

LI d-36

Generated by Doxygen 1.9.1

1 Data Structure Index	1
1.1 Data Structures	1
2 File Index	3
2.1 File List	3
3 Data Structure Documentation	5
3.1 _alimentos Struct Reference	5
3.1.1 Field Documentation	5
3.1.1.1 alimento	5
3.1.1.2 cal	5
3.1.1.3 data	6
3.1.1.4 id_cliente	6
3.1.1.5 next	6
3.1.1.6 refeicao	6
3.2 _cliente Struct Reference	6
3.2.1 Field Documentation	6
3.2.1.1 next	7
3.2.1.2 nome	7
3.2.1.3 num_cliente	7
3.2.1.4 telemovel	7
3.3 _data Struct Reference	7
3.3.1 Field Documentation	7
3.3.1.1 ano	7
3.3.1.2 dia	8
3.3.1.3 mes	8
3.4 _plano Struct Reference	8
3.4.1 Field Documentation	8
3.4.1.1 calmax	8
3.4.1.2 calmin	8
3.4.1.3 data_fim	9
3.4.1.4 data_inicio	9
3.4.1.5 id_cliente	9
3.4.1.6 next	9
3.4.1.7 refeicao	9
4 File Documentation	11
4.1 LESI_PI_TP_22531_23006_23131/Alimentos.c File Reference	11
4.2 LESI_PI_TP_22531_23006_23131/Alimentos.h File Reference	11
4.2.1 Function Documentation	12
4.2.1.1 addAlimento()	12
4.2.1.2 adicionarAlimentos()	12
4.2.1.3 dataEstaEntre()	12

4.2.1.4 guardarBaseDeDadosAlimentos()	12
4.2.1.5 introduzirAlimento()	13
4.2.1.6 lerBaseDeDadosAlimentos()	13
4.2.1.7 libertarMemoriaAlimentos()	13
4.2.1.8 NumPacientesPassamCals()	14
4.2.1.9 PacientesComRefeicaoForadeCals()	14
4.3 LESI_PI_TP_22531_23006_23131/Cliente.c File Reference	14
4.3.1 Function Documentation	15
4.3.1.1 adicionar()	15
4.3.1.2 adicionarCliente()	16
4.3.1.3 editar()	16
4.3.1.4 editarCliente()	17
4.3.1.5 encontrarCliente()	17
4.3.1.6 existeCliente()	17
4.3.1.7 guardarBaseDeDados()	18
4.3.1.8 guardarBinariocl()	18
4.3.1.9 guardarNormalcl()	18
4.3.1.10 guardarTabuladorescl()	19
4.3.1.11 lerBaseDeDados()	19
4.3.1.12 libertarMemoria()	19
4.3.1.13 obterIdAleatorio()	19
4.3.1.14 remover()	20
4.3.1.15 removerCliente()	20
4.3.1.16 visualizarCliente()	20
4.4 LESI_PI_TP_22531_23006_23131/Cliente.h File Reference	21
4.4.1 Function Documentation	21
4.4.1.1 adicionar()	21
4.4.1.2 adicionarCliente()	22
4.4.1.3 editar()	22
4.4.1.4 editarCliente()	23
4.4.1.5 encontrarCliente()	23
4.4.1.6 existeCliente()	23
4.4.1.7 guardarBaseDeDados()	24
4.4.1.8 lerBaseDeDados()	24
4.4.1.9 libertarMemoria()	24
4.4.1.10 obterIdAleatorio()	25
4.4.1.11 remover()	25
4.4.1.12 removerCliente()	25
4.4.1.13 visualizarCliente()	26
4.5 LESI_PI_TP_22531_23006_23131/ListaLigada.c File Reference	26
4.5.1 Function Documentation	26
4.5.1.1 menuPrincipal()	26

4.6 LESI_PI_TP_22531_23006_23131/ListaLigada.h File Reference	27
4.6.1 Typedef Documentation	27
4.6.1.1 Alimentos	27
4.6.1.2 Cliente	27
4.6.1.3 Data	28
4.6.1.4 Plano	28
4.6.2 Function Documentation	28
4.6.2.1 menuPrincipal()	28
4.7 LESI_PI_TP_22531_23006_23131/main.c File Reference	28
4.7.1 Function Documentation	29
4.7.1.1 displayHelp()	29
4.7.1.2 executeBinCommand()	29
4.7.1.3 executeTabCommand()	29
4.7.1.4 main()	30
4.8 LESI_PI_TP_22531_23006_23131/Plano.c File Reference	30
4.8.1 Function Documentation	31
4.8.1.1 addPlano()	31
4.8.1.2 adicionarPlano()	31
4.8.1.3 guardarBaseDeDadosPlano()	32
4.8.1.4 guardarBinario()	32
4.8.1.5 guardarNormal()	32
4.8.1.6 guardarTabuladores()	32
4.8.1.7 introduzirPlano()	33
4.8.1.8 lerBaseDeDadosPlano()	33
4.8.1.9 libertarMemoriaPlano()	33
4.8.1.10 ListarPlanosPorCliente()	34
4.8.1.11 MediaCalPorCliente()	34
4.8.1.12 TabelaDados()	34
4.9 LESI_PI_TP_22531_23006_23131/Plano.h File Reference	35
4.9.1 Function Documentation	35
4.9.1.1 addPlano()	35
4.9.1.2 adicionarPlano()	35
4.9.1.3 guardarBaseDeDadosPlano()	36
4.9.1.4 introduzirPlano()	36
4.9.1.5 lerBaseDeDadosPlano()	37
4.9.1.6 libertarMemoriaPlano()	37
4.9.1.7 ListarPlanosPorCliente()	37
4.9.1.8 MediaCalPorCliente()	37
4.9.1.9 TabelaDados()	38

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

_alimentos	5
_cliente	6
_data	7
_plano	8

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

LESI_PI_TP_22531_23006_23131/ Alimentos.c	11
LESI_PI_TP_22531_23006_23131/ Alimentos.h	11
LESI_PI_TP_22531_23006_23131/ Cliente.c	14
LESI_PI_TP_22531_23006_23131/ Cliente.h	21
LESI_PI_TP_22531_23006_23131/ ListaLigada.c	26
LESI_PI_TP_22531_23006_23131/ ListaLigada.h	27
LESI_PI_TP_22531_23006_23131/ main.c	28
LESI_PI_TP_22531_23006_23131/ Plano.c	30
LESI_PI_TP_22531_23006_23131/ Plano.h	35

Chapter 3

Data Structure Documentation

3.1 `_alimentos` Struct Reference

```
#include <ListaLigada.h>
```

Collaboration diagram for `_alimentos`:

Data Fields

- int `id_cliente`
- Data `data`
- char `refeicao` [20]
- char `alimento` [20]
- int `cal`
- struct `_alimentos` * `next`

3.1.1 Field Documentation

3.1.1.1 `alimento`

```
char _alimentos::alimento[20]
```

3.1.1.2 `cal`

```
int _alimentos::cal
```

3.1.1.3 data

```
Data _alimentos::data
```

3.1.1.4 id_cliente

```
int _alimentos::id_cliente
```

3.1.1.5 next

```
struct _alimentos* _alimentos::next
```

3.1.1.6 refeicao

```
char _alimentos::refeicao[20]
```

The documentation for this struct was generated from the following file:

- LESI_PI_TP_22531_23006_23131/[ListaLigada.h](#)

3.2 _cliente Struct Reference

```
#include <ListaLigada.h>
```

Collaboration diagram for _cliente:

Data Fields

- int [num_cliente](#)
- char [nome](#) [20]
- int [telemovel](#)
- struct [_cliente](#) * [next](#)

3.2.1 Field Documentation

3.2.1.1 next

```
struct _cliente* _cliente::next
```

3.2.1.2 nome

```
char _cliente::nome[20]
```

3.2.1.3 num_cliente

```
int _cliente::num_cliente
```

3.2.1.4 telemovel

```
int _cliente::telemovel
```

The documentation for this struct was generated from the following file:

- LESI_PI_TP_22531_23006_23131/[ListaLigada.h](#)

3.3 _data Struct Reference

```
#include <ListaLigada.h>
```

Data Fields

- int [dia](#)
- int [mes](#)
- int [ano](#)

3.3.1 Field Documentation

3.3.1.1 ano

```
int _data::ano
```

3.3.1.2 dia

```
int _data::dia
```

3.3.1.3 mes

```
int _data::mes
```

The documentation for this struct was generated from the following file:

- LESI_PI_TP_22531_23006_23131/[ListaLigada.h](#)

3.4 _plano Struct Reference

```
#include <ListaLigada.h>
```

Collaboration diagram for _plano:

Data Fields

- int [id_cliente](#)
- [Data](#) [data_inicio](#)
- [Data](#) [data_fim](#)
- char [refeicao](#) [20]
- int [calmin](#)
- int [calmax](#)
- struct [_plano](#) * [next](#)

3.4.1 Field Documentation

3.4.1.1 calmax

```
int _plano::calmax
```

3.4.1.2 calmin

```
int _plano::calmin
```

3.4.1.3 data_fim

```
Data _plano::data_fim
```

3.4.1.4 data_inicio

```
Data _plano::data_inicio
```

3.4.1.5 id_cliente

```
int _plano::id_cliente
```

3.4.1.6 next

```
struct _plano* _plano::next
```

3.4.1.7 refeicao

```
char _plano::refeicao[20]
```

The documentation for this struct was generated from the following file:

- LESI_PI_TP_22531_23006_23131/[ListaLigada.h](#)

Chapter 4

File Documentation

4.1 LESI_PI_TP_22531_23006_23131/Alimentos.c File Reference

```
#include "ListaLigada.h"
#include "Alimentos.h"
#include "Cliente.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <time.h>
```

Include dependency graph for Alimentos.c:

4.2 LESI_PI_TP_22531_23006_23131/Alimentos.h File Reference

This graph shows which files directly or indirectly include this file:

Functions

- [Alimentos * lerBaseDeDadosAlimentos \(\)](#)
Função que lê o txt de alimentos ao iniciar o programa e guarda numa lista ligada.
- void [guardarBaseDeDadosAlimentos \(Alimentos *\)](#)
Função que chama as 3 funções de guardar ficheiros.
- void [addAlimento \(Cliente *, Alimentos *\)](#)
Função que pede os dados para adicionar um Alimento.
- [Alimentos * adicionarAlimentos \(Alimentos *, Alimentos *\)](#)
Função para adicionar um alimento ao ler um ficheiro txt.
- [Alimentos * introduzirAlimento \(Alimentos *, Alimentos *\)](#)
Função para adicionar um cliente.
- void [libertarMemoriaAlimentos \(Alimentos *\)](#)
Liberta a memoria alocada pela lista & Uso de recursao.
- void [NumPacientesPassamCals \(Alimentos *, Cliente *\)](#)
Função que vê quantos Pacientes passaram o numero de calorias introduzidas.
- void [PacientesComRefeicaoForadeCals \(Alimentos *, Cliente *, Plano *\)](#)
Função que Lista todos os Pacientes fora de Calorias em uma certa refeição, por ordem decrescente.
- int [dataEstaEntre \(\)](#)

4.2.1 Function Documentation

4.2.1.1 addAlimento()

```
void addAlimento (
    Cliente * listaCliente,
    Alimentos * listaAlimentos )
```

Função que pede os dados para adicionar um Alimento.

Parameters

<i>listaCliente</i>	
<i>listaAlimentos</i>	

Here is the call graph for this function: Here is the caller graph for this function:

4.2.1.2 adicionarAlimentos()

```
Alimentos* adicionarAlimentos (
    Alimentos * listaAlimentos,
    Alimentos * novoAlimento )
```

Função para adicionar um alimento ao ler um ficheiro txt.

Parameters

<i>listaAlimentos</i>	
<i>novoAlimento</i>	

Returns

Alimentos*

Here is the caller graph for this function:

4.2.1.3 dataEstaEntre()

```
int dataEstaEntre ( )
```

4.2.1.4 guardarBaseDeDadosAlimentos()

```
void guardarBaseDeDadosAlimentos (
    Alimentos * listaAlimentos )
```

Função que chama as 3 funções de guardar ficheiros.

Parameters

<i>listaAlimentos</i>	
-----------------------	--

Here is the call graph for this function: Here is the caller graph for this function:

4.2.1.5 introduzirAlimento()

```
Alimentos* introduzirAlimento (
    Alimentos * listaAlimentos,
    Alimentos * novoAlimento )
```

Função para adicionar um cliente.

Parameters

<i>listaAlimentos</i>	
<i>novoAlimento</i>	

Returns

Alimentos*

Here is the caller graph for this function:

4.2.1.6 lerBaseDeDadosAlimentos()

```
Alimentos* lerBaseDeDadosAlimentos ( )
```

Função que lê o txt de alimentos ao iniciar o programa e guarda numa lista ligada.

Returns

Alimentos*

Verificar se existe na diretoria

iterar a cada linha do ficheiro a fim de atribuir a um objeto e adiciona-lo a uma listaHere is the call graph for this function: Here is the caller graph for this function:

4.2.1.7 libertarMemoriaAlimentos()

```
void libertarMemoriaAlimentos (
    Alimentos * lista )
```

Liberta a memoria alocada pela lista & Uso de recursao.

Parameters

<i>lista</i>	
--------------	--

Here is the caller graph for this function:

4.2.1.8 NumPacientesPassamCals()

```
void NumPacientesPassamCals (
    Alimentos * listaAlimentos,
    Cliente * listaCliente )
```

Função que vê quantos Pacientes passaram o numero de calorias introduzidas.

Parameters

<i>listaAlimentos</i>	
<i>listaCliente</i>	

Here is the call graph for this function: Here is the caller graph for this function:

4.2.1.9 PacientesComRefeicaoForadeCals()

```
void PacientesComRefeicaoForadeCals (
    Alimentos * listaAlimentos,
    Cliente * listaCliente,
    Plano * listaPlano )
```

Função que Lista todos os Pacientes fora de Calorias em uma certa refeição, por ordem decrescente.

Parameters

<i>listaAlimentos</i>	
<i>listaCliente</i>	
<i>listaPlano</i>	

for para trocar a ordem dos Pacientes por ordem decrescente

Se nao trocar de ordem dá break

Vários For's e If's para percorrer as varias listas e imprimir os cliente com as respectivas condições
Here is the call graph for this function: Here is the caller graph for this function:

4.3 LESI_PI_TP_22531_23006_23131/Cliente.c File Reference

```
#include "ListaLigada.h"
#include "Cliente.h"
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <time.h>
Include dependency graph for Cliente.c:
```

Functions

- void `guardarNormalcl` (`Cliente` *lista)
Função que envia dados para o ficheiro txt.
- void `guardarBinariocl` (`Cliente` *lista)
Função que envia dados para o ficheiro txt em binario.
- void `guardarTabuladorescl` (`Cliente` *lista)
Função que envia dados para o ficheiro txt com tabs.
- `Cliente` * `lerBaseDeDados` ()
Função le as variaveis do ficheiro txt e guarda-as na lista.
- void `guardarBaseDeDados` (`Cliente` *lista)
Guarda as variaveis num ficheiro txt.
- void `libertarMemoria` (`Cliente` *lista)
Liberta a memoria alocada pela lista e uso de recursao.
- void `adicionar` (`Cliente` *listaCliente)
Funcao para pedir dados do cliente.
- `Cliente` * `adicionarCliente` (`Cliente` *listaCliente, `Cliente` *novoCliente)
Procedimento para adicionar um cliente no espaço null.
- int `obterIdAleatorio` (`Cliente` *listaClientes)
Obter um id aleatorio.
- bool `existeCliente` (`Cliente` *lista, int id)
função que verifica se existe cliente
- void `editar` (`Cliente` *listaCliente)
funcao para editar um cliente
- void `editarCliente` (`Cliente` *listaCliente, `Cliente` *clienteEscolhido)
Dado um cliente, pede-se os dados para edita-lo.
- `Cliente` * `encontrarCliente` (`Cliente` *lista, int id)
Funcao para encontrar um cliente.
- void `remover` (`Cliente` *listaCliente)
funcao para remover o cliente
- `Cliente` * `removerCliente` (`Cliente` *lista, int id)
Função para remover um cliente da lista.
- void `visualizarCliente` (`Cliente` *listaCliente)
funcao para visualizar cliente

4.3.1 Function Documentation

4.3.1.1 adicionar()

```
void adicionar (
    Cliente * listaCliente )
```

Funcao para pedir dados do cliente.

Parameters

<i>listaCliente</i>	
---------------------	--

Variavel do tipo Cliente que retorna um ponteiro para o espaço alocado

gerar um id aleatorio

Vai percorrer a lista ate encontrar um espaço nullHere is the call graph for this function: Here is the caller graph for this function:

4.3.1.2 adicionarCliente()

```
Cliente* adicionarCliente (
    Cliente * listaCliente,
    Cliente * novoCliente )
```

Procedimento para adicionar um cliente no espaço null.

Parameters

<i>listaCliente</i>	
<i>novoCliente</i>	

Returns

Cliente*

Se a lista for NULL, devolve o meu novo cliente

Se lista for diferente de NULL, o proximo cliente vai continuar a procurar um espaco "NULL"

Da return a lista depois da atualizacaoHere is the caller graph for this function:

4.3.1.3 editar()

```
void editar (
    Cliente * listaCliente )
```

funcao para editar um cliente

Parameters

<i>listaCliente</i>	
---------------------	--

Here is the call graph for this function: Here is the caller graph for this function:

4.3.1.4 editarCliente()

```
void editarCliente (
    Cliente * listaCliente,
    Cliente * clienteEscolhido )
```

Dado um cliente, pede-se os dados para edita-lo.

Parameters

<i>listaCliente</i>	
<i>clienteEscolhido</i>	

Here is the caller graph for this function:

4.3.1.5 encontrarCliente()

```
Cliente* encontrarCliente (
    Cliente * lista,
    int id )
```

Funcao para encontrar um cliente.

Parameters

<i>lista</i>	
<i>id</i>	

Returns

Cliente*

Here is the caller graph for this function:

4.3.1.6 existeCliente()

```
bool existeCliente (
    Cliente * lista,
    int id )
```

função que verifica se existe cliente

Parameters

<i>lista</i>	
<i>id</i>	

Returns

true
false

Here is the caller graph for this function:

4.3.1.7 guardarBaseDeDados()

```
void guardarBaseDeDados (
    Cliente * lista )
```

Guarda as variaveis num ficheiro txt.

Parameters

<i>lista</i>	
--------------	--

Here is the call graph for this function: Here is the caller graph for this function:

4.3.1.8 guardarBinariocl()

```
void guardarBinariocl (
    Cliente * lista )
```

Função que envia dados para o ficheiro txt em binario.

Parameters

<i>lista</i>	
--------------	--

Here is the caller graph for this function:

4.3.1.9 guardarNormalcl()

```
void guardarNormalcl (
    Cliente * lista )
```

Função que envia dados para o ficheiro txt.

Parameters

<i>lista</i>	
--------------	--

Here is the caller graph for this function:

4.3.1.10 guardarTabuladorescl()

```
void guardarTabuladorescl (
    Cliente * lista )
```

Função que envia dados para o ficheiro txt com tabs.

Parameters

<i>lista</i>	
--------------	--

Here is the caller graph for this function:

4.3.1.11 lerBaseDeDados()

```
Cliente* lerBaseDeDados ( )
```

Função le as variaveis do ficheiro txt e guarda-as na lista.

Returns

Cliente*

Verificar se existe na diretoria

Iterar a cada linha do ficheiro a fim de atribuir a um objeto e adiciona-lo a uma lista
Here is the call graph for this function: Here is the caller graph for this function:

4.3.1.12 libertarMemoria()

```
void libertarMemoria (
    Cliente * lista )
```

Liberta a memoria alocada pela lista e uso de recursao.

Parameters

<i>lista</i>	
--------------	--

Here is the caller graph for this function:

4.3.1.13 obterIdAleatorio()

```
int obterIdAleatorio (
    Cliente * listaClientes )
```

Obter um id aleatorio.

Parameters

<i>listaClientes</i>	
----------------------	--

Returns

int

Here is the call graph for this function: Here is the caller graph for this function:

4.3.1.14 remover()

```
void remover (
    Cliente * listaCliente )
```

funcao para remover o cliente

Parameters

<i>listaCliente</i>	
---------------------	--

Here is the call graph for this function: Here is the caller graph for this function:

4.3.1.15 removerCliente()

```
Cliente* removerCliente (
    Cliente * lista,
    int id )
```

Função para remover um cliente da lista.

Parameters

<i>lista</i>	
<i>id</i>	

Returns

Cliente*

Here is the caller graph for this function:

4.3.1.16 visualizarCliente()

```
void visualizarCliente (
    Cliente * listaCliente )
```

funcao para visualizar cliente

Parameters

<i>listaCliente</i>	
---------------------	--

Here is the call graph for this function: Here is the caller graph for this function:

4.4 LESI_PI_TP_22531_23006_23131/Cliente.h File Reference

This graph shows which files directly or indirectly include this file:

Functions

- [Cliente *](#) [lerBaseDeDados](#) ()
Função le as variaveis do ficheiro txt e guarda-as na lista.
- void [guardarBaseDeDados](#) ([Cliente *](#))
Guarda as variaveis num ficheiro txt.
- void [libertarMemoria](#) ([Cliente *](#))
Liberta a memoria alocada pela lista e uso de recursao.
- void [adicionar](#) ([Cliente *](#))
Funcao para pedir dados do cliente.
- [Cliente *](#) [adicionarCliente](#) ([Cliente *](#), [Cliente *](#))
Procedimento para adicionar um cliente no espaço null.
- int [obterIdAleatorio](#) ([Cliente *](#))
Obter um id aleatorio.
- bool [existeCliente](#) ([Cliente *](#), int)
função que verifica se existe cliente
- void [editar](#) ([Cliente *](#))
funcao para editar um cliente
- void [editarCliente](#) ([Cliente *](#), [Cliente *](#))
Dado um cliente, pede-se os dados para edita-lo.
- [Cliente *](#) [encontrarCliente](#) ([Cliente *](#), int)
Funcao para encontrar um cliente.
- void [remover](#) ([Cliente *](#))
funcao para remover o cliente
- [Cliente *](#) [removerCliente](#) ([Cliente *](#), int)
Função para remover um cliente da lista.
- void [visualizarCliente](#) ([Cliente *](#))
funcao para visualizar cliente

4.4.1 Function Documentation

4.4.1.1 adicionar()

```
void adicionar (
    Cliente \* listaCliente )
```

Funcao para pedir dados do cliente.

Parameters

<i>listaCliente</i>	
---------------------	--

Variavel do tipo Cliente que retorna um ponteiro para o espaço alocado

gerar um id aleatorio

Vai percorrer a lista ate encontrar um espaço nullHere is the call graph for this function: Here is the caller graph for this function:

4.4.1.2 adicionarCliente()

```
Cliente* adicionarCliente (
    Cliente * listaCliente,
    Cliente * novoCliente )
```

Procedimento para adicionar um cliente no espaço null.

Parameters

<i>listaCliente</i>	
<i>novoCliente</i>	

Returns

Cliente*

Se a lista for NULL, devolve o meu novo cliente

Se lista for diferente de NULL, o proximo cliente vai continuar a procurar um espaco "NULL"

Da return a lista depois da atualizacaoHere is the caller graph for this function:

4.4.1.3 editar()

```
void editar (
    Cliente * listaCliente )
```

funcao para editar um cliente

Parameters

<i>listaCliente</i>	
---------------------	--

Here is the call graph for this function: Here is the caller graph for this function:

4.4.1.4 editarCliente()

```
void editarCliente (
    Cliente * listaCliente,
    Cliente * clienteEscolhido )
```

Dado um cliente, pede-se os dados para edita-lo.

Parameters

<i>listaCliente</i>	
<i>clienteEscolhido</i>	

Here is the caller graph for this function:

4.4.1.5 encontrarCliente()

```
Cliente* encontrarCliente (
    Cliente * lista,
    int id )
```

Funcao para encontrar um cliente.

Parameters

<i>lista</i>	
<i>id</i>	

Returns

Cliente*

Here is the caller graph for this function:

4.4.1.6 existeCliente()

```
bool existeCliente (
    Cliente * lista,
    int id )
```

função que verifica se existe cliente

Parameters

<i>lista</i>	
<i>id</i>	

Returns

true
false

Here is the caller graph for this function:

4.4.1.7 guardarBaseDeDados()

```
void guardarBaseDeDados (
    Cliente * lista )
```

Guarda as variaveis num ficheiro txt.

Parameters

<i>lista</i>	
--------------	--

Here is the call graph for this function: Here is the caller graph for this function:

4.4.1.8 lerBaseDeDados()

```
Cliente* lerBaseDeDados ( )
```

Função le as variaveis do ficheiro txt e guarda-as na lista.

Returns

Cliente*

Verificar se existe na diretoria

Iterar a cada linha do ficheiro a fim de atribuir a um objeto e adiciona-lo a uma lista
Here is the call graph for this function: Here is the caller graph for this function:

4.4.1.9 libertarMemoria()

```
void libertarMemoria (
    Cliente * lista )
```

Liberta a memoria alocada pela lista e uso de recursao.

Parameters

<i>lista</i>	
--------------	--

Here is the caller graph for this function:

4.4.1.10 obterIdAleatorio()

```
int obterIdAleatorio (
    Cliente * listaClientes )
```

Obter um id aleatorio.

Parameters

<i>listaClientes</i>	
----------------------	--

Returns

int

Here is the call graph for this function: Here is the caller graph for this function:

4.4.1.11 remover()

```
void remover (
    Cliente * listaCliente )
```

funcao para remover o cliente

Parameters

<i>listaCliente</i>	
---------------------	--

Here is the call graph for this function: Here is the caller graph for this function:

4.4.1.12 removerCliente()

```
Cliente* removerCliente (
    Cliente * lista,
    int id )
```

Função para remover um cliente da lista.

Parameters

<i>lista</i>	
<i>id</i>	

Returns

Cliente*

Here is the caller graph for this function:

4.4.1.13 visualizarCliente()

```
void visualizarCliente (
    Cliente * listaCliente )
```

funcao para visualizar cliente

Parameters

<i>listaCliente</i>	
---------------------	--

Here is the call graph for this function: Here is the caller graph for this function:

4.5 LESI_PI_TP_22531_23006_23131/ListaLigada.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <time.h>
#include <stdbool.h>
#include "ListaLigada.h"
#include "Cliente.h"
#include "Alimentos.h"
#include "Plano.h"
```

Include dependency graph for ListaLigada.c:

Functions

- void [menuPrincipal](#) ([Cliente](#) *listaCliente, [Alimentos](#) *listaAlimentos, [Plano](#) *listaPlano)
Função que imprime o menu do programa.

4.5.1 Function Documentation

4.5.1.1 menuPrincipal()

```
void menuPrincipal (
    Cliente * listaCliente,
    Alimentos * listaAlimentos,
    Plano * listaPlano )
```

Função que imprime o menu do programa.

Parameters

<i>listaCliente</i>	
<i>listaAlimentos</i>	
<i>listaPlano</i>	

Here is the call graph for this function: Here is the caller graph for this function:

4.6 LESI_PI_TP_22531_23006_23131/ListaLigada.h File Reference

```
#include <stdbool.h>
```

Include dependency graph for ListaLigada.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct [_data](#)
- struct [_cliente](#)
- struct [_alimentos](#)
- struct [_plano](#)

Typedefs

- typedef struct [_data](#) [Data](#)
- typedef struct [_cliente](#) [Cliente](#)
- typedef struct [_alimentos](#) [Alimentos](#)
- typedef struct [_plano](#) [Plano](#)

Functions

- void [menuPrincipal](#) ([Cliente](#) *, [Alimentos](#) *, [Plano](#) *)
Função que imprime o menu do programa.

4.6.1 Typedef Documentation

4.6.1.1 Alimentos

```
typedef struct \_alimentos Alimentos
```

4.6.1.2 Cliente

```
typedef struct \_cliente Cliente
```

4.6.1.3 Data

```
typedef struct _data Data
```

4.6.1.4 Plano

```
typedef struct _plano Plano
```

4.6.2 Function Documentation

4.6.2.1 menuPrincipal()

```
void menuPrincipal (
    Cliente * listaCliente,
    Alimentos * listaAlimentos,
    Plano * listaPlano )
```

Função que imprime o menu do programa.

Parameters

<i>listaCliente</i>	
<i>listaAlimentos</i>	
<i>listaPlano</i>	

Here is the call graph for this function: Here is the caller graph for this function:

4.7 LESI_PI_TP_22531_23006_23131/main.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <math.h>
#include "ListaLigada.h"
#include "Cliente.h"
#include "Alimentos.h"
#include "Plano.h"
Include dependency graph for main.c:
```

Functions

- void [displayHelp](#) ()

- Função para correr o -help.*

 - void `executeBinCommand` (const char *fileName)

Função que pede o ficheiro, lê-o em binario e imprime em texto normal.
- void `executeTabCommand` (const char *fileName)

Função que pede um ficheiro e lê-o um ficheiro organizado por tabs.
- int `main` (int argc, char *argv[])

Função principal que vai correr outras funções como (Ler base de dados / Menu Principal / Guardar Base de dados / Limpar Memoria)

4.7.1 Function Documentation

4.7.1.1 displayHelp()

```
void displayHelp ( )
```

Função para correr o -help.

Here is the caller graph for this function:

4.7.1.2 executeBinCommand()

```
void executeBinCommand (
    const char * fileName )
```

Função que pede o ficheiro, lê-o em binario e imprime em texto normal.

Parameters

<code>fileName</code>	
-----------------------	--

Here is the caller graph for this function:

4.7.1.3 executeTabCommand()

```
void executeTabCommand (
    const char * fileName )
```

Função que pede um ficheiro e lê-o um ficheiro organizado por tabs.

Parameters

<code>fileName</code>	
-----------------------	--

Here is the caller graph for this function:

4.7.1.4 main()

```
int main (
    int argc,
    char * argv[] )
```

Função principal que vai correr outras funções como (Ler base de dados / Menu Principal / Guardar Base de dados / Limpar Memória)

Parameters

<i>argc</i>	
<i>argv</i>	

Returns

int

Here is the call graph for this function:

4.8 LESI_PI_TP_22531_23006_23131/Plano.c File Reference

```
#include "ListaLigada.h"
#include "Plano.h"
#include "Cliente.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <time.h>
```

Include dependency graph for Plano.c:

Functions

- void [guardarNormal](#) ([Plano](#) *listaPlano)
Função que envia dados para o ficheiro txt.
- void [guardarBinario](#) ([Plano](#) *listaPlano)
Função que envia dados para o ficheiro txt em binario.
- void [guardarTabuladores](#) ([Plano](#) *listaPlano)
Função que envia dados para o ficheiro txt com tabs.
- [Plano](#) * [lerBaseDeDadosPlano](#) ()
Função que lê o txt de Planos ao iniciar o programa e guarda numa lista ligada.
- void [guardarBaseDeDadosPlano](#) ([Plano](#) *listaPlano)
Função que chama as 3 funções de guardar os dados.
- void [addPlano](#) ([Cliente](#) *listaCliente, [Plano](#) *listaPlano)
Função que pede os dados do Plano.
- [Plano](#) * [introduzirPlano](#) ([Plano](#) *listaPlano, [Plano](#) *novoPlano)
Função para adicionar um Plano.
- [Plano](#) * [adicionarPlano](#) ([Plano](#) *listaPlano, [Plano](#) *novoPlano)

- Função para adicionar um Plano ao ler a base de dados.*
- void `libertarMemoriaPlano` (`Plano *lista`)
Liberta a memoria alocada pela lista e uso de recursao.
- void `ListarPlanosPorCliente` (`Plano *listaPlano`, `Cliente *listaCliente`)
Função para listar Planos que estão entre 2 datas pedidas e que são de uma certa refeição.
- void `MediaCalPorCliente` (`Cliente *listaCliente`, `Alimentos *listaAlimentos`)
Função que faz a media de calorias de todos os clientes entre 2 datas pedidas.
- void `TabelaDados` (`Cliente *listaCliente`, `Alimentos *listaAlimentos`, `Plano *listaPlano`)
Função que faz uma tabela com todos os Planos e com as calorias consumidas.

4.8.1 Function Documentation

4.8.1.1 addPlano()

```
void addPlano (
    Cliente * listaCliente,
    Plano * listaPlano )
```

Função que pede os dados do Plano.

Parameters

<code>listaCliente</code>	
<code>listaPlano</code>	

Here is the call graph for this function: Here is the caller graph for this function:

4.8.1.2 adicionarPlano()

```
Plano* adicionarPlano (
    Plano * listaPlano,
    Plano * novoPlano )
```

Função para adicionar um Plano ao ler a base de dados.

Parameters

<code>listaPlano</code>	
<code>novoPlano</code>	

Returns

`Plano*`

Here is the caller graph for this function:

4.8.1.3 guardarBaseDeDadosPlano()

```
void guardarBaseDeDadosPlano (
    Plano * listaPlano )
```

Função que chama as 3 funções de guardar os dados.

Parameters

<i>listaPlano</i>	
-------------------	--

Here is the call graph for this function: Here is the caller graph for this function:

4.8.1.4 guardarBinario()

```
void guardarBinario (
    Plano * listaPlano )
```

Função que envia dados para o ficheiro txt em binario.

Parameters

<i>listaPlano</i>	
-------------------	--

Here is the caller graph for this function:

4.8.1.5 guardarNormal()

```
void guardarNormal (
    Plano * listaPlano )
```

Função que envia dados para o ficheiro txt.

Parameters

<i>listaPlano</i>	
-------------------	--

Here is the caller graph for this function:

4.8.1.6 guardarTabuladores()

```
void guardarTabuladores (
    Plano * listaPlano )
```

Função que envia dados para o ficheiro txt com tabs.

Parameters

<i>listaPlano</i>	
-------------------	--

Here is the caller graph for this function:

4.8.1.7 introduzirPlano()

```
Plano* introduzirPlano (
    Plano * listaPlano,
    Plano * novoPlano )
```

Função para adicionar um Plano.

Parameters

<i>listaPlano</i>	
<i>novoPlano</i>	

Returns

Plano*

Here is the caller graph for this function:

4.8.1.8 lerBaseDeDadosPlano()

```
Plano* lerBaseDeDadosPlano ( )
```

Função que lê o txt de Planos ao iniciar o programa e guarda numa lista ligada.

Returns

Plano*

Here is the call graph for this function: Here is the caller graph for this function:

4.8.1.9 libertarMemoriaPlano()

```
void libertarMemoriaPlano (
    Plano * lista )
```

Liberta a memoria alocada pela lista e uso de recursao.

Parameters

<i>lista</i>	
--------------	--

Here is the caller graph for this function:

4.8.1.10 ListarPlanosPorCliente()

```
void ListarPlanosPorCliente (
    Plano * listaPlano,
    Cliente * listaCliente )
```

Função para listar Planos que estão entre 2 datas pedidas e que são de uma certa refeição.

Parameters

<i>listaPlano</i>	
<i>listaCliente</i>	

Here is the call graph for this function: Here is the caller graph for this function:

4.8.1.11 MediaCalPorCliente()

```
void MediaCalPorCliente (
    Cliente * listaCliente,
    Alimentos * listaAlimentos )
```

Função que faz a media de calorias de todos os clientes entre 2 datas pedidas.

Parameters

<i>listaCliente</i>	
<i>listaAlimentos</i>	

Here is the call graph for this function: Here is the caller graph for this function:

4.8.1.12 TabelaDados()

```
void TabelaDados (
    Cliente * listaCliente,
    Alimentos * listaAlimentos,
    Plano * listaPlano )
```

Função que faz uma tabela com todos os Planos e com as calorias consumidas.

Parameters

<i>listaCliente</i>	
<i>listaAlimentos</i>	
<i>listaPlano</i>	

Here is the call graph for this function: Here is the caller graph for this function:

4.9 LESI_PI_TP_22531_23006_23131/Plano.h File Reference

This graph shows which files directly or indirectly include this file:

Functions

- [Plano *](#) [lerBaseDeDadosPlano](#) ()
Função que lê o txt de Planos ao iniciar o programa e guarda numa lista ligada.
- void [guardarBaseDeDadosPlano](#) ([Plano *](#))
Função que chama as 3 funções de guardar os dados.
- void [addPlano](#) ([Cliente *](#), [Plano *](#))
Função que pede os dados do Plano.
- [Plano *](#) [adicionarPlano](#) ([Plano *](#), [Plano *](#))
Função para adicionar um Plano ao ler a base de dados.
- [Plano *](#) [introduzirPlano](#) ([Plano *](#), [Plano *](#))
Função para adicionar um Plano.
- void [libertarMemoriaPlano](#) ([Plano *](#))
Liberta a memoria alocada pela lista e uso de recursao.
- void [ListarPlanosPorCliente](#) ([Plano *](#), [Cliente *](#))
Função para listar Planos que estão entre 2 datas pedidas e que são de uma certa refeição.
- void [MediaCalPorCliente](#) ([Cliente *](#), [Alimentos *](#))
Função que faz a média de calorias de todos os clientes entre 2 datas pedidas.
- void [TabelaDados](#) ([Cliente *](#), [Alimentos *](#), [Plano *](#))
Função que faz uma tabela com todos os Planos e com as calorias consumidas.

4.9.1 Function Documentation

4.9.1.1 addPlano()

```
void addPlano (
    Cliente \* listaCliente,
    Plano \* listaPlano )
```

Função que pede os dados do Plano.

Parameters

<i>listaCliente</i>	
<i>listaPlano</i>	

Here is the call graph for this function: Here is the caller graph for this function:

4.9.1.2 adicionarPlano()

```
Plano\* adicionarPlano (
```

```
Plano * listaPlano,
Plano * novoPlano )
```

Função para adicionar um Plano ao ler a base de dados.

Parameters

<i>listaPlano</i>	
<i>novoPlano</i>	

Returns

Plano*

Here is the caller graph for this function:

4.9.1.3 guardarBaseDeDadosPlano()

```
void guardarBaseDeDadosPlano (
    Plano * listaPlano )
```

Função que chama as 3 funções de guardar os dados.

Parameters

<i>listaPlano</i>	
-------------------	--

Here is the call graph for this function: Here is the caller graph for this function:

4.9.1.4 introduzirPlano()

```
Plano* introduzirPlano (
    Plano * listaPlano,
    Plano * novoPlano )
```

Função para adicionar um Plano.

Parameters

<i>listaPlano</i>	
<i>novoPlano</i>	

Returns

Plano*

Here is the caller graph for this function:

4.9.1.5 lerBaseDeDadosPlano()

```
Plano* lerBaseDeDadosPlano ( )
```

Função que lê o txt de Planos ao iniciar o programa e guarda numa lista ligada.

Returns

Plano*

Here is the call graph for this function: Here is the caller graph for this function:

4.9.1.6 libertarMemoriaPlano()

```
void libertarMemoriaPlano (
    Plano * lista )
```

Liberta a memoria alocada pela lista e uso de recursao.

Parameters

<i>lista</i>	
--------------	--

Here is the caller graph for this function:

4.9.1.7 ListarPlanosPorCliente()

```
void ListarPlanosPorCliente (
    Plano * listaPlano,
    Cliente * listaCliente )
```

Função para listar Planos que estão entre 2 datas pedidas e que são de uma certa refeição.

Parameters

<i>listaPlano</i>	
<i>listaCliente</i>	

Here is the call graph for this function: Here is the caller graph for this function:

4.9.1.8 MediaCalPorCliente()

```
void MediaCalPorCliente (
    Cliente * listaCliente,
    Alimentos * listaAlimentos )
```

Função que faz a media de calorias de todos os clientes entre 2 datas pedidas.

Parameters

<i>listaCliente</i>	
<i>listaAlimentos</i>	

Here is the call graph for this function: Here is the caller graph for this function:

4.9.1.9 TabelaDados()

```
void TabelaDados (
    Cliente * listaCliente,
    Alimentos * listaAlimentos,
    Plano * listaPlano )
```

Função que faz uma tabela com todos os Planos e com as calorias consumidas.

Parameters

<i>listaCliente</i>	
<i>listaAlimentos</i>	
<i>listaPlano</i>	

Here is the call graph for this function: Here is the caller graph for this function:

Index

- [_alimentos, 5](#)
 - [alimento, 5](#)
 - [cal, 5](#)
 - [data, 5](#)
 - [id_cliente, 6](#)
 - [next, 6](#)
 - [refeicao, 6](#)
 - [_cliente, 6](#)
 - [next, 6](#)
 - [nome, 7](#)
 - [num_cliente, 7](#)
 - [telemovel, 7](#)
 - [_data, 7](#)
 - [ano, 7](#)
 - [dia, 7](#)
 - [mes, 8](#)
 - [_plano, 8](#)
 - [calmax, 8](#)
 - [calmin, 8](#)
 - [data_fim, 8](#)
 - [data_inicio, 9](#)
 - [id_cliente, 9](#)
 - [next, 9](#)
 - [refeicao, 9](#)
- [addAlimento](#)
 - [Alimentos.h, 12](#)
- [addPlano](#)
 - [Plano.c, 31](#)
 - [Plano.h, 35](#)
- [adicionar](#)
 - [Cliente.c, 15](#)
 - [Cliente.h, 21](#)
- [adicionarAlimentos](#)
 - [Alimentos.h, 12](#)
- [adicionarCliente](#)
 - [Cliente.c, 16](#)
 - [Cliente.h, 22](#)
- [adicionarPlano](#)
 - [Plano.c, 31](#)
 - [Plano.h, 35](#)
- [alimento](#)
 - [_alimentos, 5](#)
- [Alimentos](#)
 - [ListaLigada.h, 27](#)
- [Alimentos.h](#)
 - [addAlimento, 12](#)
 - [adicionarAlimentos, 12](#)
 - [dataEstaEntre, 12](#)
 - [guardarBaseDeDadosAlimentos, 12](#)
 - [introduzirAlimento, 13](#)
 - [lerBaseDeDadosAlimentos, 13](#)
 - [libertarMemoriaAlimentos, 13](#)
 - [NumPacientesPassamCals, 14](#)
 - [PacientesComRefeicaoForadeCals, 14](#)
- [ano](#)
 - [_data, 7](#)
- [cal](#)
 - [_alimentos, 5](#)
- [calmax](#)
 - [_plano, 8](#)
- [calmin](#)
 - [_plano, 8](#)
- [Cliente](#)
 - [ListaLigada.h, 27](#)
- [Cliente.c](#)
 - [adicionar, 15](#)
 - [adicionarCliente, 16](#)
 - [editar, 16](#)
 - [editarCliente, 16](#)
 - [encontrarCliente, 17](#)
 - [existeCliente, 17](#)
 - [guardarBaseDeDados, 18](#)
 - [guardarBinariocl, 18](#)
 - [guardarNormalcl, 18](#)
 - [guardarTabuladorescl, 18](#)
 - [lerBaseDeDados, 19](#)
 - [libertarMemoria, 19](#)
 - [obterIdAleatorio, 19](#)
 - [remover, 20](#)
 - [removerCliente, 20](#)
 - [visualizarCliente, 20](#)
- [Cliente.h](#)
 - [adicionar, 21](#)
 - [adicionarCliente, 22](#)
 - [editar, 22](#)
 - [editarCliente, 22](#)
 - [encontrarCliente, 23](#)
 - [existeCliente, 23](#)
 - [guardarBaseDeDados, 24](#)
 - [lerBaseDeDados, 24](#)
 - [libertarMemoria, 24](#)
 - [obterIdAleatorio, 24](#)
 - [remover, 25](#)
 - [removerCliente, 25](#)
 - [visualizarCliente, 25](#)
- [Data](#)
 - [ListaLigada.h, 27](#)

- data
 - _alimentos, 5
- data_fim
 - _plano, 8
- data_inicio
 - _plano, 9
- dataEstaEntre
 - Alimentos.h, 12
- dia
 - _data, 7
- displayHelp
 - main.c, 29
- editar
 - Cliente.c, 16
 - Cliente.h, 22
- editarCliente
 - Cliente.c, 16
 - Cliente.h, 22
- encontrarCliente
 - Cliente.c, 17
 - Cliente.h, 23
- executeBinCommand
 - main.c, 29
- executeTabCommand
 - main.c, 29
- existeCliente
 - Cliente.c, 17
 - Cliente.h, 23
- guardarBaseDeDados
 - Cliente.c, 18
 - Cliente.h, 24
- guardarBaseDeDadosAlimentos
 - Alimentos.h, 12
- guardarBaseDeDadosPlano
 - Plano.c, 31
 - Plano.h, 36
- guardarBinario
 - Plano.c, 32
- guardarBinariocl
 - Cliente.c, 18
- guardarNormal
 - Plano.c, 32
- guardarNormalcl
 - Cliente.c, 18
- guardarTabuladores
 - Plano.c, 32
- guardarTabuladorescl
 - Cliente.c, 18
- id_cliente
 - _alimentos, 6
 - _plano, 9
- introduzirAlimento
 - Alimentos.h, 13
- introduzirPlano
 - Plano.c, 33
 - Plano.h, 36
- lerBaseDeDados
 - Cliente.c, 19
 - Cliente.h, 24
- lerBaseDeDadosAlimentos
 - Alimentos.h, 13
- lerBaseDeDadosPlano
 - Plano.c, 33
 - Plano.h, 36
- LESI_PI_TP_22531_23006_23131/Alimentos.c, 11
- LESI_PI_TP_22531_23006_23131/Alimentos.h, 11
- LESI_PI_TP_22531_23006_23131/Cliente.c, 14
- LESI_PI_TP_22531_23006_23131/Cliente.h, 21
- LESI_PI_TP_22531_23006_23131/ListaLigada.c, 26
- LESI_PI_TP_22531_23006_23131/ListaLigada.h, 27
- LESI_PI_TP_22531_23006_23131/main.c, 28
- LESI_PI_TP_22531_23006_23131/Plano.c, 30
- LESI_PI_TP_22531_23006_23131/Plano.h, 35
- libertarMemoria
 - Cliente.c, 19
 - Cliente.h, 24
- libertarMemoriaAlimentos
 - Alimentos.h, 13
- libertarMemoriaPlano
 - Plano.c, 33
 - Plano.h, 37
- ListaLigada.c
 - menuPrincipal, 26
- ListaLigada.h
 - Alimentos, 27
 - Cliente, 27
 - Data, 27
 - menuPrincipal, 28
 - Plano, 28
- ListarPlanosPorCliente
 - Plano.c, 34
 - Plano.h, 37
- main
 - main.c, 29
- main.c
 - displayHelp, 29
 - executeBinCommand, 29
 - executeTabCommand, 29
 - main, 29
- MediaCalPorCliente
 - Plano.c, 34
 - Plano.h, 37
- menuPrincipal
 - ListaLigada.c, 26
 - ListaLigada.h, 28
- mes
 - _data, 8
- next
 - _alimentos, 6
 - _cliente, 6
 - _plano, 9
- nome
 - _cliente, 7

- num_cliente
 - _cliente, [7](#)
- NumPacientesPassamCals
 - Alimentos.h, [14](#)
- obterIdAleatorio
 - Cliente.c, [19](#)
 - Cliente.h, [24](#)
- PacientesComRefeicaoForadeCals
 - Alimentos.h, [14](#)
- Plano
 - ListaLigada.h, [28](#)
- Plano.c
 - addPlano, [31](#)
 - adicionarPlano, [31](#)
 - guardarBaseDeDadosPlano, [31](#)
 - guardarBinario, [32](#)
 - guardarNormal, [32](#)
 - guardarTabuladores, [32](#)
 - introduzirPlano, [33](#)
 - lerBaseDeDadosPlano, [33](#)
 - libertarMemoriaPlano, [33](#)
 - ListarPlanosPorCliente, [34](#)
 - MediaCalPorCliente, [34](#)
 - TabelaDados, [34](#)
- Plano.h
 - addPlano, [35](#)
 - adicionarPlano, [35](#)
 - guardarBaseDeDadosPlano, [36](#)
 - introduzirPlano, [36](#)
 - lerBaseDeDadosPlano, [36](#)
 - libertarMemoriaPlano, [37](#)
 - ListarPlanosPorCliente, [37](#)
 - MediaCalPorCliente, [37](#)
 - TabelaDados, [38](#)
- refeicao
 - _alimentos, [6](#)
 - _plano, [9](#)
- remover
 - Cliente.c, [20](#)
 - Cliente.h, [25](#)
- removerCliente
 - Cliente.c, [20](#)
 - Cliente.h, [25](#)
- TabelaDados
 - Plano.c, [34](#)
 - Plano.h, [38](#)
- telemovel
 - _cliente, [7](#)
- visualizarCliente
 - Cliente.c, [20](#)
 - Cliente.h, [25](#)