

---

# **django-pagseguro2 Documentation**

***Versão 1.0.10***

**Allisson Azevedo**

06/05/2015



<b>1</b>	<b>Conteúdo</b>	<b>3</b>
1.1	Tutorial . . . . .	3
<b>2</b>	<b>Autor</b>	<b>7</b>
<b>3</b>	<b>Principais características</b>	<b>9</b>
<b>4</b>	<b>Links</b>	<b>11</b>
<b>5</b>	<b>Referências</b>	<b>13</b>



Integração da API v2 do PagSeguro com o Django. Para API v1 do PagSeguro, use o [django-pagseguro](#).



---

## Conteúdo

---

### 1.1 Tutorial

O django-pagseguro2 necessita do Django versão 1.3+, lembrando que o suporte ao python3 está presente apenas nas versões 1.5+.

#### 1.1.1 Instalação

Instale o django-pagseguro2:

```
pip install django-pagseguro2
```

#### 1.1.2 Configuração

Adicione o app pagseguro no INSTALLED\_APPS:

```
INSTALLED_APPS = (  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.staticfiles',  
    'django.contrib.admin',  
    'pagseguro',  
)
```

O django-pagseguro2 suporta as migrações do django 1.7, caso você esteja usando o South, adicione a seguinte configuração no seu settings.py:

```
SOUTH_MIGRATION_MODULES = {  
    'pagseguro': 'pagseguro.south_migrations',  
}
```

Adicione as configurações no settings.py:

```
PAGSEGURO_EMAIL = 'fulano@cicrano.com'  
PAGSEGURO_TOKEN = 'token'  
PAGSEGURO_SANDBOX = True # se o valor for True, as requisições a api serão feitas usando o PagSeguro  
PAGSEGURO_LOG_IN_MODEL = True # se o valor for True, os checkouts e transações vão ser logadas no database
```

Adicione a view que recebe as notificações no urls.py:

```
urlpatterns = patterns(
    '',
    url(r'^retorno/pagseguro/', include('pagseguro.urls')),
    # a url de retorno será /retorno/pagseguro/
)
```

### 1.1.3 Trabalhando com a API de checkout

Recomendo que você abra um shell do Django para realizar os testes:

```
python manage.py shell
```

Para começar, precisamos importar duas classes, PagSeguroItem e PagSeguroApi:

```
>>> from pagseguro.api import PagSeguroItem, PagSeguroApi
```

O PagSeguroItem representa um item que será adicionado a transação:

```
>>> item1 = PagSeguroItem(id='0001', description='Meu item 0001', amount='100.00', quantity=1)
>>> item1
<PagSeguroItem: Meu item 0001>
>>> item1.id
u'0001'
>>> item1.amount
Decimal('100.00')
>>> item1.quantity
1
>>> print(item1.shipping_cost)
None
>>> print(item1.weight)
None
```

Você pode adicionar informações como custo de envio e peso:

```
>>> item2 = PagSeguroItem(id='0002', description='Meu item 0002', amount='150.00', quantity=1, shipping_cost=25, weight=500)
>>> item2
<PagSeguroItem: Meu item 0002>
>>> item2.id
u'0002'
>>> item2.amount
Decimal('150.00')
>>> item2.quantity
1
>>> item2.shipping_cost
Decimal('25.00')
>>> item2.weight
500
```

Agora que temos os itens, podemos fazer o checkout para obter o código da transação:

```
>>> pagseguro_api = PagSeguroApi(reference='id-unico-de-referencia-do-seu-sistema')
>>> # voce poderia passar valores iniciais, ex: pagseguro_api = PagSeguroApi(email='meu@email.com', token='meu-token')
>>> # voce pode passar qualquer valor inicial que a documentacao do PagSeguro informa, exceto os itens
>>> # o email e token são carregados automaticamente pelas variáveis do settings.
>>> # adicionando itens
>>> pagseguro_api.add_item(item1)
>>> pagseguro_api.add_item(item2)
>>> # verificando os itens
```



```
>>> pagseguro_api.get_items()
[<PagSeguroItem: Meu item 0001>, <PagSeguroItem: Meu item 0002>]
>>> # removendo os itens
>>> pagseguro_api.clear_items()
>>> pagseguro_api.get_items()
[]
>>> # fazendo um checkout
>>> pagseguro_api.add_item(item1)
>>> pagseguro_api.add_item(item2)
>>> data = pagseguro_api.checkout()
>>> data
{'date': datetime.datetime(2014, 6, 7, 15, 19, 48, tzinfo=tzoffset(None, -10800)), 'status_code': 200}
>>> # agora basta redirecionar o cliente para o data['redirect_url']
>>> data['redirect_url']
'https://sandbox.pagseguro.uol.com.br/v2/checkout/payment.html?code=D0C5A7F8E5E53268849D4F89DA3363E0'
```

Você pode consultar os dados de uma transação:

```
>>> pagseguro_api = PagSeguroApi()
>>> data = pagseguro_api.get_transaction('437D1B99-A6E8-46F0-8C00-47B818615AA2')
>>> data['success']
True
>>> data['transaction']
OrderedDict([(u'date', u'2014-06-07T15:25:36.000-03:00'), (u'code', u'437D1B99-A6E8-46F0-8C00-47B818615AA2')])
>>> data['transaction']['code']
u'437D1B99-A6E8-46F0-8C00-47B818615AA2'
```

Passando parâmetros extras na inicialização do PagSeguroApi:

```
>>> from pagseguro.api import PagSeguroApi
>>> from decimal import Decimal
>>> extra_amount = Decimal('20.00')
>>> sender_email = 'user@email.com'
>>> sender_name = 'Fulano da Silva'
>>> sender_area_code = 83
>>> sender_phone = 11111111
>>> pagseguro_api = PagSeguroApi(reference='id-unico-de-referencia-do-seu-sistema', extraAmount=extraAmount)
```

Você pode passar qualquer parâmetro http, exceto os relativos aos itens. [Referência](#).

## 1.1.4 Trabalhando com Signals de checkout

Podemos usar o recurso de Signals do Django para capturar informações relacionadas aos checkouts.

Isso é bastante útil para detectar possíveis problemas na implementação.

Temos os seguintes Signals disponíveis para checkouts:

- **checkout\_realizado** (Disparado sempre que um novo checkout for feito).
- **checkout\_realizado\_com\_sucesso**
- **checkout\_realizado\_com\_erro**

Para capturar o Signal **checkout\_realizado**:

```
>>> from pagseguro.signals import checkout_realizado
>>> def load_signal(sender, data, **kwargs):
...     print(data['success'])
```

```
...
>>> checkout_realizado.connect(load_signal)
```

### 1.1.5 Trabalhando com Signals de notificação

Após a transação ser concluída pelo cliente, o PagSeguro vai enviar uma requisição do tipo POST para uma url que você escolheu previamente sempre que alguma mudança ocorrer no status.

Para ambiente de testes, eu recomendo que você utilize o [PagSeguro Sandbox](#) em conjunto com o serviço [Runscope](#) para conseguir visualizar as notificações.

Quando o PagSeguro envia uma nova notificação, Signals são disparados contendo as informações da transação.

Para cada tipo de status, existe um Signal específico, se você quiser ser notificado apenas quando a compra for paga, você deve capturar o Signal **notificacao\_status\_pago**.

Temos os seguintes Signals disponíveis para notificações:

- **notificacao\_recebida** (Disparado sempre que uma notificação for recebida).
- **notificacao\_status\_aguardando**
- **notificacao\_status\_em\_analise**
- **notificacao\_status\_pago**
- **notificacao\_status\_disponivel**
- **notificacao\_status\_em\_disputa**
- **notificacao\_status\_devolvido**
- **notificacao\_status\_cancelado**

Para capturar o Signal **notificacao\_recebida**:

```
>>> from pagseguro.signals import notificacao_recebida
>>> def load_signal(sender, transaction, **kwargs):
...     print(transaction['status'])
...
>>> notificacao_recebida.connect(load_signal)
```

Exemplo de um objeto **transaction**:

```
>>> transaction
OrderedDict([(u'date', u'2014-06-07T15:25:36.000-03:00'), (u'code', u'437D1B99-A6E8-46F0-8C00-47B818615AA2')])
>>> transaction.keys()
[u'date', u'code', u'type', u'status', u'lastEventDate', u'paymentMethod', u'grossAmount', u'discount']
>>> transaction['status']
u'3'
>>> transaction['code']
u'437D1B99-A6E8-46F0-8C00-47B818615AA2'
```

### 1.1.6 Logando checkouts e transações no database

Sempre que você configura o `PAGSEGURO_LOG_IN_MODEL = True`, todos os checkouts e transações são logados no database.

Basta acessar o `/admin/` e verificar.

---

**Autor**

---

- Allisson Azevedo



---

## Principais características

---

- Python: 2.6, 2.7, 3.2, 3.3 e 3.4.
- Django: 1.4 (apenas python2), 1.5, 1.6, 1.7, 1.8.
- Processamento de requisições http/https usando a biblioteca [requests](#).
- Excelente cobertura de testes (> 80%).
- Documentação com exemplos práticos.



---

### Links

---

- [Github](#)
- [Travis CI](#)
- [Coveralls](#)





---

## Referências

---

- [django-pagseguro](#) (Para API v1 do PagSeguro)
- [PagSeguro: Api de Pagamento](#)
- [PagSeguro: Api de Notificação](#)
- [PagSeguro: Sandbox](#)