Universidade do Minho
Escola de Engenharia

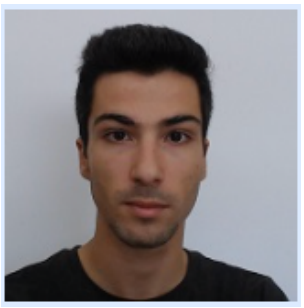
Sistema de Gestão de Vendas

Trabalho Prático de C

Grupo 26

a84610 – Gonçalo Almeida

a86788 – Lázaro Pinheiro



a80960 – Rúben Rodrigues

Laboratórios de Informática III
2º Ano, 2º Semestre
2018/2019

Introdução

O presente trabalho prático desenvolve-se no âmbito da Unidade Curricular Laboratórios de Informática III, lecionada no 2º semestre do 2º ano do curso de Engenharia Informática.

Este trabalho tem como objetivo aumentar os conhecimentos na linguagem C e, fundamentalmente, a apresentação dos desafios que se colocam a quem concebe e programa aplicações software com grandes volumes de dados e com a mais elevada complexidade algorítmica e estrutural.

Procura-se com a implementação do trabalho responder a todas as *queries* e a todos os módulos de dados.

Podemos considerar este projeto como um software que gere os produtos, os clientes e as vendas, com vista a facilitar a organização e aumentar o rendimento de uma empresa.

O mesmo trabalho prático reveste-se de uma mais valia para colocar em prática conhecimentos, fomentar e consolidar aprendizagens e desbravar terreno no mundo da linguagem C.

Módulos de Dados

Catálogo de Produtos

Para a realização deste módulo de dados, julgou-se de relevante interesse a utilização de uma das bibliotecas sugeridas pelo docente da UC, a *Glib*, a qual satisfaz todos os critérios necessários para otimizar da melhor maneira o nosso programa. Tendo por base as sugestões analisadas, percebeu-se que a implementação da estrutura *GTree* seria a mais adequada, pois representa uma árvore auto balanceada. Esta é uma coleção ordenada de pares chave/valor otimizados para uma pesquisa e travessia em ordem. O referido par chave/valor é código do Produto e 1, respetivamente, pois facilita a utilização da função predefinida na estrutura utilizada (*g_tree_lookup ()*), que retorna o valor para o qual uma chave aponta ou NULL no caso de a chave não existir.

Considerando essa situação, atribuiu-se 1 ao valor para pudermos diferenciar. A complexidade da procura é **O(log n)** em que n é o número de nodos.

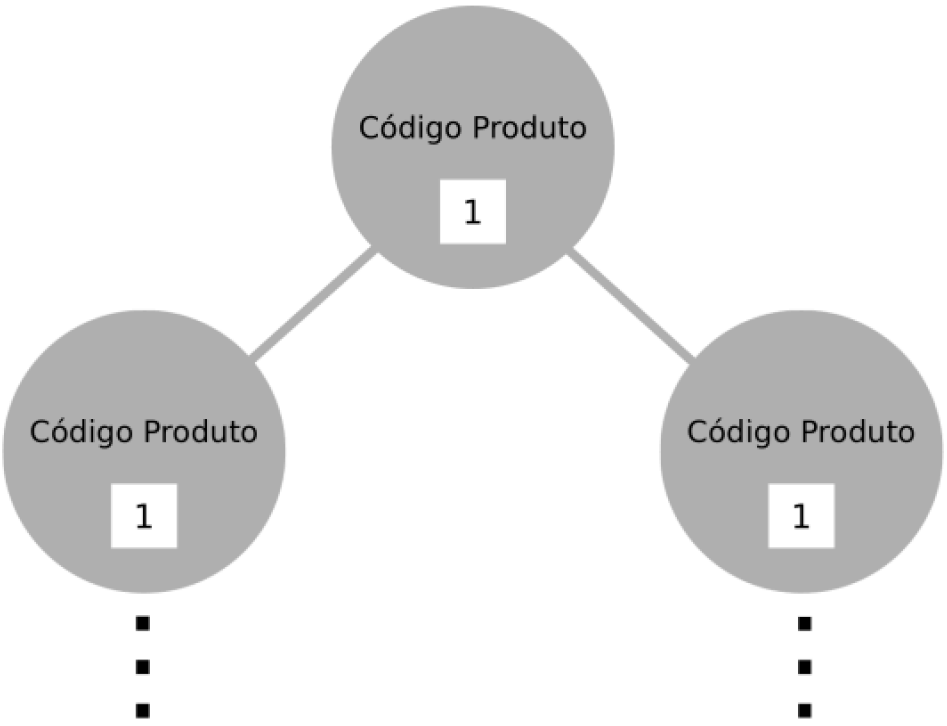


Figura 1 - Ilustração da implementação do módulo de Produtos.

Type Definitions

CatalogoProdutos.h - typedef **struct** catalogoprodutos *CatalogoProdutos;
CatalogoProdutos.h - typedef **struct** querie2struct *Querie2Struct;

API Comentada

CatalogoProdutos insereProduto(CatalogoProdutos catProdutos, char *produto);
Insere um produto no Catálogo de Produtos;
Retorna: O Catálogo de Produtos resultante da inserção de um produto;

int existeProduto(CatalogoProdutos catProdutos,char *produto);
Determina se um produto existe;
Retorna:
- 1 se existir;
- 0 se não existir;

void libertaMemoriaProdutos(CatalogoProdutos catProdutos);
Liberta a memória alocada para armazenar os dados no módulo.

Querie2Struct newQuerie2 (Querie2Struct q, char c, int n);
Struct usada para a realização da query2.

void querie2 (CatalogoProdutos catProdutos, char c, int* nTotal, char* produtos[]);
Devolve uma lista ordenada alfabeticamente dos produtos começados por uma certa letra.

Catálogo de Clientes

Conforme explicação anteriormente apresentada, pensou-se ser igualmente favorável a utilização da biblioteca *Glib*. Com o propósito de alcançar resultados semelhantes ao módulo anterior, fez-se uso de uma *Gtree*. Neste caso, a chave é um código de cliente e o valor é 1. Estes valores foram escolhidos com os mesmos critérios referidos anteriormente.

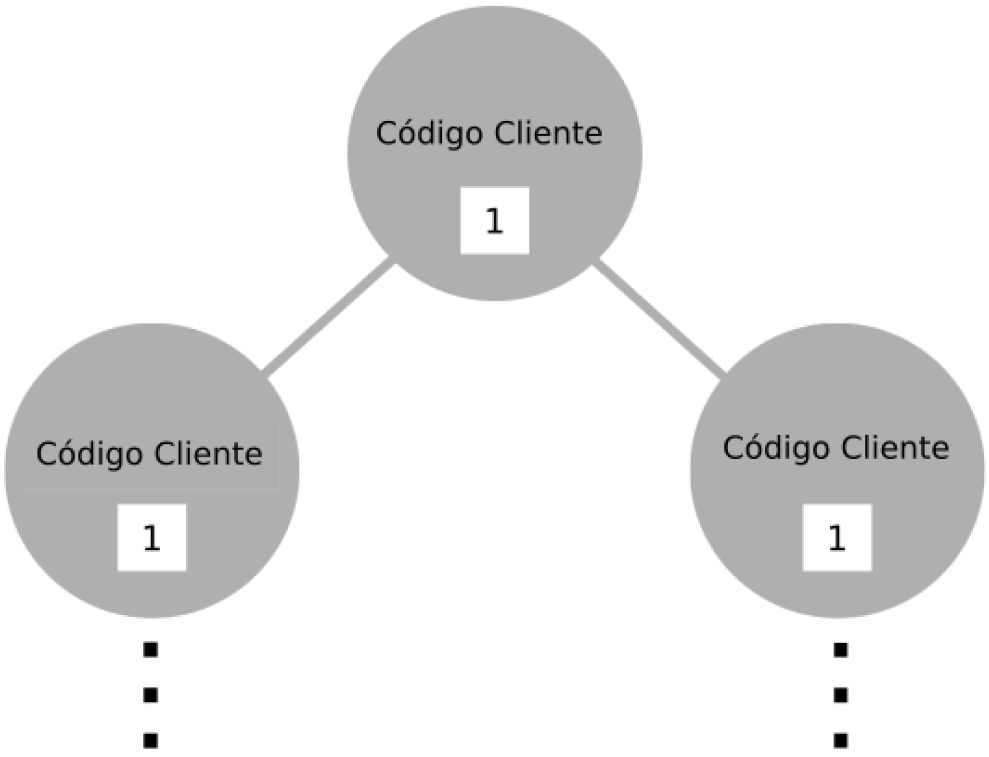


Figura 2 - Ilustração da implementação do módulo de Clientes.

Definições de tipos

CatalogoClientes.h - typedef **struct** catalogoclientes *CatalogoClientes;

API Comentada

CatalogoClientes insereCliente(CatalogoClientes catClientes,char* cliente);
Insere um cliente no Catálogo de Clientes;
Retorna: O Catálogo de Clientes resultante da inserção de um cliente.

int existeCliente(CatalogoClientes catClientes,char *cliente);
Determina se um cliente existe;
Retorna:
- 1 se existir;
- 0 se não existir;

void libertaMemoriaClientes(CatalogoClientes catClientes);
Liberta a memória alocada para armazenar os dados no módulo.

Faturação Global

Na implementação deste módulo mante-se a escolha da biblioteca anterior, usando desta vez uma nova estrutura para facilitar a procura, *GHashTable*, que consiste num par chave/valor, onde a chave é um código de produto e o valor é um *array* (filiais) de *arrays* (meses), sendo que cada célula é uma estrutura que contém o numero de vendas e o total faturado diferenciando entre o tipo normal (N) e promocional (P).
Deste modo, obtém-se uma complexidade de **O(1)** na procura de elementos. No caso de se pretender uma travessia em ordem realiza-se uma correspondência entre o catálogo de produtos e a faturação global. Este módulo não contém qualquer referência a clientes e contém todos os produtos.

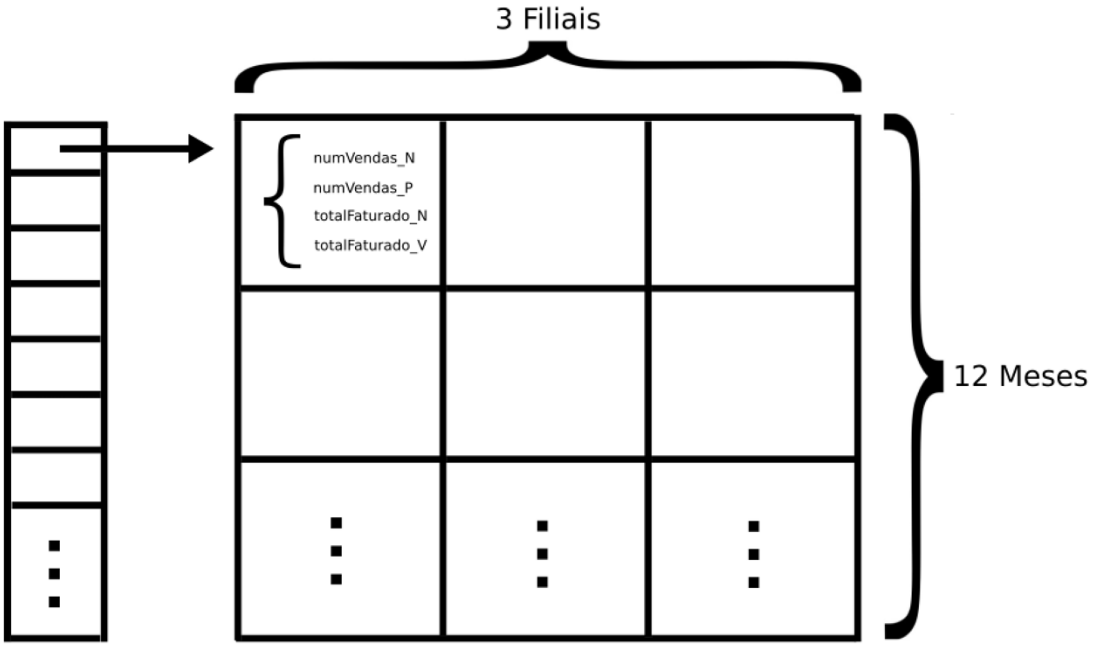


Figura 3 - Ilustração da implementação do módulo de Faturação Global.

Definições de tipos

FaturaçãoGlobal.h - typedef **struct** faturacaoglobal *FaturacaoGlobal;
FaturaçãoGlobal.h - typedef **struct** fatura *Fatura;
FaturaçãoGlobal.h - typedef Fatura *FaturacaoMes;
FaturaçãoGlobal.h - typedef FaturacaoMes *FaturacaoFilial;

API Comentada

FaturacaoGlobal inicializaFaturacao(FaturacaoGlobal faturacao,char* produto);
Inicializa uma nova faturação caso ela não exista

FaturacaoGlobal insereFaturacao(FaturacaoGlobal faturacao,char* codProd,int precoUnit,int quantidade,char tipo,int mes,int filial);
No caso de já existir uma faturação, esta função adiciona-lhe um novo elemento.

void libertaMemoriaFaturacao(FaturacaoGlobal faturacao);
Liberta a memória alocada para armazenar os dados no módulo.

void querie3G (FaturacaoGlobal faturacao, int mes, char* produto);
Query 3 caso o utilizador pretenda o resultado global.

void querie3F (FaturacaoGlobal faturacao, int mes, char* produto);
Query 3 caso o utilizador pretenda o resultado filial a filial.

void querie6 (FaturacaoGlobal faturacao, int* naoComprados);
Determina o número de clientes que não realizaram compras e o número de produtos que ninguém comprou.

void querie8F (FaturacaoGlobal faturacao, int mes1, int mes2, double* totFaturado, int* numVendas);
Total de vendas e total faturado num certo período de tempo.

Gestão Filiais

A implementação deste módulo, usa uma *GTree* ,pois pretende-se manter a ordem dos clientes, onde a chave é um código de cliente e o valor uma *GHashTable*, por forma a facilitar a procura que consiste num par chave/valor, onde a chave é um código de produto e o valor é um *array* (filiais) de *arrays* (meses), sendo que cada célula é uma estrutura que contém as quantidades vendidas desse produto diferenciando entre o tipo normal (N) e promocional (P).

Deste modo, obtém-se uma complexidade de **O(log n)** na procura dos clientes e **O(1)** na procura dos produtos. No caso de se pretender uma travessia em ordem dos produtos utiliza-se o mesmo método explicado para o módulo de Faturação Global.

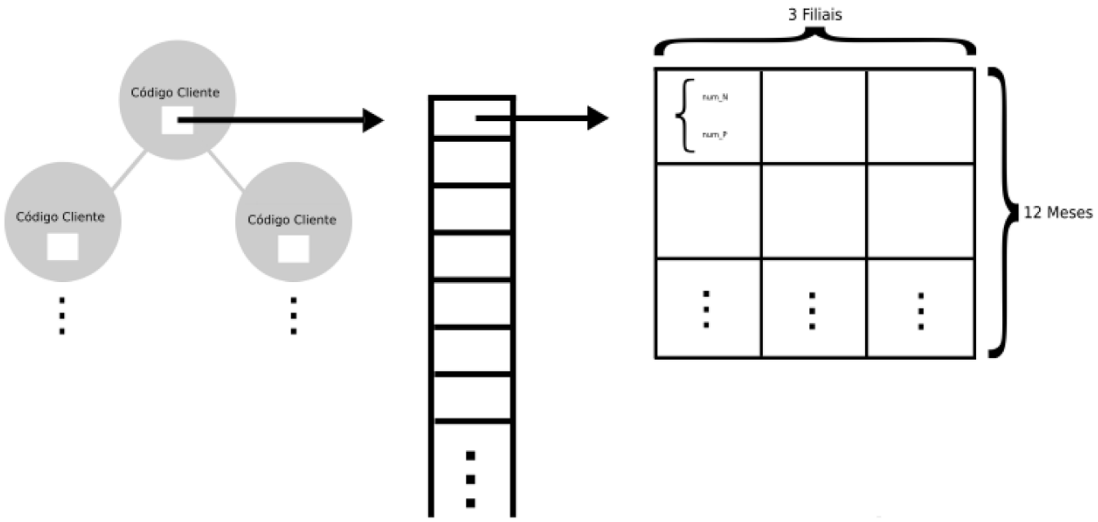


Figura 4 - Ilustração da implementação do modulo de Gestão de Filiais.

Definições de tipos

GestaoFilial.h - typedef **struct** gestaoclientes *GestaoClientes;
GestaoFilial.h - typedef **struct** gestaoprodutos *GestaoProdutos;
GestaoFilial.h - typedef **struct** quantidades *Quantidades;
GestaoFilial.h - typedef Quantidades *GestaoMes;
GestaoFilial.h - typedef GestaoMes *GestaoFilial;

API Comentada

GestaoClientes newGC(GestaoClientes gc, char* cliente);
Inicializa uma nova gestão de clientes caso ela não exista.

GestaoClientes insertGC (GestaoClientes gc, char* produto, int quantidade, char tipo, char* cliente, int mes, int filial);
Insere uma gestão de clientes.

Menu e Navegação sobre resultados

O menu do programa apresentado contém uma breve explicação de cada uma das *queries*, com um número associado à sua ordem. São 14 opções em que 12 referentes às *queries*, a 13 é a opção de ajuda que contém explicações mais aprofundadas de cada uma das *queries*, e por fim a opção 14, que visa o encerramento do programa.

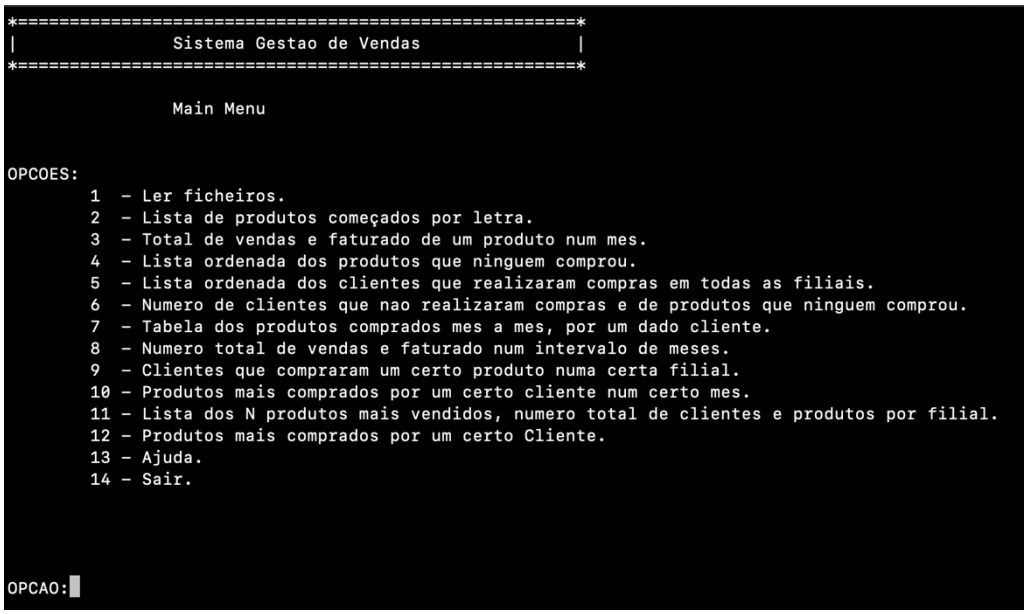


Figura 5 - Menu da aplicação SGV.

Quanto ao navegador, é chamado sempre que necessário, sempre que é preciso imprimir para o utilizador um *array* pedido, como por exemplo na *query* 2, onde é pedida a lista de Produtos começados por uma determinada letra. Neste caso, o navegador, tendo como informação a lista referida e o tamanho da mesma, imprime de 20 em 20 para o utilizador, sendo possível avançar uma página usando a tecla 'd', recuar usando a tecla 'e' e sair usando qualquer outra tecla.

Assim, é possível ao utilizador receber a informação pretendida de uma maneira mais acessível.

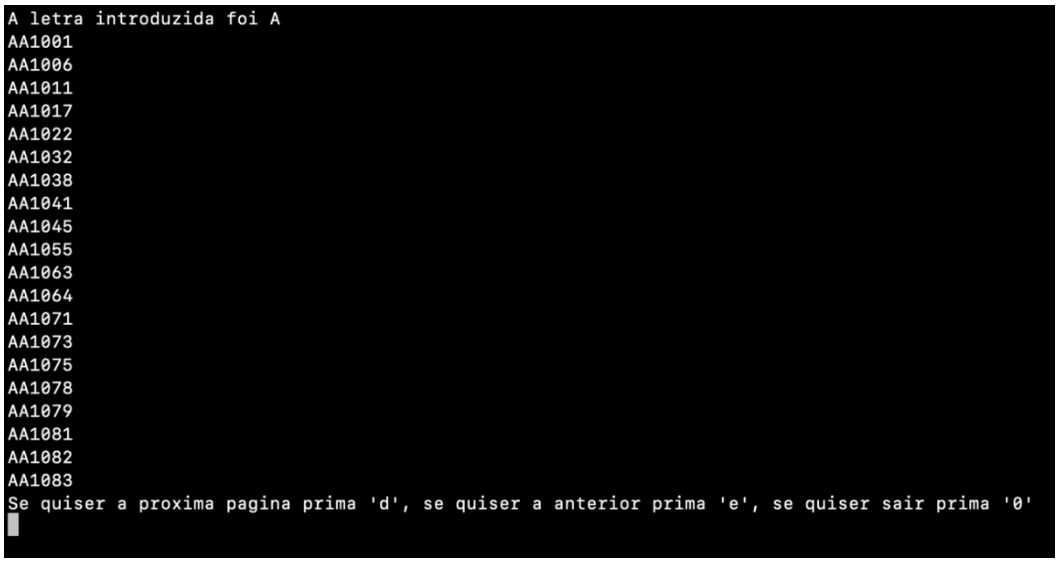


Figura 6 - Ilustração do menu de navegação.

Performance

Ficheiros

Vendas_1M.txt

A leitura do ficheiro de vendas com as estruturas implementadas tem os seguintes tempos:

- *Real* = 0.637;
- *User* = 0.594 s;
- *System* = 0.038 s.

Produtos.txt

A leitura do ficheiro de produtos com as estruturas implementadas tem os seguintes tempos:

- *Real* = 0.148 s;
- *User* = 0.130 s;
- *System* = 0.015s.

Clientes.txt

A leitura do ficheiro de cliente com as estruturas implementadas tem os seguintes tempos:

- *Real* = 0.033 s
- *User* = 0.024 s
- *System* = 0.007s

Makefile

```
CC=gcc
CFLAGS=-Wall -ansi -O2
GLIBCFLACGS=`pkg-config --cflags glib-2.0`
GLIBLIBS=`pkg-config --libs glib-2.0`

SGV: CatalogoClientes.o CatalogoProdutos.o FaturacaoGlobal.o GestaoFilial.o
Leitura.o Queries.o Navegador.o main.c
   $(CC) $(CFLAGS) $(GLIBCFLACGS) $(GLIBLIBS) CatalogoClientes.o
CatalogoProdutos.o FaturacaoGlobal.o GestaoFilial.o Leitura.o Queries.o Navegador.o
main.c -o SGV


CatalogoClientes.o: CatalogoClientes.c
   $(CC) $(CFLAGS) $(GLIBCFLACGS) -c CatalogoClientes.c -o CatalogoClientes.o



CatalogoProdutos.o: CatalogoProdutos.c
   $(CC) $(CFLAGS) $(GLIBCFLACGS) -c CatalogoProdutos.c -o CatalogoProdutos.o


FaturacaoGlobal.o: FaturacaoGlobal.c
   $(CC) $(CFLAGS) $(GLIBCFLACGS) -c FaturacaoGlobal.c -o FaturacaoGlobal.o


GestaoFilial.o: GestaoFilial.c
   $(CC) $(CFLAGS) $(GLIBCFLACGS) -c GestaoFilial.c -o GestaoFilial.o


Leitura.o: Leitura.c
   $(CC) $(CFLAGS) $(GLIBCFLACGS) -c Leitura.c -o Leitura.o


Queries.o: Queries.c
   $(CC) $(CFLAGS) $(GLIBCFLACGS) -c Queries.c -o Queries.o


Navegador.o: Navegador.c
   $(CC) $(CFLAGS) $(GLIBCFLACGS) -c Navegador.c -o Navegador.o



clean:
   rm SGV
   rm *. o
```

Grafo de dependências

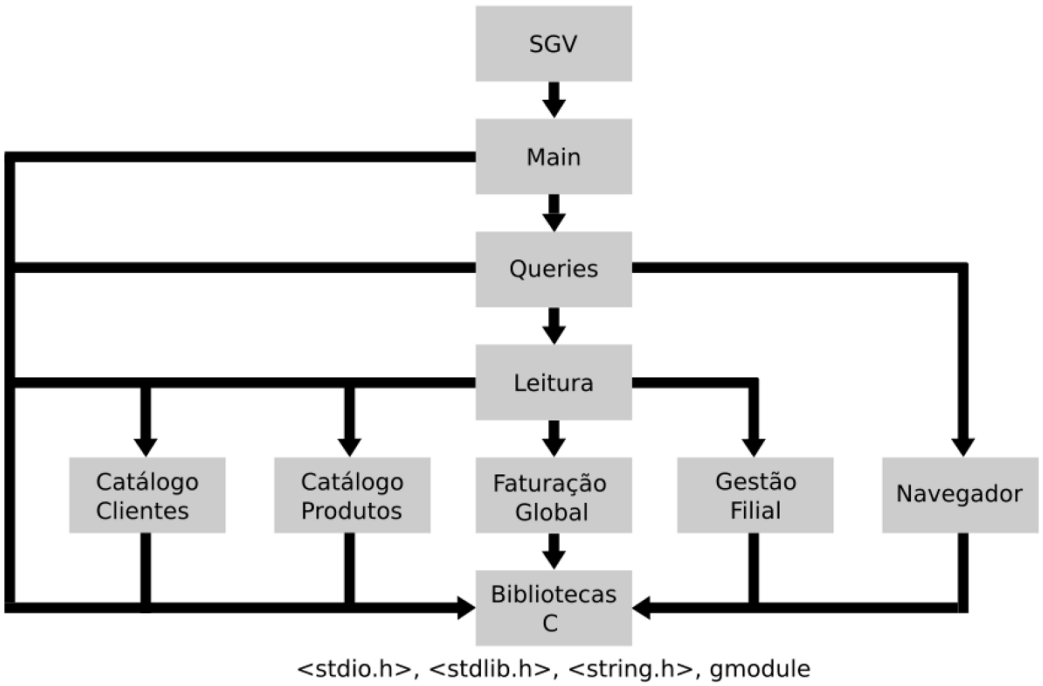


Figura 7 - Ilustração do grafo de dependências.

Conclusão

Terminada a realização do trabalho prático, chega o momento de refletir de analisar as dificuldades sentidas e os momentos de aprendizagem alcançados.

De forma geral, a realização do trabalho prático não decorreu como esperado, destacando-se a dificuldade sentida na estruturação dos módulos para uma resposta eficiente às *queries*.

Além disso, o trabalho prático não respondeu a todos os objetivos previamente estabelecidos na introdução, não tendo sido possível a implementação de todas as *queries* e de um dos módulos de dados, não concluindo, assim, o trabalho na sua totalidade.

O objetivo geral avançado no inicio deste relatório – criar uma aplicação que nos permita gerir vendas, ou seja, um software com o intuito de facilitar a organização e aumentar o rendimento de uma empresa – não foi totalmente cumprido, pese embora se tenha seguido para a execução do software as indicações dadas pelo Docente da U.C. de Laboratórios de Informática III.

Em forma de conclusão, pode ser adiantado que o trabalho não foi concretizado com sucesso, dado que o software, depois de elaborado, não apresenta resposta a todas as *queries* interativas.