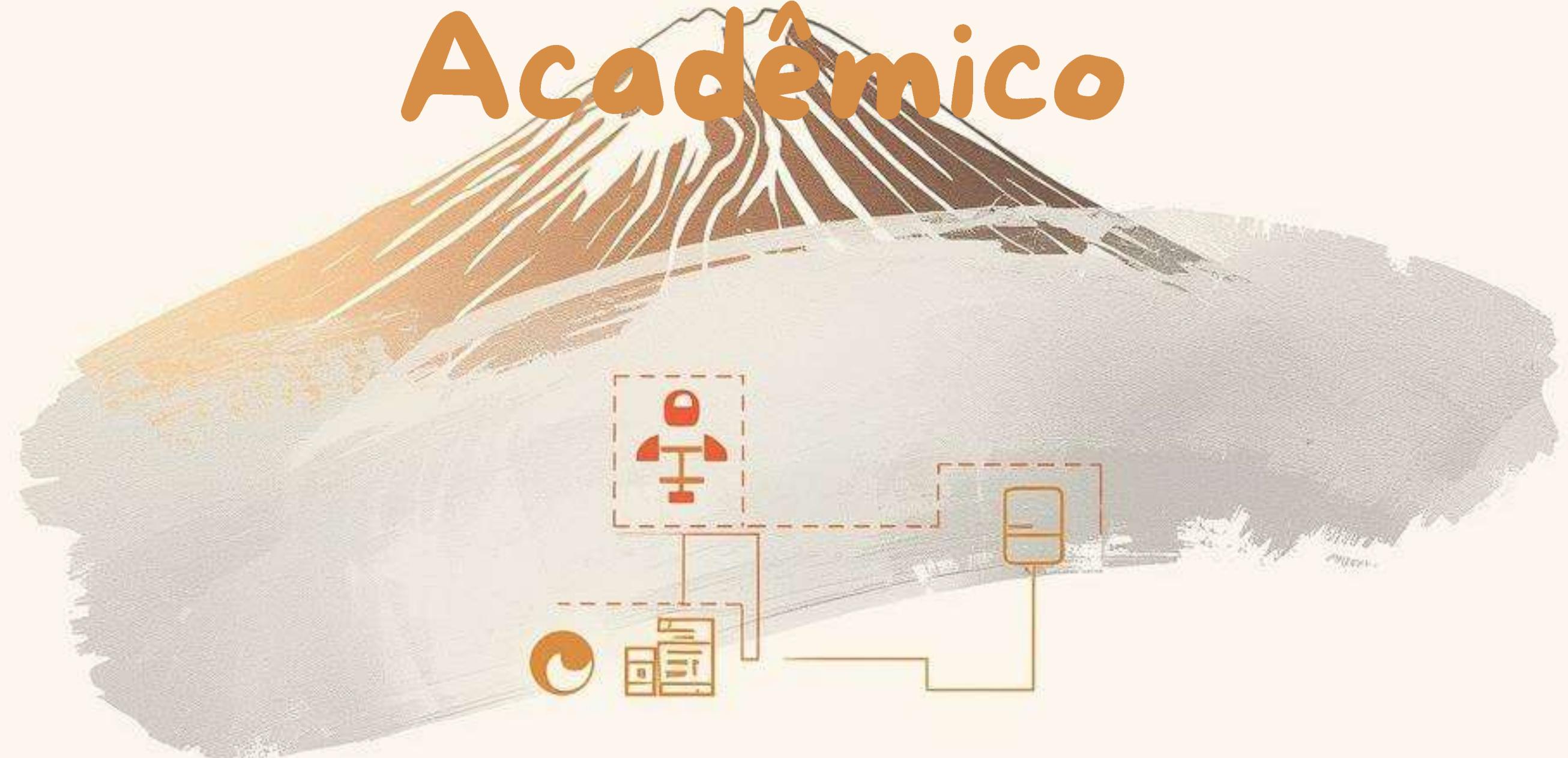


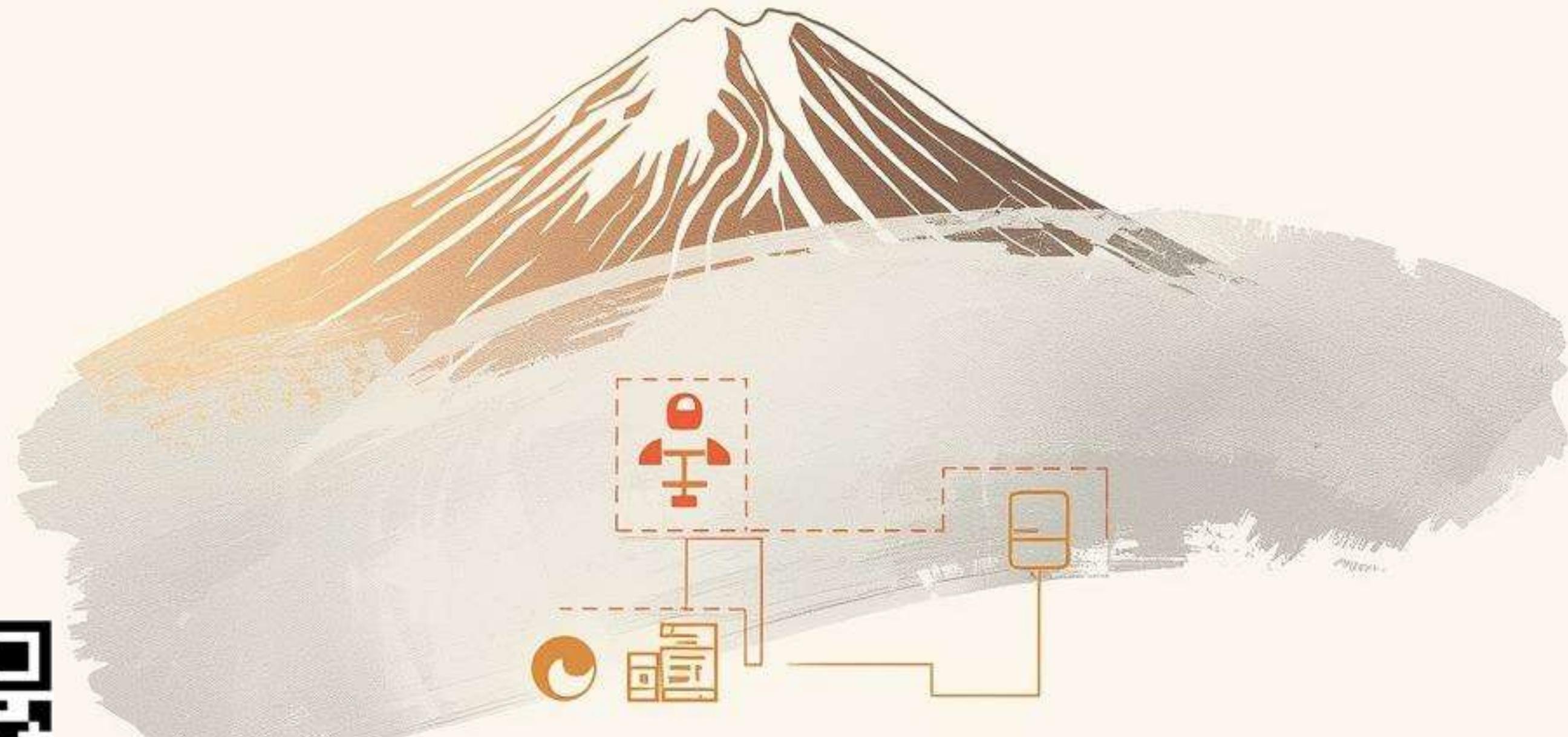
Sistema Acadêmico



Danilo Azevedo | Adriano Dias | Eduardo Juan | Rafael Botti | João Acerbi



Link para o código



Link para o Github:

https://github.com/ejuanoli/UNIP_ADS.git

* Observação: o Manual do Usuário está nos arquivos do github, assim como os outros arquivos solicitados.

筑波山記念講堂

TEMA

Uma instituição de ensino necessita de um sistema colaborativo para apoiar professores e alunos no gerenciamento de turmas, aulas e atividades. Atualmente, controles são realizados de forma descentralizada (planilhas, e-mails, mensagens em aplicativos). O sistema deve permitir cadastro de turmas e alunos, registro de aulas e diário eletrônico, upload e consulta de atividades, e módulos distribuídos em uma rede. Um dos objetivos é a eliminação do uso de papel pelos professores como medidas sustentáveis.

SUMÁRIO

- Biblioteca utilizadas
- Arquitetura do Sistema
- Detalhamento das funções
- Conexão entre Servidor e Cliente
- Banco de dados
- Design do Programa
- Diferenciais
- Créditos

BIBLIOTECAS UTILIZADAS NO SISTEMA

Gerenciamento de Dados e Sistema

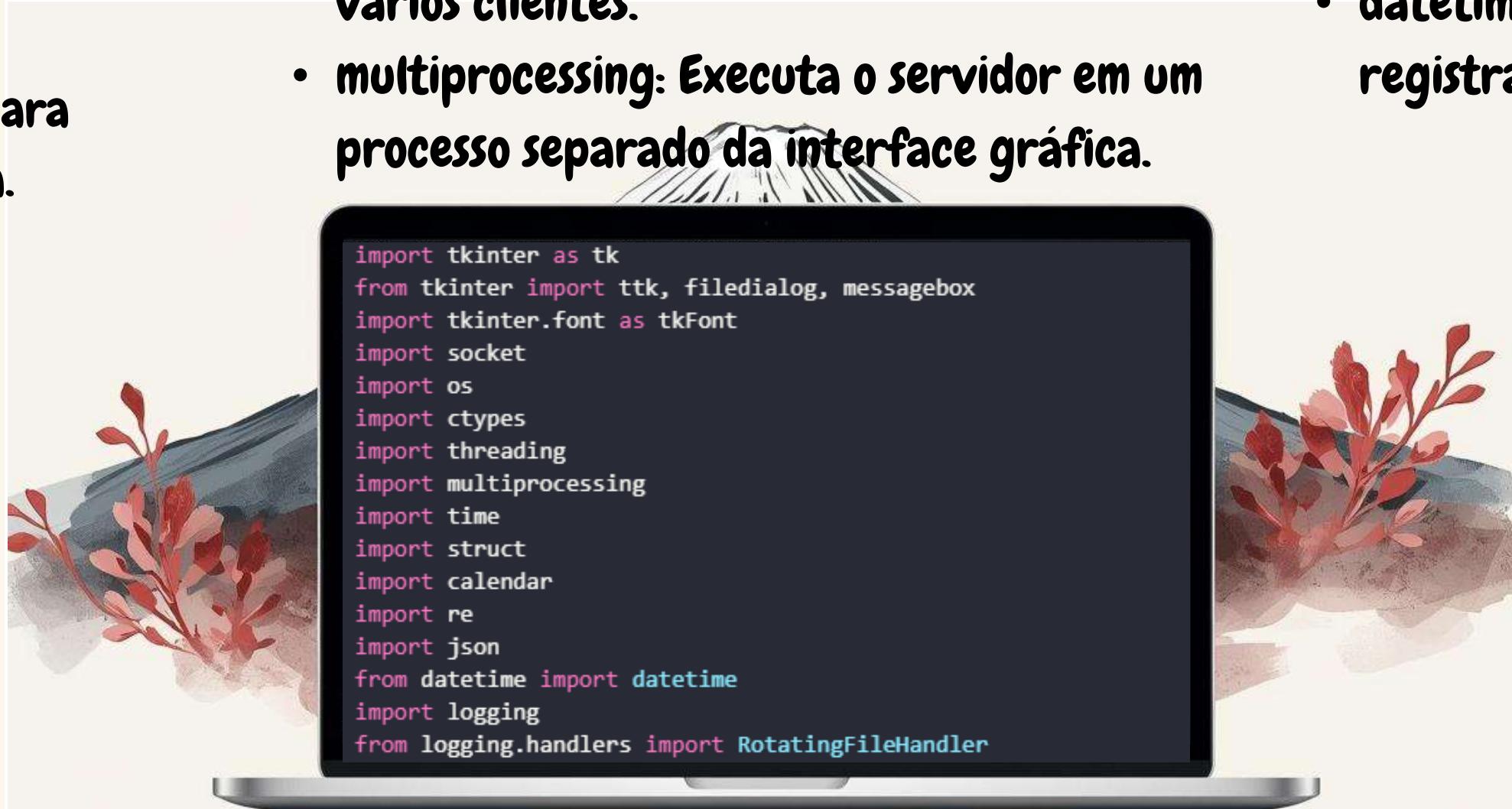
- **os:** Gerencia arquivos e pastas do sistema operacional.
- **ctypes:** Permite que o Python use o banco de dados feito em linguagem C.
- **json:** Salva e lê configurações e dados de usuários de forma organizada.
- **struct:** Grava e lê dados binários, como notas e presenças.
- **logging:** Cria registros (logs) para depurar e monitorar o sistema.

Interface e Comunicação

- **tkinter:** Constrói toda a interface gráfica do sistema (janelas, botões, menus).
- **socket:** Permite a comunicação de rede entre a aplicação e o servidor.
- **threading:** Executa tarefas simultâneas para que a interface não trave e o servidor atenda vários clientes.
- **multiprocessing:** Executa o servidor em um processo separado da interface gráfica.

Execução e Utilitários

- **time:** Fornece funções de tempo, como a criação de timestamps.
- **calendar:** Oferece funções de calendário, como a configuração da semana.
- **re:** Valida formatos de texto, como e-mails e senhas (através de expressões regulares).
- **datetime:** Manipula datas e horas para registrar eventos como o login.



ARQUITETURA DO SISTEMA

Frontend (Interface Gráfica):

- Construída em Python com a biblioteca tkinter.
- Dividida em duas partes principais: uma tela de Login e a Aplicação Principal (dashboard).
- Possui um design flexível com suporte a temas claro e escuro.

Backend (Servidor):

- Executa em um processo separado com multiprocessing para não travar a interface do usuário.
- Usa threading para conseguir atender múltiplos usuários ao mesmo tempo sem lentidão.
- Recebe os comandos, processa a lógica e gerencia o acesso aos dados.

Comunicação Cliente-Servidor:

- A interface (cliente) envia e recebe dados do backend (servidor) através de socket.
- Possui um modo offline: caso o servidor não responda, o sistema utiliza dados salvos localmente para continuar funcionando.

Camada de Dados (Como as informações são guardadas):

- Banco de Dados Principal (em C): O núcleo com turmas e alunos é gerenciado pela biblioteca em C (.dll ou .so), acessada pelo Python com ctypes.
- Dados de Usuários (users.json): Informações de login, senhas, perfis e permissões são salvas em um arquivo JSON.
- Dados Binários (.dat): Notas e presenças são armazenadas em arquivos binários para maior performance e integridade, lidos com struct.
- Arquivos de Upload: Atividades e outros documentos são salvos diretamente em pastas no sistema de arquivos.

FUNCIONALIDADES

Autenticação e Perfis:

- **Login Seguro:** Sistema de autenticação com cadastro, recuperação de senha e perfis de usuário (Aluno, Professor, Administrador).
- **Aprovação de Cadastros:** Novos usuários ficam pendentes até que um administrador aprove o acesso, garantindo controle.

Recursos e Ferramentas:

- **Gerenciador de Arquivos:** Professores podem fazer upload de materiais e atividades, que podem ser baixados pelos usuários da turma.
- **Anotações Pessoais:** Um bloco de notas integrado para que cada usuário possa criar, salvar e gerenciar suas próprias anotações.
- **Interface Personalizável:** Suporte a temas claro e escuro para maior conforto visual do usuário.
- **Relatórios e Diagnósticos:** Ferramentas para administradores visualizarem o estado do sistema e exportarem dados.

Gerenciamento Acadêmico:

- **Gestão de Turmas e Alunos:** Permite criar, editar, listar e excluir turmas, além de gerenciar os alunos dentro delas.
- **Lançamento de Notas:** Interface para professores lançarem as notas (NPI, NP2, PIM) e calcularem a média automaticamente, assim como um status de aprovado/reprovado com lógica para a nota de exame.
- **Controle de Presença:** Ferramenta para registrar a presença dos alunos em uma data específica, utilizando um calendário visual.
- **Calendário de Provas:** Permite agendar e consultar as datas das avaliações de cada turma.

CONEXÃO ENTRE SERVIDOR E CLIENTE

Interface do Usuário (Cliente):

- A Interface Gráfica (cliente) se conecta ao servidor usando o endereço de IP e a porta definidos no código (ex: 10.101.108.58, porta 65432).
- Cada ação do usuário (como "Listar Turmas" ou "Salvar Notas") é enviada como um comando de texto via socket.
- A interface aguarda uma resposta do servidor para exibir os dados na tela ou confirmar uma ação.

Plano de Contingência (Fallback):

- O sistema primeiro tenta se conectar ao servidor na rede. Se a conexão falhar, ele ativa um modo de fallback.
- Nesse modo, a interface para de tentar se conectar e passa a ler e salvar os dados em arquivos locais (`turmas_local.json`, `alunos_local.json`), permitindo o uso offline.



Servidor (Backend):

- O Servidor fica constantemente "escutando" na rede local, esperando por conexões de qualquer interface.
- Ao receber uma conexão, o servidor cria uma thread dedicada para atender aquele usuário, permitindo que vários se conectem ao mesmo tempo.
- Ele interpreta o comando recebido (ex: "ADD_TURMA..."), executa a ação correspondente e acessa os bancos de dados para buscar ou salvar as informações.



Acesso aos Bancos de Dados:

- Banco de Dados C (.dll/.so): Apenas o servidor tem acesso direto a este banco. Ele usa ctypes para chamar as funções que gerenciam turmas e alunos.
- Arquivos JSON e DAT: O servidor também é o responsável por ler e escrever nos arquivos de usuários (`users.json`), notas (`notas.dat`), presenças e outros, garantindo que os dados fiquem centralizados.
- A interface do usuário nunca acessa os bancos de dados diretamente; ela apenas solicita e recebe os dados prontos do servidor.

BANCO DE DADOS

Banco de Dados em C:

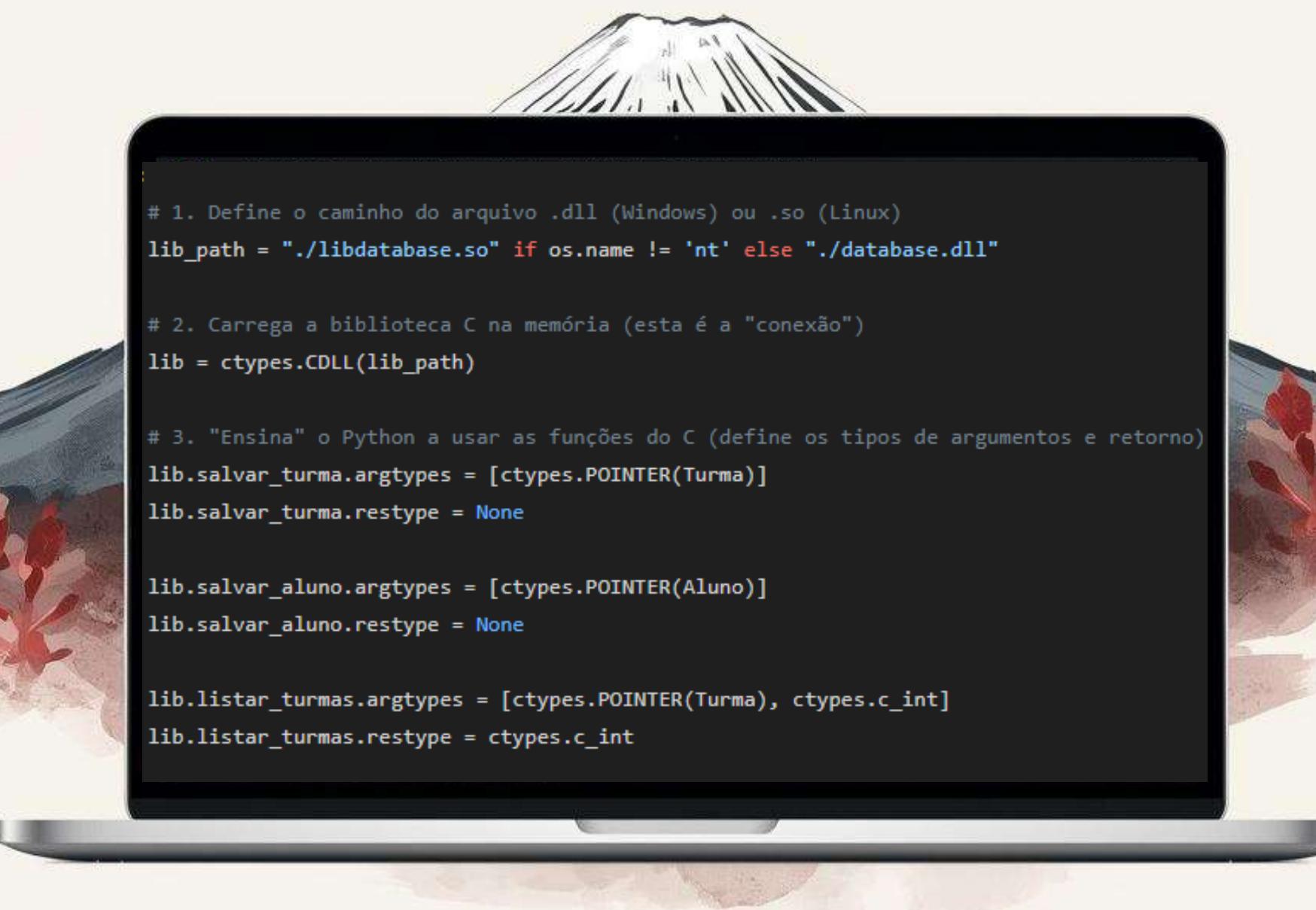
- É o núcleo que gerencia os dados principais (turmas e alunos), salvando-os diretamente em arquivos binários.
- Usa structs (estruturas) para definir como os dados são organizados na memória e nos arquivos.
- Contém toda a lógica para criar, ler, atualizar e deletar os registros acadêmicos de forma eficiente.

O Arquivo DLL(.dll):

- É o arquivo final, o resultado do código C compilado, que o Python consegue executar.
- A interface em Python não lê o código C; ela chama as funções que estão prontas dentro deste arquivo.
- Funciona como o "motor" do banco de dados, enquanto o Python (com ctypes) atua como o "painel de controle" que dá os comandos.

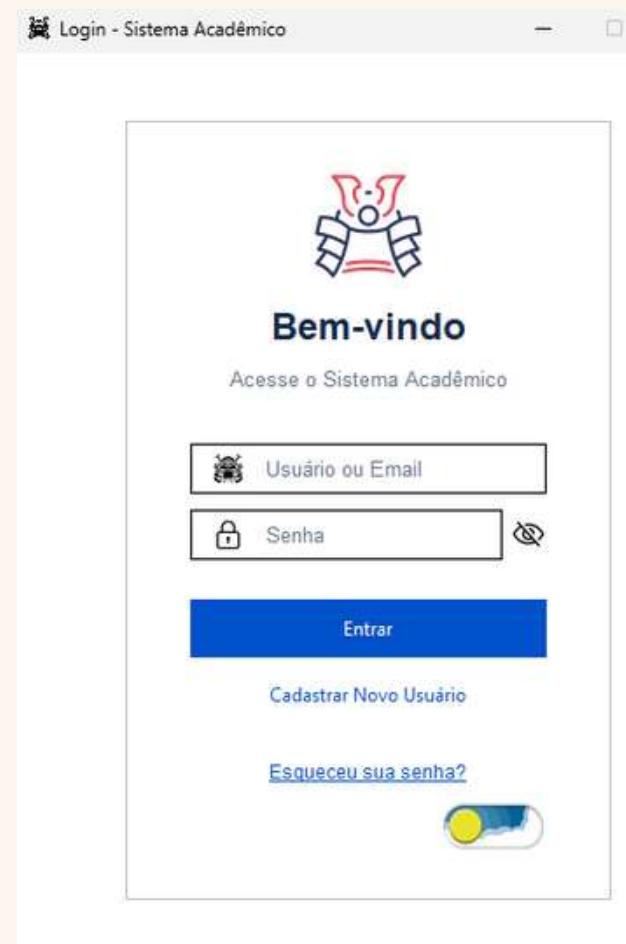
O Arquivo H (.h):

- É o "manual de instruções" do arquivo DLL.
- Ele descreve as structs e as funções que existem no código C, mas não contém a lógica delas.
- O Python usa essa "descrição" para saber exatamente como se comunicar com o DLL, entendendo quais dados enviar e o que esperar de volta.



DESIGN DO PROGRAMA

Interface do menu de login



Interface principal, com as funcionalidades

The screenshot shows the main interface of the academic system. The title bar reads 'Sistema Acadêmico - Admin: admin'. The main area is titled 'Lista de Turmas' and displays a table with one row of data:

ID	Disciplina	Professor	Turno	Total Alunos
1	matemática	tulio	Matutino	2

On the left, there is a sidebar with several functional buttons: 'Gerenciamento de Turmas' (Cadastrar Turma, Listar Turmas, Pesquisar Professor, Editar Turma, Calendário de Provas), 'Gerenciamento de Alunos' (Cadastrar Aluno, Listar Alunos (por Turma), Editar Aluno, Pesquisar Aluno, Lançar Notas, Controle de Presença), 'Atividades' (Upload de Atividade, Gerenciar Atividades), and 'Anotações' (Gerenciar Anotações). The status bar at the bottom right indicates 'Segunda-feira, 27/10/2025 - 13:41:30'.

Ícones dinâmicos e amigáveis, com estilização moderna.



Design baseado no tema do projeto do primeiro semestre,



Pim I:



DIFERENCIAIS DO PROGRAMA

Arquitetura Híbrida e Resiliente:

- Combina um backend de alta performance em C com uma interface flexível em Python.
- Possui um modo offline que ativa automaticamente se o servidor falhar, garantindo o funcionamento contínuo.
- O servidor é paralelo e multithreaded, permitindo múltiplos acessos simultâneos sem travar a interface.

Interface Moderna e Flexível:

- Suporte a temas claro e escuro com componentes visuais que se adaptam.
- Barra lateral (sidebar) retrátil para maximizar o espaço de visualização de dados.
- Componentes de interface customizados (janelas, botões) que se integram perfeitamente aos temas.

Segurança e Controle Administrativo:

- Fluxo de aprovação de cadastros: novos usuários precisam da liberação de um administrador para acessar o sistema.
- Senhas criptografadas (hash) para proteger as contas de usuário.

Diagrama de Atividades

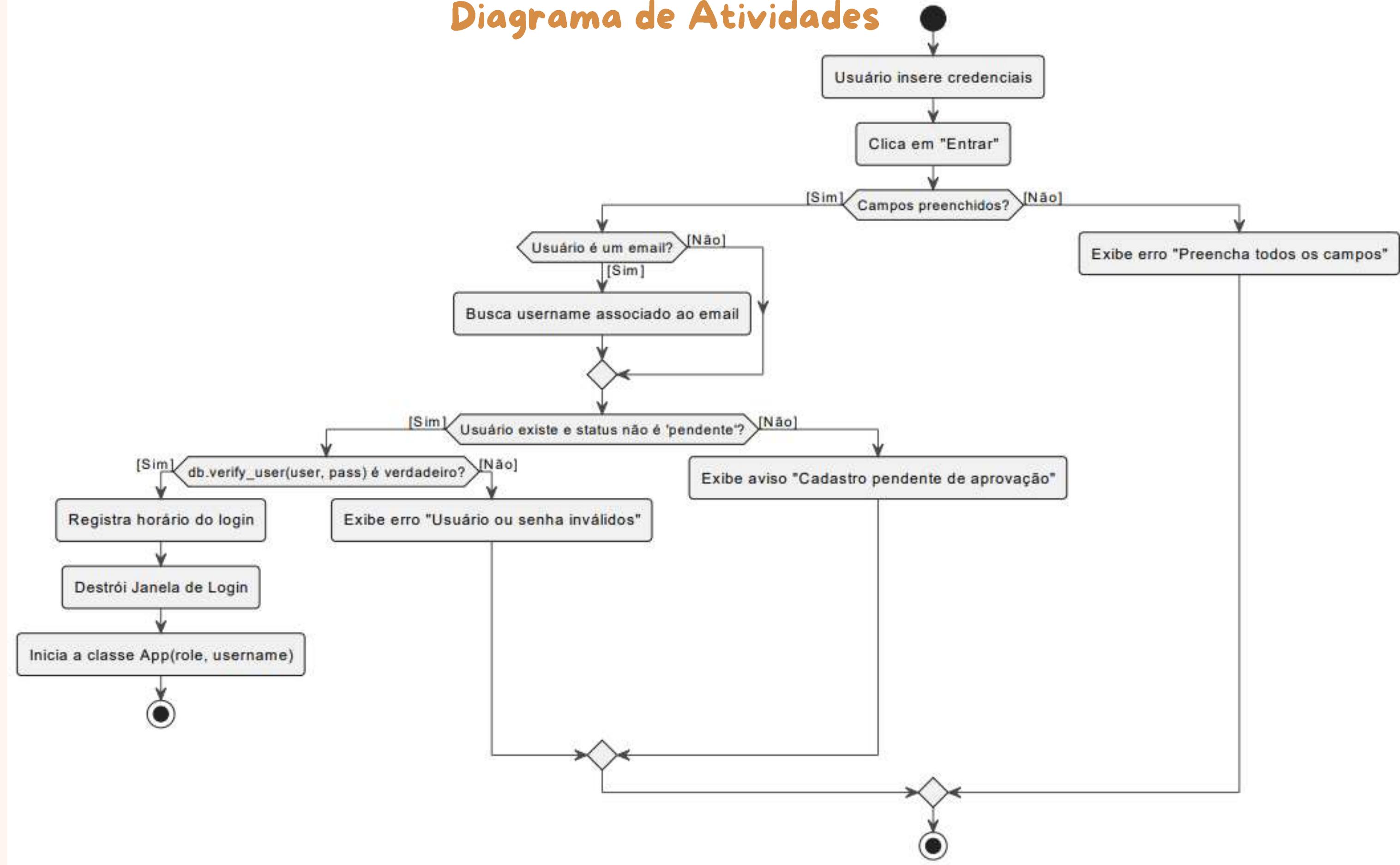


Diagrama de Classes

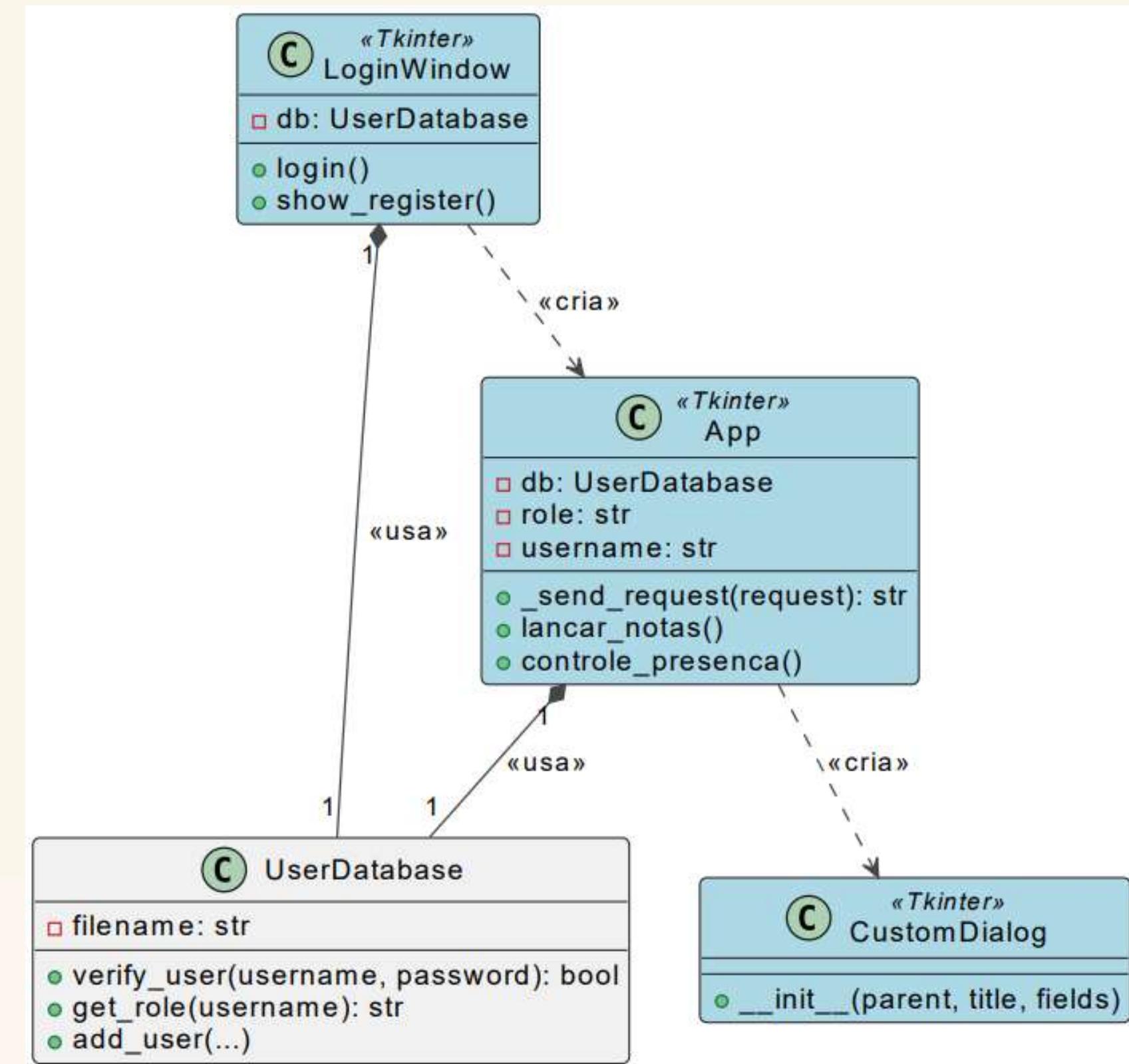


Diagrama de Sequencias

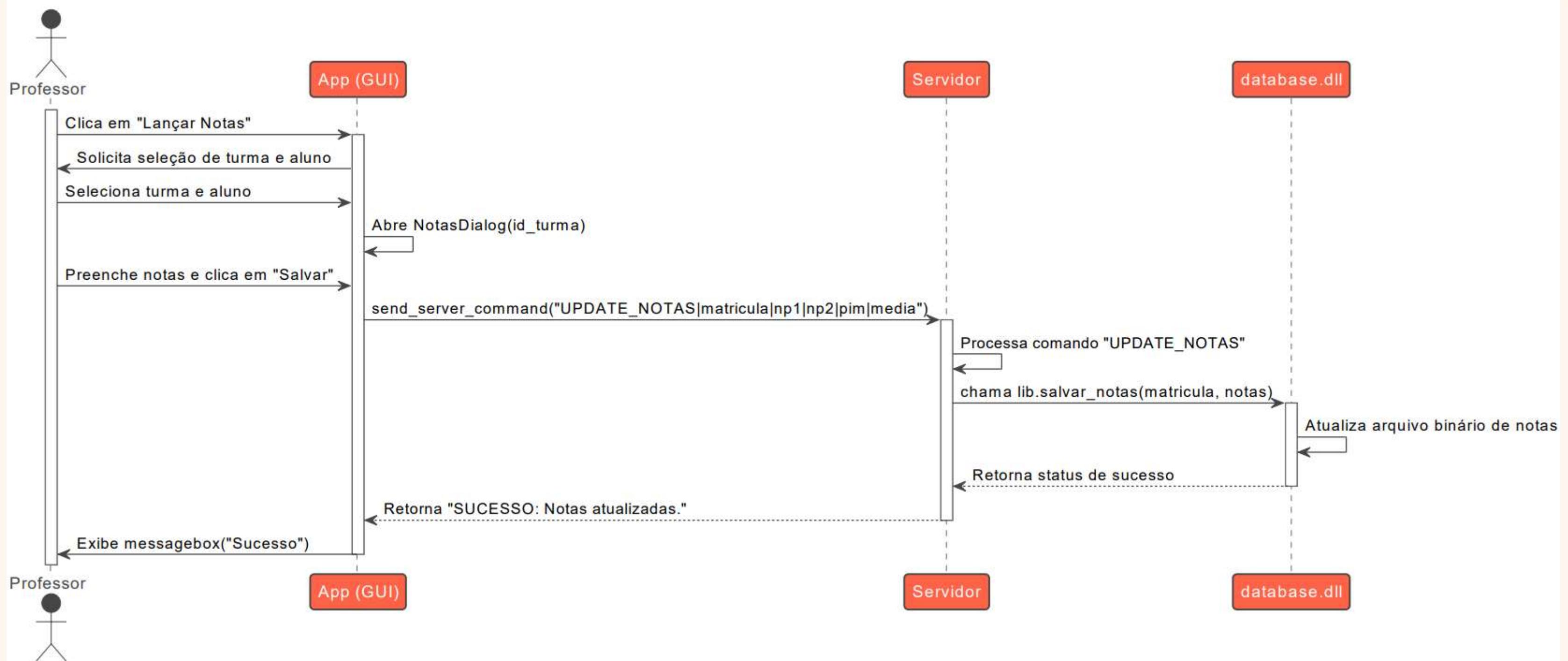


Diagrama de Casos de Uso

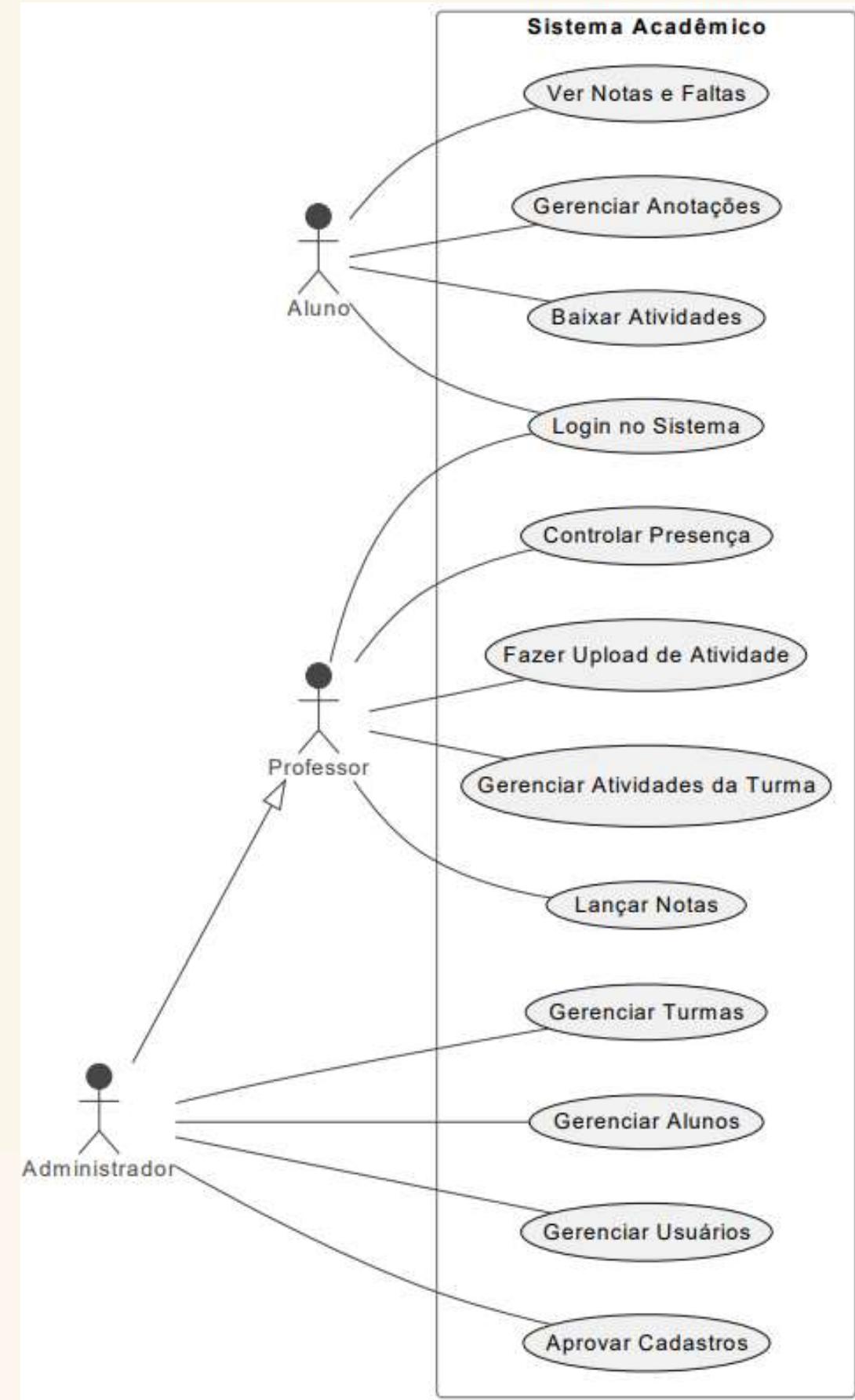
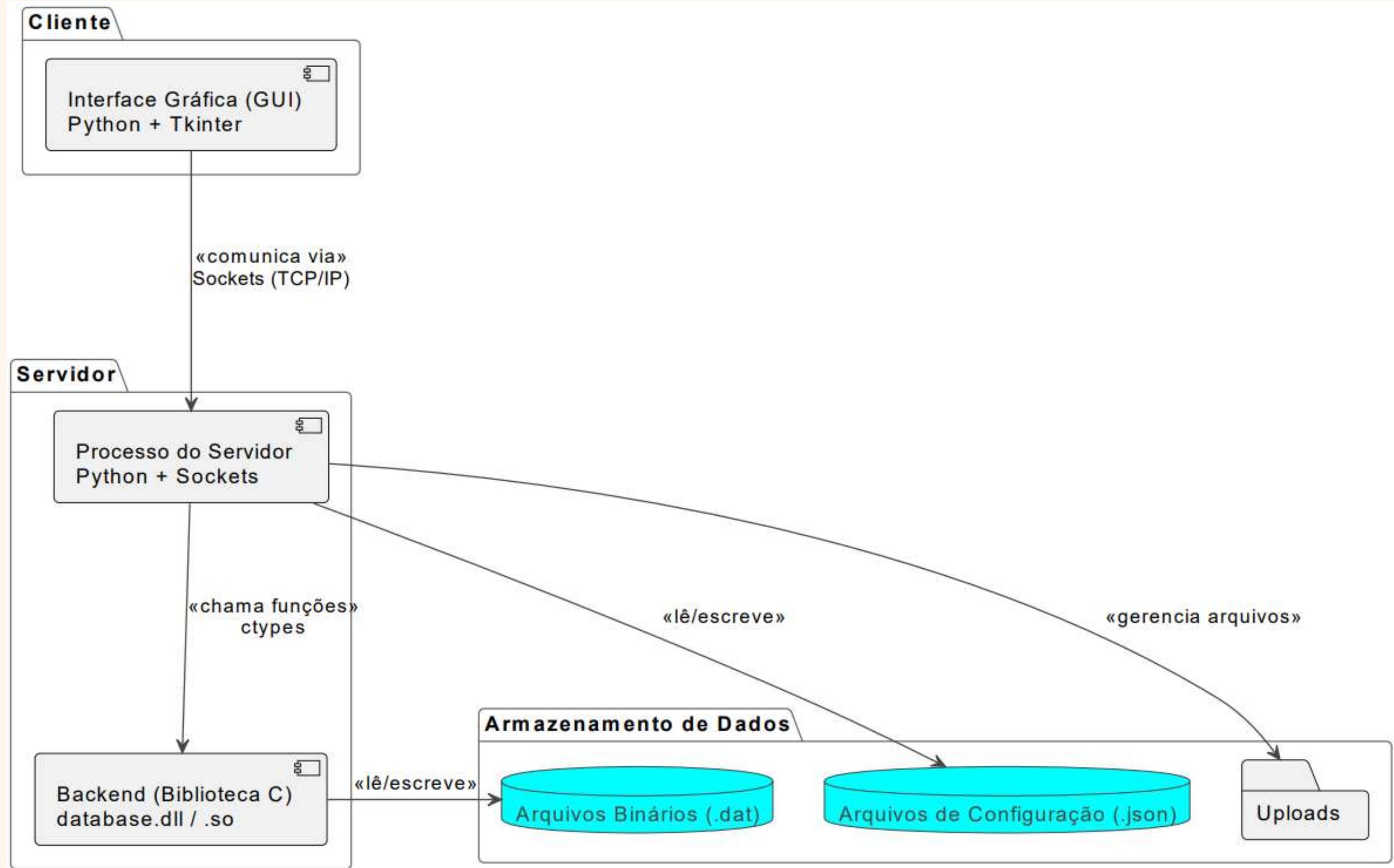


Diagrama de Arquitetura

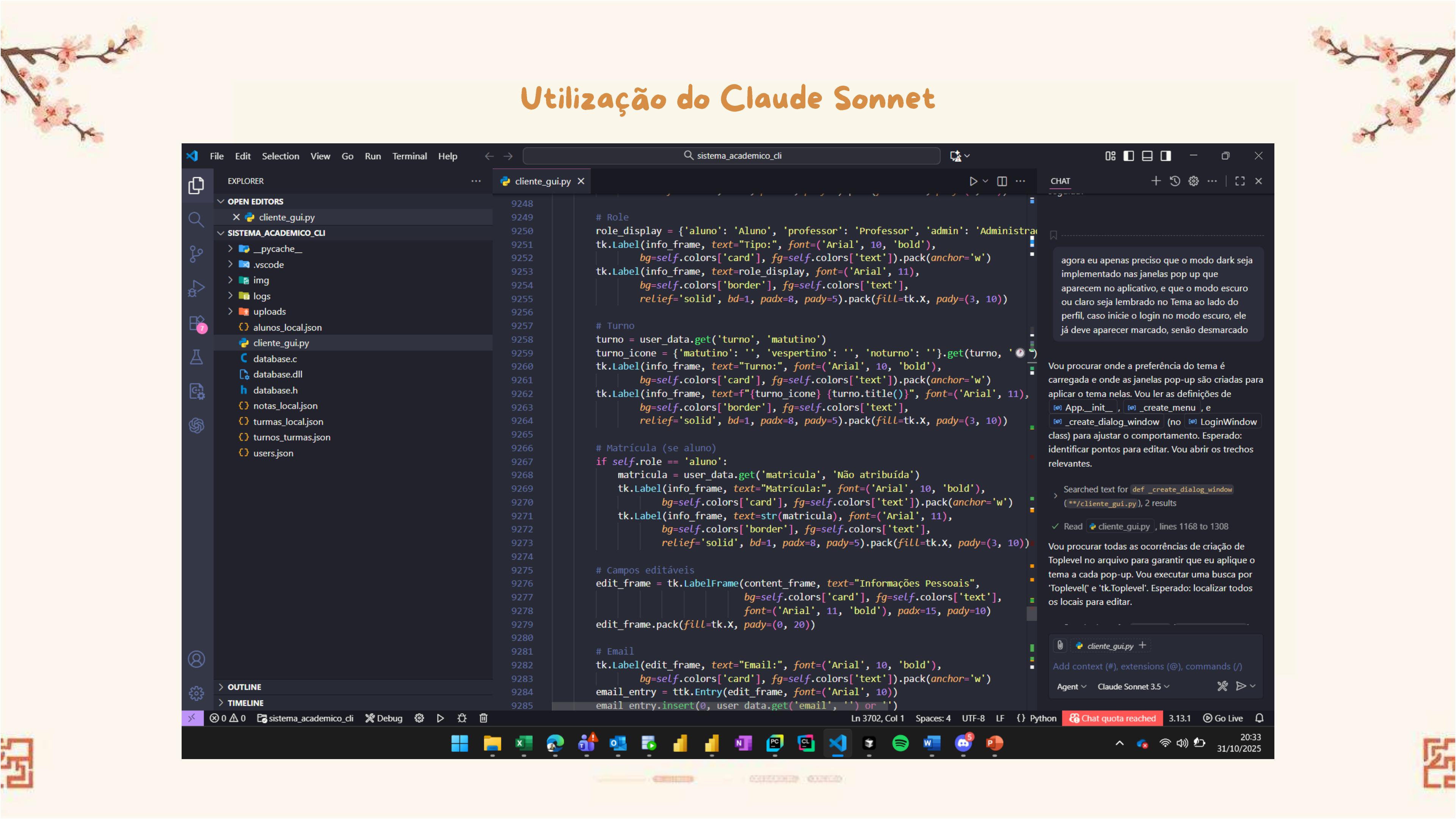


Utilização do Claude Sonnet

The screenshot shows a code editor interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** cliente_gui.py - sistema_academico_cli - Cursor.
- Left Sidebar:** Shows a project structure under 'SISTEMA_ACADEMICO_CLI' with files like __pycache__, .vscode, img, uploads, cliente_gui.py, database.c, database.dll, database.h, and users.json.
- Code Area:** The main pane displays Python code for a client application. It includes logic for handling file operations (saving, exporting, deleting notes), managing a server process, and displaying error messages using messagebox.showerror().
- Right Sidebar:**
 - A panel titled 'adapte o código para trabalhar apenas' contains a list of tasks:
 - Adicionar gerenciamento de usuários no servidor (UserDatabase integrado)
 - Adicionar comandos de servidor para operações de usuários (login, criação, atualização)
 - Adicionar gerenciamento de provas, turnos e exames no servidor
 - Adicionar gerenciamento de anotações no servidor
 - Remover todas as funções de carregamento/salvamento local do cliente
 - Adaptar o cliente para fazer todas as requisições via socket
 - A 'To-dos' section lists 6 items.
 - Search results:
 - Read cliente_gui.py L1-100
 - Thought for 1s
 - Read cliente_gui.py L260-349
 - Read cliente_gui.py L300-374
 - Grepped load_provas|save_provas|load_turnos|save_t...
 - Thought for 1s
 - Grepped get_nota_exame|set_nota_exame|get_turno_...
 - Read cliente_gui.py L9670-9769
 - Thought for 1s
 - Bottom status bar: @ 1 Tab + Browser 34.8%.
 - Bottom footer: Plan, @ for context, / for commands.

Utilização do Claude Sonnet



The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with a Python file named `cliente_gui.py` open in the center editor tab. The file contains code for a Tkinter-based application, specifically for displaying user information and managing a login window.

The code includes sections for displaying user roles (aluno, professor, admin), showing the current turn (turno), displaying matrícula (student ID), creating editables fields, and handling email input. It uses the `tk` and `ttk` modules for the graphical interface.

In the bottom right corner of the code editor, there is a "CHAT" panel where a message from Claude Sonnet is displayed:

```
agora eu apenas preciso que o modo dark seja implementado nas janelas pop up que aparecem no aplicativo, e que o modo escuro ou claro seja lembrado no Tema ao lado do perfil, caso inicie o login no modo escuro, ele já deve aparecer marcado, senão desmarcado
```

Below this message, there are two search results related to the `_create_dialog_window` method:

- Searched text for `def _create_dialog_window`
(<**/cliente_gui.py>), 2 results
- Read `cliente_gui.py`, lines 1168 to 1308

Further down, another message from Claude Sonnet asks to search for `Toplevel` in the code to ensure it applies the theme correctly to pop-up windows.

The VS Code interface also includes a sidebar with icons for Explorer, Search, and Outline, and a status bar at the bottom showing file paths, line numbers, and system status.

Utilização Gemini Pro

The screenshot shows the Google AI Studio interface with the following details:

- Left Sidebar:** Includes links for Home, Chat (highlighted), Diagrama UML Do Código, GitHub README with Code Lines, Swap Largest and Smallest Ele..., Excel Update Guide For TASL, Observações no Power BI: Tool..., View all history →, Build, Dashboard, Documentation, and a note about model mistakes.
- Top Bar:** Shows the project name "Samurai and Datetime Widget Placement", token count "270.590 tokens", and various UI icons.
- Central Chat Area:** Displays a message from the user asking about centering a ToggleButton icon vertically, followed by a response from the AI providing Python code to fix a Tkinter import issue.
- Right Sidebar:** Features sections for Gemini 2.5 Pro (model selection), System instructions (optional tone and style), Temperature slider (set to 1), Media resolution dropdown (Default), Thinking mode toggle (off), Set thinking budget toggle (off), Tools section, and Structured output toggle (off).

CRÉDITOS

Obrigado pela atenção!

Responsáveis:

Adriano Dias Junior	H7451E8
João Acerbi	R855DD4
Eduardo Juan	R590AJ8
Rafael Botti	R8497A1
Danilo Azevedo	R660648

Matrícula:

Alunos do segundo semestre de Análise e Desenvolvimento de Sistemas na UNIP, em Jundiaí.

Tecnologias Utilizadas:

Linguagens:

- Python 3.x: Linguagem principal para a interface, lógica do cliente e orquestração do sistema.
- C (DLL/SO): Linguagem usada para o backend de alta performance, gerenciando o banco de dados principal.

Armazenamento de dados:

- JSON: Formato utilizado para armazenar dados de usuários, configurações e anotações de forma legível.
- Arquivos Binários (.dat): Formato eficiente para salvar dados estruturados como notas e presenças, otimizando performance e espaço.

Inteligência Artificial (Desenvolvimento):

- Cursor: Editor de código com IA, utilizado para acelerar o desenvolvimento, refatorar e depurar o código.
- Gemini: Modelo de IA generativa, usado para auxiliar na criação da arquitetura, documentação e na resolução de problemas complexos de lógica.



Link para o Github:

https://github.com/ejuanoli/UNIP_ADS.git

筑業中華懷子學

E-mail de contato:

danilo.azevedo2002@gmail.com