Trabalho T2: Modelagem em Análise e Projeto

Pedro Cardoso, Fernando Giacomini, Felipe Nomura, Alison Carnetti

Link do GitHub com Video e Projeto

Introdução:

Levando em conta os requisitos do trabalho a ser feito foi desenvolvido um programa próprio para serem usadas as ferramentas solicitadas. Mesmo com um programa de emprego simples, o foco foi na configuração e execução correta das ferramentas de modelagem e testes.

Com isso, o código consiste em um aplicativo que calcula o valor de um frete recebendo um arquivo em csv e atualiza os valores das células com base nos valores de taxa (informados pelo usuário), peso e valor (informados no arquivo) de cada empresa de entrada. No arquivo fornecido ao App existirá uma tabela com as informações a serem usadas e editadas baseadas nas taxas inseridas.

A modelagem foi feita usando Cucumber [1], uma ferramenta para desenvolvimento guiado por comportamento, que, segundo Frederico Toledo et al (2019)[2], é uma ótima ferramenta no aspecto disciplinar já que ela força o usuário a documentar os testes automatizados antes de implementá-los[3].

Modelos e diagramas:

Para facilitar o entendimento do aplicativo foram feitos dois esquemas no diagrama de classes da Figura 1, para demonstrar a interação entre as entidades do sistema.

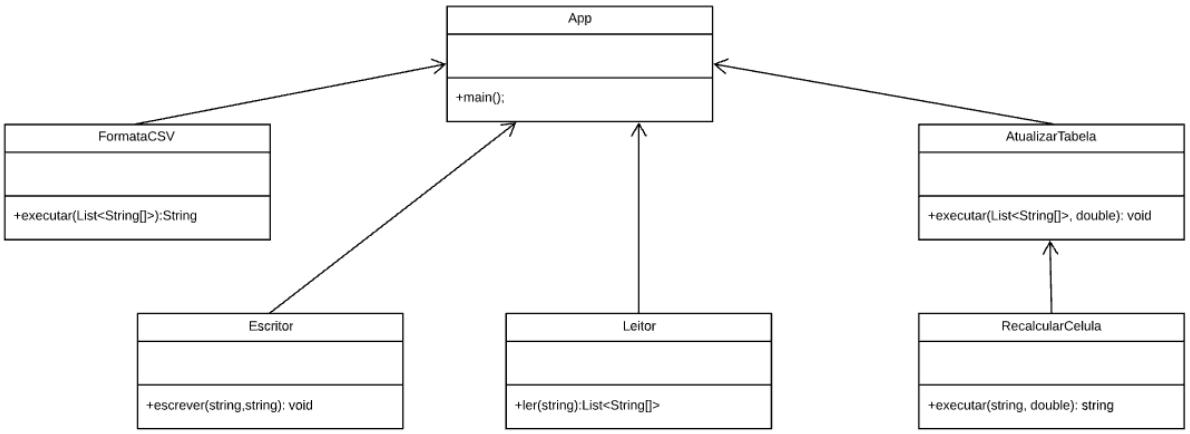


Figura 1. Modelagem UML do Aplicativo

Na parte de casos de uso, é possível visualizar a simplicidade do App, já que a única coisa a fazer é fornecer um csv, e desta forma, o programa atualizará os valores das células baseado nas taxas fornecidas pelo usuário, como pode ser visto na Figura 2, e retornará um csv atualizado.

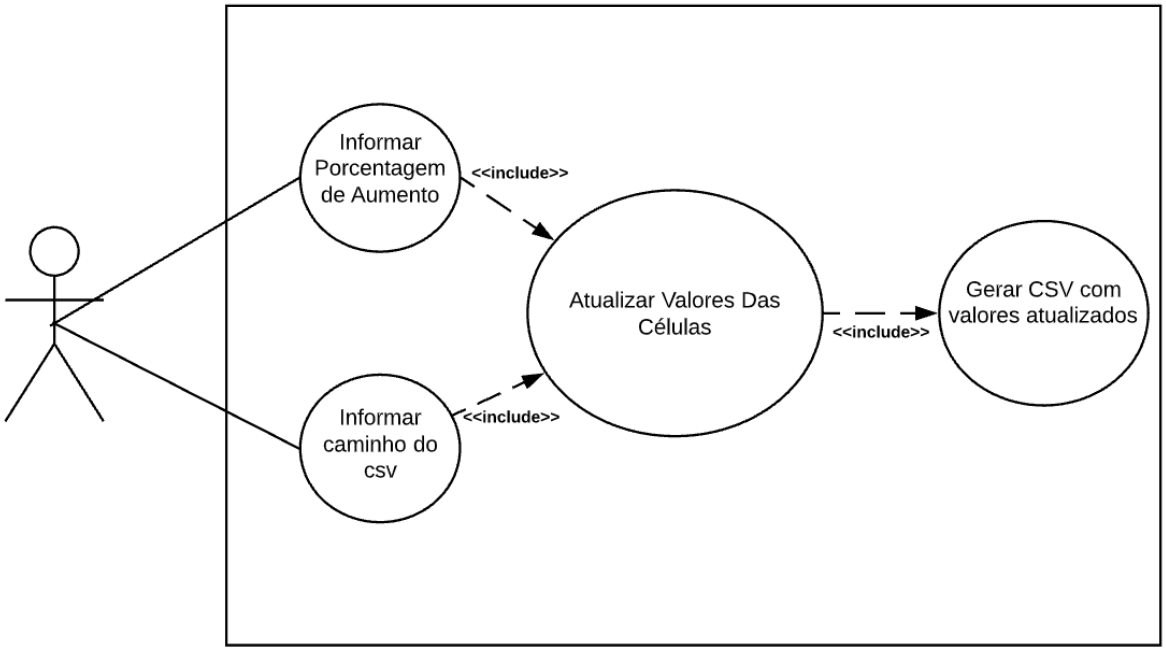


Figura 2. Diagrama de caso de uso

Na parte de testes, após a compreensão do uso correto da ferramenta Gherkin [4], foram desenvolvidos casos que exercitam os componentes do App, como visto na Figura 3, e validam o comportamento devido e esperado do código. O resultado do teste pode ser visto na Figura 4.

```
Feature: RecalcularCelula
    Como usuario
    Eu quero atualizar os valores de todas as celulas do meu arquivo csv
    informando uma porcentagem de aumento
    Portanto, nao vou precisar atualizar celula por celula manualmente!

    Scenario Outline: Atualizar valores de uma celula
        Given Uma celula com os valores de frete peso e frete valor
        When Eu adiciono um acrescimo de <taxa> na celula "<celula>"
        Then Os valores desta devem ser atualizados para "<novosvalores>"

        Examples:
            | taxa | celula               | novosvalores         |
            | 1.1  | FP = 30,00 FV = 0,50 | FP = 33,00 FV = 0,55 |
            | 1.3  | FP = 32,89 FV = 0,34 | FP = 42,76 FV = 0,44 |
            | 1.5  | FP = 33,65 FV = 0,34 | FP = 50,47 FV = 0,51 |
            | 1.2  | FP = 29,60 FV = 0,30 | FP = 35,52 FV = 0,36 |
            | 1.7  | FP = 30,28 FV = 0,30 | FP = 51,48 FV = 0,51 |
            | 1.75 | FP = 31,72 FV = 0,32 | FP = 55,51 FV = 0,56 |
            | 2.0  | FP = 32,45 FV = 0,32 | FP = 64,90 FV = 0,64 |

    Scenario Outline: Converter o acrescimo em uma taxa
        Given Um valor de "<acrescimo>"
        When solicito a conversao
        Then O valor retornado e <taxa>

        Examples:
            | acrescimo | taxa |
            | 10        | 1.1  |
            | 30        | 1.3  |
            | 50        | 1.5  |
            | 20        | 1.2  |
            | 70        | 1.7  |
            | 75        | 1.75 |
            | 100       | 2.0  |
```

Figura 3. Modelagem dos casos de teste usando Gherkin

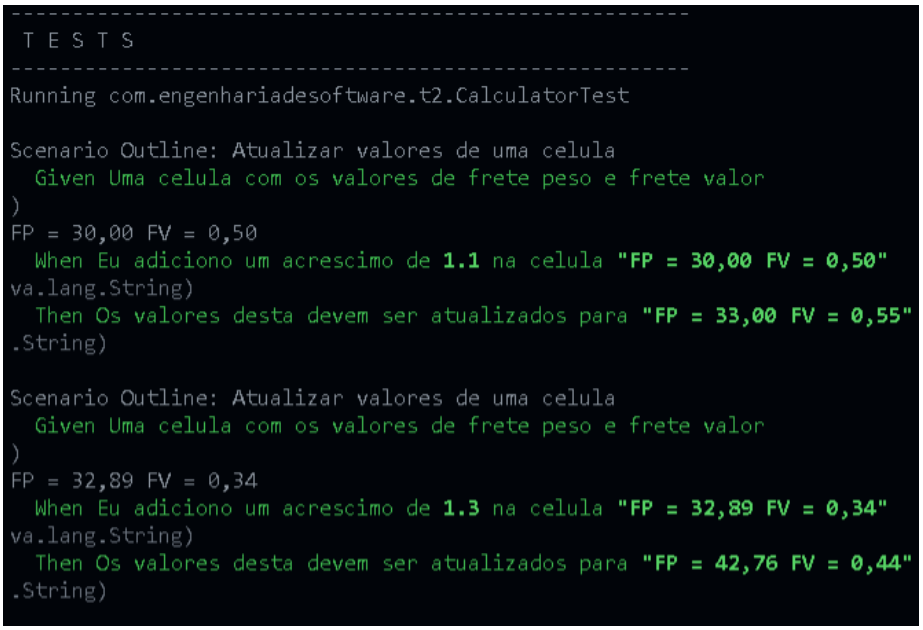
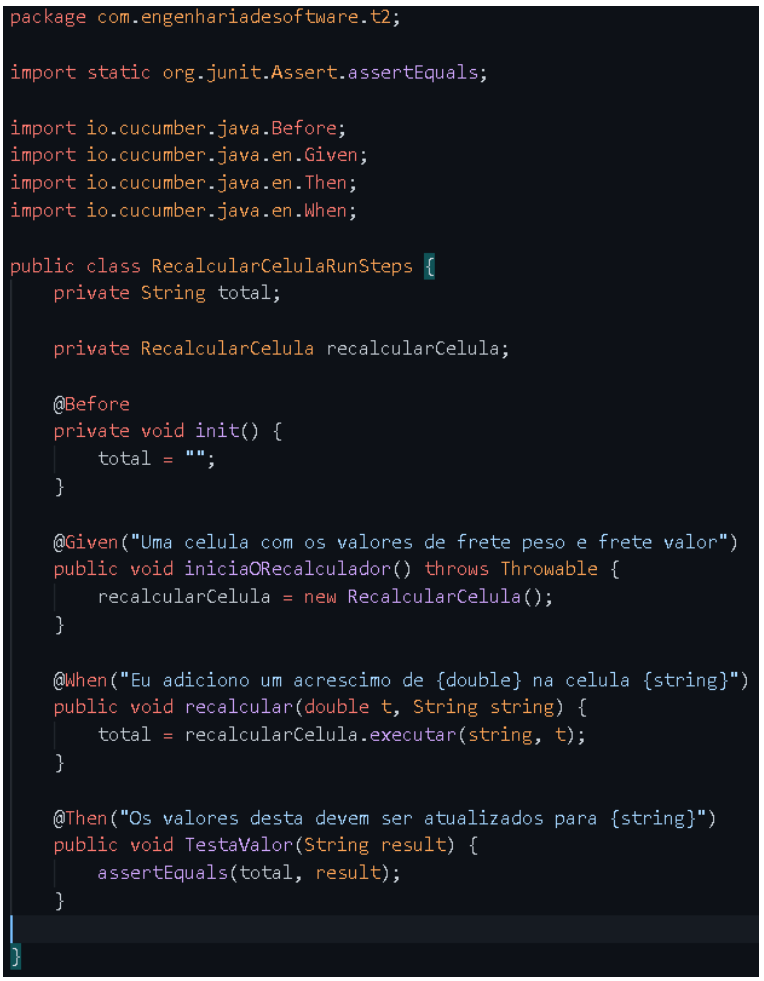


Figura 4. Exemplo de resultado do teste

Para realizarmos os testes seguimos a seguinte estrutura de passos, definidos formalmente na linguagem Gherkin como podemos ver na Figura 5.



```java
package com.engenhariadesoftware.t2;

import static org.junit.Assert.assertEquals;

import io.cucumber.java.Before;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

public class RecalcularCelulaRunSteps {
    private String total;

    private RecalcularCelula recalcularCelula;

    @Before
    private void init() {
        total = "";
    }

    @Given("Uma celula com os valores de frete peso e frete valor")
    public void iniciaORecalculador() throws Throwable {
        recalcularCelula = new RecalcularCelula();
    }

    @When("Eu adiciono um acrescimo de {double} na celula {string}")
    public void recalcular(double t, String string) {
        total = recalcularCelula.executar(string, t);
    }

    @Then("Os valores desta devem ser atualizados para {string}")
    public void TestaValor(String result) {
        assertEquals(total, result);
    }

}
```

Figura 5. Definição dos passos dos testes implementado utilizando Cucumber

Descrição das Classes:

Todas as Classes são compostas por apenas um método, onde cada método faz o que é descrito no detalhamento das classes

App:Classe principal que chama todos os outros métodos;

Métodos: Main();

Leitor: Recebe o arquivo de entrada;

Métodos: Ler();

FormatarCSV: Formata os dados para a atualização dos valores;

Métodos: Executar()

AtualizarTabela: Atualiza os valores com base nas taxas;

Métodos: Executar()

RecalcularCelula: Aplica a porcentagem passada pelo usuário aos valores de uma célula;

Métodos: Executar()

Escritor: Gera o arquivo de saída.
Métodos: Escrever()

Referências:

1. SmartBear, **Cucumber**, 2021, Disponível em: < https://cucumber.io/> Acesso em: 18 de setembro de 2021.
2. TOLEDO, Frederico et al, **A guide to Cucumber best Practices**, Disponível em: <https://dzone.com/articles/a-guide-to-good-cucumber-practices> Acesso em: 18 de setembro de 2021.
3. Baeldung, **Cucumber and Scenario Outline**, Disponível em: <https://www.baeldung.com/cucumber-scenario-outline> Acesso em: 18 de setembro de 2021.
4. SmartBear, **Gherkin Syntax**, Disponível em: <Gherkin Syntax - Cucumber Documentation> Acesso em: 18 de setembro de 2021.