

Instituto Politécnico de Leiria – Escola Superior de Tecnologia e Gestão

Engenharia Informática – PL1

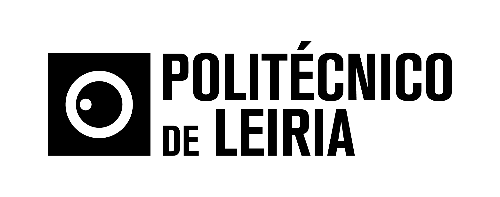
**CampusBike**

Ana Cassia Vasconcelos Cruz

Enrico Florentino Gomes

Leiria – Leiria

Janeiro de 2019



Instituto Politécnico de Leiria – Escola Superior de Tecnologia e Gestão

Engenharia Informática – PL1

**CampusBike**

Relatório do projeto

Ana Cassia Vasconcelos Cruz - 2180961

Enrico Florentino Gomes – 2181895

Programação I

Prof. Dr. Vítor Távora

Prof. Dr. Pedro Gago

Leiria – Leiria

Janeiro de 2019

**1. Introdução**

O projeto apresentado foi solicitado na Unidade Curricular (UC) de Programação I, ministrada pelos docentes Vítor Távora e Pedro Gago, no primeiro ano de Engenharia Informática.

O projeto se trata de um programa, desenvolvido na linguagem de programação C, que simula um sistema de gestão de partilha de bicicletas para deslocação entre os campi do Instituto Politécnico de Leiria.

Os pontos analisados neste relatório diz respeito à representação gráfica tal qual a descrição das estruturas de dados que foram utilizadas para o processo de construção do programa. Levando em conta a justificação para a implementação das decisões desenvolvidas ou não.

O relatório está dividido em 3 secções: introdução, desenvolvimento e conclusão (com justificações do desenvolvimento e identificação de funcionalidades não desenvolvidas).

Para elaborar o código do programa utilizamos o CodeBlocks v.17.12.

**2. Desenvolvimento**

**Descrição das estruturas de dados utilizados**

* Data (tipoData): utilizada para armazenar datas e horas em variáveis inteiras, e consequentemente, na verificação das mesmas.

typedef struct{

int dia;

int mes;

int ano;

int hora;

int minuto;

}tipoData;

* Bicicleta (tipoBicicleta): caracteriza os dados das bicicletas do sistema, com variáveis de designação, modelo, status (Disponível, Emprestada ou Avariada) e do campus atual (Residências, Campus 1, Campus 2 ou Campus 5), armazenadas em forma de string. Também há variáveis para registrar a distância percorrida (em float), quantidades de avarias e quantidade de empréstimos (ambas em inteiros) registadas na bicicleta, são utilizadas posteriormente para estatísticas. O vetor dessa estrutura pode armazenar um máximo de 15 Bicicletas.

typedef struct{

char designacao[MAXSTRING];

char modelo[MAXSTRING];

char estado[MAXESTADO];

char campus[MAXSTRING];

int quantidadeEmprestimos;

int quantidadeAvarias;

float distanciaTotal;

}tipoBicicleta;

* Utente (tipoUtente): caracteriza os dados dos utentes do sistema, com variáveis de identificação do utente (número inteiro), nome (string), telemóvel (inteiro) e do tipo do utente (string: Estudante, Docente, Técnico Administrativo ou Convidado). As variáveis utilizadas para estatísticas são as de distância total percorrida pelo utente(em float) e quantidades de empréstimos realizados pelo mesmo (em inteiros). O vetor dessa estrutura pode armazenar um máximo de 50 utentes.

typedef struct{

int numero;

char nome[MAXSTRING];

int telemovel;

char tipo[MAXSTRING];

int quantidadeEmprestimos;

float distanciaPercorrida;

}tipoUtente;

* Empréstimo (tipoEmprestimo): estrutura utilizada para registrar os dados da deslocação de um utente. Utiliza as variáveis: número de registro (número inteiro, “ID” do empréstimo), o código do utente que realizou o empréstimo (número inteiro), a designação da bicicleta (string), a data de empréstimo e a data de devolução (utilizam a estrutura tipoData descrita anteriormente), o campus de origem e o campus de destino (ambos em string), e a distância percorrida durante o empréstimo (float). Funciona na forma de vetor de memória dinâmica.

typedef struct{

int numeroRegisto;

int codigoUtente;

char designacaoBicicleta[MAXSTRING];

tipoData dataEmprestimo;

char campusOrigem[MAXSTRING];

char campusDestino[MAXSTRING];

tipoData dataDevolucao;

float distanciaPercorrida;

}tipoEmprestimo;

* Espera (tipoEspera): estrutura utilizada para armazenar pedidos de empréstimos para quando não houver bicicletas disponíveis no campus selecionado. Possui as variáveis do código do utente que está em espera (número inteiro), a data de registo do pedido de espera (utiliza a estrutura tipoData descrita anteriormente), o campus de origem e o campus de destino (ambos armazenados em string). Funciona na forma de vetor de memória dinâmica.

typedef struct{

int codigoUtente;

tipoData dataRegistro;

char campusOrigem[MAXSTRING];

char campusDestino[MAXSTRING];

}tipoEspera;

**Constantes utilizadas:**

#define MAX 500

#define MAXSTRING 100

#define MAXESTADO 12

#define MAXBICICLETA 15

#define MAXEMPRESTIMO 100

#define MAXUTENTE 50

#define MAXTELEMOVEL 999999999

#define MAXDISTANCIA 100000000

#define MAXNUMEROUTENTE 9999999

#define MIN 1

#define MINTELEMOVEL 100000000

#define DIAMAX 31

#define MESMAX 12

#define ANOMIN 1950

#define ANOMAX 2050

#define MAXNOME 50

#define MAXHORAS 24

#define MAXMINUTOS 59

#define NUMMAX 10

#define NUMMIN 0

#define DIST\_R\_C1 1

#define DIST\_R\_C2 2

#define DIST\_R\_C5 3

#define DIST\_C1\_C2 4

#define DIST\_C1\_C5 5

#define DIST\_C2\_C5 6

**Representação gráfica das estruturas utilizadas:**

* Bicicleta (tipoBicicleta)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | ... | 13 | 14 |
| Designação  Modelo  Estado  Campus  Quantidade Empréstimos  Quantidade Avarias  Distância Total |  |  |  | Designação  Modelo  Estado  Campus  Quantidade Empréstimos Quantidade Avarias Distância Total |

* Utente(tipoUtente)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | .... | 48 | 49 |
| Número  Nome  Telemóvel  Tipo  Quantidade Empréstimos  Distância Percorrida |  |  |  | Número  Nome  Telemóvel  Tipo  Quantidade Empréstimos  Distância Percorrida |

* Empréstimo(tipoEmprestimo)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | ... |  | =contador de empréstimos |
| Número Registro  Código Utente  Designação Bicicleta  tipoData Data Empréstimo  Campus Origem  Campus Destino  tipoData Data Devolução  Distância Percorrida |  |  |  | Número Registro  Código Utente  Designação Bicicleta  tipoData Data Empréstimo  Campus Origem  Campus Destino  tipoData Data Devolução  Distância Percorrida |

* Lista de Espera(tipoEspera)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | ... |  | =contador de pedidos em espera |
| Código Utente  tipoData Data Registro  Campus Origem  Campus Destino |  |  |  | Código Utente  tipoData Data Registro  Campus Origem  Campus Destino |

**3. Conclusão**

Para concluir o relatório, justificaremos as opções de implementação tomadas, e identificaremos as funcionalidades não desenvolvidas.

A estrutura de menus foi implementada utilizando uma escolha para cada opção do menu, e em seguida, abrindo um submenu com o mesmo tipo de funcionamento.

A gestão de bicicletas, utentes, empréstimos e espera foi feita utilizando vetores das respectivas estruturas mencionadas e os dados inseridos são escritos e lidos em forma de ficheiro binário, assim quando o programa é aberto, os dados de um possível uso anterior estão preservados e se não, ficheiros são criados após o primeiro uso. Porém, para a lista de espera, os critérios de atribuição de bicicletas não foram desenvolvidos com sucesso, sendo assim, optamos por o utente escolher se quer ir para a lista de espera quando não há bicicletas no campus. Deste modo, quando um utente faz a devolução de uma bicicleta e há outro na lista de espera, a bicicleta é atribuída automaticamente conforme à ordem da lista.

Os registos de avarias e reparações estão sendo gravados em um log, na forma de ficheiro de texto.

As listagens e estatísticas foram desenvolvidas utilizando as variáveis de quantificação (criadas com objetivo de serem usadas para estatísticas e listagens) existente em cada estrutura. Estas funções percorrem os vetores, à busca das informações destas variáveis em cada elemento e assim, lista os dados de acordo com o selecionado.