

My Project

Generated by Doxygen 1.9.8

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 Contexto Struct Reference	5
3.1.1 Detailed Description	5
3.1.2 Member Data Documentation	6
3.1.2.1 lista_produtos	6
3.1.2.2 logs_vendas	6
3.1.2.3 produto_selecionado	6
3.2 Fila Struct Reference	6
3.2.1 Detailed Description	6
3.2.2 Member Data Documentation	6
3.2.2.1 frente	6
3.2.2.2 mensagens	7
3.2.2.3 tamanho	7
3.2.2.4 tras	7
3.3 Log Struct Reference	7
3.3.1 Detailed Description	8
3.3.2 Member Data Documentation	8
3.3.2.1 produto_id	8
3.3.2.2 prox	8
3.3.2.3 timestamp	8
3.3.2.4 troco	8
3.3.2.5 valor_pago	8
3.4 Pilha Struct Reference	8
3.4.1 Detailed Description	9
3.4.2 Member Data Documentation	9
3.4.2.1 topo	9
3.4.2.2 valores	9
3.5 PilhaEstados Struct Reference	9
3.5.1 Detailed Description	9
3.5.2 Member Data Documentation	10
3.5.2.1 estados	10
3.5.2.2 topo	10
3.6 produto Struct Reference	10
3.6.1 Detailed Description	10
3.6.2 Member Data Documentation	11
3.6.2.1 ant	11
3.6.2.2 estoque	11

3.6.2.3 id	11
3.6.2.4 nome	11
3.6.2.5 preco	11
3.6.2.6 prox	11
4 File Documentation	13
4.1 config.c File Reference	13
4.1.1 Detailed Description	13
4.1.2 Function Documentation	14
4.1.2.1 menu_configuracao()	14
4.1.2.2 verificar_senha()	14
4.2 config.h File Reference	14
4.2.1 Detailed Description	15
4.2.2 Function Documentation	15
4.2.2.1 menu_configuracao()	15
4.2.2.2 verificar_senha()	16
4.3 config.h	16
4.4 estatisticas.c File Reference	16
4.4.1 Detailed Description	17
4.4.2 Function Documentation	17
4.4.2.1 gerar_estatisticas()	17
4.4.2.2 salvar_estatisticas_csv()	18
4.5 estatisticas.h File Reference	18
4.5.1 Detailed Description	19
4.5.2 Function Documentation	19
4.5.2.1 gerar_estatisticas()	19
4.5.2.2 salvar_estatisticas_csv()	20
4.6 estatisticas.h	20
4.7 fila.c File Reference	20
4.7.1 Detailed Description	21
4.7.2 Function Documentation	21
4.7.2.1 desenfileirar()	21
4.7.2.2 enfileirar()	22
4.7.2.3 frente_fila()	22
4.7.2.4 inicializar_fila()	22
4.8 fila.h File Reference	22
4.8.1 Detailed Description	23
4.8.2 Function Documentation	24
4.8.2.1 desenfileirar()	24
4.8.2.2 enfileirar()	24
4.8.2.3 frente_fila()	24
4.8.2.4 inicializar_fila()	25

4.9 fila.h	25
4.10 lista_produtos.c File Reference	25
4.10.1 Detailed Description	26
4.10.2 Function Documentation	26
4.10.2.1 adicionar_produto()	26
4.10.2.2 buscar_produto()	26
4.10.2.3 carregar_produtos_csv()	27
4.10.2.4 liberar_lista_produtos()	27
4.10.2.5 listar_produtos()	27
4.10.2.6 remover_produto()	28
4.10.2.7 salvar_produtos_csv()	28
4.11 lista_produtos.h File Reference	28
4.11.1 Detailed Description	29
4.11.2 Function Documentation	30
4.11.2.1 adicionar_produto()	30
4.11.2.2 buscar_produto()	30
4.11.2.3 carregar_produtos_csv()	30
4.11.2.4 liberar_lista_produtos()	31
4.11.2.5 listar_produtos()	31
4.11.2.6 remover_produto()	31
4.11.2.7 salvar_produtos_csv()	32
4.12 lista_produtos.h	32
4.13 log_vendas.c File Reference	32
4.13.1 Detailed Description	33
4.13.2 Function Documentation	33
4.13.2.1 adicionar_log()	33
4.13.2.2 carregar_logs()	34
4.13.2.3 exhibir_logs()	34
4.13.2.4 liberar_logs()	34
4.13.2.5 salvar_logs()	34
4.14 log_vendas.h File Reference	35
4.14.1 Detailed Description	36
4.14.2 Function Documentation	36
4.14.2.1 adicionar_log()	36
4.14.2.2 carregar_logs()	36
4.14.2.3 exhibir_logs()	36
4.14.2.4 liberar_logs()	37
4.14.2.5 salvar_logs()	37
4.15 log_vendas.h	37
4.16 main.c File Reference	38
4.16.1 Detailed Description	38
4.16.2 Function Documentation	38

4.16.2.1 main()	38
4.17 pagamento.c File Reference	39
4.17.1 Detailed Description	39
4.17.2 Function Documentation	39
4.17.2.1 calcular_troco()	39
4.17.2.2 desfazer_moeda()	40
4.17.2.3 inserir_moedas()	40
4.18 pagamento.h File Reference	40
4.18.1 Detailed Description	41
4.18.2 Function Documentation	41
4.18.2.1 calcular_troco()	41
4.18.2.2 desfazer_moeda()	42
4.18.2.3 inserir_moedas()	42
4.19 pagamento.h	42
4.20 pilha.c File Reference	42
4.20.1 Detailed Description	43
4.21 pilha.h File Reference	43
4.21.1 Detailed Description	45
4.22 pilha.h	45
4.23 states.c File Reference	45
4.23.1 Detailed Description	46
4.23.2 Function Documentation	46
4.23.2.1 estado_configuracao()	46
4.23.2.2 estado_menu_principal()	46
4.23.2.3 estado_pagamento()	47
4.23.2.4 estado_selecao_produto()	47
4.24 states.h File Reference	47
4.24.1 Detailed Description	48
4.24.2 Enumeration Type Documentation	48
4.24.2.1 State	48
4.24.3 Function Documentation	49
4.24.3.1 estado_configuracao()	49
4.24.3.2 estado_menu_principal()	49
4.24.3.3 estado_pagamento()	49
4.24.3.4 estado_selecao_produto()	49
4.25 states.h	50
4.26 ui.c File Reference	50
4.26.1 Detailed Description	51
4.26.2 Function Documentation	51
4.26.2.1 inicializar_interface()	51
4.26.2.2 limpar_tela()	51
4.26.2.3 pausar()	51

4.27 ui.h File Reference	51
4.27.1 Detailed Description	52
4.27.2 Function Documentation	52
4.27.2.1 inicializar_interface()	52
4.27.2.2 limpar_tela()	53
4.27.2.3 pausar()	53
4.28 ui.h	53
Index	55

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Contexto	Estrutura de contexto que armazena o estado global da aplicação	5
Fila	Estrutura para representar uma fila circular de propagandas	6
Log	Estrutura para representar um registro (log) de uma venda	7
Pilha	Estrutura para a pilha que armazena as moedas (valores float)	8
PilhaEstados	Estrutura para a pilha que armazena o histórico de estados da FSM	9
produto	Estrutura para representar um produto na máquina de refrigerantes	10

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

config.c	Implementação das funcionalidades do menu de configuração	13
config.h	Definições e protótipos para o módulo de configuração da máquina	14
estatisticas.c	Implementação das funções de geração de estatísticas de vendas	16
estatisticas.h	Protótipos para as funções de geração de estatísticas de vendas	18
fila.c	Implementação da estrutura de dados de fila circular	20
fila.h	Definições e protótipos para uma estrutura de dados de fila circular	22
lista_produtos.c	Implementação das funções de gerenciamento da lista de produtos	25
lista_produtos.h	Definições e protótipos para a lista duplamente encadeada de produtos	28
log_vendas.c	Implementação das funções de gerenciamento de logs de vendas	32
log_vendas.h	Definições e protótipos para a lista simplesmente encadeada de logs de vendas	35
main.c	Ponto de entrada principal e loop da aplicação da Máquina de Refrigerantes	38
pagamento.c	Implementação das funções de pagamento	39
pagamento.h	Protótipos para as funções relacionadas ao processo de pagamento	40
pilha.c	Implementação da estrutura de dados de pilha	42
pilha.h	Definições e protótipos para estruturas de dados de pilha	43
states.c	Implementação das funções de lógica para cada estado da FSM	45
states.h	Definição dos estados da máquina e protótipos das funções de estado	47
ui.c	Implementação das funções de interface de usuário	50
ui.h	Protótipos para funções de interface de usuário (UI) no terminal	51

Chapter 3

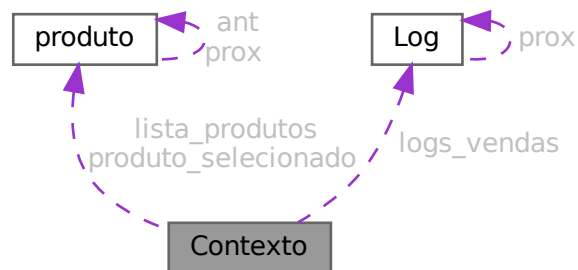
Class Documentation

3.1 Contexto Struct Reference

Estrutura de contexto que armazena o estado global da aplicação.

```
#include <config.h>
```

Collaboration diagram for Contexto:



Public Attributes

- [Produto](#) * [lista_produtos](#)
- [Log](#) * [logs_vendas](#)
- [Produto](#) * [produto_selecionado](#)

3.1.1 Detailed Description

Estrutura de contexto que armazena o estado global da aplicação.

Agrupar os ponteiros para as principais estruturas de dados utilizadas pela máquina de refrigerantes.

3.1.2 Member Data Documentation

3.1.2.1 lista_produtos

`Produto* Contexto::lista_produtos`

Ponteiro para a cabeça da lista de produtos.

3.1.2.2 logs_vendas

`Log* Contexto::logs_vendas`

Ponteiro para a cabeça da lista de logs de vendas.

3.1.2.3 produto_selecionado

`Produto* Contexto::produto_selecionado`

Ponteiro para o produto atualmente selecionado para compra.

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

- [config.h](#)

3.2 Fila Struct Reference

Estrutura para representar uma fila circular de propagandas.

```
#include <fila.h>
```

Public Attributes

- char [mensagens](#) [MAX_PROPAGANDAS][TAMANHO_MAX_MSG]
- int [frente](#)
- int [tras](#)
- int [tamanho](#)

3.2.1 Detailed Description

Estrutura para representar uma fila circular de propagandas.

3.2.2 Member Data Documentation

3.2.2.1 frente

`int Fila::frente`

Índice da frente da fila.

3.2.2.2 mensagens

```
char Fila::mensagens[MAX_PROPAGANDAS][TAMANHO_MAX_MSG]
```

Array de strings para as mensagens.

3.2.2.3 tamanho

```
int Fila::tamanho
```

Número de elementos na fila.

3.2.2.4 tras

```
int Fila::tras
```

Índice da traseira da fila.

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

- [fila.h](#)

3.3 Log Struct Reference

Estrutura para representar um registro (log) de uma venda.

```
#include <log_vendas.h>
```

Collaboration diagram for Log:



Public Attributes

- char [timestamp](#) [20]
- int [produto_id](#)
- float [valor_pago](#)
- float [troco](#)
- struct [Log](#) * [prox](#)

3.3.1 Detailed Description

Estrutura para representar um registro (log) de uma venda.

3.3.2 Member Data Documentation

3.3.2.1 produto_id

```
int Log::produto_id
```

ID do produto vendido.

3.3.2.2 prox

```
struct Log* Log::prox
```

Ponteiro para o próximo log na lista.

3.3.2.3 timestamp

```
char Log::timestamp[20]
```

Data e hora da transação.

3.3.2.4 troco

```
float Log::troco
```

Valor do troco devolvido.

3.3.2.5 valor_pago

```
float Log::valor_pago
```

Valor total pago pelo cliente.

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

- [log_vendas.h](#)

3.4 Pilha Struct Reference

Estrutura para a pilha que armazena as moedas (valores float).

```
#include <pilha.h>
```


Public Attributes

- float [valores](#) [MAX_PILHA]
- int [topo](#)

3.4.1 Detailed Description

Estrutura para a pilha que armazena as moedas (valores float).

3.4.2 Member Data Documentation

3.4.2.1 topo

```
int Pilha::topo
```

Índice do topo da pilha.

3.4.2.2 valores

```
float Pilha::valores[MAX_PILHA]
```

Array para armazenar os valores.

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

- [pilha.h](#)

3.5 PilhaEstados Struct Reference

Estrutura para a pilha que armazena o histórico de estados da FSM.

```
#include <pilha.h>
```

Public Attributes

- [State estados](#) [MAX_PILHA]
- int [topo](#)

3.5.1 Detailed Description

Estrutura para a pilha que armazena o histórico de estados da FSM.

3.5.2 Member Data Documentation

3.5.2.1 estados

`State PilhaEstados::estados[MAX_PILHA]`

Array para armazenar os estados.

3.5.2.2 topo

`int PilhaEstados::topo`

Índice do topo da pilha.

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

- [pilha.h](#)

3.6 produto Struct Reference

Estrutura para representar um produto na máquina de refrigerantes.

```
#include <lista_produtos.h>
```

Collaboration diagram for produto:



Public Attributes

- `int id`
- `char nome [50]`
- `float preco`
- `int estoque`
- `struct produto * prox`
- `struct produto * ant`

3.6.1 Detailed Description

Estrutura para representar um produto na máquina de refrigerantes.

3.6.2 Member Data Documentation

3.6.2.1 ant

```
struct produto* produto::ant
```

Ponteiro para o produto anterior na lista.

3.6.2.2 estoque

```
int produto::estoque
```

Quantidade disponível em estoque.

3.6.2.3 id

```
int produto::id
```

Identificador único do produto.

3.6.2.4 nome

```
char produto::nome[50]
```

Nome do produto.

3.6.2.5 preco

```
float produto::preco
```

Preço do produto em reais.

3.6.2.6 prox

```
struct produto* produto::prox
```

Ponteiro para o próximo produto na lista.

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

- [lista_produtos.h](#)

Chapter 4

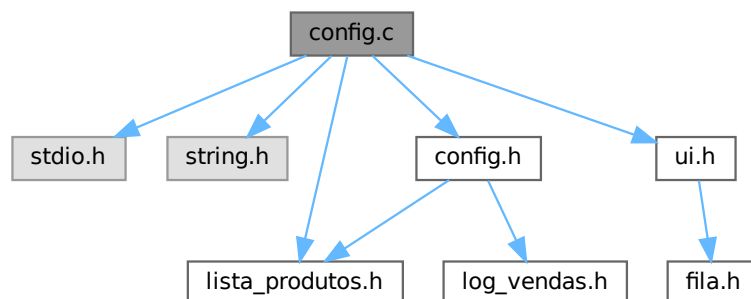
File Documentation

4.1 config.c File Reference

Implementação das funcionalidades do menu de configuração.

```
#include <stdio.h>
#include <string.h>
#include "config.h"
#include "lista_produtos.h"
#include "ui.h"
```

Include dependency graph for config.c:



Functions

- int [verificar_senha](#) ()
Solicita e verifica a senha do administrador.
- void [menu_configuracao](#) (Contexto *ctx)
Exibe e gerencia o menu de configuração.

4.1.1 Detailed Description

Implementação das funcionalidades do menu de configuração.

4.1.2 Function Documentation

4.1.2.1 menu_configuracao()

```
void menu_configuracao (
    Contexto * ctx )
```

Exibe e gerencia o menu de configuração.

Permite que o administrador adicione, remova e liste produtos. O acesso a este menu é protegido por senha.

Parameters

in, out	ctx	Ponteiro para o contexto da aplicação, que será modificado.
---------	-----	---

4.1.2.2 verificar_senha()

```
int verificar_senha ( )
```

Solicita e verifica a senha do administrador.

- Permite até 3 tentativas antes de bloquear o acesso.

Returns

1 se a senha estiver correta, 0 caso contrário.

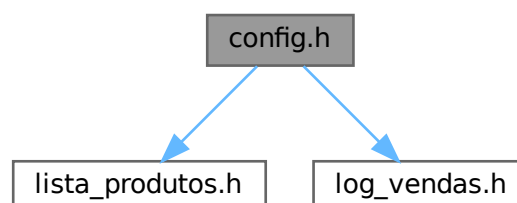
4.2 config.h File Reference

Definições e protótipos para o módulo de configuração da máquina.

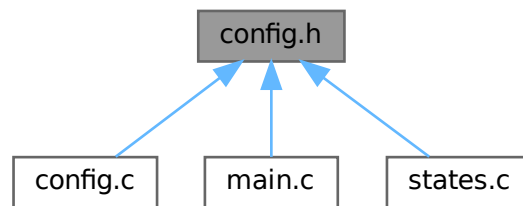
```
#include "lista_produtos.h"
```

```
#include "log_vendas.h"
```

Include dependency graph for config.h:



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



Classes

- struct [Contexto](#)
Estrutura de contexto que armazena o estado global da aplicação.

Macros

- `#define SENHA_PADRAO "admin123"`
Senha padrão para acesso ao menu de configuração.

Functions

- int [verificar_senha](#) ()
Solicita e verifica a senha do administrador.
- void [menu_configuracao](#) ([Contexto](#) *ctx)
Exibe e gerencia o menu de configuração.

4.2.1 Detailed Description

Definições e protótipos para o módulo de configuração da máquina.

Este arquivo contém a estrutura de contexto da aplicação, a senha de administrador e os protótipos das funções relacionadas ao menu de configuração.

4.2.2 Function Documentation

4.2.2.1 menu_configuracao()

```
void menu_configuracao (  
    Contexto * ctx )
```

Exibe e gerencia o menu de configuração.

Permite que o administrador adicione, remova e liste produtos. O acesso a este menu é protegido por senha.

Parameters

<code>in, out</code>	<code>ctx</code>	Ponteiro para o contexto da aplicação, que será modificado.
----------------------	------------------	---

4.2.2.2 verificar_senha()

```
int verificar_senha ( )
```

Solicita e verifica a senha do administrador.

- Permite até 3 tentativas antes de bloquear o acesso.

Returns

1 se a senha estiver correta, 0 caso contrário.

4.3 config.h

[Go to the documentation of this file.](#)

```
00001
00010 #ifndef CONFIG_H
00011 #define CONFIG_H
00012
00013 #include "lista_produtos.h"
00014 #include "log_vendas.h"
00015
00019 #define SENHA_PADRAO "admin123"
00020
00027 typedef struct {
00028     Produto* lista_produtos;
00029     Log* logs_vendas;
00030     Produto* produto_selecionado;
00031 } Contexto;
00032
00038 int verificar_senha();
00039
00047 void menu_configuracao(Contexto *ctx);
00048
00049 #endif
```

4.4 estatisticas.c File Reference

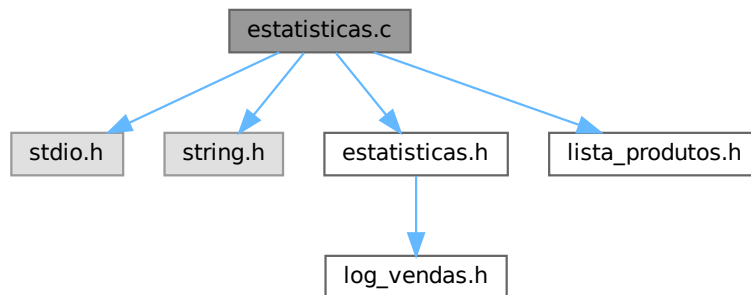
Implementação das funções de geração de estatísticas de vendas.

```
#include <stdio.h>
#include <string.h>
#include "estatisticas.h"
```



```
#include "lista_produtos.h"
```

Include dependency graph for estatisticas.c:



Functions

- void [gerar_estatisticas](#) ([Log](#) *logs)
Gera e exibe estatísticas com base nos logs de vendas.
- void [salvar_estatisticas_csv](#) ([Log](#) *logs, const char *filename)
Salva as estatísticas de vendas em um arquivo CSV.

4.4.1 Detailed Description

Implementação das funções de geração de estatísticas de vendas.

4.4.2 Function Documentation

4.4.2.1 gerar_estatisticas()

```
void gerar_estatisticas (
    Log * logs )
```

Gera e exibe estatísticas com base nos logs de vendas.

- Calcula o total de vendas, valor arrecadado, valor médio e o produto mais vendido. Também invoca a função para salvar as estatísticas em um arquivo.

Parameters

in	<i>logs</i>	Ponteiro para a cabeça da lista de logs de vendas.
----	-------------	--

Note

Atualmente, o nome do produto mais vendido é um placeholder. Uma futura implementação poderia usar o contexto para buscar o nome real do produto a partir da lista de produtos.

4.4.2.2 salvar_estatisticas_csv()

```
void salvar_estatisticas_csv (
    Log * logs,
    const char * filename )
```

Salva as estatísticas de vendas em um arquivo CSV.

Parameters

in	<i>logs</i>	Ponteiro para a cabeça da lista de logs de vendas.
in	<i>filename</i>	O nome do arquivo CSV onde as estatísticas serão salvas.

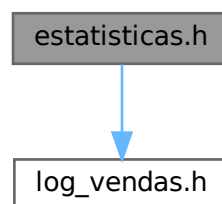
.

4.5 estatisticas.h File Reference

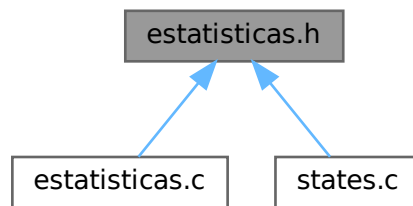
Protótipos para as funções de geração de estatísticas de vendas.

```
#include "log_vendas.h"
```

Include dependency graph for estatisticas.h:



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



Functions

- void [gerar_estatisticas](#) ([Log](#) *logs)
Gera e exibe estatísticas com base nos logs de vendas.
- void [salvar_estatisticas_csv](#) ([Log](#) *logs, const char *filename)
Salva as estatísticas de vendas em um arquivo CSV.

4.5.1 Detailed Description

Protótipos para as funções de geração de estatísticas de vendas.

4.5.2 Function Documentation

4.5.2.1 gerar_estatisticas()

```
void gerar_estatisticas (
    Log * logs )
```

Gera e exibe estatísticas com base nos logs de vendas.

- Calcula o total de vendas, valor arrecadado, valor médio e o produto mais vendido. Também invoca a função para salvar as estatísticas em um arquivo.

Parameters

in	logs	Ponteiro para a cabeça da lista de logs de vendas.
----	------	--

- Calcula o total de vendas, valor arrecadado, valor médio e o produto mais vendido. Também invoca a função para salvar as estatísticas em um arquivo.

Parameters

in	logs	Ponteiro para a cabeça da lista de logs de vendas.
----	------	--

Note

Atualmente, o nome do produto mais vendido é um placeholder. Uma futura implementação poderia usar o contexto para buscar o nome real do produto a partir da lista de produtos.

4.5.2.2 salvar_estatisticas_csv()

```
void salvar_estatisticas_csv (
    Log * logs,
    const char * filename )
```

Salva as estatísticas de vendas em um arquivo CSV.

Parameters

in	<i>logs</i>	Ponteiro para a cabeça da lista de logs de vendas.
in	<i>filename</i>	O nome do arquivo CSV onde as estatísticas serão salvas.

.

4.6 estatisticas.h

[Go to the documentation of this file.](#)

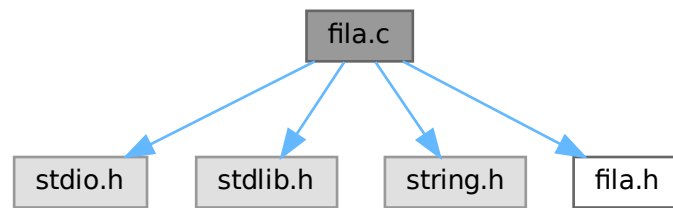
```
00001
00006 #ifndef ESTATISTICAS_H
00007 #define ESTATISTICAS_H
00008
00009 #include "log_vendas.h"
00010
00017 void gerar_estatisticas(Log* logs);
00018
00024 void salvar_estatisticas_csv(Log* logs, const char* filename);
00025
00026 #endif
```

4.7 fila.c File Reference

Implementação da estrutura de dados de fila circular.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "fila.h"
```

Include dependency graph for fila.c:



Functions

- void `inicializar_fila` (`Fila *fila`)
Inicializa a fila.
- void `enqueuear` (`Fila *fila`, `const char *mensagem`)
Adiciona uma mensagem ao final da fila (enqueuear).
- `const char *` `dequeuear` (`Fila *fila`)
Remove e retorna a mensagem da frente da fila (dequeuear).
- `const char *` `frente_fila` (`Fila *fila`)
Retorna a mensagem na frente da fila sem removê-la.

4.7.1 Detailed Description

Implementação da estrutura de dados de fila circular.

4.7.2 Function Documentation

4.7.2.1 dequeuear()

```
const char * dequeuear (  
    Fila * fila )
```

Remove e retorna a mensagem da frente da fila (dequeuear).

Parameters

<code>in, out</code>	<code>fila</code>	Ponteiro para a fila.
----------------------	-------------------	-----------------------

Returns

Ponteiro para a mensagem removida, ou NULL se a fila estiver vazia.

4.7.2.2 enfileirar()

```
void enfileirar (
    Fila * fila,
    const char * mensagem )
```

Adiciona uma mensagem ao final da fila (enfileirar).

Parameters

in, out	<i>fila</i>	Ponteiro para a fila.
in	<i>mensagem</i>	Mensagem a ser adicionada.

4.7.2.3 frente_fila()

```
const char * frente_fila (
    Fila * fila )
```

Retorna a mensagem na frente da fila sem removê-la.

Parameters

in	<i>fila</i>	Ponteiro para a fila.
----	-------------	-----------------------

Returns

Ponteiro para a mensagem da frente, ou NULL se a fila estiver vazia.

4.7.2.4 inicializar_fila()

```
void inicializar_fila (
    Fila * fila )
```

Inicializa a fila.

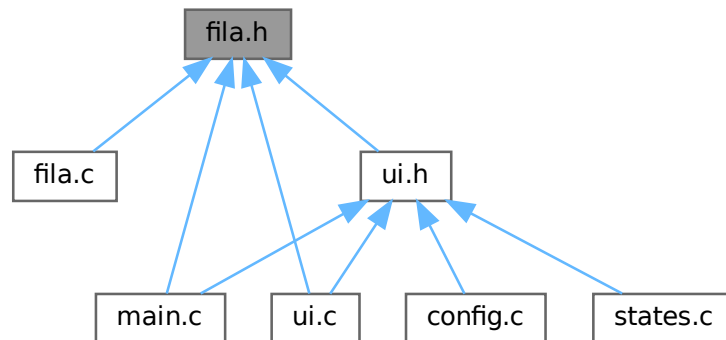
Parameters

out	<i>fila</i>	Ponteiro para a fila a ser inicializada.
-----	-------------	--

4.8 fila.h File Reference

Definições e protótipos para uma estrutura de dados de fila circular.

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



Classes

- struct [Fila](#)
Estrutura para representar uma fila circular de propagandas.

Macros

- #define **MAX_PROPAGANDAS** 10
- #define **TAMANHO_MAX_MSG** 100

Typedefs

- typedef struct [Fila](#) **Fila**
Estrutura para representar uma fila circular de propagandas.

Functions

- void [inicializar_fila](#) ([Fila](#) *fila)
Inicializa a fila.
- void [enfileirar](#) ([Fila](#) *fila, const char *mensagem)
Adiciona uma mensagem ao final da fila (enfileirar).
- const char * [desenfileirar](#) ([Fila](#) *fila)
Remove e retorna a mensagem da frente da fila (desenfileirar).
- const char * [frente_fila](#) ([Fila](#) *fila)
Retorna a mensagem na frente da fila sem removê-la.

4.8.1 Detailed Description

Definições e protótipos para uma estrutura de dados de fila circular.

- Usada para gerenciar as propagandas da máquina.

4.8.2 Function Documentation

4.8.2.1 desenfileirar()

```
const char * desenfileirar (  
    Fila * fila )
```

Remove e retorna a mensagem da frente da fila (desenfileirar).

Parameters

in, out	<i>fila</i>	Ponteiro para a fila.
---------	-------------	-----------------------

Returns

Ponteiro para a mensagem removida, ou NULL se a fila estiver vazia.

4.8.2.2 enfileirar()

```
void enfileirar (  
    Fila * fila,  
    const char * mensagem )
```

Adiciona uma mensagem ao final da fila (enfileirar).

Parameters

in, out	<i>fila</i>	Ponteiro para a fila.
in	<i>mensagem</i>	Mensagem a ser adicionada.

4.8.2.3 frente_fila()

```
const char * frente_fila (  
    Fila * fila )
```

Retorna a mensagem na frente da fila sem removê-la.

Parameters

in	<i>fila</i>	Ponteiro para a fila.
----	-------------	-----------------------

Returns

Ponteiro para a mensagem da frente, ou NULL se a fila estiver vazia.

4.8.2.4 inicializar_fila()

```
void inicializar_fila (
    Fila * fila )
```

Inicializa a fila.

Parameters

out	<i>fila</i>	Ponteiro para a fila a ser inicializada.
-----	-------------	--

4.9 fila.h

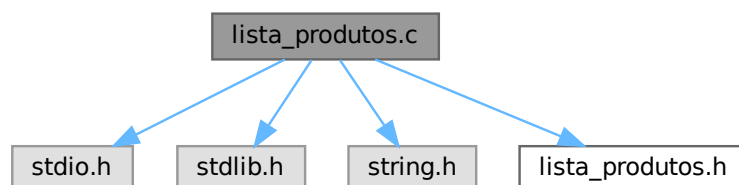
[Go to the documentation of this file.](#)

```
00001
00007 #ifndef FILA_H
00008 #define FILA_H
00009
00010 #define MAX_PROPAGANDAS 10
00011 #define TAMANHO_MAX_MSG 100
00012
00016 typedef struct Fila {
00017     char mensagens[MAX_PROPAGANDAS][TAMANHO_MAX_MSG];
00018     int frente;
00019     int tras;
00020     int tamanho;
00021 } Fila;
00022
00027 void inicializar_fila(Fila* fila);
00028
00034 void enfileirar(Fila* fila, const char* mensagem);
00035
00041 const char* desenfileirar(Fila* fila);
00042
00048 const char* frente_fila(Fila* fila);
00049
00050 #endif
```

4.10 lista_produtos.c File Reference

Implementação das funções de gerenciamento da lista de produtos.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "lista_produtos.h"
Include dependency graph for lista_produtos.c:
```



Functions

- `Produto * adicionar_produto (Produto *head, int id, const char *nome, float preco, int estoque)`
Adiciona um novo produto ao início da lista duplamente encadeada.
- `Produto * remover_produto (Produto *head, int id)`
Remove um produto da lista pelo seu ID.
- `Produto * buscar_produto (Produto *head, int id)`
Busca um produto na lista pelo seu ID.
- `void listar_produtos (Produto *head)`
Lista todos os produtos da lista no terminal.
- `void liberar_lista_produtos (Produto *head)`
Libera toda a memória alocada para a lista de produtos.
- `void salvar_produtos_csv (Produto *head, const char *filename)`
Salva a lista de produtos em um arquivo no formato CSV.
- `Produto * carregar_produtos_csv (const char *filename)`
Carrega uma lista de produtos a partir de um arquivo CSV.

4.10.1 Detailed Description

Implementação das funções de gerenciamento da lista de produtos.

4.10.2 Function Documentation

4.10.2.1 adicionar_produto()

```
Produto * adicionar_produto (
    Produto * head,
    int id,
    const char * nome,
    float preco,
    int estoque )
```

Adiciona um novo produto ao início da lista duplamente encadeada.

Parameters

in	<i>head</i>	Ponteiro para a cabeça da lista.
in	<i>id</i>	ID do novo produto.
in	<i>nome</i>	Nome do novo produto.
in	<i>preco</i>	Preço do novo produto.
in	<i>estoque</i>	Quantidade em estoque do novo produto.

Returns

Ponteiro para a nova cabeça da lista.

4.10.2.2 buscar_produto()

```
Produto * buscar_produto (
```

```
Produto * head,  
int id )
```

Busca um produto na lista pelo seu ID.

Parameters

in	<i>head</i>	Ponteiro para a cabeça da lista.
in	<i>id</i>	ID do produto a ser buscado.

Returns

Ponteiro para o nó do produto encontrado, ou NULL se não for encontrado.

4.10.2.3 carregar_produtos_csv()

```
Produto * carregar_produtos_csv (  
    const char * filename )
```

Carrega uma lista de produtos a partir de um arquivo CSV.

Parameters

in	<i>filename</i>	Nome do arquivo CSV a ser lido.
----	-----------------	---------------------------------

Returns

Ponteiro para a cabeça da nova lista de produtos carregada.

4.10.2.4 liberar_lista_produtos()

```
void liberar_lista_produtos (  
    Produto * head )
```

Libera toda a memória alocada para a lista de produtos.

Parameters

in	<i>head</i>	Ponteiro para a cabeça da lista a ser liberada.
----	-------------	---

4.10.2.5 listar_produtos()

```
void listar_produtos (  
    Produto * head )
```

Lista todos os produtos da lista no terminal.

Parameters

in	<i>head</i>	Ponteiro para a cabeça da lista.
----	-------------	----------------------------------

4.10.2.6 remover_produto()

```
Produto * remover_produto (
    Produto * head,
    int id )
```

Remove um produto da lista pelo seu ID.

Parameters

in	<i>head</i>	Ponteiro para a cabeça da lista.
in	<i>id</i>	ID do produto a ser removido.

Returns

Ponteiro para a nova cabeça da lista.

4.10.2.7 salvar_produtos_csv()

```
void salvar_produtos_csv (
    Produto * head,
    const char * filename )
```

Salva a lista de produtos em um arquivo no formato CSV.

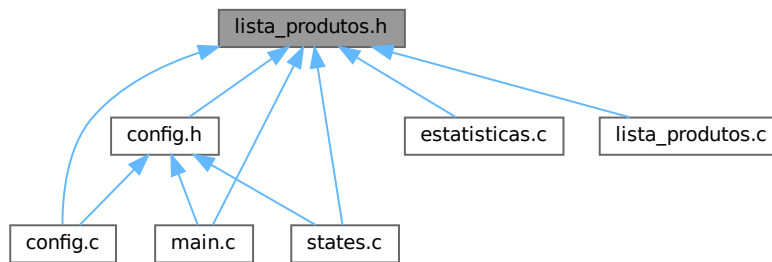
Parameters

in	<i>head</i>	Ponteiro para a cabeça da lista.
in	<i>filename</i>	Nome do arquivo CSV de destino.

4.11 lista_produtos.h File Reference

Definições e protótipos para a lista duplamente encadeada de produtos.

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



Classes

- struct [produto](#)

Estrutura para representar um produto na máquina de refrigerantes.

Typedefs

- typedef struct [produto](#) **Produto**

Estrutura para representar um produto na máquina de refrigerantes.

Functions

- [Produto](#) * [adicionar_produto](#) ([Produto](#) *head, int id, const char *nome, float preco, int estoque)
Adiciona um novo produto ao início da lista duplamente encadeada.
- [Produto](#) * [remover_produto](#) ([Produto](#) *head, int id)
Remove um produto da lista pelo seu ID.
- [Produto](#) * [buscar_produto](#) ([Produto](#) *head, int id)
Busca um produto na lista pelo seu ID.
- void [listar_produtos](#) ([Produto](#) *head)
Lista todos os produtos da lista no terminal.
- void [liberar_lista_produtos](#) ([Produto](#) *head)
Libera toda a memória alocada para a lista de produtos.
- void [salvar_produtos_csv](#) ([Produto](#) *head, const char *filename)
Salva a lista de produtos em um arquivo no formato CSV.
- [Produto](#) * [carregar_produtos_csv](#) (const char *filename)
Carrega uma lista de produtos a partir de um arquivo CSV.

4.11.1 Detailed Description

Definições e protótipos para a lista duplamente encadeada de produtos.

4.11.2 Function Documentation

4.11.2.1 adicionar_produto()

```
Produto * adicionar_produto (
    Produto * head,
    int id,
    const char * nome,
    float preco,
    int estoque )
```

Adiciona um novo produto ao início da lista duplamente encadeada.

Parameters

in	<i>head</i>	Ponteiro para a cabeça da lista.
in	<i>id</i>	ID do novo produto.
in	<i>nome</i>	Nome do novo produto.
in	<i>preco</i>	Preço do novo produto.
in	<i>estoque</i>	Quantidade em estoque do novo produto.

Returns

Ponteiro para a nova cabeça da lista.

4.11.2.2 buscar_produto()

```
Produto * buscar_produto (
    Produto * head,
    int id )
```

Busca um produto na lista pelo seu ID.

Parameters

in	<i>head</i>	Ponteiro para a cabeça da lista.
in	<i>id</i>	ID do produto a ser buscado.

Returns

Ponteiro para o nó do produto encontrado, ou NULL se não for encontrado.

4.11.2.3 carregar_produtos_csv()

```
Produto * carregar_produtos_csv (
    const char * filename )
```

Carrega uma lista de produtos a partir de um arquivo CSV.

Parameters

in	<i>filename</i>	Nome do arquivo CSV a ser lido.
----	-----------------	---------------------------------

Returns

Ponteiro para a cabeça da nova lista de produtos carregada.

4.11.2.4 liberar_lista_produtos()

```
void liberar_lista_produtos (
    Produto * head )
```

Libera toda a memória alocada para a lista de produtos.

Parameters

in	<i>head</i>	Ponteiro para a cabeça da lista a ser liberada.
----	-------------	---

4.11.2.5 listar_produtos()

```
void listar_produtos (
    Produto * head )
```

Lista todos os produtos da lista no terminal.

Parameters

in	<i>head</i>	Ponteiro para a cabeça da lista.
----	-------------	----------------------------------

4.11.2.6 remover_produto()

```
Produto * remover_produto (
    Produto * head,
    int id )
```

Remove um produto da lista pelo seu ID.

Parameters

in	<i>head</i>	Ponteiro para a cabeça da lista.
in	<i>id</i>	ID do produto a ser removido.

Returns

Ponteiro para a nova cabeça da lista.

4.11.2.7 salvar_produtos_csv()

```
void salvar_produtos_csv (
    Produto * head,
    const char * filename )
```

Salva a lista de produtos em um arquivo no formato CSV.

Parameters

in	<i>head</i>	Ponteiro para a cabeça da lista.
in	<i>filename</i>	Nome do arquivo CSV de destino.

4.12 lista_produtos.h

[Go to the documentation of this file.](#)

```
00001
00006 #ifndef LISTA_PRODUTOS_H
00007 #define LISTA_PRODUTOS_H
00008
00012 typedef struct produto {
00013     int id;
00014     char nome[50];
00015     float preco;
00016     int estoque;
00017     struct produto* prox;
00018     struct produto* ant;
00019 } Produto;
00020
00030 Produto* adicionar_produto(Produto* head, int id, const char* nome, float preco, int estoque);
00031
00038 Produto* remover_produto(Produto* head, int id);
00039
00046 Produto* buscar_produto(Produto* head, int id);
00047
00052 void listar_produtos(Produto* head);
00053
00058 void liberar_lista_produtos(Produto* head);
00059
00065 void salvar_produtos_csv(Produto* head, const char* filename);
00066
00072 Produto* carregar_produtos_csv(const char* filename);
00073
00074 #endif
```

4.13 log_vendas.c File Reference

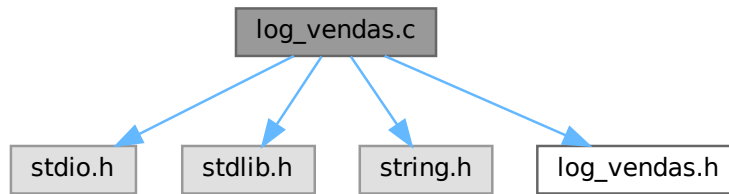
Implementação das funções de gerenciamento de logs de vendas.

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



```
#include "log_vendas.h"
```

Include dependency graph for log_vendas.c:



Functions

- `Log * adicionar_log (Log *head, const char *timestamp, int produto_id, float valor_pago, float troco)`
Adiciona um novo log de venda ao início da lista.
- `void salvar_logs (const char *filename, Log *head)`
Salva a lista de logs em um arquivo binário.
- `Log * carregar_logs (const char *filename)`
Carrega a lista de logs a partir de um arquivo binário.
- `void liberar_logs (Log *head)`
Libera toda a memória alocada para a lista de logs.
- `void exibir_logs (Log *head)`
Exibe todos os logs de vendas no terminal.

4.13.1 Detailed Description

Implementação das funções de gerenciamento de logs de vendas.

4.13.2 Function Documentation

4.13.2.1 adicionar_log()

```
Log * adicionar_log (
    Log * head,
    const char * timestamp,
    int produto_id,
    float valor_pago,
    float troco )
```

Adiciona um novo log de venda ao início da lista.

Parameters

in	<i>head</i>	Ponteiro para a cabeça da lista de logs.
in	<i>timestamp</i>	String com a data e hora da venda.
in	<i>produto_id</i>	ID do produto vendido.
in	<i>valor_pago</i>	Valor inserido pelo cliente.
in	<i>troco</i>	Valor do troco.

Returns

Ponteiro para a nova cabeça da lista de logs.

4.13.2.2 carregar_logs()

```
Log * carregar_logs (
    const char * filename )
```

Carrega a lista de logs a partir de um arquivo binário.

Parameters

in	<i>filename</i>	Nome do arquivo a ser lido.
----	-----------------	-----------------------------

Returns

Ponteiro para a cabeça da nova lista de logs carregada.

4.13.2.3 exibir_logs()

```
void exibir_logs (
    Log * head )
```

Exibe todos os logs de vendas no terminal.

Parameters

in	<i>head</i>	Ponteiro para a cabeça da lista de logs.
----	-------------	--

4.13.2.4 liberar_logs()

```
void liberar_logs (
    Log * head )
```

Libera toda a memória alocada para a lista de logs.

Parameters

in	<i>head</i>	Ponteiro para a cabeça da lista a ser liberada.
----	-------------	---

4.13.2.5 salvar_logs()

```
void salvar_logs (
    const char * filename,
    Log * head )
```

Salva a lista de logs em um arquivo binário.

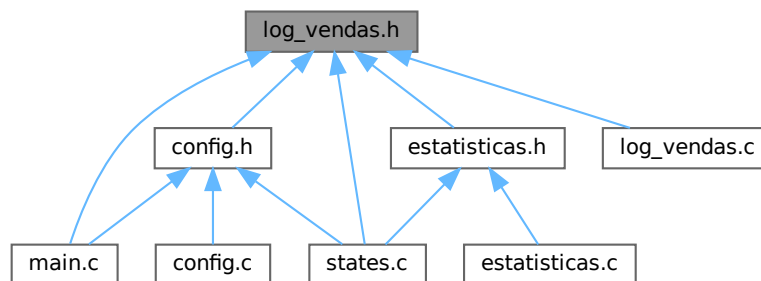
Parameters

in	<i>filename</i>	Nome do arquivo de destino.
in	<i>head</i>	Ponteiro para a cabeça da lista de logs.

4.14 log_vendas.h File Reference

Definições e protótipos para a lista simplesmente encadeada de logs de vendas.

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



Classes

- struct [Log](#)
Estrutura para representar um registro (log) de uma venda.

Typedefs

- typedef struct [Log](#) **Log**
Estrutura para representar um registro (log) de uma venda.

Functions

- [Log](#) * [adicionar_log](#) ([Log](#) *head, const char *timestamp, int produto_id, float valor_pago, float troco)
Adiciona um novo log de venda ao início da lista.
- void [salvar_logs](#) (const char *filename, [Log](#) *head)
Salva a lista de logs em um arquivo binário.
- [Log](#) * [carregar_logs](#) (const char *filename)
Carrega a lista de logs a partir de um arquivo binário.
- void [liberar_logs](#) ([Log](#) *head)
Libera toda a memória alocada para a lista de logs.
- void [exibir_logs](#) ([Log](#) *head)
Exibe todos os logs de vendas no terminal.

4.14.1 Detailed Description

Definições e protótipos para a lista simplesmente encadeada de logs de vendas.

4.14.2 Function Documentation

4.14.2.1 adicionar_log()

```
Log * adicionar_log (
    Log * head,
    const char * timestamp,
    int produto_id,
    float valor_pago,
    float troco )
```

Adiciona um novo log de venda ao início da lista.

Parameters

in	<i>head</i>	Ponteiro para a cabeça da lista de logs.
in	<i>timestamp</i>	String com a data e hora da venda.
in	<i>produto_id</i>	ID do produto vendido.
in	<i>valor_pago</i>	Valor inserido pelo cliente.
in	<i>troco</i>	Valor do troco.

Returns

Ponteiro para a nova cabeça da lista de logs.

4.14.2.2 carregar_logs()

```
Log * carregar_logs (
    const char * filename )
```

Carrega a lista de logs a partir de um arquivo binário.

Parameters

in	<i>filename</i>	Nome do arquivo a ser lido.
----	-----------------	-----------------------------

Returns

Ponteiro para a cabeça da nova lista de logs carregada.

4.14.2.3 exibir_logs()

```
void exibir_logs (
    Log * head )
```

Exibe todos os logs de vendas no terminal.

Parameters

in	<i>head</i>	Ponteiro para a cabeça da lista de logs.
----	-------------	--

4.14.2.4 liberar_logs()

```
void liberar_logs (  
    Log * head )
```

Libera toda a memória alocada para a lista de logs.

Parameters

in	<i>head</i>	Ponteiro para a cabeça da lista a ser liberada.
----	-------------	---

4.14.2.5 salvar_logs()

```
void salvar_logs (  
    const char * filename,  
    Log * head )
```

Salva a lista de logs em um arquivo binário.

Parameters

in	<i>filename</i>	Nome do arquivo de destino.
in	<i>head</i>	Ponteiro para a cabeça da lista de logs.

4.15 log_vendas.h

[Go to the documentation of this file.](#)

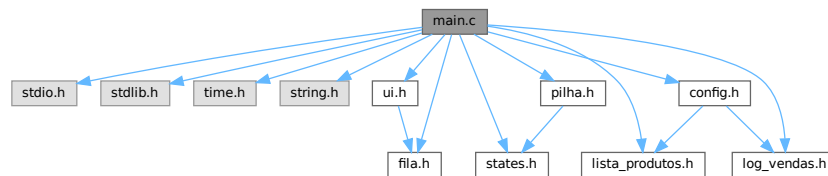
```
00001  
00006 #ifndef LOG_VENDAS_H  
00007 #define LOG_VENDAS_H  
00008  
00012 typedef struct Log {  
00013     char timestamp[20];  
00014     int produto_id;  
00015     float valor_pago;  
00016     float troco;  
00017     struct Log* prox;  
00018 } Log;  
00019  
00029 Log* adicionar_log(Log* head, const char* timestamp, int produto_id, float valor_pago, float troco);  
00030  
00036 void salvar_logs(const char* filename, Log* head);  
00037  
00043 Log* carregar_logs(const char* filename);  
00044  
00049 void liberar_logs(Log* head);  
00050  
00055 void exibir_logs(Log* head);  
00056  
00057 #endif
```

4.16 main.c File Reference

Ponto de entrada principal e loop da aplicação da Máquina de Refrigerantes.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include "ui.h"
#include "states.h"
#include "lista_produtos.h"
#include "log_vendas.h"
#include "fila.h"
#include "pilha.h"
#include "config.h"
```

Include dependency graph for main.c:



Functions

- int [main](#) ()
Função principal que executa a máquina de refrigerantes.

4.16.1 Detailed Description

Ponto de entrada principal e loop da aplicação da Máquina de Refrigerantes.

- Este arquivo inicializa todas as estruturas de dados, gerencia a máquina de estados finitos (FSM) e controla o fluxo principal da aplicação.

4.16.2 Function Documentation

4.16.2.1 main()

```
int main ( )
```

Função principal que executa a máquina de refrigerantes.

- Inicializa o contexto, as estruturas de dados, carrega os dados de arquivos e entra no loop da máquina de estados. Ao sair, salva os dados e libera a memória alocada.

Returns

0 em caso de sucesso, 1 em caso de erro de alocação.

4.17 pagamento.c File Reference

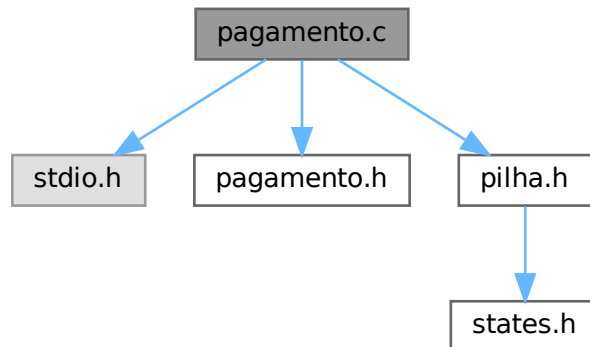
Implementação das funções de pagamento.

```
#include <stdio.h>
```

```
#include "pagamento.h"
```

```
#include "pilha.h"
```

Include dependency graph for pagamento.c:



Functions

- int `inserir_moedas` (float *valor_acumulado)
Coleta as moedas inseridas pelo usuário.
- float `calcular_troco` (float valor_inserido, float preco)
Calcula o troco com base no valor inserido e no preço do produto.
- float `desfazer_moeda` ()
Desfaz a inserção da última moeda, usando a pilha de moedas.

4.17.1 Detailed Description

Implementação das funções de pagamento.

4.17.2 Function Documentation

4.17.2.1 `calcular_troco()`

```
float calcular_troco (  
    float valor_inserido,  
    float preco )
```

Calcula o troco com base no valor inserido e no preço do produto.

Parameters

in	<i>valor_inserido</i>	Valor total inserido pelo usuário.
in	<i>preco</i>	Preço do produto.

Returns

Valor do troco, ou um valor negativo se o pagamento for insuficiente.

4.17.2.2 desfazer_moeda()

```
float desfazer_moeda ( )
```

Desfaz a inserção da última moeda, usando a pilha de moedas.

Returns

O valor da moeda removida, ou 0 se não houver moedas para remover.

4.17.2.3 inserir_moedas()

```
int inserir_moedas (
    float * valor_acumulado )
```

Coleta as moedas inseridas pelo usuário.

- A função usa um ponteiro para retornar o valor total acumulado. O valor de retorno da função indica o status da operação.

Parameters

out	<i>valor_acumulado</i>	Ponteiro para a variável que armazenará o total inserido.
-----	------------------------	---

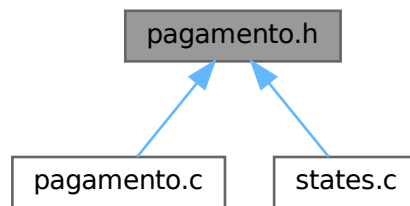
Returns

1 se a operação foi concluída com sucesso (usuário digitou 0).
0 se a operação foi cancelada (usuário digitou -2).

4.18 pagamento.h File Reference

Protótipos para as funções relacionadas ao processo de pagamento.

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



Functions

- int `inserir_moedas` (float *valor_acumulado)
Coleta as moedas inseridas pelo usuário.
- float `calcular_troco` (float valor_inserido, float preco)
Calcula o troco com base no valor inserido e no preço do produto.
- float `desfazer_moeda` ()
Desfaz a inserção da última moeda, usando a pilha de moedas.

4.18.1 Detailed Description

Protótipos para as funções relacionadas ao processo de pagamento.

4.18.2 Function Documentation

4.18.2.1 `calcular_troco()`

```
float calcular_troco (  
    float valor_inserido,  
    float preco )
```

Calcula o troco com base no valor inserido e no preço do produto.

Parameters

in	<i>valor_inserido</i>	Valor total inserido pelo usuário.
in	<i>preco</i>	Preço do produto.

Returns

Valor do troco, ou um valor negativo se o pagamento for insuficiente.

4.18.2.2 desfazer_moeda()

```
float desfazer_moeda ( )
```

Desfaz a inserção da última moeda, usando a pilha de moedas.

Returns

O valor da moeda removida, ou 0 se não houver moedas para remover.

4.18.2.3 inserir_moedas()

```
int inserir_moedas (
    float * valor_acumulado )
```

Coleta as moedas inseridas pelo usuário.

- A função usa um ponteiro para retornar o valor total acumulado. O valor de retorno da função indica o status da operação.

Parameters

out	valor_acumulado	Ponteiro para a variável que armazenará o total inserido.
-----	-----------------	---

Returns

- 1 se a operação foi concluída com sucesso (usuário digitou 0).
- 0 se a operação foi cancelada (usuário digitou -2).

4.19 pagamento.h

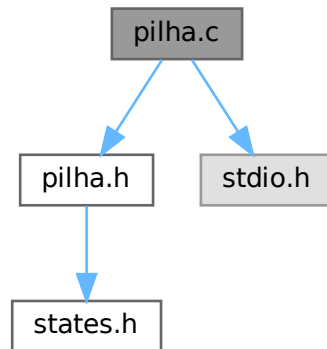
[Go to the documentation of this file.](#)

```
00001
00006 #ifndef PAGAMENTO_H
00007 #define PAGAMENTO_H
00008
00017 int inserir_moedas(float *valor_acumulado);
00018
00025 float calcular_troco(float valor_inserido, float preco);
00026
00031 float desfazer_moeda();
00032
00033 #endif
```

4.20 pilha.c File Reference

Implementação da estrutura de dados de pilha.

```
#include "pilha.h"
#include <stdio.h>
Include dependency graph for pilha.c:
```



Functions

- void `inicializar_pilha` ([Pilha](#) *`pilha`)
- int `pilha_vazia` ([Pilha](#) *`pilha`)
- int `pilha_cheia` ([Pilha](#) *`pilha`)
- void `empilhar` ([Pilha](#) *`pilha`, float `valor`)
- float `desempilhar` ([Pilha](#) *`pilha`)
- void `inicializar_pilha_estados` ([PilhaEstados](#) *`pilha`)
- int `pilha_estados_vazia` ([PilhaEstados](#) *`pilha`)
- int `pilha_estados_cheia` ([PilhaEstados](#) *`pilha`)
- void `empilhar_estado` ([PilhaEstados](#) *`pilha`, [State](#) `estado`)
- [State](#) `desempilhar_estado` ([PilhaEstados](#) *`pilha`)

4.20.1 Detailed Description

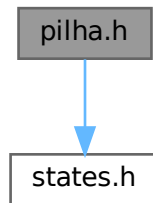
Implementação da estrutura de dados de pilha.

4.21 pilha.h File Reference

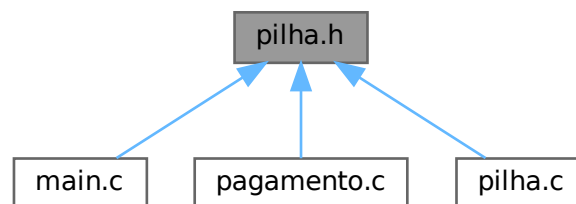
Definições e protótipos para estruturas de dados de pilha.

```
#include "states.h"
```

Include dependency graph for pilha.h:



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



Classes

- struct [Pilha](#)
Estrutura para a pilha que armazena as moedas (valores float).
- struct [PilhaEstados](#)
Estrutura para a pilha que armazena o histórico de estados da FSM.

Macros

- `#define MAX_PILHA 100`

Functions

- void `inicializar_pilha` ([Pilha](#) *pilha)
- int `pilha_vazia` ([Pilha](#) *pilha)
- int `pilha_cheia` ([Pilha](#) *pilha)
- void `empilhar` ([Pilha](#) *pilha, float valor)
- float `desempilhar` ([Pilha](#) *pilha)
- void `inicializar_pilha_estados` ([PilhaEstados](#) *pilha)
- int `pilha_estados_vazia` ([PilhaEstados](#) *pilha)
- int `pilha_estados_cheia` ([PilhaEstados](#) *pilha)
- void `empilhar_estado` ([PilhaEstados](#) *pilha, [State](#) estado)
- [State](#) `desempilhar_estado` ([PilhaEstados](#) *pilha)

4.21.1 Detailed Description

Definições e protótipos para estruturas de dados de pilha.

- Contém implementações para uma pilha de floats (usada para moedas) e uma pilha de estados (usada para o histórico de navegação).

4.22 pilha.h

[Go to the documentation of this file.](#)

```

00001
00008 #ifndef PILHA_H
00009 #define PILHA_H
00010
00011 #include "states.h"
00012
00013 #define MAX_PILHA 100
00014
00018 typedef struct {
00019     float valores[MAX_PILHA];
00020     int topo;
00021 } Pilha;
00022
00023 void inicializar_pilha(Pilha *pilha);
00024 int pilha_vazia(Pilha *pilha);
00025 int pilha_cheia(Pilha *pilha);
00026 void empilhar(Pilha *pilha, float valor);
00027 float desempilhar(Pilha *pilha);
00028
00032 typedef struct {
00033     State estados[MAX_PILHA];
00034     int topo;
00035 } PilhaEstados;
00036
00037 void inicializar_pilha_estados(PilhaEstados *pilha);
00038 int pilha_estados_vazia(PilhaEstados *pilha);
00039 int pilha_estados_cheia(PilhaEstados *pilha);
00040 void empilhar_estado(PilhaEstados *pilha, State estado);
00041 State desempilhar_estado(PilhaEstados *pilha);
00042
00043 #endif

```

4.23 states.c File Reference

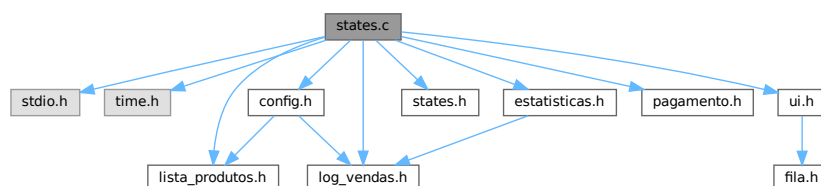
Implementação das funções de lógica para cada estado da FSM.

```

#include <stdio.h>
#include <time.h>
#include "config.h"
#include "states.h"
#include "ui.h"
#include "lista_produtos.h"
#include "pagamento.h"
#include "log_vendas.h"
#include "estatisticas.h"

```

Include dependency graph for states.c:



Functions

- [State estado_menu_principal](#) ([Contexto](#) *ctx)
Lógica do estado do menu principal.
- [State estado_selecao_produto](#) ([Contexto](#) *ctx)
Lógica do estado de seleção de produto.
- [State estado_pagamento](#) ([Contexto](#) *ctx)
Lógica do estado de pagamento.
- [State estado_configuracao](#) ([Contexto](#) *ctx)
Lógica do estado de configuração.

4.23.1 Detailed Description

Implementação das funções de lógica para cada estado da FSM.

4.23.2 Function Documentation

4.23.2.1 estado_configuracao()

```
State estado_configuracao (
    Contexto * ctx )
```

Lógica do estado de configuração.

Parameters

in, out	ctx	Contexto da aplicação, pode ser modificado pelo menu de configuração.
---------	-----	---

Returns

Próximo estado da FSM (sempre MENU_PRINCIPAL ao sair do menu de config).

4.23.2.2 estado_menu_principal()

```
State estado_menu_principal (
    Contexto * ctx )
```

Lógica do estado do menu principal.

Parameters

in	ctx	Contexto da aplicação (usado para acessar logs).
----	-----	--

Returns

Próximo estado da FSM.

4.23.2.3 estado_pagamento()

```
State estado_pagamento (
    Contexto * ctx )
```

Lógica do estado de pagamento.

Parameters

in, out	ctx	Contexto da aplicação, usado para obter o produto, decrementar estoque e registrar o log.
---------	-----	---

Returns

Próximo estado da FSM.

4.23.2.4 estado_selecao_produto()

```
State estado_selecao_produto (
    Contexto * ctx )
```

Lógica do estado de seleção de produto.

Parameters

in, out	ctx	Contexto da aplicação, o produto selecionado será armazenado aqui.
---------	-----	--

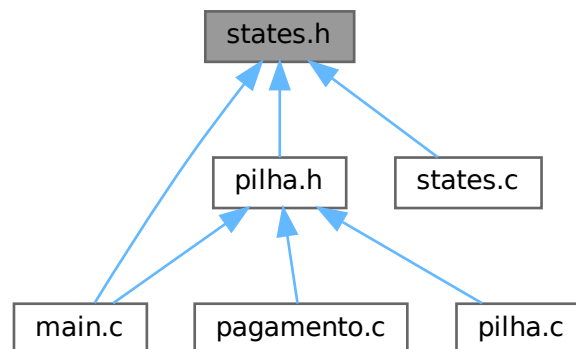
Returns

Próximo estado da FSM.

4.24 states.h File Reference

Definição dos estados da máquina e protótipos das funções de estado.

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



Enumerations

- enum `State` {
`MENU_PRINCIPAL` , `SELECAO_PRODUTO` , `PAGAMENTO` , `CONFIGURACAO` ,
`SAIR` , `VOLTAR` }

Enumeração dos possíveis estados da máquina de estados finitos (FSM).

Functions

- `State estado_menu_principal ()`
Função que implementa a lógica do estado MENU_PRINCIPAL.
- `State estado_selecao_produto ()`
Função que implementa a lógica do estado SELECAO_PRODUTO.
- `State estado_pagamento ()`
Função que implementa a lógica do estado PAGAMENTO.
- `State estado_configuracao ()`
Função que implementa a lógica do estado CONFIGURACAO.

4.24.1 Detailed Description

Definição dos estados da máquina e protótipos das funções de estado.

4.24.2 Enumeration Type Documentation

4.24.2.1 State

enum `State`

Enumeração dos possíveis estados da máquina de estados finitos (FSM).

Enumerator

MENU_PRINCIPAL	Estado do menu principal.
SELECAO_PRODUTO	Estado de seleção de produto.
PAGAMENTO	Estado de pagamento.
CONFIGURACAO	Estado do menu de configuração.
SAIR	Estado final para encerrar a aplicação.
VOLTAR	Estado especial para acionar a funcionalidade "voltar".

4.24.3 Function Documentation

4.24.3.1 estado_configuracao()

`State estado_configuracao ()`

Função que implementa a lógica do estado CONFIGURACAO.

Returns

O próximo estado para o qual a máquina deve transitar.

4.24.3.2 estado_menu_principal()

`State estado_menu_principal ()`

Função que implementa a lógica do estado MENU_PRINCIPAL.

Returns

O próximo estado para o qual a máquina deve transitar.

4.24.3.3 estado_pagamento()

`State estado_pagamento ()`

Função que implementa a lógica do estado PAGAMENTO.

Returns

O próximo estado para o qual a máquina deve transitar.

4.24.3.4 estado_selecao_produto()

`State estado_selecao_produto ()`

Função que implementa a lógica do estado SELECAO_PRODUTO.

Returns

O próximo estado para o qual a máquina deve transitar.

4.25 states.h

[Go to the documentation of this file.](#)

```

00001
00006 #ifndef STATES_H
00007 #define STATES_H
00008
00012 typedef enum {
00013     MENU_PRINCIPAL,
00014     SELECAO_PRODUTO,
00015     PAGAMENTO,
00016     CONFIGURACAO,
00017     SAIR,
00018     VOLTAR
00019 } State;
00020
00025 State estado_menu_principal();
00026
00031 State estado_selecao_produto();
00032
00037 State estado_pagamento();
00038
00043 State estado_configuracao();
00044
00045 #endif

```

4.26 ui.c File Reference

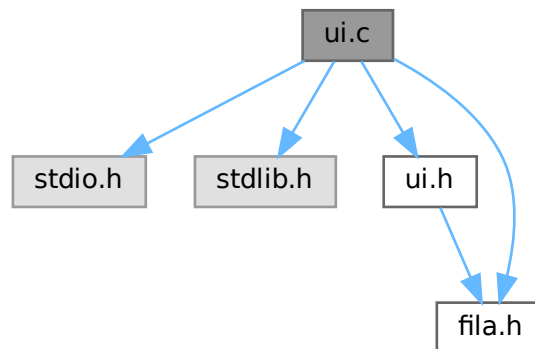
Implementação das funções de interface de usuário.

```

#include <stdio.h>
#include <stdlib.h>
#include "ui.h"
#include "fila.h"

```

Include dependency graph for ui.c:



Functions

- void [limpar_tela](#) ()
Limpa a tela do terminal.
- void [pausar](#) ()
Pausa a execução do programa até que o usuário pressione Enter.
- void [inicializar_interface](#) (Fila *propagandas)
Exibe a tela de boas-vindas da aplicação.

4.26.1 Detailed Description

Implementação das funções de interface de usuário.

4.26.2 Function Documentation

4.26.2.1 inicializar_interface()

```
void inicializar_interface (
    Fila * propagandas )
```

Exibe a tela de boas-vindas da aplicação.

Parameters

in	<i>propagandas</i>	Ponteiro para a fila de propagandas, para exibir a primeira.
----	--------------------	--

4.26.2.2 limpar_tela()

```
void limpar_tela ( )
```

Limpa a tela do terminal.

- Tenta usar "clear" (Linux/macOS) ou "cls" (Windows).

4.26.2.3 pausar()

```
void pausar ( )
```

Pausa a execução do programa até que o usuário pressione Enter.

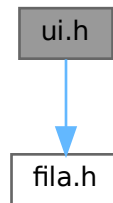
- Esta versão corrigida consome quaisquer caracteres residuais no buffer de entrada e espera por um único Enter para continuar.

4.27 ui.h File Reference

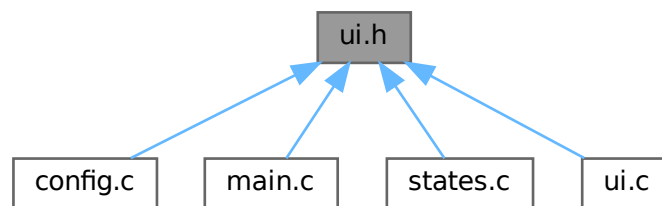
Protótipos para funções de interface de usuário (UI) no terminal.

```
#include "fila.h"
```

Include dependency graph for ui.h:



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



Functions

- void `limpar_tela()`
Limpa a tela do terminal.
- void `pausar()`
Pausa a execução do programa até que o usuário pressione Enter.
- void `inicializar_interface(Fila *propagandas)`
Exibe a tela de boas-vindas da aplicação.

4.27.1 Detailed Description

Protótipos para funções de interface de usuário (UI) no terminal.

4.27.2 Function Documentation

4.27.2.1 inicializar_interface()

```
void inicializar_interface (  
    Fila * propagandas )
```

Exibe a tela de boas-vindas da aplicação.

Parameters

in	<i>propagandas</i>	Ponteiro para a fila de propagandas, para exibir a primeira.
----	--------------------	--

4.27.2.2 limpar_tela()

```
void limpar_tela ( )
```

Limpa a tela do terminal.

- Tenta usar "clear" (Linux/macOS) ou "cls" (Windows).

4.27.2.3 pausar()

```
void pausar ( )
```

Pausa a execução do programa até que o usuário pressione Enter.

- Útil para permitir que o usuário leia as mensagens antes da tela ser limpa.
- Esta versão corrigida consome quaisquer caracteres residuais no buffer de entrada e espera por um único Enter para continuar.

4.28 ui.h

[Go to the documentation of this file.](#)

```
00001
00006 #ifndef UI_H
00007 #define UI_H
00008
00009 #include "fila.h" // Incluído para o parâmetro de inicializar_interface
00010
00015 void limpar_tela();
00016
00021 void pausar();
00022
00027 void inicializar_interface(Fila *propagandas);
00028
00029 #endif
```


Index

- adicionar_log
 - log_vendas.c, [33](#)
 - log_vendas.h, [36](#)
- adicionar_produto
 - lista_produtos.c, [26](#)
 - lista_produtos.h, [30](#)
- ant
 - produto, [11](#)
- buscar_produto
 - lista_produtos.c, [26](#)
 - lista_produtos.h, [30](#)
- calcular_troco
 - pagamento.c, [39](#)
 - pagamento.h, [41](#)
- carregar_logs
 - log_vendas.c, [34](#)
 - log_vendas.h, [36](#)
- carregar_produtos_csv
 - lista_produtos.c, [27](#)
 - lista_produtos.h, [30](#)
- config.c, [13](#)
 - menu_configuracao, [14](#)
 - verificar_senha, [14](#)
- config.h, [14](#)
 - menu_configuracao, [15](#)
 - verificar_senha, [16](#)
- CONFIGURACAO
 - states.h, [49](#)
- Contexto, [5](#)
 - lista_produtos, [6](#)
 - logs_vendas, [6](#)
 - produto_selecionado, [6](#)
- desenfileirar
 - fila.c, [21](#)
 - fila.h, [24](#)
- desfazer_moeda
 - pagamento.c, [40](#)
 - pagamento.h, [41](#)
- enfileirar
 - fila.c, [21](#)
 - fila.h, [24](#)
- estado_configuracao
 - states.c, [46](#)
 - states.h, [49](#)
- estado_menu_principal
 - states.c, [46](#)
- states.h, [49](#)
- estado_pagamento
 - states.c, [46](#)
 - states.h, [49](#)
- estado_selecao_produto
 - states.c, [47](#)
 - states.h, [49](#)
- estados
 - PilhaEstados, [10](#)
- estatisticas.c, [16](#)
 - gerar_estatisticas, [17](#)
 - salvar_estatisticas_csv, [18](#)
- estatisticas.h, [18](#)
 - gerar_estatisticas, [19](#)
 - salvar_estatisticas_csv, [20](#)
- estoque
 - produto, [11](#)
- exibir_logs
 - log_vendas.c, [34](#)
 - log_vendas.h, [36](#)
- Fila, [6](#)
 - frente, [6](#)
 - mensagens, [6](#)
 - tamanho, [7](#)
 - tras, [7](#)
- fila.c, [20](#)
 - desenfileirar, [21](#)
 - enfileirar, [21](#)
 - frente_fila, [22](#)
 - inicializar_fila, [22](#)
- fila.h, [22](#)
 - desenfileirar, [24](#)
 - enfileirar, [24](#)
 - frente_fila, [24](#)
 - inicializar_fila, [24](#)
- frente
 - Fila, [6](#)
- frente_fila
 - fila.c, [22](#)
 - fila.h, [24](#)
- gerar_estatisticas
 - estatisticas.c, [17](#)
 - estatisticas.h, [19](#)
- id
 - produto, [11](#)
- inicializar_fila
 - fila.c, [22](#)

- fila.h, 24
- inicializar_interface
 - ui.c, 51
 - ui.h, 52
- inserir_moedas
 - pagamento.c, 40
 - pagamento.h, 42
- liberar_lista_produtos
 - lista_produtos.c, 27
 - lista_produtos.h, 31
- liberar_logs
 - log_vendas.c, 34
 - log_vendas.h, 37
- limpar_tela
 - ui.c, 51
 - ui.h, 53
- lista_produtos
 - Contexto, 6
- lista_produtos.c, 25
 - adicionar_produto, 26
 - buscar_produto, 26
 - carregar_produtos_csv, 27
 - liberar_lista_produtos, 27
 - listar_produtos, 27
 - remover_produto, 28
 - salvar_produtos_csv, 28
- lista_produtos.h, 28
 - adicionar_produto, 30
 - buscar_produto, 30
 - carregar_produtos_csv, 30
 - liberar_lista_produtos, 31
 - listar_produtos, 31
 - remover_produto, 31
 - salvar_produtos_csv, 31
- listar_produtos
 - lista_produtos.c, 27
 - lista_produtos.h, 31
- Log, 7
 - produto_id, 8
 - prox, 8
 - timestamp, 8
 - troco, 8
 - valor_pago, 8
- log_vendas.c, 32
 - adicionar_log, 33
 - carregar_logs, 34
 - exibir_logs, 34
 - liberar_logs, 34
 - salvar_logs, 34
- log_vendas.h, 35
 - adicionar_log, 36
 - carregar_logs, 36
 - exibir_logs, 36
 - liberar_logs, 37
 - salvar_logs, 37
- logs_vendas
 - Contexto, 6
- main
 - main.c, 38
- main.c, 38
 - main, 38
- mensagens
 - Fila, 6
- menu_configuracao
 - config.c, 14
 - config.h, 15
- MENU_PRINCIPAL
 - states.h, 49
- nome
 - produto, 11
- PAGAMENTO
 - states.h, 49
- pagamento.c, 39
 - calcular_troco, 39
 - desfazer_moeda, 40
 - inserir_moedas, 40
- pagamento.h, 40
 - calcular_troco, 41
 - desfazer_moeda, 41
 - inserir_moedas, 42
- pausar
 - ui.c, 51
 - ui.h, 53
- Pilha, 8
 - topo, 9
 - valores, 9
- pilha.c, 42
- pilha.h, 43
- PilhaEstados, 9
 - estados, 10
 - topo, 10
- preco
 - produto, 11
- produto, 10
 - ant, 11
 - estoque, 11
 - id, 11
 - nome, 11
 - preco, 11
 - prox, 11
- produto_id
 - Log, 8
- produto_selecionado
 - Contexto, 6
- prox
 - Log, 8
 - produto, 11
- remover_produto
 - lista_produtos.c, 28
 - lista_produtos.h, 31
- SAIR
 - states.h, 49

- salvar_estatisticas_csv
 - estatisticas.c, [18](#)
 - estatisticas.h, [20](#)
- salvar_logs
 - log_vendas.c, [34](#)
 - log_vendas.h, [37](#)
- salvar_produtos_csv
 - lista_produtos.c, [28](#)
 - lista_produtos.h, [31](#)
- SELECAO_PRODUTO
 - states.h, [49](#)
- State
 - states.h, [48](#)
- states.c, [45](#)
 - estado_configuracao, [46](#)
 - estado_menu_principal, [46](#)
 - estado_pagamento, [46](#)
 - estado_selecao_produto, [47](#)
- states.h, [47](#)
 - CONFIGURACAO, [49](#)
 - estado_configuracao, [49](#)
 - estado_menu_principal, [49](#)
 - estado_pagamento, [49](#)
 - estado_selecao_produto, [49](#)
 - MENU_PRINCIPAL, [49](#)
 - PAGAMENTO, [49](#)
 - SAIR, [49](#)
 - SELECAO_PRODUTO, [49](#)
 - State, [48](#)
 - VOLTAR, [49](#)
- tamanho
 - Fila, [7](#)
- timestamp
 - Log, [8](#)
- topo
 - Pilha, [9](#)
 - PilhaEstados, [10](#)
- tras
 - Fila, [7](#)
- troco
 - Log, [8](#)
- ui.c, [50](#)
 - inicializar_interface, [51](#)
 - limpar_tela, [51](#)
 - pausar, [51](#)
- ui.h, [51](#)
 - inicializar_interface, [52](#)
 - limpar_tela, [53](#)
 - pausar, [53](#)
- valor_pago
 - Log, [8](#)
- valores
 - Pilha, [9](#)
- verificar_senha
 - config.c, [14](#)
 - config.h, [16](#)
- VOLTAR
 - states.h, [49](#)