# Requisitos

O presente documento expõe as funcionalidades do programa criado no decorrer deste trabalho prático, decorrente da disciplina Paradigmas da Programação.

O programa criado tem como principal objectivo a seriação de Candidaturas, realizadas por *Freelancers,* a um Anúncio de uma determinada Tarefa. Cada Tarefa é criada por um Colaborador o qual pode criar mais do que uma Tarefa. Existem dois métodos de Seriação, Seriação1 e Seriação2, cujos critérios de ordenação estão definidos no enunciado deste trabalho prático.

Ao correr a aplicação o utilizador é saudado com uma janela (Figure 1), de cabeçalho "Tarefas". Esta janela possui uma caixa, onde poderá escolher 1 colaborador dentro de uma lista, um botão com o nome "Candidaturas" e uma tabela vazia com os detalhes/atributos de cada Tarefa.

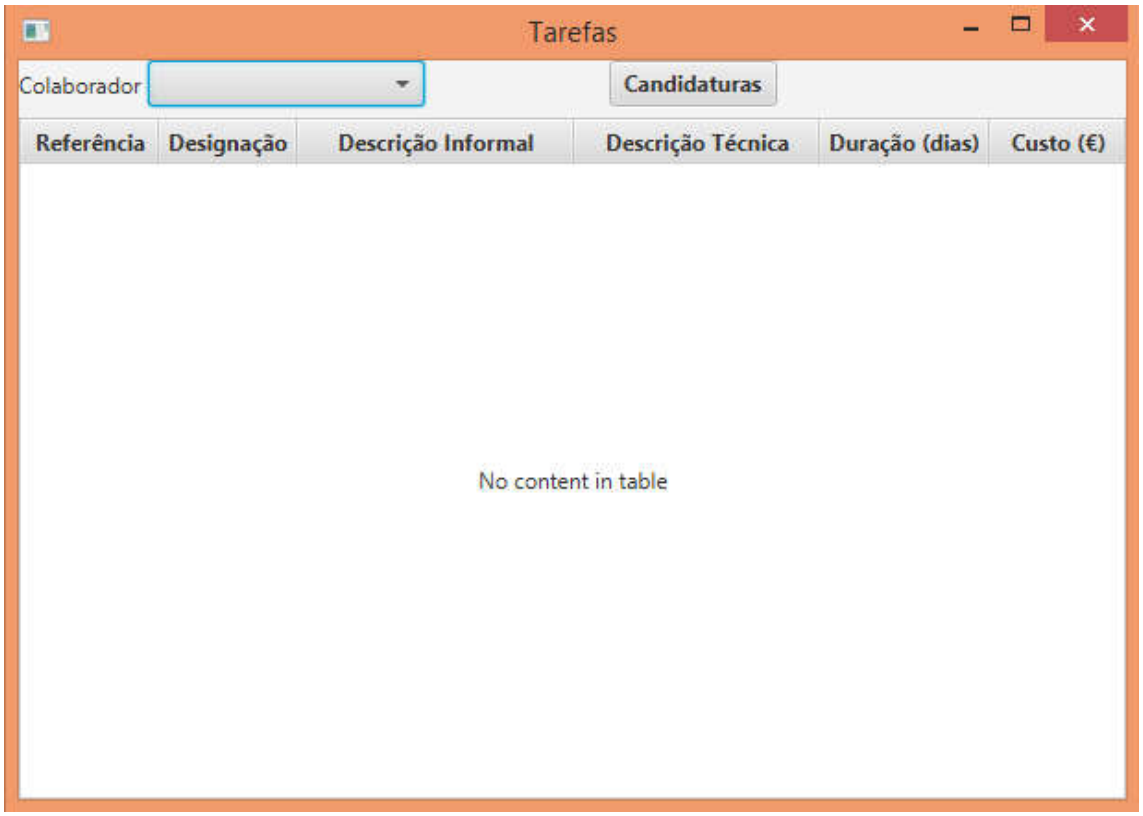



Figure 1 Janela de Tarefas vazia.

Após escolher um colaborador, a tabela imediatamente abaixo é actualizada com as tarefas que o colaborador possui activas (Figure 2), podendo seleccionar uma tarefa para apresentar as candidaturas realizadas à mesma.

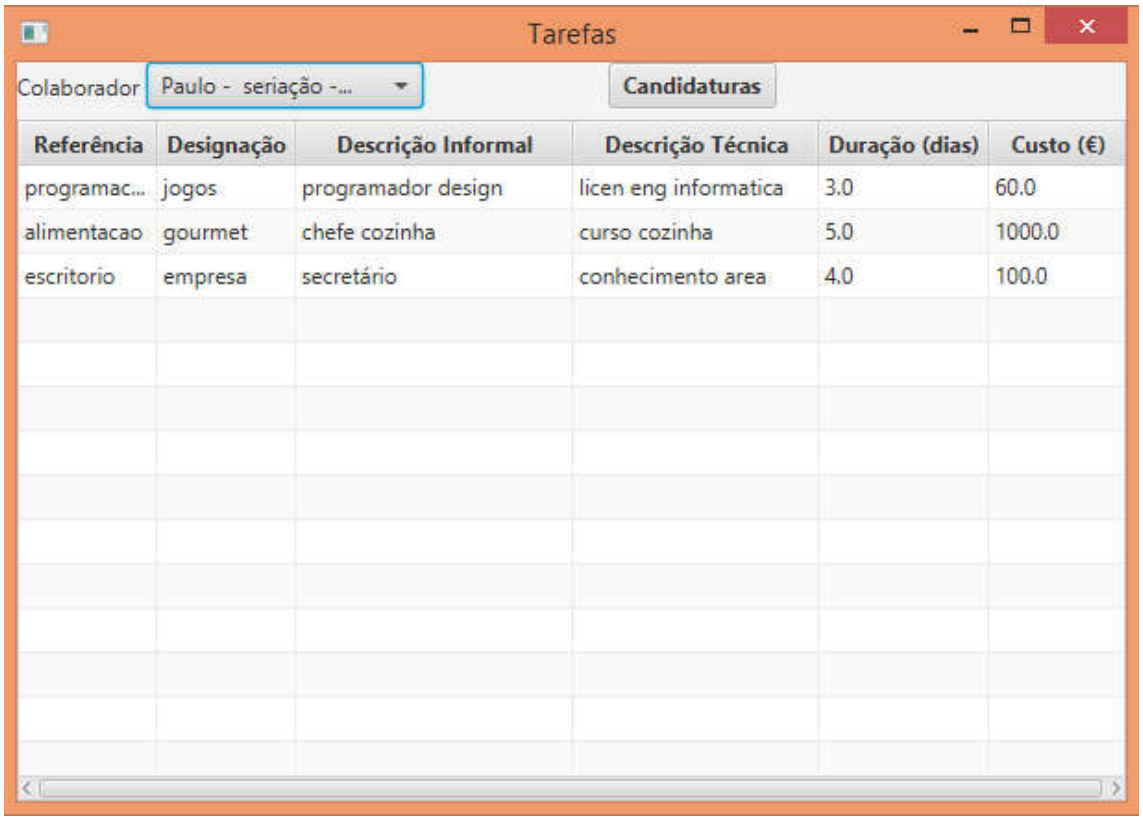



Figure 2 Janela de tarefas preenchida.

Caso o utilizador carregue no botão "Candidaturas" antes de seleccionar um colaborador da caixa do canto superior esquerdo ou antes de seleccionar uma candidatura, o utilizador é saudado com uma janela de alerta para a situação indicada (Figure 3)

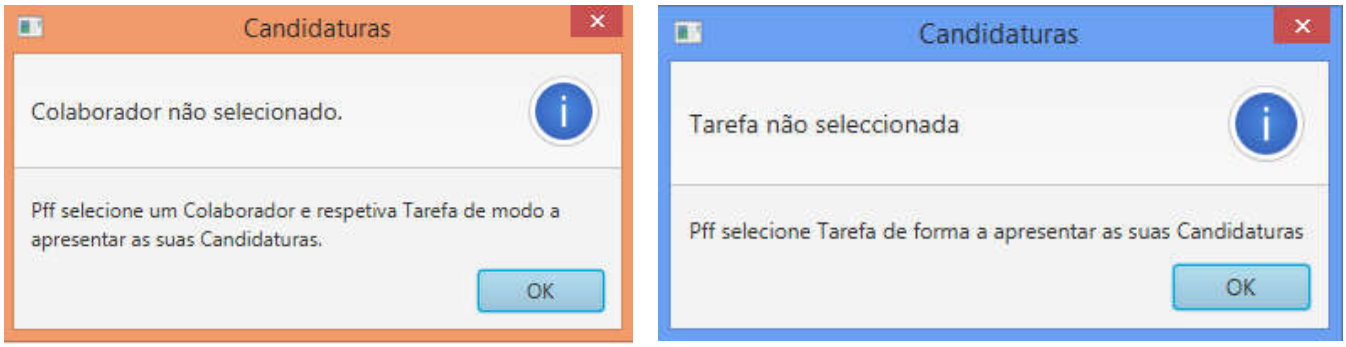



Figure 3 Janelas de erro para a não selecção do colaborador e respectiva tarefa.

Após correta selecção de um colaborador e de uma tarefa, estão reunidas as condições para serem apresentadas ao utilizador todas as candidaturas existentes à tarefa seleccionada, sendo este processo despoletado pelo pressionar do botão “Candidaturas”. O utilizador é saudado com uma nova janela de seu título “Candidaturas” onde poderá ver uma caixa de texto com a designação da tarefa previamente escolhida, uma tabela já preenchida com os atributos de cada candidatura e um botão com o nome “Seriar”.

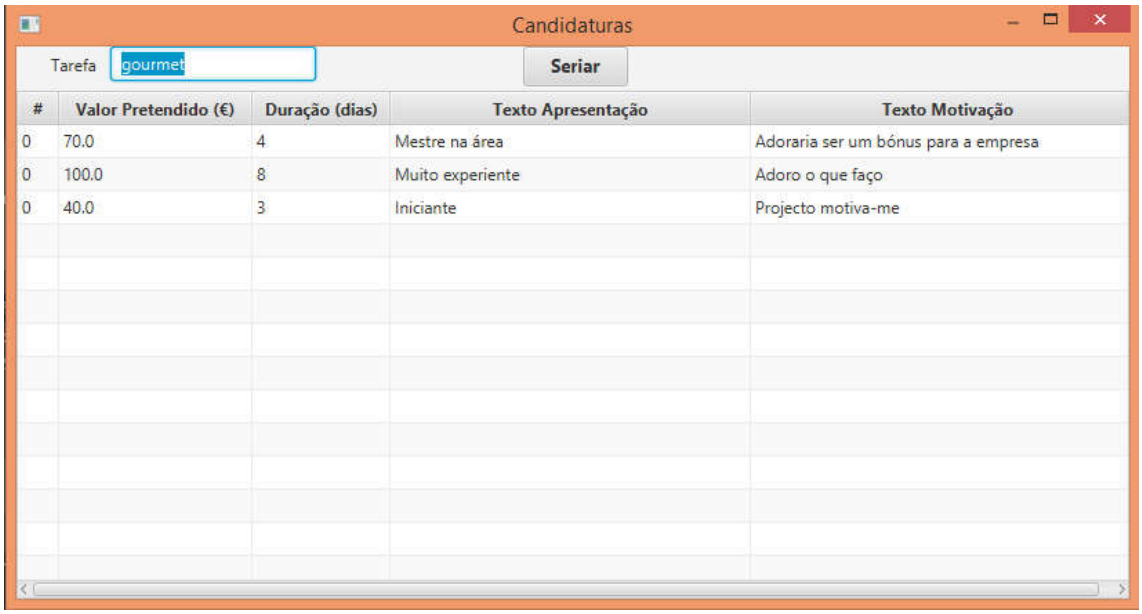



Figure 4 Janela de visualização das candidaturas à tarefa seleccionada.

De notar que na tabela, a primeira coluna mais à esquerda, indicativa da classificação de cada candidatura após seriação, está pré-preenchida com zeros dado que as candidaturas ainda não foram seriadas.

Nesta fase, o utilizador está já em condições para seriar as candidaturas, apenas necessita de seleccionar o tipo de regimento/critério de seriação a ser utilizado. Tendo em vista este objectivo, o utilizador ao carregar no botão “Seriar” será saudado com uma pequena janela onde poderá escolher uma seriação, de entre uma lista previamente existente (Figure 5)


Figure 5 Janela de selecção do regimento de seriação.

A designação da tarefa está também presente nesta janela para informar o utilizador de qual tarefa está realmente a realizar a seriação. Após escolha, ou não, do tipo de seriação o utilizador pode carregar tanto no botão “Seriar” como no botão “Cancelar” sendo que ambos fazem desaparecer a janela corrente e retornam para a janela anterior de visualização das candidaturas à tarefa seleccionada, com a diferença de que a seriação será realizada caso carregue no botão “Seriar”.

Caso o utilizador opte por “Seriar”, a tabela da janela de visualização das candidaturas é actualizada, ou seja, as candidaturas serão apresentadas ordenadas pela classificação (Figure 6).

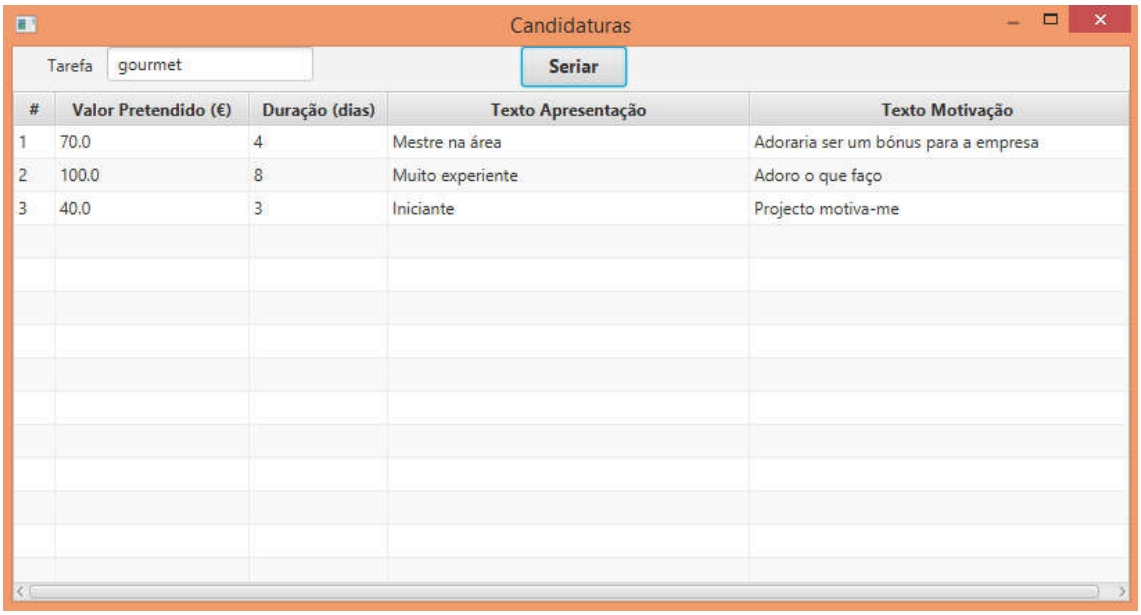



Figure 6 Janela de visualização das candidaturas seriadas à tarefa seleccionada.

Ao fechar-se esta janela voltamos para a janela principal onde o utilizador pode escolher outra tarefa ou outro colaborador e respectiva tarefa a seriar.

Para que estas funcionalidades fossem possíveis houve necessidade de se criarem vários métodos em algumas classes específicas podendo identificar-se os seguintes: todas as colunas das tabelas apresentadas necessitaram de métodos *ge*t de forma a possibilitarem a recolha do respectivo atributo a ser impresso na tabela,

Para possibilitar a realização das seriações às candidaturas de uma determinada tarefa e de forma a utilizar a interface *Comparator* do java, foi necessário criar 2 classes de Seriação (Seriacao1 e Seriacao2) que implementassem tanto a interface *Comparator* como a interface *Seriavel* criada exclusivamente para este projeto. Esta interface *Seriavel* define um contrato

para a implementação do método *compare(Candidatura,Candidatura)* com todas as classes que a implementem. Este método *compare* é essencial para a utilização da interface *Comparator* do java pois substitui o método *compare* quando chamado sobre um objecto, neste caso do tipo Candidatura.

Dentro da classe *Candidatura* do foram implementados os métodos *mediaGrau* e *desvioPadrao* que calculam e retornam, respectivamente, a média e o desvio padrão dos graus de proficiência em cada uma das competências técnicas do Freelancer ao qual pertence a candidatura. Estes dois métodos serão utilizados na implementação dos métodos compare nas classes que implementam a interface *Seriavel.*

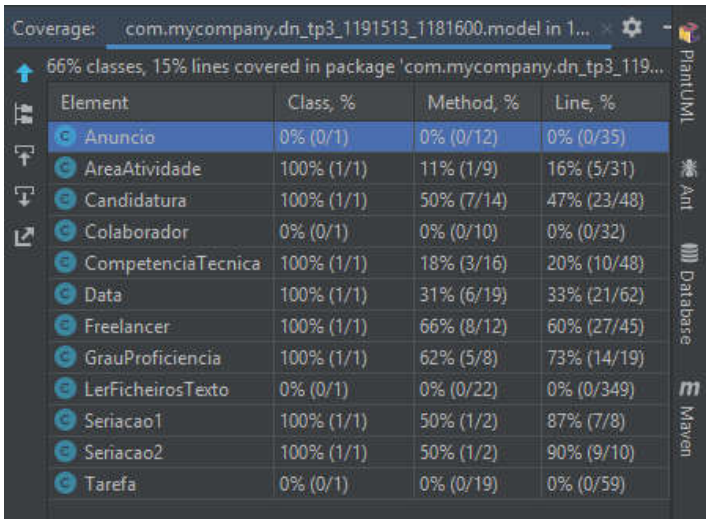
Para finalizar, o método *compare* da classe *Seriacao1* foca-se, numa primeira instância na comparação da *mediaGrau*, depois no custo e após este na data da candidatura. O método *compare* na classe *Seriacao2* foca-se primeiro na comparação da *mediaGrau*, depois no *desvioPadrao*, logo após no custo e por fim na data da candidatura (Figure 7).

```java
@Override
public int compare(Candidatura c1, Candidatura c2) {
    int c = Double.compare(c2.mediaGrau(), c1.mediaGrau());
    if (c == 0) {
        c = Double.compare(c1.getCusto(), c2.getCusto());
        if (c == 0) {
            c = c1.getDataCand().compareTo(c2.getDataCand());
        }
    }
    return c;
}
```

```java
@Override
public int compare(Candidatura c1, Candidatura c2) {
    int c = Double.compare(c2.mediaGrau(), c1.mediaGrau());
    if (c == 0) {
        c = Double.compare(c1.desvioPadrao(), c2.desvioPadrao());
        if (c == 0) {
            c = Double.compare(c1.getCusto(), c2.getCusto());
            if (c == 0) {
                c = c1.getDataCand().compareTo(c2.getDataCand());
            }
        }
    }
    return c;
}
```

Figure 7 Métodos compare das classes Seriacao1 (esquerda) e Seriacao2 (direita).

Relativamente aos testes unitários implementados para o programa desenvolvido, a Figure 8 mostra as percentagens de código cobertas para as diferentes classes, sendo que as classes de Seriação se aproximam bastante dos 100% dada a sua importância no trabalho.




| Element | Class, % | Method, % | Line, % |
|---|---|---|---|
| Anuncio | 0% (0/1) | 0% (0/12) | 0% (0/35) |
| AreaAtividade | 100% (1/1) | 11% (1/9) | 16% (5/31) |
| Candidatura | 100% (1/1) | 50% (7/14) | 47% (23/48) |
| Colaborador | 0% (0/1) | 0% (0/10) | 0% (0/32) |
| CompetenciaTecnica | 100% (1/1) | 18% (3/16) | 20% (10/48) |
| Data | 100% (1/1) | 31% (6/19) | 33% (21/62) |
| Freelancer | 100% (1/1) | 66% (8/12) | 60% (27/45) |
| GrauProficiencia | 100% (1/1) | 62% (5/8) | 73% (14/19) |
| LerFicheirosTexto | 0% (0/1) | 0% (0/22) | 0% (0/349) |
| Seriacao1 | 100% (1/1) | 50% (1/2) | 87% (7/8) |
| Seriacao2 | 100% (1/1) | 50% (1/2) | 90% (9/10) |
| Tarefa | 0% (0/1) | 0% (0/19) | 0% (0/59) |

Figure 8 Java Code Coverage dos testes unitários implementados.