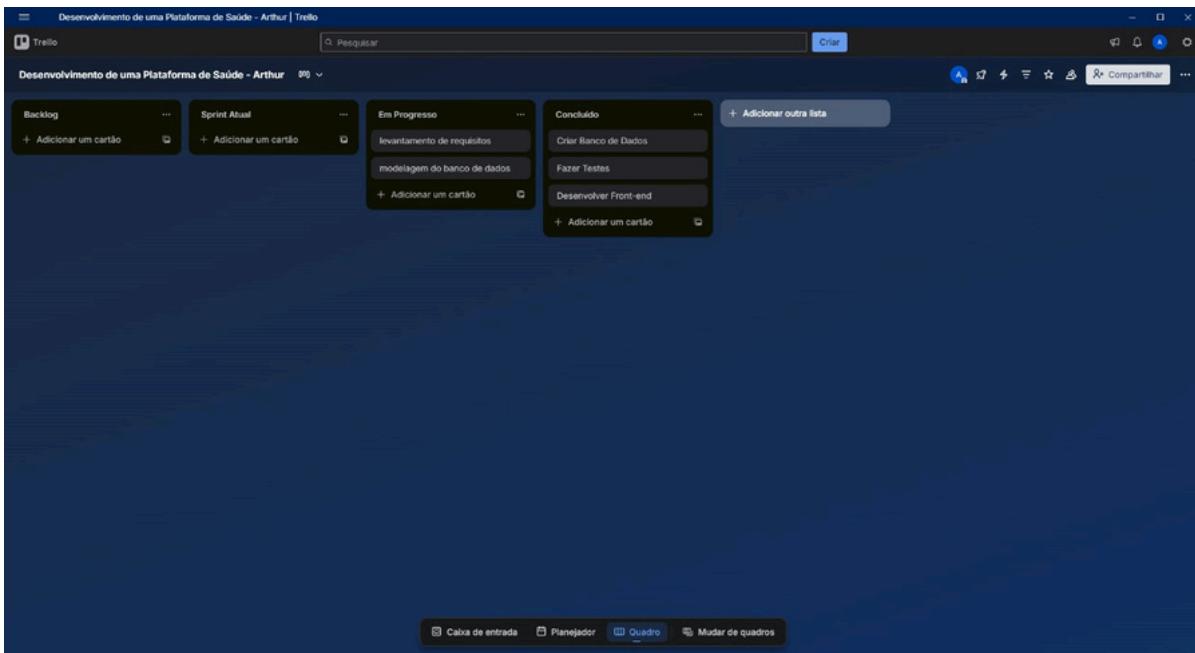


## **INTRODUÇÃO**

- O presente trabalho tem como objetivo desenvolver soluções tecnológicas para a "Clínica Vida+", um estabelecimento de saúde que enfrenta desafios relacionados ao crescimento da demanda e à falta de informatização de seus processos. A proposta consiste em integrar conhecimentos de gestão de projetos, desenvolvimento de software, lógica matemática, algoritmos e modelagem de sistemas para modernizar a clínica. Serão apresentadas soluções práticas, incluindo a organização do projeto via metodologia ágil (Scrum), o desenvolvimento de um script em Python para cadastro e estatísticas, a aplicação de lógica proposicional para regras de negócio, a estruturação de algoritmos de fila e a modelagem UML para o sistema de gestão.

## **1: Desenvolvimento** Gestão Ágil com Scrum

- O objetivo desta primeira Sprint foi iniciar a estruturação do projeto da Clínica Vida+. Foram definidas as tarefas prioritárias no Backlog, focando nos requisitos funcionais principais, como o cadastro de pacientes e agendamento. Durante a execução, movemos para a lista "Em Progresso" as tarefas de levantamento de requisitos e modelagem do banco de dados. Os desafios encontrados envolveram a definição das regras de negócio para atendimentos de emergência, que foram sanadas após revisão com a diretoria (simulação). Ao final, as tarefas de planejamento foram movidas para "Concluído", permitindo o início do desenvolvimento do código na próxima etapa.



## 2: Desenvolvimento do Programa em Python

Abaixo segue o código fonte desenvolvido para o sistema de cadastro e estatísticas

```
import os

# Lista para armazenar os dicionários de pacientes
pacientes = []

def cadastrar_paciente():
    print("\n--- CADASTRO DE PACIENTE ---")
    nome = input("Nome do paciente: ")
    try:
        idade = int(input("Idade: "))
        telefone = input("Telefone: ")

        # Dicionário do paciente
        paciente = {
            "nome": nome,
            "idade": idade,
            "telefone": telefone
        }
        pacientes.append(paciente)
    except ValueError:
        print("Por favor, entre com um número válido para a idade.")
```

```
    print("Paciente cadastrado com sucesso!")

except ValueError:

    print("Erro: A idade deve ser um número inteiro.")


def ver_estatisticas():

    print("\n--- ESTATÍSTICAS ---")

    total = len(pacientes)

    if total == 0:

        print("Nenhum paciente cadastrado.")

        return

    soma_idades = sum(p["idade"] for p in pacientes)
    media_idade = soma_idades / total

    mais_novo = min(pacientes, key=lambda p: p["idade"])
    mais_velho = max(pacientes, key=lambda p: p["idade"])

    print(f"Total de pacientes: {total}")
    print(f"Média de idade: {media_idade:.1f} anos")
    print(f"Paciente mais novo: {mais_novo['nome']} ({mais_novo['idade']} anos)")
    print(f"Paciente mais velho: {mais_velho['nome']} ({mais_velho['idade']} anos)")


def buscar_paciente():

    print("\n--- BUSCAR PACIENTE ---")

    nome_busca = input("Digite o nome para buscar: ").lower()
    encontrado = False

    for p in pacientes:
        if nome_busca in p["nome"].lower():
            print(f"Nome: {p['nome']} | Idade: {p['idade']} | Tel: {p['telefone']}")
            encontrado = True

    if not encontrado:
        print("Paciente não encontrado.")


def listar.todos():

    print("\n--- LISTA GERAL ---")
```

```
if not pacientes:
    print("Lista vazia.")

else:
    for p in pacientes:
        print(f"Nome: {p['nome']} | Idade: {p['idade']} | Tel: {p['telefone']}")

# Loop Principal
while True:
    print("\n==== SISTEMA CLÍNICA VIDA+ ====")
    print("1. Cadastrar paciente")
    print("2. Ver estatísticas")
    print("3. Buscar paciente")
    print("4. Listar todos os pacientes")
    print("5. Sair")

    opcao = input("Escolha uma opção: ")

    if opcao == '1':
        cadastrar_paciente()
    elif opcao == '2':
        ver_estatisticas()
    elif opcao == '3':
        buscar_paciente()
    elif opcao == '4':
        listar.todos()
    elif opcao == '5':
        print("Saindo do sistema...")
        break
    else:
        print("Opção inválida!")
```

```
C:\WINDOWS\py.exe
5. Sair
Escolha uma opção: 2

--- ESTATÍSTICAS ---
Total de pacientes: 1
Média de idade: 22.0 anos
Paciente mais novo: Heloisa Saraiva (22 anos)
Paciente mais velho: Heloisa Saraiva (22 anos)

==== SISTEMA CLÍNICA VIDA+ ====
1. Cadastrar paciente
2. Ver estatísticas
3. Buscar paciente
4. Listar todos os pacientes
5. Sair
Escolha uma opção:
```

### 3: Passo 3: Lógica Computacional e Tabelas Verdade

#### 1. Expressões Lógicas:

- **Consulta Normal:**  $(A \wedge B \wedge C) \vee (B \wedge C \wedge D)$
- **Emergência:**  $C \wedge (B \vee D)$

#### 2. Tabela Verdade (Consulta Normal):

Linhas	A (Agend.)	B (DocOK)	C (Médico)	D (PgtoOK)	(A $\wedge$ B $\wedge$ C)	(B $\wedge$ C $\wedge$ D)	Resultado Final (OU)	
1	V	V	V	V	V	V	V	
2	V	V	V	F	V	F	V	
3	V	V	F	V	F	F	F	
4	V	V	F	F	F	F	F	
5	V	F	V	V	F	F	F	
6	V	F	V	F	F	F	F	
7	V	F	F	V	F	F	F	
8	V	F	F	F	F	F	F	
9	F	V	V	V	F	V	V	
10	F	V	V	F	F	F	F	
11	F	V	F	V	F	F	F	
12	F	V	F	F	F	F	F	
13	F	F	V	V	F	F	F	
14	F	F	V	F	F	F	F	
15	F	F	F	V	F	F	F	
16	F	F	F	F	F	F	F	

#### 4. Análise:

- **Consulta Normal:** O paciente é atendido em 3 situações diferentes (Linhas 1, 2 e 9).
- **Emergência:** Considerando as combinações de B e D quando C é verdadeiro, existem 3 situações favoráveis (Documento OK, ou Pagamento OK, ou ambos, desde que tenha médico).

5. Situação Prática:Dados: A=F, B=V, C=V, D=F.

- **Consulta Normal:** (F e V e V) OU (V e V e F) -> F OU F = **NÃO ATENDIDO**.
- **Emergência:** V E (V ou F) -> V E V = **SIM, ATENDIDO**.

# Algoritmos e Estrutura de Dados

## VARIAVEIS

Fila: Vetor[1..3] de Registro (Nome, CPF)

Inicio, Fim, i: Inteiro

## INICIO

// 1. Inserir 3 pacientes (Simulação de chegada)

ESCREVAL("--- Chegada de Pacientes ---")

PARA i DE 1 A 3 FACA

ESCREVA("Digite Nome do paciente ", i, ": ")

LEIA(Fila[i].Nome)

ESCREVA("Digite CPF do paciente ", i, ": ")

LEIA(Fila[i].CPF)

FIM\_PARA

// 2. Remover o primeiro (Atendimento)

ESCREVAL("--- Chamando Próximo Paciente ---")

ESCREVAL("Atendendo: ", Fila[1].Nome)

// Deslocar a fila (o 2 vira 1, o 3 vira 2)

Fila[1] <- Fila[2]

Fila[2] <- Fila[3]

Fila[3] <- VAZIO // Última posição fica livre

// 3. Mostrar quem ainda está na fila

ESCREVAL("--- Pacientes Restantes na Fila ---")

ESCREVAL("1º da fila: ", Fila[1].Nome)

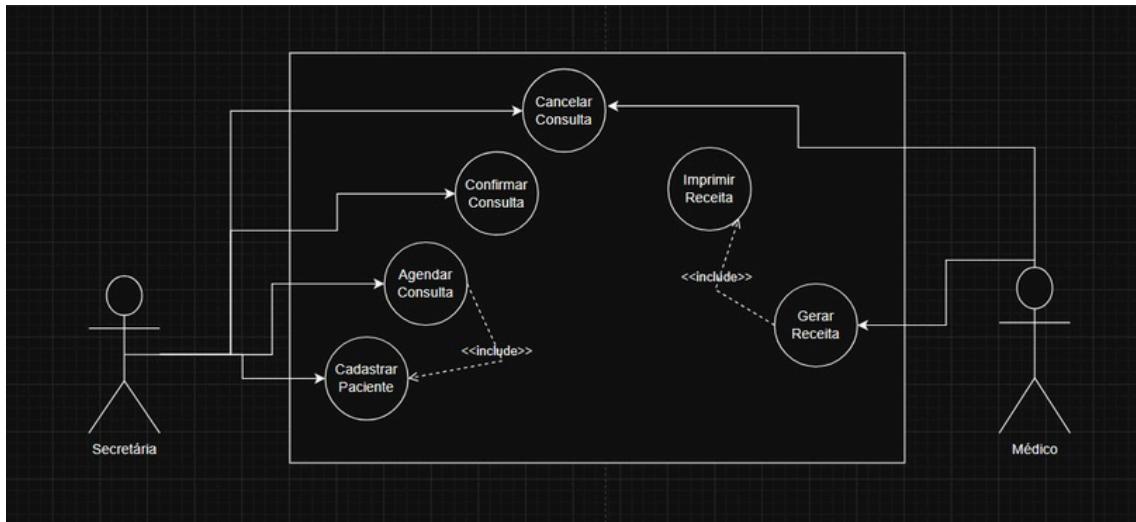
ESCREVAL("2º da fila: ", Fila[2].Nome)

FIM



## 5: Modelagem de Sistemas (UML)

- A seguir, o Diagrama de Casos de Uso representando as interações entre Secretária, Médico e o Sistema



## • 6 CONCLUSÃO

- A realização deste Projeto Integrado permitiu simular um cenário real de desenvolvimento de software para a área de saúde. Através da metodologia Scrum, foi possível compreender a importância da organização e divisão de tarefas. A implementação em Python demonstrou como estruturas de dados (listas e dicionários) são fundamentais para o gerenciamento de informações. Além disso, a aplicação da lógica matemática e a construção de algoritmos de fila garantiram que as regras de negócio fossem sólidas e eficientes. Por fim, a modelagem UML serviu como um mapa visual essencial para o entendimento das funcionalidades do sistema. Conclui-se que a integração dessas disciplinas é vital para formar um profissional capaz de resolver problemas complexos no mercado de TI.

## 7. REFERÊNCIAS

MANZANO, J. A. N. G. Algoritmos: Lógica para Desenvolvimento de Programação de Computadores. São Paulo: Érica, 2016.

PYTHON SOFTWARE FOUNDATION. Python Language Reference, version 3.x. Disponível em: <http://www.python.org>. Acesso em: 03 dez. 2025.

SOMMERVILLE, Ian. Engenharia de Software. 10. ed. São Paulo: Pearson, 2019.

TRELLO. Gerenciamento de Projetos. Disponível em: <https://trello.com>. Acesso em: 03 dez. 2025.