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.

TAREFAS	SPRINT 1	SPRINT 2	SPRINT 3	SPRINT 4
TAREFAS	int Main	Função de cadastro de voo	Integração do codigo	Documentação completa
TAREFAS	Função de cadastro de funcionario	Excessões e limites no voo	Testes	Video
TAREFAS	Função de cadastro de passageiro			

Backlog Sprint 1:

SPRINT 1	FAZER	FAZENDO	FEITO
	Selecionar casos de teste		int Main
	Testes unitarios		Função de cadastro de funcionario
			Função de cadastro de passageiro

Backlog Sprint 2:

SPRINT 2	FAZER	FAZENDO	FEITO
	Selecionar casos de teste	Excessões e limites no voo	Função de cadastro de voo
	Testes unitários		

Backlog Sprint 3:

SPRINT 3	FAZER	FAZENDO	FEITO
	Selecionar casos de teste	Testes gerais e unitarios	Integração do codigo
	Adicionar casos de teste		

Backlog Sprint 4:

SPRINT 4	FAZER	FAZENDO	FEITO
	Video	Documentação completa	Teste de sistema
	Lista das funções		Registro do backlog

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 .

2. **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 .

3. **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

PASSO	ENTRADA	RESULTADO ESPERADO
Selecionar a opção de cadastrar cliente	Codigo: 13	Cadastrar um cliente
	Nome : Caio	
	Endereço: Rua Dom José Pereira Lara	
	Número: 31 98806 9293	
	Possui fidelidade? Não	

3- Cadastrar um funcionário

PASSO	ENTRADA	RESULTADO ESPERADO
Selecionar a opção de cadastrar tripulante	Codigo: 1313	Cadastrar um tripulante
	Nome : Carlos	
	Número: 31 99752 9293	
	Cargo: piloto	

4- Cadastrar um voo sem todos os tripulantes

PASSO	ENTRADA	RESULTADO ESPERADO
Selecionar a opção de cadastrar voo	Codigo do voo: 131313	Retornar invalido, uma vez que faltam os codigos de tripulantes
	Data do voo: 12/12/2024	
	Horario do voo: 13:30	
	Destino do voo: Belo Horizonte	
	Codigo do avião: 1	
	Codigo do piloto: 1313	
	Codigo do copiloto:	
	Codigo do comissario:	
	Tarifa: 30	

5- Cadastrar um voo com todos os tripulantes

PASSO	ENTRADA	RESULTADO ESPERADO
Selecionar a opção de cadastrar voo	Codigo do voo: 131313	Cadastrar um voo
	Data do voo: 12/12/2024	
	Horario do voo: 13:30	
	Destino do voo: Belo Horizonte	
	Codigo do avião: 1	
	Codigo do piloto: 1313	
	Codigo do copiloto: 1414	
	Codigo do comissario: 1515	
	Tarifa: 30	

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;
}
```