Instituto Politecnico de Coimbra

Instituto Superior de Engenharia

Curso Engenharia Informática - Regime Pós Laboral

Ano Letivo: 2017/2018

Relatório de Trabalho Prático

"Guitarras p'Alugar"

Filipe A. N. Silva

Nr: 21260367

Coimbra, 13 de junho de 2018

INTRODUÇÃO

Índice

INTRODUÇÃO

Este trabalho prático foi realizado com o intuito de abordar a cadeira de Programação do segundo semestre do primeiro ano no curso de Engenharia Informática do Instituto Superior de Engenharia de Coimbra.

Foi proposto com este trabalho o desenvolvimento de uma aplicação para gerir o negócio de uma loja de aluguer de guitarras. Usando esta aplicação, o utilizador conseguirá mais facilmente gerir os seus alugueres de uma maneira eficaz e simples, onde esta aplicação organizará tanto as guitarras disponiveis como os seus clientes.

O programa foi feito utilizando a linguagem C através do IDE Netbeans. Como referido no enunciado este proprama será chamado “Guitarras p’Alugar”.

ESTRUTURAS DE DADOS

O programa baseia-se principalmente em 3 tipos de Estruturas de dados sendo que estas utilizam outras.

As tres principais estruturas de dados são as guitarras, clientes e alugueres: sendo a estrutura guitarras dependente só de si, e as restantes estruturas estão interligadas; a estrutura Cliente está ligada à estrutura aluguer podendo esta apresentar valores ou NULL; a estrutura aluguer por sua vez conte dentro de si outra estrutura do tipo Data.

ESTRUTURAS DE DADOS

Estrutura Guitarra

Esta estrutura tem como função armazenar a informação de cada guitarra.

```
struct Guitarra {
    int tam;
    int id;
    int precoDia;
    int valor;
    int estado;
    char nome[TAM];
};
```

- A varivel "tam" do tipo int declara o tamanho do vetor.
- A variavel "id" do tipo int declara a identificação única de cada guitarra.
- A variavel "precoDia" do tipo int declara o preço do aluguer por dia de cada guitarra.
- A variavel "valor" do tipo int declara o valor total de cada guitarra.
- A variavel "estado" do tipo int declara o estado de cada guitarra, isto é, (0 - disponível / 1 - alugada / 2 - danificada).
- A variavél "nome" do tipo char tem como função armazenar o nome de cada guitarra.

Com o objectivo de explicar este método, decidi utilizar estas variáveis pois a partir destas é possivel efetuar todas as operações necessarias no programa, ou seja, p.e. caso exista uma guitarra com o mesmo nome o id irá diferencia-las uma da outra pois cada id é único.

ESTRUTURAS DE DADOS

Estrutura Data

Esta estrutura tem como função armazenar a data de um dia.

```
struct Data {
    int dia;
    int mes;
    int ano;
};
```

- A varivel "dia" do tipo int declara o dia de um ano.
- A varivel "mes" do tipo int declara o mês de um ano.
- A varivel "ano" do tipo int declara o ano.

Esta estrutura será util na estrutura Aluguer pois é necessaria para registar o dia do inicio e da conclusao de um aluguer.

ESTRUTURAS DE DADOS

Estrutura Aluguer

Esta estrutura tem como função armazenar a informação de um aluguer.

```
struct Aluguer {
    int estado;
    int idGuitarra;
    data dataInicio;
    data dataEntrega;
    aluguer *prox;
};
```

- A varivel "estado" do tipo int declara o estado de um aluguer , isto é, (0 – a decorrer / 1 – entregue / 2 – entregue danificada)
- A varivel "idGuitarra" do tipo int é um identificar da guitarra do aluguer.
- A varivel "dataInicio" do tipo data (anteriormente explicada) declara o inicio de um aluguer.
- A varivel "dataEntrega" do tipo data (anteriormente explicada) declara o fim de um aluguer.
- A varivel "prox" do tipo aluguer é um ponteiro do tipo Aluguer, ou seja , um ponteiro da própria estrutura, será utilizado para correr a lista Aluguer.

Esta estrutura será a base para a gestão dos alugueres do programa sendo que a cada aluguer todas as variaveis são preenchidas contudo caso não ainda não tenha havido entrega a variável dataEntrega, será colocada a 0 todas as váriaveis da sua estrutura.

ESTRUTURAS DE DADOS

Estrutura Cliente

Esta estrutura tem como função armazenar a informação de cada cliente.

```
struct Cliente {
    int nif;
    int nAlugueres;
    char nome[TAM];
    aluguer *alu;
    cliente *prox;
};
```

- A varivel "nif" do tipo int declara o identificador único de cada cliente.
- A varivel "nAlugueres" do tipo int declara o o numero total de alugueres de cada cliente.
- A varivel "nome" do tipo char declara o nome de cada cliente.
- A varivel "alu" do tipo Aluguer é um ponteiro para a estrutura aluguer podendo estar a apontar para um espaço de memória ou estar a NULL.
- A varivel "prox" do tipo Cliente é um ponteiro do tipo Cliente, ou seja , um ponteiro da própria estrutura, será utilizado para correr a lista Cliente.

Esta estrutura será utilizada para criar uma estrutura dinamica do tipo lista ligada de 2 niveis. Irá correr os clientes existentes e para cada um destes, poderá correr todos os seus alugueres, caso não tenham alugueres o ponteiro alu será colocado a NULL.

ESTRUTURAS DINÂMICAS

No âmbito de criar o programa mais fluido e eficaz o programa será gerido de forma dinâmica, assim evitando desperdícios de memória, a alocação dinâmica de memória deve ser sempre utilizada quando não se sabe ao certo quanto espaço de memória será necessário durante a execução do programa e conforme a necessidade de armazenamento.

Este tipo de alocação evita desperdicios de recursos pois a gestão de memória será alternada dependendo da necessidade de memória ao longo do programa assim apenas alocando ou libertar memória quando necessário. Com a utilização deste tipo de estruturas, o programador tem a possibilidade de, ao contrário da utilização de *varivaeis estáticas* onde a memória reservada é igual, manusear a memória do computador de forma a conseguir utilizar e alocar/realocar a memória necessária, e assim executar aquilo que é a vontade do programador.

Neste programa será implementado 2 estruturas dinâmicas. A informação das guitarras será armazenada num vetor dinâmico e a informação dos clientes e respetivos alugueres será mantida numa estrutura dinâmica do tipo lista ligada de 2 níveis, na gestão de memória das estruturas será utilizado 2 funções, malloc e realloc.

```
void *malloc(size_t size)
```

A função devolve um ponteiro para a memória alocada.

Caso haja um erro na alocação é devolvido NULL

Numero de bytes a ser reservados na memória

```
void *realloc(void *ptr, size_t size)
```

A função devolve um ponteiro para a nova memória alocada

Ponteiro para o bloco de memória que se deseja alterar

Numero de bytes a ser reservados na memória

```
void free (void *ptr)
```

Esta função não devolve nenhum valor

Ponteiro para o espaço de memória a libertar

ESTRUTURAS DINÂMICAS

Vetor Dinâmico

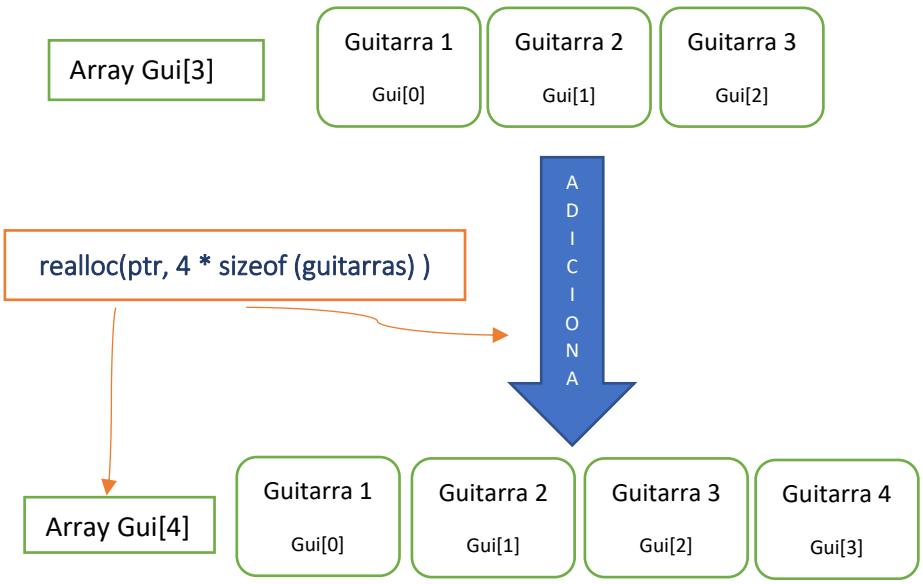
O funcionamento de um vetor dinâmico é idêntico ao funcionamento de um vetor digamos estático, diferenciando no facto de poder variar o seu tamanho.

Neste tipo de memória é alocado memória sequencialmente/em blocos.

Como já foi dito anteriormente, a informação das guitarras será armazenada num vetor dinâmico.

Este vetor é criado no inicio da execução do programa. O programa abre o ficheiro guitarras, caso exista, e lê quantas guitarras existe e cria um vetor dinâmico do tipo guitarras com o tamanho do numero de guitarras lidas.

Caso não exista ficheiro ou guitarras o programa preenche uma guitarra para o inicio do vetor com valores irrelevantes, que serão subestituidos por uma futura guitarra ou caso não seja adicionado nenhuma guitarra será desalocado a memória deste, o vetor dinâmico, sempre que seja necessário adicionar ou remover elementos ira realocar a sua memória, ou seja, a memória ajusta-se à necessidade do array, seja adicionar elementos ou elimina-los.



ESTRUTURAS DINÂMICAS

Listas Ligadas

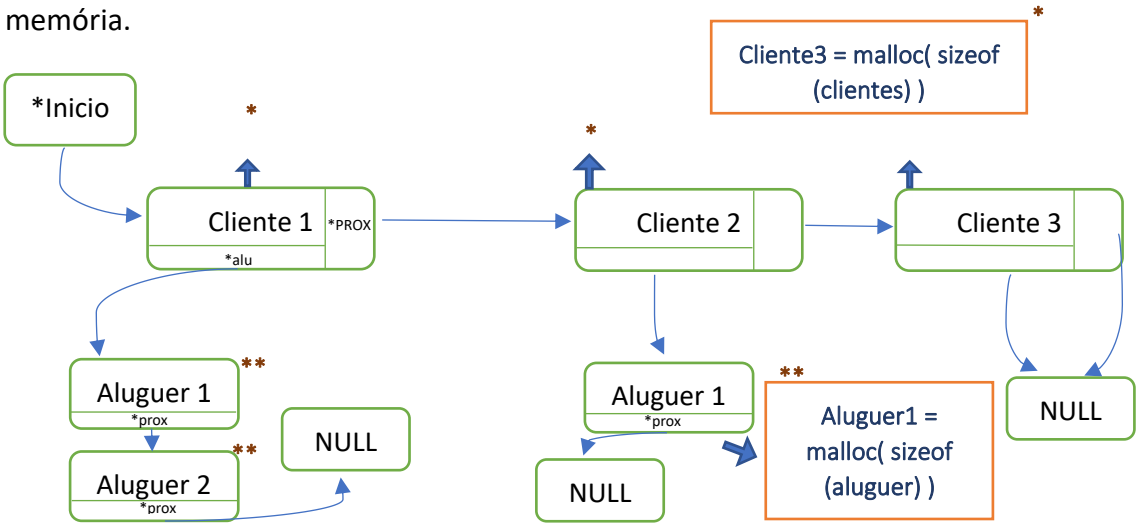
Como já foi dito anteriormente, a informação dos clientes e seus alugueres será armazenada em listas ligadas, sendo esta de dois niveis, sendo assim uma lista de listas, isto é, haverá uma lista alugueres dentro de uma lista clientes.

Numa estrutura tipo lista ligada cada elemento tem memória reservada para si apenas,ou seja, contráriamente ao vetor dinâmico as listas ligadas não estão ordenadas em blocos na memória, mas sim disperças na memória interligadas por ponteiros. O primeiro elemento da lista é a fundação da lista, pois caso aconteça algo perde-se a estrutura toda, sendo que no último elemento o ponteiro aponta para *NULL*.

Esta lista é criado no inicio da execução do programa. O programa abre o ficheiro clientes, caso exista, e passa a informação da primeira linha (referente aos clientes) para variàveis locais, após preencher as variaveis locais é alocado memória para a estrutura clientes passando de imediato a informação das variaveis locais para a lista.

Após ser preenchido a informação do cliente será verificado o numero de alugueres,já declarado na estrutura Clientes, lendo assim o numero de linhas consuante o numero de alugueres. Caso haja alugueres ou seja, é de imediato alocado memória para preencher a lista alugueres. Primeiramente é lido o id da guitarra e o seu estado, de seguida é passado a restante linha para uma variavel, onde esta de seguida será formatada consuate o numero de variaveis que contem, ou seja, caso seja um aluguer com entrega ou não.

Caso não exista ficheiro ou clientes o ponteiro fica a NULL onde poderá futuramente ser utilizado para alocação de memória.



FICHEIROS UTILIZADOS

Neste programa é utilizado 3 ficheiros sendo 2 destes ficheiros de texto e o outro um ficheiro binário.

Os 2 ficheiros de texto são relativamente às guitarras e aos clientes, estes ficheiros encontram-se na directoria "ficheiros_texto".

O ficheiro binário refere-se aos clientes banidos e encontra-se na directoria "ficheiro_binario".

Será utilizado 6 funções para gerir os ficheiros. Começando pelas funçõs de abertura e encerramento sendo estas fopen,fclose respetivamente. As funções fprint para escrita e fscanf para leitura nos ficheiros de texto. Concluindo com as funções fread e fwrite para leitura e escrita no ficheiro binário, respetivamente.

```
FILE *fopen(const char *nome, const char *modo)
```

A função devolve um ponteiro do tipo FILE. Caso haja erro a abrir a função devolve NULL

Uma string contento o nome do ficheiro a abrir.

String a determinar o modo de abertuda podendo este ser:

- r (leitura) - w (escrita)

- a (acrescentar)

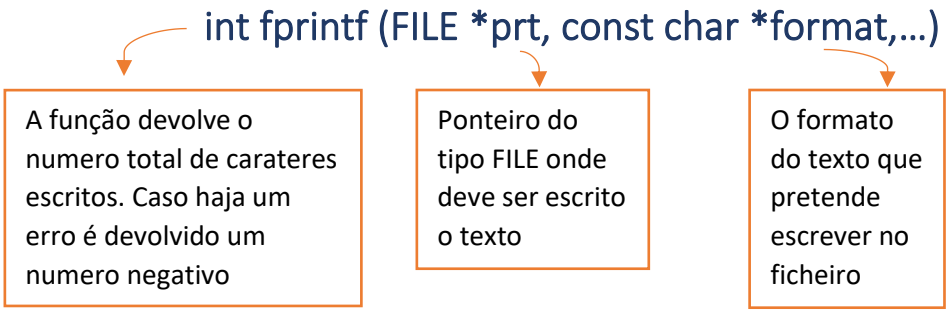
Seguido de:
- t (modo texto) - b (modo binário)
Caso não seja especificado este ultimo será aberto em modo texto

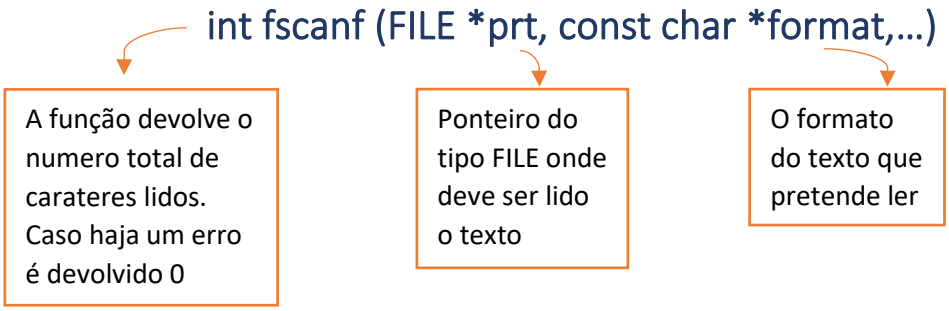
```
FILE *fclose(FILE *ptr)
```

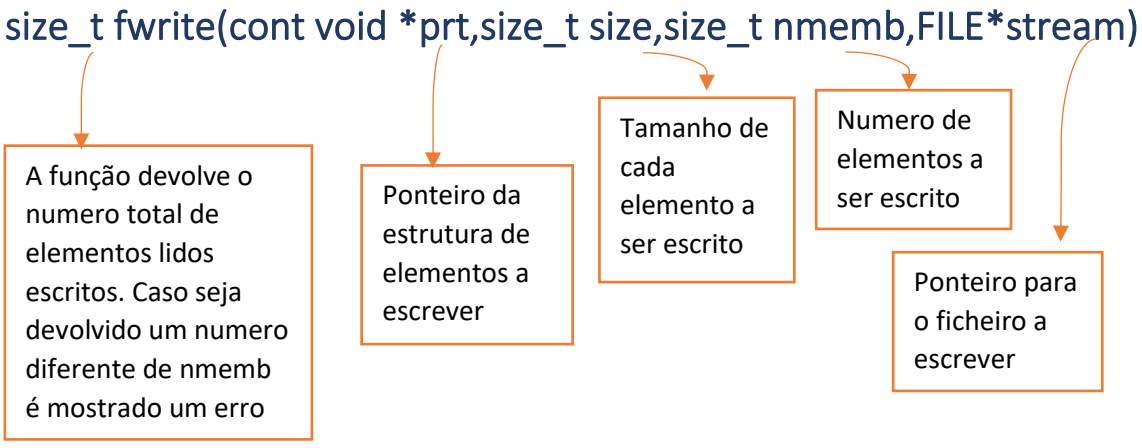
A função devolve 0. Caso haja erro a fechar a funºão devolver EOF

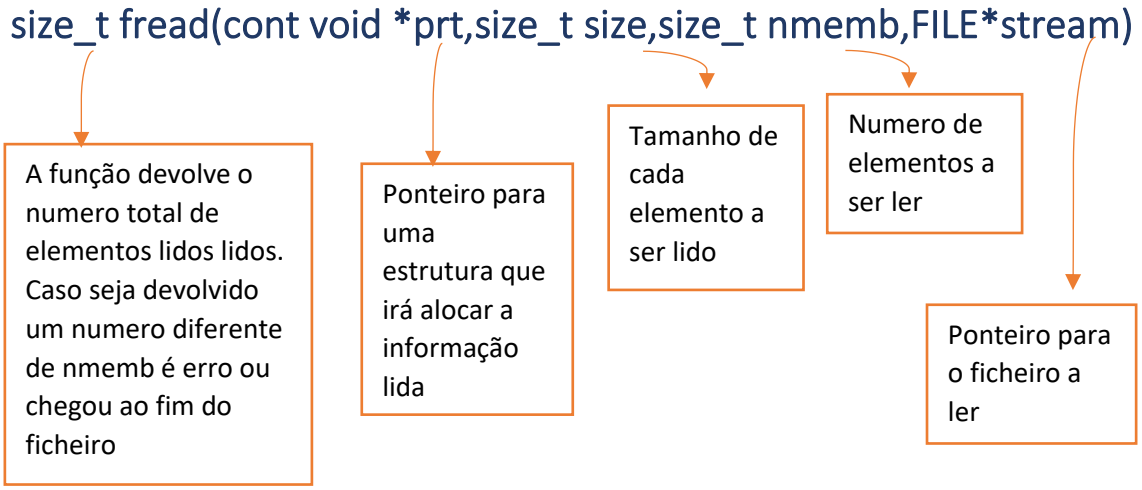
O ponteiro do ficheiro tipo FILE que pretende fechar

FICHEIROS UTILIZADOS









FICHEIROS UTILIZADOS

Ficheiro Guitarras

O ficheiro guitarras é um ficheiro de texto. O ficheiro denomina-se "guitarras" e encontra-se na directoria com o nome "ficheiros_texto", este ficheiro tem todas as informações relativas às guitarras.

O ficheiro poderá ou não existir, caso não exista será criado um ficheiro de texto com o nome "guitarras".

O ficheiro guitarras terá a seguinte estrutura:

```
ID PrecoPorDia Valor Estado Nome
```

O *ID* refere-se ao id da guitarra, o *PrecoPorDia* refere-se ao preço por dia de cada guitarra, o *Valor* refere-se ao valor respetivo à guitarra, o *Estado* refere-se ao estado da guitarra, podendo esta estar disponivel, alugada ou danificada, atribuindo a este campo 0,1,2 respetivamente e finalmente o nome da guitarra.

O programa será inicializado pela primeira vez com o seguinte ficheiro guitarras:

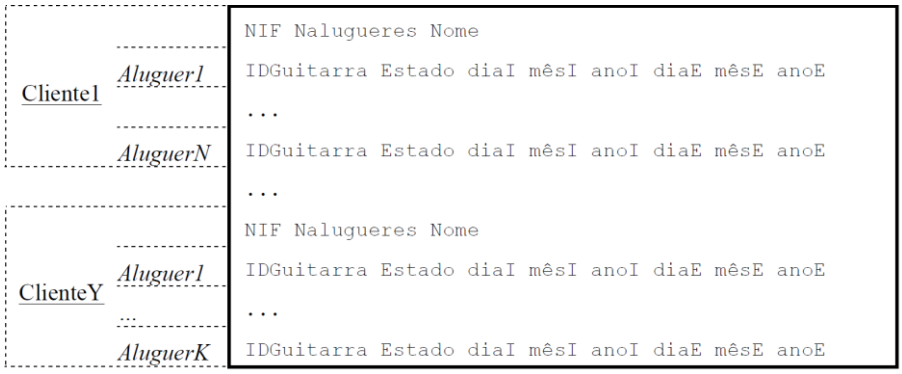
```
1       10      1000    0       LUCILLE
4       100     500     1       BETTY JEAN
5       25      350     1       ROCKY
```

FICHEIROS UTILIZADOS

Ficheiro Clientes

O ficheiro clientes é um ficheiro de texto.O ficheiro denomina-se “clientes” e encontra-se na directoria com o nome “ficheiros_texto”, este ficheiro contém todas as informações relativas aos clientes e alugueres.

O ficheiro clientes poderá ou não existir, caso não exista será criado um ficheiro de texto com o nome “clientes”.



O ficheiro guitarras terá a seguinte estrutura:

Àrea relativa aos clientes:

O *NIF* refere-se ao nif de cada cliente sendo este único, o *Nalugueres* refere-se ao nr de alugueres total de cada cliente,ou seja, alugueres a decorrer mais alugueres concluidos , o *Nome* refere-se ao nome do cliente.

Àrea relativa aos alugueres:

O *IDGuitarra* refere-se ao id da guitarra, o *Estado* sendo este o situação da guitarra relativamente ao aluguer ou seja, a decorrer, entregue ou entregue danificada, *diaI,mêsI,anoI, refere-se à data do Inicio do aluguer e concluindo com diaE,mêsE,anoE, referindo à data da entrega da guitarra, podendo este campo estar preenchido ou vazio.*

O programa será inicializado pela primeira vez com o seguinte ficheiro clientes:

```
123456789       0       Paulo Silva

555666777       2       Marta Nunes Cabral
4   1   12  12  2017    17  12  2017
5   0   20  04  2018

123123123       3       Maria Pimentel
5   2   01  10  2017    10  10  2017
5   1   13  01  2018    15  01  2018
4   0   17  04  2018
```

FICHEIROS UTILIZADOS

Ficheiro Banidos

O ficheiro banidos é um ficheiro binário. O ficheiro denomina-se “banidos” e encontra-se na directoria com o nome “ficheiro_binario”, este ficheiro contém todas as informações relativas aos clientes banidos, os ficheiros banidos poderá ou não existir, caso não exista será criado um ficheiro binário com o nome “banidos”.

O ficheiro terá como conteudo a estrutura clientes e será utilizado logo no inicio da execução do programa para verificar se existem clientes banidos no vetor de clientes, será tambem aberto na tentativa de adicionar um novo cliente verificando se o utilizador está a tentar adicionar um cliente banido.

ESTRUTURA DO PROGRAMA

O programa está estruturado em 3 fases, a parte na inicialização do programa, a fase intermédia que consiste na interação do utilizador com o programa, o programa termina com a fase da finalização do programa.

A parte inicial e final do programa é automática na qual o utilizador não intrevêm, ou seja, o utilizador apenas interage com as funções da fase intermédia.

Dividindo o programa 3 fases, cada uma tem as seguintes funções:

Fase inicial:

iniciaGui:

- criaFichGui
- contaGuitarras
- criaVetGui
- preencheGuitarra
- vetGuiCriaVetNovo

iniciaCli:

- criaFichCli
- preencheCliente
- eliminaBanidos
- verificaBanidos

Fase intermédia:

Guitarras:

- vverExisteGui
- adicionarGuitarra
- historico
- listarGuitarras
- guitarrasAlugadas
- arranjaGui

Clientes:

- calculaAtraso
- correLista
- verExisteCli
- adicionarClientes
- eliminarCliente
- mostrarEstdos
- listarClientes
- mostrarBanidos

Alugueres:

- dataHoje
- calculaTempo
- dataPrevistaEntrega
- banir
- banimentoAuto
- novoAluguer
- VerExisteCli
- criaAluguer
- concluiAluguer
- listarTodosAlugueres
- listarAlugueresAtuais
- alterarMaxAlu
- guitarrasbaratas
- nrGuitarrasbaratas

Fase Final:

atualizaFicheiros:

- atualizaGuitarras
- atualizaClientes

ESTRUTURA DO PROGRAMA

Fase inicial

A fase inicial consiste na criação e criação do vetor dinâmico das guitarras e na criação das listas ligadas referentes aos clientes e alugueres.

Guitarras

O programa começa primeiramente pela criação do vetor guitarras. A função iniciaGui começa primeiramente por contar o número de guitarras caso não exista o fixeiro de texto pertencente às guitarras cria de seguida este mesmo. Após contar o numero de guitarras cria o vetor dinâmico. A função finaliza retomando o ponteiro do vetor.

```
guitarra* iniciaGui() {
    int contaGui;
    guitarra *vetGui;

    contaGui = contaGuitarras();
    if (contaGui == 0) {
        vetGui = vetGuiCriaVetGuiNovo();
    } else {
        vetGui = criaVetGui(contaGui);
        preencheGuitarra(vetGui);
    }

    return vetGui;
}
```

A função começa por contar o numero de guitarras existentes no ficheiro e devolve o numero de guitarras lidas. Dependendo do numero de guitarras lidas é chamada as funções para, caso não exista guitarras, criar um elemento para o inicio do vetor dinâmico, ou caso exista guitarras, cria o vetor dinamico do tipo guitarras com o tamanho do numero de guitarras existentes e de seguida preenche-o

```
int contaGuitarras() {
    int id, precoDia, valor, estado, conta = 0;
    char nome[TAM];

    FILE *f;
    f = fopen("ficheiros_texto\\guitarras.txt", "rt");
    if (f == NULL) {
        criaFichGui();
        return 0;
    } else {
        while (fscanf(f, " %d %d %d %d %[^\n]", &id, &precoDia, &valor, &estado, nome) == 5) {
            conta++;
        }
    }

    fclose(f);
    return conta;
}
```

A função contaGuitarras conta o numero de linhas lidas no ficheiro. Caso não exista o ficheiro acede à função criaFichGui para criar o ficheiro guitarras

```c
guitarra* vetGuiCriaVetGuiNovo() {
    guitarra *aux;

    aux = malloc(sizeof (guitarra));
    if (aux == NULL) {
        puts("erro a criar vetor guitarras!");
        return NULL;
    }

    aux[0].estado = 0;
    aux[0].id = 0;
    strcpy(aux[0].nome, "NULL");
    aux[0].precoDia = 0;
    aux[0].tam = 0;
    aux[0].valor = 0;

    return aux;
}
```

A função vetGuiCriVetGuiNovo é chamada caso não exista guitarras criando o vetor que poderá futuramente ser alterado

```c
guitarra* criaVetGui(int contaGui) {
    guitarra *aux;
    int i;

    aux = malloc(sizeof (guitarra) * contaGui);
    if (aux == NULL) {
        puts("erro a criar vetor guitarras!");
        return NULL;
    }

    for (i = 0; i < contaGui; i++)
        aux[i].tam = contaGui;

    return aux;
}
```

A função criaVetGui cria um vetor do tipo guitarras com o tamanho de guitarras lidas e coloca o seu tamanho em todos os seus elementos

```c
void preencheGuitarra(guitarra *vetGui) {
    int id, precoDia, valor, estado, i;
    char nome[TAM];

    FILE *f;
    f = fopen("ficheiros_texto\\guitarras.txt", "rt");
    if (f == NULL) {
        printf("Erro a abrir ficheiro guitarras!");
        return;
    } else {
        for (i = 0; fscanf(f, " %d %d %d %d %[^\n]s", &id, &precoDia, &valor, &estado, nome) == 5; i++) {

            vetGui[i].id = id;
            vetGui[i].precoDia = precoDia;
            vetGui[i].valor = valor;
            vetGui[i].estado = estado;
            strcpy(vetGui[i].nome, nome);

        }
    }

    fclose(f);
    return;
}
```

A função preencheGuitarra é chamada após ter sido criado o vetor dinâmico. A função tem como objetivo preencher o vetor com as informações das guitarras no ficheiro de texto

ESTRUTURA DO PROGRAMA

Fase inicial

Clientes

Após ser criado o vetor dinâmico das guitarras o programa cria a lista dos clientes e a lista de alugueres relativamente a cada um deles.

```
cliente* iniciaCli(guitarra *gui) {
    cliente *vec = NULL;

    vec = preencheCliente();

    if (vec != NULL)
        verificaBanidos(vec, gui);

    return vec;
}
```

A função iniciaCli começa por chamar a função preencheCliente e verificar se existe o ficheiro clientes, caso este não existe cria-o e devolve um ponteiro do tipo cliente não alocando memória para este, futuramente poderá alocar ao adicionar um cliente. Caso exista clientes após ter sido criado a lista é verificado se existe clientes banidos na lista eliminando-os, caso exista, da lista.

```
cliente* preencheCliente() {
    char linha[TAM];
    int i;

    cliente *lista = NULL, *novoCli = NULL, *auxLista = NULL;
    cliente novo;

    aluguer* novoAlu = NULL, *aux = NULL;

    FILE *f;
    f = fopen("ficheiros_texto\\clientes.txt", "rt");
    if (f == NULL) {
        criaFichCli();
        return lista;
    }

    int entrou = 0;
    while (fscanf(f, "%d %d %[^\n]", &novo.nif, &novo.nAlugueres, novo.nome) == 3) {
        novo.alu = NULL;
        novo.prox = NULL;

        novoCli = malloc(sizeof (cliente));
        if (novoCli == NULL) {
            printf("Erro a alocar memoria!");
            return lista;
        }

        *novoCli = novo;

        if (entrou = 0)
            lista = novoCli;

        entrou = 1;

        if (lista == NULL)
            lista = novoCli;
        else {
            auxLista = lista;
            while (auxLista->prox != NULL)
                auxLista = auxLista->prox;
            auxLista->prox = novoCli;
        }


        aux = novoCli->alu;
```

A função preencheCliente verifica a existencia do ficheiro dos clientes e caso não existe cria-o e devolve um ponteiro do tipo cliente.

Caso exista já o ficheiro lê a primeira linha e aloca memória para poder preencher em seguida com os dados lidos do ficheiro.

O cliente é adicionado à lista por ordem, ou seja, o primeiro cliente da lista será o primeiro cliente no ficheiro de texto.

(*continuação…)*

```
        aux = novoCli->alu;

        for (i = 0; i < novoCli->nAlugueres; i++) {

            novoAlu = malloc(sizeof (aluguer));
            if (novoAlu == NULL) {
                printf("Erro a alocar memoria!");
                return lista;
            }

            novoAlu->prox = NULL;

            fscanf(f, "%d %d", &novoAlu->idGuitarra, &novoAlu->estado);

            fgets(linha, TAM, f);

            if (sscanf(linha, "%d %d %d %d %d %d", &novoAlu->dataInicio.dia, &novoAlu->dataInicio.mes, &novoAlu->dataInicio.ano,
                    &novoAlu->dataEntrega.dia, &novoAlu->dataEntrega.mes, &novoAlu->dataEntrega.ano) != 6)
                novoAlu->dataEntrega.dia = novoAlu->dataEntrega.mes = novoAlu->dataEntrega.ano = 0;

            if (novoCli->alu == NULL)
                novoCli->alu = novoAlu;
            else
                aux->prox = novoAlu;

            aux = novoAlu;


        }
    }

    fclose(f);
    return lista;
}
```

De seguida é verificado quantos alugueres tem o cliente, caso seja maior que 0 é alocado memória para criar a lista dos alugueres.

Também neste caso é colocado na lista os alugueres consuante o ficheiro de texto.

No fim do progrma é devolvido o ponteiro para o inicio do vetor cliente.

ESTRUTURA DO PROGRAMA

Fase inicial

Clientes

```c
cliente* verificaBanidos(cliente *vec, guitarra *gui) {
    FILE *f;
    cliente banido;

    f = fopen("ficheiro_binario\\banidos.dat", "rb");
    if (f == NULL) {
        return vec;
    }

    int i, entrou = 0;
    cliente* auxCli = vec;
    while (fread(&banido, sizeof (cliente), 1, f) == 1) {
        auxCli = vec;
        while (auxCli != NULL) {
            if (auxCli->nif == banido.nif) {
                if (entrou == 0)
                    puts("Existem clientes banidos no ficheiro dos clientes!\n");
                entrou = 1;
                vec = eliminaBanidos(vec, auxCli->nif, gui);
                auxCli = vec;
            }
            auxCli = auxCli->prox;
        }
    }


    fclose(f);
    if (entrou == 1)
        fim2();
    return vec;
}
```

A função verificabanido verifica se existe clientes banidos na lista, criada anteriormente, eliminando da lista os clientes que se encontram banidos

ESTRUTURA DO PROGRAMA

Fase intermédia

A fase intermédia é onde existe interação com o utiliador, pois é nesta fase que o utilizador, por exemplo, adiciona clientes ou elimina-os, cria alugueres, etc.

```c
void menu(guitarra *vGui, cliente *vCli) {
    int menu, entrou = 0;
    char term, termm[1];
    char lixo = '\0';
    do {
        if (entrou == 0) {
            printf("Bem vindo ao sistema de gerencia de \"Guitarras p'Alugar\"\n");
            entrou = 1;
        }
        puts("Menu:");
        puts("1- Guitarras");
        puts("2- Clientes");
        puts("3- Alugueres");
        puts("0- Sair");
        printf("Opcao: ");
        if (scanf(" %2d%c", &menu, &term) != 2 || term != '\n')
            menu = -1;
        else if (menu < 0 || menu > 3)
            menu = -2;

        if (menu == 1) {
            printf("\n\n");
            vGui = menuGuitarras(vGui, vCli);
            putchar('\n');
        } else if (menu == 2) {
            printf("\n\n");
            vCli = menuClientes(vCli);
            putchar('\n');
        } else if (menu == 3) {
            printf("\n\n");
            vCli = menuAlugueres(vGui, vCli);
            putchar('\n');
        } else if (menu == 0) {
            atualizaFicheiros(vGui, vCli);
            return;
        } else {
            putchar('\a');
            printf("\nOpcao errada!\n");
            if (menu == -2)
                fim2();
            if (menu == -1)
                fim();
        }
    } while (1);
}
```

Quando o cliente inicia o programa ,caso não haja clientes banidos na lista, é lhe mostrado este menu onde poderá aceder às guitarras,clientes e alugueres.

ESTRUTURA DO PROGRAMA

Fase intermédia

Guitarras

```c
guitarra * menuGuitarras(guitarra *vetor, cliente * vCliente) {
    int menu;
    char term;

    do {
        puts("Menu Guitarras\n");
        puts("Caso queira sair a qualquer momento digite 0 !");
        puts("1- Adicionar uma guitarra ao stock");
        puts("2- Historico de alugueres");
        puts("3- Listar todas as guitarras");
        puts("4- Listar guitarras alugadas");
        puts("5- Arranjar guitarras danificadas");
        puts("6- Eliminar guitarras");
        puts("0- Voltar ao menu inicial");
        printf("Opcao: ");
        if (scanf("%2d%c", &menu, &term) != 2 || term != '\n')
            menu = -1;
        else if (menu < 0 || menu > 6)
            menu = -2;

        if (menu == 1) {
            putchar('\n');
            vetor = adicionarGuitarra(vetor);
            putchar('\n');
        } else if (menu == 2) {
            putchar('\n');
            historico(vetor, vCliente);
            putchar('\n');
        } else if (menu == 3) {
            putchar('\n');
            listarGuitarras(vetor);
            putchar('\n');
        } else if (menu == 4) {
            putchar('\n');
            guitarrasAlugadas(vetor, vCliente);
            putchar('\n');
        } else if (menu == 5) {
            putchar('\n');
            arranjaGui(vetor);
            putchar('\n');
        } else if (menu == 6) {
            putchar('\n');
            vetor = eliminaGui(vetor);
            putchar('\n');
        } else if (menu == 0) {
            printf("\n\n");
            return vetor;
        } else {
            putchar('\a');
            printf("\nOpcao errada!");
            if (menu == -1)
                fim();
            if (menu == -2)
                fim2();
        }
    } while (1);
}
```

Caso o utilizador aceda ao menu guitarras será este menu que lhe será mostrado.

Neste menu o utilizador poderá aceder a 6 funções das quais poderá ser chamadas nestas mesmas, outras funções existentes

ESTRUTURA DO PROGRAMA

Fase intermédia

Guitarras

- AdicionarGuitarra

```c
guitarra* adicionarGuitarra(guitarra *v) {
    int i;
    guitarra nova;
    char term;

    puts("\n\nMenu adicionar guitarra:");

    nova.estado = 0;
    nova.tam = v[0].tam + 1;

    if (v[0].id != 0) {
        puts("\nID's em uso:");
        for (i = 0; i < v[0].tam; i++)
            printf("- %d\t%s\n", v[i].id, v[i].nome);
    }


    printf("\nIntroduza o ID da nova guitarra (maior que 0): ");
    if (scanf("%10d%c", &nova.id, &term) != 2 || term != '\n') {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        fim();
        return v;
    } else if (nova.id == 0) {
        return v;
    } else if (nova.id < 0 || nova.id > 2147483647) {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        fim2();
        return v;
    }



    for (i = 0; i < v[0].tam; i++) {
        if (v[i].id == nova.id) {
            puts("\nO id introduzido ja se encontra em uso!\n");
            fim2();
            return v;
        }
    }

    printf("\nIntroduza o preco por dia: ");
    if (scanf("%10d%c", &nova.precoDia, &term) != 2 || term != '\n') {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        fim();
        return v;
    } else if (nova.id == 0) {
        return v;
    } else if (nova.precoDia < 0 || nova.precoDia > 2147483647) {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        fim2();
        return v;
    }
```

A função adicionaGuitarras tal como o nome indica adiciona guitarras ao stock da loja.

Começa por mostrar ao utilziador as guitarras existentes na loja .

De seguida é pedido ao utilizador o id, preço por dia do aluguer e o valor da guitarra, verificando se é válido todos estes parametros.

(continuação)

```
    printf("\nIntroduza o seu valor: ");
    if (scanf("%10d%c", &nova.valor, &term) != 2 || term != '\n') {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        fim();
        return v;
    } else if (nova.valor < 0 || nova.valor > 2147483647) {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        fim2();
        return v;
    }
    if (nova.valor == 0)
        return v;

    if (nova.valor <= nova.precoDia) {
        puts("\nValor invalido o preco por dia necessita ser maior que o valor da guitarra!\n");
        fim2();
        return v;
    }

    printf("\nIntroduza o nome da nova guitarra : ");
    scanf(" %49[^\n]", nova.nome);

    for (i = 0; nova.nome[i] = toupper(nova.nome[i]); i++);

    if (v[0].tam == 0) {
        v[0] = nova;
    } else {
        guitarra *auxGui = NULL;
        auxGui = realloc(v, (v[0].tam + 1) * sizeof (guitarra));
        if (auxGui == NULL) {
            puts("Erro a alocar memoria");
            fim2();
            return v;
        }
        v = auxGui;

        for (i = 0; i < nova.tam; i++)
            v[i].tam = nova.tam;

        for (i = nova.tam - 1;; i--) {
            if (v[i - 1].id > nova.id)
                v[i] = v[i - 1];
            else {
                v[i] = nova;
                break;
            }
        }
    }

    printf("\nA guitarra %s com ID:%d foi adicionada!\n", nova.nome, nova.id);

    fim();
    return v;
}
```

Após verificar os valores ditos ateriormente é pedido o nome da nova guitarra, aceita-se guitarras com o mesmo nome precisando necessáriamente que tenham um id diferente.

É adicionada a nova guitarra ao vetor e é devolcido o ponteiro para o vetor.

ESTRUTURA DO PROGRAMA

Fase intermédia

Guitarras

- historico

```
void historico(guitarra *vGui, cliente *vCli) {
    int i, tam = vGui[0].tam, atraso;
    int meses[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

    cliente *auxCli = vCli;
    aluguer *auxAlu;
    puts("\n\nHistorico das guitarras:");

    if (vverExisteGui(vGui) == 0)
        return;


    for (i = 0; i < tam; i++) {

        printf("- %s :\n", vGui[i].nome);
        vCli = auxCli;
        while (vCli) {
            auxAlu = vCli->alu;
            while (auxAlu) {
                if ((auxAlu->idGuitarra == vGui[i].id) && (auxAlu->dataEntrega.ano != 0)) {
                    if (auxAlu->dataInicio.mes < auxAlu->dataEntrega.mes) {
                        for (i = 0; i < 12; i++)
                            if (i == auxAlu->dataInicio.mes) {
                                meses[i];
                                break;
                            }
                        atraso = meses[i] - auxAlu->dataInicio.dia + auxAlu->dataEntrega.dia;
                    } else
                        atraso = auxAlu->dataEntrega.dia - auxAlu->dataInicio.dia;

                    if (atraso > 7)
                        atraso = atraso - 7;
                    else
                        atraso = 0;

                    printf("\tCliente: %s  Datas: %d/%d/%d - %d/%d/%d  Atraso de entrega(dias): %d\n",
                            vCli->nome,
                            auxAlu->dataInicio.dia, auxAlu->dataInicio.mes, auxAlu->dataInicio.ano,
                            auxAlu->dataEntrega.dia, auxAlu->dataEntrega.mes, auxAlu->dataEntrega.ano,
                            atraso);
                }
                auxAlu = auxAlu->prox;
            }
            vCli = vCli->prox;
        }
        printf("\n");
    }

    fim2();
    return;
}
```

Nesta função é verificado o histórico de alugueres de cada guitarra e o seu atraso na entrega

ESTRUTURA DO PROGRAMA

Fase intermédia

Guitarras

- listarGuitarras

```c
void listarGuitarras(guitarra *gui) {
    puts("\n\nLista das guitarras existentes:");
    int i, tam = 0, tamMax = 0;

    if (vverExisteGui(gui) == 0)
        return;


    for (i = 0; i < gui[0].tam; i++) {
        tam = strlen(gui[i].nome);
        if (tam > tamMax)
            tamMax = tam;
    }

    tamMax += 8;

    int nrT = 0;
    char tab[50];
    int maxT = tamMax / 8;

    for (i = 0; i < gui[0].tam; i++) {
        tam = strlen(gui[i].nome) + 8;

        nrT = maxT - (tam / 8);

        strcpy(tab, "\t");
        for (int j = 0; j != nrT; j++)
            strcat(tab, "\t");

        printf("- Nome: %s%sID: %d\tEstado: %d   Preco/dia: %d\tValor: %d", gui[i].nome, tab, gui[i].id, gui[i].estado, gui[i].precoDia, gui[i].valor);

        putchar('\n');
    }

    fim2();
    return;
}
```

Nesta função listado todas as guitarras existentes

ESTRUTURA DO PROGRAMA

Fase intermédia

Guitarras

- guitarrasAlugadas

```
void guitarrasAlugadas(guitarra *vGui, cliente *vCli) {
    int i, tam = vGui[0].tam;
    cliente *auxCli = vCli;
    aluguer *auxAlu;

    puts("\n\nMenu guitarras atualmente alugadas:\n");

    if (vverExisteGui(vGui) == 0)
        return;

    for (i = 0; i < tam; i++) {

        vCli = auxCli;

        while (vCli != NULL) {
            auxAlu = vCli->alu;
            while (auxAlu != NULL) {
                if ((auxAlu->idGuitarra == vGui[i].id) && (auxAlu->dataEntrega.ano == 0)) {
                    printf("- Nome: %s   ID: %d   Preco/Dia: %d   Valor: %d   Cliente: %s   NIF: %d\n", vGui[i].nome, vGui[i].id,
                            vGui[i].precoDia, vGui[i].valor, vCli->nome, vCli->nif);
                }
                auxAlu = auxAlu->prox;
            }
            vCli = vCli->prox;
        }
    }

    fim2();
    return;
}
```

Nesta função é listadas as guitarras atualmente alugadas

ESTRUTURA DO PROGRAMA

Fase intermédia

Guitarras

- arranjaGui

```
void arranjaGui(guitarra *gui) {
    puts("\n\nMenu arranjar guitarras:");

    if (vverExisteGui(gui) == 0)
        return;

    int i, entrou = 0;
    for (i = 0; i < gui[0].tam; i++)
        if (gui[i].estado == 2)
            entrou = 1;
    if (entrou == 0) {
        puts("\nNao existem guitarras danificadas!\n");
        fim2();
        return;
    }

    puts("\nGuitarras danificadas:\n");
    for (i = 0; i < gui[0].tam; i++)
        if (gui[i].estado == 2)
            printf("- ID: %d \tNome: %s\n", gui[i].id, gui[i].nome);

    int id;
    char term;
    printf("\nIntroduza o ID da guitarra arranjada: ");
    if (scanf("%10d%c", &id, &term) != 2 || term != '\n') {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        fim();
        return;
    } else if (id < 0 || id > 2147483647) {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        fim2();
        return;
    }
    if (id == 0)
        return;

    entrou = 0;
    for (i = 0; i < gui[0].tam; i++)
        if ((gui[i].estado == 2)&&(gui[i].id == id)) {
            entrou = 1;
            gui[i].estado = 0;
            break;
        }
    if (entrou == 0) {
        printf("\a");
        puts("O ID introduzido nao pertence a nenhuma guitarra danificada!\n");
        fim2();
        return;
    }

    printf("\nA guitarra %s foi arranjada!\n", gui[i].nome);

    fim2();
    return;
}
```

Nesta função é arranjada uma guitarra que esteja danificada

ESTRUTURA DO PROGRAMA

Fase intermédia

Clientes

```c
cliente* menuClientes(cliente *v) {
    int menu;
    char term;

    do {
        puts("Menu Clientes:\n");
        puts("Caso queira sair a qualquer momento digite 0 !");
        puts("1- Adicionar cliente");
        puts("2- Remover cliente");
        puts("3- Mostrar estados");
        puts("4- Listar clientes ativos");
        puts("5- Listar clientes banidos");
        puts("0- Voltar ao menu inicial");
        printf("Opcao: ");
        if (scanf("%2d%c", &menu, &term) != 2 || term != '\n')
            menu = -1;
        else if (menu < 0 || menu > 5)
            menu = -2;

        if (menu == 1) {
            putchar('\n');
            v = adicionarClientes(v);
        } else if (menu == 2) {
            putchar('\n');
            v = eliminarCliente(v);
            putchar('\n');
        } else if (menu == 3) {
            putchar('\n');
            mostrarEstdos(v);
            putchar('\n');
        } else if (menu == 4) {
            putchar('\n');
            listarClientes(v);
            putchar('\n');
        } else if (menu == 5) {
            putchar('\n');
            mostrarBanidos();
            putchar('\n');
        } else if (menu == 0) {
            printf("\n\n");
            return v;
        } else {
            putchar('\a');
            printf("\nOpcao errada!");
            if (menu == -2)
                fim2();
            if (menu == -1)
                fim();
        }
    } while (1);
}
```

Caso o utilizador aceda ao menu clientes será este menu que lhe será mostrado.

Neste menu o utilizador poderá aceder a 5 funções das quais poderá ser chamadas nestas mesmas, outras funções existentes

ESTRUTURA DO PROGRAMA

Fase intermédia

Clientes

- adicionarClientes

```
cliente* adicionarClientes(cliente *lista) {
    cliente novo, *aux = NULL, *auxNovo = NULL;
    puts("\n\nMenu adicionar clientes:");

    aux = lista;
    char term;

    if (lista != NULL) {
        puts("NIF's em uso:");
        correLista(lista);
    }

    printf("\nIntroduza o NIF: ");
    if (scanf("%10d%c", &novo.nif, &term) != 2 || term != '\n') {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        fim();
        return lista;
    } else if (novo.nif == 0) {
        return lista;
    } else if (novo.nif < 100000000 || novo.nif > 9999999999) {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        fim2();
        return lista;
    }

    FILE *f;
    cliente banido;
    f = fopen("ficheiro_binario\\banidos.dat", "rb");
    int binario = 0;
    if (f == NULL) {
        binario = 1;
    } else if (binario == 0)
        while (fread(&banido, sizeof (cliente), 1, f) == 1)
            if (banido.nif == novo.nif) {
                puts("\nO cliente introduzido foi banido, nao podera ser adicionado!\n");
                fim2();
                return lista;
            }
    fclose(f);

    while (aux != NULL) {
        if (aux->nif == novo.nif) {
            puts("\nJa existe um cliente com o nif introduzido!\n");
            fim2();
            return lista;
        }
        aux = aux->prox;
    }

    auxNovo = malloc(sizeof (cliente));
    if (auxNovo == NULL) {
        printf("\nErro a alocar memoria!\n");
        fim2();
        return lista;
    }

    printf("\nIntroduza o nome do novo cliente: ");
    scanf(" %49[^\n]", novo.nome);
```

Esta função como o próprio nome indica adiciona clientes à lista.

É pedido ao utilizador o NIF do novo cliente, sendo este único e com 9 digitos.

É verificfado se o cliente que pretende adicionar já foi banido caso sim é negada a adição do cliente.

De seguida é pedido o nome, sendo que este é permitido ser repetido por outro cliente

(continuação)

```
    int i;
    int x = strlen(novo.nome);
    for (i = 0; i < x; i++) {
        if ((isalpha(novo.nome[i]) && novo.nome[i - 1] == ' ') || i == 0) {
            novo.nome[i] = toupper(novo.nome[i]);
        }
    }

    *auxNovo = novo;

    auxNovo->alu = NULL;
    auxNovo->prox = NULL;
    auxNovo->nAlugueres = 0;

    cliente *inicioLista = lista, *auxLista = NULL;

    if (inicioLista == NULL)
        lista = auxNovo;
    else {
        auxLista = inicioLista;
        while (auxLista->prox != NULL)
            auxLista = auxLista->prox;
        auxLista->prox = auxNovo;
    }

    printf("\nO cliente %s com NIF %d foi adicionado!\n", novo.nome, novo.nif);

    fim();
    return lista;
}
```

Após serem preenchidos e verificados todos os requezitos é adicionado o cliente ao ultimo elementro da lista. Finalizando a função com o retomar do ponteiro para a lista

ESTRUTURA DO PROGRAMA

Fase intermédia

Clientes

- eliminarCliente

```c
cliente* eliminarCliente(cliente *lista) {
    cliente *atual, *anterior = NULL;
    int nif;
    atual = lista;
    char term;

    puts("Menu eliminar cliente:\n");

    if (verExisteCli(lista) == 0)
        return lista;


    puts("Clientes ativos:");
    correLista(lista);

    printf("\nIntroduza o NIF do cliente a eliminar: ");
    if (scanf("%10d%c", &nif, &term) != 2 || term != '\n') {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        return lista;
    } else if (nif == 0) {
        return lista;
    } else if (nif < 100000000 || nif > 9999999999) {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        return lista;
    }


    cliente *correrLista = lista;
    while (correrLista != NULL) {
        if (correrLista->nif == nif)
            break;
        correrLista = correrLista->prox;
    }
    if (correrLista == NULL) {
        puts("\nNenhum cliente foi encontrado com o NIF introduzido!");
        fim2();
        return lista;
    }

    while (atual != NULL && (atual->nif != nif)) {
        anterior = atual;
        atual = atual->prox;
    }
    if (anterior == NULL)
        lista = atual->prox;
    else
        anterior->prox = atual->prox;

    printf("\n- O cliente %s foi eliminado!\n", atual->nome);

    free(atual);

    fim2();
    return lista;
}
```

Esta função como o próprio nome indica elimina clientes do vetor.

No inicio da função é mostrado todos os clientes átivos e de seguida é pedido o NIF do cliente a eliminar.

É verificado se o NIF corresponde aos clientes ativos e é de seguida eliminado o cliente da lista.

A função devolve um ponteiro ara o inicio da lista

ESTRUTURA DO PROGRAMA

Fase intermédia

Clientes

- mostraEstados

```c
void mostrarEstdos(cliente *lista) {
    puts("\nMenu mostar estado:\n");

    if (verExisteCli(lista) == 0)
        return;

    int nif;
    char term;

    correLista(lista);

    printf("\nIntroduza NIF do cliente que pretende consultar: ");
    if (scanf("%i0d%c", &nif, &term) != 2 || term != '\n') {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        putchar('\n');
        return;
    } else if (nif == 0) {
        return;
    } else if (nif < 100000000 || nif > 9999999999) {
        printf("\a");
        puts("\nNIF invalido!\n");
        putchar('\n');
        return;
    }

    cliente *cli = lista;
    while (cli != NULL) {
        if (cli->nif == nif)
            break;

        cli = cli->prox;
    }
    if (cli == NULL) {
        puts("\nNenhum cliente foi encontrado com o NIF introduzido!");
        fim();
        return;
    }

    int contaGui = 0, contaDani = 0, atraso, contaAtraso = 0, contaAlu;
    aluguer *auxAlu = cli->alu;
    while (auxAlu != NULL) {
        if (auxAlu->estado == 0)
            contaGui++;

        if (auxAlu->estado == 2)
            contaDani++;

        if (auxAlu->dataEntrega.ano != 0) {
            atraso = calculaAtraso(auxAlu->dataInicio, auxAlu->dataEntrega);
            if (atraso > 7)
                contaAtraso++;
        }

        auxAlu = auxAlu->prox;
    }

    contaAlu = cli->nAlugueres - contaGui;
    printf("\n- %s:\n\n", cli->nome);
    printf("- Guitarras que detem: %d\n- Nr de alugueres: %d\n- Nr de entregas com atraso: %d\n- Nr de guitarras danificadas: %d\n", contaGui, contaAlu, contaAtraso, contaDani);

    fim2();
    return;
}
```

A função mostrarEstados apresenta o estado de um cliente introduzido, isto é, quantas quitarras detem consigo, quantos alugueres já concluiu com sucesso, quantos alugueres já concluiu com atraso e o número de guitarras danificadas

ESTRUTURA DO PROGRAMA

Fase intermédia

Clientes

- listarClientes

```c
void listarClientes(cliente *lista) {
    puts("\nLista das guitarras existentes:\n");
    int i, tam = 0, tamMax = 0;

    if (verExisteCli(lista) == 0)
        return;


    cliente *cli = lista;
    while (cli != NULL) {
        tam = strlen(cli->nome);
        if (tam > tamMax)
            tamMax = tam;
        cli = cli->prox;
    }

    tamMax += 8;

    int nrT = 0;
    char tab[50];
    int maxT = tamMax / 8;

    cli = lista;
    while (cli != NULL) {
        tam = strlen(cli->nome) + 8;

        nrT = maxT - (tam / 8);

        strcpy(tab, "\t");
        for (int j = 0; j != nrT; j++)
            strcat(tab, "\t");

        printf("- Nome: %s%sNIF: %d\n", cli->nome, tab, cli->nif);

        putchar('\n');
        cli = cli->prox;
    }

    fim2();
    return;
}
```

A função lista todos os clientes ativos

ESTRUTURA DO PROGRAMA

Fase intermédia

Clientes

- mostrarBanidos

```c
void mostrarBanidos() {
    puts("\nMenu clientes banidos:");
    FILE *f;
    cliente banido;
    f = fopen("ficheiro_binario\\banidos.dat", "rb");
    if (f == NULL) {
        puts("\nNao existe clientes Banidos!\n");
        fim2();
        return;
    }

    while (fread(&banido, sizeof (cliente), 1, f) == 1) {
        if (banido.nAlugueres == 1)
            printf("\n- NIF: %d\tNome:%s\tMotivo: Guitarras danificadas\n", banido.nif, banido.nome);
        if (banido.nAlugueres == 0)
            printf("\n- NIF: %d\tNome:%s\tMotivo: Atraso na entrega\n", banido.nif, banido.nome);
    }

    fclose(f);

    fim2();
    return;
}
```

A função lista todos os clientes banidos através do ficheiro binário “banidos”

ESTRUTURA DO PROGRAMA

Fase intermédia

Clientes

Funções auxiliares

- calculaAtraso

```c
int calculaAtraso(data inicio, data fim) {
    int tempoAlu = 0;
    data auxData;
    verificaBissexto(fim.ano);
    if ((inicio.mes != fim.mes) || (inicio.ano != fim.ano)) {
        auxData = inicio;
        while (1) {
            if (auxData.ano == fim.ano && auxData.mes == fim.mes && auxData.dia == fim.dia)
                return tempoAlu;
            else if (auxData.dia > meses[(auxData.mes) - 1] - 1) {
                auxData.dia = 1;
                auxData.mes += 1;
            } else if (auxData.mes > 12) {
                auxData.mes = 1;
                auxData.ano += 1;
            } else
                auxData.dia += 1;

            tempoAlu++;
        }
    } else
        tempoAlu = fim.dia - inicio.dia;

    if (tempoAlu == 0)
        tempoAlu = 1;

    return tempoAlu;
}

void correLista(cliente *lista) {
    cliente *correrLista = lista;
    while (correrLista != NULL) {
        printf("- NIF: %d\tNome: %s\n", correrLista->nif, correrLista->nome);
        correrLista = correrLista->prox;
    }
}
```

Esta função apesar de não ser acessivel ao utilizador é utilizada para calcular quantas vezes um cliente entregou guitarras atrasadas

ESTRUTURA DO PROGRAMA

Fase intermédia

Alugueres

```
cliente* menuAlugueres(guitarra* vGui, cliente* vCli) {
    int menu, auxMaxAluguer = 5, auxValorGuitarraBarata = 500, auxNrGuiBarata = 6;
    cliente* novoV = NULL;
    novoV = vCli;
    char term;

    valorGuitarraBarata = auxValorGuitarraBarata;
    maxAluguer = auxMaxAluguer;
    nrGuibarata = auxNrGuiBarata;

    dataAtual = dataHoje(vCli);
    if (dataAtual == NULL) {
        return vCli;
    }

    vCli = banimentoAuto(vCli, dataAtual, vGui);

    do {
        printf("Menu Alugueres\n");
        puts("1- Criar aluguer");
        puts("2- Concluir um aluguer");
        puts("3- Lista de todos os alugueres");
        puts("4- Lista de alugueres a decorrer");
        puts("5- Alterar max. alugueres");
        puts("6- Alterar o valor considerado por guitarras baratas");
        puts("7- Alterar o nr minimo de guitarras baratas a alugar");
        puts("0- Voltar ao menu inicial");
        printf("Opcao: ");
        if (scanf("%2d%c", &menu, &term) != 2 || term != '\n')
            menu = -1;
        else if (menu < 0 || menu > 7)
            menu = -2;

        if (menu == 1) {
            putchar('\n');
            vCli = criaAluguer(vGui, vCli);
            putchar('\n');
        } else if (menu == 2) {
            putchar('\n');
            vCli = concluiAluguer(vGui, vCli);
            putchar('\n');
        } else if (menu == 3) {
            putchar('\n');
            listarTodosAlugueres(vCli);
            putchar('\n');
        } else if (menu == 4) {
            putchar('\n');
            listarAlugueresAtuais(vCli);
            putchar('\n');
        } else if (menu == 5) {
            putchar('\n');
            maxAluguer = alterarMaxAlu(maxAluguer);
            putchar('\n');
        } else if (menu == 6) {
            putchar('\n');
            valorGuitarraBarata = guitarrasbaratas(valorGuitarraBarata);
            putchar('\n');
        } else if (menu == 7) {
            putchar('\n');
            nrGuibarata = nrGuitarrasbaratas(nrGuibarata);
            putchar('\n');
        } else if (menu == 0) {
            printf("\n\n");
            return vCli;
        } else {
            putchar('\a');
            printf("\nOpcao errada!");
            if (menu == -2)
                fim2();
            if (menu == -1)
                fim();
        }
    } while (1);
}
```

Caso o utilizador aceda ao menu alugueres será este menu que lhe será mostrado.

Neste menu o utilizador poderá aceder a 7 funções das quais poderá ser chamadas nestas mesmas, outras funções existentes

ESTRUTURA DO PROGRAMA

Fase intermédia

Alugueres

- criaAluguer

```c
cliente* criaAluguer(guitarra* gui, cliente* cli) {
    int nif;
    int entrou = 0, i;
    char term;

    puts("\nMenu criar Aluguer:");

    if (VerExisteCli(cli) == 0)
        return cli;

    if (gui[0].id == 0) {
        puts("\nNao existe guitarras!");
        fim2();
        return cli;
    }

    cliente* correLista = cli;
    if (correLista == NULL) {
        puts("\nNao existem clientes para efetuar aluguer!");
        fim2();
        return cli;
    }

    puts("\nClientes:");

    while (correLista != NULL) {
        printf("- NIF: %d\tNome: %s\n", correLista->nif, correLista->nome);
        correLista = correLista->prox;
    }

    printf("\n- Introduza o NIF do cliente que tenciona um novo aluguer: ");
    if (scanf("%10d%c", &nif, &term) != 2 || term != '\n') {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        fim();
        return cli;
    } else if (nif == 0) {
        return cli;
    } else if (nif < 100000000 || nif > 9999999999) {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        fim2();
        return cli;
    }

    cliente *aux = cli;
    while (aux != NULL) {
        if (aux->nif == nif)
            break;
        aux = aux->prox;
    }

    if (aux == NULL) {
        printf("\a");
        puts("\nO cliente introduzido nao existe !\n");
        fim2();
        return cli;
    }
```

A função criaAluguer , cria um aluguer para um determinado cliente, esta função adiciona um elemento à lista de alugueres e também adiciona um aluguer ao cliente

(continuação)

```
aluguer *auxAluguer = NULL;
int contaAluguer = 0;
auxAluguer = aux->alu;
while (auxAluguer != NULL) {
    if ((auxAluguer->dataInicio.dia != 0)&&(auxAluguer->dataEntrega.dia == 0))
        contaAluguer++;
    auxAluguer = auxAluguer->prox;
}

if (contaAluguer >= maxAluguer) {
    printf("\nO cliente %s tem atualmente %d guitarras alugadas sendo %d o limite maximo permitido !\n\n", aux->nome, contaAluguer, maxAluguer);
    fim2();
    return cli;
}

puts("\nGuitarras disponiveis:");
for (i = 0; i < gui[0].tam; i++)
    if (gui[i].estado == 0)
        printf("- ID:%d\t%s\n", gui[i].id, gui[i].nome);


int ID;
printf("\nIntroduza o ID da guitarra a alugar: ");
if (scanf("%10d%c", &ID, &term) != 2 || term != '\n') {
    printf("\a");
    puts("\nNumero introduzido invalido!\n");
    fim();
    return cli;
} else if (ID == 0) {
    return NULL;
} else if (ID < 0 || ID > 2147483647) {
    printf("\a");
    puts("\nNumero introduzido invalido!\n");
    fim2();
    return cli;
}
entrou = 0;
for (i = 0; i < gui[0].tam; i++)
    if (gui[i].id == ID) {
        entrou = 1;
        break;
    }
if ((entrou != 1) || (gui[i].estado == 2)) {
    printf("\a");
    puts("\nO Id introduzido nao existe!\n");
    fim2();
    return cli;
}
if (gui[i].estado == 1) {
    puts("\nA guitarra introduzida encontra-se alugada!\n");
    fim2();
    return cli;
}
```

É pedido ao utilizador o nif do cliente que pretende fazer um aluguer e o id da guitarra a alugar.

É verificado se tanto o id da guitarra como o nif do cliente se são válidos

(continuação)

```
    auxAluguer = aux->alu;
    contaAluguer = 0;
    while (auxAluguer != NULL) {
        if ((auxAluguer->dataInicio.dia != 0)&&(auxAluguer->dataEntrega.dia != 0))
            contaAluguer++;
        auxAluguer = auxAluguer->prox;
    }

    if ((gui[i].valor > valorGuitarraBarata) && (contaAluguer < nrGuibarata)) {
        printf("\nO cliente nao pode alugar a guitarra, o seu valor e %d euros, necessita de ter pelo menos "
               "%d guitarras baratas alugadas(menos que %d euros)!\n\n",
               gui[i].valor, nrGuibarata, valorGuitarraBarata);
        fim2();
        return cli;
    }


    data dataEntrega = dataPrevistaEntrega(dataAtual);


    printf("\nA guitarra deve ser devolvida ate a seguinte data: %d/%d/%d\n", dataEntrega.dia, dataEntrega.mes, dataEntrega.ano);
    printf("Caso a guitarra seja entregue no 7 Dia tera um custo de: %d euros\n\n", gui[i].precoDia * 7);

    aluguer *novoAlu = NULL;
    novoAlu = malloc(sizeof (aluguer));
    if (novoAlu == NULL) {
        printf("Erro a alocar memoria!");
        fim2();
        return cli;
    }

    novoAlu->prox = NULL;
    novoAlu->estado = 1;
    novoAlu->idGuitarra = ID;
    novoAlu->dataInicio = *dataAtual;
    novoAlu->dataEntrega.dia = 0;
    novoAlu->dataEntrega.mes = 0;
    novoAlu->dataEntrega.ano = 0;
    aux->nAlugueres += 1;

    aux->alu = novoAluguer(aux->alu, novoAlu);

    gui[i].estado = 1;

    fim2();
    return cli;
}
```

É verificado se o cliente poderá efetuar um novo aluguer, é definico inicialmente que o numero máximo de guitarras alugadas seria 5.

É verificado se o cliente já alugou um certo número de guitarras baratas (definido inicialmente 500 euros) caso alugue uma guitarrra cara.

É mostrado ao cliente o preço máximo do aluguer,caso não se atrase e a data que máxima que deverá ser entregue.

É adicionado ao ultimo nó dos alugueres o novo aluguer criado.

A função no final devolve um o ponteiro para o inicio da lista

ESTRUTURA DO PROGRAMA

Fase intermédia

Alugueres

- concluiAluguer

```
cliente* concluiAluguer(guitarra* gui, cliente* cli) {
    int i, entrou;
    char term;

    puts("\nMenu concluir Aluguer:");

    if (VerExisteCli(cli) == 0)
        return cli;

    cliente* correLista = cli;
    aluguer *correAlu = correLista->alu;
    while (correLista != NULL) {
        while (correAlu != NULL) {
            if (correAlu->dataEntrega.ano == 0)
                break;
            correAlu = correAlu->prox;
        }
        if (correAlu->dataEntrega.ano == 0)
            break;
        correLista = correLista->prox;
    }
    if (correLista == NULL) {
        puts("\nNao existem alugueres por concluir!");
        fim();
        return cli;
    }
    puts("\nGuitarras alugadas:");
    for (i = 0; i < gui[0].tam; i++)
        if (gui[i].estado == 1)
            printf("- NIF: %d\tNome: %s\n", gui[i].id, gui[i].nome);

    int ID;
    printf("\nIntroduza o ID da guitarra a concluir o aluguer: ");
    if (scanf("%10d%c", &ID, &term) != 2 || term != '\n') {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        fim();
        return cli;
    } else if (ID == 0) {
        return cli;
    } else if (ID < 0 || ID > 2147483647) {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        fim();
        return cli;
    }
    entrou = 0;
    for (i = 0; i < gui[0].tam; i++)
        if ((gui[i].id == ID)&&(gui[i].estado == 1)) {
            entrou = 1;
            break;
        }
    if (entrou != 1) {
        printf("\a");
        puts("\nA guitarra introduzida nao se encontra alugada!\n");
        fim();
        return cli;

    } else if (gui[i].estado == 0 || gui[i].estado == 2) {
        puts("\nO Id introduzido nao existe!\n");
        fim();
        return cli;
    }
```

A função concluiAluguer, conclui um aluguer que esteja a decorrer.

Começa por mostrar as guitarras atualmente alugadas e pede ao utilizador que introduza o id da guitarra a concluir aluguer. É verificado se o id introduzido é válido

(continuação)

```
cliente* auxCli = cli;
aluguer* auxAlu = NULL, *aluAnterior = NULL;
entrou = 0;
while (auxCli != NULL) {
    auxAlu = auxCli->alu;
    while (auxAlu != NULL) {
        if ((auxAlu->idGuitarra == ID)&&(auxAlu->dataEntrega.ano == 0)) {
            entrou = 1;
            break;
        }
        aluAnterior = auxAlu;
        auxAlu = auxAlu->prox;
    }
    if (entrou == 1)
        break;
    auxCli = auxCli->prox;
}

int menuDanificada, danificada = 0;

puts("\nA guitarra encontra-se danificada?");
puts("1 - Sim");
puts("2 - Nao");
printf("Escolha: ");
if (scanf("%2d%c", &menuDanificada, &term) != 2 || term != '\n') {
    printf("\a");
    puts("\nNumero introduzido invalido!\n");
    fim();
    return cli;
} else if (menuDanificada == 0) {
    return cli;
} else if (menuDanificada < 0 || menuDanificada > 2) {
    printf("\a");
    puts("\nNumero introduzido invalido!\n");
    fim2();
    return cli;
}
if (menuDanificada == 1) {
    danificada = 1;
    aluguer* auxAlu2 = auxCli->alu;
    entrou = 0;
    gui[i].estado = 2;
    //CALCULA QUANTAS GUITARRAS DANIFICADAS O CLIENTE JA DANIFICOU
    while (auxAlu2 != NULL) {
        if (auxAlu2->estado == 2)
            entrou += 1;
        auxAlu2 = auxAlu2->prox;
    }

    if (entrou == 3) {
        printf("\n- O cliente %s foi banido pois ja danificou mais de 3 guitarras\n", auxCli->nome);
        auxCli->nAlugueres = 1;
        cli = banir(cli, auxCli);
        fim2();
        return cli;
    }
    printf("\nO cliente tem de pagar um multa por dano no valor de: %d euros\n", gui[i].valor);
} else if (menuDanificada != 2) {
    printf("\a");
    puts("\nResposta Incorreta!\n");
    fim2();
    return cli;
}
```

De seguida é verificado se a guitarra encontra-se danificada, caso esteja danificada é verificado se já foram danificadas mais de 3 guitarras, se sim o cliente é banido, se não danificou mais de 3 guitarras terá de pagar uma multa no valor da guitarra

(continuação)

```
    int tempoAlu, tempoAtraso;

    tempoAtraso = tempoAlu = calculaTempo(auxAlu, dataAtual);

    if (tempoAtraso > 7) {
        tempoAtraso -= 7;
        int multa = 10;
        printf("\nO cliente tem de pagar uma multa no valor de %d euros por ter entrego com %d dias de atraso\n", tempoAtraso * multa, tempoAtraso);
    }

    int custo;
    custo = tempoAlu * gui[i].precoDia;
    printf("\nCusto do aluguer: %d\n", custo);

    auxAlu->dataEntrega = *dataAtual;

    if (danificada == 1) {
        auxAlu->estado = 2;
    } else {
        gui[i].estado = 0;
        auxAlu->estado = 1;
    }

    fim2();
    return cli;
}
```

Em seguida é calculado o tempo de atraso, caso haja um atraso se este for maior que 7 dias será automaticamente banido, caso haja atraso e seja menor que 7 dias será multado em 10 euros por cada dia de atraso. É calculado o preço do aluguer e é definido a data de entrega do aluguer. É devolvido um ponteiro para o inicio da lista clientes

ESTRUTURA DO PROGRAMA

Fase intermédia

Alugueres

- listarTodosAlugueres

```c
void listarTodosAlugueres(cliente* cli) {
    cliente* aux = cli;
    aluguer* auxAluguer = NULL;

    puts("\nMenu listar alugueres atuais e concluidos:\n");

    if (VerExisteCli(cli) == 0)
        return;

    while (aux != NULL) {
        printf("- %s:\n", aux->nome);

        auxAluguer = aux->alu;
        while (auxAluguer != NULL) {
            if (auxAluguer->dataEntrega.ano != 0)
                printf("- DataInicio: %02d/%02d/%1d\tDataEntrega: %02d/%02d/%02d\n", auxAluguer->dataInicio.dia, auxAluguer->dataInicio.mes, auxAluguer->dataInicio.ano,
                    auxAluguer->dataEntrega.dia, auxAluguer->dataEntrega.mes, auxAluguer->dataEntrega.ano);
            else
                printf("- DataInicio: %02d/%02d/%02d\n", auxAluguer->dataInicio.dia, auxAluguer->dataInicio.mes, auxAluguer->dataInicio.ano);
            auxAluguer = auxAluguer->prox;
        }
        printf("\n");
        aux = aux->prox;
    }

    fim2();
    return;
}
```

Nesta função é listado todos os alugueres, tanto os que estão a decorrer como os que estão por concluir

ESTRUTURA DO PROGRAMA

Fase intermédia

Alugueres

- listarAlugueresAtuais

```
void listarAlugueresAtuais(cliente* cli) {
    cliente* aux = cli;
    aluguer* auxAluguer = NULL;
    data dataPrevista, *dataAluguer;
    int atraso;

    puts("\nMenu alugueres a decorrer:\n");

    if (VerExisteCli(cli) == 0)
        return;

    while (aux != NULL) {
        printf("- %s:\n", aux->nome);
        auxAluguer = aux->ali;
        while (auxAluguer != NULL) {
            if (auxAluguer->dataEntrega.ano == 0) {

                dataPrevista = dataPrevistaEntrega(&auxAluguer->dataInicio);
                atraso = calculaTempo(auxAluguer, dataAtual);
                if (atraso > 7)
                    atraso -= 7;
                else
                    atraso = 0;
                printf("- ID: %d\tInicio: %02d/%02d/%1d\tEntrega prevista: %02d/%02d/%02d\tAtraso: %d\n", auxAluguer->idGuitarra, auxAluguer->dataInicio.dia, auxAluguer->dataInicio.mes, auxAluguer->dataInicio.ano,
                        dataPrevista.dia, dataPrevista.mes, dataPrevista.ano, atraso);
            }
            auxAluguer = auxAluguer->prox;
        }
        printf("\n");
        aux = aux->prox;
    }

    fim2();
    return;
}
```

Nesta função é listado os alugueres que estão a decorrer, é mostrado para cada cliente as guitarras que alugou, a data inicio e a data prevista de entrega, ainda é mostrado tambem, caso haja, os dias de atraso

ESTRUTURA DO PROGRAMA

Fase intermédia

Alugueres

- alterarMaxAlu

```c
int alterarMaxAlu(int maxAluguer) {
    int max;
    char term;

    puts("\nMenu alterar maximo alugueres:\n");

    printf("Introduza o numero max de alugueres: ");
    if (scanf("%10d%c", &max, &term) != 2 || term != '\n') {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        fim();
        return maxAluguer;
    } else if (max < 1 || max > 2147483647) {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        fim();
        return maxAluguer;
    } else
        printf("\nNovo numero maximo de alugueres: %d\n", max);

    fim2();
    return max;
}
```

Esta função altera o numero máximo de alugueres que cada cliente poderá fazer. É devolvido o numero máximo de alugueres.

ESTRUTURA DO PROGRAMA

Fase intermédia

Alugueres

- guitarrasbaratas

```
int guitarrasbaratas(int valorGuitarraBarata) {
    int max;
    char term;

    puts("\nMenu alterar valor guitarras baratas:\n");

    printf("Introduza o valor a considerar por guitarras baratas: ");
    if (scanf("%10d%c", &max, &term) != 2 || term != '\n') {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        fim();
        return valorGuitarraBarata;
    } else if (max < 1 || max > 2147483647) {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        fim();
        return valorGuitarraBarata;
    } else
        printf("\nNovo valor guitarras baratas: %d\n", max);

    fim2();
    return max;
}
```

Esta função altera o valor a ser considerado por guitarrasbaratas. É devolvido o valor de guitarras baratas

ESTRUTURA DO PROGRAMA

Fase intermédia

Alugueres

- nrGuitarrasbaratas

```c
int nrGuitarrasbaratas(int nrGuibarata) {
    int max;
    char term;

    puts("\nMenu alterar nr de guitarras baratas necessarias para aluguer de uma cara:\n");

    printf("Introduza o valor a considerar por guitarras baratas: ");
    if (scanf("%10d%c", &max, &term) != 2 || term != '\n') {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        fim();
        return nrGuibarata;
    } else if (max < 0 || max > 2147483647) {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        fim();
        return nrGuibarata;
    } else
        printf("\nNovo numero de guitarras baratas necessarias para aluguer de uma cara: %d\n", max);

    fim2();
    return max;
}
```

Esta função altera o numero necessário de guitarras baratas a alugar para poder alugar uma guitarra cara

ESTRUTURA DO PROGRAMA

Fase intermédia

Alugueres

Funções auxiliares

- dataHoje

```
data* dataHoje(cliente* vCli) {

    cliente* auxCli = vCli;
    aluguer* auxAluguer;
    int dia, mes, ano;
    char term;
    data* dataHoje;

    puts("Introduza a data atual:");

    printf("Dia: ");
    if (scanf("%3d%c", &dia, &term) != 2 || term != '\n') {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        fim2();
        return NULL;
    } else if (dia == 0) {
        return NULL;
    } else if (dia < 1 || dia > 2147483647) {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        fim2();
        return NULL;
    }
    printf("Mes: ");
    if (scanf("%3d%c", &mes, &term) != 2 || term != '\n') {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        fim2();
        return NULL;
    } else if (mes == 0) {
        return NULL;
    } else if (mes < 1 || mes > 12) {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        fim2();
        return NULL;
    }
    printf("Ano (Ex. 2010): ");
    if (scanf("%5d%c", &ano, &term) != 2 || term != '\n') {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        fim2();
        return NULL;
    } else if (ano == 0) {
        return NULL;
    } else if (ano < 0 || ano > 2147483647) {
        printf("\a");
        puts("\nNumero introduzido invalido!\n");
        fim2();
        return NULL;
    }

    verificaBissexto(ano);

    if (dia > meses[mes - 1]) {
        printf("\nData introduzida invalida !\n\n");
        fim2();
        return NULL;
    }
    while (auxCli != NULL) {
        auxAluguer = auxCli->alu;
        while (auxAluguer != NULL) {
            if (auxAluguer->dataEntrega.dia == 0) {
                if (auxAluguer->dataInicio.ano > ano) {
                    printf("\a");
                    printf("\nData introduzida invalida !\n\n");
                    fim2();
                    return NULL;
                } else if ((auxAluguer->dataInicio.mes > mes)&&(auxAluguer->dataInicio.ano == ano)) {
                    printf("\a");
                    printf("\nData introduzida invalida !\n\n");
                    fim2();
                    return NULL;
                } else if ((auxAluguer->dataInicio.dia > dia)&&(auxAluguer->dataInicio.mes == mes)&&(auxAluguer->dataInicio.ano == ano)) {
                    printf("\a");
                    printf("\nData introduzida invalida !\n\n");
                    fim2();
                    return NULL;
                }
            } else {
                if (auxAluguer->dataEntrega.ano > ano) {
                    printf("\a");
                    printf("\nData introduzida invalida !\n\n");
                    fim2();
                    return NULL;
                } else if ((auxAluguer->dataEntrega.mes > mes)&&(auxAluguer->dataEntrega.ano == ano)) {
                    printf("\a");
                    printf("\nData introduzida invalida !\n\n");
                    fim2();
                    return NULL;
                } else if ((auxAluguer->dataEntrega.dia > dia)&&(auxAluguer->dataEntrega.mes == mes)&&(auxAluguer->dataEntrega.ano == ano)) {
                    printf("\a");
                    printf("\nData introduzida invalida !\n\n");
                    fim2();
                    return NULL;
                }
            }
            auxAluguer = auxAluguer->prox;
        }
        auxCli = auxCli->prox;
    }

    dataHoje = malloc(sizeof (data));
    if (dataHoje == NULL) {
        puts("Erro a alocar memoria ! ");
        fim2();
        return NULL;
    }
    dataHoje->ano = ano;
    dataHoje->mes = mes;
    dataHoje->dia = dia;


    return dataHoje;
}
```

A função dataHoje é chamada no inicio da execução do menu alugueres.

Tem como propósito definir a data do dia da utilização do programa para poder ser futuramente utilizada em diferentes funções com diferentes objetivos

ESTRUTURA DO PROGRAMA

Fase intermédia

Alugueres

Funções auxiliares

- banir

```c
cliente* banir(cliente* lista, cliente* cli) {
    FILE *f;
    cliente banir = *cli;
    int i;

    f = fopen("ficheiro_binario\\banidos.dat", "ab");
    if (f == NULL) {
        puts("\nErro a abrir ficheiro binario!\n");
        return lista;
    }

    if ((fwrite(&banir, sizeof (cliente), 1, f)) != 1) {
        puts("\nErro a escrever no ficheiro binario!\n");
        return lista;
    }

    fclose(f);

    cliente *atual, *anterior = NULL;
    int nif;
    atual = lista;

    nif = cli->nif;

    while (atual != NULL && (atual->nif != nif)) {
        anterior = atual;
        atual = atual->prox;
    }
    if (atual == NULL) {
        puts("Nao existe nenhum cliente com o nome introduzido!");
        return lista;
    }
    if (anterior == NULL)
        lista = atual->prox;
    else
        anterior->prox = atual->prox;
    free(atual);

    return lista;
}
```

A função banir elimina um cliente da lista ligada e coloca-o no ficheiro binário "banidos".

É atualizada a lista ainda dentro desta função

A função retoma a lista atualizada

ESTRUTURA DO PROGRAMA

Fase intermédia

Alugueres

Funções auxiliares

- banimentoAutomático

```c
// SE O N ALUGUERES É 0 FOI BANIMENTO POR TEMPO SE FOR 1 É BANIMENTO POR DANIFICAR GUITARRAS
cliente* banimentoAuto(cliente* cli, data* dataAtual, guitarra* gui) {
    cliente* auxCli = NULL;
    aluguer* auxAlu = NULL;
    int i, entrou, totalAtraso, atraso;

    auxCli = cli;
    while (auxCli != NULL) {
        totalAtraso = 0;
        atraso = 0;
        entrou = 0;
        auxAlu = auxCli->alu;
        while (auxAlu != NULL) {
            if (auxAlu->dataEntrega.ano == 0) {
                atraso = calculaTempo(auxAlu, dataAtual);
                if (atraso > 7) {
                    atraso -= 7;
                    totalAtraso += atraso;
                    if (totalAtraso > 20) {
                        for (i = 0; i < gui[0].tam; i++)
                            if ((gui[i].id == auxAlu->idGuitarra)&&(gui[i].estado == 1))
                                gui[i].estado = 0;

                        putchar('\n');
                        printf("- O cliente %s foi banido!\n", auxCli->nome);
                        auxCli->nAlugueres = 0;
                        cli = banir(cli, auxCli);
                        auxCli = cli;
                        entrou = 1;
                        break;
                    }
                }
            }
            auxAlu = auxAlu->prox;
        }
        if (entrou == 0)
            auxCli = auxCli->prox;
    }

    char term;
    printf("\nPrima qualquer tecla: ");
    scanf("%c", &term);
    putchar('\n');

    return cli;
}
```

A função banimentoAuto é chamada logo após ser introduzida a data do dia. Calculando automáticamente os utilizadores que não entregaram as guitarras alugadas dentro dos 20 dias. A função retoma a lista atualizada

ESTRUTURA DO PROGRAMA

Fase intermédia

Alugueres

Funções auxiliares

- novoAluguer

```
aluguer* novoAluguer(aluguer *inicio, aluguer* novo) {
    aluguer* auxLista = NULL;
    aluguer* inicioLista = inicio;

    if (inicioLista == NULL)
        inicio = novo;
    else {
        auxLista = inicioLista;
        while (auxLista->prox != NULL)
            auxLista = auxLista->prox;
        auxLista->prox = novo;
    }
    return inicio;
}
```

A função novoAluguer é chamada durante a criação de aluguer adicionando um nó à lista de alugueres e retomando o ponteiro para o inicio da lista

ESTRUTURA DO PROGRAMA

Fase Final

A fase final consiste na atualização dos ficheiros de texto. Atualizando com os novos elementos.

atualizaFicheiros

A função atualizaFicheiros é chamada quando no menu principal o utilizador escolhe “sair”.

A função contem outras 2 funções, atualizaGuitarras e atualizaClientes.

Começa por atualizar o ficheiro guitarras e em seguida o ficheiro clientes.

```
void atualizaFicheiros(guitarra *gui, cliente * cli) {
    atualizaGuitarras(gui);
    atualizaClientes(cli);
}
```

ESTRUTURA DO PROGRAMA

ESTRUTURA DO PROGRAMA

Fase Final

atualizaGuitarras

```
void atualizaGuitarras(guitarra *gui) {
    int i, j;
    FILE *f;
    f = fopen("ficheiros_texto\\guitarras.txt", "wt");
    if (f == NULL) {
        printf("Erro a abrir ficheiro guitarras!");
        return;
    } else {
        for (i = 0; i < gui[0].tam; i++) {
            fprintf(f, "%d\t%d\t%d\t%d\t%s", gui[i].id, gui[i].precoDia, gui[i].valor, gui[i].estado, gui[i].nome);
            if (i != ((gui[i].tam) - 1)) {
                fprintf(f, "\n");
            }
        }
    }
    fclose(f);
    return;
}
```

A função abre o ficheiro guitarras e em seguida imprime a informações contidas no vetor das guitarras.

atualizaClientes

```
void atualizaClientes(cliente *cli) {
    FILE *f;
    f = fopen("ficheiros_texto\\clientes.txt", "wt");
    if (f == NULL) {
        printf("Erro a abrir ficheiro Clientes!");
        return;
    }
    int entrou = 0;
    cliente* auxCli = cli;
    aluguer* auxAluguer = NULL;
    while (auxCli != NULL) {
        fprintf(f, "%d\t%d\t%s", auxCli->nif, auxCli->nAlugueres, auxCli->nome);
        auxAluguer = auxCli->alu;
        entrou = 0;

        while (auxAluguer != NULL) {
            if (entrou == 0) {
                fprintf(f, "\n");
            }
            entrou = 1;
            if (auxAluguer->dataEntrega.dia == 0) {
                fprintf(f, "%d   %d\t%02d  %02d  %02d", auxAluguer->idGuitarra, auxAluguer->estado,
                        auxAluguer->dataInicio.dia, auxAluguer->dataInicio.mes, auxAluguer->dataInicio.ano);

            } else {
                fprintf(f, "%d   %d\t%02d  %02d  %02d\t%02d  %02d  %02d", auxAluguer->idGuitarra, auxAluguer->estado,
                        auxAluguer->dataInicio.dia, auxAluguer->dataInicio.mes, auxAluguer->dataInicio.ano,
                        auxAluguer->dataEntrega.dia, auxAluguer->dataEntrega.mes, auxAluguer->dataEntrega.ano);
            }
            if (auxAluguer->prox != NULL)
                fprintf(f, "\n");

            auxAluguer = auxAluguer->prox;
        }

        if (auxCli->prox != NULL)
            fprintf(f, "\n\n");

        auxCli = auxCli->prox;
    }
    putchar('\n');
    fclose(f);
    return;
}
```

A função abre o ficheiro guitarras e em seguida imprime a informação de um cliente. Após ser imprimido a informação do cliente é imprimido todos os seus alugueres

MANUAL DE UTILIZAÇÃO

O programa destina-se a utilizadores que possua uma loja de guitarras/alugue guitarras. O programa necessita de ser executado numa plataforma eletrónica, preferencialmente um computador e é necessário um dispositivos perifericos de preferencialmente um teclado.

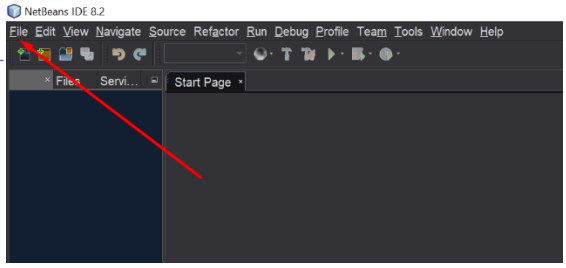
Antes de correr o programa deve-se verificar, caso seja a primeira vez que executa o programa, se as as pastas ficheiros_texto e ficheiro_binario se encontram vazias, caso contrário a pasta ficheiro_texto deverá ter 2 ficheiros de texto, clientes e guitarras, e a pasta ficheiro_binário deverá apenas conter o ficheiro banidos.

MANUAL DE UTILIZAÇÃO

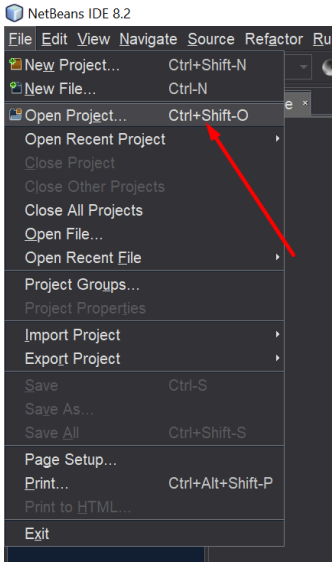
Inicializar o programa

O programa deverá ser iniciado no IDE “NetBeans” da seguinte forma:

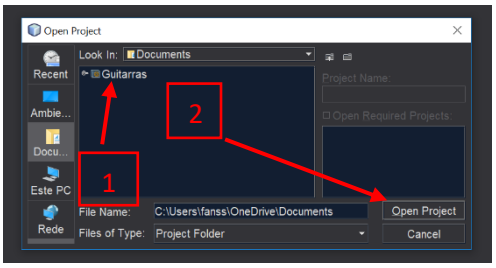
(1) Abrir o IDE



(2) Abrir o menu FILE

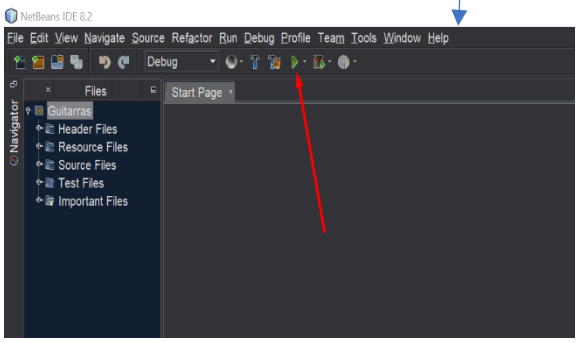


(3) Clicar em Open Project

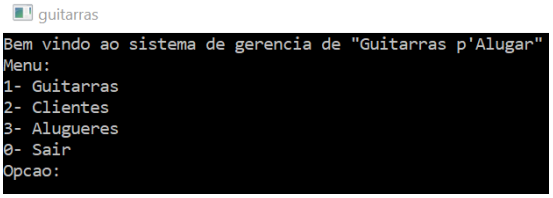


(4) Ir à diretoria onde se encontra o programa selecionar o progrma e clicar em Open Project

(5) Executar o projecto

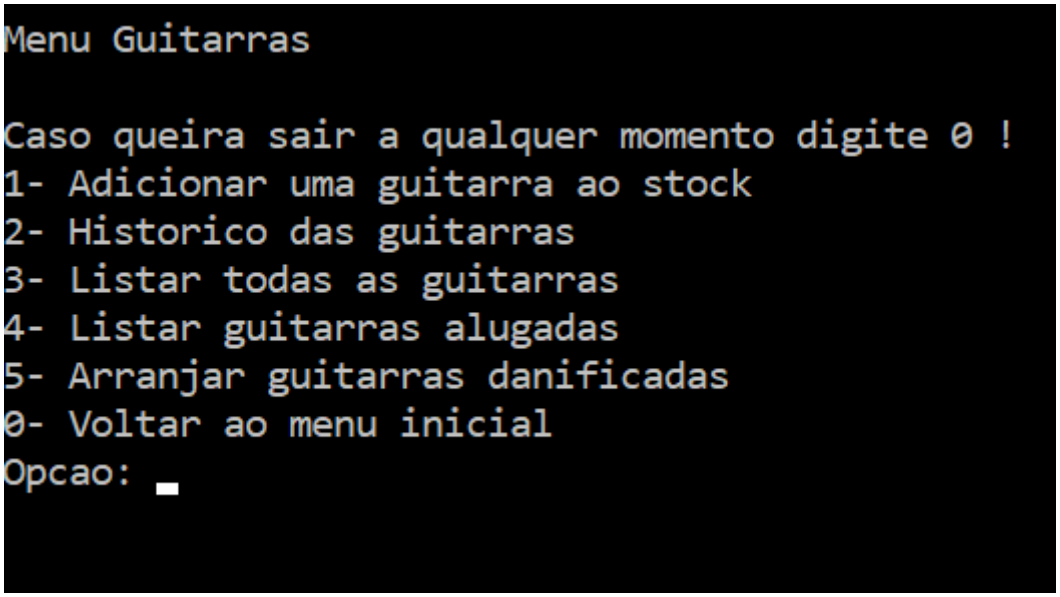


(6) Será aberto o seguinte menu. Deverá escolher uma opção que pretende aceder digitando um numero de 0 a 3. Ao longo do programa sempre que necessite sair de uma função digite 0



MANUAL DE UTILIZAÇÃO

Menu Guitarras



Caso escolha o menu guitarras será mostrado as seguintes opções que poderá aceder. Deverá escolher uma opção digitando um numero de 0 a 6. Para voltar ao menu principal digite 0

MANUAL DE UTILIZAÇÃO

Menu Guitarras

Função adicionar guitarras

```
Menu adicionar guitarra:

ID's em uso:
- 1      LUCILLE
- 4      BETTY JEAN
- 5      ROCKY

Introduza o ID da nova guitarra (maior que 0): 2

Introduza o preco por dia: 50

Introduza o seu valor: 500

Introduza o nome da nova guitarra : caroll

A guitarra CAROLL com ID:2 foi adicionada!


Prima qualquer tecla:
```

Para aceder à função adicionar guitarras digite 1 no menu guitarras

Deverá primeiramente por escolher o id da guitarra que pretende adicionar inteiro positivo diferente dos que já estão em uso.

Em seguita introduza um valor inteiro maior que zero do preço diário do aluguer da guitarra.

Em seguida digite o valor da guitarra, necessita ser maior que o preço diário do aluguer.

Finalize a adição da guitarra digitando o nome que pretende dar à mesma

MANUAL DE UTILIZAÇÃO

Menu Guitarras

Função historico guitarras

```
Historico das guitarras:
- LUCILLE :

- CAROLL :

- BETTY JEAN :
        Cliente: Marta Nunes Cabral  Datas: 12/12/2017 - 17/12/2017  Atraso de entrega(dias): 0

- ROCKY :
        Cliente: Maria Pimentel  Datas: 1/10/2017 - 10/10/2017  Atraso de entrega(dias): 2
        Cliente: Maria Pimentel  Datas: 13/1/2018 - 15/1/2018  Atraso de entrega(dias): 0


Prima qualquer tecla:
```

Para aceder à função historico das guitarras digite 2 no menu guitarras

Irá ser mostrado o historico de cada guitarra em seus respetivos alugueres.

MANUAL DE UTILIZAÇÃO

Menu Guitarras

Função listar guitarras

```
Historico das guitarras:
- LUCILLE :

- CAROLL :

- BETTY JEAN :
        Cliente: Marta Nunes Cabral  Datas: 12/12/2017 - 17/12/2017  Atraso de entrega(dias): 0

- ROCKY :
        Cliente: Maria Pimentel  Datas: 1/10/2017 - 10/10/2017  Atraso de entrega(dias): 2
        Cliente: Maria Pimentel  Datas: 13/1/2018 - 15/1/2018  Atraso de entrega(dias): 0


Prima qualquer tecla:
```

Para aceder à função historico das guitarras digite 3 no menu guitarras

Esta função irá listar todas as guitarras existentes e suas respetivas carateristicas

MANUAL DE UTILIZAÇÃO

Menu Guitarras

Função guitarras atualmente alugadas

```
Menu Guitarras

Caso queira sair a qualquer momento digite 0 !
1- Adicionar uma guitarra ao stock
2- Historico de alugueres
3- Listar todas as guitarras
4- Listar guitarras alugadas
5- Arranjar guitarras danificadas
6- Eliminar guitarras
0- Voltar ao menu inicial
Opcao: 4



Menu guitarras atualmente alugadas:

- Nome: BETTY JEAN   ID: 4   Preco/Dia: 100   Valor: 500   Cliente: Maria Pimentel   NIF: 123123123
- Nome: ROCKY   ID: 5   Preco/Dia: 25   Valor: 350   Cliente: Marta Nunes Cabral   NIF: 555666777


Prima qualquer tecla:
```

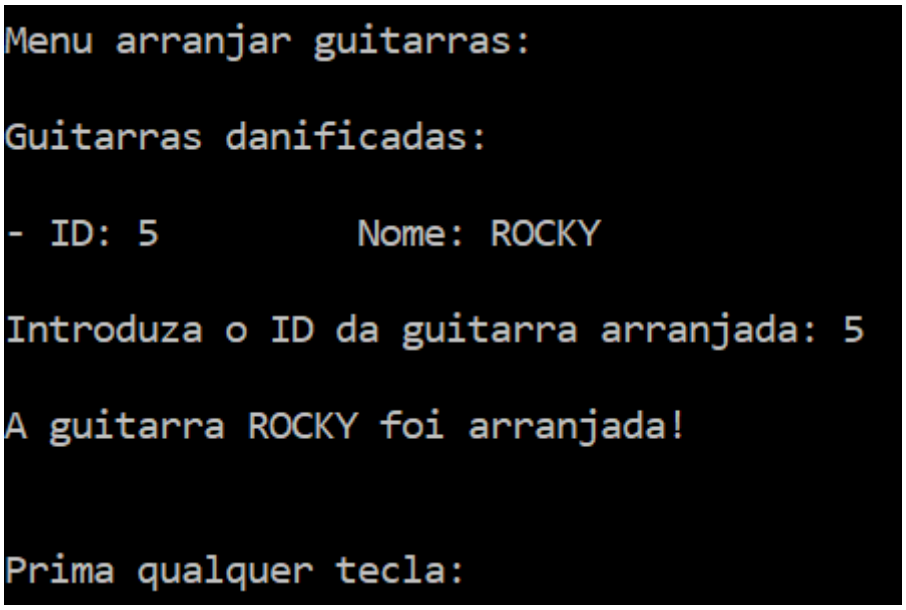
Para aceder à função guitarras atualmente alugadas digite 4 no menu guitarras

Esta função irá listar todas as guitarras que se encontram alugadas, o seu id, preço por dia, o nome e nif do cliente que a alugou

MANUAL DE UTILIZAÇÃO

Menu Guitarras

Função arranjar guitarras danificadas



Para aceder à função arranjar guitarras alugadas digite 5 no menu guitarras

Esta função deferá ser usada após uma guitarra que estivesse préviamente danificada se encontre agora arranjada.

Deverá apenas indicar o id da guitarra que foi reparada

MANUAL DE UTILIZAÇÃO

Menu Clientes

```
Menu Clientes:

Caso queira sair a qualquer momento digite 0 !
1- Adicionar cliente
2- Remover cliente
3- Mostrar estados
4- Listar clientes ativos
5- Listar clientes banidos
0- Voltar ao menu inicial
Opcao:
```

Caso escolha o menu clientes será mostrado as seguintes opções que poderá aceder. Deverá escolher uma opção digitando um numero de 0 a 5. Para voltar ao menu principal digite 0

MANUAL DE UTILIZAÇÃO

Menu Clientes

Função adicionar clientes

```
Menu adicionar clientes:
NIF's em uso:
- NIF: 123456789        Nome: Paulo Silva

Introduza o NIF: 111222333

Introduza o nome do novo cliente: filipe a. n. silva

O cliente Filipe A. N. Silva com NIF 111222333 foi adicionado!


Prima qualquer tecla:
```

Para aceder à função adicionar clientes digite 1 no menu clientes.

Esta função irá adicionar um cliente, deverá indicar um nif único e o nome do novo cliente até 49 caracteres

MANUAL DE UTILIZAÇÃO

Menu Clientes

Função eliminar clientes

```
Menu eliminar cliente:

Clientes ativos:
- NIF: 123456789        Nome: Paulo Silva
- NIF: 111222333        Nome: Filipe A. N. Silva

Introduza o NIF do cliente a eliminar: 111222333

- O cliente Filipe A. N. Silva foi eliminado!


Prima qualquer tecla:
```

Para aceder à função eliminar clientes digite 2 no menu clientes.

Esta função irá eliminar um cliente, deverá apenas indicar o nif do utilizador

MANUAL DE UTILIZAÇÃO

Menu Clientes

Função mostrar estado

```
Menu mostar estado:

- NIF: 123456789        Nome: Paulo Silva

Introduza NIF do cliente que pretende consultar: 123456789

- Paulo Silva:

- Guitarras que detem: 0
- Nr de alugueres: 1
- Nr de entregas com atraso: 0
- Nr de guitarras danificadas: 1


Prima qualquer tecla:
```

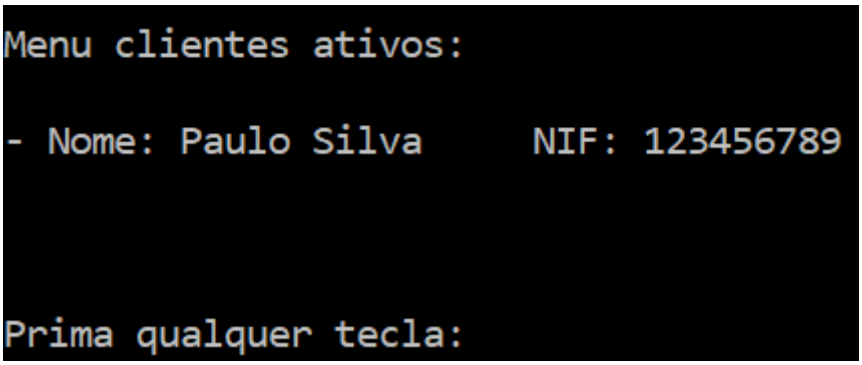
Para aceder à função mostrar estado digite 3 no menu clientes.

Esta função irá mostrar um estado de um determinado cliente. Deverá indicar o nif do utilizador e será mostrado quantas guitarras tem atualmente alugadas, nr total de alugueres, numero de entregas em atraso, e numero de guitarras danificadas

MANUAL DE UTILIZAÇÃO

Menu Clientes

Função listar clientes ativos



Para aceder à função listar clientes ativos digite 4 no menu clientes.

Esta função irá mostrar o nome e o NIF de todos os clientes ativos

MANUAL DE UTILIZAÇÃO

Menu Clientes

Função listar clientes banidos

```
Menu clientes banidos:

- NIF: 555666777        Nome:Marta Nunes Cabral Motivo: Atraso na entrega

- NIF: 123123123        Nome:Maria Pimentel     Motivo: Atraso na entrega


Prima qualquer tecla:
```

Para aceder à função listar clientes banidos digite 5 no menu clientes.

Esta função irá mostrar o nif, nome e a razão do banimento de cada cliente

MANUAL DE UTILIZAÇÃO

Menu Alugueres

```
Introduza a data atual:
Dia: 1
Mes: 6
Ano (Ex. 2010): 2018

Prima qualquer tecla:
```

Caso escolha o menu aluguer será lhe pedido que digite a data de hoje primeiramente

```
Menu Alugueres
1- Criar aluguer
2- Concluir um aluguer
3- Lista de todos os alugueres
4- Lista de alugueres a decorrer
5- Alterar max. alugueres
6- Alterar o valor considerado por guitarras baratas
7- Alterar o nr minimo de guitarras baratas a alugar
0- Voltar ao menu inicial
Opcao:
```

Após digitar a data de hoje será mostrado as seguintes opções que poderá aceder. Deverá escolher uma opção digitando um numero de 0 a 7. Para voltar ao menu principal digite 0

MANUAL DE UTILIZAÇÃO

Menu Alugueres

Função criar aluguer

```
Menu criar Aluguer:

Clientes:
- NIF: 123456789        Nome: Paulo Silva

- Introduza o NIF do cliente que tenciona um novo aluguer: 123456789

Guitarras disponiveis:
- ID:1  LUCILLE
- ID:4  BETTY JEAN
- ID:5  ROCKY

Introduza o ID da guitarra a alugar: 5

A guitarra deve ser devolvida ate a seguinte data: 8/6/2018
Caso a guitarra seja entregue no 7 Dia tera um custo de: 175 euros



Prima qualquer tecla:
```

Para aceder à função criar aluguer digite 1 no menu aluguer.

A função irá pedir que digite o nif do cliente que pretende alugar uma guitarra e a guitarra que pretende alugar, será mostrado a data máxima para devolver e o custo máximo do aluguer

MANUAL DE UTILIZAÇÃO

Menu Alugueres

Função concluir aluguer

```
Menu concluir Aluguer:

Guitarras alugadas:
- ID: 5 Nome: ROCKY

Introduza o ID da guitarra a concluir o aluguer: 5

A guitarra encontra-se danificada?
1 - Sim
2 - Nao
Escolha: 1

O cliente tem de pagar um multa por dano no valor de: 350 euros

Custo do aluguer: 25


Prima qualquer tecla:
```

Para aceder à função concluir aluguer digite 2 no menu aluguer.

A função irá pedir que digite o id da guitarra a concluir o aluguer, questionado sobre o estado da guitarra, caso não esteja danificada irá ser impresso o total de aluguer, caso esteja danificada irá ser cobrado um multa pelo dano no valor da guitarra, por cada dia de atraso na entregue é uma multa de 10 euros. Caso acumule 20 dias de atraso em todos os seus alugueres será banido

MANUAL DE UTILIZAÇÃO

Menu Alugueres

Função listar todos os alugueres

```
Menu listar alugueres atuais e concluidos:

- Paulo Silva:
- DataInicio: 01/06/2018        DataEntrega: 01/06/2018
- DataInicio: 01/06/2018        DataEntrega: 01/06/2018



Prima qualquer tecla:
```

Para aceder à função listar todos os alugueres digite 3 no menu aluguer.

A função irá listar todos os alugueres que já foram concluidos e que estão a decorrer

MANUAL DE UTILIZAÇÃO

Menu Alugueres

Função alugueres a decorrer

```
Menu alugueres a decorrer:

- Paulo Silva:
- ID: 4 Inicio: 01/06/2018      Entrega prevista: 08/06/2018    Atraso: 0



Prima qualquer tecla:
```

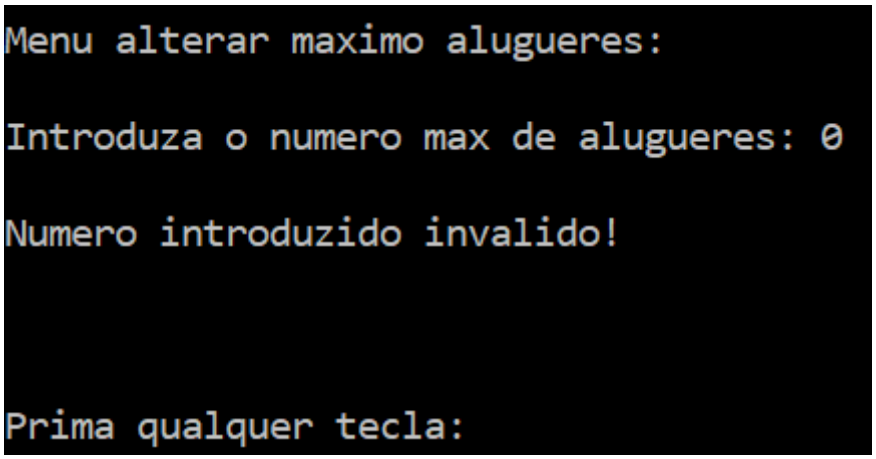
Para aceder à função listar todos os alugueres digite 4 no menu aluguer.

A função irá listar todos os alugueres estão a decorrer mostrando o id da guitarra, a data de inicio e a data prevista para entrega e o numero de dias em atraso

MANUAL DE UTILIZAÇÃO

Menu Alugueres

Função alterar numero máximo de alugueres



Para aceder à função alterar numero máximo de alugueres digite 5 no menu aluguer.

A função permite alterar o numero máximo de alugueres que um cliente pode fazer ao mesmo tempo.

MANUAL DE UTILIZAÇÃO

Menu Alugueres

Função alterar o valor a considerar por guitarras baratas

```
Menu alterar valor guitarras baratas:

Introduza o valor a considerar por guitarras baratas: 1200

Novo valor guitarras baratas: 1200


Prima qualquer tecla:
```

Para aceder à alterar o valor a considerar por guitarras baratas digite 6 no menu aluguer.

A função permite alterar o valor a considerar por guitarras baratas

MANUAL DE UTILIZAÇÃO

Menu Alugueres

Função alterar o numero minimo de guitarras baratas a alugar

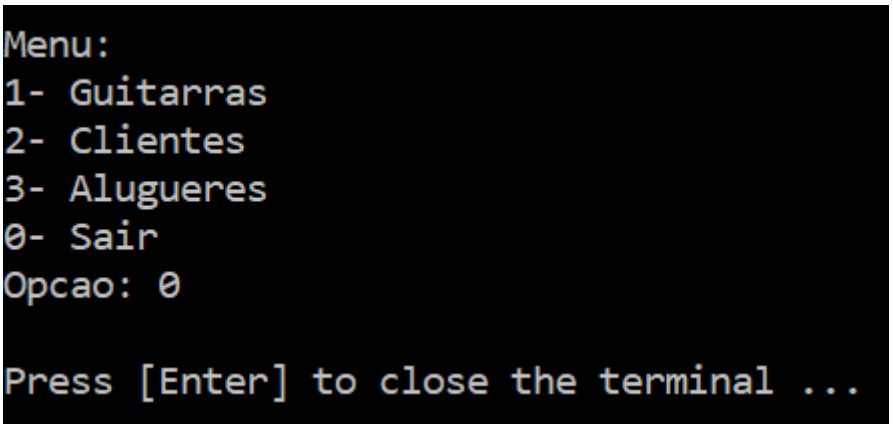
```
Menu alterar nr de guitarras baratas necessarias para aluguer de uma cara:

Introduza o valor a considerar por guitarras baratas: 1

Novo numero de guitarras baratas necessarias para aluguer de uma cara: 1


Prima qualquer tecla:
```

Para aceder à alterar o numero minimo de guitarras baratas a alugar digite 7 no menu aluguer.

A função permite alterar o numero minimo de guitarras baratas a alugar antes de poder alugar uma cara

MANUAL DE UTILIZAÇÃO

Finalizar o programa



Para finalizar o programa e atualizar todas as alterações feitas volte ao menu principal/inicial e digite 0. Irá terminar o programa e atualizar todos os ficheiros

CONCLUSÃO

Este trabalho consiste em geral na programação de uma aplicação para gerir o negócio de aluguer de guitarras. O trabalho baseia-se essencialmente na manipulação de memória dinâmica e ficheiros.

Com este trabalho consegui desenvolver e melhorar significamente a minha habilidade e conhecimento de programação.

Inicialmente tinha imensas dificuldades com ponteiros e certos aspectos do programa que não percebia bem mas ao longo da realização do trabalho foram feitas imensas pesquisas e perguntas aos professores para o esclarecimento de dúvidas que eventualmente iam aparecendo.

Concluio assim que foi-me imensamente útil a realização deste trabalho pois desenvolvi bastante as minhas qualidades programativas. Investi exaustivamente imenso tempo e dedicação neste trabalho para o poder levár à perfeição e valeu o esforço pois estou consciente que melhorei bastante.