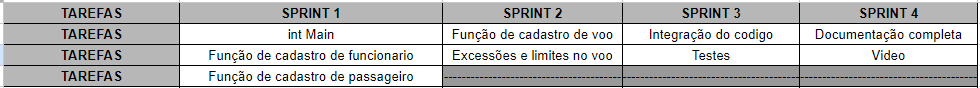
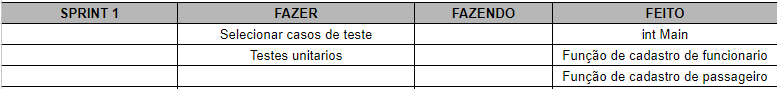
**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS INSTITUTO DE CIÊNCIAS EXATAS E INFORMÁTICA UNIDADE EDUCACIONAL PRAÇA DA LIBERDADE Bacharelado em Engenharia de Software**   
Caio Souza de Resende (não faço AEDS I, optei por fazer sozinho)  
Sistema Voo Seguro  
  
**Apresentação:** Este sistema foi criado para auxiliar empresas aéreas a organizarem seus voos e alocar seus passageiros para os destinos desejados. Com esse sistema as empresas e seus funcionários poderão otimizar seu tempo, ter uma organização melhor da escala e focar seu tempo em outros processos dentro das empresas.

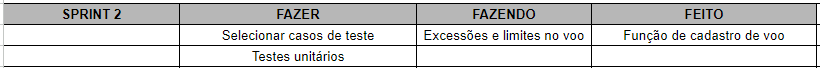
**Backlog do produto:**

**Backlog geral:**

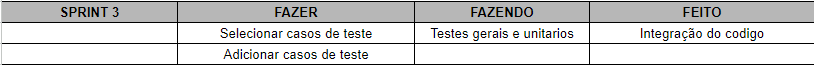
Como não faço Aeds I, optei por realizar o trabalho sozinho. Com isso uma simples planilha me ajudou a manter controle dos prazos das sprints.



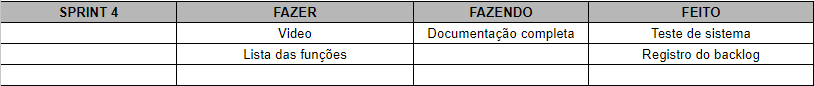
**Backlog Sprint 1:**

**Backlog Sprint 2:**

**Backlog Sprint 3:**



**Backlog Sprint 4:**



**Lista de funções e parâmetros:**

1. **void cadastrar\_passageiro()**

Essa função faz o cadastro do passageiro, coletando todas as informações necessárias para que ele seja cadastrado e identificado dentro do sistema da companhia aérea. Após isso ela armazena os dados em um arquivo .txt .

1. **void cadastrar\_tripulacao()**

Essa função pega as credenciais do tripulante (funcionário), deixando ele registrado

e possibilitando ele ser alocado para um voo. Após isso ela armazena os dados em um arquivo .txt .

1. **void cadastrar\_voo()**

Essa função usa as informações dos tripulantes para que possa criar um voo, ele só pode ser criado com pelo menos 1 piloto, 1 co-piloto e 1 comissário, além de destino, data e horário de saída. Após isso ela armazena os dados em um arquivo .txt .

**Caso de teste:**

**Pré requisitos:**

* O sistema deve estar devidamente configurado e em execução
* Os arquivos “Passageiro.txt” , “Tripulante.txt” e “Voo.txt” devem estar disponíveis para a leitura e escrita

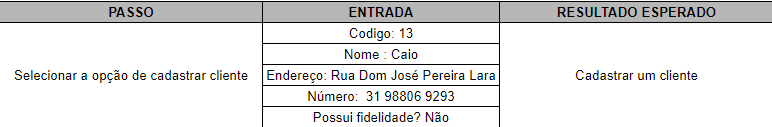
**Passos de teste e os resultados esperados**

**1 -** Inicializar o sistema

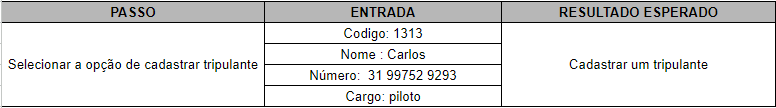
-Passo: Executar o programa

-Resultado Esperado: O sistema deve e ser funcional na máquina do funcionário

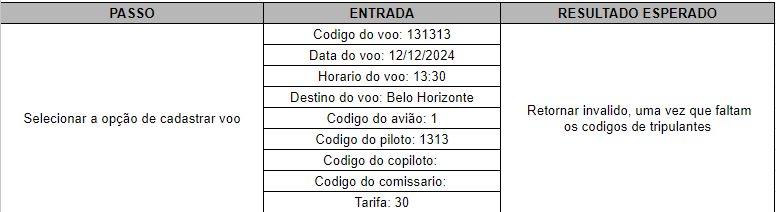
**2-** Cadastrar um cliente



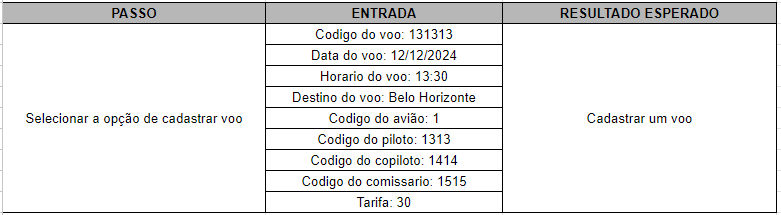
**3-** Cadastrar um funcionário



**4-** Cadastrar um voo sem todos os tripulantes



**5-** Cadastrar um voo com todos os tripulantes



**Código do sistema:**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <stdbool.h>

#define MAX\_PASSENGERS 100

#define MAX\_CREW 50

#define MAX\_FLIGHTS 50

// Estruturas

typedef struct {

int codigo;

char nome[100];

char endereco[200];

char telefone[15];

bool fidelidade;

int pontos\_fidelidade;

} Passageiro;

typedef struct {

int codigo;

char nome[100];

char telefone[15];

char cargo[20]; // piloto, copiloto, comissário

} Tripulacao;

typedef struct {

int codigo;

char data[11];

char hora[6];

char origem[50];

char destino[50];

char codigo\_aviao[10];

int codigo\_piloto;

int codigo\_copiloto;

int codigo\_comissario;

char status[10]; // ativo ou inativo

float tarifa;

} Voo;

// Bases de dados

Passageiro passageiros[MAX\_PASSENGERS];

int total\_passageiros = 0;

Tripulacao tripulantes[MAX\_CREW];

int total\_tripulantes = 0;

Voo voos[MAX\_FLIGHTS];

int total\_voos = 0;

// Funções auxiliares

bool codigo\_existe\_passageiro(int codigo) {

for (int i = 0; i < total\_passageiros; i++) {

if (passageiros[i].codigo == codigo) {

return true;

}

}

return false;

}

bool codigo\_existe\_tripulacao(int codigo) {

for (int i = 0; i < total\_tripulantes; i++) {

if (tripulantes[i].codigo == codigo) {

return true;

}

}

return false;

}

void limpar\_buffer() {

int c;

while ((c = getchar()) != '\n' && c != EOF);

}

void cadastrar\_passageiro() {

if (total\_passageiros >= MAX\_PASSENGERS) {

printf("Capacidade de passageiros atingida!\n");

return;

}

Passageiro p;

printf("Digite o código do passageiro: ");

scanf("%d", &p.codigo);

limpar\_buffer();

if (codigo\_existe\_passageiro(p.codigo)) {

printf("Código já utilizado!\n");

return;

}

printf("Digite o nome do passageiro: ");

fgets(p.nome, sizeof(p.nome), stdin);

p.nome[strcspn(p.nome, "\n")] = 0;

printf("Digite o endereço do passageiro: ");

fgets(p.endereco, sizeof(p.endereco), stdin);

p.endereco[strcspn(p.endereco, "\n")] = 0;

printf("Digite o telefone do passageiro: ");

fgets(p.telefone, sizeof(p.telefone), stdin);

p.telefone[strcspn(p.telefone, "\n")] = 0;

printf("O passageiro possui fidelidade? (1 para sim, 0 para não): ");

scanf("%d", (int\*)&p.fidelidade);

limpar\_buffer();

p.pontos\_fidelidade = 0;

passageiros[total\_passageiros++] = p;

printf("Passageiro cadastrado com sucesso!\n");

}

void cadastrar\_tripulacao() {

if (total\_tripulantes >= MAX\_CREW) {

printf("Capacidade de tripulação atingida!\n");

return;

}

Tripulacao t;

printf("Digite o código do tripulante: ");

scanf("%d", &t.codigo);

limpar\_buffer();

if (codigo\_existe\_tripulacao(t.codigo)) {

printf("Código já utilizado!\n");

return;

}

printf("Digite o nome do tripulante: ");

fgets(t.nome, sizeof(t.nome), stdin);

t.nome[strcspn(t.nome, "\n")] = 0;

printf("Digite o telefone do tripulante: ");

fgets(t.telefone, sizeof(t.telefone), stdin);

t.telefone[strcspn(t.telefone, "\n")] = 0;

printf("Digite o cargo do tripulante (piloto, copiloto, comissário): ");

fgets(t.cargo, sizeof(t.cargo), stdin);

t.cargo[strcspn(t.cargo, "\n")] = 0;

tripulantes[total\_tripulantes++] = t;

printf("Tripulante cadastrado com sucesso!\n");

}

void cadastrar\_voo() {

if (total\_voos >= MAX\_FLIGHTS) {

printf("Capacidade de voos atingida!\n");

return;

}

Voo v;

printf("Digite o código do voo: ");

scanf("%d", &v.codigo);

limpar\_buffer();

printf("Digite a data (DD/MM/AAAA): ");

fgets(v.data, sizeof(v.data), stdin);

v.data[strcspn(v.data, "\n")] = 0;

printf("Digite a hora (HH:MM): ");

fgets(v.hora, sizeof(v.hora), stdin);

v.hora[strcspn(v.hora, "\n")] = 0;

printf("Digite a origem: ");

fgets(v.origem, sizeof(v.origem), stdin);

v.origem[strcspn(v.origem, "\n")] = 0;

printf("Digite o destino: ");

fgets(v.destino, sizeof(v.destino), stdin);

v.destino[strcspn(v.destino, "\n")] = 0;

printf("Digite o código do avião: ");

fgets(v.codigo\_aviao, sizeof(v.codigo\_aviao), stdin);

v.codigo\_aviao[strcspn(v.codigo\_aviao, "\n")] = 0;

printf("Digite o código do piloto: ");

scanf("%d", &v.codigo\_piloto);

limpar\_buffer();

printf("Digite o código do copiloto: ");

scanf("%d", &v.codigo\_copiloto);

limpar\_buffer();

printf("Digite o código do comissário: ");

scanf("%d", &v.codigo\_comissario);

limpar\_buffer();

printf("Digite a tarifa: ");

scanf("%f", &v.tarifa);

limpar\_buffer();

// Verificações de tripulação

if (!codigo\_existe\_tripulacao(v.codigo\_piloto) || !codigo\_existe\_tripulacao(v.codigo\_copiloto)) {

strcpy(v.status, "inativo");

printf("Voo marcado como inativo. Piloto e copiloto obrigatórios.\n");

} else {

strcpy(v.status, "ativo");

}

voos[total\_voos++] = v;

printf("Voo cadastrado com sucesso!\n");

}

int main() {

int opcao;

do {

printf("\nSistema de Gerenciamento de Voos\n");

printf("1. Cadastrar Passageiro\n");

printf("2. Cadastrar Tripulação\n");

printf("3. Cadastrar Voo\n");

printf("4. Sair\n");

printf("Escolha uma opção: ");

scanf("%d", &opcao);

limpar\_buffer();

switch (opcao) {

case 1:

cadastrar\_passageiro();

break;

case 2:

cadastrar\_tripulacao();

break;

case 3:

cadastrar\_voo();

break;

case 4:

printf("Saindo do sistema.\n");

break;

default:

printf("Opção inválida!\n");

}

} while (opcao != 4);

return 0;

}