

OFICINA JÁ PASSEI!

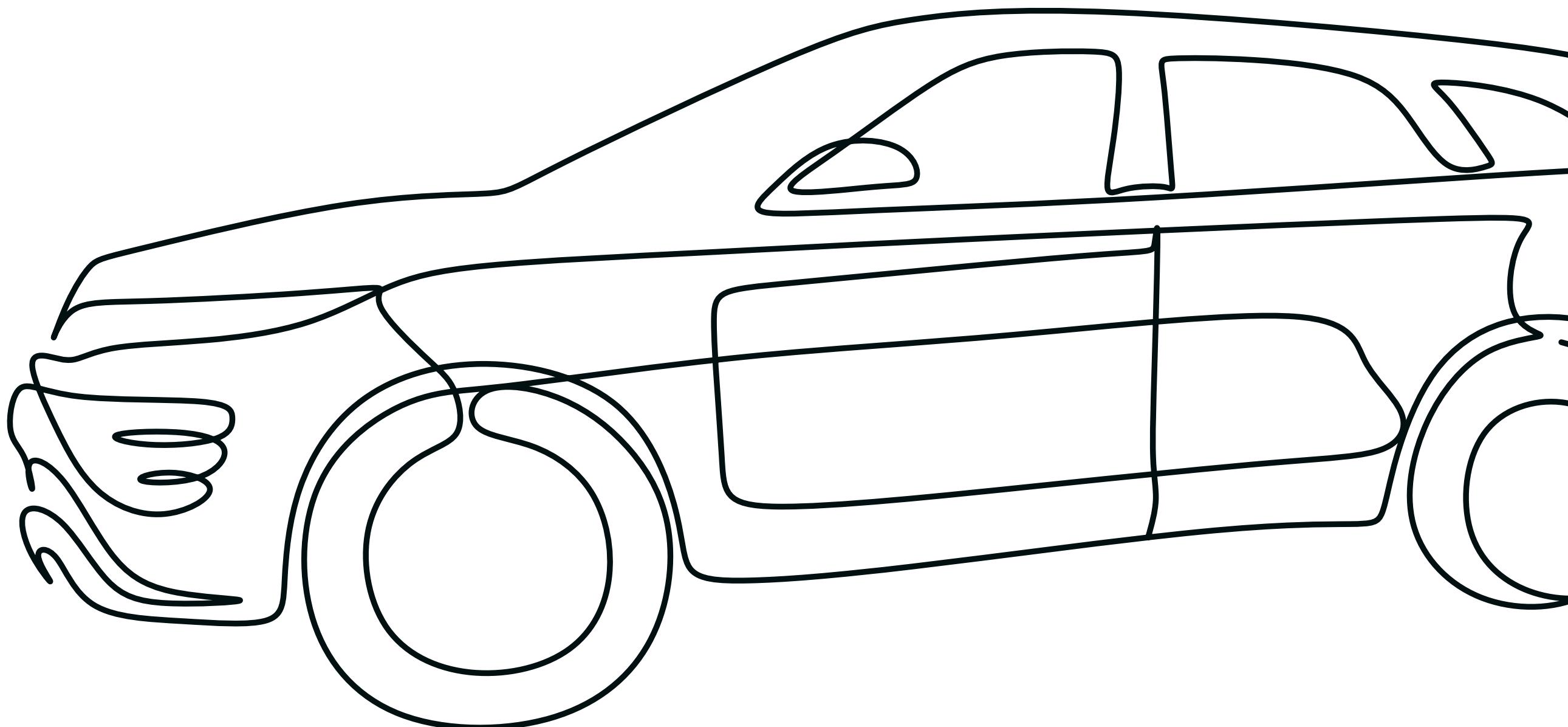
Sistema de Cadastro de Clientes para Oficina Mecânica

José Jonas

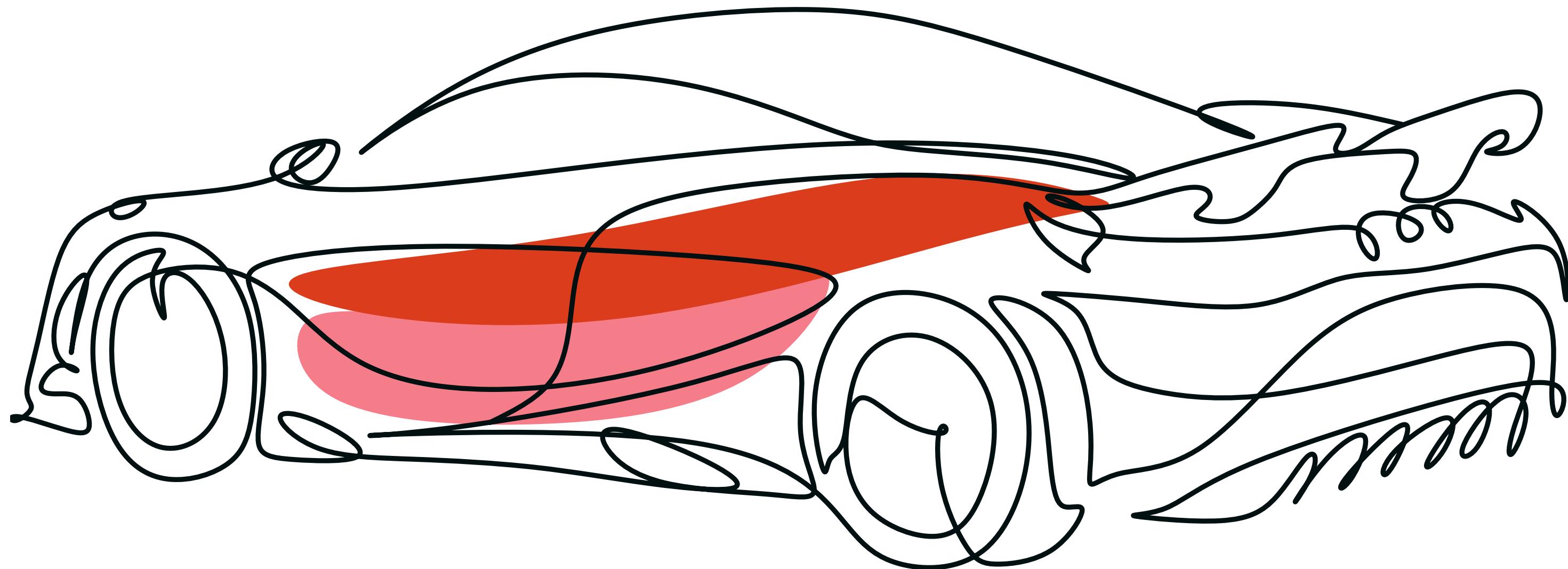
Endel Azevedo

Fábio Augusto

Gustavo Eufrázio



ESTRUTURAS PRINCIPAIS:



Bibliotecas e variáveis:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <locale.h>
#include <ctype.h>

#define ARQUIVO_CSV "clientes_oficina.csv"

#define MAX_NOME 100
#define MAX_DATA 30
#define MAX_TELEFONE 15
#define MAX_EMAIL 100
#define MAX_MODELO 30
#define MAX_COR 20
#define MAX_PLACA 8
#define MAX_SERVICO 1000
```

Estrutura que representa um cliente:

```
typedef struct ElementoLista {
    char nome[MAX_NOME];
    char dataCadastro[MAX_DATA];
    char telefone[MAX_TELEFONE];
    char email[MAX_EMAIL];
    char modeloCarro[MAX_MODELO];
    char corCarro[MAX_COR];
    char placaCarro[MAX_PLACA];
    char servicoRealizado[MAX_SERVICO];

    struct ElementoLista* proximo;
    struct ElementoLista* anterior;
} ElementoLista;

typedef struct Lista {
    ElementoLista* inicio;
    ElementoLista* fim;
} Lista;
```

Gerenciamento da lista:

```
// Cria lista vazia
Lista* criarLista() {
    Lista* l = (Lista*)malloc(sizeof(Lista));
    if (!l) {
        perror("Erro ao alocar lista");
        exit(EXIT_FAILURE);
    }
    l->inicio = NULL;
    l->fim = NULL;
    return l;
}

// Libera toda a lista
void liberarLista(Lista* lista) {
    if (!lista) return;
    ElementoLista* cur = lista->inicio;
    while (cur) {
        ElementoLista* nxt = cur->proximo;
        free(cur);
        cur = nxt;
    }
    free(lista);
}
```

Salvando o arquivo em CSV

```
void salvarEmArquivo(Lista* lista) {
    if (!lista) return;
    FILE *f = fopen(ARQUIVO_CSV, "w");
    if (!f) {
        perror("Erro ao abrir arquivo para salvar");
        return;
    }

    // Cabeçalho opcional
    fprintf(f, "\"Nome\";\"DataCadastro\";\"Telefone\";\"Email\";\"ModeloCarro\";\"CorCarro\";\"PlacaCarro\";\"Serviço\"\n");

    ElementoLista* cur = lista->inicio; //cur aponta para a posição inicial da lista
    while (cur) {
        grava_campo_csv(f, cur->nome); fputc(';', f);
        grava_campo_csv(f, cur->dataCadastro); fputc(';', f);
        grava_campo_csv(f, cur->telefone); fputc(';', f);
        grava_campo_csv(f, cur->email); fputc(';', f);
        grava_campo_csv(f, cur->modeloCarro); fputc(';', f);
        grava_campo_csv(f, cur->corCarro); fputc(';', f);
        grava_campo_csv(f, cur->placaCarro); fputc(';', f);
        grava_campo_csv(f, cur->servicoRealizado); fputc('\n', f);
        cur = cur->proxima;
    }

    fclose(f);
}

// Carrega dados do CSV para a lista
void carregarDeArquivo(Lista* lista) {
    if (!lista) return;
```

Exibir um cliente:

```
void exibirCliente(ElementoLista* c) {
    if (!c) return;
    printf("\n-----\n");
    printf("Nome: %s\n", c->nome);
    printf("Data de Cadastro: %s\n", c->dataCadastro);
    printf("Telefone: %s\n", c->telefone);
    printf("Email: %s\n", c->email);
    printf("Modelo do carro: %s\n", c->modeloCarro);
    printf("Cor do carro: %s\n", c->corCarro);
    printf("Placa do carro: %s\n", c->placaCarro);
    printf("Serviço(s) realizado(s): %s\n", c->servicoRealizado);
    printf("-----\n");
}
```

Exibir de toda lista:

```
void exibirLista(Lista* lista) {
    if (!lista) return;
    if (lista->inicio == NULL) {
        limpa_tela();
        printf("\nLista Vazia!\n");
        return;
    }
    limpa_tela();
    ElementoLista* aux = lista->inicio;
    int cont = 1;
    printf("\n--- Lista de Clientes ---\n");
    while (aux) {
        printf("\nCliente %d:\n", cont++);
        exibirCliente(aux);
        aux = aux->proximo;
    }
    printf("\nQuantidade de contatos: %d\n", cont-1);
    printf("\nPressione Enter para continuar... ");
    getchar();
    limpa_tela();
}
```

```

void inserirCliente(Lista* lista) {
    if (!lista) return;
    ElementoLista* novo = (ElementoLista*) malloc(sizeof(ElementoLista));
    if (!novo) {
        perror("Erro ao alocar memória");
        return;
    }
    limpa_tela();
    printf("\n--- Inserir Cliente ---\n");
    printf("Nome: ");
    leitura_segura(novo->nome, sizeof(novo->nome));
    printf("Data de cadastro (DD/MM/AAAA): ");
    leitura_segura(novo->dataCadastro, sizeof(novo->dataCadastro));
    printf("Telefone: ");
    leitura_segura(novo->telefone, sizeof(novo->telefone));
    printf("Email: ");
    leitura_segura(novo->email, sizeof(novo->email));
    printf("Modelo do carro: ");
    leitura_segura(novo->modeloCarro, sizeof(novo->modeloCarro));
    printf("Cor do carro: ");
    leitura_segura(novo->corCarro, sizeof(novo->corCarro));
    printf("Placa do carro: ");
    leitura_segura(novo->placaCarro, sizeof(novo->placaCarro));
    printf("Serviço(s) realizado(s): ");
    leitura_segura(novo->servicoRealizado, sizeof(novo->servicoRealizado));
    novo->proxima = NULL;
    novo->anterior = lista->fim;

    if (lista->fim == NULL)
        lista->inicio = novo;
    else
        lista->fim->proxima = novo;
    lista->fim = novo;
    // SALVAMENTO AUTOMÁTICO após inserção
    salvarEmArquivo(lista);
    limpa_tela();
    printf("\nContato salvo!\n");
}

```

Função para inserir cliente:

```

void buscar_exibir_cliente(Lista* lista) {
    if (!lista) return;
    if (lista->inicio == NULL) {
        limpa_tela();
        printf("\nLista Vazia!\n");
        return;
    }
    char nomeBusca[MAX_NOME];
    printf("Digite o nome do cliente que deseja buscar: ");
    leitura_segura(nomeBusca, sizeof(nomeBusca));

    ElementoLista* aux = lista->inicio;
    while (aux) {
        if (compara_ignorando_case(aux->nome, nomeBusca) == 0) {
            printf("\nCliente encontrado:\n");
            exibirCliente(aux);
            printf("\nPressione Enter para continuar...");
            getchar();
            limpa_tela();
            return;
        }
        aux = aux->proximo;
    }

    printf("\nCliente não encontrado.\n");
    printf("\nPressione Enter para continuar...");
    getchar();
    limpa_tela();
}

```

função para exibir o cliente:

```

void remover_elemento(Lista* lista, ElementoLista* e) {
    if (!lista || !e) return;

    // Ajusta o ponteiro 'proxima' do elemento anterior
    if (e->anterior) e->anterior->proxima = e->proxima;
    else lista->inicio = e->proxima; // 'e' era o primeiro, o novo primeiro é o proximo

    // Ajusta o ponteiro 'anterior' do elemento seguinte
    if (e->proxima) e->proxima->anterior = e->anterior;
    else lista->fim = e->anterior; // 'e' era o ultimo, o novo ultimo é o anterior

    free(e);
}

```

função pra remover elementos

```

void limparTodaLista(Lista* lista) {
    if (!lista) return;
    char resp[8];
    printf("Tem certeza que deseja apagar TODOS os clientes? (S/N): ");
    leitura_segura(resp, sizeof(resp));

    if (resp[0] != 'S' && resp[0] != 's') {
        printf("Operação cancelada.\n");
        return;
    }

    ElementoLista* cur = lista->inicio;
    while (cur) {
        ElementoLista* nx = cur->proxima;
        free(cur);
        cur = nx;
    }
    lista->inicio = lista->fim = NULL;
    //salva automatico apos remocao completa
    salvarEmArquivo(lista);
    printf("Lista completamente removida!\n");
}

```

função pra limpar lista

```

void percorrerClientes(Lista* lista) {
    if (!lista || !lista->inicio) {
        limpa_tela();
        printf("\nLista Vazia!\n");
        return;
    }

    ElementoLista* atual = lista->inicio;
    int indice = 1;
    limpa_tela();
    printf("Contato %d:\n", indice);
    exibirCliente(atual);

    int tecla;
    do {
        printf("\nUse as setas ( <-- | --> ) para navegar ou ESC para sair.\n");
        tecla = getch();
        if (tecla == 0 || tecla == 224) {
            tecla = getch();
            switch (tecla) {
                case 77: // direita
                    if (atual->proxima) {
                        atual = atual->proxima;
                        indice++;
                        limpa_tela();
                        printf("Contato %d:\n", indice);
                        exibirCliente(atual);
                    } else {
                        printf("\nFIM DA LISTA\n");
                    }
                    break;
                case 75: // esquerda
                    if (atual->anterior) {
                        atual = atual->anterior;
                        indice--;
                        limpa_tela();
                        printf("Contato %d:\n", indice);
                        exibirCliente(atual);
                    } else {
                        printf("\nINÍCIO DA LISTA\n");
                    }
                    break;
            }
        }
    } while (tecla != 27); // enquanto não apertrem ESC
    limpa_tela();
}

```

Função pra percorrer a lista de clientes:

```
void buscar_editar_ou_remover(Lista* lista) {
    if (!lista || !lista->inicio) {
        limpa_tela();
        printf("\nLista Vazia!\n");
        return;
    }

    char nomeBusca[MAX_NOME];
    printf("Digite o nome do contato: ");
    leitura_segura(nomeBusca, sizeof(nomeBusca));

    ElementoLista* atual = lista->inicio;
    while (atual) {
        if (compara_ignorando_case(atual->nome, nomeBusca) == 0) {
            printf("\nContato encontrado:\n");
            exibirCliente(atual);

            printf("\nEscolha uma opção:\n");
            printf("1. Editar\n");
            printf("2. Remover\n");
            printf("3. Cancelar\n");
            printf("Opção: ");
        }
    }
}
```

continue...:

```
char optbuf_str[8];
leitura_segura(optbuf_str, sizeof(optbuf_str));
int opcao = atoi(optbuf_str); // Converte string para int

if (opcao == 1) {
    printf("\nO que deseja editar?\n");
    printf("1. Nome\n2. Data de Cadastro\n3. Telefone\n4. Email\n5. Modelo do carro\n6. Cor do carro\n7. Placa do carro\n8. Servico realizado\n9. Tudo\n0. Voltar\n");
    printf("Opção: ");

leitura_segura(optbuf_str, sizeof(optbuf_str));
int ed = atoi(optbuf_str);
switch (ed) {
    case 1: printf("Digite o novo nome: "); leitura_segura(atual->nome, sizeof(atual->nome)); break;
    case 2: printf("Digite a nova data: "); leitura_segura(atual->dataCadastro, sizeof(atual->dataCadastro)); break;
    case 3: printf("Digite o novo telefone: "); leitura_segura(atual->telefone, sizeof(atual->telefone)); break;
    case 4: printf("Digite o novo email: "); leitura_segura(atual->email, sizeof(atual->email)); break;
    case 5: printf("Digite o modelo do carro: "); leitura_segura(atual->modeloCarro, sizeof(atual->modeloCarro)); break;
    case 6: printf("Digite a cor do carro: "); leitura_segura(atual->corCarro, sizeof(atual->corCarro)); break;
    case 7: printf("Digite a placa do carro: "); leitura_segura(atual->placaCarro, sizeof(atual->placaCarro)); break;
    case 8: printf("Digite o(s) serviço(s): "); leitura_segura(atual->servicoRealizado, sizeof(atual->servicoRealizado)); break;
    case 9:
        printf("Nome: "); leitura_segura(atual->nome, sizeof(atual->nome));
        printf("Data de cadastro: "); leitura_segura(atual->dataCadastro, sizeof(atual->dataCadastro));
        printf("Telefone: "); leitura_segura(atual->telefone, sizeof(atual->telefone));
        printf("Email: "); leitura_segura(atual->email, sizeof(atual->email));
        printf("Modelo do carro: "); leitura_segura(atual->modeloCarro, sizeof(atual->modeloCarro));
        printf("Cor do carro: "); leitura_segura(atual->corCarro, sizeof(atual->corCarro));
        printf("Placa: "); leitura_segura(atual->placaCarro, sizeof(atual->placaCarro));
        printf("Serviço(s): "); leitura_segura(atual->servicoRealizado, sizeof(atual->servicoRealizado));
        break;
}
```

```
int main() {
    setlocale(LC_ALL, "portuguese");

    Lista* lista = criarLista();
    carregarDeArquivo(lista);

    char optbuf[16];
    int opcao = -1;

    for (;;) {
        printf("\n\n-----\n");
        printf("      Oficina ESPARTANOS!\n");
        printf("-----\n");

        printf("Menu:\n");
        printf("1. Exibir lista de clientes\n");
        printf("2. Inserir cliente\n");
        printf("3. Buscar e exibir cliente\n");
        printf("4. Remover lista completa de clientes\n");
        printf("5. Percorrer lista de clientes (setas)\n");
        printf("6. Editar ou Remover cliente (buscar)\n");
        printf("0. Sair\n");
        printf("Escolha uma opção: ");

        leitura_segura(optbuf, sizeof(optbuf));
        opcao = atoi(optbuf);
```

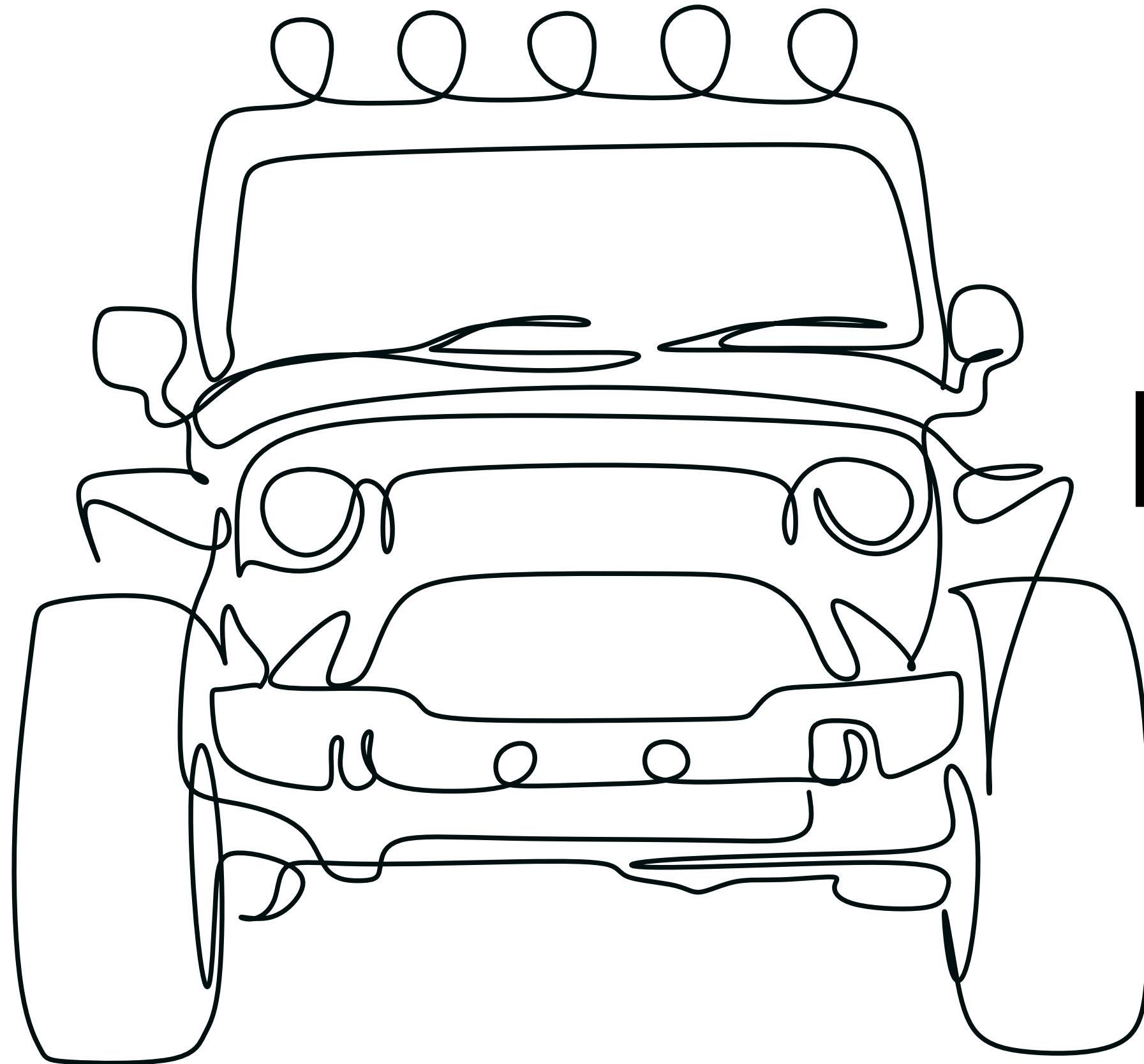
```
switch (opcao) {
    case 1: exibirLista(lista); break;
    case 2: inserirCliente(lista); break;
    case 3: buscar_exibir_cliente(lista); break;
    case 4: limpa_tela(); limparTodaLista(lista); break;
    case 5: percorrerClientes(lista); break;
    case 6: buscar_editar_ou_remover(lista); break;
    case 0:
        printf("Saindo...\n");
        // garante salvar ao sair
        salvarEmArquivo(lista);
        liberarLista(lista);
        return 0;
    default:
        printf("Opção inválida! Tente novamente.\n");
        break;
}
return 0;
```

Segurança para evitar Buffer Overflow e não sensível a maiúsculas e minúsculas:

```
static void remove_newline(char *s) {
    if (!s) return;
    size_t len = strlen(s);
    if (len == 0) return;
    if (s[len-1] == '\n') s[len-1] = '\0';
}

static void leitura_segura(char *buf, size_t size) {
    if (!fgets(buf, (int) size, stdin)) {
        buf[0] = '\0';
        return;
    }
    remove_newline(buf);
}

static int compara_ignorando_case(const char *a, const char *b) {
    while (*a && *b) {
        unsigned char ca = (unsigned char) tolower((unsigned char)*a);
        unsigned char cb = (unsigned char) tolower((unsigned char)*b);
        if (ca != cb) return ca - cb;
        a++; b++;
    }
    return (unsigned char) tolower((unsigned char)*a) - (unsigned char) tolower((unsigned char)*b);
}
```



PERGUNTAS?