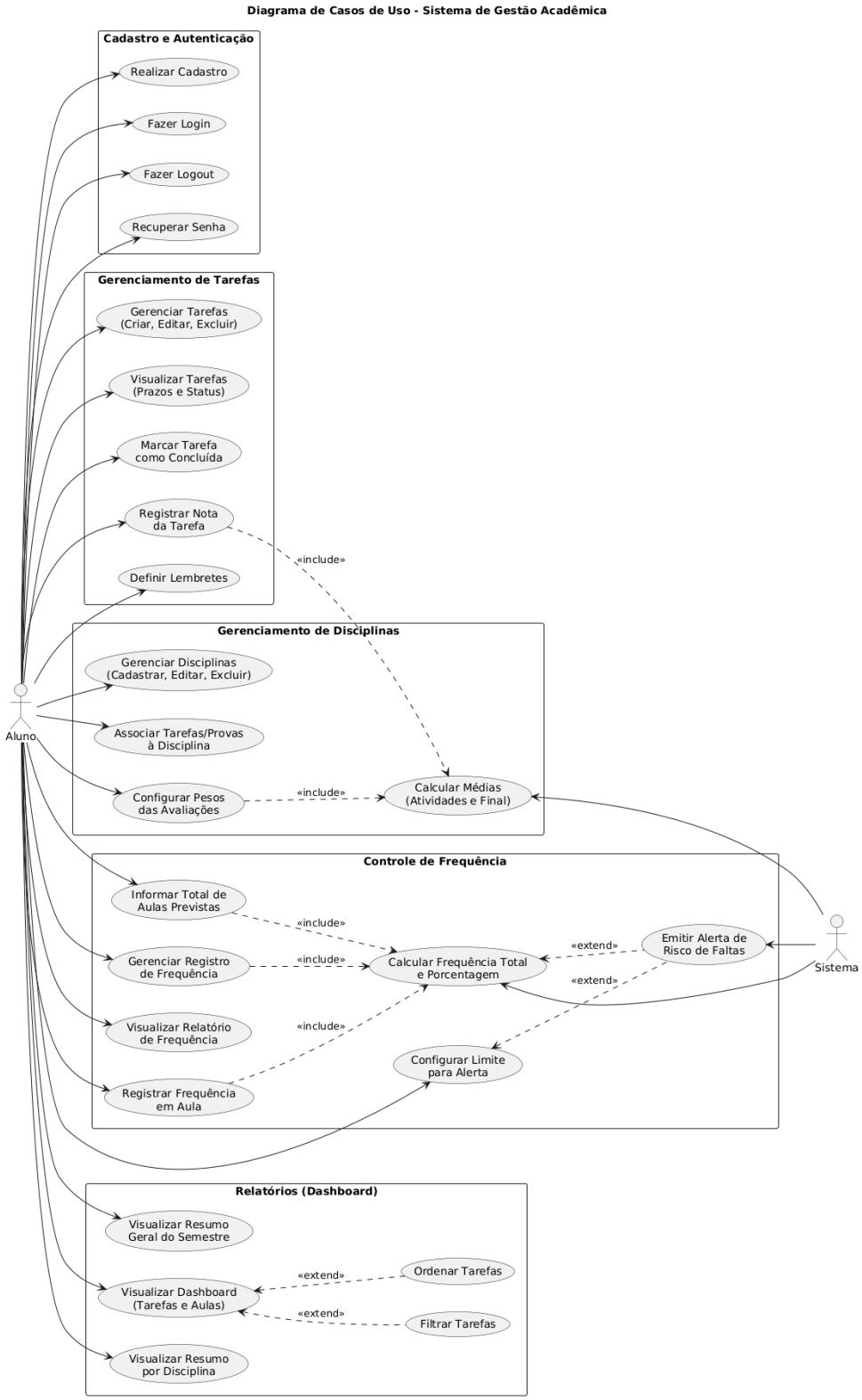

Figura 1: Diagrama de Casos de Uso - Sistema de Gestão Acadêmica

4 Diagrama de Componentes

O diagrama a seguir ilustra a arquitetura do sistema e o relacionamento entre seus componentes.

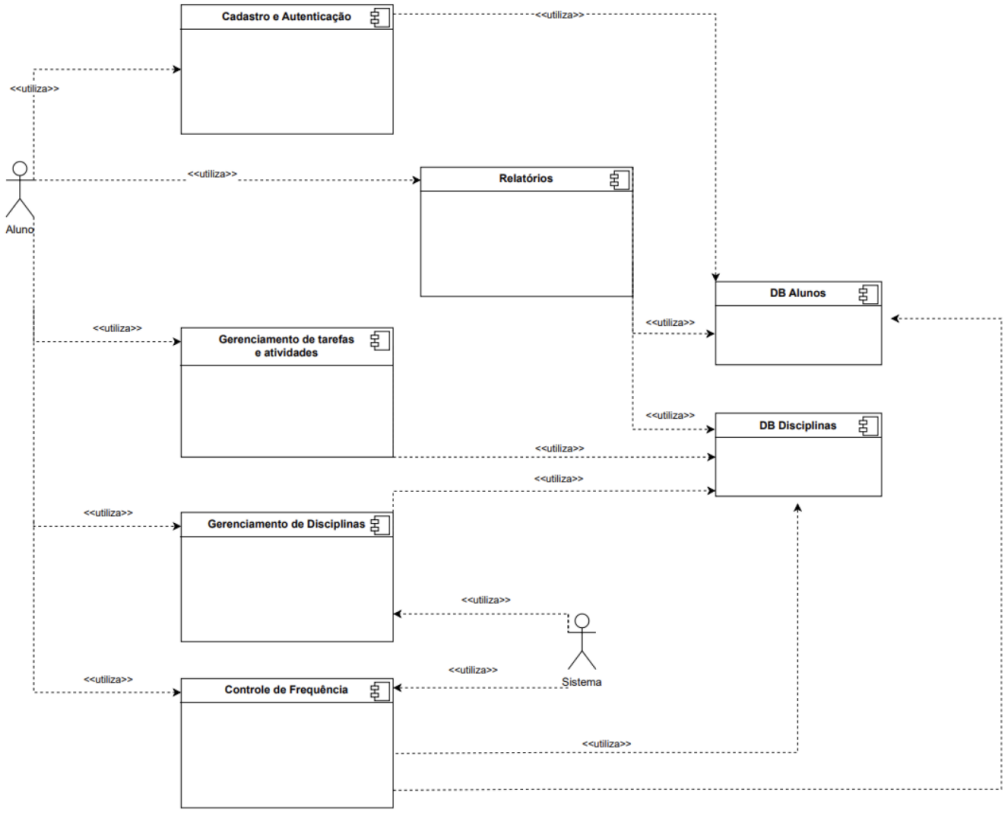

Figura 2: Diagrama de Componentes do Sistema

5 Detalhamento do Componente: Cadastro e Autenticação

O componente "**Cadastro e Autenticação**"foi selecionado para detalhamento aprofundado, incluindo descrições estendidas, modelagem de classes e sequências.

5.1 Descrições Estendidas dos Casos de Uso

5.1.1 Caso de uso: Cadastro no sistema

Ator principal: Aluno

Interessados e Interesses:

Aluno: Deseja realizar o cadastro no sistema para gerenciamento semestral de tarefas acadêmicas e no acompanhamento das frequências em disciplinas matriculadas.

Pré-Condições: O aluno tem acesso ao sistema e possui os dados necessários para o cadastro.

Pós-Condições: O aluno tem seus dados cadastrados na base de dados de alunos e possui um cadastro registrado.

Cenário de Sucesso Principal:

1. O aluno acessa o sistema de gerenciamento de tarefas.
2. O aluno deseja criar um cadastro.
3. O aluno manda seus dados para o cadastro.
4. O sistema realiza o cadastro.

Fluxo alternativo 1: (1-3)

4. O sistema não consegue realizar o cadastro.

5.1.2 Caso de uso: Fazer login no sistema

Ator principal: Aluno

Interessados e Interesses:

Aluno: Deseja acessar sua conta pessoal para visualizar e gerenciar suas disciplinas, notas e tarefas de forma segura.

Sistema: Necessita identificar o usuário para apresentar os dados corretos e restringir o acesso a informações privadas.

Pré-Condições: O aluno já possui um cadastro realizado e o sistema está disponível.

Pós-Condições: O aluno é autenticado e direcionado para o painel principal (dashboard) com sua sessão iniciada.

Cenário de Sucesso Principal:

1. O aluno acessa a tela de login do sistema.
2. O aluno insere suas credenciais de acesso (usuário/e-mail e senha).
3. O aluno solicita a entrada no sistema.
4. O sistema valida as credenciais fornecidas.
5. O sistema libera o acesso e redireciona o aluno para a página inicial.

Fluxo alternativo 1: (1-4)

5. O aluno não está cadastrado no sistema.
6. O aluno deve seguir para página de cadastro.

5.1.3 Caso de uso: Fazer logout do sistema

Ator principal: Aluno

Interessados e Interesses:

Aluno: Deseja encerrar sua sessão para garantir a segurança de seus dados e evitar que terceiros acessem suas informações acadêmicas.

Pré-Condições: O aluno deve estar logado no sistema.

Pós-Condições: A sessão do aluno é encerrada, e ele é redirecionado para a tela de login ou página pública.

Cenário de Sucesso Principal:

1. O aluno, estando logado, seleciona a opção de sair (logout).
2. O sistema recebe a requisição de encerramento.
3. O sistema finaliza a sessão do usuário.
4. O sistema redireciona o aluno para a tela de login.

5.1.4 Caso de uso: Recuperar senha

Ator principal: Aluno

Interessados e Interesses:

Aluno: Deseja redefinir sua senha de acesso caso a tenha esquecido, para poder acessar o sistema de gerenciamento de suas atividades acadêmicas.

Pré-Condições: O aluno possui um cadastro ativo, mas não se lembra da credencial de senha.

Pós-Condições: O sistema indica as instruções e muda senha do usuário no banco de dados.

Cenário de Sucesso Principal:

1. O aluno acessa o menu principal e seleciona a opção de recuperação de senha.
2. O sistema solicita o e-mail ou identificador associado à conta.
3. O aluno fornece o dado solicitado.
4. O sistema verifica a existência do usuário na base de dados.
5. O sistema solicita a nova senha do aluno.
6. O aluno redefine a senha.

5.2 Identificação de Classes

Para o componente de Cadastro e Autenticação, foram identificadas as seguintes classes, já pensando na compossição do código orientado a objetos que precisará ser desacoplado e seguir o principio de separação de responsabilidades, utilizando o padrão MVC:

Classe: Aluno

- **Atributos:** nome, email, senha.
- **Métodos:** `inicialização`.

Classe: View

- **Responsabilidade:** Interface com o usuário.
- **Métodos:** menu, get_dados, tela_inicial, dados_incorretos, get_email, get_senha, email_ja_cadastrado.

Classe: Controller

- **Responsabilidade:** Controle do fluxo de interação.

- **Métodos:** cria_aluno, verifica_email, verifica_aluno.

Classe: Model

- **Responsabilidade:** Persistência de dados.
- **Atributos:** df (DataFrame).
- **Métodos:** acha_aluno, insere_aluno, muda_senha.

5.3 Diagrama de Classes

O diagrama abaixo apresenta o modelo de domínio planejado para o componente, definindo as multiplicidades das relações.

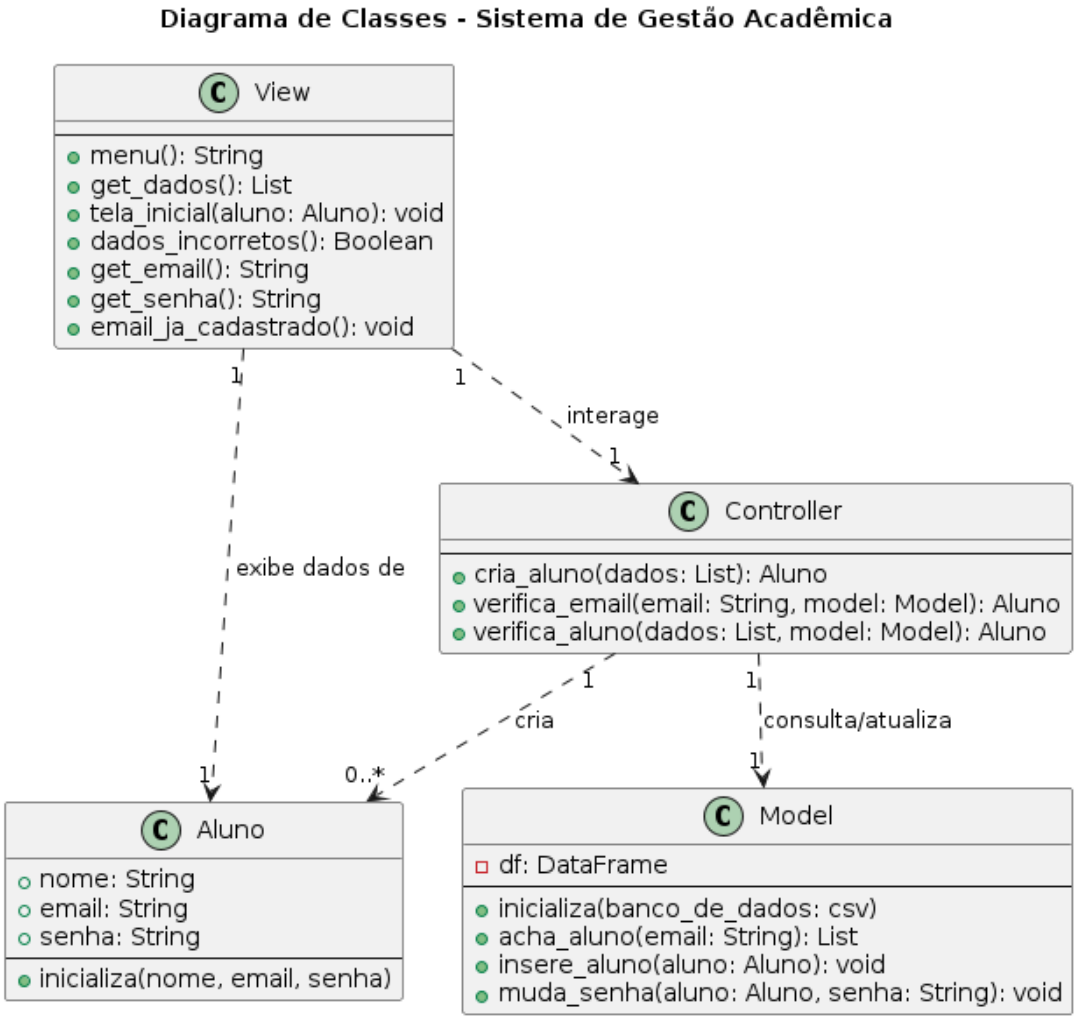

Figura 3: Diagrama de Classes - Componente de Cadastro (Domínio)

5.4 Diagramas de Sequência

Abaixo estão apresentados os diagramas de sequência para as funcionalidades do componente utilizando o padrão MVC.

5.4.1 Fluxo de Login

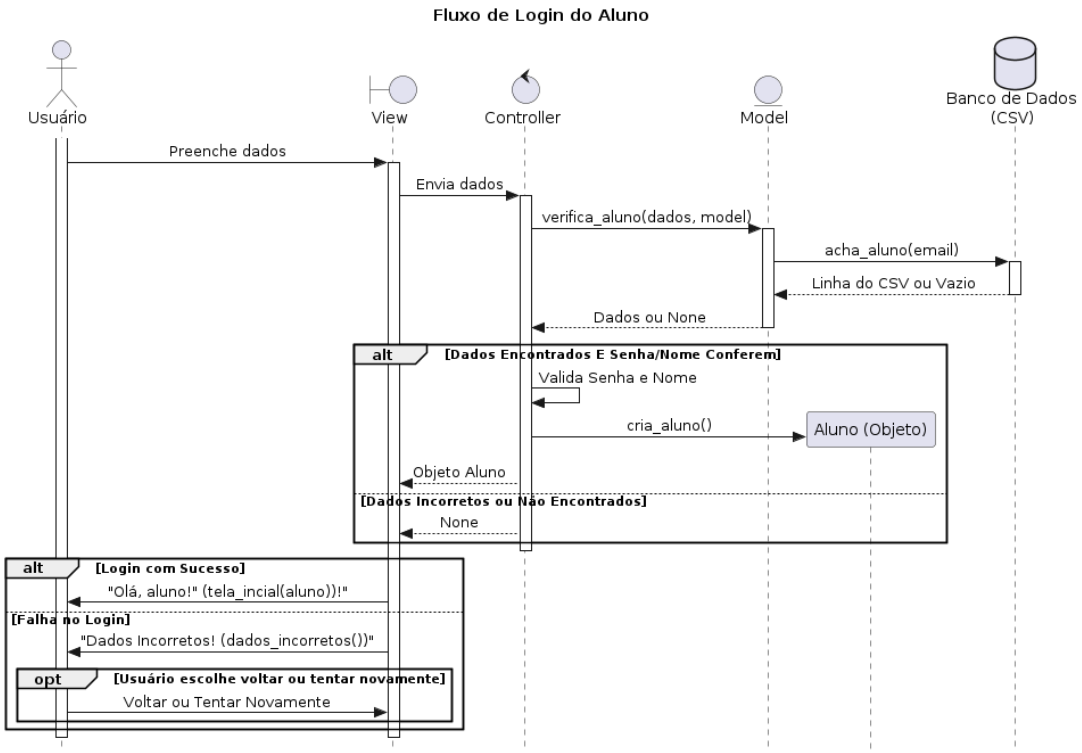

Figura 4: Diagrama de Sequência - Login

5.4.2 Fluxo de Logout

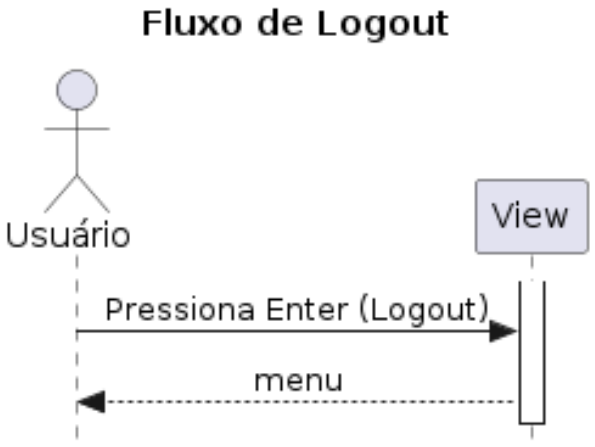

Figura 5: Diagrama de Sequência - Logout

5.4.3 Fluxo de Cadastro

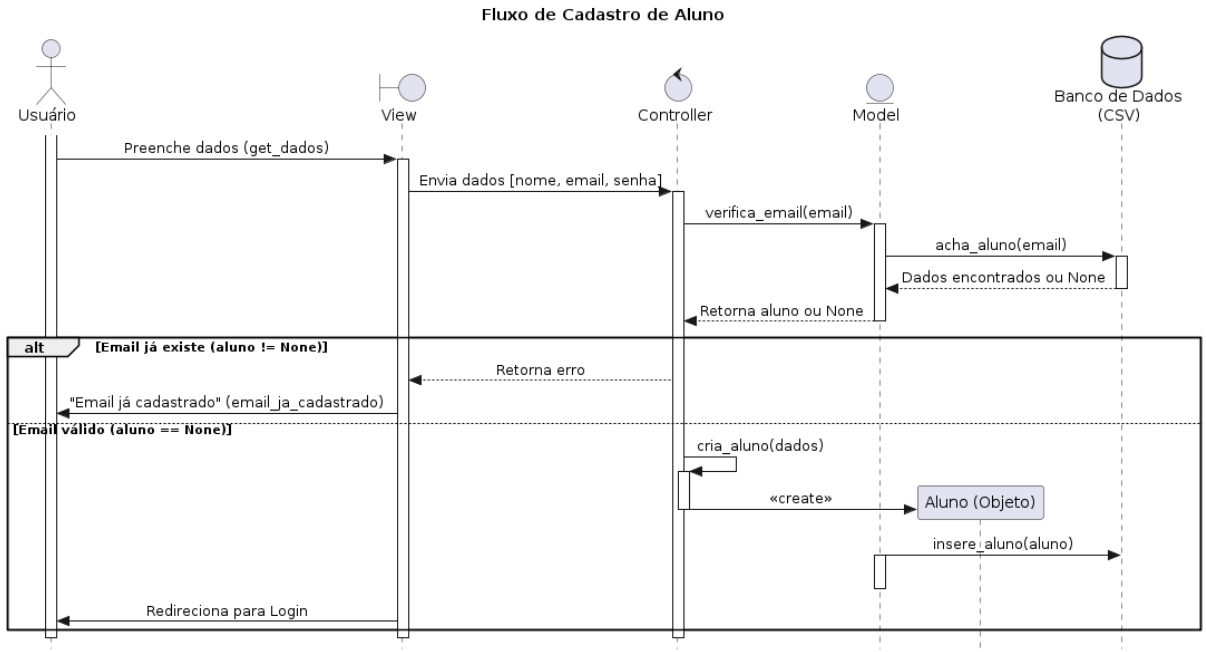

Figura 6: Diagrama de Sequência - Cadastro

5.4.4 Fluxo de Recuperação de Senha

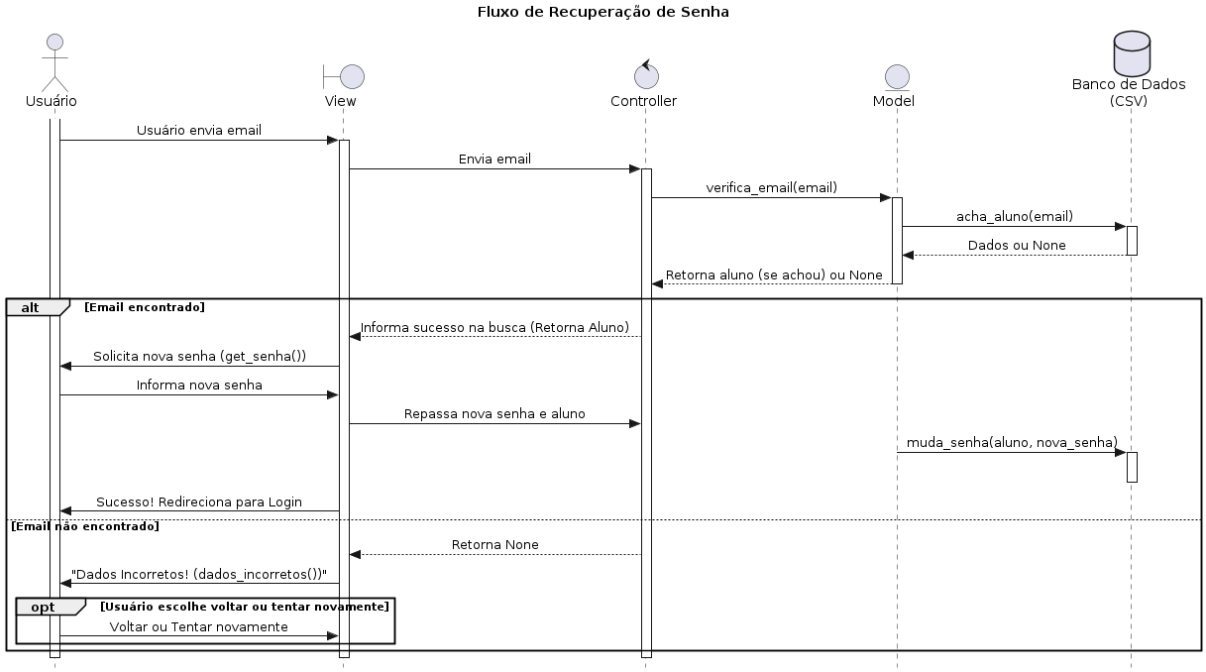

Figura 7: Diagrama de Sequência - Recuperação de Senha

6 Implementação (Código Fonte)

Conforme requisitos do projeto, abaixo apresentamos a implementação das classes e métodos desenvolvidos para o componente de Cadastro e Autenticação, a partir do planejamento realizado até o momento.

Nota Importante: Para a correta execução do código abaixo, é necessário que o arquivo `banco_de_dados.csv` esteja presente no mesmo diretório do script principal. Este arquivo deve conter, no mínimo, o cabeçalho com as colunas esperadas respectivamente (Nome, Email, Senha) para garantir a persistência e recuperação dos dados.

```
import pandas as pd

class Aluno:
    def __init__(self, dados):
        self.nome = dados[0]
        self.email = dados[1]
        self.senha = dados[2]

class View:
    def __init__(self):
        pass

    def menu(self):
        print("--------------")
        print("Bem vindo ao sistema de gerenciamento de diciplinas")
        print("1 - Cadastro")
        print("2 - Login")
        print("3 - Recuperar Senha")

        while(True):
            opcao = input("Digite sua opcao (1, 2 ou 3): ")

            if(opcao == '1'):
                return "Cadastro"
            elif(opcao == '2'):
                return "Login"
            elif(opcao == '3'):
                return "Recu_Senha"
            else:
                print("Opcao invalida!")

    def get_dados(self):
        nome = input("Digite seu nome: ")
        email = input("Digite seu email: ")
        senha = input("Digite sua senha: ")

```

```
37          return [nome, email, senha]
38
39      def tela_inicial(self, aluno):
40          print("---------------")
41          print(f"Olá, {aluno.nome}!")
42          input("Pressione Enter para fazer Logout...")
43
44      def dados_incorretos(self):
45          print("---------------")
46          print("Dados Incorretos!")
47
48          print("1 - Voltar ao Menu")
49          print("2 - Tentar novamente")
50          while(True):
51              opcao = input("Digite sua opcao (1 ou 2): ")
52
53              if(opcao == '1'):
54                  return True
55              elif(opcao == '2'):
56                  return False
57              else:
58                  print("Opcao invalida!")
59
60      def get_email(self):
61          return input("Digite seu email: ")
62
63      def get_senha(self):
64          return input("Digite sua nova senha: ")
65
66      def email_ja_cadastrado(self):
67          print("Email ja cadastrado! Tente outro")
68
69  class Controller:
70      def __init__(self):
71          pass
72
73      def cria_aluno(self, dados: list):
74          return Aluno(dados)
75
76      def verifica_email(self, email, model):
77          dados = model.acha_aluno(email)
78
79          if(dados != None):
80              return self.cria_aluno(dados)
81          else:
82              return None
83
```

```
84      def verifica_aluno(self, dados, model):
85          dados_achados = model.acha_aluno(dados[1])
86
87          if(dados_achados == None):
88              return None
89
90          if(dados_achados[2] == dados[2]):
91              if(dados_achados[0] == dados[0]):
92                  return self.cria_aluno(dados_achados)
93
94  class Model:
95      def __init__(self, banco_de_dados):
96          self.df = pd.read_csv(banco_de_dados)
97          self.df = self.df[0:0]
98          self.df.to_csv(banco_de_dados, index=False)
99
100     def acha_aluno(self, email):
101         linha = self.df.loc[self.df['Email'] == email]
102         if linha.empty:
103             return None
104
105         return linha[['Nome', 'Email', 'Senha']].iloc[0].tolist()
106
107     def insere_aluno(self, aluno):
108         self.df.loc[len(self.df)] = [aluno.nome, aluno.email, aluno.
                senha]
109         self.df.to_csv("banco_de_dados.csv", index=False)
110
111     def muda_senha(self, aluno, senha):
112         self.df.loc[self.df['Email'] == aluno.email, 'Senha'] = senha
113         self.df.to_csv("banco_de_dados.csv", index=False)
114
115 def main():
116     view = View()
117     controller = Controller()
118     model = Model("./banco_de_dados.csv")
119
120     opcao = view.menu()
121
122     while(True):
123         if(opcao == "Menu"):
124             opcao = view.menu()
125
126         elif(opcao == "Cadastro"):
127             print("--------------\nCadastro")
128             dados = view.get_dados()
129
```

```
130             #Verifica se esse email ja foi cadastrado
131             aluno = controller.verifica_email(dados[1], model)
132             if(aluno != None):
133                 view.email_ja_cadastrado()
134                 continue
135 
136             aluno = controller.cria_aluno(dados)
137 
138             model.insere_aluno(aluno)
139             opcao = "Login"
140 
141         elif(opcao == "Login"):
142             print("--------------\nLogin")
143             dados = view.get_dados()
144 
145             aluno = controller.verifica_aluno(dados, model)
146 
147             if(aluno != None):
148                 view.tela_inicial(aluno)
149 
150                 opcao = "Menu"
151             else:
152                 mudar = view.dados_incorretos()
153                 if(mudar == True):
154                     opcao = "Menu"
155         elif(opcao == "Recu_Senha"):
156             print("--------------\nRecuperar Senha")
157             email = view.get_email()
158 
159             aluno = controller.verifica_email(email, model)
160 
161             if(aluno != None):
162                 senha = view.get_senha()
163 
164                 model.muda_senha(aluno, senha)
165 
166                 opcao = "Login"
167             else:
168                 mudar = view.dados_incorretos()
169                 if(mudar == True):
170                     opcao = "Menu"
171 
172 if __name__ == "__main__":
173     main()
```

Listing 1: Código Fonte - main.py

7 Discussão

Esta seção descreve as dificuldades e desafios encontrados durante a execução do projeto, bem como os principais aprendizados obtidos pelo grupo.

7.1 Dificuldades e Desafios

Um dos principais desafios encontrados pelo grupo foi a adaptação ao processo rigoroso de modelagem antes da implementação, uma vez que a equipe, inicialmente, não estava acostumada a realizar um planejamento tão extenso e detalhado antes de iniciar a codificação propriamente dita.

Especificamente, a elaboração dos diagramas do componente escolhido (Cadastro e Autenticação) apresentou dificuldades significativas:

- **Diagrama de Classes:** Uma dificuldade central foi a visualização e definição das classes não explícitas no domínio do problema, como *Controller*, *View* e *Model*. Inicialmente, o grupo tendeu a focar apenas nas entidades de dados (como Aluno), sem perceber a necessidade de classes de controle e interface para um bom design de software. Ao perceber a necessidade de mais classes para garantir o desacoplamento do código e a separação de responsabilidades (padrão MVC), o grupo optou por uma estratégia diferente: modelar primeiramente os diagramas de sequência, permitindo visualizar a troca de mensagens necessária para realizar cada caso de uso, o que, por sua vez, revelou naturalmente quais classes (e métodos) seriam necessários. Somente após essa etapa é que o diagrama de classes foi consolidado, com a adição de informações estruturais importantes como as multiplicidades e os tipos de relacionamento.

- **Diagramas de Sequência:** A modelagem da interação entre os objetos para realizar os casos de uso (Login, Cadastro, etc.) também foi desafiadora. Garantir que a troca de mensagens entre as classes (View, Controller, Model/Aluno) estivesse coerente com o objetivo do projeto e cobrisse todos os fluxos (principal e alternativos) demandou diversas revisões.

7.2 Aprendizados Importantes

Apesar das dificuldades iniciais, a execução deste projeto proporcionou aprendizados valiosos, destacando-se o ganho de percepção sobre a importância da modelagem de software:

- **Visão Geral do Projeto:** O grupo compreendeu que modelar o projeto, especialmente nas etapas iniciais de definição dos Casos de Uso e elaboração do Diagrama de Casos de Uso, é fundamental para obter uma visão clara e abrangente do sistema,

permitindo identificar requisitos e funcionalidades que poderiam ter passado despercebidos se tivéssemos partido direto para o código. Adicionalmente, a elaboração do diagrama de componentes foi essencial para visualizar como as diferentes partes do sistema interagem entre si, mesmo que apenas um componente tenha sido implementado em código, essa modelagem permitiu compreender a arquitetura geral e a importância do desacoplamento para a manutenibilidade e escalabilidade do software, demonstrando como as dependências entre os módulos devem ser gerenciadas em um contexto mais amplo.

- **Planejamento como Guia:** A experiência demonstrou na prática que o tempo investido no planejamento e na modelagem (UML) resulta em uma implementação mais organizada e menos sujeita a erros estruturais. O código final tornou-se uma consequência natural das decisões tomadas durante a fase de design, validando a importância dessa etapa.

8 Uso de IA Generativa

Esta seção descreve o uso de ferramentas de Inteligência Artificial Generativa no desenvolvimento deste trabalho, detalhando as ferramentas empregadas e refletindo sobre o impacto no aprendizado.

8.1 Fins de Utilização e Ferramentas

A Inteligência Artificial Generativa, especificamente o modelo de linguagem **Gemini**, foi utilizada em diversas etapas estratégicas do desenvolvimento deste relatório e do projeto como um todo:

- **Geração de Diagramas em PlantUML:** O Gemini foi utilizado para gerar o código PlantUML correspondente ao Diagrama de Casos de Uso, traduzindo os requisitos textuais levantados pelo grupo para a sintaxe correta da ferramenta de diagramação.

- **Estruturação e Diagramação do Relatório (LaTeX):** A IA foi fundamental para criar a estrutura inicial do relatório em LaTeX. Com base nas etapas do projeto já modeladas e escritas pelo grupo, o modelo auxiliou na organização do conteúdo, criação de seções, formatação de listas e inserção de imagens.

- **Modelagem de Descrições Estendidas:** As descrições estendidas dos casos de uso também contaram com o auxílio da IA para sua formatação e detalhamento inicial (embora os prompts específicos utilizados nesta etapa não tenham sido recuperados para inclusão neste documento).

8.2 Reflexão sobre o Aprendizado

O grupo avaliou o uso da IA Generativa como **positivo**. A principal vantagem percebida foi a facilitação do trabalho mecânico e repetitivo, como a codificação da sintaxe LaTeX ou PlantUML, permitindo que os integrantes focassem seus esforços nas atividades intelectuais de análise e design.

No entanto, é crucial destacar que o uso da ferramenta não substituiu o domínio do conteúdo por parte do grupo.

- **Análise Crítica Ativa:** Tanto nos diagramas quanto nos casos de uso extendidos gerados, o grupo precisou exercer uma revisão crítica constante. Os resultados gerados pela IA frequentemente precisavam de ajustes e correções para representar fielmente o design desejado e as especificidades do projeto, exigindo que tivessemos clareza sobre os conceitos apresentados na disciplina.

- **Modelagem de Casos de Uso Estendidos:** Adicionalmente, para a geração das descrições estendidas, foi necessário primeiro elaborar manualmente um "caso de uso base"robusto para servir de exemplo no prompt. Isso demonstrou que a IA funcionou como uma alavanca de produtividade, mas a base intelectual partiu do conhecimento prévio do grupo sobre como um caso de uso deve ser estruturado.

- **Produtividade na Documentação:** O uso da IA para a geração do relatório LaTeX foi particularmente benéfico, uma vez que o conteúdo do trabalho (textos, códigos, modelagens) já estava quase totalmente desenvolvido e estruturado pelo grupo, a ferramenta serviu como um eficiente meio de diagramação e formatação. Ela agilizou a tarefa de transformar o conteúdo bruto em um documento bem formatado, lidando com a complexidade da sintaxe LaTeX e permitindo que o foco permanecesse na qualidade das informações apresentadas, e não apenas na formatação do documento.

Em conclusão, o uso da IA diminuiu a carga de trabalho operacional sem desincentivar o aprendizado. Ela atuou como uma ferramenta de apoio que, ao remover barreiras mecânicas de formatação e sintaxe, permitiu ao grupo dedicar mais tempo à qualidade da modelagem e à lógica do sistema.

8.3 Links para Prompts Utilizados

Abaixo estão os links para as conversas e prompts utilizados durante a elaboração do projeto:

- https://gemini.google.com/share/970ab50a11d8
- https://gemini.google.com/share/39e31d6d6178