

UNIVERSIDADE FEDERAL DO RIO GRANDE - FURG
ENGENHARIA DE COMPUTAÇÃO
SISTEMAS DISTRIBUÍDOS

IGOR OLIVEIRA DE SOUSA - 118301
THIAGO JANSEN SAMPAIO COSTA - 116502

RELATÓRIO
Implementação com API do MercadoPago em PHP

RIO GRANDE
2021

IGOR OLIVEIRA DE SOUSA - 118301
THIAGO JANSEN SAMPAIO COSTA - 116502

RELATÓRIO

Implementação com API do MercadoPago em PHP

**Trabalho apresentado à disciplina de
Sistemas Distribuídos como requisito
para obtenção de nota.**

Prof. Bruno Lopes Dalmazo

RIO GRANDE
2021

1. INTRODUÇÃO

O trabalho consiste numa implementação cliente/servidor utilizando a API do MercadoPago para criação de uma página de venda com checkout transparente. A partir da aplicação, o usuário consegue realizar a compra do produto via cartão de crédito de forma parcelada ou não diretamente na página. Enquanto o vendedor recebe seu pagamento na sua conta do MercadoPago.

Este relatório visa explicar detalhadamente nossa aplicação. O documento inclui detalhes da implementação, desafios encontrados, testes e funcionamento.

2. DESENVOLVIMENTO

Na aplicação, foi utilizado o framework Bootstrap para desenvolver o front-end e o PHP para o back-end. A partir da documentação da API do MercadoPago, foi criada uma integração entre o sistema de pagamento deles e nossa aplicação.

Um dos pontos mais importantes numa integração via API de pagamentos são as chaves pública e privada, chamadas de Public Key e Access token, respectivamente, no MercadoPago. A chave pública é utilizada para acessar os recursos que necessita no frontend. Com ela é possível coletar os dados dos cartões de crédito e convertê-los em um token que pode ser enviado aos servidores de forma segura. Enquanto que com a chave privada é possível chamar o resto das APIs a partir da nossa aplicação, como processar um pagamento e realizar um reembolso. Essas credenciais foram disponibilizadas através do MercadoPago Developer e é pré-requisito para a integração.

Além das credenciais, também foram utilizadas as bibliotecas do MercadoPago, a MercadoPago.js. A partir dela é possível cuidar dos dados sensíveis dos clientes, cumprir com os padrões de segurança e manter sempre atualizado.

2.1. index.html

O index.html é o arquivo 'main' de nossa aplicação, sua estrutura é um html puro utilizando a framework bootstrap para ser mais dinâmico.

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>SlumberShop</title>
  <!-- Bootstrap core CSS -->
  <link
    rel="stylesheet"
    href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
    integrity="sha384-MCW98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
    crossorigin="anonymous"
  />
  <!-- Custom styles for this template -->
  <link href="style.css" rel="stylesheet" />
</head>

<body>
  <div ...
  </div>

  <div class="pricing-header px-3 py-3 pt-md-5 pb-md-4 mx-auto text-center">...
  </div>

  <div class="container">...
  </div>

  <!-- Bootstrap core JavaScript
  | | |
  | | | -->
  <!-- Placed at the end of the document so the pages load faster -->

  <script
    src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"
    integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy"
    crossorigin="anonymous"
  ></script>
  <script
    src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
    integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abTTE1Pi6jizo"
    crossorigin="anonymous"
  ></script>
  <script
    src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"
    integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk6WLaUAdn689aCwoqbBJiSnjAK/L8WvCWPIpM49"
    crossorigin="anonymous"
  ></script>
</body>
</html>

```

Imagem 1. Código index.html

2.2. checkout.php e viacep.php

Posteriormente a navegação nos leva para a página checkout onde o usuário deve passar suas informações de endereço e do cartão, tal processo é feito em duas partes. A primeira é o 'submit' do cep, como pode ser visto na imagem 3 , fazemos o include do arquivo viaCep (que pode ser visto na imagem 2), para puxar as informações recebidas via API VIACEP, tais informações é colocada nos inputs correspondentes, para o funcionamento desse seguimento é necessário que o usuário informe o CEP e realize o submit.

```

<?php
// Realizamos o include_once para conseguirmos nos comunicar com o arquivo responsavel pela consulta do CEP
include_once(' ../lib/viaCep.php');
?>

```

Imagem 2. Tag php do código checkout.php

```

<form class="needs-validation" novalidate method="POST" name="cep" id="cep">
  <div class="row">
    <div class="col-md-6 mb-3">
      <label for="zip">CEP</label>
      <div style="display: flex;">
        <input type="text" class="form-control" name="zip" id="zip" value="<?php echo $address->cep?>"
          placeholder="" required>
        <div class="invalid-feedback">
          Por favor, adicione seu CEP.
        </div>
      </div>
      <button class="" type="submit">Buscar</button>
    </div>
  </div>
</form>

```

Imagem 3. Primeiro formulário do código checkout.php

```

<!-- Form principal da Aplicação onde as primeiras interações com MercadoPago São feitas -->
<form class="needs-validation" novalidate method="POST" name="pay" id="pay" action="./controllers.php">
  <div class="mb-3">
    <label for="address">Endereço</label>
    <input type="text" class="form-control" name="addr" id="addr" value="<?php echo $address->logradouro?>"
      placeholder="Rua General Portinho, 464" required>
    <div class="invalid-feedback">
      Por favor, adicione seu endereço completo.
    </div>
  </div>
  <div class="row">
    <div class="col-md-6 mb-3">
      <label for="number">Número</label>
      <input type="text" class="form-control" name="number" id="number" placeholder="123">
    </div>
    <div class="col-md-6 mb-3">
      <label for="address2">Complemento<span class="text-muted">(Optional)</span></label>
      <input type="text" class="form-control" name="address2" id="address2" placeholder="Apartment or suite">
    </div>
  </div>
  <div class="row">
    <div class="col-md-5 mb-3">
      <label for="country">País</label>
      <select class="custom-select d-block w-100" name="country" id="country" required>
        <option value="">Opções ... </option>
        <option selected>Brasil</option>
      </select>
      <div class="invalid-feedback">
        Por favor, selecione um país válido.
      </div>
    </div>
    <div class="col-md-4 mb-3">
      <label for="state">Estado</label>
      <select class="custom-select d-block w-100" value="<?php echo $address->uf ?>" name="state" id="state"
        required>
        <option>Opções ... </option>
        <option selected>
          <?php echo $address->uf ?>
        </option>
      </select>
      <div class="invalid-feedback">
        Por favor, selecione um estado válido.
      </div>
    </div>
  </div>
</form>

```

Imagem 4. Parte do segundo formulário do código checkout.php

Posteriormente temos um segundo ‘form’ (podendo ser observado nas imagens 5 e 6), ele engloba as informações de endereço oriundas da API, e também informações pessoais e do cartão disponibilizadas pelo usuário. Aqui ao contrário do ‘forms’ anterior, temos uma interação em tempo real com a utilização do

código 'api.js' que irá colocar a bandeira do cartão, o select das parcelas e ainda possuem alguns inputs 'hidden' que são usados para armazenar certas informações vitais que ou são usadas ou vem da API do MercadoPago.

```
<h4 class="mb-3">Informações pessoais</h4>
<div class="row">
  <div class="col-md-6 mb-3">
    <label for="firstName">Nome</label>
    <input type="text" class="form-control" name="firstName" id="firstName" placeholder="" value=""
      required>
    <div class="invalid-feedback">
      Por favor, adicione seu nome.
    </div>
  </div>
  <div class="col-md-6 mb-3">
    <label for="lastName">Sobrenome</label>
    <input type="text" class="form-control" name="lastName" id="lastName" placeholder="" value="" required>
    <div class="invalid-feedback">
      Por favor, adicione seu sobrenome.
    </div>
  </div>
</div>
<div class="mb-3">
  <label for="email">Email</label>
  <input type="email" name="email" class="form-control" id="email" placeholder="you@example.com" required>
  <div class="invalid-feedback">
    Por favor, adicione seu email.
  </div>
</div>
<hr class="mb-4">
<h4 class="mb-3">Informações do Cartão</h4>
<!-- Parte do input que trabalha com MercadoPago -->
<div class="row">
  <div class="col-md-6 mb-3">
    <label for="cc-name">Nome do titular</label>
    <input type="text" class="form-control" id="cardholderName" data-checkout="cardholderName"
      placeholder="" required />
    <small class="text-muted">Como está escrito no cartão</small>
    <div class="invalid-feedback">
      Nome do título é obrigatório
    </div>
  </div>
  <div class="col-md-6 mb-3">
    <label for="cc-number">Número do cartão</label>
    <div class="input-group">
      <div class="input-group-prepend">
        <!-- Aparti do uso a api retorna a bandeira do cartão e colocamos aki -->
        <span class="input-group-text">
          <div class="brand"></div>
        </span>
      </div>
    </div>
  </div>
</div>
```

Imagem 5. Parte do segundo formulário do código checkout.php

```

    <!-- Os inputs que envolve o cartão possuem varias caracteristicas para proteção -->
    <input type="text" class="form-control" id="cardNumber" data-checkout="cardNumber" placeholder=""
        onselectstart="return false" onpaste="return false" onCopy="return false" onCut="return false"
        onDrag="return false" onDrop="return false" autocomplete=off required />
    <div class="invalid-feedback">
        Número do cartão é obrigatório
    </div>
</div>
</div>
<div class="row">
    <div class="col-md-3 mb-2"><label for="cc-expiration">Mês da Validade</label><input type="text"
        class="form-control" id="cardExpirationMonth" data-checkout="cardExpirationMonth" placeholder=""
        onselectstart="return false" onpaste="return false" onCopy="return false" onCut="return false"
        onDrag="return false" onDrop="return false" autocomplete=off required /></div>
    <div class="col-md-4 mb-2"><label for="cc-expiration">Ano da Validade</label><input type="text"
        class="form-control" id="cardExpirationYear" data-checkout="cardExpirationYear" placeholder=""
        onselectstart="return false" onpaste="return false" onCopy="return false" onCut="return false"
        onDrag="return false" onDrop="return false" autocomplete=off required /></div>
    <div class="col-md-5 mb-3">
        <label for="cc-cvv">Código de segurança (CVV)</label>
        <input type="text" class="form-control" id="securityCode" data-checkout="securityCode" placeholder="123"
            onselectstart="return false" onpaste="return false" onCopy="return false" onCut="return false"
            onDrag="return false" onDrop="return false" autocomplete=off required />
        <div class="invalid-feedback">
            Código de segurança é obrigatório
        </div>
    </div>
</div>
<div class="row">
    <div class="col-md-6 mb-3">
        <!-- O mercado Pago fornece o tipo dos documentos que aceita -->
        <label for="docType">Documento</label>
        <select class="custom-select d-block w-100" id="docType" data-checkout="docType"></select>
    </div>
    <div class="col-md-6 mb-3">
        <label for="docNumber">Número do Documento</label>
        <input type="text" class="form-control" id="docNumber" data-checkout="docNumber" placeholder=""
            required>
    </div>
</div>
<!-- Depois verificar a bandeira e o valor o MercadoPago fornece as parcelas já com juros -->
<div hidden="true" id="parcelas" class="mb-3"><label for="installments">Parcelas</label>
<select id="installments" class="form-control" name="installments"></select>
</div>
<input type="hidden" name="amount" id="amount" value="18390.85" />
<input type="hidden" name="description" value="SlumberShop" />
<input type="hidden" name="paymentMethodId" />
<hr class="mb-4">
<button class="btn btn-primary btn-lg btn-block" type="submit">Finalizar compra</button>
</form>

```

Imagem 6. Resto do segundo formulário do código checkout.php

Para finalizar essa seção, temos o código “viaCep.php” que é o responsável pela aquisição das informações de endereço, sua utilização é extremamente simples basta (como está comentado em código), utilizar uma url fornecida em sua bibliografia com o cep fornecido, assim a API retorna um json com as informações.

```

<?php
// Pré-setamos a variavel address para nao ocorrer erros
$address = (object) [
    'cep' => '',
    'logradouro' => '',
    'bairro' => '',
    'localidade' => '',
    'uf' => '',
];
$_SESSION["zip"] = "";
// É realizado um storage em session do zip para não ocorrer erros
// E a verificação se alguma submissão post foi realizada
if ( isset ( $_POST['zip'] ) ) {
    $cep = $_POST['zip'];
    $cep = preg_replace('/^[^0-9]/', '', $cep);
    if ( preg_match('/^[0-9]{5}-?[0-9]{3}$/', $cep) ) {
        // Depois de realizada uma verificação por expressão regulares
        // Utilizamos a API com o CEP informado
        $url = "https://viacep.com.br/ws/{$cep}/json/";

        $address = json_decode(file_get_contents($url));
        // Decodificamos e salvamos em session o cep
        $_SESSION["zip"] = $address->cep;
    } else {
        $address->cep = "CEP inválido!";
    }
}
?>

```

Imagem 7. Código viaCep.php

2.3. controllers.php

Quando concluído o formulário e realizado o submit vamos ao código controllers.php ele tem uma função bem simples, primeiramente pegar todas as variáveis mandadas pelo método post, já filtrando e em seguidas salva algumas informações pessoais e de endereço em session, e finaliza seu trabalho criando o método payment do mercadopago com as informações enviadas, já definindo em session o seu resultado.


```

1  <?php
2  session_start();
3  require(' ../config/config.php');
4  require (' ../lib/vendor/autoload.php');
5
6  // Primeiro passo é o recolhimento e filtragem de todos os dados do form pelo metodo post
7
8  #Variables
9  $email=filter_input(INPUT_POST,'email',FILTER_VALIDATE_EMAIL);
10 $cardNumber=filter_input(INPUT_POST,'cardNumber',FILTER_DEFAULT);
11 $securityCode=filter_input(INPUT_POST,'securityCode',FILTER_DEFAULT);
12 $cardExpirationMonth=filter_input(INPUT_POST,'cardExpirationMonth',FILTER_DEFAULT);
13 $cardExpirationYear=filter_input(INPUT_POST,'cardExpirationYear',FILTER_DEFAULT);
14 $cardholderName=filter_input(INPUT_POST,'cardholderName',FILTER_DEFAULT);
15 $docType=filter_input(INPUT_POST,'docType',FILTER_DEFAULT);
16 $docNumber=filter_input(INPUT_POST,'docNumber',FILTER_DEFAULT);
17 $installments=filter_input(INPUT_POST,'installments',FILTER_DEFAULT);
18 $amount=filter_input(INPUT_POST,'amount',FILTER_DEFAULT);
19 $description=filter_input(INPUT_POST,'description',FILTER_DEFAULT);
20 $paymentMethodId=filter_input(INPUT_POST,'paymentMethodId',FILTER_DEFAULT);
21 $token=filter_input(INPUT_POST,'token',FILTER_DEFAULT);
22
23
24 $firstName=filter_input(INPUT_POST,'firstName',FILTER_DEFAULT);
25 $lastName=filter_input(INPUT_POST,'lastName',FILTER_DEFAULT);
26 // $zip=filter_input(INPUT_POST,'zip',FILTER_DEFAULT);
27 $address=filter_input(INPUT_POST,'addr',FILTER_DEFAULT);
28 $number=filter_input(INPUT_POST,'number',FILTER_DEFAULT);
29 $complement=filter_input(INPUT_POST,'address2',FILTER_DEFAULT);
30 $country=filter_input(INPUT_POST,'country',FILTER_DEFAULT);
31 $state=filter_input(INPUT_POST,'state',FILTER_DEFAULT);

```

Imagem 8 Recolhimento de variáveis do controllers.php.

```

33 $infos = [
34     'firstName' => $firstName,
35     'lastname' => $lastName,
36     'zip' => $_SESSION['zip'],
37     'email' => $email,
38     'address' => $address,
39     'number' => $number,
40     'complement' => $complement,
41     'country' => $country,
42     'state' => $state,
43     'price' => $amount,
44     'parcelas' => $installments];
45
46 // Guardamos as informações em session para serem utilizadas depois
47 $_SESSION['infos']=$infos;
48
49 #Method
50 // Aqui onde a requisição de pagamento é feito ja levando as informações necessarias e realizando o pagamento
51 // Retornando um todas as informações possíveis da transação
52 // Necessitamos criar um objeto tipo payment, passar as informações e chamar a função save onde a requisição realizada
53 MercadoPago\SDK::setAccessToken(SAND_TOKEN);
54 $payment = new MercadoPago\Payment();
55 $payment->transaction_amount = $amount;
56 $payment->token = $token;
57 $payment->description = $description;
58 $payment->installments = $installments;
59 $payment->payment_method_id = $paymentMethodId;
60 $payment->payer = array(
61     "email" => $email
62 );
63 $payment->save();
64
65 #Result
66 $_SESSION['payment']=$payment;
67 header("location: http://localhost/api/checkout/result.php");
68
69 //echo '<pre>',print_r($payment),'</pre>';
70 ?>

```

Imagem 9 Chamada do método payment do controllers.php.

2.4. result.php

A última página de nossa aplicação é a result.php ela tem um layout simples pois é focada em apresentação dos dados da transação. Inicialmente temos que ressaltar o uso da classe `ClassException`, onde todas as mensagens de erros ou status estão pré-estabelecidas, outra coisa é a utilização de funcionamentos do MercadoPago como mostra na linha da imagem 10.

```
1  <?php
2
3  use Doctrine\Common\Collections\Expr\Value;
4
5
6  // Aqui puxamos as informações necessarias para amostragem do status
7  require('../lib/vendor/autoload.php');
8  $exception=new \Classes\ClassException();
9  session_start();
10
11  $info = $_SESSION['infos'];
12  $status = $_SESSION['payment'];
13  ?>
```

Imagem 10 Tag php do result.php.

Temos duas apresentações nesta página, a primeira é um card que contém informações enviadas pelo cliente, salvas em sessão, e algumas retiradas do status da requisição da api. Já a segunda parte é a mensagem do resultado, utilizando a `ClassException` e um css para diferenciar os tipos de status (success, alert e error).

```

<!DOCTYPE html>
<html lang="en">
<head>...
</head>
<body class="bg-light">
<div
  class="d-flex flex-column flex-md-row align-items-center p-3 px-md-4 mb-3 bg-white border-bottom shadow-sm"
  >
  <h5 class="my-0 mr-md-auto font-weight-normal">SlumberShop</h5>
  <a
    class="btn btn-outline-primary"
    href="http://localhost/api/"
    >Home</a>
  </div>
<div class="container">
  <div class="py-5 text-center">
    <h2> Status:</h2>
  </div>
  <!-- Primeiramente é mostrado as informações passadas pelo cliente -->
  <div class="card">
    <h4>Clientes </h4>
    <h7><?php echo $info['firstName']. " ".$info['lastName']. " : ".$status->card->cardholder->identification->type." : ".$status->card->cardholder->identification->number ?></h7>
    <h4>Endereço: </h4>
    <h7><?php echo "Rua: ".$info['address']. " , ".$info['number'] ?></h7>
    <h7><?php echo "Complemento: ".$info['complement']. " Cep: ".$info['zip'] ?> </h7>
    <h7><?php echo "Estado: ".$info['state']. " , ".$info['country'] ?></h7>
    <h4>Valor: </h4>
    <h7><?php echo $info['parcelas']. "X de ".$status->transaction_details->installment_amount." R$ ( ".$status->transaction_details->total_paid_amount." R$ )" ?></h7>
  </div>
  <!-- E depois o Resultado da transação -->
  <div class="result">
    <?php echo <pre>' . print_r($SESSION['payment']).</pre> ?> -->
    <?php $exception->setPayment($SESSION['payment']); ?>
    <div class="<?php echo $exception->verifyTransaction()['class']; ?>">
      <?php echo $exception->verifyTransaction()['message']; ?>
    </div>
  </div>
  <!-- Bootstrap core JavaScript -->
  <!-- Placed at the end of the document so the pages load faster -->
  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js" integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy" cross
  <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q81/X+96Dz08rT7abK41JStoIAqVgRVzpbzo5smXKp4YfRvH+8abTTE1P6j1zo" crossorigin="anonymous"></s
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js" integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBkEWLauAadn689aCwoqBj1SnJAK/L8VWCP1PM49" cr
</body>
<footer class="my-5 pt-5 text-muted text-center text-small">
  <p class="mb-1">&copy; 2017-2018 Company Name</p>
  <ul class="list-inline">
    <li class="list-inline-item"><a href="#">Privacy</a></li>
    <li class="list-inline-item"><a href="#">Terms</a></li>
    <li class="list-inline-item"><a href="#">Support</a></li>
  </ul>
</footer>
</html>

```

Imagem 11 Código result.php.

2.5. api.js

Por último, mas não menos importante, temos o código api.js, ele realiza algumas funcionalidades em tempo real, para isso temos um sistema de inicialmente verificar a interação do usuário com o componente com o “*doc.querySelector()*”, após isso utilizamos o comando ‘*addEventListener()*’, para chamadas de funções específicas de cada caso, para mero detalhamento temos função para recolhimento da bandeira do cartão e das parcelas disponíveis, depois de seis dígitos escritos, e temos uma função de *getToken()* realizada ao evento de submissão do form.

```

1  (function (win, doc) {
2      'use strict';
3
4      // Primeiro Setamos a chave de confirmação com MercadoPago
5      //Public Key
6      window.Mercadopago.setPublishableKey(
7          'TEST-2330c5b5-1047-4d33-b663-3c9b18d6680e'
8      );
9      //Docs Type
10     window.Mercadopago.getIdentificationTypes();
11
12     //Nas funções subsequentes utilizamos a logica de eventos, porem sempre verificamos se componente que queremos existe
13     //Como podemos ver nas linhas 38 e 99
14     //Card bin
15     function cardBin(event) {
16         // Após a verificação de evento realizada na linha 3 verificamos se ja foi digitado pelo menos 6 digitos do cartão
17         // Pois após 6 digitos podemos retirar via api a bandeira e as parcelas, como pode ser vista nas linhas 23 e 29
18         // Com as fuções getPaymentMethod e getInstallments respectivamente
19         let textLength = event.target.value.length;
20         // console.log(textLength);
21         if (textLength ≥ 6) {
22             let bin = event.target.value.substring(0, 6);
23             window.Mercadopago.getPaymentMethod(
24                 {
25                     bin: bin,
26                 },
27                 setPaymentMethodInfo // função na linha 62 responsavel por colocar a thumb do cartão no html
28             );
29             Mercadopago.getInstallments(
30                 {
31                     bin: bin,
32                     amount: parseFloat(document.querySelector('#amount').value),
33                 },
34                 setInstallmentInfo // função na linha 45 responsavel por fazer um select com as parcelas e colocar no html
35             );
36         }
37     }
38     if (doc.querySelector('#cardNumber')) {
39         // Verificação de existencia e de evento do input do numero do cartão
40         let cardNumber = doc.querySelector('#cardNumber');
41         cardNumber.addEventListener('keyup', cardBin, false);
42     }
43
44     //Set Installments
45     function setInstallmentInfo(status, response) {
46         // Como já mencionado função responsavel por criar o select das parcelas que inicialmente fica em modo hidden
47         let label = response[0].payer_costs;
48         let installmentsSel = doc.querySelector('#installments');
49         installmentsSel.options.length = 0;
50
51         label.map(function (elem, ind, obj) {
52             let txtOpt = elem.recommended_message;
53             let valOpt = elem.installments;
54             installmentsSel.options[installmentsSel.options.length] = new Option(
55                 txtOpt,
56                 valOpt
57             );
58         });
59     }
60 }

```

Imagem 12. Código api.js

```

61 //Set Payment
62 function setPaymentMethodInfo(status, response) {
63     // Função responsável por colocar a bandeira do cartão e por retirar o hidden do select das parcelas
64     if (status === 200) {
65         const paymentMethodElement = doc.querySelector(
66             'input[name=paymentMethodId]'
67         );
68
69         doc.querySelector('#parcelas').hidden = false;
70         paymentMethodElement.value = response[0].id;
71         doc.querySelector('.brand').innerHTML =
72             '';
73     } else {
74         alert('payment method info error: ${response}');
75     }
76 }
77 //Create Token, necessario para transação do MercadoPago
78 function sendPayment(event) {
79     event.preventDefault();
80     window.Mercadopago.createToken(event.target, sdkResponseHandler);
81 }
82
83 // Após a requisição do token é verificado se deu certo, e depois o token é colocado como um input para que seja possível pega-lo no controllers.php
84 function sdkResponseHandler(status, response) {
85     if (status === 200 || status === 201) {
86         let form = doc.querySelector('#pay');
87
88         let card = doc.createElement('input');
89         card.setAttribute('name', 'token');
90         card.setAttribute('type', 'text');
91         card.setAttribute('value', response.id);
92         form.appendChild(card);
93         // finaliza o form e submete
94         form.submit();
95     }
96 }
97
98 // Verificação de existencia do forms
99 if (doc.querySelector('#pay')) {
100     let formPay = doc.querySelector('#pay');
101     // Espera o evento de submit para chamar a função de criação do token
102     formPay.addEventListener('submit', sendPayment, false);
103 }
104 }(window, document);
105

```

Imagem 13. Fim do código api.js

3. DESAFIOS ENCONTRADOS

Os maiores desafios foram acerca do entendimento do funcionamento de uma API e da documentação do MercadoPago, após consolidar bem os conceitos, o desenvolvimento fluiu melhor. Além disso, na implementação da API via CEP houve bastante erro, o que acabou levando um tempo não esperado para resolver. A princípio o tratamento dos erros parecia ser o grande empecilho, porém com a documentação do MercadoPago, ficou bastante simples essa questão.

4. FUNCIONAMENTO

Para utilizar a aplicação, é necessário um instalar um servidor local de PHP (recomendado o XAMPP com versão PHP7+) e o gerenciador de dependência Composer. Para instalar o XAMPP, acesse o site oficial do software (https://www.apachefriends.org/pt_br/index.html) e faça o download do arquivo de acordo com seu sistema operacional. Agora para instalar o Composer, acesse o site (<https://getcomposer.org/>) e siga o tutorial de download do site.

Com ambiente de desenvolvimento configurado, execute o XAMPP, clique na opção “Explorer” (Imagem 14) para abrir o diretório raiz do programa e adicione o código fonte dentro da pasta “htdocs”.

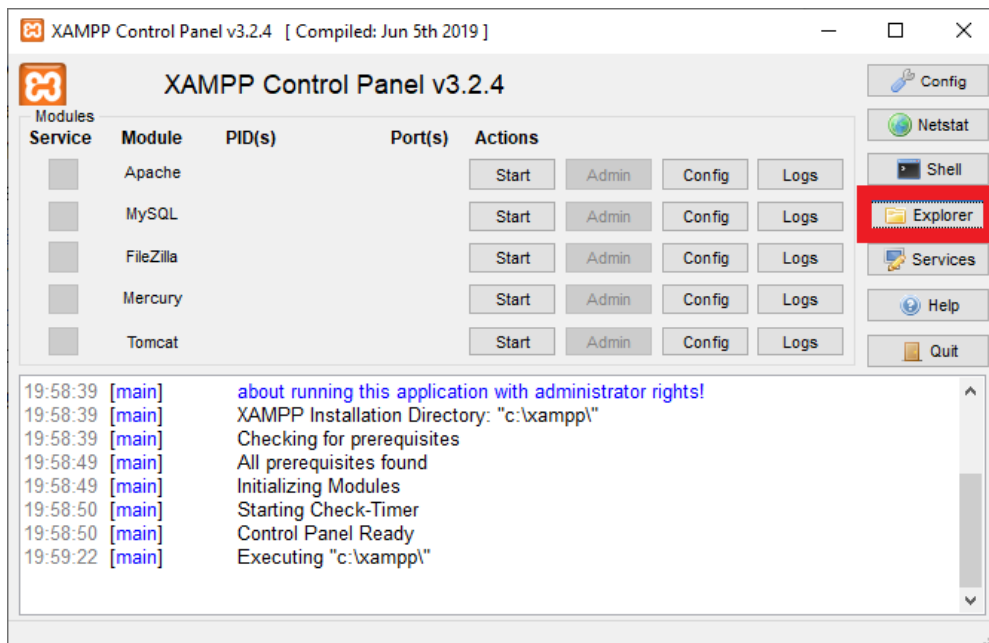


Imagem 14. Abrindo o diretório raiz do XAMPP.

Após adicionar o código fonte na pasta raiz do XAMPP, clique em “Start” (Imagem 15) para iniciar o servidor local.

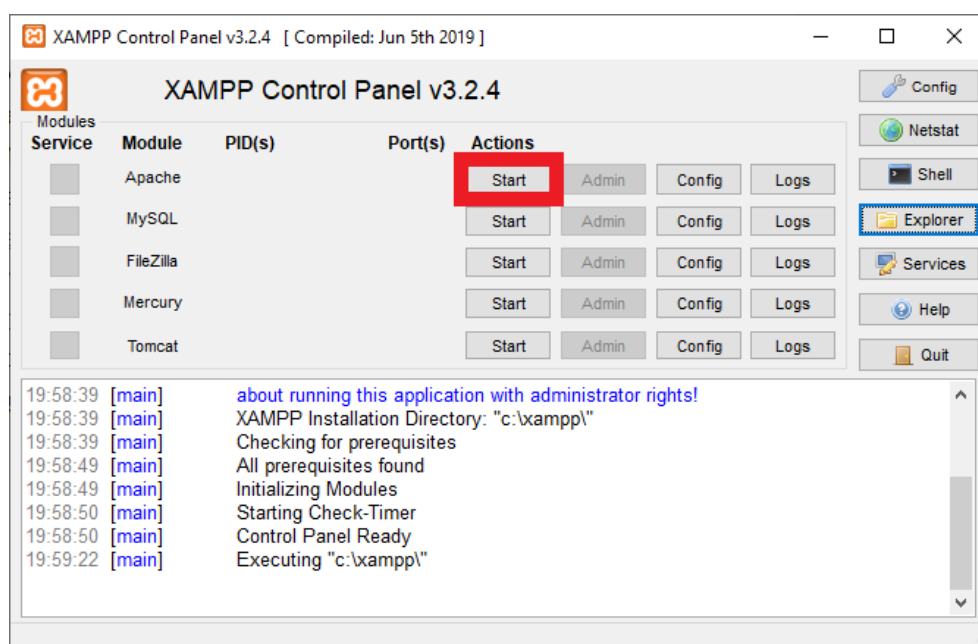


Imagem 15. Executando o servidor local no XAMPP.

Em seguida, uma aba do navegador irá se abrir no endereço “<http://localhost>”, encontre o arquivo “index” e clique sobre ele para abrir a página principal da aplicação, se necessário.

Na página inicial da aplicação, foi criado um produto meramente ilustrativo com descrição e especificações técnicas para criar uma melhor experiência de compra (Imagem 17).

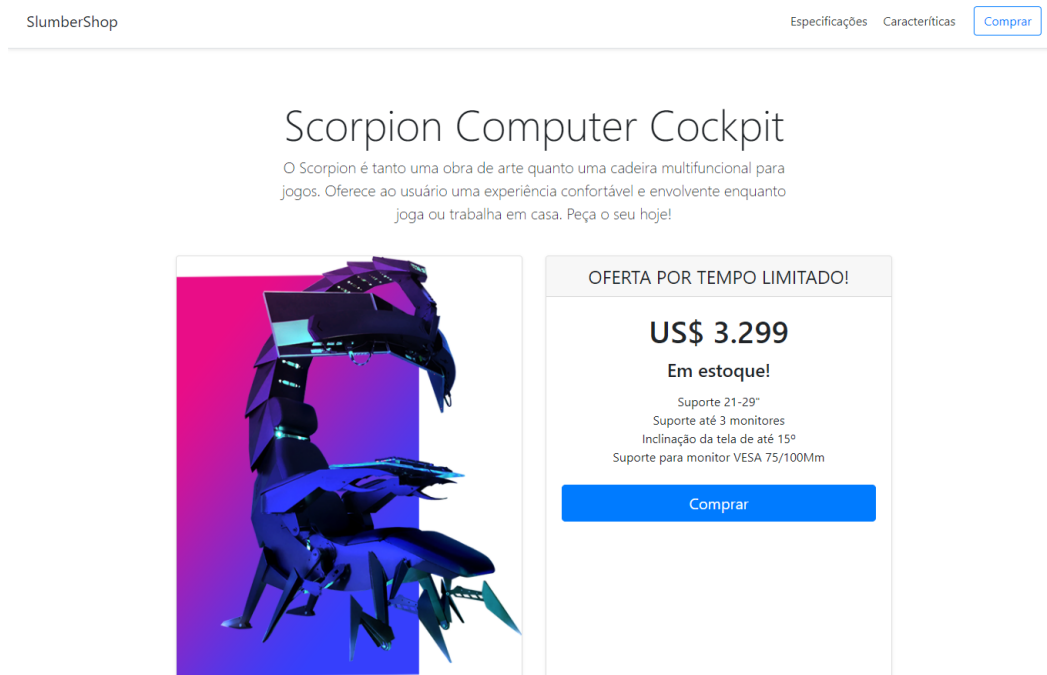


Imagem 17. Primeira seção da página inicial da aplicação.

Ao clicar em “comprar”, o usuário é redirecionado para o checkout transparente, onde deve inserir seus dados de entrega e pagamento (Imagem 18). Ao adicionar o CEP, os campos de endereço se autocompletam graças a API viacep.com.br. Na seção de pagamento, o usuário pode realizar pagamentos via cartão de créditos em parcelas de acordo com o valor do produto e a bandeira do cartão. Nesta aplicação, são aceitos cartões da bandeira Visa, Mastercard, Hipercard, America, Express, Diners, Elo e Cartão MercadoLivre.

Confirmação de Compra

Abaixo está o formulário para confirmação de sua compra, por favor preencher campos abaixo.

Informações de endereço

CEP

Buscar

Endereço

Rua General Portinho, 464

Número

123

Complemento(Opcional)

Apartamento ou casa

País

Brasil

Estado

Informações pessoais

Nome

Sobrenome

Email

you@example.com

Informações do Cartão

Nome do titular

Número do cartão

Como está escrito no cartão

Mês da Validade

Ano da Validade

Código de segurança (CVV)

123

Documento

CPF

Número do Documento

Finalizar compra

Imagem 18. Checkout transparente.

Ao finalizar o processo de pagamento, o usuário é direcionado para a página de confirmação de pedido, onde seus dados de entrega e status de pagamento são mostrados (Imagem 19).

Status:

Cliente:
Thiago Jansen; CPF : 10787997927

Endereço:
Rua: Rua Engenheiro Elmer Lawsorensen Corttheill, 123
Complemento: Cep:
Estado: RS, Brasil

Valor:
10X de 2144.19 R\$ (21441.89 R\$)

Pronto, seu pagamento foi aprovado! Você verá o nome
MERCADOPAGO na sua fatura de cartão de crédito.
Entraremos em contato com você!

Imagem 19. Página de confirmação de pedido.

5. TESTES

Por conta da aplicação ser a nível de teste, foram utilizados cartões de fictícios disponibilizados pelo MercadoPago para verificar se os pagamentos foram criados corretamente e que as mensagens de notificação estavam chegando de forma efetiva.

| Cartão | Número | Código de segurança | Data de vencimento |
|------------------|---------------------|---------------------|--------------------|
| Mastercard | 5031 4332 1540 6351 | 123 | 11/25 |
| Visa | 4235 6477 2802 5682 | 123 | 11/25 |
| American Express | 3753 651535 56885 | 1234 | 11/25 |

Para testar diferentes resultados de pagamento, utilizamos os seguintes códigos no nome do titular do cartão:

- APRO: Pagamento aprovado.
- CONT: Pagamento pendente.
- OTHE: Recusado por erro geral.
- CALL: Recusado com validação para autorizar.
- FUND: Recusado por quantia insuficiente.
- SECU: Recusado por código de segurança inválido.
- EXPI: Recusado por problema com a data de vencimento.
- FORM: Recusado por erro no formulário.

5.1. Pagamento aprovado

Informações pessoais

Nome

Thiago

Sobrenome

Jansen

Email

thiago1@hotmail.com


Informações do Cartão

Nome do titular

APRO

Como está escrito no cartão

Número do cartão

 5031433215406351

Mês da Validade

11

Ano da Validade

2025

Código de segurança (CVV)

123

Documento

CPF

Número do Documento

10787997927

Parcelas

5 parcelas de R\$ 4.033,48 (R\$ 20.167,40)

Finalizar compra

Status:

Cliente:

Thiago Jansen; CPF : 10787997927

Endereço:

Rua: Rua Engenheiro Elmer Lawsorensen Corttheill, 123

Complemento: Dois andar Cep:

Estado: RS, Brasil

Valor:

5X de 4033.48 R\$ (20167.41 R\$)

Pronto, seu pagamento foi aprovado! Você verá o nome
MERCADOPAGO na sua fatura de cartão de crédito.
Entraremos em contato com você!

5.2. Pagamento pendente

Informações pessoais

Nome

Igor

Sobrenome

Oliveira

Email


ab@gmail.com

Informações do Cartão

Nome do titular

CONT

Como está escrito no cartão



Número do cartão

5031432215406351

Mês da Validade

11

Ano da Validade

2025

Código de segurança (CVV)

123

Documento

CPF

Número do Documento

10787997927

Parcelas

6 parcelas de R\$ 3.403,53 (R\$ 20.421,18)

Finalizar compra

Status:

Cliente:

Igor Oliveira; CPF : 10787997927

Endereço:

Rua: Rua Engenheiro Elmer Lawsorense Corttheill, 123

Complemento: Dois andar Cep:

Estado: RS, Brasil

Valor:

7X de 2949,1 R\$ (20643.73 R\$)

Estamos processando o pagamento. Em até 2 dias úteis
informaremos por e-mail o resultado.

5.3. Recusado por erro geral

Informações pessoais

Nome

Igor

Sobrenome

Oliveira

Email


thiaggio1@hotmail.com

Informações do Cartão

Nome do titular

OTHE

Número do cartão

5031433215406351

Como está escrito no cartão

Mês da Validade

11

Ano da Validade

2025

Código de segurança (CVV)

123

Documento

CPF

Número do Documento

10787997927

Parcelas

8 parcelas de R\$ 2.617,25 (R\$ 20.938,00)

Finalizar compra

Status:

Cliente:
Igor Oliveira; CPF : 10787997927

Endereço:
Rua: Rua Engenheiro Elmer Lawsorensen Corttheill, 123
Complemento: Dois andar Cep:
Estado: RS, Brasil

Valor:
8X de 2617.25 R\$ (20937.98 R\$)

O cartão não processou seu pagamento

5.4. Recusado com validação para autorizar

Informações pessoais

Nome

Sobrenome

Igor

Oliveira

Email

thhiaggo1@hotmail.com

Informações do Cartão

Nome do titular

Número do cartão

CALL

VISA

4235647728025682

Como está escrito no cartão

Mês da Validade

Ano da Validade

Código de segurança (CVV)

11

2025

123

Documento

Número do Documento

CPF

10787997927

Parcelas

7 parcelas de R\$ 2.949,10 (R\$ 20.643,70)

Finalizar compra

Cliente:

Igor Oliveira; CPF : 10787997927

Endereço:

Rua: Rua Engenheiro Elmer Lawsorensen Corttheill, 123
Complemento: Dois andar Cep:
Estado: RS, Brasil

Valor:

7X de 2949.1 R\$ (20643.73 R\$)

Você deve autorizar o pagamento do valor ao Mercado Pago.

5.5. Recusado por código de segurança inválido

Informações pessoais

Nome

Sobrenome

Thiago

Jansen

Email

thiago.jcsampaio@gmail.com

Informações do Cartão

Nome do titular

Número do cartão

SECU

VISA

4235647728025682

Como está escrito no cartão

Mês da Validade

Ano da Validade

Código de segurança (CVV)

11

2025

123

Documento

Número do Documento

CPF

10787997927

Parcelas

8 parcelas de R\$ 2.617,25 (R\$ 20.938,00)

Finalizar compra

Status:

Cliente:
Thiago Jansen; CPF : 10787997927

Endereço:
Rua: Rua Engenheiro Elmer Lawsorense Corttheill, 123
Complemento: Dois andar Cep:
Estado: RS, Brasil

Valor:
8X de 2617.25 R\$ (20937.98 R\$)

Confira o código de segurança.

5.6. Recusado por saldos insuficientes

Informações pessoais

Nome

Thiago

Sobrenome

Jansen

Email

thiago.jcsampaio@gmail.com

Informações do Cartão

Nome do titular

FUND

Número do cartão

VISA

4235647728025682

Como está escrito no cartão

Mês da Validade

11

Ano da Validade

2025

Código de segurança (CVV)

123

Documento

CPF

Número do Documento

10787997927

Parcelas

1 parcela de R\$ 18.390,85 (R\$ 18.390,85)

Finalizar compra

Status:

Cliente:

Thiago Jansen; CPF : 10787997927

Endereço:

Rua: Rua Engenheiro Elmer Lawsorensen Corttheill, 123

Complemento: Dois andar Cep:

Estado: RS, Brasil

Valor:

1X de 18390.85 R\$ (18390.85 R\$)

O cartão possui saldo insuficiente.

5.7. Recusado por problema com a data de vencimento

informações pessoais

Nome

Sobrenome

Thiago

Jansen

Email

thiago.jcsampaio@gmail.com

Informações do Cartão

Nome do titular

Número do cartão

EXPI

 375365153556885

Como está escrito no cartão

Mês da Validade

Ano da Validade

Código de segurança (CVV)

11

2025

1234

Documento

Número do Documento

CPF

10787997927

Parcelas

12 parcelas de R\$ 1.835,87 (R\$ 22.030,44)

Finalizar compra

Status:

Cliente:

Thiago Jansen; CPF : 10787997927

Endereço:

Rua: Rua Engenheiro Elmer Lawsorense Corttheill, 123

Complemento: Dois andar Cep:

Estado: RS, Brasil

Valor:

12X de 1835.87 R\$ (22030.4 R\$)

Confira a data de validade.

5.8. Recusado por erro no formulário

Informações pessoais

Nome

Thiago

Sobrenome

Jansen

Email

thiago.jcsampaio@gmail.com


Informações do Cartão

Nome do titular

FORM

Como está escrito no cartão

Número do cartão

 375365153556885

Mês da Validade

11

Ano da Validade

2030

Código de segurança (CVV)

1241

Documento

CPF

Número do Documento

10787997927

Parcelas

1 parcela de R\$ 18.390,85 (R\$ 18.390,85)

Finalizar compra

Status:

Cliente:
Thiago Jansen; CPF : 10787997927

Endereço:
Rua: Rua Engenheiro Elmer Lawsorensen Corttheill, 123
Complemento: Dois andar Cep:
Estado: RS, Brasil

Valor:
1X de 18390.85 R\$ (18390.85 R\$)

Confira os dados.

6. CONCLUSÃO

A partir deste trabalho, foi possível identificar a importância das API para o desenvolvimento web em geral. Com isso, as integrações se tornam fáceis e padronizadas para sistemas externos.