CONGRESSO CATARINENSE DE CIÊNCIA DA COMPUTAÇÃO
2019. Ano 6, Vol. 6. ISSN: 2319-0655

CRIAÇÃO DE UM FRAMEWORK TRANSACIONAL UTILIZANDO A API DO PAGSEGURO

Vinício Gramkow[1], Rodrigo Curvello[1]
[1]Instituto Federal Catarinense
viniciogramkow14@gmail.com, rodrigo.curvello@gmail.com

Abstract. *This meta-article describe the implementation of a Java language framework, an object-oriented programming II, which can be used for an API provided by PagSeguro, where the user can create, cancel or cancel a sale.*
Key-words: *Framework; PagSeguro; Programming Oriented Objects;*

Resumo. *Este meta-artigo descreve a implementação de um framework na linguagem de programação Java, na matéria de Programação Orientada a objeto II, utiliza para isto a API disponibilizada pelo PagSeguro, onde o usuário pode criar, cancelar ou estornar uma venda.*
Palavras-chave: *Framework; PagSeguro; Programação Orientada Objetos;*

1. Introdução

Muitas das vezes quando precisamos criar um novo projeto de software nos deparamos com diversas dificuldades em resolver problemas e erros que possam vir a surgir. No entanto, muitas das vezes os erros que aparecem já possuem uma solução pronta e padronizada, neste momento entram os conceitos de programação orientada a objeto juntamente com os designs patterns.

Diante disto foi proposto na matéria de Programação Orientada a Objetos II do Instituto Federal Catarinense – Campus Rio do sul a implementação de um framework transacional utilizando os conceitos de programação orientada a objetos juntamente com o design pattern fluent interface, a intuição é disponibilizar para outros desenvolvedores uma implementação de métodos de pagamento de forma simples e segura, e para isto fazendo-se o uso da API disponibilizada pelo PagSeguro, da linguagem de programação Java e da IDE eclipse.

2. Framework

Framework é um template com diversas funções que podem ser usadas pelo desenvolvedor. Com ele, é desnecessário gastar tempo para reproduzir a mesma função em diferentes projetos, auxiliando em um gerenciamento ágil de projetos. Em outras palavras, ele é uma estrutura base, uma plataforma de desenvolvimento, como uma espécie de arcabouço. Os frameworks fazem com que você não tenha que se preocupar

em ficar reescrevendo códigos, podendo focar somente na resolução de problemas, ou seja, direcionando seus esforços para o objetivo final.

3. Desing Pattern

Design patterns (Padrões de projeto) são soluções de templates abstratas de alto nível. Pense nelas como um "blueprint" (desenho técnico ou documentação de uma arquitetura, etc.) para soluções e não como uma solução por si própria. Você não achará um framework que você poderá simplesmente aplicar para a sua aplicação; ao invés disso, você chegará ao design patterns através da refatoração do seu código e generalização do seu problema. Design patterns não são somente aplicados em desenvolvimento de software; design patterns podem ser encontrados em diversas áreas da vida, da engenharia até da arquitetura. Em fato, foi o arquiteto Christopher Alexander que introduziu a ideia de patterns em 1970 para construir um vocabulário comum para discussões sobre design.

4. Fluent Interface

Fluent Interface é uma forma de nomear métodos para serem usados em uma construção que dê a impressão de estar escrevendo um texto. Ele desiste de uma nomenclatura comum e padronizada para adotar algo que faça sentido quando o método for usado tentando simular um texto fluente em língua humana. É uma forma de deixar o código mais declarativo. Para conseguir isto estes métodos retornam sempre o objeto que ele está manipulando, assim o próximo método pode ser usado em sequência. Isto é chamado de encadeamento de método.

O quadro 1 apresenta a utilização do Fluent Interface na implementação do projeto, o objeto LinkConexao é instanciado e chamado os dois construtores.

O quadro 2 apresenta a criação dos construtores na classe LinkConexao.

```
1  public String Link(String dominio, String endPoint) {
2          LinkConexao con = new LinkConexao();
3          return con.dominio(dominio).endPoint(endPoint).toString();
4  }
```

Quadro 1 – Usando o Fluent Interface para criar o link e fazer a conexão com a API do PagSeguro.

Fonte – Acervo do Autor.

```
1  public LinkConexao dominio(String dominio) {
2          this.dominio = dominio;
```

```
3         return this;
4     }
5     public LinkConexao endPoint(String endPoint) {
6         this.endPoint = endPoint;
7         return this;
8     }
```

Quadro 2 – Construtores pertencentes à classe LinkConexao.

Fonte – Acervo do Autor.

5. Implementado a API do PagSeguro

A partir dos conceitos, foi desenvolvido um *framework* que contém implementações de criar, cancelar ou estornar uma venda, isto utilizando a API disponibilizada pelo PagSeguro.

A classe Checkout é responsável por fazer a conexão com a API, a conexão é feita utilizando a biblioteca HttpUrlConnection do Java utilizando o método POST.

Esta classe contém o método conectar(figura 1), nele o usuário por parâmetros passa um e-mail previamente cadastrado no PagSeguro como vendedor ou empresarial, um token de segurança gerado pelo PagSeguro, um domínio do PagSeguro (sandbox.pagseguro) para um ambiente de testes, ou (pagseguro) para o ambiente real, um endPoint da API (checkout) para criar uma venda, (transactions/cancels) para cancelar uma transação, (transactions/refunds) para estornar uma transação.

Além disso, o método ainda por parâmetros recebe os dados, previamente criados pela classe criar dados (figura 2), devidamente de acordo com o endPoint anteriormente passado ao método.

```java
public String conectar(String email, String token, String dominio, String endPoint, String dados) {
    CriarLink link = new CriarLink();
    try {
        String url = link.Link(dominio, endPoint) + "?email=" + email + "&token=" + token;
        URL obj = new URL(url);

        HttpURLConnection con = (HttpURLConnection) obj.openConnection();

        con.setReadTimeout(15000);
        con.setConnectTimeout(15000);
        con.setDoOutput(true);
        con.setDoInput(true);
        con.setRequestMethod("POST");

        DataOutputStream wr = new DataOutputStream(con.getOutputStream());
        wr.write(dados.getBytes());
        wr.flush();
        wr.close();

        BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream(), "ISO-8859-1"));
        String inputLine;
        StringBuffer response = new StringBuffer();

        while((inputLine = in.readLine()) != null) {
            response.append(inputLine);
        }
        in.close();

        return response.toString();
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}
```

Figura 1 – Método conectar.

Fonte: Acervo do Autor.

```java
public class CriarDados {

    public String checkout( List<Itens> item, Pessoa pessoa, Endereco endereco, String reference, int shippingType, String shippingCost, String extraAmount) {
        CriarDadosCheckout checkout = new CriarDadosCheckout();
        return checkout.criar(item, pessoa, endereco, reference, shippingType, shippingCost);
    }

    public String cancelarTransacao(String code) {
        CriarDadosCancelarTransacao cancels = new CriarDadosCancelarTransacao();
        return cancels.criar(code);
    }

    public String estornar(String code, String value) {
        CriarDadosEstorno refound = new CriarDadosEstorno();
        return refound.criar(code, value);
    }

}
```

Figura 2 – Classe de criação dos dados.

Fonte: Acervo do autor.

6.Conclusão

Após conclusão da implementação foi possível obter resultados satisfatórios, realizando criação de vendas (figura 3), cancelamento e estorno, a partir do *framework* criando os dados conforme o endPoint informado no momento da execução.

A utilização do padrão de projeto Fluent Interface juntamente com a linguagem Java, facilitou a criação do link para a conexão da API, anteriormente conceituado na matéria de Programação Orientada a Objetos II.

Por fim, fica uma possível ideia de implementação de outros métodos referentes a API, como consultas para o vendedor.

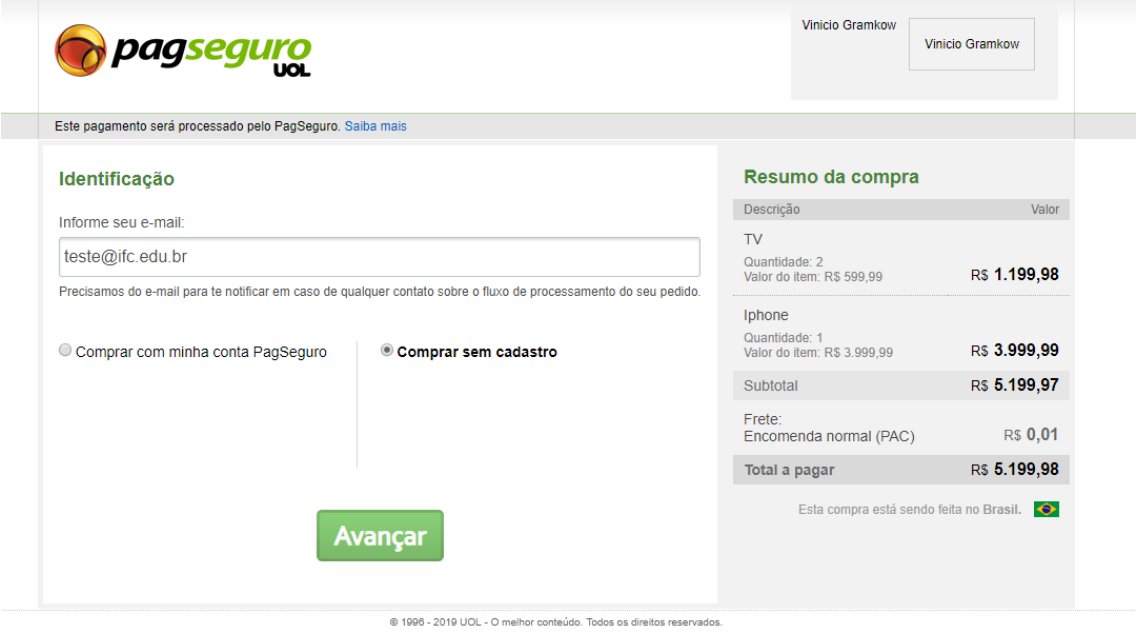

Figura 3 – Finalização da venda (tela do cliente) gerada pela API.
Fonte: Acervo do Autor.

7. Referências

CONSULTING, Gaea. **Entenda o que é Framework.** Disponível em: <https://www.princiweb.com.br/blog/programacao/design-patterns/o-que-sao-design-patterns.html>.>. Acesso em: 04 jul. 2019.

SCHISSATO, Jéssica; PEREIRA, Rodolfo. **O que são Design Patterns?** 2012. Disponível em: <https://www.princiweb.com.br/blog/programacao/design-patterns/o-que-sao-design-patterns.html>. Acesso em: 04 jul. 2019.