ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Projeto de Laboratório de Programação

Licenciatura em Engenharia Informática

Licenciatura em Segurança Informática em Redes de Computadores

2022/2023

Grupo 100

Bruno Miguel Magalhães Soares Pontvianne Ferreira – 8220191

Miguel Correia Ribeiro Rangel Tavares – 8220229

Gabriel Ferreira Moreira – 8220225

1. Introdução

Face ao desafio proposto pela empresa “Móveis para Todos” aos alunos do 1º ano da área de informática da ESTG, irá ser desenvolvida uma aplicação utilizando a linguagem de programação “C” no domínio “main”. Durante todo o processo de criação e desenvolvimento deste programa irá ser utilizado o IDE NetBeans.

Ao longo de todo o processo, o nosso grupo tem como objetivo conceber uma aplicação prática, de fácil compreensão e utilização, colocando-nos desta forma no “ponto de vista” do utilizador e, simultaneamente, desenvolver um código simples, legível e otimizado com foco nas funcionalidades propostas, e adicionais, e respetiva manutenção.

A aplicação desenvolvida tem como objetivo cumprir e obedecer às funcionalidades solicitadas, bem como, incluir e suportar funcionalidades “extra”.

Inicialmente irá ser desenvolvida uma linha de pensamento lógica, cuidada e objetiva, que funcionará como a “base” de todo o projeto. Pretendemos então ter uma organização uniforme a todos os desenvolvedores do programa para uma agilidade acrescida na criação e progresso do trabalho desenvolvido.

2. Funcionalidades requeridas

Estruturas de dados criadas:

Note-se que as variáveis “extra” não são solicitadas no enunciado e apenas estão aqui apresentadas pois são declaradas nas respetivas estruturas.

Biblioteca “*perfil.h*”

Data		Morada	
Tipo	**Variáveis**	**Tipo**	**Variáveis**
int	dia, mes, ano	char	*cidade, *rua, *codigoPostal, *numeroPorta

Data:

Criamos esta estrutura com o propósito de conter todas as variáveis necessárias para armazenar qualquer data (dia/mês/ano).

Todas as variáveis são do tipo “int” pois: não existem dias, meses ou anos que este tipo de variável não armazene e também porque é a variável que ocupa menos memória para o propósito designado.

Morada:

Criamos esta estrutura com o propósito de conter todas as variáveis necessárias para armazenar qualquer morada possível (rua, código postal, cidade, número da porta).

Todas as variáveis são do tipo “char” pois todos os dados que a constituem são/contêm caracteres. São, também, inicializadas como strings dinâmicas devido ao facto de, todas elas, poderem ter diferentes tamanhos.

Ambas as estruturas (Data e Morada) serão de seguida utilizadas na estrutura “Cliente”.

Cliente	
Tipo	**Variáveis**
int	numeroCliente, numeroEncomendas(extra)
char	*nif, *pais, *password(extra), *nome, *apelido
Data	data(extra)
Morada	morada
Estado	estado
float	dinheiroGasto(extra)

Cliente:

Criamos esta estrutura com o propósito de conter todas as variáveis necessárias para armazenar todo o tipo de dados referentes ao perfil Cliente.

A variável “numeroCliente” é do tipo “int”, pois armazena um número atribuído aleatoriamente a cada conta (de cliente) criada. A variável “numeroEncomendas” também é do tipo “int” pois armazena o número de encomendas feita por um cliente.

Todas as variáveis do tipo “char” (“*nif”, “*pais”, “*password”, “*nome”, “*apelido”) são strings dinâmicas que armazenam o respetivo dado.

A variável “data”, do tipo “Data”, armazena uma data, neste caso, de nascimento.

A variável “morada”, do tipo “Morada”, armazena a morada respetiva a cada cliente.

A variável “estado”, do tipo “Estado”, informa se o perfil está Ativo ou Inativo.

A variável “dinheiroGasto” é do tipo “float”, pois armazena a quantidade de dinheiro gasta por um cliente na empresa (a variável preço, na estrutura “Encomenda” é do tipo float).

Clientes	
Tipo	**Variáveis**
int	contador
Cliente	*cliente

Clientes:

Criamos esta estrutura com o propósito de ser o array que armazena os clientes.

A variável “contador”, do tipo “int”, representa o número de clientes criados. Esta variável

é incrementada (+ 1), após ser inicializada a 0, por cada cliente criado.

A variável “*cliente”, do tipo “Cliente”, é um apontador para a estrutura “Cliente” que contém os dados de um perfil.

Admin	
Tipo	**Variáveis**
int	numeroAdmin(extra)
char	*password(extra), *nome(extra), *apelido(extra)

Admin:

Criamos esta estrutura com o propósito de conter todas as variáveis necessárias para armazenar todo o tipo de dados referentes ao perfil Administrador.

A variável “numeroAdmin” é do tipo “int”, pois armazena um número atribuído aleatoriamente a cada conta (de administrador) criada.

Todas as variáveis do tipo “char” (“*password”, “*nome”, “*apelido”) são strings dinâmicas que armazenam o respetivo dado.

Admins	
Tipo	**Variáveis**
int	contador
Admin	*admin

Admins:

Criamos esta estrutura com o propósito de ser o array que armazena os administradores.

A variável “contador”, do tipo “int”, representa o número de administradores criados. Esta variável é incrementada (+ 1), após ser inicializada a 0, por cada administrador criado.

A variável “*admin”, do tipo “Admin”, é um apontador para a estrutura “Admin” que contém os dados de um perfil.

Biblioteca “*encomenda.h*”

MovelEncomenda	
Tipo	**Variáveis**
int	quantidade
char	codigoProtuto[TAMANHO_CODIGO_MOVEL]

MovelEncomenda:

Criamos esta estrutura com o propósito de conter os dados de um móvel numa encomenda.

A variável “quantidade” é do tipo “int” pois representa a quantidade, do móvel em questão, pedida na encomenda.

A variável “codigoProduto” é do tipo “char” pois armazena o código do móvel encomendado.

Encomenda	
Tipo	**Variáveis**
int	numeroCliente, contador
Data	Data, dataEntrega
float	preco
MovelEncomenda	*movelEncomenda

Encomenda:

Criamos esta estrutura com o propósito de conter todas as variáveis necessárias para armazenar todo o tipo de dados referentes a uma encomenda.

A variável “numeroCliente” é do tipo “int”, pois armazena o número de conta (de cliente) à qual a encomenda está atribuída.

A variável “contador”, do tipo “int”, representa o número de móveis adicionados à encomenda. Esta variável é incrementada (+ 1), após ser inicializada a 0, por cada móvel criado.

A variável “data”, do tipo “Data”, representa a data na qual a encomenda foi realizada. A variável “dataEntrega”, do mesmo tipo, representa a data prevista para a entrega da encomenda.

A variável “preco” é do tipo “float”, pois armazena o preço total da encomenda.

A variável “*movelEncomenda” é do tipo “MovelEncomenda”, pois corresponde ao array de móveis pertencentes à encomenda.

Encomendas	
Tipo	**Variáveis**
int	contador
Encomenda	*encomenda

Encomendas:

Criamos esta estrutura com o propósito de ser o array que armazena as encomendas.

A variável "contador", do tipo "int", representa o número de encomendas criadas. Esta variável é incrementada (+ 1), após ser inicializada a 0, por cada encomenda criada.

A variável "*encomenda", do tipo "Encomenda", é um apontador para a estrutura "Encomenda".

Biblioteca "*moveis.h*"

Material	
Tipo	**Variáveis**
int	contador
char	*nomeMaterial, *unidade, codigoMaterial[TAMANHO_CODIGO_MATERIAL]

Material:

Criamos esta estrutura com o propósito de conter todas as variáveis necessárias para armazenar todo o tipo de dados referentes a um material.

A variável “contador”, do tipo “int”, assume duas funções:

1. Quando utilizada num móvel (exemplo: movel.material[(posição)].contador) esta corresponde à quantidade necessária do material para o móvel.
2. Quando utilizada na estrutura “Materiais” (exemplo: materiais.material[(posição)].contador) esta corresponde ao número de vezes que o material foi utilizado num móvel encomendado.

A variáveis “*nomeMaterial” e “*unidade”, do tipo “char”, são strings dinâmicas que armazenam o: o nome do material e o tipo de unidade (par ou unidade), respetivamente.

A variável “codigoMaterial”, também do tipo “char”, armazena o código referente ao material, que tem, obrigatoriamente, 6 caracteres.

Materiais	
Tipo	**Variáveis**
int	contador
Material	*material

Materiais:

Criamos esta estrutura com o propósito de ser o array que armazena os materiais.

A variável “contador”, do tipo “int”, representa o número de materiais criados. Esta variável é incrementada (+ 1), após ser inicializada a 0, por cada material criado.

A variável “*material”, do tipo “Material”, é um apontador para a estrutura “Material”.

Movel	
Tipo	**Variáveis**
float	preco, comprimento, largura, altura
char	codigoProduto[TAMANHO_CODIGO_MOVEL], *nome
Material	*material
int	contadorMateriais
EstadoM	estado

Movel:

Criamos esta estrutura com o propósito de conter todas as variáveis necessárias para armazenar todo o tipo de dados referentes a um móvel.

A variável “preco” é do tipo “float” pois armazena o valor do móvel.

As variáveis “comprimento”, “largura” e “altura” também são do tipo “float”, pois armazenam as dimensões do móvel.

As variáveis “codigoProduto” e “*nome”, do tipo “char”, armazena: o código do móvel e o seu nome, respetivamente.

A variável “*material”, do tipo”Material”, corresponde ao array de materiais necessários para o móvel.

A variável “contadorMateriais” é do tipo “int”, pois informa quantos materiais distintos são usados no móvel.

A variável “estado” é do tipo “EstadoM”, pois informa se o móvel está Ativo ou Inativo.

Moveis	
Tipo	**Variáveis**
int	contador
Movel	*movel

Moveis:

Criamos esta estrutura com o propósito de ser o array que armazena os móveis.

A variável “contador”, do tipo “int”, representa o número de móveis criados. Esta variável é incrementada (+ 1), após ser inicializada a 0, por cada móvel criado.

A variável “*movel”, do tipo “Movel”, é um apontador para a estrutura “Movel”.

A aplicação requer a existência de dois perfis distintos: Administrador (perfil que permite tratar de toda a logística e gerência da empresa) e Cliente (perfil que permite aceder aos produtos e registar encomendas).

Funcionalidades do perfil Administrador

Face às funcionalidades propostas, é possível ao administrador gerir: os clientes, os produtos e a produção.

Gestão de clientes:

Qualquer administrador pode: criar, editar e remover/desativar clientes.

Para isso basta aceder à sua conta e selecionar a opção “Gestao de clientes”.

Dentro desse menu basta escolher a função que deseja. Caso escolha a opção “Criar cliente” é solicitado ao administrador que introduza os dados do cliente, tais como: país, NIF, nome, apelido, data de nascimento (ano, mês e dia), morada (cidade, rua, número da porta e código postal) e password. Posteriormente é executada a função “criarCliente();” que recebe como argumentos os dados previamente enumerados, com a adição do apontador para o número de cliente, que recebe o número gerado aleatoriamente pelo programa. Por fim essa função atribui

ao cliente o seu número (número esse que é único). O programa regressa ao menu “Gestao de clientes”.

Caso escolha a opção “Editar Cliente” é solicitado ao administrador o número de cliente que deseja alterar. Verifica-se se o número corresponde a alguma conta, se o cliente existe ou não, através de uma estrutura de repetição que compara o número de cliente dado pelo administrador com todos os números de cliente armazenados. Caso o cliente não exista o administrador é informado e o programa regressa ao menu “Gestao de clientes”, caso contrário é solicitado ao administrador o que pretende alterar no perfil que introduziu. Qualquer uma das 6 opções disponíveis solicita ao utilizador a nova entrada para a informação que escolher, por exemplo, se o administrador escolher alterar o nome e apelido é lhe solicitado que introduza os novos dados para essas duas informações, de seguida é utilizada a função “editarNomeClientes();” que recebe o novo nome e o novo apelido como argumento e os altera naquele perfil em especifico, é também utilizado este mesmo raciocínio para todas as outras opções.

Caso escolha a opção “Remover ou desativar/ativar cliente” é perguntado ao administrador se deseja “Remover”, “Ativar” ou “Desativar” o cliente. Após a sua escolha é lhe pedido que introduza o número do cliente que deseja remover, desativar ou ativar, sendo depois verificado se esse cliente efetivamente existe. Supondo que o administrador tinha optado por remover o cliente é verificado se este tem ou teve alguma encomenda em registo, caso o cliente tenha uma encomenda em registo apenas será possível alterar o estado da sua conta e o programa regressa ao menu que apresenta as opções de remoção e alteração de estado da conta, caso contrário é utilizada a função “removerCliente();” que apaga o cliente dos registos da empresa. Se o administrador optar por ativar ou desativar o cliente, ou até mesmo se o cliente tiver encomendas em registo, pode ainda optar por “Ativar” ou “Desativar” o cliente. Ao escolher cada uma destas opções é novamente solicitado que introduza o número de cliente que deseja alterar e também é novamente verificado se o cliente existe, depois, através da função “desativarAtivarCliente();”, é verificado se o estado para o qual o administrador deseja mudar o perfil não é o estado atual, ou seja, verifica-se se o administrador não está a ativar um perfil que se encontra ativo ou vice-versa. Caso isto aconteça o administrador é alertado sobre tal e é reenviado para o menu com as opções de remoção e alteração de estado da conta. Se o estado atual for diferente daquele que o administrador quer estabelecer, então a mesma função que verificava se estes eram iguais (“desativarAtivarCliente();”), também os altera com base no que foi solicitado.

Gestão de produtos:

Qualquer administrador pode: criar, editar e remover/desativar produtos (móveis).

Para isso basta aceder à sua conta e selecionar a opção “Gestao dos moveis”.

Dentro desse menu basta escolher a função que deseja. Caso escolha a opção “Adicionar movel” é solicitado ao administrador que introduza todos os dados referentes a um móvel, nomeadamente: código, nome, respetivas dimensões (comprimento, largura e altura), preço e materiais a utilizar e respetiva quantidade. Atenção, caso não existam materiais o administrador não conseguirá criar nenhum móvel. De seguida é utilizada a função “criarMovel();” que recebe como argumentos todos os dados previamente inseridos e cria o móvel pretendido.

Caso escolha a opção “Editar movel” é pedido o código do móvel que deseja alterar, verifica-se se o móvel existe ou não através da função “verificarMovel();”, se o móvel não existir o

administrador é levado de volta para o menu “Gestao dos moveis”, caso contrário é lhe perguntado o que deseja alterar no móvel em questão e é executada uma função adequada à sua escolha, exemplo: se o administrador quiser alterar o nome do móvel é executada uma função que receba como argumento o novo nome a dar ao móvel e realize essa alteração.

Caso escolha a opção “Remover movel” é pedido o código do móvel que deseja remover, verifica-se se o móvel existe ou não através da função “verificarMovel();”, se o móvel não existir o administrador é levado de volta para o menu “Gestao dos moveis”, caso contrário é eliminado o móvel em questão.

Gestão de produção:

Qualquer administrador pode obter a lista de componentes (materiais) para satisfazer as encomendas de uma dada semana do ano.

Para isso basta aceder à sua conta e selecionar a opção “Gestao de producao”.

De seguida basta escolher a opção “Ver material necessário para uma semana” ser-lhe-á solicitado duas datas distintas para poder escolher o período de tempo em que quer verificar os materiais, posteriormente será utilizada a função “materiaisSemana();” para enumerar essa lista.

Funcionalidades do perfil Cliente

Face às funcionalidades propostas, é possível aos clientes registar uma encomenda.

Para tal basta darem aceder à sua conta e escolher a opção “Registar encomendas”, de seguida é lhes pedido que introduzam os móveis que desejam comprar e as respetivas quantidades, posteriormente é executada a função “criarEncomenda();” que recebe todas as informações previamente inseridas e cria a respetiva encomenda associada ao número do cliente.

3. Funcionalidades propostas

As estruturas de dados utilizadas paras as 5 listagens propostas são as mesmas que foram previamente mencionadas nas “Funcionalidades requeridas”.

A partir de qualquer um dos dois menus principais (“Menu admin” e “Menu cliente”) é possível selecionar a opção “Extras”. Esta opção abre um menu com as 5 listagens criadas.

As funcionalidades propostas pelo grupo têm como objetivo apresentar dados, em listagem, que podem ser consultados por qualquer um dos perfis, e que dão a conhecer:

Os 5 clientes que mais dinheiro gastaram na empresa:

Esta função recorre a vários ciclos “for” que verificam quais os 5 clientes que mais dinheiro gastaram na empresa, percorrendo todos os perfis de cliente e comparando o valor da variável “dinheiroGasto” de cada perfil. No final da função estes dados são organizados e depois listados. É dado a conhecer: o número do cliente, o seu nome e apelido e o valor que este gastou.

Os 5 clientes que mais encomendas fizeram na empresa:

Esta função recorre a vários ciclos “for” que verificam quais os 5 clientes que mais encomendas têm em registo, percorrendo todos os perfis de cliente e comparando o valor da variável “numeroEncomendas” de cada perfil. No final da função estes dados são organizados e depois listados. É dado a conhecer: o número do cliente, o seu nome e apelido e o número de encomendas que estão registadas em seu nome.

Os 5 materiais mais utilizados:

Esta função recorre a vários ciclos “for” que verificam quais os materiais mais utilizados nos móveis encomendados. No final da função estes dados são organizados e depois listados. É dado a conhecer: o nome do material e o seu código.

Os 5 móveis com mais materiais:

Esta função recorre a vários ciclos “for” que coloca os 5 materiais mais usados num “array” e os organiza. No final da função estes dados são listados. É dado a conhecer: o nome do móvel, o seu código e o número de materiais distintos que o constituem.

Os 5 móveis mais caros:

Esta função recorre a vários ciclos “for” que verificam quais os móveis mais caros já criados na empresa. No final da função estes dados são organizados e depois listados. É dado a conhecer: o nome do móvel, o seu código e o seu preço.

4. Estrutura analítica do projeto

Desde o início que o projeto é idealizado e realizado em conjunto.

Começamos por definir as prioridades, como por exemplo, criar os inputs e definir as estruturas de dados para os perfis. Seguiu-se a criação das “CRUDS”, da biblioteca “moveis.h” e “encomenda.h”, e respetivas funções e estruturas. Mantemos o foco nas funcionalidades propostas, mas ao mesmo tempo nos possíveis extras que vamos conseguindo implementar.

Dividimos o projeto em três partes especificas: ter uma “main” bem trabalhada e simples, funcionalidades propostas (o que engloba as bibliotecas) e funcionalidades extra (o que também engloba as respetivas funções e bibliotecas).

Dentro de cada uma destas “divisões” mantemos sempre a ideia e o objetivo de criar algo prático e, acima de tudo, otimizado e organizado.

O projeto tem sido repartido de igual forma, o que garante que cada um dos desenvolvedores está ciente do que está a ser desenvolvido e do projeto que está a criar.

Dito isto, o trabalho foi repartido da seguinte forma:

Bruno Ferreira (nº 8220191):

- Desenvolvimento da “main”;
- Funções da biblioteca “input.h”;
- Funções de editar e desativar os perfis (clientes e administradores);
- Implementação dos extras;
- Documentação igualmente repartida;
- Relatório;

Miguel Tavares (nº 8220229):

- Funções da biblioteca “input.h”
- Funções de criar perfis (clientes e administradores);
- Funções das bibliotecas “encomenda.h” e “moveis.h”;
- Funções de listagem do projeto (listar clientes, administradores, móveis, encomendas);
- Alocação de memória dinâmica;
- Ajuda na implementação dos extras;
- Documentação igualmente repartida;

Gabriel Moreira (nº 8220225):

- Funções da biblioteca “input.h”
- Funções de criar perfis (clientes e administradores);
- Funções de remover perfis (clientes e administradores);
- Funções das bibliotecas “encomenda.h” e “moveis.h”;
- Alocação de memória dinâmica;
- Ficheiros;
- Documentação igualmente repartida;

5. Funcionalidades implementadas

O grupo implementou variadas funcionalidades no programa, de maneira a torná-lo mais prático e fácil de utilizar.

Implementamos um menu de registo que permite ao utilizador aceder à sua conta, ou então criar uma caso ainda não o tenha feito. Se o utilizador for um administrador, ao criar a sua conta, será pedido que introduza uma palavra passe geral fornecida aos administradores para que possa então criar a sua conta. Durante o processo de criação dessa conta o administrador é obrigado a introduzir uma palavra passe a seu gosto.

Cada cliente pode: criar, editar e remover a própria conta (caso tenha encomendas em registo o cliente tem que contactar um administrador para alterar o estado do perfil), cancelar encomendas por ele registadas e consultar informações acerca das respetivas encomendas.

Cada administrador pode: criar, editar e remover a própria conta, procurar por dados de um cliente em específico, listar os dados de todos os clientes registados, adicionar móveis, listar os dados de todos os móveis registados, verificar todas as encomendas em registo, verificar as encomendas em registo de um determinado cliente e criar, editar e remover materiais.

6. Conclusão

Durante todo este processo foi desenvolvido um programa otimizado, prático, de fácil compreensão e manutenção, tal como tínhamos definido.

Todo o programa foi fruto de um trabalho conjunto e organizado, que teve sempre como objetivo a criação de algo simples e funcional.

Por fim, consideramos que alcançamos o pretendido pelo projeto e que estivemos a par do desafio.