PROJETO INTEGRADO II

Nome: André Toshiro Sudo

RA: 2025072740

Curso: Análise e Desenvolvimento de Sistemas

Polo: Anhanguera EAD Osasco

Cotia – São Paulo

2025

Nome: André Toshiro Sudo

RA: 2025072740

PROJETO INTEGRADO II

Trabalho apresentado ao curso de Análise e Desenvolvimento de Sistemas da Anhanguera Educacional, como parte das exigências para a obtenção de nota da disciplina Projeto Integrado II.

Cotia – São Paulo

2025

1 Introdução

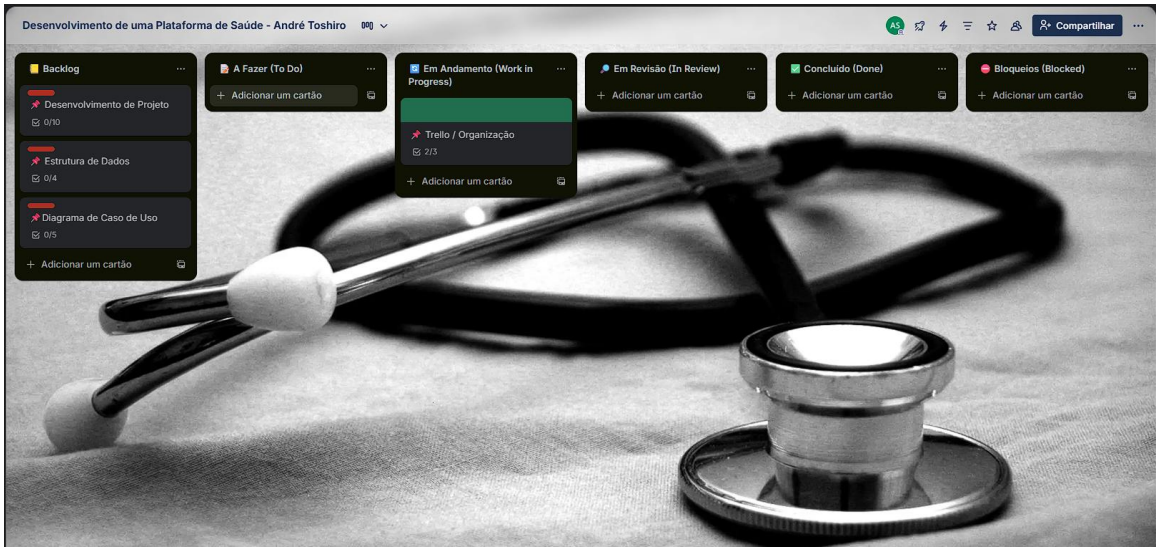
Objetivo deste trabalho foi o desenvolvimento de atividades envolvendo a organização de projetos com Scrum, criação de um sistema simples em Python, aplicação de lógica booleana, desenvolvimento de algoritmos em pseudocódigo e modelagem por meio de casos de uso.

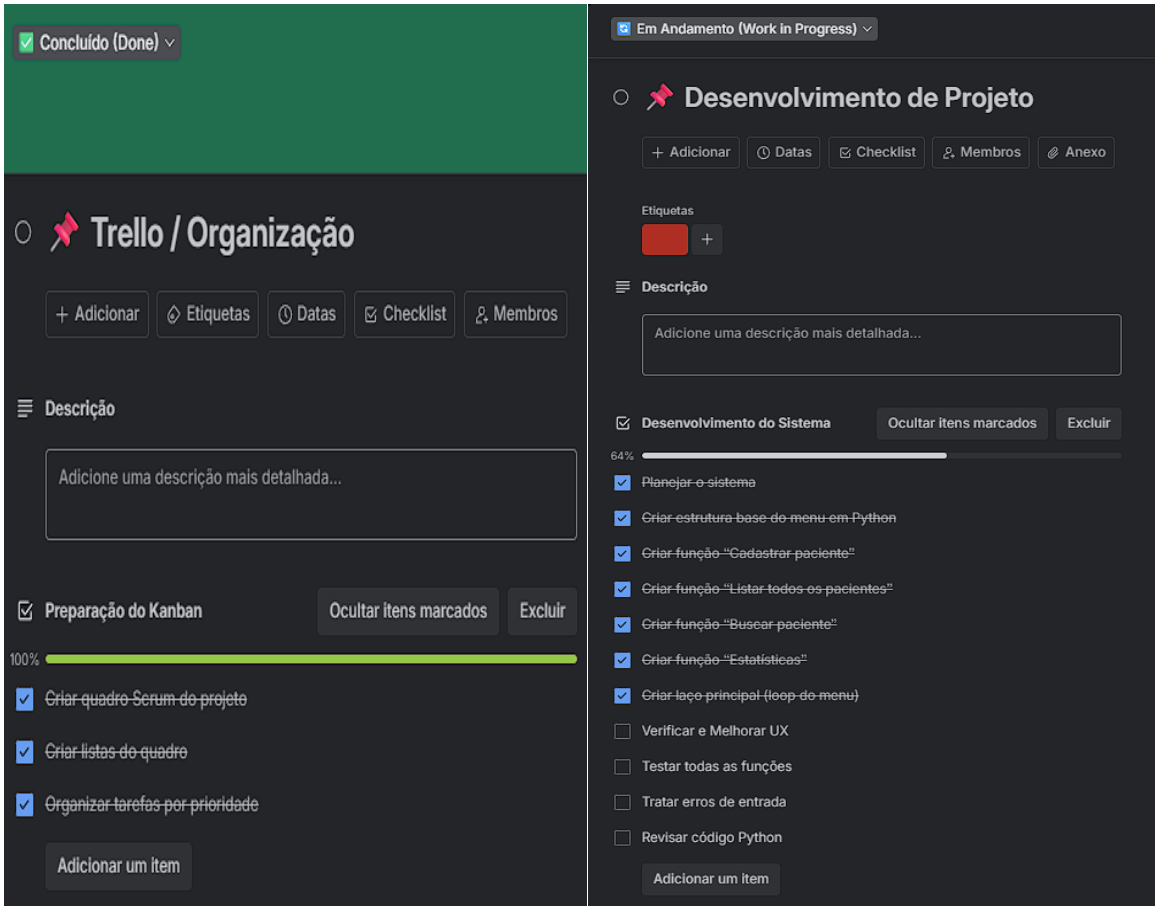
2 Quadro Scrum

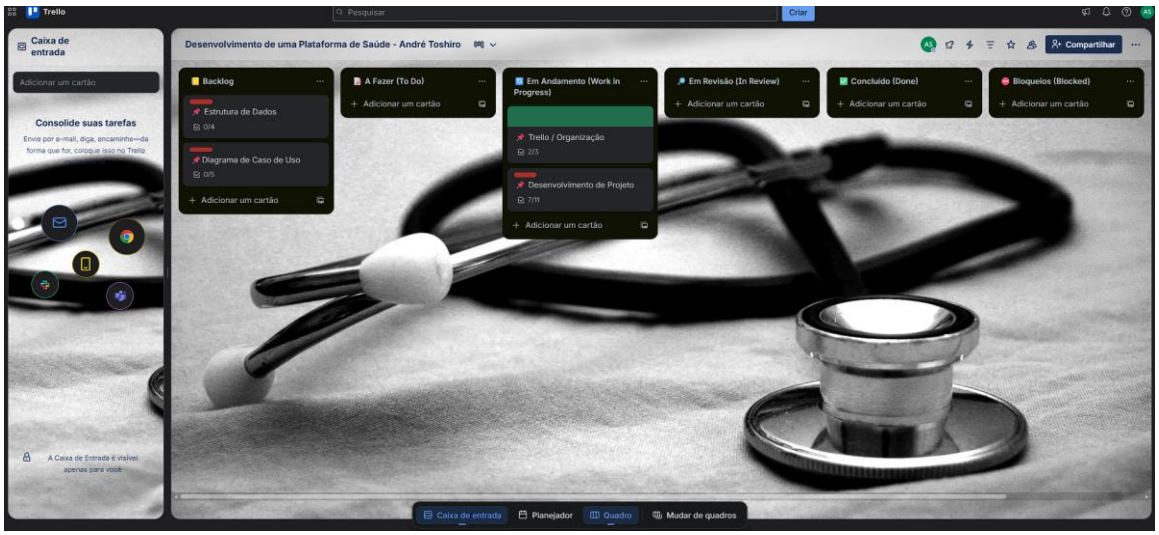
Para a organização do projeto, foi utilizado um quadro Scrum no Trello com as colunas: Backlog, A Fazer, Em Andamento e Concluído. Essa metodologia permitiu visualizar o andamento das tarefas e facilitar o gerenciamento das etapas de desenvolvimento.

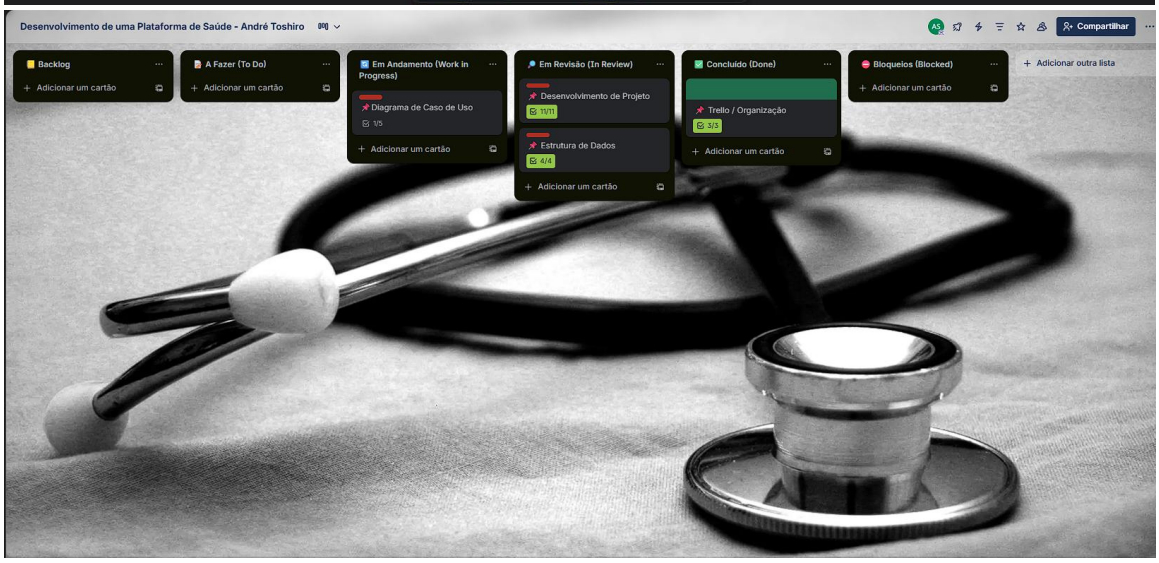
Trello utiliza o paradigma Kanban para gerenciamento de projetos. Os projetos são representados por quadros (boards), que contêm listas com várias tarefas. Cada tarefa é representada por meio de cartões criados dentro das listas. Cartões podem ser movidos, copiados ou compartilhados entre as listas, de modo a alterar seu progresso. Usuários podem ser adicionados nos cartões.

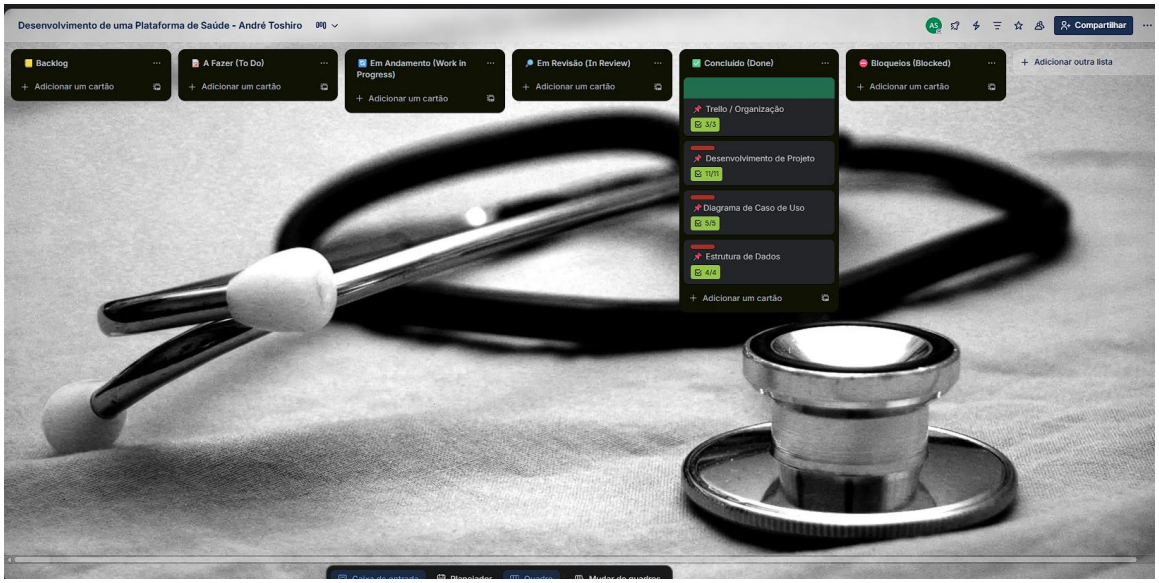
As tarefas foram organizadas da seguinte forma:




Concluído (Done)
Trello / Organização
+ Adicionar
Etiquetas
Datas
Checklist
Membros
Descrição
Adicione uma descrição mais detalhada...
Preparação do Kanban
Ocultar itens marcados
Excluir
100%
Criar quadro Scrum do projeto
Criar listas do quadro
Organizar tarefas por prioridade
Adicionar um item
Em Andamento (Work in Progress)
Desenvolvimento de Projeto
+ Adicionar
Datas
Checklist
Membros
Anexo
Etiquetas
Descrição
Adicione uma descrição mais detalhada...
Desenvolvimento do Sistema
Ocultar itens marcados
Excluir
64%
Planejar o sistema
Criar estrutura base do menu em Python
Criar função "Cadastrar paciente"
Criar função "Listar todos os pacientes"
Criar função "Buscar paciente"
Criar função "Estatísticas"
Criar laço principal (loop do menu)
Verificar e Melhorar UX
Testar todas as funções
Tratar erros de entrada
Revisar código Python
Adicionar um item


Trello
Pesquisar
Criar
Caixa de entrada
Desenvolvimento de uma Plataforma de Saúde - André Toshiro
Compartilhar
Adicionar um cartão
Consolide suas tarefas
Envie por e-mail, diga, encaminhe—da forma que for, coloque isso no Trello
A Caixa de Entrada é visível apenas para você.
Backlog
Estrutura de Dados
0/4
Diagrama de Caso de Uso
0/5
A Fazer (To Do)
Em Andamento (Work in Progress)
Trello / Organização
2/3
Desenvolvimento de Projeto
7/11
Em Revisão (In Review)
Concluído (Done)
Bloqueios (Blocked)
Caixa de entrada
Planejador
Quadro
Mudar de quadros


Desenvolvimento de uma Plataforma de Saúde - André Toshiro
Compartilhar
Backlog
A Fazer (To Do)
Em Andamento (Work in Progress)
Diagrama de Caso de Uso
1/5
Em Revisão (In Review)
Desenvolvimento de Projeto
11/11
Estrutura de Dados
4/4
Concluído (Done)
Trello / Organização
3/3
Bloqueios (Blocked)
Adicionar um cartão
Adicionar outra lista


Desenvolvimento de uma Plataforma de Saúde - André Toshiro
Compartilhar
Backlog
A Fazer (To Do)
Em Andamento (Work in Progress)
Em Revisão (In Review)
Concluído (Done)
Trello / Organização
3/3
Desenvolvimento de Projeto
11/11
Diagrama de Caso de Uso
5/5
Estrutura de Dados
4/4
Bloqueios (Blocked)
Adicionar um cartão
Adicionar outra lista

3 Desenvolvimento do programa em Python

Objetivo

O objetivo deste sistema foi a criação de uma aplicação interativa, desenvolvida em linguagem Python, capaz de auxiliar a Clínica Vida+ na organização de seus processos internos. A partir da situação-problema apresentada no projeto integrado, buscou-se implementar funcionalidades que permitissem realizar o cadastro de pacientes, controle de agendamentos, registro de informações, cálculos estatísticos, geração de arquivos e apoio às rotinas administrativas da clínica.

Base de desenvolvimento

O programa foi estruturado de forma modular, utilizando funções específicas para cada operação, permitindo clareza na manutenção do código e facilitando futuras expansões. Além disso, o sistema foi construído com foco na experiência do usuário, empregando menus intuitivos, validações de entrada e mensagens informativas, garantindo uma navegação simples mesmo para usuários com pouca experiência em tecnologia.

Arquitetura Geral do Sistema

O desenvolvimento foi baseado no uso de **listas, dicionários e arquivos JSON**, permitindo armazenar informações de forma organizada e persistente. O sistema também utiliza diretórios e arquivos auxiliares para gerar relatórios, registrar logs e armazenar receitas médicas.
Optei por organizar a aplicação em três grupos principais de usuários: **Pacientes**, **Médicos** e **Gestão**. Essa divisão foi escolhida para refletir o funcionamento real da clínica e garantir que cada perfil tivesse acesso apenas às funcionalidades necessárias para seu papel.

Além de melhorar a segurança, essa separação torna o sistema mais simples de usar, evita confusão entre funções e facilita futuras manutenções. O código implementa isso por meio de hubs específicos hub_paciente(), hub_medico() e hub_gestao() que direcionam o usuário ao conjunto correto de operações.

Pacientes
Os pacientes acessam apenas recursos relacionados a seus próprios dados. Podem visualizar e editar seu cadastro, criar e gerenciar agendamentos e consultar suas faturas. Esse conjunto de permissões garante que o paciente tenha autonomia, mas sem interferir em informações administrativas ou médicas.

Médicos
Os médicos têm acesso ao gerenciamento dos atendimentos. Eles conseguem visualizar seus agendamentos, alterar o status de consultas e gerar receitas médicas, que são automaticamente salvas no sistema. Dessa forma, o fluxo do atendimento é organizado e separado das funções administrativas e financeiras.

Gestão (Administrador)
O grupo da gestão concentra as funcionalidades mais amplas do sistema: criação e remoção de usuários, cadastro completo de pacientes, emissão e controle de faturas, exportação de relatórios e acesso a notificações e logs. Esse perfil possui mais permissões porque representa a equipe administrativa que coordena e supervisiona toda a operação da clínica.

Estrutura de Armazenamento de Dados

Para manter as informações organizadas e permitir que o sistema funcione mesmo após ser fechado, optei por utilizar **arquivos JSON** como forma de armazenamento. Esse formato é simples, leve e completamente integrado ao Python, facilitando tanto a leitura quanto a escrita dos dados sem a necessidade de um banco de dados complexo.

O sistema possui um diretório chamado **dados**, onde estão localizados arquivos como:

users.json – armazena os usuários cadastrados, incluindo nome, senha e função (paciente, médico ou gestão).

pacientes.json – registra as informações dos pacientes, como nome, idade, telefone e vínculo com o usuário.

appointments.json – guarda todos os agendamentos criados no sistema.

invoices.json – armazena as faturas geradas pela gestão, com parcelas e valores.

notifications.json – recebe mensagens enviadas pelos usuários para a gestão.

actions.log – registra ações importantes realizadas no sistema.

Cada um desses arquivos é carregado no início da execução por meio da função reload_all(), que utiliza rotinas como load_json() e save_json().

Esse processo garante que quaisquer alterações feitas durante o uso como um novo agendamento, um paciente editado ou uma fatura atualizada sejam imediatamente salvas e recuperadas na abertura seguinte.

Cadastro de Pacientes

O sistema permite inserir nome, idade e telefone de cada paciente, validando as informações fornecidas para evitar erros comuns, como idade inválida ou telefone em formatos incorretos.
Esses cadastros são armazenados no arquivo pacientes.json para garantir persistência dos dados entre execuções. O processo é acessado no menu através de funções como criar_usuario() e das rotinas específicas para pacientes.

Geração de Estatísticas Básicas

Atendendo ao requisito do projeto, o sistema calcula e apresenta estatísticas fundamentais sobre os pacientes cadastrados. No hub de gestor temos a opção de criar arquivo de texto chamado relatorio_estatisticas que permite percorre todos os registros e exibe os dados:

- número total de pacientes;
- idade média;
- identificação do paciente mais novo;
- identificação do paciente mais velho.

Listagem Completa de Pacientes

Adicionei a função em que exibição de todos os pacientes cadastrados. O sistema apresenta essa listagem por meio da função lists_tds(), que organiza os dados e permite que o usuário visualize de forma clara o nome, idade, telefone e o username vinculado de cada paciente.

Menu Interativo e Tratamento de Erros

Todas as funcionalidades são disponibilizadas em um menu simples e intuitivo, em conformidade com o modelo solicitado no PDF. O menu funciona em loop contínuo até que o usuário escolha sair, garantindo facilidade de navegação. O sistema também inclui tratamento de erros, como entradas inválidas, confirmações de operações e validações de dados, o que torna a utilização mais segura e evita falhas durante o uso.

4 Implementação da Lógica Booleana do Sistema

Nesta etapa do projeto, desenvolvi as regras lógicas responsáveis por determinar se um paciente pode ou não ser atendido em duas situações diferentes: **Consulta Normal** e **Emergência**. Todas as condições utilizadas foram definidas previamente no enunciado do Projeto Integrado e implementadas diretamente no sistema, garantindo fidelidade às especificações.

Para isso, trabalhei com quatro variáveis booleanas que representam o estado do paciente e da clínica:

- **A:** o paciente tem agendamento marcado;
- **B:** os documentos estão em dia;
- **C:** há médico disponível;
- **D:** o paciente está com os pagamentos em dia.

Com base nessas condições, desenvolvi duas expressões lógicas. A primeira corresponde à **Consulta Normal**, onde o atendimento é liberado quando o paciente cumpre todas as exigências clássicas de agendamento, documentação e disponibilidade do médico, ou quando, mesmo sem agendamento, possui documentos válidos, há médico disponível e não possui pendências financeiras. Já a segunda expressão representa a **Emergência**, onde o atendimento exige apenas a presença de um médico disponível e a regularidade dos documentos ou pagamentos.

Após definir as regras, implementei a geração completa das tabelas verdade, totalizando 16 combinações possíveis entre as variáveis A, B, C e D. Para cada combinação, o sistema avalia automaticamente se o paciente seria atendido ou não em cada modalidade. Também desenvolvi uma função que conta quantas situações resultam em atendimento, permitindo analisar a flexibilidade de cada tipo de consulta.

Por fim, incluí no sistema o caso prático solicitado no PDF. Nele, o paciente chega sem agendamento, com documentos válidos, com médico disponível, mas com pagamentos pendentes. O sistema avalia essa combinação e informa claramente se o paciente seria atendido em uma consulta normal ou apenas em emergência. Essa parte foi essencial para mostrar o funcionamento da lógica aplicada a uma situação real.

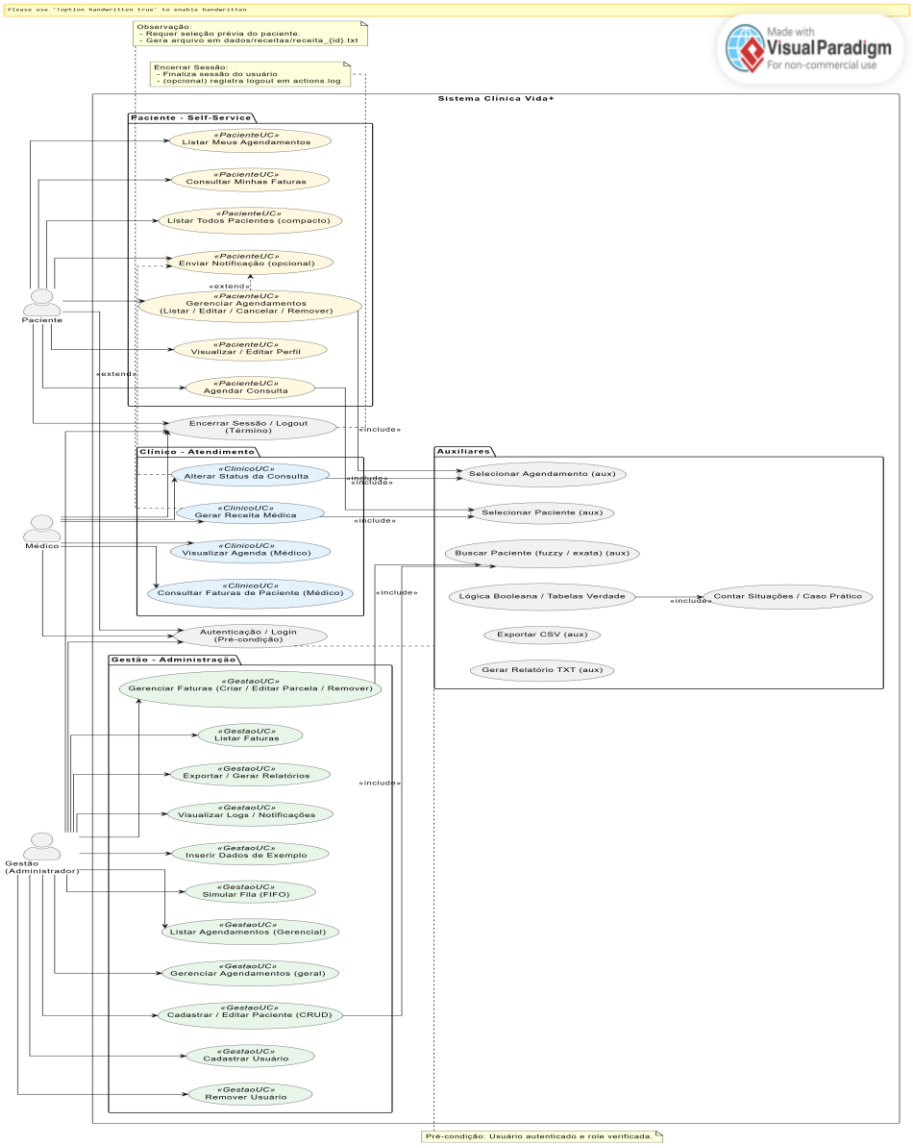
5 Implementação do sistema de secretariado

No Passo 4 do projeto, foi solicitado que eu construísse um algoritmo capaz de simular uma fila de atendimento, seguindo o princípio FIFO (*First In, First Out*), onde o primeiro paciente a chegar é sempre o primeiro a ser atendido. Para atender a essa exigência, implementei uma estrutura de fila utilizando o módulo deque, da biblioteca collections, que oferece operações rápidas e adequadas para esse tipo de funcionalidade.

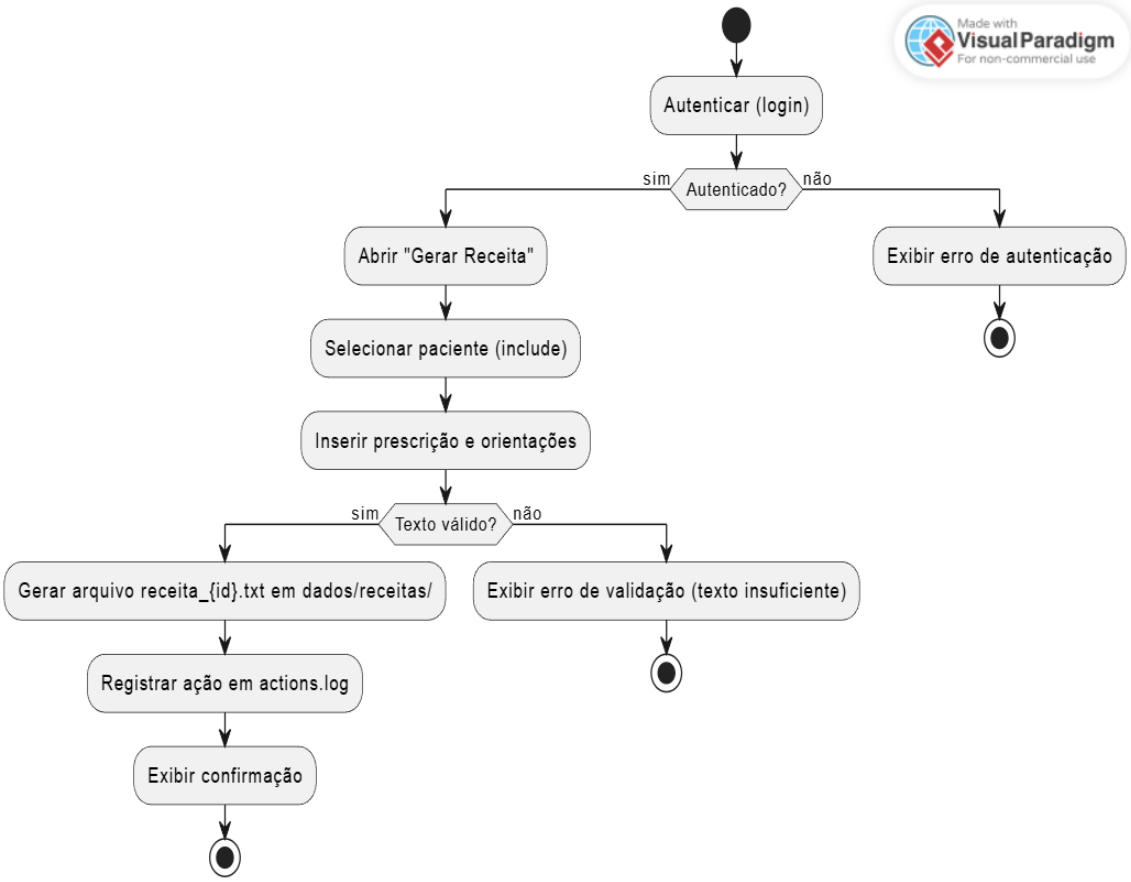
A simulação funciona da seguinte maneira: o sistema solicita ao usuário que informe o nome e o CPF de três pacientes, inserindo cada um deles na fila conforme são digitados. Após o preenchimento, o programa remove automaticamente o primeiro paciente da fila, representando o atendimento realizado. Em seguida, o sistema exibe os pacientes restantes, mantendo a ordem original de chegada.

6 Diagrama do sistema

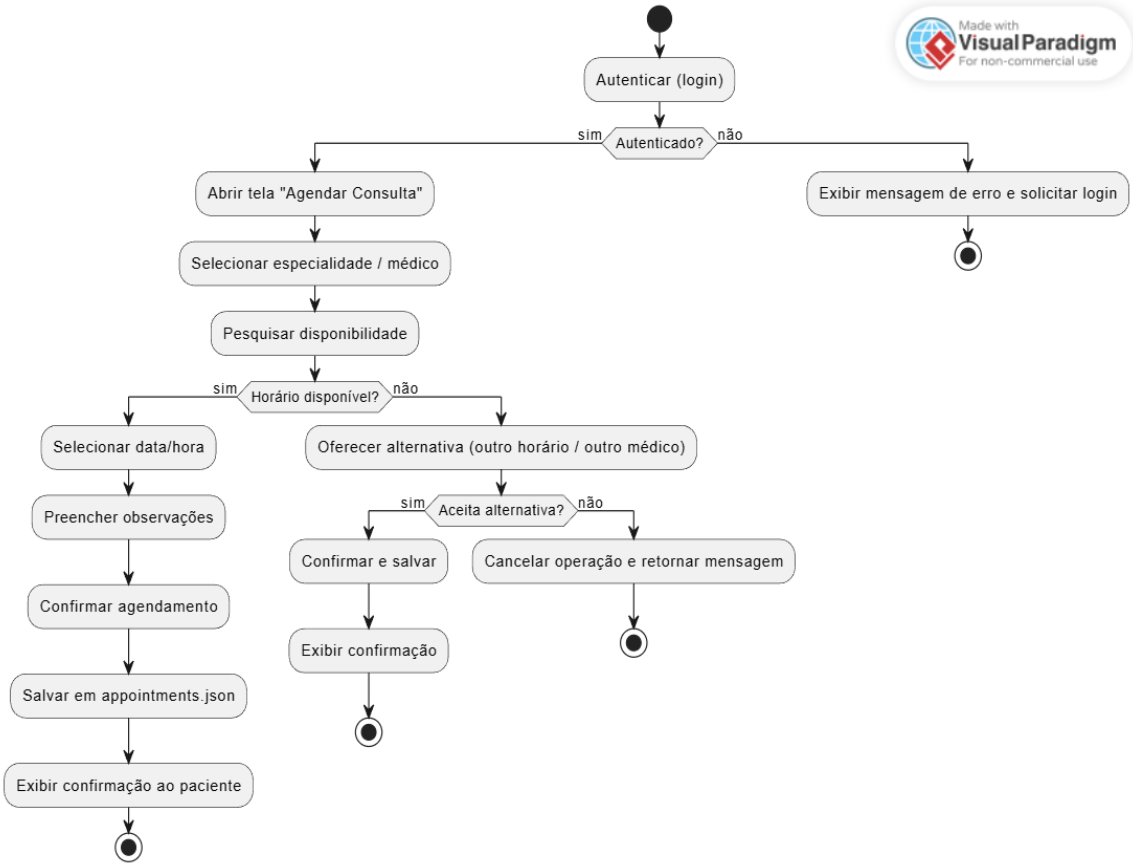
Diagrama é uma representação gráfica usada para demonstrar um **esquema simplificado** ou um resumo sobre um assunto. É utilizado setas e símbolos para facilitar a leitura.



O primeiro diagrama construído foi o **Diagrama de Caso de Uso**, que apresenta as funcionalidades disponíveis para os três atores principais do sistema: **Paciente**, **Médico** e **Gestão (Administrador)**. Cada ator possui permissões distintas e acessa diferentes partes do sistema, respeitando exatamente os limites e responsabilidades estabelecidos no escopo do projeto. Nesse diagrama, foi possível representar todas as ações suportadas pelo sistema, como agendamentos, gerenciamento de pacientes, emissão de receitas médicas, controle financeiro, geração de relatórios e tratamento da fila de atendimento.



Além das associações diretas entre atores e funcionalidades, o diagrama também apresenta os relacionamentos «include» e «extend», utilizados para representar dependências e fluxos opcionais dentro do sistema. O relacionamento «include» foi aplicado quando uma operação depende obrigatoriamente de outra para ser executada, como no caso da ação de **Gerar Receita**, que sempre exige a **Seleção de um Paciente** previamente. Já o relacionamento «extend» foi utilizado nos casos em que uma operação pode acontecer de forma opcional, como o envio de notificações após alterar um agendamento ou cancelar um atendimento. Essa modelagem torna o diagrama mais fiel ao funcionamento real da clínica e às validações existentes no código.



Como complemento ao diagrama de caso de uso, também desenvolvi **dois diagramas de atividade**, que foram fundamentais para detalhar visualmente o fluxo interno das operações mais importantes: **Agendar Consulta** e **Gerar Receita Médica**.

7 Conclusão

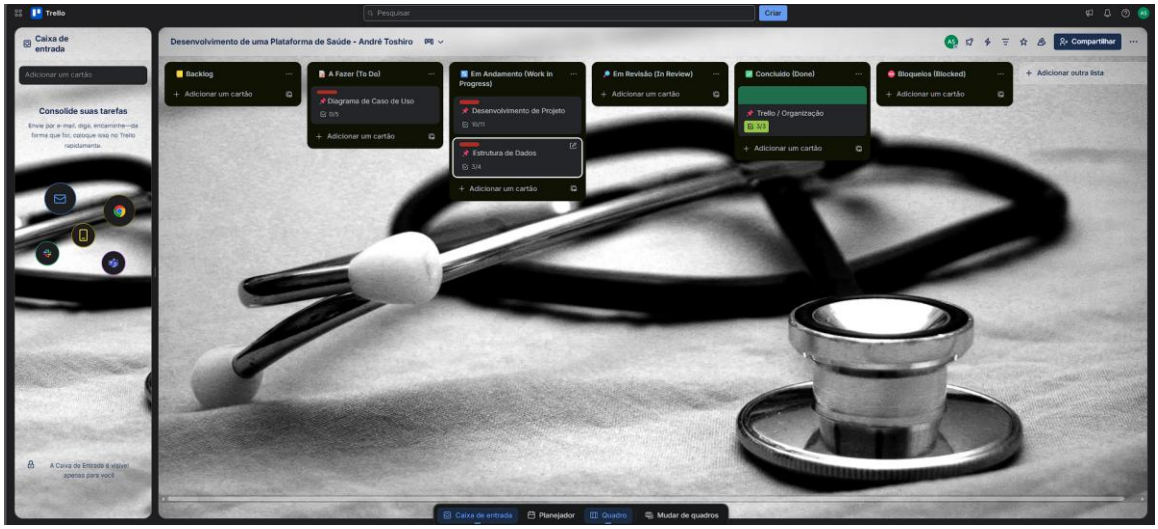
O desenvolvimento deste sistema para a Clínica Vida+ representou uma experiência significativa tanto no aprimoramento das minhas habilidades técnicas quanto na compreensão prática da modelagem e implementação de soluções reais dentro da área da saúde. Ao longo do projeto, pude aplicar conceitos fundamentais de programação, estruturação de dados, boas práticas de organização de código e modelagem UML, traduzindo requisitos reais em funcionalidades concretas.

A utilização dos diagramas de caso de uso e atividade foi essencial para estruturar o raciocínio e garantir que todas as etapas do fluxo fossem planejadas antes da implementação. Esses diagramas também permitiram validar o comportamento esperado do sistema com clareza, destacando as regras obrigatórias, fluxos alternativos e responsabilidades de cada perfil de usuário. Com isso, consegui desenvolver um sistema coeso, organizado e funcional, capaz de lidar com cadastros, agendamentos, receitas médicas, relatórios administrativos, gerenciamento de faturas e simulação da fila de atendimento.

8 Referencias

Ferramentas:
Visual Paradigm. Disponível em: https://online.visual . Acesso em: 03 Dezembro 2025.

TRELLO. Organize tudo com quadros. Disponível em: https://trello.com/ . Acesso em: 03 Dezembro 2025.



TRELLO. Projeto de organização. Disponível em: https://trello.com/invite/b/691cf9d9f98b94bd63e7d07e/ATTIdd030fe2388a7209cbfe1479e302dc532DA85BB4/desenvolvimento-de-uma-plataforma-de-saude-andre-toshiro
GitHub. Projeto do sistema em Python disponível em: https://github.com/KuromoriOficial/TCC_project.git

Fontes de pesquisa:

Significados, Diagrama. Disponível em: https://www.significados.com.br/diagrama https://www.significados.com.br/diagrama/ . Acesso em: 03 Dezembro 2025.

Aprender Estatistica Fácil. Disponível em: https://estatisticafacil.org/glossario/o-que-e-booleano-definicao-e-aplicacoes/ . Acesso em: 03 Dezembro 2025.