Desenvolvimento de uma Biblioteca Java para Internacionalização de Softwares

Bruno Fonseca Machado[1], **Ricardo Frohlich da Silva**[1]

[1]Curso de Sistemas de Informação - Universidade Franciscana (UFN)
CEP 97.010-030 – Santa Maria – RS – Brasil

{bruno.fonseca,ricardo.frohlich}@ufn.edu.br

Abstract. *The objective of the paper now is present the development of a library that helps the internationalization and localization of softwares under development. Based on the benefits that internationalization can provide in the context of systems in development, which can be aimed at different markets. To achieve this goal, best practice of the Scrum agile development methodology will be used, in addition to the Java programming language, in combination with different APIs and Classes, finally, completing the development of the library and making it available for its use.*

Resumo. *O objetivo agora do trabalho é apresentar o desenvolvimento de uma biblioteca, a qual auxilie a internacionalização e localização de softwares em desenvolvimento. Baseando-se nos benefícios que a internacionalização pode proporcionar no contexto de desenvolvimento de sistemas, os quais possam ser destinados a diferentes mercados. Para atingir este objetivo, foi utilizado boas práticas da metodologia de desenvolvimento ágil Scrum, além da linguagem de programação Java, em combinação a diferentes APIs e Classes, por fim, concluindo o desenvolvimento da biblioteca e a disponibilização para sua utilização.*

1. Introdução

No nosso dia a dia é possível observar que dentre a infinidade de softwares utilizados, que a grande maioria não foi desenvolvida no Brasil, embora estejam devidamente traduzidos e localizados para a língua portuguesa [Burzynski et al. 2010].

Isto é decorrente da excedente expansão tecnológica global ocorrida, aliada a globalização da economia, que acabou transformando mercados externos muito mais acessíveis e importantes, assim possibilitando que um software desenvolvido em um certo país possa ser comercializado no mundo inteiro [Rosa 2020]. Outro pertinente fator é que o Brasil é o décimo primeiro maior mercado de software do mundo, movimentando aproximadamente 20.5 bilhões de dólares em 2019 [AbesSoftware 2017].

A internacionalização de software surgiu com o objetivo de potencializar a venda e distribuição de software para o mundo inteiro, tendo em vista que cada mercado possui suas peculiaridades locais e seus devidos aspectos culturais distintos, além da barreira do idioma de cada país [Rosa 2020].

Com o conhecimento desta importância da internacionalização para o desenvolvimento de softwares, este trabalho visou projetar e desenvolver uma biblioteca para auxiliar desenvolvedores de softwares utilizando da linguagem de programação Java, em conjunto a diversas *APIs* e Classes. Desta maneira, disponibilizou uma forma mais acessível

e simplificada para a realização da internacionalização de software, efetuando a mesma a partir da chamada de métodos definidos na biblioteca. Para alcançar isto, os objetivos específicos foram os seguintes:

- Estudar sobre a internacionalização e localização de software;
- Pesquisar sobre APIs para a tradução de texto;
- Compreender a utilização da biblioteca Locale do Java;
- Pesquisar sobre a utilização de APIs para a conversão de moedas;
- Utilizar boas práticas da metodologia Scrum no desenvolvimento do projeto;
- Testar a biblioteca utilizando um software desenvolvido previamente pelo autor;
- Implementar a biblioteca;

2. Referencial teórico

Nesta seção serão apresentados os conceitos necessários para o desenvolvimento do presente trabalho, além das tecnologias que serão utilizadas para a realização do que será proposto. Serão apresentados os conceitos sobre à internacionalização e localização de software, a linguagem de programação *Java* e suas bibliotecas, para assim, ter um melhor entendimento das técnicas e tecnologias que serão utilizadas para a criação que foi proposto.

2.1. Internacionalização de Software

A Internacionalização (l18n) é o processo de engenharia de software que tem como objetivo o de tornar o mais flexível e neutro em termos de relações culturais, financeiras e legais de um país, ou seja, é uma maneira de projetar um certo produto para que ele seja adaptado a diferentes culturas [Gillam 1999].

O processo de Internacionalização deverá começar logo no início da etapa de desenvolvimento de uma aplicação, onde todos os elementos de código dependentes de uma linguagem possam ser separados, garantindo a existência de uma arquitetura base que possa permitir que o código seja alterado, dependendo de diferentes usuários de diferentes locais, assim sem necessitar que o código possa ser reescrito do zero [Gillam 1999].

Para a internacionalização de um software existem diversos aspectos principais que deverão ser analisados com devida atenção, tais como [de Aragão 2004]:

- Esquemas de cores e escolhas de gráficos que possam ofender potenciais clientes.
- Caixas de diálogo grandes o suficiente para acomodar uma possível expansão de texto.
- Funcionalidade que suportem vários tipos de formatos de data, moeda, hora.

A Tabela 1 a seguir, mostra a existência de elementos que se diferenciam de um pais para outro.

2.2. Localização de Software

A localização (l10n) é o nome dado ao processo de adaptar ou desenvolver um software a um local específico, também é chamado de *locale*. A internacionalização de um software irá ocorrer no país que o produto for originalmente desenvolvido, já a localização

Tabela 1. Os diferentes aspectos entre países.

	Países				
	Brasil	**Estados Unidos**	**Espanha**	**Japão**	**Alemanha**
Idioma	Português	Inglês	Espanhol	Japonês	Alemão
Moeda	Real	Dolár Americano	Euro	Iene	Euro
Símbolo Moeda	R$	$	€	¥	€
Formato de Data	dd-mm-aaaa	mm-dd-aaaa	dd-mm-aaaa	aaaa-mm-dd	dd-mm-aaaa
Fuso Horário	UTC -03:00	UTC -04:00	UTC +02:00	UTC +09:00	UTC +1:00
Sistema de Medida	Métrico	Imperial	Métrico	Métrico	Métrico

se refere a um processo de infusão em um produto de um contexto cultural específico [Batista 2014].

Segundo Collins (2002) a localização de cada país será baseada em uma pesquisa efetuada no mesmo, pois muitos aspectos de conhecimento do país são quase imutáveis, como a linguagem. Já o significado das cores, por exemplo, é mais delicado e sujeito a mudanças culturais. O autor cita a importância do uso de pessoas que tenham o conhecimento atual no país alvo, em vez de pessoas que já viveram ou trabalharam lá, pois o conhecimento pode não ser mais o mesmo.

O processo de localização possui duas dimensões fundamentais: localização de texto (*text localization*) e a localização cultural (*cultural localization*), onde a primeira representa a tradução da interface e documentação da aplicação, já a segunda parte concebe a respeito de todas as convenções culturais de uma determinada região [Batista 2014].

2.3. Linguagem de Programação Java

O Java é uma linguagem computacional de programação orientada a objetos, utilizada para o desenvolvimento de aplicações para diversos propósitos diferentes desde ferramentas de software, aplicações web até mesmo a aplicativos *Android*. Foi desenvolvida para ser flexível, onde desenvolvedores possam escrever um código e utilizarem em qualquer máquina diferente. As aplicações Java são compiladas em *bytecode* que podem ser executados em qualquer Máquina virtual Java (*JVM*) independentemente da arquitetura do computador [IBM 2020].

A sua linguagem é derivada da linguagem C, então suas regras de sintaxe assemelham-se bastantes as regras existentes no C. Possui uma estrutura a qual se inicia com pacotes, já dentro destes pacotes é onde estão localizadas as classes e nestas classes é onde estão todos os métodos, variáveis e constantes [IBM 2020].

2.3.1. Bibliotecas Java

Uma biblioteca é uma coleção de recursos utilizados por programas no desenvolvimento de um software. Possuindo como objetivo, o de compartilhar soluções através de funções ou métodos, que já foram criados por outros desenvolvedores, e que em grande parte do tempo já estejam prontos para uso.

Por exemplo, o Java não possui operações matemáticas mais complexas para isto, é utilizada a biblioteca *math* onde são encontradas diversas funções específicas para certas operações matemáticas [Oracle 2018].

Dentro da linguagem Java existe a *Java Class Library* (JCL), que é um conjunto de bibliotecas de classes pré-escritas, que podem ser chamadas a qualquer momento. Por conta da plataforma Java não depender de um sistema operacional especifico, ela oferece um conjunto de classes que possuem as funções mais comuns aos sistemas operacionais modernos [Oracle 2018].

Os recursos da JCL são acessados por meio de classes que são fornecidas em pacotes, tais como: java.lang, java.math, java.util, etc.

É pertinente citar que no pacote java.util existe o objeto *Locale* o qual é muito importante para a internacionalização e localização, pois nela são armazenados os códigos de línguas que serão consultados por outros métodos para determinar como eles serão tratados [Oracle 2018].

2.4. Metodologia Scrum

Dentre as diversas metodologias ágeis que existem para o desenvolvimento e gestão de projetos, o Scrum é uma das mais populares e empregadas nesse meio. O Scrum é um *framework* para a gestão de projetos baseado no empirismo, ou seja, sustentando que o conhecimento se adquire de decisões já tomadas e de experiências anteriores. Assim ajudando pessoas, empresas e times a gerar um valor através de soluções adaptáveis para problemas complexos [Schwaber and Sutherland 2020].

O Scrum busca o desenvolvimento por ciclos, chamados de *Sprints*. Onde cada um deles acontece de uma forma contínua, sendo um após o outro e possuindo tamanhos e durações variadas, as quais serão definidas nas reuniões de planejamento [Cardinal 2013].

Alguns dos principais elementos que podem ser citados que são utilizados no Scrum, são os seguintes:

• *Product Backlog*: O *Product Backlog* é responsável por conter uma progressiva e ordenada lista de tudo que será necessário realizar ou melhorar no produto. Por ser uma lista progressiva ela irá evoluir de acordo com o avanço do produto, assim se adaptando a novos formatos.

• *Sprint Backlog*: O termo é dado ao *Product Backlog* junto ao plano de entrega dos itens que forem selecionados para a *Sprint*.

• *Daily Scrum*: São reuniões diárias que envolvem a equipe de desenvolvimento, com objetivo de abordar o andamento da *Sprint*, além da criação do plano do que deverá ser feito para as próximas 24h.

A Figura 1 mostra todo o ciclo de funcionamento de uma *Sprint*, as quais são fundamentais no Scrum [Cardinal 2013].

2.5. Google Cloud Translation API

Com a globalização da internet envolvendo diversos países em diferentes línguas foi criada uma demanda para a tradução de websites e textos online. Assim, em 2006 foi introduzida pela Google a ferramenta Google Tradutor com a função de traduzir vários idiomas diferentes [Cloud 2020].

A Google Cloud Translation API traduz instantaneamente mais de cem idiomas diferentes, além de detectar automaticamente com alta precisão o idioma escrito. Possui

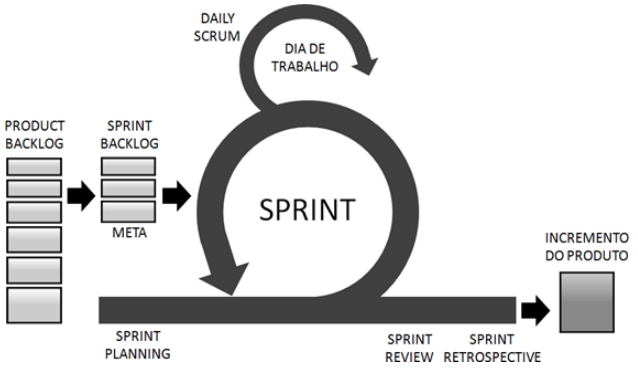

Figura 1. O ciclo do Scrum

a importante função de personalizar como a tradução deverá ser aplicada para termos e frases específicas que variam de seu contexto, assim facilitando na localização de certas regiões especificas. O custo pelo seu uso é contado a partir da quantidade de caracteres utilizada, quanto mais caracteres forem utilizados, maior será o seu valor [Cloud 2020].

2.6. API de Cotações de Moedas - Fixer API

A taxa de câmbio e suas cotações são fundamentais para a economia global e seus agentes econômicos como o governo, o consumidor e as indústrias. Assim a sua cotação atualizada se torna essencial para acompanhar as flutuações e mudanças de valor variando de uma moeda para outra [Santos Oliveira et al. 2006].

A *Fixer API* é uma *API* que pode ser utilizada para a realização da conversão de câmbio, possuindo uma precisão de 6 casas decimais e com uma frequência de até 60 segundos. Sendo capaz de entregar em tempo real o valor atual de cerca de 170 moedas mundiais. O seu custo é parcialmente gratuito, contando com diferentes planos que variam desde um plano com uso limitado gratuito até várias categorias de planos pagos [FixerAPI 2018].

2.7. Trabalhos Relacionados

Nesta seção, serão apresentados os trabalhos relacionados que foram utilizados como base para o desenvolvimento desse trabalho. Foram escolhidos baseados na semelhança com o presente trabalho.

2.7.1. Modelo de Internacionalização em Bases de Dados

O trabalho desenvolvido por Santos (2015), pretende a utilização de tecnologias como *Java*, *Zend Framework* e *Hypersonic SQL* para o desenvolvimento de um modelo de internacionalização para bases de dados e salienta a importância da divisão de todo processo em etapas para uma melhor eficácia de desenvolvimento desta árdua e complexa tarefa, para obter assim um produto final de qualidade.

No trabalho é apresentado o desenvolvimento de um módulo para que ele seja utilizado em uma base de dados, o qual separe todos os dados que forem dependentes de um idioma especifico. Assim, facilitando a edição desses dados independentes o suficiente para que eles sejam editados por pessoas sem qualquer conhecimento em bases de dados.

Além disto, os dados alterados ainda poderão ser recarregados de volta a base de dados com ela ainda em funcionamento.

Por fim, o trabalho conclui que foi possível incluir em uma base de dados um modelo para a sua internacionalização simples e flexiva tanto para o programador, quanto para o tradutor. E validando a possibilidade de integração de um mesmo módulo para também uma base de dados fechada.

2.7.2. Desenvolvimento e Aplicação de um processo de localização de software: Internacionalização de Produtos

O trabalho de Rodrigues e Filho (2019) apresenta o desenvolvimento de um modelo de processo para a localização de software, proposto para ajudar fábricas de *software* para a localização e tradução de seus produtos.

Para a sua gestão e planejamento, utilizaram o *framework SCRUM* para a melhor gestão do projeto possível, e do *software Bizagi* para a modelagem de todos os processos envolvidos. Como tecnologias utilizadas, utilizaram do *software Localise* para o gerenciamento das traduções e da linguagem de programação *JSON* para facilitar a troca de dados entre a ferramenta *Localise* e o seu *software* requerido.

Rodrigues e Filho (2019), baseados nos resultados obtidos em seu trabalho, aplicaram a localização e tradução com sucesso ao *software* em desenvolvimento *Virtoo*, o qual foi destinado a Angola.

2.7.3. Abordagem metodológica para auxiliar no processo de localização de um ERP Open Source

O trabalho elaborado por Batista (2014), apresenta o desenvolvimento de um *framework* para a localização de um ERP (*Enterprise Resource Planning*) *Open Source*, ou seja, um Sistema integrado de gestão empresarial.

Para que a localização seja realizada, o *framework* desenvolvido exige que exista no ERP de origem um conjunto de componentes específicos. O processo onde estes componentes serão localizados poderá ser realizado em uma escala de cinco níveis, variando do nível de localização pretendido. A partir deste processo, cada um desses níveis irá gerar um relatório e estes relatórios serão o primeiro resultado do *framework*. Assim, baseado nesses relatórios será onde irá ser executada a localização do ERP.

Por fim, Batista (2014) acaba por com sucesso validar o *framework* proposto através do ERP gratuito *Dolibarr*.

2.7.4. Internacionalizando e localizando aplicações Java na Web

O artigo desenvolvido por Schwarzer (2012), apresenta e demonstra como utilizar os recursos da linguagem *Java* para internacionalizar e localizar aplicações *web*, de uma maneira que ele possa se adaptar a diversos idiomas e regiões.

No artigo são apresentadas as classes de localização encontradas na plataforma *Java*, as responsáveis pela região geográfica especifica e pelo armazenamento dos dados que variam de acordo com a localidade. Junto a isto, é apresentado pelo autor, os métodos disponíveis para localização de datas, horas, caracteres especiais e valores numéricos que variam de acordo com cada localidade.

2.7.5. Considerações sobre os trabalhos relacionados

Os trabalhos relacionados citados demonstram diversos tópicos relacionados para o desenvolvimento deste trabalho, pois cada um deles se combinam e se complementam de certa maneira. Enfatizando desde as técnicas e metodologias para localização e internacionalização, as tecnologias utilizadas e os softwares desenvolvidos voltados ao mesmo propósito.

Entre todos os trabalhos citados, o trabalho de e Rodrigues e Filho (2019) é o que mais se assemelha com o trabalho proposto afim de criar um modelo de processo para localização de um software, porém, utilizando da linguagem de programação diferente a *JSON*. Seu trabalho contribuirá para o entendimento dos diversos processos e tecnologias envolvidas para a localização e internacionalização de softwares destinados a apoiar fábricas de software.

Já o trabalho de Santos (2015) explora a utilização da mesma tecnologia que será utilizada neste trabalho, porém com um objetivo diferente o da criação de um modelo para internacionalização de uma base de dados. Os conceitos e métodos utilizados por Santos (2015) serão de grande importância para a realização do presente trabalho, como o da utilização de um módulo que separe todos os dados dependentes de um idioma específico para que assim sejam internacionalizados separadamente.

O trabalho de Batista (2014) explora a importante divisão do processo de localização, podendo ser em uma escala de cinco níveis diferentes. Onde cada nível irá gerar um relatório variando do grau de localização necessário. Esta divisão apresentada e exemplificada pelo autor irá apoiar muito para a tarefa de localização e internacionalização do trabalho.

As técnicas, métodos e classes apresentadas por Schwarzer (2012) para a utilização na linguagem *Java* servirá como base principal para a sua utilização no trabalho, trazendo conceitos, exemplos e a metodologia necessária para realizá-la. O que contribuirá muito para a compreensão e a realização da internacionalização utilizando da linguagem *Java*.

3. Projeto

Este trabalho tem como objetivo desenvolver uma biblioteca, que busque por atender a necessidade de localizar e internacionalizar softwares que estejam em diferentes idiomas. A sua principal função será a de permitir que softwares em desenvolvimento possam ser traduzidos e localizados da maneira mais eficaz.

Neste projeto, foi utilizado alguns artefatos da metodologia Scrum como *Product Backlog*, *Sprints* e não existirá o papel de *Scrum Master*. As atividades, serão selecionadas no *Product Backlog* e organizadas em *Sprints*. Será utilizada a ferramenta Trello para

controlar as *Sprints* desenvolvidas. Nesta seção será apresentada a modelagem desenvolvida, em conjunto as etapas elaboradas com o uso da metodologia Scrum.

3.1. Proposta

A biblioteca deverá possibilitar uma maneira de tradução eficiente e ser compreensível para a utilização do usuário, pois textos e frases são dos elementos mais comuns encontrados em softwares que deverão ser internacionalizados. Na Figura 2 é demonstrado o fluxo que a biblioteca deverá seguir para a tradução de informações.

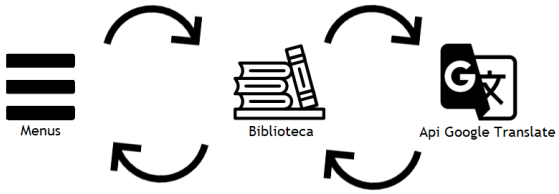

Figura 2. Representação da tradução

Primeiramente, os elementos que necessitem a sua devida tradução e o idioma selecionado deverão ser enviadas a partir de um método para a biblioteca, em seguida, a biblioteca enviará estas informações para *API* de tradução. A qual terá a função de traduzir estes elementos e enviar de volta a biblioteca, que retornará as informações já traduzidas de volta a aplicação.

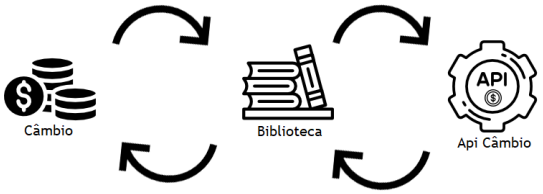

Figura 3. Representação da conversão de câmbio

A Figura 3 representa o fluxo de informações da biblioteca para a conversão de câmbio, que não irá se diferenciar do fluxo da tradução, pois ambos métodos estarão conectados a *APIs*. Para a conversão de moedas realizada pela biblioteca, será utilizada da *Fixer API*, onde a biblioteca deverá enviar a a moeda e o seu valor, que serão enviados a *API* onde retornarão convertidos e atualizados de volta a biblioteca, que repassará a aplicação.

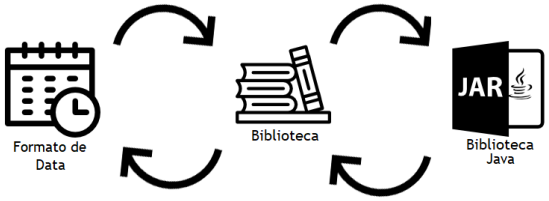

Figura 4. Representação da conversão de Formatos de Data

Para conversão de data e hora que é representada pela Figura 4, por exemplo, deverão ser utilizadas as classes previamente existentes na linguagem Java como a classe *Locale*.

Utilizando de diagramas de atividades foi realizável a construção da biblioteca por funcionalidades. Na Figura 5, é apresentado o diagrama de atividades, que será responsável por demonstrar todo o fluxo das atividades contidas na biblioteca. Sendo as seguintes as atividades:

- Traduzir Idioma.
- Converter Moedas.
- Converter Medidas.
- Converter Formato de Data e Hora.

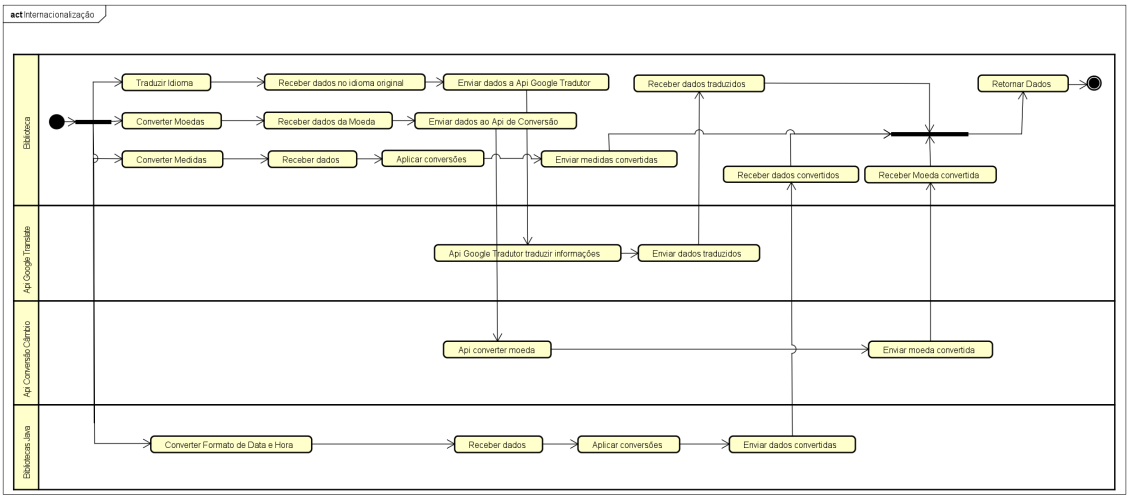

Figura 5. Diagrama de Atividades

Com a utilização de um diagrama de domínio foi possível observar uma visão conceitual do projeto da biblioteca, na Figura 6 é ilustrado a digrama de domínio criado.

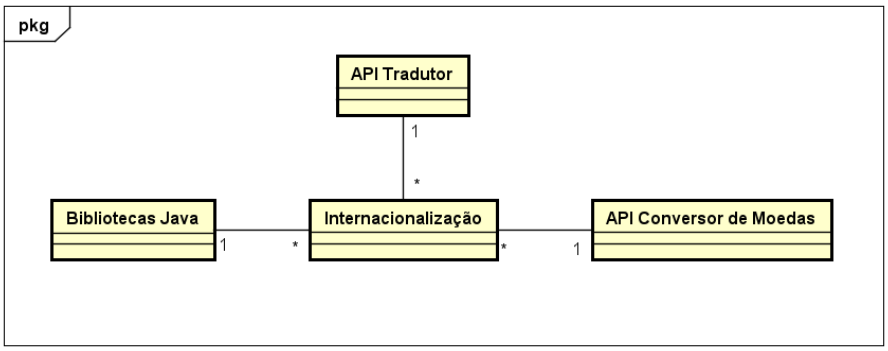

Figura 6. Diagrama de Dominio

3.2. Product Backlog

Esta subseção apresenta as funcionalidades que deverão atender ao que foi requisitado para o desenvolvimento da biblioteca que será aplicada para o desenvolvimento destas funcionalidades.

Os requisitos serão divididos em requisitos funcionais e requisitos não funcionais. Ou seja, os requisitos funcionais irão descrever o que a biblioteca irá realizar e os requisitos não funcionais como a biblioteca irá realizar. Na Tabela 2 são listados os requisitos funcionais (RF), os quais são essenciais para o desenvolvimento da biblioteca.

Tabela 2. Requisitos Funcionais (RF).

ID	Descrição	Relevância	Complexidade
RF1	Método para a tradução de frases e textos utilizando de uma API.	Essencial	Alta
RF2	Método para Conversão de Moedas utilizando de uma API.	Essencial	Média
RF3	Método para a Conversão de Medidas.	Essencial	Média
RF3	Método para Conversão de Formatos de Data e Hora.	Essencial	Baixa

Seguindo o que é descrito na metodologia ágil *Scrum* foi criado um *Product Backlog* com as atividades necessárias para o desenvolvimento da biblioteca, que foram determinados a partir dos requisitos funcionais e não funcionais citados anteriormente. Para o desenvolvimento da biblioteca, as atividades foram divididas em 6 *Sprints* e planejado o tempo de desenvolvimento, conforme apresentado na Tabela 4.

Tabela 3. Os *Products Backlog* determinados

Sequência	Product Backlog	Tempo de desenvolvimento Estimado (Hrs)
1	Criar método para tradução utilizando da API Translate.	180
2	Criar método para conversão de moedas utilizando da API.	180
3	Desenvolver método para Conversão de medidas utilizando das bibliotecas Java.	100
4	Adaptar método para conversão de data utilizando das bibliotecas Java.	100
5	Adaptar método para conversão de hora utilizando das bibliotecas Java.	100
6	Realizar a validação de todos métodos desenvolvidos em um software já desenvolvido.	60

3.3. Desenvolvimento da Biblioteca

Para o desenvolvimento da função de tradução de idiomas da biblioteca, foi utilizada a API de tradução Google Cloud Translate da Google. Foi necessário realizar um cadastro no site da Google Cloud Plataform, após a realização do cadastro, foi gerada uma "*key*" exclusiva, essencial para o funcionamento da API. Baseando-se no já determinado *Product Backlog* número 1, que pode ser visualizado na Tabela 3, foi criada uma *Sprint* para a realização da criação da função de tradução. Grande parte do processo de desenvolvimento, foi baseado nos exemplos contidos na própria documentação da API de tradução.

Na Figura 7 é possível observar o código do método de tradução da biblioteca, em primeiro lugar, como pode ser observado na linha 18, o serviço de tradução da API é inicializado. Em seguida, os dados de idioma e texto são inseridos em uma lista especifica

da API, além destes elementos, também é inserido a lista uma “*Key*”. Como pode ser visualizada na linha 25, que tem seu funcionamento similar a uma chave, esta chave é gerada exclusivamente para o usuário cadastrado na API de tradução.

Os dados contidos nesta lista serão executados pela API, que em seguida, irá armazenar estas informações em uma variável do tipo *TranslationsListResponse*. Que como pode ser observada na linha 27, foi chamada “resposta”. E que tem por função de transformar os dados enviados via *HTTP* para o tipo *JSON*, após isto, ocorrerá um laço de repetição para armazenar o elemento traduzido para a variável ”textoFinal”. E por fim, as informações já traduzidas serão retornadas à aplicação.

```
1  package biblioteca;
2
3  import java.io.IOException;
12
13 public class traduzirElementos  {
14     public String TraduzirDados(String idioma, String texto) throws GeneralSecurityException, IOException {
15
16         String textoFinal = null;
17
18         Translate traducao = new Translate.Builder(GoogleNetHttpTransport.newTrustedTransport(),
19                 GsonFactory.getDefaultInstance(), null).setApplicationName("Exemplo-Bruno").build();
20
21         /* Nesta lista serão inseridas as informações que deverão serem traduzidas e o idioma final. */
22         Translate.Translations.List list = traducao.new Translations().list(Arrays.asList(texto), idioma);
23
24         /* Será inserida na lista a Key gerada pela API da Google */
25         list.setKey("AIzaSyAFXIECpRcKg0sN_moGr6TCt9G_xhUkvVE");
26
27         TranslationsListResponse resposta = list.execute();
28
29         for (TranslationsResource translationsResource : resposta.getTranslations()) {
30             textoFinal = translationsResource.getTranslatedText();
31
32         }
33         return textoFinal;
34     }
35 }
```

Figura 7. Código da função de tradução

Para a função de conversão de moedas da biblioteca, foi utilizada a API gratuita *Currency Converter API*. Para o uso da API, como na anterior, foi necessário realizar um cadastro no seu site, para adquirir uma ”*Key*”. Baseando-se no já determinado *Product Backlog* número 2, foi criada uma *Sprint* para a realização da criação da função de conversão das moedas.

Primeiramente, será enviado ao site específico da API, todos os argumentos, sendo a moeda inicial, e a moeda final, isto tudo em conjunto com a *Key*, como pode ser observada na linha 20. Após isto, foi realizada a conexão com o site da API, que retornará a moeda inicial convertida para a final, e devidamente convertida no formato *JSON*. Em seguida, o valor da moeda em seu formato *JSON* será manipulada e armazenada para ser utilizada, após isto, o valor convertido será multiplicado pelo valor inicial, para assim obtermos o valor totalmente convertido.

Com base no *Product Backlog* número 3, foi criada uma *Sprint* para a realização da criação da função de conversão de medidas, baseando na conversão do padrão métrico, para o padrão de medida imperial. Isto em razão, de serem os sistemas de medidas mais comuns. Foi desenvolvida na biblioteca, diversas funções de conversão de um padrão para outro, e vice-versa. Como, por exemplo, de galão para litro, de libra para quilograma, de

```java
public class conversorMoedas {

    private double valorFinal = 0;

    public double converterMoeda(String moedaInicial, String moedaFinal, Double valorInicial) {

        try {

            String simbolo = moedaInicial + "_" + moedaFinal;

            String url = "https://free.currconv.com/api/v7/convert?q=" + moedaInicial + "_" + moedaFinal
                    + "&compact=ultra&apiKey=d07b273266d96d2d2a43"; // URL para acessar a API

            URL obj = new URL(url);
            HttpURLConnection con = (HttpURLConnection) obj.openConnection(); // Conexão Http

            BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));

            String inputLine;
            StringBuffer response = new StringBuffer();
            while ((inputLine = in.readLine()) != null) { // Realiza a leitura do JSON

                response.append(inputLine);
            }
            in.close();

            JSONObject respostaJSON = new JSONObject(response.toString()); // Converte o JSON para JSONObject para ser
                                                                           // manipulado

            valorFinal = respostaJSON.getDouble(simbolo); // Seleciona a moeda
            valorFinal = valorFinal * valorInicial;

        } catch (Exception e) {
            System.out.println("Ocorreu um erro para a conversão da "+moedaInicial+" para a "+moedaFinal+".");
        }

        return valorFinal;
    }
}
```

Figura 8. Código da função de conversão de moedas

milha para quilômetro, e vice-versa.

Como pode ser observado na Figura 9, a função receberá como argumento o valor destinado a ser convertido, após isto, a fórmula para a conversão será aplicada no valor recebido, retornado em seguida de volta para a aplicação.

```java
package biblioteca;

public class converterGalaoParaLitro {
    public double converter(double valor) {
        double total = 0;
        total = valor * 3.78541;
        return total;
    }
}
```

Figura 9. Código da função de conversão de medida

A partir dos *Product Backlog* número 4 e 5, foi criado apenas uma *Sprint* com a finalidade da criação de uma função para a conversão de data e hora, como pode ser observada na Figura 10. No seu desenvolvimento, como pode ser observado na Figura 8, foi utilizado o pacote *java.time*, que teve por objetivo facilitar a conversão dos formatos de data e de *timezone*, a função deverá receber apenas um argumento, este argumento será o local final da conversão da data, como, por exemplo: ”Europe/Paris”.

Como pode ser observado na linha 13, a função irá armazenar a hora atual da máquina, que em seguida, será convertida conforme a *timezone* que foi recebida. Após isto, será determinado o formato de data que será utilizado, que pode ser observado na linha 17 e 18. Por fim, o padrão determinado será aplicado a data e hora convertida.

```java
import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;

public class converterDataEHora {

    public String converter(String zonaDoArquivo) {

        String resultadoFinal = "";

        ZonedDateTime horaDaMaquina = ZonedDateTime.now();
        ZoneId zonaFinal = ZoneId.of(zonaDoArquivo);
        ZonedDateTime horaEDataConvertida = horaDaMaquina.withZoneSameInstant(zonaFinal);

        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
        formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");

        resultadoFinal = horaEDataConvertida.format(formatter);

        return resultadoFinal;
    }
}

```

Figura 10. Código da função de conversão de data e hora

3.4. Resultados

Para a realização da validação da biblioteca, foi utilizado o software de um mini mercado, que já foi previamente desenvolvido pelo autor deste trabalho. Primeiramente, no momento o qual o usuário, for realizar a execução da aplicação do minimercado pela primeira vez, a aplicação irá identificar automaticamente a língua selecionada no sistema, e a partir desta linguagem selecionada será a qual o sistema irá funcionar, até que por algum motivo seja alterada pelo usuário nas configurações do sistema.

Logo após a língua ser detectada pela aplicação, será criado no computador do usuário um arquivo TXT, neste arquivo a aplicação irá adicionar a língua detectada pelo sistema.

Para a internacionalização da aplicação do mini mercado, no menu ferramentas, foi adicionado uma ferramenta chamada “Internacionalização” , esta funcionalidade tem como objetivo a de abrir um menu onde as opções de internacionalização do software possam ser encontrados, sendo elas:

• Selecione a língua: Será onde a língua do sistema poderá ser selecionada, a partir dela a *timezone* do sistema também será selecionada.

• Selecione a moeda: Será onde a moeda do sistema poderá ser selecionada;

• Sistema de Medidas: Será onde o padrão das medidas poderá ser selecionado;

Qualquer mudança de linguagem realizada pela tela de internacionalização da aplicação, irá modificar o arquivo TXT e a base de dados da aplicação, atualizando-o para as informações escolhidas.

Em todo momento, o qual for chamada a função de tradução da biblioteca, ela irá verificar a língua que estiver no arquivo, e a partir dela irá realizar a tradução. Após as opções desejadas serem selecionadas, elas serão aplicadas a aplicação do minimercado. Na Figura 11 pode ser observado o resultado deste menu.

Para a utilização da função de tradução desenvolvida na biblioteca, foram utiliza-

Figura 11. Tela Inicial da Aplicação

das todas as telas do sistema, à exceção da tela de relatórios, que não possuía nenhuma função na aplicação. Para cada uma dessas telas, dentro do seu desenvolvimento, foram selecionados todos os elementos os quais necessitariam tradução, ou seja, todas as palavras ou frases. Assim, no momento em que a tela for aberta, o sistema baseando-se na linguagem previamente selecionada na configuração do sistema, realizaria a tradução de todos os elementos utilizando da função de tradução da biblioteca.

Na Figura 12, é possível visualizar uma das telas da aplicação que foi traduzida, esta referente ao gerenciamento de estoque/inventário, antes e depois ser traduzida pela biblioteca da língua portuguesa para a língua inglesa.

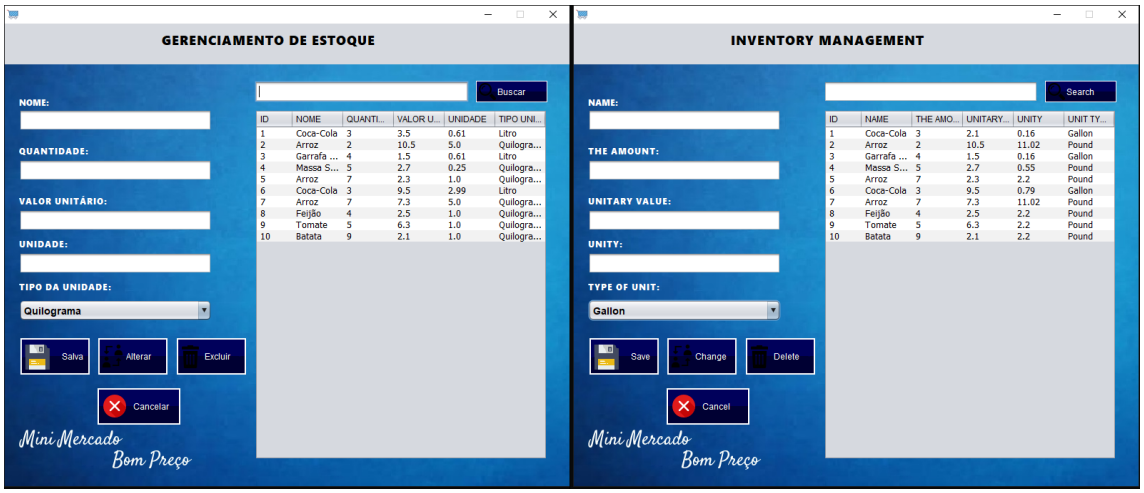

Figura 12. Antes e depois do frame

Para a utilização da função de conversão de moedas da biblioteca, foi aplicada da seguinte maneira. No momento o qual, a moeda escolhida do sistema for alterada no menu de internacionalização, a aplicação irá percorrer toda a base de dados do sistema, com objetivo de buscar todos os valores monetários, e a partir destes valores e da moeda determinada, irá realizar a conversão de todos eles.

Além disto, na base de dados do minimercado, foi criada uma tabela chamada Internacionalização, desta tabela será onde será armazenada a moeda atual dos produtos da base de dados. Na Figura 13, pode ser observado na coluna Valor Unitário, a conversão de valores realizada pela biblioteca, da moeda real para a dólar.

ID	NOME	QUANTI...	VALOR UNITÁRIO	UNID...	TIPO...
1	Coca-Cola	3	2.1	0.16	Gallon
2	Arroz	2	10.5	11.02	Pound
3	Garrafa ...	4	1.5	0.16	Gallon
4	Massa S...	5	2.7	0.55	Pound
5	Arroz	7	2.3	2.2	Pound
6	Coca-Cola	3	9.5	0.79	Gallon
7	Arroz	7	7.3	11.02	Pound
8	Feijão	4	2.5	2.2	Pound
9	Tomate	5	6.3	2.2	Pound
10	Batata	9	2.1	2.2	Pound

ID	NOME	QUANTI...	VALOR UNITÁ...	UNID...	TIPO ...
1	Coca-Cola	3	0.38	0.61	Litro
2	Arroz	2	1.89	5.0	Quilo...
3	Garrafa ...	4	0.27	0.61	Litro
4	Massa S...	5	0.49	0.25	Quilo...
5	Arroz	7	0.41	1.0	Quilo...
6	Coca-Cola	3	1.71	2.99	Litro
7	Arroz	7	1.31	5.0	Quilo...
8	Feijão	4	0.45	1.0	Quilo...
9	Tomate	5	1.13	1.0	Quilo...
10	Batata	9	0.38	1.0	Quilo...

Figura 13. Antes e depois do frame

Para a validação das funções relacionadas a conversão de medidas da biblioteca, foram desenvolvidas quatro funções de conversão de medidas, que convertesse o sistema métrico e o sistema imperial, sendo elas: Litro para *Gallon* (Galão), Quilograma para *Pound* (Libra), *Gallon* para Litro e *Pound* para Quilograma;

O uso destas funções na biblioteca, funcionou da seguinte maneira: no momento que, o usuário fosse alterar o sistema de medidas da biblioteca por via do menu de internacionalização, a aplicação do minimercado percorreria toda a base de dados do estoque, alterando todas medidas, do sistema métrico para o sistema imperial, ou vice-versa. Na Figura 13, é possível observar também a conversão das medidas, na coluna Tipo.

Para a validação da função de conversão de data e hora, foi escolhido a tela de realizar vendas. No momento que o usuário realizar a configuração da aplicação pelo menu de internacionalização, no arquivo junto ao idioma será salvo a *timezone*, como, por exemplo, se o usuário selecionou a língua portuguesa será salva a *timezone* ”America/SaoPaulo”. Após isto, no momento em que o usuário entrar na tela venda, o sistema irá ler o arquivo e verificar a *timezone* salva e assim, a data será automaticamente convertida.

4. Conclusões

Este trabalho apresentou a proposta de desenvolvimento de uma biblioteca destinada à internacionalização de softwares em desenvolvimento, utilizando da linguagem de programação Java. Em decorrência da essencial importância da internacionalização no nosso dia a dia, além de no desenvolvimento de softwares destinados a diferentes mercados.

A partir das boas práticas da metodologia de desenvolvimento Scrum, foi possível obter a melhor gestão e planejamento, aliada a linguagem de programação Java que conteve uma infinidade de recursos e possibilidades que proporcionaram um eficiente desenvolvimento da biblioteca.

Para trabalhos futuros, sugere-se um aprimoramento na função de conversão de moedas, por conta da alta volatilidade referente as suas cotações, e além disto, aprimorar a biblioteca para alcançar um maior nível de internacionalização em todas suas áreas.

Por fim, o desenvolvimento de uma biblioteca para a internacionalização de softwares, na linguagem *Java*, obteve total sucesso em todas as funcionalidades previstas nos *Product Backlog*, como a tradução de elementos, a conversão de moedas, medidas e a conversão de formato de data e hora, e isso com a devida exatidão ao tempo previsto no mesmo.

Referências

AbesSoftware (2017). Mercado brasileiro de software: panorama e tendências. https://abessoftware.com.br/wp-content/uploads/2020/10/ABES-EstudoMercadoBrasileirodeSoftware2020.pdf. Acessado em: 01-06-2021.

Batista, M. (2014). Abordagem metodológica para auxiliar no processo de localização de um erp open source. https://repositorio.iscte-iul.pt/handle/10071/8696. Acessado em: 02-03-2021.

Burzynski, O. R., Graeml, A. R., and Balbinot, Z. (2010). The internationalization of the software market: Opportunities and challenges for brazilian companies. https://www.scielo.br/j/jistm/a/DHRKPj6jdL4bfWf4JyYwR5s/?lang=en. Acessado em: 18-06-2021.

Cardinal, M. (2013). Executable specifications with scrum. Acessado em: 15-06-2021.

Cloud, G. (2020). Google cloud translation api. https://cloud.google.com/translate/?hl=pt-BR. Acessado em: 21-05-2021.

de Aragão, D. D. (2004). O sistema unl nos processos de internacionalizacao e localizacao de software. https://repositorio.ufsc.br/bitstream/handle/123456789/183872/TCC_Diego.pdf?sequence=-1. Acessado em: 02-03-2021.

FixerAPI (2018). About fixer.io. https://fixer.io/about. Acessado em: 15-06-2021.

Gillam, R. (1999). Developing global applicationsin java. http://unicode.org/iuc/iuc15/ta4/slides.pdf. Acessado em: 02-03-2021.

IBM (2020). Fundamentos da linguagem java. https://developer.ibm.com/br/languages/java/tutorials/j-introtojava1/. Acessado em: 02-03-2021.

Oracle (2018). Fundamentos da linguagem java. https://docs.oracle.com/javase/6/docs/technotes/guides/index.html#base. Acessado em: 02-03-2021.

Rosa, R. H. (2020). Revolução informacional e os avanços tecnológicos da informática e das telecomunicações. https://www.seer.ufal.br/index.php/cir/article/view/3482/3029. Acessado em: 01-06-2021.

Schwaber, K. and Sutherland, J. (2020). O guia do scrum. https://andrelmgomes.com.br/wp-content/uploads/2020/11/Guia-do-Scrum-2020-PT-BR-EN-US-1.pdf. Acessado em: 10-05-2021.

A. APÊNDICE A

Tutorial de Instalação da Biblioteca: Para a utilização da biblioteca primeiramente é necessário adicionar o arquivo JAR da biblioteca em conjunto a uma lista de outros JARS.

Instalação dos JARs pelo NetBeans:

1. Botão direito no projeto desejado - Propriedades

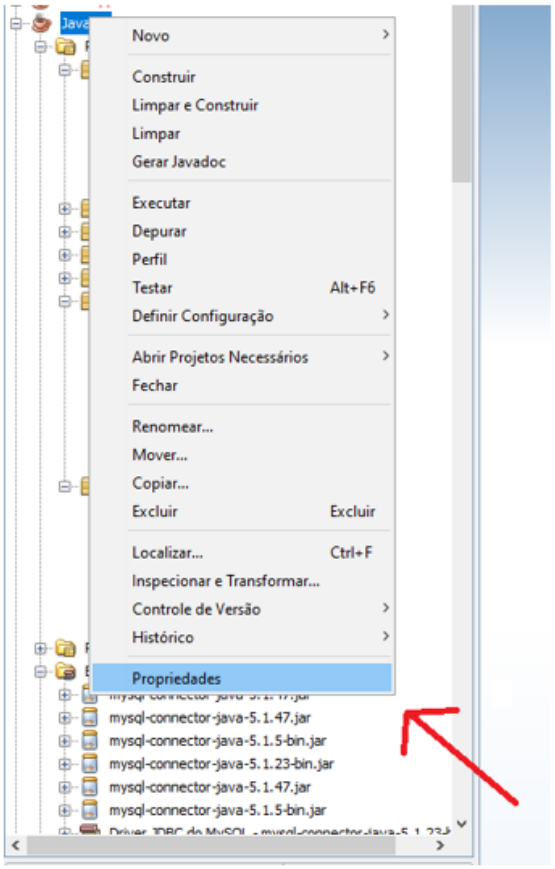

Figura 14. Propriedades do projeto

2. Bibliotecas - Adicionar JAR/PASTA

3. Selecionar todos JARs, e clicar em Abrir

Funcionalidades: Primeiramente, a biblioteca deverá ser importada para a classe onde será usada.Em uma classe, onde mais de uma funcionalidade da biblioteca for utilizada, pode ser utilizado a linha:

import biblioteca.*;

Com esta linha, a classe poderá utilizar de todas funcionalidades da biblioteca, não apenas a classe de traduzir elementos.Para a utilização da função de tradução, ela deverá ser instanciada.

1) Tradução de Elementos

Primeiramente, a função de traduzir deverá ser instanciada:

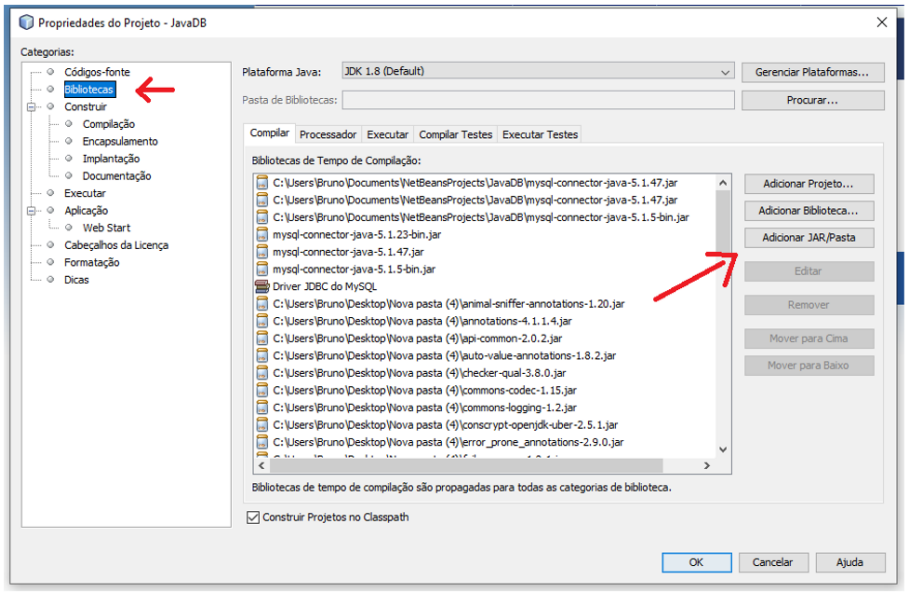

Figura 15. Propriedades do projeto

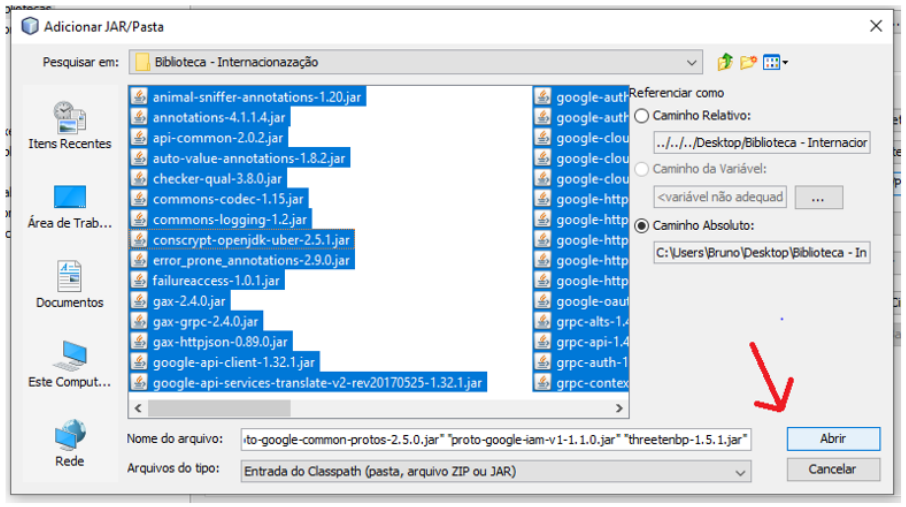

Figura 16. Todas as JARs selecionadas

```
8    * @author Bruno
9    */
10 import biblioteca.traduzirElementos;
11
12 public class TesteConversãoBiblioteca {
13
```

Figura 17. A Biblioteca sendo importada

A função para a tradução de elementos receberá dois argumentos:

- Língua: A língua para que o texto deverá ser traduzido.
- Texto: Os elementos que deverão serem traduzidos, como palavras ou fra-

```
17
18          traduzirElementos traduzir = new traduzirElementos();
19
```

Figura 18. Instanciamento da função

```
19
20          String lingua = "en";
21          String texto = "Olá meus amigos!";
22
23          traduzir.TraduzirDados(lingua, texto);
24
```

Figura 19. Variáveis e chamada da função

ses. O argumento relacionado a língua deverá ser enviado de acordo com uma lista de códigos baseados na ISO-639-1, que pode ser encontrado no site da própria API: https://cloud.google.com/translate/docs/languages

Observação: O texto que for enviado a função para ser traduzido, terá seu idioma detectado automaticamente, ou seja, a língua do elemento que for enviado não precisará ser especificada.

```
Saída - JavaDB (run)
run:
Hello my friends!
CONSTRUÍDO COM SUCESSO (tempo total: 1 segundo)
```

Figura 20. Resultado Final

2) Conversão de Moeda

Em primeiro lugar, a função de conversão moeda deverá ser instanciada, como pode ser observado na linha 57 da Figura 21. A função deverá receber três argumentos, sendo eles:

- Moeda Inicial: A moeda origem, ou seja, a moeda do valor que será enviado;
- Moeda Final: A moeda que tem por objetivo ser convertida;
- Valor Inicial: O valor que deverá ser convertido.

Após os valores serem enviados para a função, ela retornará o valor final convertido. Como pode ser observado na Figura 22.

```
56
57          conversorMoedas conversorMoeda = new conversorMoedas();
58
59          String moedaInicial = "USD";
60          String moedaFinal = "BRL";
61          Double valorInicial = 5.00;
62
63          System.out.println("O resultado da conversão é de: "
64                  + conversorMoeda.converterMoeda(moedaInicial, moedaFinal, valorInicial) + ".");
65
```

Figura 21. Instanciamento da função e chamada

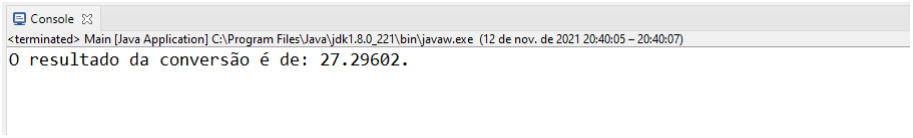

Figura 22. Resultado Final

3) Conversão de Medidas

Primeiramente, a função de conversão de data e hora deverá ser instanciada, como na linha 35 da Figura 23. A função receberá um argumento sobre o local que tem por objetivo ser convertido, e este deverá ser no formato *String* , a lista dos locais que podem serem utilizados pela função, pode ser encontrado no site do pacote java.time: https://docs.oracle.com/javase/8/docs/api/java/time/package-summary.html.

Por fim, a biblioteca irá retornar a data e hora totalmente convertida. Como pode ser observado na Figura 24.

```
56
57          conversorMoedas conversorMoeda = new conversorMoedas();
58
59          String moedaInicial = "USD";
60          String moedaFinal = "BRL";
61          Double valorInicial = 5.00;
62
63          System.out.println("O resultado da conversão é de: "
64                  + conversorMoeda.converterMoeda(moedaInicial, moedaFinal, valorInicial) + ".");
65
```

Figura 23. Instanciamento da função e chamada

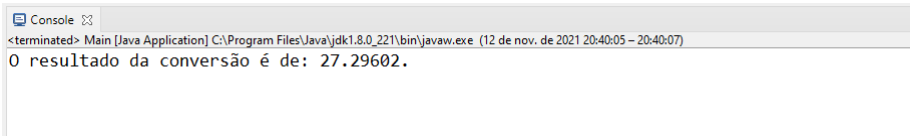

Figura 24. Resultado Final

4) Conversão de Data e Hora

Foram desenvolvidas quatro funções para a conversão de medidas, sendo elas:

• Litro Para Galão;

• Galão Para Litro;

• Libra Para Quilograma;

• Quilograma Para Libra;

Para o uso de cada uma delas, primeiramente, cada uma delas deverá ser instanciada, como pode ser observado na linha 41 até a linha 44 da Figura 25. A função receberá apenas um argumento no formato *Double*, que deverá ser o valor que será convertido.

Por fim, a biblioteca irá o valor final totalmente convertido. Como pode ser observado na Figura 26.

```
40
41          converterLitroParaGalao litroParaGalao = new converterLitroParaGalao();
42          converterGalaoParaLitro galaoParaLitro = new converterGalaoParaLitro();
43          converterLibraParaQuilograma libraParaKg = new converterLibraParaQuilograma();
44          converterQuilogramaParaLibra kgParaLb = new converterQuilogramaParaLibra();
45
46
47          Double galao = 8.0;
48          Double litro = 2.0;
49          Double libra = 5.0;
50          Double kg = 7.0;
51
52          System.out.println("O resultado da conversão de litro para galão é de: " + litroParaGalao.converter(galao) + ".");
53          System.out.println("O resultado da conversão de galão para litro é de: " + galaoParaLitro.converter(litro) + ".");
54          System.out.println("O resultado da conversão de libra para quilograma é de: " + libraParaKg.converter(libra) + ".");
55          System.out.println("O resultado da conversão de quilograma para libra é de: " + kgParaLb.converter(kg) + ".");
56
```

Figura 25. Instanciamento da função e chamada

```
Console
<terminated> Main [Java Application] C:\Program Files\Java\jdk1.8.0_221\bin\javaw.exe (12 de nov. de 2021 21:03:50 – 21:03:50)
O resultado da conversão de litro para galão é de: 2.113376.
O resultado da conversão de galão para litro é de: 7.57082.
O resultado da conversão de libra para quilograma é de: 2.26796.
O resultado da conversão de quilograma para libra é de: 15.432339999999998.
```

Figura 26. Resultado Final