Análise e Teste de Software:
O Sistema de Software TrazAqui!

DI, Universidade do Minho

Ano lectivo 2020/2021

João Saraiva & José Nuno Macedo

1 TrazAqui: Sistema de Software

No ano lectivo 2018/2019, no contexto da disciplina de *Programação Orientada a Objectos* (POO) leccionada no Departamento de Informática da Universidade do Minho, os alunos tiveram de desenvolver em grupo uma aplicação Java para gerir um serviço de entrega de encomendas ao domicilio. O enunciado desse trabalho, onde se descrevem todos os requisitos do sistema de software a desenvolver, encontra-se anexado a este documento.

No contexto da disciplina de Análise e Teste de Software (ATS) pretende-se que os alunos apliquem as técnicas de análise e teste de software, estudadas nas aula, de modo a analisar a qualidade das soluções desenvolvidas pelos alunos de POO[1]. Assim, os alunos de ATS deverão:

- Analisar a qualidade do código fonte do(s) sistema(s) de software. Nesta análise os alunos deverão identificar *bad smells* no código fonte e o seu *technical debt*.
- Aplicar refactorings de modo a eliminar os *bad smells* encontrados, e deste modo reduzir (eliminar?) o *technical debt*.
- Testar o software de modo a ter mais garantias que ele cumpre os *requisitos* do enunciado da aplicação TrazAqui.
- Gerar inputs aleatórios para a aplicação TrazAqui que simulem execuções reais (tal como foi fornecido em POO)
- Analisar a performance (tempo de execução e consumo de energia) da versão inicial do software (i.e., com *smells*) e a obtida depois de eliminados smells.

Tarefa 1: Qualidade do Código Fonte da Aplicação TrazAqui

Nesta tarefa os alunos irão utilizar sistemas como o SonarQube, IDEs do Java, etc para analisar a qualidade do código da aplicação desenvolvida pelos alunos de POO. Métricas de software serão também utilizadas para encontrar *bad smells*.

[1] As soluções desenvolvidas por alunos de POO estão disponíveis na página de ATS.

Tarefa 2: Refactoring da Aplicação **TrazAqui**

Nesta tarefa serão utilizadas ferramentas como o autorefactor, IDEs do Java que suportam refactoring, ou o jStanley para identificar e eliminar os *bad smells* e *red smells* existentes no software fornecido.

Um estudo detalhado sobre os *smells* encontrados na(s) aplicação(ões) fornecidas, os refactorings aplicados e o *technical debt* obtidos deverão ser incluídos no relatório.

Tarefa Extra: A tarefa 1 e 2 analisam estaticamente o código fontes das aplicações desenvolvidas pelos alunos. Uma vez que existem dezenas de soluções os alunos deverão automatizar essas duas tarefas de modo a não terem de analisar manualmente cada uma delas. Assim, devem definir scripts que (em *batch*) fazem a deteção e refactoring de todas as aplicações. Estas scripts, para cada solução, devem produzir uma nova versão (com smells eliminados), e ainda as métricas calculadas (smells, complexidade ciclomática, etc). As métricas de cada solução devem depois ser integradas num único ficheiro (csv, Excel), para permitir uma análise global dos resultados.

Tarefa 3: Teste das Aplicações **TrazAqui**

Nesta tarefa serão utilizadas técnicas de teste de software, para efectuar o teste unitário e o teste de regressão da aplicação TrazAqui. Serão ainda utilizadas sistemas para a geração automática de casos de teste para gerar testes unitários e ainda inputs para simular a execução real da aplicação.

Nesta tarefa deverão ser realizadas as seguintes sub-tarefas:

- Utilizar o sistema EvoSuite para gerar automaticamente testes unitários para a aplicação TrazAqui.
- Utilizar o sistema JaCoCo, ou qualquer outro suportado por IDEs do Java, para analisar a cobertura dos testes.
- Gerar automaticamente ficheiros de logs da aplicação TrazAqui, utilizando preferencialmente o sistema *QuickCheck*. Um ficheiro de logs exemplo é disponibilizado com a aplicação. O gerador a desenvolver deve gerar ficheiros de logs segundo a notação facilmente identificada nesse exemplo. O gerador deve ainda poder ser parameterizado com alguma noção de dimensão dos logs a gerar, de modo a ser possível exercitar/monitorizar depois a aplicação TrazAqui com inputs diferentes.

Tarefa 4: Análise de Desempenho da Aplicação **TrazAqui**

Nesta tarefa será feita uma análise detalhada do desempenho da aplicação TrazAqui, nomeadamente em termos de tempo de execução e consumo de energia. A aplicação será executada utilizando a framework de geração de inputs reais definidos na Tarefa 3. Duranta a execução da aplicação o tempo e consumo de energia serão monitorizados.

Tarefa Extra: De modo a analisar a influência dos *bad smells* e *red smells* no desempenho do software as diferentes versões (com e sem *smells*) do software TrazAqui serão comparados. Inputs de tamanhos diferentes deverão ser também considerados na análise.

2 Entrega e Avaliação

A entrega do projeto será feita no dia 10/01/2021. Na entrega pretende-se que os projetos apresentem o projeto oralmente (por exemplo, usando slides), e mostrem a ferramenta construída a funcionar.

A avaliação do projeto terá em conta as seguintes componentes:

Tarefa 1:	10%
Tarefa 2:	10%
Extras 1-2:	10%
Tarefa 3:	20%
Tarefa 4:	20%
Extra 4:	10%
Apresentação/Relatório/Defesa:	20%

POO (MiEI/LCC)

2019/2020

Enunciado do Trabalho Prático

Conteúdo

1 TrazAqui! **1**

2 Encomendas **3**

3 Sobre o tempo de viagem no transporte de encomendas **3**

4 Requisitos básicos **3**

5 Teste de Sistema **4**

6 Relatório **4**

7 Salvaguarda do estado da aplicação **5**

8 Patamares de classificação do projecto **5**

9 Cronograma **5**

1 TrazAqui!

Um grupo de alunos de POO pensou em criar uma aplicação que permita conjugar as actuais necessidades de entregar encomendas a pessoas que estão confinadas nas sua habitações e necessitam de um serviço que permita:

1. ter voluntários que façam entregam de encomendas a um utilizador (que se encontra na sua habitação)
2. ter empresas que façam entregas de encomendas
3. ter lojas que adiram a este serviço para poderem ser entregues as encomendas aos utilizadores

Nas lojas, a exemplo do que acontece actualmente com o projecto `www.posso-ir.com` podemos ter lojas que alimentam esta informação sobre o tamanho das filas existentes e outras lojas que não alimentam esta informação.

Cada um dos voluntários está localizado num determinado sítio (de onde temos as suas coordenadas GPS em termos de latitude e longitude) e indica qual é o raio geográfico no qual se movimenta (por exemplo 5kms). Em função das encomendas para entrega o sistema escolherá um voluntário que esteja livre e que poderá aceder à loja por esta se encontrar dentro do seu raio de acção.

As empresas de entregas possuem também uma localização de origem e um raio de acção, mas associam o transporte do serviço a um custo. O custo será calculado em função do peso da encomenda e da distância à loja e depois à casa do utilizador, bem como da taxa que cada empresa de transportes cobre por km. É possível desde já perspectivar que existam empresas de transporte que aceitem apenas uma encomenda e outras que tenham capacidade para transportar `N` encomendas e tenham a noção de rota.

Quer os voluntários quer as empresas de transporte guardam sempre o registo das encomendas que transportaram, para que utilizador do serviço é que se destinavam e a loja onde a foram recolher.

Tal como em outros serviços semelhantes o sistema guarda os pedidos de encomenda, a levantar de determinada loja, que são feitos pelos utilizadores. Posteriormente os voluntários e as empresas podem escolher os serviços de entrega que pretendem efectuar e o sistema, determina se aceita, ou não, essa intenção. No caso das empresas, e uma vez que existe um custo envolvido, terá de ser o utilizador a validar que quer que a encomenda seja de facto transportada por aquela empresa. No caso das lojas que possuem informação de fila de espera é feita uma estimativa do tempo que demora a recolher a encomenda. Essa estimativa tem em consideração o tempo que se percorre do sítio onde está quem recolhe a encomenda até à loja (é necessário saber a que velocidade se desloca) e o tempo de espera em função da fila de espera da loja. Para tal as lojas deverão estimar em média o tempo de atendimento habitual. Nas lojas onde não existe informação de fila de espera assume-se que os voluntários e as empresas transportadoras assumem o risco de poder passar mais tempo a ser atendidos e tal, no caso das empresas, não leva a aumento da tarifa de transporte.

Pretende-se que a aplicação a ser desenvolvida dê suporte a toda esta funcionalidade. O processo deve abranger todos os mecanismos de criação de utilizadores, voluntários, empresas transportadoras e lojas e posteriormente a mecânica de selecção das encomendas a transportar e a imputação do preço quando o mesmo é devido (no caso das empresas de transporte). Pretende-se ainda que o sistema guarde registo de todas as operações efectuadas e que depois tenha mecanismos para as disponibilizar (exemplo: entregas para um utilizador, extracto de viagens de uma empresa num determinado período, valor facturado por uma empresa num determinado período, etc.).

Cada perfil deve apenas conseguir aceder às informações e funcionalidades respectivas.

- Os utilizadores da *TrazAqui! - Serviço de Entregas Encomendas em Casa* poderão:
  - solicitar a entrega de uma encomenda que foi pedida a uma loja;
  - aceitar, ou não, o serviço de entrega proposto por uma empresa transportadora (os serviços feitos por voluntários são automaticamente aceites pelo sistema);
  - aceder à informação das entregas efectuadas num determinado período e por voluntário ou transportador;
  - classificar o voluntário ou a empresa de transportes mediante o grau de satisfação com o serviço;
- Os voluntários poderão:

  - sinalizar que estão dispostos para recolher encomendas;
  - escolher ir buscar uma encomenda de um utilizador que é disponibilizada por uma loja;
  - fazer o transporte da encomenda e registar quanto tempo demorou;

- As empresas transportadoras poderão:
  - sinalizar que estão dispostos para recolher encomendas;
  - determinar o preço de transporte de uma encomenda em função da distância e do tempo de espera na loja (estimado ou fornecido pela loja);
  - fazer o transporte da encomenda e registar quanto tempo demorou e o custo associado;

- As lojas poderão:
  - sinalizar que existe uma encomenda de um utilizador para ser entregue;
  - se possível, indicar a quantidade de pessoas em fila para serem atendidas;

2 Encomendas

Sobre as encomendas além de guardarem informação sobre a quem se destinam e quem as vendeu, possuem também atributos que identificam o peso (que é utilizado para determinar o valor de transporte por parte das transportadoras) e os produtos transportados. Deverá ser prevista a existências de `Encomendas Médicas` que só poderão ser transportadas por voluntários ou empresas certificados para o efeito. Voluntários e transportadores que estão certificados para as transportar podem, se assim o pretenderem, transportar estas encomendas. Para tal tem disponível o método `public boolean aceitoTransporteMedicamentos()` que determina se no momento aceitam transporte de encomendas de medicamentos. Para mudar o estado de aceitação desse tipo de encomendas terá de ser implementado o método `public void aceitaMedicamentos(boolean state)`.

3 Sobre o tempo de viagem no transporte de encomendas

Como factor de diferenciação na solução que cada grupo de POO vai encontrar para resolver este problema, podem pensar que o tempo de viagem (e também o custo no caso das empresas transportadoras) além de ser função de um factor associado à distância pode também ser função das condições atmosféricas (é uma sugestão!) ou condicionantes relacionados com o tamanho da fila de espera. Nesse caso será necessário acrescentar informação a passar nos métodos que suportem estes requisitos.

4 Requisitos básicos

Identificam-se, de seguida, os requisitos básicos que o programa deverá cumprir: O programa deverá ainda fornecer uma interface de utilizador (em modo texto) que permita o acesso às funcionalidades.

- o estado da aplicação deverá estar pré-carregado com um conjunto de dados significativos, que permita testar toda a aplicação no dia da entrega.

- registar um utilizador, um voluntário, uma empresa transportadora e uma loja;
- validar o acesso à aplicação utilizando as credenciais (email e password);
- inserir pedidos de encomendas a uma loja, por parte de um utilizador;
- inserir informação de encomenda pronta a ser entregue, por parte das lojas;
- indicar que se pretende transportar a encomenda. No caso de ser uma empresa transportadora o utilizador que solicitou o serviço terá de aprovar que aceita o preço. Tanto voluntários como empresas apenas podem fazer transporte de encomendas de acordo com o raio de acção que tiverem definido;
- classificar, por parte do utilizador, o serviço de entrega;
- ter acesso no perfil de cada uma das entidades à informação sobre as encomendas transportadas (em função de data/hora de transporte);
- indicar o total facturado por uma empresa transportadora num determinado período;
- determinar a listagens dos 10 utilizadores que mais utilizam o sistema (em número de encomendas transportadas);
- determinar a listagens das 10 empresas transportadoras que mais utilizam o sistema (em número de kms percorridos);
- gravar o estado da aplicação em ficheiro, para que seja possível retomar mais tarde a execução.

5 Teste de Sistema

Será fornecido um ficheiro de *logs* que deverá ser carregado inicialmente e que conterá dados para teste.

6 Relatório

O relatório deve descrever o trabalho realizado para desenvolver a aplicação solicitada. No mínimo, devem ser abordados os seguintes pontos:

- Capa com identificação da Unidade Curricular e do grupo.
- Descrição da arquitectura de classes utilizada (classes, atributos, etc.) e das decisões que foram tomadas na sua definição. Deverá ser entregue um **Diagrama de Classes** com a arquitectura de classes que suporta o programa desenvolvido.
- Descrição da aplicação desenvolvida (ilustração das funcionalidades).

7 Salvaguarda do estado da aplicação

O programa deve permitir que em qualquer momento se possa guardar em ficheiro a informação existente em memória sobre a informação relevante das entidades. A gravação deve ser feita de forma a permitir que o estado que foi gravado seja recuperado novamente. Na altura da entrega do projecto deve ser também entregue um estado (guardado em ficheiro) que possa ser carregado durante a apresentação.

8 Patamares de classificação do projecto

Este projecto de POO tem previstos dois patamares de dificuldade em função dos requisitos anteriormente identificados:

1. Requisitos básicos de criação de utilizadores, voluntários, empresas e lojas e registo de transporte de encomendas: nota máxima 16 valores;

2. Itens anteriores mais gestão de factores de aleatoriedade, na duração da viagem e tempo de espera, e gestão dos diferentes tipos de encomendas: nota máxima 20 valores

Para que um projecto seja considerado positivo (com nota maior ou igual a 10) será necessário que se consiga efectuar, durante a apresentação, o registo e posterior consulta do transporte de uma encomenda. Obviamente que a nota a atribuir deverá reflectir a estruturação da solução que deverá respeitar as normas da programação orientada aos objectos, cf aulas teóricas, nomeadamente o encapsulamento, a abstração de implementação e a capacidade de a aplicação evoluir de forma controlada (eg: através da incorporação de novos tipos de entidades que fazem transportes ou da definição de novos tipos de encomendas).

9 Cronograma

A entrega do projecto far-se-á de forma faseada, nas seguintes *milestones*:

1. Entrega de projecto com uma versão inicial das declarações das classes (pelo menos com as variáveis de instância) e um diagrama de classes. Entrega da composição do grupo.
   Data Limite: 1 de Maio (esta fase é eliminatória, isto é, os grupos que não entregarem não poderão submeter o projecto final).

2. Entrega final de código e relatório de projecto (feita por via electrónica no elearning)
   Data Limite: *data a definir com a direcção de curso*

3. Apresentação *presencial* do projecto
   Semana de: *data a definir com a direcção de curso*