|  |
| --- |
| Programação Orientada Por Objetos  Curso Técnico Superior Profissional  CTeSP  2018/2019 |
| Relatório Técnico |

Docente: Patrícia Dinis

Aluno: nº181210022 - Pedro Grilo

Índice

[1. Introdução 3](#_Toc12567847)

[2. Problema 3](#_Toc12567848)

[3. Solução proposta 4](#_Toc12567849)

[a. Descrição de aspetos particulares relevantes 5](#_Toc12567850)

[4. Utilização da aplicação 6](#_Toc12567851)

[a. Inserir nova ficha de empregado 7](#_Toc12567852)

[b. Pesquisa de empregados por código 7](#_Toc12567853)

[c. Inserir vários empregados 7](#_Toc12567854)

[d. Apresentar número total de empregados por categoria 7](#_Toc12567855)

[e. Apresentar todos os empregados 7](#_Toc12567856)

[f. Apresentar total de salários a pagar 7](#_Toc12567857)

[5. Discussão – análise da solução proposta 8](#_Toc12567858)

[6. Conclusão e trabalhos futuro 8](#_Toc12567859)

[7. Referências 8](#_Toc12567860)

[8. Anexos 8](#_Toc12567861)

# Introdução

Este relatório foi realizado no âmbito da disciplina **de Progamação Orientada Por Objetos**, com o **objetivo de descrever o problema que foi construído, ou seja, a Gestão de Recursos Humanos**

Tem também como objetivo mostrar as soluções propostas para o problema, por exemplo, como a Gestão de Recursos Humanos foi resolvida ou com que programa foi criada, descrever um resumo da utilização da aplicação, uma discussão da análise da solução proposta, e uma conclusão com objetivos de futuros trabalhos.

Os conceitos fundamentais da programação orientadas por objetos está bastante alicerçada na solução que se pretende que responda às necessidades do projeto, nomeadamente os conceitos de classe, objeto, abstração de dados, proteção e encapsulamento. Para além disto estão presentes os conceitos avançados tais como o polimorfismo, hierarquia, herança e agregação.

# Problema

O objetivo deste problema é desenvolver uma aplicação, utilizando a linguagem **JAVA**, de forma a implementar uma aplicação que gerisse os recursos humanos de uma empresa. No problema, destacam-se as seguintes entidades:

* **A Empresa;**
* **Os Tipos de Empregados;**
* **Ficheiros**
* **JavaFX**

# Solução proposta

Este **projeto** foi dividido em **duas** **fases**:

* A **primeira** **fase**, consistia em realizar o projeto em consola
* E a **segunda** **fase,** consistia em utilizar os recursos feitos da primeira fase e implementares num interface gráfica com o **JavaFX**.

O **projeto** na **primeira fase** foi realizado inicialmente pela base do programa, ou seja, criar as **classes com as respetivas interfaces, menus, opções, etc**. Após concluir, de seguida, foram feitas as opções do **Menu Gestão**, pois era onde continha a maior parte do código, tal como, Inserir uma nova ficha de empregado; Verificar se existe algum empregado com um código dado como parâmetro; Obter a ficha de empregado de código dado como parâmetro, se existir; Dada uma lista de empregados contratados, inseri-la no registo atual; Determinar o número atual de empregados de uma dada categoria; Devolver o conjunto atual de fichas de empregados; Devolver o conjunto atual de fichas de empregados, filtrados por categoria; Calcular, a qualquer momento, o total de salários a pagar; Calcular e apresentar os custos trimestrais, semestrais e anuais com salários, devendo ter em conta os subsídios de natal e de férias;

**Após concluir a etapa anterior** e em seguimento com a aula de exceções, começaram a ser **criadas as exceções, com o respetivo enumerado onde contém todos os erros**.

**Para testes e afins**, foi criado **o Menu Empregados**, onde era possível registar o dia trabalhado, introduzir o valor de vendas num determinado mês ou até mesmo adicionar a distância percorrida.

De seguida, foi feito a parte dos **ficheiros**, onde guarda o **objeto** **empresa** num ficheiro e após o início do programa, este é lido automaticamente, e com isto damos como concluído este tópico: Guardar a lista atual de empregados num ficheiro de texto;

Para **finalizar** a **primeira** **fase** do projeto, foi analisado o código todo, corrigindo os *bugs*, e organizando e otimizando o código.

Na **segunda fase**, o objetivo era criar uma interface gráfica utilizando o **JavaFX.**

Inicialmente, começou por ser feito a maquete da interface numa folha A4, para mais tarde ser copiada para o *Scene Builder* (programa que ajuda a desenvolver a interface).

No *Scene Builder*, foi utilizado um pacote (.**JAR**), que continha estilos de botões, caixa de textos, entre outros. Este pacote encontra-se na pasta do projeto, para ser incluído no **IDE** do programador.

Após montar o *design*da aplicação, foi a parte de fazer a programação dos objetos. Com isto aplicou-se os mesmos métodos de desenvolvimento da primeira fase (programação dos itens dos menus, exceções, ficheiros e correção de *bugs).* Esta parte foi mais rápida de desenvolver, pois os métodos já estavam praticamente todos feitos.

## Descrição de aspetos particulares relevantes

O uso de ficheiros para armazenar os empregados da empresa no disco rígido;

Utilização do JavaFX e do *Scene Builder*para a realização de uma interface gráfica de modo que o utilizador



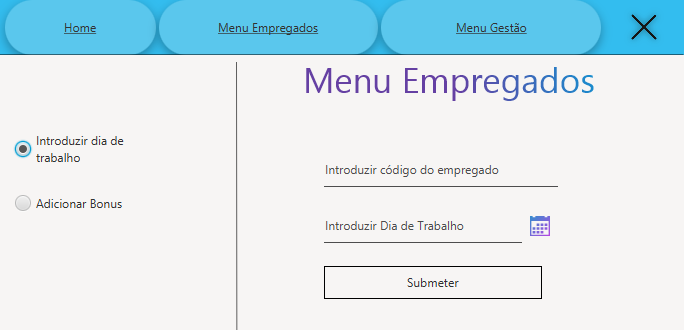
Fig1. Menu Inicial

# Utilização da aplicação

Para utilização da aplicação, é recomendado o sistema operativo *Windows*.

Após a abertura da aplicação, o utilizador depara-se com um **Menu Inicial**, onde lhe é apresentado duas opções, as quais são:

* **Menu Empregados;**
* **Menu Gestão;**

Se o utilizador escolher a opção **Menu Empregados,** é-lhe apresentado uma janela onde encontrará duas opções, as quais são:

* **Introduzir dia de trabalho;**
* **Adicionar Bónus**

Se o utilizador selecionar a primeira opção (**Introduzir dia de trabalho**), o utilizador acede ao menu**,** onde será possível introduzir o código do empregado e o seu dia de trabalho.

Fig2. Menu Empregados

Se o utilizador selecionar a segunda opção (**Adicionar Bónus**), o utilizador acede ao menu, que serve para o utilizador adicionar vendas caso seja um empregado comercial ou adicionar distancia se for motorista.

Se o utilizador selecionar a opção (**Menu Gestão**), o utilizador terá acesso a um **submenu**, onde será possível por exemplo: **Inserir uma nova ficha de empregado, apresentar todos os empregados, ou até mesmo ver o total de salários a pagar.**



Fig3. Menu Gestão

## Inserir nova ficha de empregado

Neste **menu** é possível **inserir um novo empregado**, escolhendo inicialmente o tipo de empregado, de seguida o código, o nome e a data de entrada na empresa.

## Pesquisa de empregados por código

Neste **menu** pode se **pesquisar um empregado e ter acesso à ficha do mesmo**, introduzindo o código do empregado.

## Inserir vários empregados

**É possível inserir vários empregados de uma só vez**, pois vai se adicionando a uma lista temporária no final é que se confirma.

## Apresentar número total de empregados por categoria

É apresentado um **gráfico de barras** com a **quantidade** de **empregados em cada categoria**.

## Apresentar todos os empregados

Neste **menu, existe uma lista com todos os empregados**, onde é possível ver a **ficha** de **cada** **empregado** selecionado. É possível também pesquisar por código ou por nome.

## Apresentar total de salários a pagar

É **possível ver o total de salário a pagar por mês, trimestre ou semestre** conforme o ano selecionado.

# Discussão – análise da solução proposta

O facto de terem sido utilizados **ficheiros**, **facilita** **bastante**, não só para o **programador**, como também ao **utilizador**, pois desta forma o utilizador não tem de se registar os empregados sempre que entra no programa. Para o programador, torna-se muito mais fácil testar o programa em si e controlar, o que acontece com o programa em geral.

Infelizmente com tantas funcionalidades, vem **uns pequenos inconvenientes** que é o facto de o ficheiro irá ocupar espaço no computador, porem não é muito. Outro **inconveniente** é que se o utilizador alterar algo no ficheiro, o mesmo fica corrompido. Caso seja preciso alterar algo manualmente, é bastante complicado, sendo que o ficheiro é escrito em binário e os leitores de texto comuns não conseguem ler binário com facilidade mostrando muitas vezes caracteres especiais no seu lugar.

# Conclusão e trabalhos futuro

Com este trabalho, foi possível retirar várias **conclusões**, como a aprendizagem e **utilização** de **ficheiros binários, excecçoes, interfaces, o uso do JavaFX,** e **aprendizagem** **autónoma** do programa *Scene Builder****.***

Conclui-se também**, que não foi um trabalho fácil,** pois houve **várias** **verificações**, as quais deram algum trabalho, **a aprendizagem do *Scene Builder***  e a manipulação de ficheiros, mas após algum estudo obtive sucesso ao utilizá-los.

Concluindo, em futuros trabalhos teremos de **melhorar o código fonte**, para um código mais **reduzido e mais eficiente**, e **uma interface gráfica mais aperfeiçoada.**

# Referências

<https://stackoverflow.com/>

<https://netbeans.org/>

<https://www.jetbrains.com/idea>

<https://www.geeksforgeeks.org>

<https://github.com/jfoenixadmin/JFoenix>

<https://gluonhq.com/products/scene-builder/>

# Anexos

jfoenix-8.0.8.jar – pacote de estilos do *Scene Builder*