Pagamentos STRIPE

Configurações da conta

- [ ] Criar uma conta na STRIPE
- [ ] Verificar sua conta
- [ ] Completar cadastro

Código

- [ ] Instalar Django, stripe python-decouple
- [ ] Criar um projeto Django
- [ ] Definir sua PK e SK em um arquivo .env

```
STRIPE_SECRET_KEY=sk_token
    STRIPE_PUBLIC_KEY=pk_token
```

- [ ] Importe as configurações de .env para .settings

```
from decouple import config

    STRIPE_SECRET_KEY = config('STRIPE_SECRET_KEY')
    STRIPE_PUBLIC_KEY = config('STRIPE_PUBLIC_KEY')
```

- [ ] Crie um APP chamado 'produtos' e instale o APP
- [ ] Defina uma URL para produtos
- [ ] Crie uma URL para a função home
- [ ] Crie a função home

```
def home(request):
        return render(request, 'home.html')
```

- [ ] Crie o home.html
- [ ] Crie a model Produto

```
class Produto(models.Model):
        nome = models.CharField(max_length=50)
        preco = models.FloatField()

        def __str__(self) -> str:
            return self.nome

        def exibe_preco(self):
            return "{:.2f}".format(self.preco)
```

- [ ] Faça as migrações
- [ ] Cadastre no admin
- [ ] Crie um superuser
- [ ] Defina sua SK

```
stripe.api_key = settings.STRIPE_SECRET_KEY
```

- [ ] Cria a view que irá criar um pagamento

```
def create_checkout_session(request, id):
        produto = Produto.objects.get(id = id)
        YOUR_DOMAIN = "http://127.0.0.1:8000"
        checkout_session = stripe.checkout.Session.create(
            line_items=[
                {
                    'price_data': {
                    'currency': 'BRL',
                    'unit_amount': int(produto.preco),
                        'product_data': {
                            'name': produto.nome
                        }

                    },
                    'quantity': 1,
                },
            ],
            payment_method_types=[
                'card',
                'boleto',
            ],
            metadata={
                'id_produto': produto.id,
            },
            mode='payment',
            success_url=YOUR_DOMAIN + '/sucesso',
            cancel_url=YOUR_DOMAIN + '/erro',
        )
        return JsonResponse({'id': checkout_session.id})
```

- [ ] Crie a URL para create_checkout_session

```
path('create-checkout-session/<int:id>', views.create_checkout_session, name="create_checkout_session"),
```

- [ ] Crie as URL's e views de sucesso e erro
- [ ] Defina o HTML em home.html

```
<html>
    <head>

        <script src="https://polyfill.io/v3/polyfill.min.js?version=3.52.1&features=fetch"></script>
        <script src="https://js.stripe.com/v3/"></script>
    </head>
    <body>
        <body>
            <section>
              <div class="product">

                <div class="description">
                  <h3>{{produto.nome}}</h3>
                  <h5>{{produto.preco}}</h5>
                </div>
              </div>

                <button type="submit" id="checkout-button">Checkout</button>

            </section>
          </body>
      {% csrf_token %}
    </body>
    <script type="text/javascript">
      const csrftoken = document.querySelector('[name=csrfmiddlewaretoken]').value;
      // Create an instance of the Stripe object with your publishable API key
      var stripe = Stripe("{{ STRIPE_PUBLIC_KEY }}");
      var checkoutButton = document.getElementById("checkout-button");
      checkoutButton.addEventListener("click", function () {
        fetch("{% url 'create_checkout_session' produto.id %}", {
          method: "POST",
          headers: {
              'X-CSRFToken': csrftoken
          }
        })
          .then(function (response) {
            return response.json();
          })
          .then(function (session) {
            return stripe.redirectToCheckout({ sessionId: session.id });
          })
          .then(function (result) {
            // If redirectToCheckout fails due to a browser or network
            // error, you should display the localized error message to your
            // customer using error.message.
            if (result.error) {
              alert(result.error.message);
            }
          })
          .catch(function (error) {
            console.error("Error:", error);
          });
      });
    </script>
    </html>
```

- [ ] Envie para o HTML a PK e o Produto

```
def home(request):
        produto = Produto.objects.get(id = 1)
        return render(request, 'home.html', {'produto': produto, 'STRIPE_PUBLIC_KEY': settings.STRIPE_PUBLIC_KEY})
```

Webhook

- [ ] Instale o CLI STRIPE
- [ ] Descompacte o arquivo

```
tar -xvf stripe_1.7.4_linux_x86_64.tar.gz
```

- [ ] Faça login

```
./stripe login
```

- [ ] Crie uma URL para receber as solicitações via WebHook

```
path('stripe_webhook', views.stripe_webhook, name="stripe_webhook")
```

- [ ] Crie a View 'stripe_webhook'

```
@csrf_exempt
    def stripe_webhook(request):
      payload = request.body

      # For now, you only need to print out the webhook payload so you can see
      # the structure.
      print(payload)

      return HttpResponse(status=200)
```

- [ ] Execute o CLI STRIPE para ouvir as requisições e enviar ao seu end-point

```
stripe listen --forward-to localhost:8000/stripe_webhook
```

- [ ] Veja um erro de segurança

```
curl -X POST \
      -H "Content-Type: application/json" \
      --data '{ type: "checkout.session.completed" }' \
      -is http://localhost:8000/stripe_webhook
```

- [ ] Defina o token do WebHook em .env

```
STRIPE_WEBHOOK_SECRET=seu_token
```

- [ ] Importe para o settings.py

```
STRIPE_WEBHOOK_SECRET = config('STRIPE_WEBHOOK_SECRET')
```

- [ ] Verifique se a solicitação veio da sua conta da STRIPE

```
@csrf_exempt
    def stripe_webhook(request):
        payload = request.body
        sig_header = request.META['HTTP_STRIPE_SIGNATURE']
        event = None
        endpoint_secret = settings.STRIPE_WEBHOOK_SECRET

        try:
            event = stripe.Webhook.construct_event(
            payload, sig_header, endpoint_secret
            )
        except ValueError as e:
            # Invalid payload
            return HttpResponse(status=400)
        except stripe.error.SignatureVerificationError as e:
            # Invalid signature
            return HttpResponse(status=400)
        print(payload)

        return HttpResponse(status=200)
```

- [ ] Verifique se o pagamento foi aprovado

```
if event['type'] == 'checkout.session.completed':
            session = event['data']['object']

            print('Aprovada')
```