
zoop-wrapper

jan. 21, 2021

| | | |
|----------|--|-----------|
| 1 | Instalando | 3 |
| 2 | Configuração | 5 |
| 3 | Documentação da Zoop | 7 |
| 4 | Recursos disponíveis | 9 |
| 4.1 | Entrada de dados | 10 |
| 4.2 | Separação de atributos | 11 |
| 4.3 | Eventos de Transação | 12 |
| 4.4 | Transação de Cartão de crédito | 13 |
| 4.5 | Exemplos de conta bancária | 13 |
| 4.6 | Exemplos de comprador | 14 |
| 4.7 | Exemplos de cartão | 17 |
| 4.8 | Exemplos de vendedor | 18 |
| 4.9 | Exemplos de transação | 23 |
| 4.10 | Exemplos de webhook | 27 |
| 4.11 | constants | 29 |
| 4.12 | exceptions | 29 |
| 4.13 | utils | 30 |
| 4.14 | response | 30 |
| 4.15 | wrapper | 31 |
| 4.16 | models | 43 |
| | Índice de Módulos Python | 69 |
| | Índice | 71 |

Cliente não oficial da Zoop feito em Python, para realizar integração com o gateway de pagamento.

[Documentação oficial da Zoop](#)

CAPÍTULO 1

Instalando

Nosso pacote está hospedado no [PyPI](#)

```
pip install zoop-wrapper
```


CAPÍTULO 2

Configuração

Para utilizar o *zoop-wrapper* é necessário ter duas constantes/variáveis. sendo elas:

```
ZOOP_KEY='chave de autenticação recebida da zoop'  
MARKETPLACE_ID='ID do market place'
```

Recomendamos criar um arquivo *.env* contendo essas variáveis de ambiente.

Podem ser criadas diretamente no terminal utilizando (não recomendado):

```
export ZOOP_KEY='chave de autenticação recebida da zoop'  
export MARKETPLACE_ID='ID do market place'
```

Podem ser criadas também diretamente no *arquivo.py*

Perigo: Fazer isso além de não ser recomendado é uma **FALHA** de segurança.

CAPÍTULO 3

Documentação da Zoop

A Zoop fornece diversas formas de comunicação. Sendo uma telas API's baseadas na tecnologia REST. A documentação da API da zoop não é uma das melhores, mas está disponível abertamente.

Aviso: Não temos conhecimento se TODOS os testes podem ser realizados sem ônus ao desenvolvedor.
As transações de cartão podem ser extornadas e não há problema em gerar boletos (não paga a baixa).

Saiba mais na [documentação oficial da Zoop](#)

Recursos disponíveis

Market Place

- detalhes

Webhooks

- Cadastro
- listagem
- detalhes
- remoção

Buyer

- Atualização
- Cadastro
- listagem
- detalhes
- remoção

Seller

- Atualização
- Cadastro
- listagem
- detalhes
- remoção

Token

- Cadastro de token cartão de crédito
- Cadastro de token conta bancária

- detalhes

Cartão de crédito

- Conexão
- detalhes
- remoção

Conta bancária

- Atualização
- Conexão
- listagem
- detalhes
- remoção

Boleto

- detalhes

Transação

- listagem
- detalhes
- cancelamento
- Cadastro transação boleto
- Cadastro transação cartão de crédito

4.1 Entrada de dados

As entradas de dados podem ser feitas utilizando os `models` Python declarados na `lib` ou dicionários Python.

Nos exemplos documentados utilizaremos os `models` declarados, mas pode ser utilizados `dict`'s ao invés.

Por exemplo na criação de uma transação de cartão presente poderia ter sido utilizado o seguinte dicionário:

```
t = {
    "customer": seller_brian,
    "description": "Uma descrição breve da motivação da sua transação",
    "on_behalf_of": seller_denise,
    "payment_type": "credit",
    "source": {
        "amount": "1234",
        "card": {
            "card_number": Faker("credit_card_number").generate(),
            "expiration_month": "05",
            "expiration_year": "2030",
            "holder_name": "foo",
            "security_code": 123,
        },
        "usage": "single_use",
    },
}
```

4.2 Separação de atributos

Nos nossos `models` são declarados todos os atributos que a zoop retorna e que podem ser úteis.

Porém nem todos os atributos que a zoop retorna devem ser enviados no método de criação.

Aviso: Essa lib não faz a validação de um input que não deveria ser enviado!

Existe apenas a distinção de atributos *required* e *non_required*.

4.2.1 Atributos obrigatórios

Geralmente os atributos *required* são todos os atributos que a zoop precisa receber na criação.

Nota: Existem alguns atributos opcionais que estão como *required*. Como por exemplo *Transaction.description*

4.2.2 Atributos opcionais

Já os atributos *non_required* são os que PODEM ser enviados na criação e os atributos que NÃO podem ser enviados na criação.

4.2.3 Atributos de leitura

Não existe essa classificação na lib.

São os atributos que NÃO podem ser enviados na criação. Esses atributos são gerados pela própria zoop e retornados pela API deles.

Aviso: Atualmente esses atributos estão na lista de *non_required*!

4.2.4 Exemplo de cartão

Na criação de cartão enviamos:

```
card_token = Token(  
    card_number=Faker("credit_card_number").generate(),  
    expiration_month="05",  
    expiration_year="2030",  
    holder_name="foo",  
    security_code=123,  
)
```

e isso nos retorna:

```
{
  "id": "4abf4010cc93414ca585463fdc7b44d6",
  "resource": "card",
  "description": null,
  "card_brand": "Visa",
  "first4_digits": "4821",
  "last4_digits": "9566",
  "expiration_month": "5",
  "expiration_year": "2030",
  "holder_name": "foo",
  "is_active": true,
  "is_valid": true,
  "is_verified": false,
  "customer": "0e084bb6a60f47e8ac45949d5040eb92",
  "fingerprint": "e793b5f0ee2362d01d9879d40d99dd9df401c36d735df6b0d34807fcc42c1e6d",
  "address": null,
  "verification_checklist": {
    "postal_code_check": "unchecked",
    "security_code_check": "fail",
    "address_line1_check": "unchecked"
  },
  "metadata": {},
  "uri": "/v1/marketplaces/foo/cards/4abf4010cc93414ca585463fdc7b44d6",
  "created_at": "2020-05-22T15:07:34+00:00",
  "updated_at": "2020-05-22T15:07:35+00:00"
}
```

4.3 Eventos de Transação

No mundo de transações, existem alguns eventos. Dentro deles temos por exemplo:

```
- success
- failed
- canceled
- reversed
- charged_back
- pre_authorized
- captured
- voided
```

4.3.1 Charge Back

Processo no qual o gateway de pagamento cobra a um vendedor o valor da compra pago pelo comprador. Retornando o dinheiro ao comprador.

4.3.2 Void

Estorno de transação.

4.3.3 Reverse

A transação foi revertida. Isso acontece devido à alguma falha técnica em algum momento do fluxo. Com isso todo o fluxo feito é revertido.

4.4 Transação de Cartão de crédito

No mundo de transações de cartão de crédito, existem alguns eventos. Dentro deles temos por exemplo:

```
- pre autorização
- captura
```

4.4.1 Pré autorização

A pré autorização realiza o bloqueio do valor no cartão de crédito do comprador.

Importante: Mas isso não realiza o pagamento, vulgo débito do valor!

4.4.2 Captura

A captura realiza o pagamento propriamente dito do valor passado!

Importante: O valor passado pode ser menor ou igual ao valor pré autorizado. Nesse cenário a diferença é desbloqueada automaticamente para o comprador.

Aviso: Se o valor desejado a ser capturado for maior do que o valor pré autorizado, a transação deverá ser estornada e feita novamente.

4.5 Exemplos de conta bancária

4.5.1 Criar conta bancária

4.5.2 Lista as contas bancárias de um vendedor

```
import os

from zoop_wrapper import ZoopWrapper
from examples.utils import dump_response

"""
Nesse momento as constantes podem ser criadas no arquivo .py.
Mas é recomendado utilizar como variável de ambiente em um '.env'
"""
from zoop_wrapper.constants import MARKETPLACE_ID, ZOOP_KEY
```

(continues on next page)

```
client = ZoopWrapper(marketplace_id=MARKETPLACE_ID, key=ZOOP_KEY)

seller_brian = "0b05a360f4b749498f74e13004c08024"

response = client.list_bank_accounts_by_seller(seller_brian)

dump_response(response, os.path.basename(__file__).split(".")[0])
```

4.5.3 Remover a conta bancária de um vendedor

4.6 Exemplos de comprador

4.6.1 Criar comprador

```
import os

from factory.faker import Faker
from pycpfcnpj import gen

from zoop_wrapper import ZoopWrapper, Buyer, Address
from examples.utils import dump_response

"""
Nesse momento as constantes podem ser criadas no arquivo .py.
Mas é recomendado utilizar como variável de ambiente em um '.env'
"""

from zoop_wrapper.constants import MARKETPLACE_ID, ZOOP_KEY

client = ZoopWrapper(marketplace_id=MARKETPLACE_ID, key=ZOOP_KEY)

cpf_ou_cnpj = Faker("random_element", elements=[gen.cpf(), gen.cnpj()]).generate()

b = Buyer(
    address=Address(
        city="Natal",
        country_code="BR",
        line1="foo",
        line2="123",
        line3="barbar",
        neighborhood="fooofoo",
        postal_code="59100000",
        state="RN",
    ),
    birthdate="1994-12-27",
    email="foo@bar.com",
    first_name="foo",
    last_name="foo",
    phone_number="+55 84 99999-9999",
    taxpayer_id=cpf_ou_cnpj,
)
```

(continues on next page)

(continuação da página anterior)

```
response = client.add_buyer(b)

dump_response(response, os.path.basename(__file__).split(".")[0])
```

4.6.2 Listar compradores

```
import os

from zoop_wrapper import ZoopWrapper
from examples.utils import dump_response

"""
Nesse momento as constantes podem ser criadas no arquivo .py.
Mas é recomendado utilizar como variável de ambiente em um '.env'
"""

from zoop_wrapper.constants import MARKETPLACE_ID, ZOOB_KEY

client = ZoopWrapper(marketplace_id=MARKETPLACE_ID, key=ZOOB_KEY)

response = client.list_buyers()

dump_response(response, os.path.basename(__file__).split(".")[0])
```

4.6.3 Pegar comprador

```
import os

from zoop_wrapper import ZoopWrapper
from examples.utils import dump_response

"""
Nesse momento as constantes podem ser criadas no arquivo .py.
Mas é recomendado utilizar como variável de ambiente em um '.env'
"""

from zoop_wrapper.constants import MARKETPLACE_ID, ZOOB_KEY

client = ZoopWrapper(marketplace_id=MARKETPLACE_ID, key=ZOOB_KEY)

buyer_id = "ffe4b7a1f19c4a9da85b6d72c0b6201c"

response = client.retrieve_buyer(buyer_id)

dump_response(response, os.path.basename(__file__).split(".")[0])
```

4.6.4 Atualizar comprador

```
import os

from zoop_wrapper import ZoopWrapper, Buyer, Address
from examples.utils import dump_response

"""
Nesse momento as constantes podem ser criadas no arquivo .py.
Mas é recomendado utilizar como variável de ambiente em um '.env'
"""
from zoop_wrapper.constants import MARKETPLACE_ID, ZOOP_KEY

client = ZoopWrapper(marketplace_id=MARKETPLACE_ID, key=ZOOP_KEY)

b = Buyer(
    address=Address(
        city="Natal",
        country_code="BR",
        line1="foo",
        line2="123",
        line3="barbar",
        neighborhood="fooofoo",
        postal_code="59150000",
        state="RN",
    ),
    birthdate="1994-12-27",
    email="foo@bar.com",
    first_name="foo",
    last_name="foo",
    phone_number="+55 84 99999-9999",
    taxpayer_id="19249382944",
)

response = client.update_buyer("ffe4b7a1f19c4a9da85b6d72c0b6201c", b)

dump_response(response, os.path.basename(__file__).split(".")[0])
```

4.6.5 Remover comprador

```
import os

from zoop_wrapper import ZoopWrapper
from examples.utils import dump_response

"""
Nesse momento as constantes podem ser criadas no arquivo .py.
Mas é recomendado utilizar como variável de ambiente em um '.env'
"""
from zoop_wrapper.constants import MARKETPLACE_ID, ZOOP_KEY

client = ZoopWrapper(marketplace_id=MARKETPLACE_ID, key=ZOOP_KEY)
```

(continues on next page)

(continuação da página anterior)

```
response = client.remove_buyer("f85c8b84749c431ab0db044812ca7a57")
dump_response(response, os.path.basename(__file__).split(".")[0])
```

4.6.6 Buscar comprador

```
import os

from zoop_wrapper import ZoopWrapper
from examples.utils import dump_response

"""
Nesse momento as constantes podem ser criadas no arquivo .py.
Mas é recomendado utilizar como variável de ambiente em um '.env'
"""

from zoop_wrapper.constants import MARKETPLACE_ID, ZOOP_KEY

client = ZoopWrapper(marketplace_id=MARKETPLACE_ID, key=ZOOP_KEY)

cpf_ou_cnpj = "19249382944"

response = client.search_buyer(cpf_ou_cnpj)

dump_response(response, os.path.basename(__file__).split(".")[0])
```

4.7 Exemplos de cartão

4.7.1 Criar cartão

```
import os

from factory.faker import Faker

from zoop_wrapper import ZoopWrapper, Token
from examples.utils import dump_response

"""
Nesse momento as constantes podem ser criadas no arquivo .py.
Mas é recomendado utilizar como variável de ambiente em um '.env'
"""

from zoop_wrapper.constants import MARKETPLACE_ID, ZOOP_KEY

client = ZoopWrapper(marketplace_id=MARKETPLACE_ID, key=ZOOP_KEY)

card_token = Token(
    card_number=Faker("credit_card_number").generate(),
    expiration_month="05",
    expiration_year="2030",
    holder_name="foo",
    security_code=123,
```

(continues on next page)

```
)

from examples.seller.retrieve_seller import seller_id # noqa

customer_id = seller_id

response = client.add_card(card_token, customer_id)

dump_response(response, os.path.basename(__file__).split(".")[0])
```

4.7.2 Pegar cartão

```
import os

from zoop_wrapper import ZoopWrapper
from examples.utils import dump_response

"""
Nesse momento as constantes podem ser criadas no arquivo .py.
Mas é recomendado utilizar como variável de ambiente em um '.env'
"""

from zoop_wrapper.constants import MARKETPLACE_ID, ZOOP_KEY

client = ZoopWrapper(marketplace_id=MARKETPLACE_ID, key=ZOOP_KEY)

card_id = "4abf4010cc93414ca585463fdc7b44d6"

response = client.retrieve_card(card_id)

dump_response(response, os.path.basename(__file__).split(".")[0])
```

4.8 Exemplos de vendedor

4.8.1 Criar vendedor empresa

```
import os

from pycpfcnpj import gen

from zoop_wrapper import ZoopWrapper, Seller, Address
from examples.utils import dump_response

"""
Nesse momento as constantes podem ser criadas no arquivo .py.
Mas é recomendado utilizar como variável de ambiente em um '.env'
"""

from zoop_wrapper.constants import MARKETPLACE_ID, ZOOP_KEY
```

(continues on next page)

(continuação da página anterior)

```

client = ZoopWrapper(marketplace_id=MARKETPLACE_ID, key=ZOOB_KEY)

s = Seller(
    business_address=Address(
        city="Natal",
        country_code="BR",
        line1="foo",
        line2="123",
        line3="barbar",
        neighborhood="foofoo",
        postal_code="59100000",
        state="RN",
    ),
    business_email="foo",
    business_name="foo",
    business_opening_date="foo",
    business_phone="foo",
    business_website="foo",
    ein=gen.cnpj(),
)

response = client.add_seller(s)

dump_response(response, os.path.basename(__file__).split(".")[0])

```

4.8.2 Criar vendedor PF

```

import os

from pycpfcnpj import gen

from zoop_wrapper import ZoopWrapper, Seller, Address
from examples.utils import dump_response

"""
Nesse momento as constantes podem ser criadas no arquivo .py.
Mas é recomendado utilizar como variável de ambiente em um '.env'
"""

from zoop_wrapper.constants import MARKETPLACE_ID, ZOOB_KEY

client = ZoopWrapper(marketplace_id=MARKETPLACE_ID, key=ZOOB_KEY)

s = Seller(
    address=Address(
        city="Natal",
        country_code="BR",
        line1="foo",
        line2="123",
        line3="barbar",
        neighborhood="foofoo",
        postal_code="59100000",
        state="RN",
    ),
)

```

(continues on next page)

```
    birthdate="1994-12-27",
    email="foo@bar.com",
    first_name="foo",
    last_name="foo",
    phone_number="+55 84 99999-9999",
    taxpayer_id=gen.cpf(),
)

response = client.add_seller(s)

dump_response(response, os.path.basename(__file__).split(".")[0])
```

4.8.3 Listar vendedores

```
import os

from zoop_wrapper import ZoopWrapper
from examples.utils import dump_response

"""
Nesse momento as constantes podem ser criadas no arquivo .py.
Mas é recomendado utilizar como variável de ambiente em um '.env'
"""

from zoop_wrapper.constants import MARKETPLACE_ID, ZOOP_KEY

client = ZoopWrapper(marketplace_id=MARKETPLACE_ID, key=ZOOP_KEY)

response = client.list_sellers()

dump_response(response, os.path.basename(__file__).split(".")[0])
```

4.8.4 Pegar vendedor

```
import os

from zoop_wrapper import ZoopWrapper
from examples.utils import dump_response

"""
Nesse momento as constantes podem ser criadas no arquivo .py.
Mas é recomendado utilizar como variável de ambiente em um '.env'
"""

from zoop_wrapper.constants import MARKETPLACE_ID, ZOOP_KEY

client = ZoopWrapper(marketplace_id=MARKETPLACE_ID, key=ZOOP_KEY)

seller_id = "0e084bb6a60f47e8ac45949d5040eb92"

response = client.retrieve_seller(seller_id)
```

(continues on next page)

(continuação da página anterior)

```
dump_response(response, os.path.basename(__file__).split(".")[0])
```

4.8.5 Atualizar vendedor

```
import os

from zoop_wrapper import ZoopWrapper, Seller, Address
from examples.utils import dump_response

"""
Nesse momento as constantes podem ser criadas no arquivo .py.
Mas é recomendado utilizar como variável de ambiente em um '.env'
"""

from zoop_wrapper.constants import MARKETPLACE_ID, ZOOP_KEY

client = ZoopWrapper(marketplace_id=MARKETPLACE_ID, key=ZOOP_KEY)

s = Seller(
    address=Address(
        city="Natal",
        country_code="BR",
        line1="foo",
        line2="123",
        line3="barbar",
        neighborhood="foofoo",
        postal_code="59100000",
        state="RN",
    ),
    birthdate="1994-12-27",
    email="foo@bar.com",
    first_name="foo",
    last_name="bar 2",
    phone_number="+55 84 99999-9999",
    taxpayer_id="13543402480",
)

seller_id = "0e084bb6a60f47e8ac45949d5040eb92"

response = client.update_seller(seller_id, s)

dump_response(response, os.path.basename(__file__).split(".")[0])
```

4.8.6 Remover vendedor

```
import os

from zoop_wrapper import ZoopWrapper
from examples.utils import dump_response

"""
```

(continues on next page)

```
Nesse momento as constantes podem ser criadas no arquivo .py.  
Mas é recomendado utilizar como variável de ambiente em um '.env'  
"""  
from zoop_wrapper.constants import MARKETPLACE_ID, ZOOP_KEY  
  
client = ZoopWrapper(marketplace_id=MARKETPLACE_ID, key=ZOOP_KEY)  
  
seller_id = "79bb6be6ac8a477cbc1d8f79236a0f75"  
  
response = client.remove_seller(seller_id)  
  
dump_response(response, os.path.basename(__file__).split(".")[0])
```

4.8.7 Buscar vendedor empresa

```
import os  
  
from zoop_wrapper import ZoopWrapper  
from examples.utils import dump_response  
  
"""  
Nesse momento as constantes podem ser criadas no arquivo .py.  
Mas é recomendado utilizar como variável de ambiente em um '.env'  
"""  
from zoop_wrapper.constants import MARKETPLACE_ID, ZOOP_KEY  
  
client = ZoopWrapper(marketplace_id=MARKETPLACE_ID, key=ZOOP_KEY)  
  
cnpj = "44431904079819"  
  
response = client.search_business_seller(cnpj)  
  
dump_response(response, os.path.basename(__file__).split(".")[0])
```

4.8.8 Buscar vendedor PF

```
import os  
  
from zoop_wrapper import ZoopWrapper  
from examples.utils import dump_response  
  
"""  
Nesse momento as constantes podem ser criadas no arquivo .py.  
Mas é recomendado utilizar como variável de ambiente em um '.env'  
"""  
from zoop_wrapper.constants import MARKETPLACE_ID, ZOOP_KEY  
  
client = ZoopWrapper(marketplace_id=MARKETPLACE_ID, key=ZOOP_KEY)
```

(continues on next page)

(continuação da página anterior)

```

cpf = "13543402480"

response = client.search_individual_seller(cpf)

dump_response(response, os.path.basename(__file__).split(".")[0])

```

4.9 Exemplos de transação

4.9.1 Criar transação de cartão de crédito não presente

4.9.2 Criar transação de cartão de crédito presente

```

import os

from factory.faker import Faker

from zoop_wrapper import ZoopWrapper, Transaction, Source, Token
from examples.utils import dump_response

"""
Nesse momento as constantes podem ser criadas no arquivo .py.
Mas é recomendado utilizar como variável de ambiente em um '.env'
"""

from zoop_wrapper.constants import MARKETPLACE_ID, ZOOP_KEY

client = ZoopWrapper(marketplace_id=MARKETPLACE_ID, key=ZOOP_KEY)

seller_brian = "0b05a360f4b749498f74e13004c08024"
seller_denise = "25037b2978b14e7fa5b902d9322e8426"

# Essa flag indica se a transação deve ser apenas pré autorizada ou capturada
capture_flag = True

# Equivalente à R$543,21
quantia_em_centavos = "54321"

t = Transaction(
    customer=seller_brian,
    description="Uma descrição breve da motivação da sua transação",
    on_behalf_of=seller_denise,
    payment_type="credit",
    capture=capture_flag,
    source=Source(
        amount=quantia_em_centavos,
        card=Token(
            card_number=Faker("credit_card_number").generate(),
            expiration_month="05",
            expiration_year="2030",
            holder_name="foo",
            security_code=123,
        ),
        usage="single_use",
    ),
)

```

(continues on next page)

```

    ),
)

response = client.add_transaction(t)

dump_response(response, os.path.basename(__file__).split(".")[0])

```

4.9.3 Criar transação de boleto

```

import os

from zoop_wrapper import (
    Fine,
    Interest,
    Discount,
    BillingInstructions,
    Invoice,
    Transaction,
    ZoopWrapper,
)

from examples.utils import dump_response

"""
Nesse momento as constantes podem ser criadas no arquivo .py.
Mas é recomendado utilizar como variável de ambiente em um '.env'
"""

from zoop_wrapper.constants import MARKETPLACE_ID, ZOOP_KEY
from examples.seller.retrieve_seller import seller_id
from examples.buyer.retrieve_buyer import buyer_id

client = ZoopWrapper(marketplace_id=MARKETPLACE_ID, key=ZOOP_KEY)

# seller_id = "3b94dc92dbad422ea49d44364f3b7b4b"
buyer_or_seller_id = buyer_id

quantia_em_centavos = "3000"
vencimento = "2020-11-20"
pre_vencimento = "2020-11-10"
limite = "2020-11-30"

t = Transaction(
    amount=quantia_em_centavos,
    customer=buyer_or_seller_id,
    description="meu boleto gerado para teste",
    on_behalf_of=seller_id,
    capture=True,
    payment_type="boleto",
    payment_method=Invoice(
        expiration_date=vencimento,
        payment_limit_date=limite,
        billing_instructions=BillingInstructions(
            late_fee=Fine(

```

(continues on next page)

(continuação da página anterior)

```

        mode=Fine.PERCENTAGE,
        percentage=2,
    ),
    interest=Interest(
        mode=Interest.MONTHLY_PERCENTAGE,
        percentage=1,
    ),
    discount=[
        Discount(
            amount=200,
            limit_date=pre_vencimento,
            mode=Discount.FIXED,
        ),
    ],
),
),
)

# _data = {
#     'amount': '1000',
#     'currency': 'BRL',
#     'description': 'meu boleto gerado para teste',
#     'on_behalf_of': seller_id,
#     'customer': buyer_or_seller_id,
#     'payment_type': 'boleto',
#     'payment_method': {
#         'expiration_date': '2020-06-20',
#         'payment_limit_date': '2020-06-30',
#         'billing_instructions': {
#             'late_fee': {
#                 'mode': BillingConfiguration.PERCENTAGE_MODE,
#                 'percentage': 30,
#                 'start_date': '2020-06-20'
#             },
#             'interest': {
#                 'mode': BillingConfiguration.MONTHLY_PERCENTAGE_MODE,
#                 'percentage': 30,
#                 'start_date': '2020-06-20'
#             },
#             'discount': [{
#                 'mode': BillingConfiguration.FIXED_MODE,
#                 'amount': 300,
#                 'limit_date': '2020-06-20'
#             }]
#         }
#     }
# }

response = client.add_transaction(t)

dump_response(response, os.path.basename(__file__).split(".")[0])

```

4.9.4 Extornar transação de cartão

```
import os

from zoop_wrapper import ZoopWrapper
from examples.utils import dump_response

"""
Nesse momento as constantes podem ser criadas no arquivo .py.
Mas é recomendado utilizar como variável de ambiente em um '.env'
"""
from zoop_wrapper.constants import MARKETPLACE_ID, ZOOP_KEY

client = ZoopWrapper(marketplace_id=MARKETPLACE_ID, key=ZOOP_KEY)

card_transaction_id = "b8d82a6296b346f58fa02bd47b14c095"
response = client.cancel_transaction(card_transaction_id)

dump_response(response, os.path.basename(__file__).split(".")[0])
```

4.9.5 Listar transações

```
import os

from zoop_wrapper import ZoopWrapper
from examples.utils import dump_response

"""
Nesse momento as constantes podem ser criadas no arquivo .py.
Mas é recomendado utilizar como variável de ambiente em um '.env'
"""
from zoop_wrapper.constants import MARKETPLACE_ID, ZOOP_KEY

client = ZoopWrapper(marketplace_id=MARKETPLACE_ID, key=ZOOP_KEY)

response = client.list_transactions()

dump_response(response, os.path.basename(__file__).split(".")[0])
```

4.9.6 Listar transações de um seller

```
import os

from zoop_wrapper import ZoopWrapper
from examples.utils import dump_response

"""
Nesse momento as constantes podem ser criadas no arquivo .py.
Mas é recomendado utilizar como variável de ambiente em um '.env'
"""
from zoop_wrapper.constants import MARKETPLACE_ID, ZOOP_KEY
```

(continues on next page)

(continuação da página anterior)

```

client = ZoopWrapper(marketplace_id=MARKETPLACE_ID, key=ZOOB_KEY)

response = client.list_transactions_for_seller("27e17b778b404a83bf8e25ec995e2ffe")

dump_response(response, os.path.basename(__file__).split(".")[0])

```

4.9.7 Pegar detalhes da transação

```

import os

from zoop_wrapper import ZoopWrapper
from examples.utils import dump_response

"""
Nesse momento as constantes podem ser criadas no arquivo .py.
Mas é recomendado utilizar como variável de ambiente em um '.env'
"""

from zoop_wrapper.constants import MARKETPLACE_ID, ZOOB_KEY

client = ZoopWrapper(marketplace_id=MARKETPLACE_ID, key=ZOOB_KEY)

response = client.retrieve_transaction("e8405efa7eaa49ccbca49ec667685dd3")

dump_response(response, os.path.basename(__file__).split(".")[0])

```

4.10 Exemplos de webhook

4.10.1 Criar webhook

```

import os

from zoop_wrapper import ZoopWrapper, Webhook
from examples.utils import dump_response

"""
Nesse momento as constantes podem ser criadas no arquivo .py.
Mas é recomendado utilizar como variável de ambiente em um '.env'
"""

from zoop_wrapper.constants import MARKETPLACE_ID, ZOOB_KEY

client = ZoopWrapper(marketplace_id=MARKETPLACE_ID, key=ZOOB_KEY)

wh = Webhook(
    description="asd",
    url="http://google.com",
    method="POST",
    events=["document.created", "document.updated"],

```

(continues on next page)

```
)  
  
response = client.add_webhook(wh)  
  
dump_response(response, os.path.basename(__file__).split(".")[0])
```

4.10.2 Deletar webhook

```
import os  
  
from zoop_wrapper import ZoopWrapper  
from examples.utils import dump_response  
  
"""  
Nesse momento as constantes podem ser criadas no arquivo .py.  
Mas é recomendado utilizar como variável de ambiente em um '.env'  
"""  
from zoop_wrapper.constants import MARKETPLACE_ID, ZOOP_KEY  
  
client = ZoopWrapper(marketplace_id=MARKETPLACE_ID, key=ZOOP_KEY)  
  
webhook_id = "2ad882a6f17b4ab194295bdff17d23ad"  
  
response = client.remove_webhook(webhook_id)  
  
dump_response(response, os.path.basename(__file__).split(".")[0])
```

4.10.3 Listar webhook

```
import os  
  
from zoop_wrapper import ZoopWrapper  
from examples.utils import dump_response  
  
"""  
Nesse momento as constantes podem ser criadas no arquivo .py.  
Mas é recomendado utilizar como variável de ambiente em um '.env'  
"""  
from zoop_wrapper.constants import MARKETPLACE_ID, ZOOP_KEY  
  
client = ZoopWrapper(marketplace_id=MARKETPLACE_ID, key=ZOOP_KEY)  
  
response = client.list_webhooks()  
  
dump_response(response, os.path.basename(__file__).split(".")[0])
```


4.10.4 Pegar webhook

```
import os

from zoop_wrapper import ZoopWrapper
from examples.utils import dump_response

"""
Nesse momento as constantes podem ser criadas no arquivo .py.
Mas é recomendado utilizar como variável de ambiente em um '.env'
"""
from zoop_wrapper.constants import MARKETPLACE_ID, ZOOP_KEY

client = ZoopWrapper(marketplace_id=MARKETPLACE_ID, key=ZOOP_KEY)

webhook_id = "577feddd61b14ec9ab202dfabdce831e"

response = client.retrieve_webhook(webhook_id)

dump_response(response, os.path.basename(__file__).split(".")[0])
```

4.11 constants

```
zoop_wrapper.constants.MARKETPLACE_ID = ''
    Marketplace id da Zoop

zoop_wrapper.constants.ZOOP_KEY = ''
    Chave de autenticação da Zoop
```

4.12 exceptions

exception zoop_wrapper.exceptions.**FieldError** (*name, reason*)
Base: Exception

Exceção para ser usada quando a validação de algum campo falha.

__init__ (*name, reason*)

Parâmetros

- **name** – nome do campo
- **reason** – motivo do erro

to_dict ()
transforma exceção para um dict

Retorna dict

exception zoop_wrapper.exceptions.**ValidationError** (*entity, errors*)
Base: Exception

Exceção para ser usada quando a validação de um ZoopObject ocorre

__init__ (*entity, errors*)

Parâmetros

- **entity** – entidade na qual o erro ocorreu
- **errors** – lista de qualquer coisa (preferencialmente *FieldError*)

`parse_errors()`

Traduz os erros do tipo *FieldError* para dict na listagem de erros

Returns: lista de objetos serializáveis

4.13 utils

`zoop_wrapper.utils.convert_currency_float_value_to_cents(value)`

Converte o valor recebido (que pode ser `str<int>`, `int`, `float`, `str<float>`) para um inteiro em centavos.

Essa função trunca a terceira casa decimal em diante de um float.

Exemplos

1234 => 1234 56.78 => 5678 56.78123 => 5678 56.7 => 5670 653.55 => 65355 “9876” => 9876 “91.23” => 9123

Parâmetros **value** – Valor a ser convertido

Retorna número inteiro em centavos

`zoop_wrapper.utils.get_logger(name)`

factory de Logger's

Parâmetros **name** – nome para gerar o logger

Retorna novo logger para `zoop_wrapper.{name}`

4.14 response

class `zoop_wrapper.response.ZoopResponse`

Base: `requests.models.Response`

Uma `requests.Response` recebida da API da Zoop.

Perigo: Essa classe NÃO é utilizada no código (não é instanciada).

Está na biblioteca apenas para ter o type hinting. Esses atributos são adicionados ao objeto `requests.Response`

data

json da resposta

Type dict

4.15 wrapper

class `zoop_wrapper.wrapper.ZoopWrapper` (*marketplace_id=None, key=None*)

Base: `zoop_wrapper.wrapper.bank_account.BankAccountWrapper`, `zoop_wrapper.wrapper.buyer.BuyerWrapper`, `zoop_wrapper.wrapper.card.CardWrapper`, `zoop_wrapper.wrapper.invoice.InvoiceWrapper`, `zoop_wrapper.wrapper.seller.SellerWrapper`, `zoop_wrapper.wrapper.transaction.TransactionWrapper`, `zoop_wrapper.wrapper.webhook.WebhookWrapper`

Zoop Wrapper

It contains methods for all resources.

4.15.1 zoop_wrapper.wrapper.base module

class `zoop_wrapper.wrapper.base.BaseZoopWrapper` (*marketplace_id=None, key=None*)

Base: `zoop_wrapper.wrapper.base.RequestsWrapper`

wrapper da Zoop API

__marketplace_id
marketplace id da zoop

__key
chave de autenticação da zoop

BASE_URL = 'https://api.zoop.ws/v1/marketplaces/'

__post_instance (*url, instance: zoop_wrapper.models.base.ZoopObject*)
http post com instância de um *ZoopObject*.

Parâmetros

- **url** – url da requisição
- **instance** – instância a ser utilizada

Raises *ValidationError* – quando a instância passada não é um *ZoopObject*. # noqa

Retorna (*ZoopResponse*)

__put_instance (*url, instance: zoop_wrapper.models.base.ZoopObject*)
http put com instância de um *ZoopObject*.

Parâmetros

- **url** – url da requisição
- **instance** – instância a ser utilizada

Raises *ValidationError* – quando a instância passada não é um *ZoopObject*. # noqa

Retorna (*ZoopResponse*)

class `zoop_wrapper.wrapper.base.RequestsWrapper` (*base_url*)

Base: `object`

wrapper da lib requests

__base_url
Url base para construir os requests

static `_RequestsWrapper__process_response` (*response*) →
zoop_wrapper.response.ZoopResponse

Processa a resposta.

Adiciona o *data* carregado do `requests.Response.json()`.

Adiciona *error* na resposta se tiver ocorrido erros

Parâmetros *response* (`requests.Response`) – resposta a ser processada

Raises `HttpError` – quando a resposta não foi ok (200 <= status <= 299)!

Retorna ‘objeto’ (`ZoopResponse`) de resposta http

_construct_url (*action=None*, *identifier=None*, *subaction=None*, *search=None*,
sub_action_before_identifier=False)

Constrói a url para o request.

Parâmetros

- **action** – nome do resource
- **identifier** – identificador de detalhe (ID)
- **search** – query com url args para serem buscados
- **sub_action_before_identifier** – flag para inverter a posição do identifier e subaction
- **subaction** – subação do resource

Exemplos

```
>>> rw = RequestsWrapper()
>>> rw._construct_url(action='seller', identifier='1', subaction='bank_
↳accounts', search='account_number=1') # noqa:
'rw.__base_url/seller/1/bank_accounts/search?account_number=1'
```

Retorna url completa para o request

_delete (*url*) → zoop_wrapper.response.ZoopResponse
http delete

Parâmetros *url* – url de requisição

Retorna (`ZoopResponse`)

_get (*url*) → zoop_wrapper.response.ZoopResponse
http get

Parâmetros *url* – url de requisição

Retorna (`ZoopResponse`)

_post (*url, data*) → zoop_wrapper.response.ZoopResponse
http post

Parâmetros

- **url** – url de requisição
- **data** (*dict*) – dados da requisição

Retorna (`ZoopResponse`)

`_put (url, data) → zoop_wrapper.response.ZoopResponse`
 http put

Parâmetros

- **url** – url de requisição
- **data** (*dict*) – dados da requisição

Retorna (*ZoopResponse*)

4.15.2 zoop_wrapper.wrapper.bank_account module

class `zoop_wrapper.wrapper.bank_account.BankAccountWrapper` (*marketplace_id=None, key=None*)

Base: `zoop_wrapper.wrapper.base.BaseZoopWrapper`

Possui os métodos do resource *BankAccount*

Aviso: Não importe isso diretamente!

Essa classe precisa de métodos presentes em outro wrapper

`BankAccountWrapper.add_bank_account_token` (*token: zoop_wrapper.models.token.Token*)
 Adiciona um *Token* para uma *BankAccount*.

Parâmetros **token** – *Token* para *BankAccount*.

Retorna response with instance of *Token*

`add_bank_account` (*data: dict*)
 Adiciona uma *BankAccount*.

Exemplos

```
>>> data = {
    'account_number': 'foo',
    'bank_code': 'foo',
    'holder_name': 'foo',
    'routing_number': 'foo',
    'taxpayer_id' or 'ein': 'foo',
    'type': 'foo'
}
```

Parâmetros **data** – dict of data

Retorna response with instance of *BankAccount*

`list_bank_accounts_by_seller` (*identifier*)
 Lista todas as *BankAccount*'s.

Retorna response with instances of *BankAccount*

`remove_bank_account` (*identifier: str*)
 Remove todas as *BankAccount* de um *Seller* usando o *identifier* deste.

Parâmetros **identifier** – uuid id

Retorna *ZoopResponse*

retrieve_bank_account (*identifier*)

Retorna uma *BankAccount*.

Parâmetros *identifier* – uuid id da *BankAccount*

Retorna response with instance of *BankAccount*

4.15.3 zoop_wrapper.wrapper.buyer module

class `zoop_wrapper.wrapper.buyer.BuyerWrapper` (*marketplace_id=None, key=None*)

Base: `zoop_wrapper.wrapper.base.BaseZoopWrapper`

Possui os métodos do resource *Buyer*

add_buyer (*data*: *Union[dict, zoop_wrapper.models.buyer.Buyer]*) →
zoop_wrapper.response.ZoopResponse
Adiciona um *Buyer*

Exemplos

```
>>> data = {
    "birthdate": 'foo',
    "email": "foo",
    "first_name": "foo",
    "last_name": "foo",
    "phone_number": "foo",
    "taxpayer_id": "foo",
    "address": {
        "city": "foo",
        "country_code": "foo"
        "line1": "foo",
        "line2": "foo",
        "line3": "foo",
        "neighborhood": "foo",
        "postal_code": "foo",
        "state": "foo",
    }
}
```

Parâmetros *data* (dict ou *Buyer*) – dados do *Buyer*

Retorna *ZoopResponse*

list_buyers () → `zoop_wrapper.response.ZoopResponse`

Lista todos os *Buyer*'s

Retorna *ZoopResponse*

remove_buyer (*identifier: str*) → `zoop_wrapper.response.ZoopResponse`

Remove um *Buyer*

Parâmetros *identifier* – uuid id

Retorna *ZoopResponse*

retrieve_buyer (*identifier: str*) → `zoop_wrapper.response.ZoopResponse`

Pega um *Buyer*

Parâmetros `identifier` – uuid id

Retorna *ZoopResponse*

`search_buyer` (*identifier: str*) → *zoop_wrapper.response.ZoopResponse*
 Buscar um *Buyer* pelo CPF ou CNPJ

Nota: Sim, o atributo é o *taxpayer_id* para os dois. Veja o código para entender.

Parâmetros `identifier` – CPF ou CNPJ

Retorna *ZoopResponse*

`update_buyer` (*identifier: str, data: Union[dict, zoop_wrapper.models.buyer.Buyer]*) → *zoop_wrapper.response.ZoopResponse*
 Atualiza um *Buyer*.

Exemplos

```
>>> data = {
    "birthdate": "1994-12-27",
    "email": "foo@bar.com",
    "first_name": "foo",
    "last_name": "foo",
    "phone_number": "+55 84 99999-9999",
    "taxpayer_id": "foo",
    "address": {
        "city": "Natal",
        "country_code": "BR"
        "line1": "foo",
        "line2": "123",
        "line3": "barbar",
        "neighborhood": "foofoo",
        "postal_code": "59152250",
        "state": "BR-RN",
    }
}
```

Parâmetros

- **`identifier`** – id do *Buyer*
- **`data`** – dados do *Buyer*

Retorna *ZoopResponse*

4.15.4 `zoop_wrapper.wrapper.card` module

class `zoop_wrapper.wrapper.card.CardWrapper` (*marketplace_id=None, key=None*)

Base: `zoop_wrapper.wrapper.base.BaseZoopWrapper`

Possui os métodos do resource *Card*

Aviso: Não importe isso diretamente!

Essa classe precisa de métodos presentes em outro wrapper

`_CardWrapper__add_card_token` (*card_token*: *zoop_wrapper.models.token.Token*)

Cria um *Token* do tipo *Card*

Parâmetros *card_token* – instância do *Token*

Retorna *ZoopResponse* com instância do *Token*

`add_card` (*data*: *Union[dict, zoop_wrapper.models.token.Token]*, *customer_identifier*: *str*)

Adiciona um cartão de crédito utilizando um Token de cartão de crédito

Exemplos

```
>>> data = {
    "card_number": "foo",
    "expiration_month": "foo",
    "expiration_year": "foo",
    "holder_name": "foo",
    "security_code": "foo"
}
```

Parâmetros

- **data** – dicionário de dados
- **customer_identifier** – uuid do consumidor (*Buyer* ou *Seller*) # noqa

Retorna *ZoopResponse* com instância do *Card*

`retrieve_card` (*identifier*)

retrieve card

Parâmetros *identifier* – uuid id

Retorna response without instance

4.15.5 zoop_wrapper.wrapper.invoice module

class `zoop_wrapper.wrapper.invoice.InvoiceWrapper` (*marketplace_id=None*, *key=None*)

Base: *zoop_wrapper.wrapper.base.BaseZoopWrapper*

Possui os métodos do resource for *Invoice*

`retrieve_invoice` (*identifier*)

Pega um *Invoice*

Parâmetros *identifier* – uuid id

Retorna resposta com instância do *Invoice*

4.15.6 zoop_wrapper.wrapper.seller module

class `zoop_wrapper.wrapper.seller.SellerWrapper` (*marketplace_id=None, key=None*)
 Base: `zoop_wrapper.wrapper.base.BaseZoopWrapper`

Possui os métodos do resource `Seller`

`_SellerWrapper__search_seller` (***kwargs*) → `zoop_wrapper.response.ZoopResponse`
 Busca um `Seller`.

Parâmetros `kwargs` – dicionário de valores a serem buscados

Retorna `ZoopResponse`

`add_seller` (*data: Union[dict, zoop_wrapper.models.seller.Seller]*) → `zoop_wrapper.response.ZoopResponse`
 Adiciona um `Seller`.

Exemplos

```
>>> data = {
    "birthdate": "1994-12-27",
    "email": "foo@bar.com",
    "first_name": "foo",
    "last_name": "foo",
    "phone_number": "+55 84 99999-9999",
    "taxpayer_id": "foo",
    "address": {
        "city": "Natal",
        "country_code": "BR"
        "line1": "foo",
        "line2": "123",
        "line3": "barbar",
        "neighborhood": "foofoo",
        "postal_code": "59152250",
        "state": "RN",
    }
}
```

```
>>> data = {
    "business_email": "foo",
    "business_name": "foo",
    "business_opening_date": "foo",
    "business_phone": "foo",
    "business_website": "foo",
    "ein": "foo",
    "owner": {
        "birthdate": "foo",
        "email": "foo",
        "first_name": "foo",
        "last_name": "foo",
        "phone_number": "foo",
        "taxpayer_id": "foo",
        "address": {
            "city": "Natal",
            "country_code": "BR"
            "line1": "foo",
            "line2": "123",
```

(continues on next page)

```

        "line3": "barbar",
        "neighborhood": "fooofoo",
        "postal_code": "59152250",
        "state": "RN",
    }
},
"business_address": {
    "city": "Natal",
    "country_code": "BR"
    "line1": "foo",
    "line2": "123",
    "line3": "barbar",
    "neighborhood": "fooofoo",
    "postal_code": "59152250",
    "state": "RN",
}
}
}

```

Parâmetros data – dados do *Seller*

Retorna *ZoopResponse*

list_seller_bank_accounts (*identifier: str*) → *zoop_wrapper.response.ZoopResponse*

Lista *BankAccount*'s de algum *Seller*

Parâmetros identifier – id do *Seller*

Retorna *ZoopResponse*

list_sellers () → *zoop_wrapper.response.ZoopResponse*

lista *Seller*'s existentes na Zoop.

Retorna *ZoopResponse*

remove_seller (*identifier: str*) → *zoop_wrapper.response.ZoopResponse*

Remove um *Seller*;

Parâmetros identifier – id do *Seller*

Retorna *ZoopResponse*

retrieve_seller (*identifier: str*) → *zoop_wrapper.response.ZoopResponse*

Pega um *Seller*

Parâmetros identifier – id do *Seller*

Retorna *ZoopResponse*

search_business_seller (*identifier: str*) → *zoop_wrapper.response.ZoopResponse*

search seller by CNPJ

Parâmetros identifier – ein (Employer Identification Number) is equivalent to CNPJ #

noqa:

Retorna response with instance of Seller

search_individual_seller (*identifier: str*) → *zoop_wrapper.response.ZoopResponse*

search seller by CPF

Parâmetros identifier – taxpayer_id is equivalent to CPF # noqa:

Retorna response with instance of Seller

`update_seller` (*identifier*: *str*; *data*: *Union[dict, zoop_wrapper.models.seller.Seller]*) → *zoop_wrapper.response.ZoopResponse*
 Atualiza um *Seller*.

Exemplos

```
>>> data = {
    "birthdate": "1994-12-27",
    "email": "foo@bar.com",
    "first_name": "foo",
    "last_name": "foo",
    "phone_number": "+55 84 99999-9999",
    "taxpayer_id": "foo",
    "address": {
        "city": "Natal",
        "country_code": "BR"
        "line1": "foo",
        "line2": "123",
        "line3": "barbar",
        "neighborhood": "foofoo",
        "postal_code": "59152250",
        "state": "BR-RN",
    }
}
```

```
>>> data = {
    "business_email": "foo",
    "business_name": "foo",
    "business_opening_date": "foo",
    "business_phone": "foo",
    "business_website": "foo",
    "ein": "foo",
    "owner": {
        "birthdate": "foo",
        "email": "foo",
        "first_name": "foo",
        "last_name": "foo",
        "phone_number": "foo",
        "taxpayer_id": "foo",
        "address": {
            "city": "foo",
            "country_code": "foo"
            "line1": "foo",
            "line2": "foo",
            "line3": "foo",
            "neighborhood": "foo",
            "postal_code": "foo",
            "state": "foo",
        }
    }
    "business_address": {
        "city": "foo",
        "country_code": "foo"
        "line1": "foo",
        "line2": "foo",
        "line3": "foo",
    }
```

(continues on next page)

```

        "neighborhood": "foo",
        "postal_code": "foo",
        "state": "foo",
    }
}

```

Parâmetros

- **identifier** – id do *Seller*
- **data** – dados do *Seller*

Retorna *ZoopResponse*

4.15.7 zoop_wrapper.wrapper.transaction module

class `zoop_wrapper.wrapper.transaction.TransactionWrapper` (*marketplace_id=None*,
key=None)

Base: `zoop_wrapper.wrapper.base.BaseZoopWrapper`

Possui os métodos do resource *Transaction*

`_capture_or_void_transaction` (*identifier*, *sub_action*, *amount=None*)

Estorna ou captura uma *Transaction*.

O *amount* é opcional, e deve ser um valor em centavos ou real. Caso ele não seja passado, o valor da transação é utilizado. Caso ele seja um valor menor do que a transação, é feita uma ação parcial no valor passado.

Aviso: o *amount* não pode ser maior do que o valor da quantia!

Exemplos

```

>>> ZoopWrapper()._capture_or_void_transaction('1', 'void')
>>> ZoopWrapper()._capture_or_void_transaction('1', 'void', '10.00')
>>> ZoopWrapper()._capture_or_void_transaction('1', 'capture', '10,00')
>>> ZoopWrapper()._capture_or_void_transaction('1', 'void', '1000')

```

Parâmetros

- **identifier** – uuid id da *Transaction*
- **sub_action** – string da ação a ser feita. 'void' ou 'capture'
- **amount** – quantia em centavos da ação a ser feita

Retorna response

`add_transaction` (*data: Union[dict, zoop_wrapper.models.transaction.Transaction]*)

Adiciona uma *Transaction*.

Exemplos

```
>>> data = {
    'amount' : 'foo',
    'currency' : 'BRL',
    'customer': 'foo',
    'description' : 'foo',
    'on_behalf_of' : 'foo',
    'payment_type' : 'foo',
    'reference_id' : 'foo',
    'payment_method' : {
        'body_instructions' : instructions,
        'expiration_date' : expiration_date,
        'payment_limit_date' : payment_limit_date,
        'billing_instructions' : {
            'discount' : discount
            'interest' : interest,
            'late_fee' : late_fee,
        }
    }
}
```

```
>>> data = {
    'amount': '1000',
    'currency': 'BRL',
    'customer': 'buyer_id',
    'description': 'meu boleto gerado para teste',
    'on_behalf_of': 'seller_id',
    'payment_type': 'boleto',
    'payment_method': {
        'expiration_date': '2020-06-20',
        'payment_limit_date': '2020-06-30',
        'billing_instructions': {
            'late_fee': {
                'mode': 'FIXED',
                'percentage': 30,
                'start_date': '2020-06-20'
            },
            'interest': {
                'mode': 'MONTHLY_PERCENTAGE',
                'percentage': 30,
                'start_date': '2020-06-20'
            },
            'discount': [{
                'amount': 300,
                'limit_date': '2020-06-20'
                'mode': 'FIXED',
            }]
        }
    }
}
```

Parâmetros data – dict of data

Retorna response with instance of Transaction

cancel_transaction (*identifier, amount=None*)

Estorna uma *Transaction*.

O `amount` é opcional, e deve ser um valor em centavos ou real. Caso ele não seja passado, o valor da transação é utilizado. Caso ele seja um valor menor do que a transação, é feita uma ação parcial no valor passado.

Aviso: o `amount` não pode ser maior do que o valor da quantia!

Exemplos

```
>>> ZoopWrapper().cancel_transaction('1', '10.00')
>>> ZoopWrapper().cancel_transaction('1', '10,00')
>>> ZoopWrapper().cancel_transaction('1', '1000')
```

Parâmetros

- **identifier** – uuid id da *Transaction*
- **amount** – quantia em centavos a ser estornada

Retorna response

capture_transaction (*identifier*, *amount=None*)

Captura uma *Transaction*.

O `amount` é opcional, e deve ser um valor em centavos ou real. Caso ele não seja passado, o valor da transação é utilizado. Caso ele seja um valor menor do que a transação, é feita uma ação parcial no valor passado.

Aviso: o `amount` não pode ser maior do que o valor da quantia!

Exemplos

```
>>> ZoopWrapper().capture_transaction('1', '10.00')
>>> ZoopWrapper().capture_transaction('1', '10,00')
>>> ZoopWrapper().capture_transaction('1', '1000')
```

Parâmetros

- **identifier** – uuid id da *Transaction*
- **amount** – quantia em centavos a ser capturada

Retorna response

list_transactions ()

Lista todas as *Transaction*'s

Retorna response

list_transactions_for_seller (*identifier*)

Lista todas as *Transaction*'s de um *Seller*

Parâmetros

- **identifier** – uuid id do *Seller*
- **offset** – ”

Retorna response

retrieve_transaction (*identifier*)

Retorna uma *Transaction*.

Parâmetros identifier – uuid id da *Transaction*

Retorna response

4.16 models

4.16.1 zoop_wrapper.models.base module

class zoop_wrapper.models.base.**Address** (*allow_empty=False, **kwargs*)

Base: *zoop_wrapper.models.base.ZoopObject*

Represents a physical address.

line1

complete street name

line2

number

line3

complement

neighborhood

neighborhood

city

city

state

Código ISO 3166-2 para o estado

postal_code

postal code

country_code

ISO 3166-1 alpha-2 - códigos de país de duas letras

classmethod **get_non_required_fields** ()

get set of non required fields

Retorna set of fields

class zoop_wrapper.models.base.**BusinessOrIndividualModel** (*allow_empty=False, **kwargs*)

Base: *zoop_wrapper.models.base.MarketPlaceModel*

Represents a Business Or Individual Model

It has dynamic types!

Can be Business or Individual.

taxpayer_id

cpf válido para type *INDIVIDUAL_TYPE*

```
ein
    cnpj para type BUSINESS_TYPE
BUSINESS_IDENTIFIER = 'ein'
BUSINESS_TYPE = 'business'
INDIVIDUAL_IDENTIFIER = 'taxpayer_id'
INDIVIDUAL_TYPE = 'individual'
URI = {'business': 'businesses', 'individual': 'individuals'}
```

get_all_fields()
get all fields for instance.
if type is *BUSINESS_TYPE* then call *get_business_required_fields()* and *get_business_non_required_fields()*
else type is *INDIVIDUAL_TYPE!* then call *get_individual_required_fields()* and *get_individual_non_required_fields()*
Retorna set of all fields

classmethod get_business_non_required_fields()
get set of non required fields for *BUSINESS_TYPE*.
Retorna set of fields

classmethod get_business_required_fields()
get set of required fields for *BUSINESS_TYPE*
Retorna set of fields

classmethod get_individual_non_required_fields()
get set of non required fields for *INDIVIDUAL_TYPE*
Retorna set of fields

classmethod get_individual_required_fields()
get set of required fields for *INDIVIDUAL_TYPE*
Retorna set of fields

get_type()
get the dynamic type from instance
Retorna *BUSINESS_TYPE* or *INDIVIDUAL_TYPE*

get_type_uri()
get the dynamic type uri for instance based on *get_type()*
Retorna uri string for type from *URI*

get_validation_fields()
Get validation fields for instance.
if type is *BUSINESS_TYPE* then call *get_business_required_fields()*
else type is *INDIVIDUAL_TYPE!* then call *get_individual_required_fields()*
Retorna set of fields to be used on validation

init_custom_fields (*taxpayer_id=None, ein=None, **kwargs*)
Chama *set_identifier()*.
Parâmetros

- **taxpayer_id** – cpf value
- **ein** – cnpj value
- ****kwargs** – dict of kwargs

set_identifier (*taxpayer_id=None, ein=None, **kwargs*)

Declara os atributos *taxpayer_id* ou (ou exclusivo) *ein*. Exatamente um deles deve ser passado e válido, e não os dois.

kwargs are there to be called from *Seller.init_custom_fields()* and *BankAccount.init_custom_fields()* without getting *taxpayer_id* or *ein* variables.

Parâmetros

- **taxpayer_id** – cpf
- **ein** – cnpj
- ****kwargs** – kwarg

classmethod validate_identifiers (*taxpayer_id, ein*)

Valida tupla de valores de identificação.

Raises *ValidationError* quando é passado os dois, ou nenhum, ou quando o identificador passado é inválido # noqa

class *zoop_wrapper.models.base.FinancialModel* (*allow_empty=False, **kwargs*)

Base: *zoop_wrapper.models.base.ZoopObject*

Have financial attributes.

status

pending or active string

account_balance

amount of balance

current_balance

current amount of balance

description

description

delinquent

boolean of verification

payment_methods

?

default_debit

?

default_credit

?

classmethod get_non_required_fields ()

get set of non required fields

Retorna set of fields

class *zoop_wrapper.models.base.MarketPlaceModel* (*allow_empty=False, **kwargs*)

Base: *zoop_wrapper.models.base.ResourceModel*

This class represents a *ResourceModel* which belongs to some marketplace from Zoop.

marketplace_id
identifier string

classmethod get_non_required_fields()
get set of non required fields

Retorna set of fields

class zoop_wrapper.models.base.**PaymentMethod** (*allow_empty=False, **kwargs*)

Base: *zoop_wrapper.models.base.ResourceModel*

Have some payment method attributes

description
text description

customer
uuid id

address
Address Model

classmethod get_non_required_fields()
get set of non required fields

Retorna set of fields

init_custom_fields (*address=None, **kwargs*)
initialize *address* with *Address*

Parâmetros

- **address** – dict of data or *Address*
- ****kwargs** – dic of kwargs

class zoop_wrapper.models.base.**Person** (*allow_empty=False, **kwargs*)

Base: *zoop_wrapper.models.base.ZoopObject*

Represents a person.

address
Address model

birthdate
birthdate

email
email

first_name
first name

last_name
last name

phone_number
phone number

taxpayer_id
cpf válido

full_name
get full name of the person

Retorna string with the full name

classmethod `get_non_required_fields()`
get set of non required fields

Retorna set of fields

classmethod `get_required_fields()`
get set of required fields

Retorna set of fields

init_custom_fields (*address=None, **kwargs*)
Initialize *address* with *Address*

Parâmetros

- **address** – dict of data or *Address*
- ****kwargs** –

validate_custom_fields (***kwargs*)

O *taxpayer_id* precisa ser um CPF válido. Então verificamos isso.

Parâmetros

- **raise_exception** – Quando algum campo está faltando ou CPF é inválido
- ****kwargs** –

class `zoop_wrapper.models.base.ResourceModel` (*allow_empty=False, **kwargs*)
Base: `zoop_wrapper.models.base.ZoopObject`

Represents a Model that is a resource.

id
identifier string

resource
type string

uri
uri string

created_at
date of creation

updated_at
date of update

metadata
dict with metadata

RESOURCE = None

classmethod `get_non_required_fields()`
get set of non required fields

Retorna set of fields

class `zoop_wrapper.models.base.SocialModel` (*allow_empty=False, **kwargs*)
Base: `zoop_wrapper.models.base.ZoopObject`

Have social sites uri's

facebook
facebook profile url?

twitter
twitter profile url?

classmethod `get_non_required_fields()`
get set of non required fields

Retorna set of fields

class `zoop_wrapper.models.base.VerificationModel` (*allow_empty=False, **kwargs*)
Base: `zoop_wrapper.models.base.ZoopObject`

Have some verification attributes.

postal_code_check
boolean of verification

address_line1_check
boolean of verification

classmethod `get_required_fields()`
get set of required fields

Retorna set of fields

class `zoop_wrapper.models.base.ZoopObject` (*allow_empty=False, **kwargs*)
Base: object

This class represent a bare Zoop object.

_allow_empty
boolean

__init__ (*allow_empty=False, **kwargs*)
initialize all fields from `get_all_fields()` as attributes from kwargs on instance.

Then call `validate_fields()`.

Parâmetros

- **allow_empty** – boolean which disable validation of required fields
- ****kwargs** – dictionary of args

classmethod `from_dict` (*data, allow_empty=False, **kwargs*)
to construct a instance of this class from dict

Parâmetros

- **data** – dict of data
- **allow_empty** – boolean
- ****kwargs** – kwargs
- **data** – dict of data may be None
- **allow_empty** – boolean
- ****kwargs** – kwargs

Raises `ValidationError` – se data não for do tipo ‘dict’ ou for None

Retorna instance initialized of cls

classmethod `from_dict_or_instance` (*data, **kwargs*)
Esse método existe para fazer um tratamento dos inputs de dados.

O atributo `data` pode ser um dict ou um `ZoopObject`.

Verifica se `data` já é uma instância da classe `ZoopObject` or uma subclasse.

Se não for, chama `from_dict()`.

Parâmetros

- **data** – dict of data or instance
- ****kwargs** – kwargs

Retorna instance initialized of `cls`

`get_all_fields()`

Método para pegar todos os campos!

Isso é necessário para classes/instances com diferentes campos obrigatórios definidos por um tipo dinâmico!

Tais como `Seller`, `BankAccount`, `Fine` e `Token`.

O padrão é `get_validation_fields()` + `get_non_required_fields()`.

Retorna set de todos os campos

`classmethod get_non_required_fields()`

get set of non required fields

Retorna set of fields

`get_original_different_fields_mapping()`

Método de mapeamento de nomes diferentes de atributo => API zoop a ser estendido.

Retorna Dicionário de `nome_custom => nome_oringial`

`classmethod get_required_fields()`

get set of required fields

Retorna set of fields

`get_validation_fields()`

Método para pegar os campos de validação!

Isso é necessário para classes/instances com diferentes campos obrigatórios definidos por um tipo dinâmico!

Tais como `Seller`, `BankAccount`, `Fine` e `Token`.

O padrão é `get_required_fields()`.

Retorna set de campos para serem utilizados na validação

`init_custom_fields(**kwargs)`

this method exists to set custom attributes such as `ZoopObject` instances. Since all attributes set on `__init__()` are dict 's or variables.

Parâmetros ****kwargs** – dictionary of args

`static is_value_empty(value)`

Verify if value passed is considered empty!

value may be None. As we set on `__init__()`:

```
value = kwargs.get(field_name, None)
```

value may be {} if it was a `ZoopObject` with `allow_empty!`

value may be [{}] if it was a list of `ZoopObject`'s with `allow_empty!!`

Parâmetros **value** – Value to be verified

Retorna boolean

static `make_data_copy_with_kwargs (data, **kwargs)`
make a new data dict from previous data dict with added kwargs

if data is None create a new empty dict.

data may be None for the cases we are explicitly calling with `allow_empty=True` on `init_custom_fields()` for some custom `ZoopObject` instance set. Such as:

```
instance = ZoopObject()
setattr(
    instance, 'address',
    Address.from_dict_or_instance(None, allow_empty=True)
)
```

Parâmetros

- **data** – dict of data may be None
- ****kwargs** – dict of kwargs

Retorna new dict of data

`to_dict ()`
serialize self to dict

Retorna dict of instance

validate_custom_fields (***kwargs*)
Método de validação a ser estendido para fazer uma validação especializada.

Esse método originalmente retorna uma lista vazia pois ele serve para ser sobrescrito pelas classes especializadas adicionando comportamento de validação!

Retorna Lista de erros a serem levantados.

validate_fields (*raise_exception=True, **kwargs*)
Valida na instância os campos retornados do conjunto `get_validation_fields()`.

Se `_allow_empty` é True não validar!

Esse método deve chamar o `validate_custom_fields()` para praticidade de extensão e especialização!

Parâmetros **raise_exception** – flag que dita se a exceção deve ser lançada ou não

Raises `ValidationError` se (algum campo obrigatório está faltando ou ocorreu algum erro no `validate_custom_fields()`) e `raise_exception==True` # noqa

4.16.2 zoop_wrapper.models.bank_account module

class `zoop_wrapper.models.bank_account.BankAccount` (*allow_empty=False, **kwargs*)

Base: `zoop_wrapper.models.base.BusinessOrIndividualModel`

Represent a Bank Account. <https://docs.zoop.co/reference#conta-banc%C3%A1ria>

The `RESOURCE` is used to identify this Model. Used to check against `resource!`

account_number
account number

```

bank_code
    code of bank

holder_name
    name of owner

routing_number
    agency code in BR

type
    type of account

address
    Address model

bank_name
    name of bank

country_code
    country code

customer
    id of owner

description
    description

debitable
    boolean of verification

fingerprint
    ?

is_active
    boolean of verification

is_verified
    boolean of verification

last4_digits
    last 4 digits of account number

phone_number
    phone number

verification_checklist
    VerificationCheckList model

CHECKING_TYPE = 'Checking'

RESOURCE = 'bank_account'

SAVING_TYPE = 'Savings'

TYPES = {'Checking', 'Savings'}

classmethod get_non_required_fields()
    get set of non required fields

    Retorna set of fields

classmethod get_required_fields()
    get set of required fields

    Retorna set of fields

```

init_custom_fields (*type=None, address=None, verification_checklist=None, **kwargs*)

Initialize *address* as *Address*.

Initialize *verification_checklist* as *BankAccountVerificationModel*.

Parâmetros

- **type** (*str*) – value containing type
- **address** (dict or *Address*) – address
- **verification_checklist** (dict or *BankAccountVerificationModel*) – verifications # noqa
- ****kwargs** –

classmethod validate_type (*type*)

Validate bank account type

Parâmetros **type** (*str*) – value of type to be validated

Raises *ValidationError* – when type is not in *TYPES*

class zoop_wrapper.models.bank_account.**BankAccountVerificationModel** (*allow_empty=False, **kwargs*)

Base: *zoop_wrapper.models.base.VerificationModel*

Have some bank account verification attributes.

deposit_check

boolean of verification

classmethod get_required_fields ()

get set of required fields

Retorna set of fields

4.16.3 zoop_wrapper.models.buyer module

class zoop_wrapper.models.buyer.**Buyer** (*allow_empty=False, **kwargs*)

Base: *zoop_wrapper.models.base.MarketPlaceModel*, *zoop_wrapper.models.base.Person*, *zoop_wrapper.models.base.SocialModel*, *zoop_wrapper.models.base.FinancialModel*

Represent a buyer. <https://docs.zoop.co/reference#comprador-1>

The *RESOURCE* is used to identify this Model. Used to check against *resource*!

default_receipt_delivery_method

?

RESOURCE = 'buyer'

classmethod get_non_required_fields ()

get set of non required fields

Retorna set of fields

validate_custom_fields (***kwargs*)

O *taxpayer_id* precisa ser um CPF ou CNPJ válido. Então verificamos isso.

Parâmetros ****kwargs** –

4.16.4 zoop_wrapper.models.card module

```

class zoop_wrapper.models.card.Card (allow_empty=False, **kwargs)
    Base: zoop_wrapper.models.base.PaymentMethod

    Represent a Card. https://docs.zoop.co/reference#card%C3%A3o

    The RESOURCE is used to identify this Model. Used to check against resource!

card_brand
    company name

expiration_month
    month of expiration

expiration_year
    year of expiration

fingerprint
    unique card identifier from company of card ?

first4_digits
    first 4 digits of card

holder_name
    owner name

is_active
    boolean of verification

is_valid
    boolean of verification

is_verified
    boolean of verification

last4_digits
    last 4 digits of card

verification_checklist
    CardVerificationChecklist model

RESOURCE = 'card'

classmethod get_non_required_fields()
    get set of non required fields

    Retorna set of fields

classmethod get_required_fields()
    get set of required fields

    Retorna set of fields

init_custom_fields (verification_checklist=None, **kwargs)
    Initialize verification_checklist as CardVerificationChecklist

Parâmetros
    • verification_checklist – dict of data or CardVerificationChecklist
    • **kwargs – kwargs

class zoop_wrapper.models.card.CardVerificationChecklist (allow_empty=False,
    **kwargs)
    Base: zoop_wrapper.models.base.VerificationModel

```

Represent a credit card verification

security_code_check

boolean of verification

classmethod get_required_fields()

get set of required fields

Retorna set of fields

4.16.5 zoop_wrapper.models.invoice module

class zoop_wrapper.models.invoice.**BaseModeObject** (*allow_empty=False, **kwargs*)

Base: *zoop_wrapper.models.base.ZoopObject*

Um objeto base que possui modos de quantia e porcentagem

MODES = {}

classmethod get_fixed_required_fields()

get_mode_required_fields_mapping()

classmethod get_percentage_required_fields()

classmethod get_required_fields()

get set of required fields

Retorna set of fields

get_validation_fields()

Método para pegar os campos de validação!

Isso é necessário para classes/instances com diferentes campos obrigatórios definidos por um tipo dinâmico!

Tais como *Seller, BankAccount, Fine e Token*.

O padrão é *get_required_fields()*.

Retorna set de campos para serem utilizados na validação

init_custom_fields (*mode=None, **kwargs*)

É necessário configurar o mode antes pois ele influencia no *get_validation_fields()*

class zoop_wrapper.models.invoice.**BillingInstructions** (*allow_empty=False, **kwargs*)

Base: *zoop_wrapper.models.base.ZoopObject*

Represents billing instructions (fine, interest and discount)

discount

list of optional discount rules # noqa

Type list of BillingConfiguration

interest

optional interest rules

Type BillingConfiguration

late_fee

optional fine rules

Type BillingConfiguration

classmethod `get_non_required_fields()`

Conjunto de campos não obrigatórios

Retorna set de campos

init_custom_fields (*late_fee=None, interest=None, discount=None, **kwargs*)

Inicializa `late_fee`, `interest` e `discount`.

Parâmetros

- **discount** – dict or instance of BillingConfiguration model
- **interest** – dict or instance of BillingConfiguration model
- **late_fee** – dict or instance of BillingConfiguration model
- ****kwargs** – kwargs

class `zoop_wrapper.models.invoice.Discount` (*allow_empty=False, **kwargs*)

Base: `zoop_wrapper.models.invoice.BaseModeObject`

Representa um desconto!

<https://docs.zoop.co/docs/multa-juros-e-descontos#descontos>

FIXED = 'FIXED'

MODES = {'FIXED', 'PERCENTAGE'}

PERCENTAGE = 'PERCENTAGE'

get_mode_required_fields_mapping()

classmethod `get_required_fields()`

get set of required fields

Retorna set of fields

class `zoop_wrapper.models.invoice.Fine` (*allow_empty=False, **kwargs*)

Base: `zoop_wrapper.models.invoice.BaseModeObject`

Representa a multa!

<https://docs.zoop.co/docs/multa-juros-e-descontos#multa>

FIXED = 'FIXED'

MODES = {'FIXED', 'PERCENTAGE'}

PERCENTAGE = 'PERCENTAGE'

get_mode_required_fields_mapping()

classmethod `get_non_required_fields()`

get set of non required fields

Retorna set of fields

class `zoop_wrapper.models.invoice.Interest` (*allow_empty=False, **kwargs*)

Base: `zoop_wrapper.models.invoice.BaseModeObject`

Representa um juros!

<https://docs.zoop.co/docs/multa-juros-e-descontos#juros>

DAILY_AMOUNT = 'DAILY_AMOUNT'

DAILY_PERCENTAGE = 'DAILY_PERCENTAGE'

```
MODES = {'DAILY_AMOUNT', 'DAILY_PERCENTAGE', 'MONTHLY_PERCENTAGE'}
```

```
MONTHLY_PERCENTAGE = 'MONTHLY_PERCENTAGE'
```

```
get_mode_required_fields_mapping()
```

```
classmethod get_non_required_fields()
```

```
get set of non required fields
```

Retorna set of fields

```
class zoop_wrapper.models.invoice.Invoice(allow_empty=False, **kwargs)
```

```
Base: zoop_wrapper.models.base.PaymentMethod
```

Represents a invoice ('boleto' in BR). <https://docs.zoop.co/reference#boleto>

```
billing_instructions
```

```
optional billing instructions # noqa
```

Type *BillingInstructions*

```
security_code_check
```

```
verification of security code
```

Type bool

```
RESOURCE = 'boleto'
```

```
classmethod get_non_required_fields()
```

```
get set of non required fields
```

Retorna set of fields

```
classmethod get_required_fields()
```

```
get set of required fields
```

Retorna set of fields

```
init_custom_fields(billing_instructions=None, **kwargs)
```

```
initialize billing_instructions with BillingInstructions
```

Parâmetros

- **billing_instructions** (dict or *BillingInstructions*) – data
- ****kwargs** –

4.16.6 zoop_wrapper.models.seller module

```
class zoop_wrapper.models.seller.Seller(allow_empty=False, **kwargs)
```

```
Base: zoop_wrapper.models.base.BusinessOrIndividualModel, zoop_wrapper.models.base.Person, zoop_wrapper.models.base.FinancialModel, zoop_wrapper.models.base.SocialModel
```

Represent a seller. <https://docs.zoop.co/reference#vendedor-1>

The *RESOURCE* is used to identify this Model. Used to check against *resource*!

```
decline_on_fail_security_code
```

```
value of verification
```

Type bool

```
decline_on_fail_zipcode
```

```
value of verification
```

Type bool

is_mobile
value of verification

Type bool

mcc
?

merchant_code
?

show_profile_online
?

statement_descriptor
?

terminal_code
?

type
individual or business string

Type str

taxpayer_id
Optional value

Type str

website
Optional value

Type str

business_address
Optional value

Type *Address*

business_description
optional value

Type str

business_email
optional value

Type str

business_facebook
optional value

Type str

business_name
optional value

Type str

business_opening_date
optional value

Type str

business_phone

optional value

Type str

business_twitter

optional value

Type str

business_website

optional value

Type str

ein

optional value

Type str

owner

Optional value

Type *Person*

RESOURCE = 'seller'

full_name

Get full name for the *Seller*.

If dynamic type is *BUSINESS_TYPE* it will have the owner attribute.

Else dynamic type is *INDIVIDUAL_TYPE*. So we call the super() which will find the method on Person class.

Retorna string with the full name

classmethod get_business_non_required_fields()

Get set of non required fields for *BUSINESS_TYPE*

Retorna set of fields

classmethod get_business_required_fields()

Get set of required fields for *BUSINESS_TYPE*

Retorna "set" of fields

classmethod get_individual_non_required_fields()

Get set of non required fields for *INDIVIDUAL_TYPE*

Retorna set of fields

classmethod get_individual_required_fields()

Get set of required fields for *INDIVIDUAL_TYPE*

Retorna set of fields

classmethod get_non_required_fields()

get set of non required fields

Retorna set of fields

classmethod get_required_fields()

get set of required fields

Retorna set of fields

init_custom_fields (*business_address=None, owner=None, **kwargs*)

If dynamic type is *BUSINESS_TYPE* then initialize *owner* with *Person* and initialize *business_address* with *Address*.

Else dynamic type is *INDIVIDUAL_TYPE*! Then initialize self with *Person*.

Parâmetros

- **business_address** (dict or *Address*) – data
- **owner** (dict or *Person*) – data
- ****kwargs** – kwargs

validate_custom_fields (***kwargs*)

Caso o vendedor seja *BUSINESS_TYPE* precisamos validar os campos pelo *BusinessOrIndividualModel*.

Caso o vendedor seja *INDIVIDUAL_TYPE* precisamos validar os campos pelo *Person*.

Parâmetros ****kwargs** –

4.16.7 zoop_wrapper.models.token module

class zoop_wrapper.models.token.**Token** (*allow_empty=False, **kwargs*)

Base: *zoop_wrapper.models.base.ResourceModel*

Token is a resource used to link a *BankAccount* Or *Card* and a *Seller* or *Buyer*. <https://docs.zoop.co/reference#token-1>

The *RESOURCE* is used to identify this Model. Used to check against *resource*!

It has dynamic types!

It can be *CARD_TYPE* or *BANK_ACCOUNT_TYPE*.

But before creation it won't have attribute type. So we need to verify by other attributes. After created on Zoop it will have type.

token_type

value for identified token type

Type str

type

optional *BANK_ACCOUNT_TYPE* or *CARD_TYPE*. It has collision with of *BankAccount.type*. So we need the above *token_type*.

Type str

used

optional value of verification

Type bool

bank_account

optional value (for created token of 'bank_account' type)

Type *BankAccount*

card

optional value (for created token of 'card' type)

Type *Card*

holder_name
owner name (for both token of 'bank_account' and 'card' type)
Type str

account_number
account number for *BANK_ACCOUNT_TYPE*
Type str

taxpayer_id
identifier for *BANK_ACCOUNT_TYPE* of *INDIVIDUAL_TYPE*
Type str

ein
identifier for *BANK_ACCOUNT_TYPE* of *BUSINESS_TYPE*
Type str

bank_code
bank code for *BANK_ACCOUNT_TYPE*
Type str

routing_number
agency code in BR for *BANK_ACCOUNT_TYPE*
Type str

card_number
card number for *CARD_TYPE*
Type str

expiration_month
month of expiration for *CARD_TYPE*
Type str

expiration_year
year of expiration for *CARD_TYPE*
Type str

security_code
security code for *CARD_TYPE*
Type str

BANK_ACCOUNT_IDENTIFIER = 'bank_code'
BANK_ACCOUNT_TYPE = 'bank_account'
CARD_IDENTIFIER = 'card_number'
CARD_TYPE = 'card'
IDENTIFIERS = {'bank_code', 'card_number'}
RESOURCE = 'token'
TYPES = {'bank_account', 'card'}
get_all_fields()
Get all fields for instance.
fields is *get_validation_fields()*

if *token_type* is *CARD_TYPE* return fields union *get_card_non_required_fields()*.

else *token_type* is *BANK_ACCOUNT_TYPE* return fields union *get_bank_account_non_required_fields()*.

Retorna set of all fields

classmethod *get_bank_account_non_required_fields()*

Get set of non required fields for *BANK_ACCOUNT_TYPE*

Retorna set of fields

classmethod *get_bank_account_required_fields()*

get set of required fields for *BANK_ACCOUNT_TYPE*

Retorna set of fields

get_bank_account_type()

Get bank account type for creation token of :class:'.BankAccount.

Raises TypeError – when called from a token not from ‘bank_account’ type

Retorna value with bank_account type

classmethod *get_card_non_required_fields()*

Get set of non required fields for *CARD_TYPE*.

Retorna set of fields

classmethod *get_card_required_fields()*

Get set of required fields for *CARD_TYPE*.

Retorna set of fields

classmethod *get_non_required_fields()*

get set of non required fields

Retorna set of fields

get_validation_fields()

Get validation fields for instance.

if *token_type* is *CARD_TYPE* card return *get_card_required_fields()*.

else *token_type* is *BANK_ACCOUNT_TYPE*! fields is *get_bank_account_required_fields()*.

if bank account type is *INDIVIDUAL_TYPE* return fields union *get_individual_required_fields()*.

else bank account type is *BUSINESS_TYPE* return fields union *get_business_required_fields()*.

Retorna set of fields to be validated

init_custom_fields (*type=None, card=None, bank_account=None, **kwargs*)

if type is *BANK_ACCOUNT_TYPE* or *CARD_TYPE* token is created!

set *card* or *bank_account* attributes accordingly.

else token is not created!

We must identify token type from attr's passed searching for *CARD_IDENTIFIER* or *BANK_ACCOUNT_IDENTIFIER*. After identifying type if it was *BANK_ACCOUNT_TYPE* set business or individual identifier from *BankAccount* method (which is from *BusinessOrIndividualModel*).

Parâmetros

- **bank_account** (dict or *BankAccount*) – data
- **card** (dict or *Card*) – data
- **type** (*str*) – type for token or bank account
- ****kwargs** – kwargs

validate_custom_fields (***kwargs*)

Valida campos do token.

Se for um token de cartão, valida o *card_number*.

Parâmetros ****kwargs** –

Retorna Lista com os erros ocorridos (se tiver algum!)

4.16.8 zoop_wrapper.models.transaction module

class zoop_wrapper.models.transaction.**History** (*allow_empty=False, **kwargs*)

Base: *zoop_wrapper.models.base.ZoopObject*

Represents a update for *Transaction*

amount

amount value for the update

authorization_code

??

authorization_nsu

??

authorizer

??

authorizer_id

??

created_at

datetime for the update

gatewayResponseTime

??

id

uuid identifier

operation_type

type for the update

response_code

??

response_message

??

status

status for the update

transaction

transaction uuid identifier

```
classmethod get_non_required_fields()
    get set of non required fields
```

Retorna set of fields

```
class zoop_wrapper.models.transaction.InstallmentPlan(allow_empty=False,
                                                    **kwargs)
```

Base: *zoop_wrapper.models.base.ZoopObject*

```
INSTALLMENT_PLAN_MODES = {'interest_free', 'with_interest'}
```

```
INTEREST_FREE_MODE = 'interest_free'
```

```
WITH_INTEREST_MODE = 'with_interest'
```

```
classmethod _validate_number_installments(number_installments)
```

Esse método verifica se:

- *number_installments* é inteiro
- *number_installments* é um valor inteiro entre 1 e 12 incluindo as bordas

Retorna bool

```
classmethod get_required_fields()
    get set of required fields
```

Retorna set of fields

```
validate_custom_fields(**kwargs)
```

Método de validação a ser estendido para fazer uma validação especializada.

Esse método originalmente retorna uma lista vazia pois ele serve para ser sobrescrito pelas classes especializadas adicionando comportamento de validação!

Retorna Lista de erros a serem levantados.

```
class zoop_wrapper.models.transaction.PointOfSale(allow_empty=False, **kwargs)
```

Base: *zoop_wrapper.models.base.ZoopObject*

Represents something (?)

```
entry_mode
    ??
```

```
identification_number
    ??
```

```
classmethod get_non_required_fields()
    get set of non required fields
```

Retorna set of fields

```
class zoop_wrapper.models.transaction.Source(allow_empty=False, **kwargs)
```

Base: *zoop_wrapper.models.base.ZoopObject*

```
CARD_NOT_PRESENT_TYPE = 'card_not_present_type'
```

```
CARD_PRESENT_TYPE = 'card_present_type'
```

```
SOURCE_TYPES = {'card_not_present_type', 'card_present_type'}
```

```
get_all_fields()
```

Pega todos os campos da instância.

Retorna set de todos os campos

classmethod `get_card_not_present_required_fields()`

Método get do set de required fields para CARD_TYPE quando o cartão é presente.

Retorna set de campos

classmethod `get_card_present_required_fields()`

Método get do set de non required fields para CARD_TYPE.

Retorna set de campos

classmethod `get_non_required_fields()` → set

get set of non required fields

Retorna set of fields

classmethod `get_required_fields()`

get set of required fields

Retorna set of fields

get_validation_fields()

Pega campos de validação da instância.

O conjunto de campos é construído com base no card_type.

Se for `CARD_PRESENT_TYPE` utiliza o `get_card_present_required_fields()`.

Se não, utiliza o `get_card_not_present_required_fields()`.

Retorna set de campos para ser validados

init_custom_fields (*card=None, type='card', currency='BRL', installment_plan=None, **kwargs*)

this method exists to set custom attributes such as ZoopObject instances. Since all attributes set on `__init__()` are dict's or variables.

Parâmetros ****kwargs** – dictionary of args

class `zoop_wrapper.models.transaction.Transaction` (*allow_empty=False, **kwargs*)

Base: `zoop_wrapper.models.base.ResourceModel`

Represents a transaction <https://docs.zoop.co/reference#transa%C3%A7%C3%A3o>

The `RESOURCE` is used to identify this Model. Used to check against `resource!`

amount

integer amount value in 'centavos'

Type int

app_transaction_uid

??

business

??

capture

flag que designa se será uma transação simples {true} ou uma composta (com pre autorização) {false} #noqa

Type bool

captured

flag indica se a transação foi capturada ou não

Type bool

confirmed
value of cofirmation
Type str

currency
coin currency string
Type str

customer
customer uuid identifier
Type str

description
value description
Type str

discounts
??

expected_on
datetime string
Type str

fee_details
??

fees
??

gateway_authorizer
??

history
transaction updates
Type list of *History*

individual
??

installment_plan
??

location_latitude
??

location_longitude
??

on_behalf_of
seller uuid identifier
Type str

original_amount
original amount value
Type int

payment_method
payment method used

Type *Card* or *Invoice*

payment_type
payment type

Type str

point_of_sale
??

Type *PointOfSale*

pre_authorization
??

reference_id
??

refunded
boolean of verification

Type bool

refunds
??

rewards
??

sales_receipt

statement_descriptor
value description

Type str

status
value for status

Type str

transaction_number
??

voided
boolean of verification

Type bool

BOLETO_TYPE = 'boleto'

CARD_TYPE = 'credit'

PAYMENT_TYPES = {'boleto', 'credit'}

RESOURCE = 'transaction'

get_all_fields()
Pega todos os campos para instância.

O conjunto de campos é construído com base no *get_validation_fields()* com a união do *get_non_required_fields()*.

Retorna set de todos os campos

classmethod *get_boleto_required_fields()*

```
classmethod get_card_required_fields()
```

```
classmethod get_non_required_fields()
```

get set of non required fields

Retorna set of fields

```
classmethod get_required_fields()
```

get set of required fields

Retorna set of fields

```
get_validation_fields()
```

Pega os campos de validação para uma instância.

O conjunto de campos é feito com base no *payment_type*.

Se for *CARD_TYPE* utiliza o *get_card_required_fields()*.

Se não, ele é *payment_type* é *BOLETO_TYPE*! Utiliza o *get_boleto_required_fields()*.

Retorna set de campos para serem validados

```
init_custom_fields(amount=None, currency='BRL', history=None, id=None, payment_method=None, payment_type=None, point_of_sale=None, source=None, **kwargs)
```

Initialize *payment_method* as *Card* or *Invoice* based on data.

Initialize *point_of_sale* as *PointOfSale*.

Initialize *history* as list of *History*.

Parâmetros

- **currency** (*str*) – default currency is 'BRL'. So users may not need to pass currency!
- **history** (dict or *History* or list of either) – history data. May be a list of dict or list of *History* # noqa
- **payment_method** (dict or *Card* or *Invoice*) – payment method data # noqa
- **payment_type** (*str*) – value for payment type
- **point_of_sale** (dict or *PointOfSale*) – point of sale data
- ****kwargs** – kwargs

4.16.9 zoop_wrapper.models.utils module

```
zoop_wrapper.models.utils._get_model_class_from_resource(resource)
```

Get model class from resource

Exemplos

```
>>> _get_model_class_from_resource('seller')
Seller
>>> _get_model_class_from_resource('bank_account')
BankAccount
```

Parâmetros **resource** (*str*) – value of resource

Raises *ValueError* – when the resource is not identified

Retorna *ResourceModel* subclass

`zoop_wrapper.models.utils.get_instance_from_data(data)`
Factory Pattern for *ResourceModel* subclasses

Exemplos

```
>>> data = {'resource': 'seller'}
>>> get_instance_from_data(data)
Seller.from_dict(data)
>>> data = {'resource': 'bank_account'}
>>> get_instance_from_data(data)
BankAccount.from_dict(data)
```

Parâmetros `data` (*dict*) – data

Retorna *ResourceModel* subclass or None

Z

- zoop_wrapper.constants, 29
- zoop_wrapper.exceptions, 29
- zoop_wrapper.models.bank_account, 50
- zoop_wrapper.models.base, 43
- zoop_wrapper.models.buyer, 52
- zoop_wrapper.models.card, 53
- zoop_wrapper.models.invoice, 54
- zoop_wrapper.models.seller, 56
- zoop_wrapper.models.token, 59
- zoop_wrapper.models.transaction, 62
- zoop_wrapper.models.utils, 67
- zoop_wrapper.response, 30
- zoop_wrapper.utils, 30
- zoop_wrapper.wrapper, 31
- zoop_wrapper.wrapper.bank_account, 33
- zoop_wrapper.wrapper.base, 31
- zoop_wrapper.wrapper.buyer, 34
- zoop_wrapper.wrapper.card, 35
- zoop_wrapper.wrapper.invoice, 36
- zoop_wrapper.wrapper.seller, 37
- zoop_wrapper.wrapper.transaction, 40

Símbolos

| | | | |
|---|---|--|---|
| <code>__BankAccountWrapper__add_bank_account_token()</code> | (método <code>zoop_wrapper.wrapper.bank_account.BankAccountWrapper</code>), 33 | <code>get()</code> | (método <code>zoop_wrapper.wrapper.base.RequestsWrapper</code>), 32 |
| <code>__CardWrapper__add_card_token()</code> | (método <code>zoop_wrapper.wrapper.card.CardWrapper</code>), 36 | <code>__get_model_class_from_resource()</code> | (no módulo <code>zoop_wrapper.models.utils</code>), 67 |
| <code>__RequestsWrapper__process_response()</code> | (método estático <code>zoop_wrapper.wrapper.base.RequestsWrapper</code>), 31 | <code>__post()</code> | (método <code>zoop_wrapper.wrapper.base.RequestsWrapper</code>), 32 |
| <code>__SellerWrapper__search_seller()</code> | (método <code>zoop_wrapper.wrapper.seller.SellerWrapper</code>), 37 | <code>__post_instance()</code> | (método <code>zoop_wrapper.wrapper.base.BaseZoopWrapper</code>), 31 |
| <code>__base_url</code> | (atributo <code>zoop_wrapper.wrapper.base.RequestsWrapper</code>), 31 | <code>__put()</code> | (método <code>zoop_wrapper.wrapper.base.RequestsWrapper</code>), 32 |
| <code>__init__()</code> | (método <code>zoop_wrapper.exceptions.FieldError</code>), 29 | <code>__put_instance()</code> | (método <code>zoop_wrapper.wrapper.base.BaseZoopWrapper</code>), 31 |
| <code>__init__()</code> | (método <code>zoop_wrapper.exceptions.ValidationError</code>), 29 | <code>__validate_number_installments()</code> | (método de clase <code>zoop_wrapper.models.transaction.InstallmentPlan</code>), 63 |
| <code>__init__()</code> | (método <code>zoop_wrapper.models.base.ZoopObject</code>), 48 | A | |
| <code>__key</code> | (atributo <code>zoop_wrapper.wrapper.base.BaseZoopWrapper</code>), 31 | <code>account_balance</code> | (atributo <code>zoop_wrapper.models.base.FinancialModel</code>), 45 |
| <code>__marketplace_id</code> | (atributo <code>zoop_wrapper.wrapper.base.BaseZoopWrapper</code>), 31 | <code>account_number</code> | (atributo <code>zoop_wrapper.models.bank_account.BankAccount</code>), 50 |
| <code>__allow_empty</code> | (atributo <code>zoop_wrapper.models.base.ZoopObject</code>), 48 | <code>account_number</code> | (atributo <code>zoop_wrapper.models.token.Token</code>), 60 |
| <code>__capture_or_void_transaction()</code> | (método <code>zoop_wrapper.wrapper.transaction.TransactionWrapper</code>), 40 | <code>add_bank_account()</code> | (método <code>zoop_wrapper.wrapper.bank_account.BankAccountWrapper</code>), 33 |
| <code>__construct_url()</code> | (método <code>zoop_wrapper.wrapper.base.RequestsWrapper</code>), 32 | <code>add_buyer()</code> | (método <code>zoop_wrapper.wrapper.buyer.BuyerWrapper</code>), 34 |
| <code>__delete()</code> | (método <code>zoop_wrapper.wrapper.base.RequestsWrapper</code>), 32 | <code>add_card()</code> | (método <code>zoop_wrapper.wrapper.card.CardWrapper</code>), 36 |
| | | <code>add_seller()</code> | (método <code>zoop_wrapper.wrapper.seller.SellerWrapper</code>), 37 |

| | |
|---|--|
| add_transaction() | (método BaseModeObject (classe em zoop_wrapper.wrapper.transaction.TransactionWrapper), zoop_wrapper.models.invoice), 54 |
| 40 | BaseZoopWrapper (classe em zoop_wrapper.wrapper.base), 31 |
| address (atributo zoop_wrapper.models.bank_account.BankAccount), | billing_instructions (atributo zoop_wrapper.models.invoice.Invoice), 56 |
| 51 | BillingInstructions (classe em zoop_wrapper.models.invoice), 54 |
| address (atributo zoop_wrapper.models.base.PaymentMethod), | birthdate (atributo zoop_wrapper.models.base.Person), 46 |
| 46 | Address (classe em zoop_wrapper.models.base), 43 |
| address_line1_check (atributo zoop_wrapper.models.base.VerificationModel), | BOLETO_TYPE (atributo zoop_wrapper.models.transaction.Transaction), 66 |
| 48 | amount (atributo zoop_wrapper.models.transaction.History), 62 |
| amount (atributo zoop_wrapper.models.transaction.Transaction), | business (atributo zoop_wrapper.models.transaction.Transaction), 64 |
| 64 | amount (atributo zoop_wrapper.models.transaction.Transaction), |
| 64 | business_address (atributo zoop_wrapper.models.seller.Seller), 57 |
| app_transaction_uid (atributo zoop_wrapper.models.transaction.Transaction), | business_description (atributo zoop_wrapper.models.seller.Seller), 57 |
| 64 | business_email (atributo zoop_wrapper.models.seller.Seller), 57 |
| authorization_code (atributo zoop_wrapper.models.transaction.History), | business_facebook (atributo zoop_wrapper.models.seller.Seller), 57 |
| 62 | business_identifier (atributo zoop_wrapper.models.base.BusinessOrIndividualModel), 44 |
| authorization_nsu (atributo zoop_wrapper.models.transaction.History), | 62 |
| 62 | authorizer (atributo zoop_wrapper.models.transaction.History), |
| 62 | business_name (atributo zoop_wrapper.models.seller.Seller), 57 |
| authorizer_id (atributo zoop_wrapper.models.transaction.History), | business_opening_date (atributo zoop_wrapper.models.seller.Seller), 57 |
| 62 | business_phone (atributo zoop_wrapper.models.seller.Seller), 57 |
| | business_twitter (atributo zoop_wrapper.models.seller.Seller), 58 |
| B | |
| bank_account (atributo zoop_wrapper.models.token.Token), 59 | BUSINESS_TYPE (atributo zoop_wrapper.models.base.BusinessOrIndividualModel), 44 |
| BANK_ACCOUNT_IDENTIFIER (atributo zoop_wrapper.models.token.Token), 60 | business_website (atributo zoop_wrapper.models.seller.Seller), 58 |
| BANK_ACCOUNT_TYPE (atributo zoop_wrapper.models.token.Token), 60 | BusinessOrIndividualModel (classe em zoop_wrapper.models.base), 43 |
| bank_code (atributo zoop_wrapper.models.bank_account.BankAccount), | Buyer (classe em zoop_wrapper.models.buyer), 52 |
| 50 | bank_code (atributo zoop_wrapper.models.token.Token), |
| 60 | BuyerWrapper (classe em zoop_wrapper.wrapper.buyer), 34 |
| bank_name (atributo zoop_wrapper.models.bank_account.BankAccount), | |
| 51 | C |
| BankAccount (classe em zoop_wrapper.models.bank_account), 50 | cancel_transaction() (método zoop_wrapper.wrapper.transaction.TransactionWrapper), 41 |
| BankAccountVerificationModel (classe em zoop_wrapper.models.bank_account), 52 | capture (atributo zoop_wrapper.models.transaction.Transaction), 64 |
| BankAccountWrapper (classe em zoop_wrapper.wrapper.bank_account), 33 | capture_transaction() (método zoop_wrapper.wrapper.transaction.TransactionWrapper), 42 |
| BASE_URL (atributo zoop_wrapper.wrapper.base.BaseZoopWrapper), 31 | |

captured (atributo `zoop_wrapper.models.transaction.Transaction`),
 64
 card (atributo `zoop_wrapper.models.token.Token`), 59
 Card (clase em `zoop_wrapper.models.card`), 53
 card_brand (atributo `zoop_wrapper.models.card.Card`), 53
 CARD_IDENTIFIER (atributo `zoop_wrapper.models.token.Token`), 60
 CARD_NOT_PRESENT_TYPE (atributo `zoop_wrapper.models.transaction.Source`),
 63
 card_number (atributo `zoop_wrapper.models.token.Token`), 60
 CARD_PRESENT_TYPE (atributo `zoop_wrapper.models.transaction.Source`),
 63
 CARD_TYPE (atributo `zoop_wrapper.models.token.Token`), 60
 CARD_TYPE (atributo `zoop_wrapper.models.transaction.Transaction`), 66
 CardVerificationChecklist (clase em `zoop_wrapper.models.card`), 53
 CardWrapper (clase em `zoop_wrapper.wrapper.card`), 35
 CHECKING_TYPE (atributo `zoop_wrapper.models.bank_account.BankAccount`),
 51
 city (atributo `zoop_wrapper.models.base.Address`), 43
 confirmed (atributo `zoop_wrapper.models.transaction.Transaction`),
 64
 convert_currency_float_value_to_cents() (no módulo `zoop_wrapper.utils`), 30
 country_code (atributo `zoop_wrapper.models.bank_account.BankAccount`),
 51
 country_code (atributo `zoop_wrapper.models.base.Address`), 43
 created_at (atributo `zoop_wrapper.models.base.ResourceModel`),
 47
 created_at (atributo `zoop_wrapper.models.transaction.History`),
 62
 currency (atributo `zoop_wrapper.models.transaction.Transaction`), 65
 current_balance (atributo `zoop_wrapper.models.base.FinancialModel`),
 45
 customer (atributo `zoop_wrapper.models.bank_account.BankAccount`),
 51
 customer (atributo `zoop_wrapper.models.base.PaymentMethod`), 46
 customer (atributo `zoop_wrapper.models.transaction.Transaction`), 65
 DAILY_AMOUNT (atributo `zoop_wrapper.models.invoice.Interest`), 55
 DAILY_PERCENTAGE (atributo `zoop_wrapper.models.invoice.Interest`), 55
 data (atributo `zoop_wrapper.response.ZoopResponse`),
 30
 debitable (atributo `zoop_wrapper.models.bank_account.BankAccount`),
 51
 decline_on_fail_security_code (atributo `zoop_wrapper.models.seller.Seller`), 56
 decline_on_fail_zipcode (atributo `zoop_wrapper.models.seller.Seller`), 56
 default_credit (atributo `zoop_wrapper.models.base.FinancialModel`),
 45
 default_debit (atributo `zoop_wrapper.models.base.FinancialModel`),
 45
 default_receipt_delivery_method (atributo `zoop_wrapper.models.buyer.Buyer`), 52
 delinquent (atributo `zoop_wrapper.models.base.FinancialModel`),
 45
 deposit_check (atributo `zoop_wrapper.models.bank_account.BankAccountVerificationModel`),
 52
 description (atributo `zoop_wrapper.models.bank_account.BankAccount`),
 51
 description (atributo `zoop_wrapper.models.base.FinancialModel`),
 45
 description (atributo `zoop_wrapper.models.base.PaymentMethod`),
 46
 description (atributo `zoop_wrapper.models.transaction.Transaction`),
 65
 discount (atributo `zoop_wrapper.models.invoice.BillingInstructions`),
 54
 Discount (clase em `zoop_wrapper.models.invoice`), 55
 discounts (atributo `zoop_wrapper.models.transaction.Transaction`),
 65
E
 ein (atributo `zoop_wrapper.models.base.BusinessOrIndividualModel`),
 44
 ein (atributo `zoop_wrapper.models.seller.Seller`), 58
 ein (atributo `zoop_wrapper.models.token.Token`), 60
 email (atributo `zoop_wrapper.models.base.Person`), 46
 entry_mode (atributo `zoop_wrapper.models.transaction.PointOfSale`),
 63

| | | | |
|--|-------------------------------|--|---------|
| expected_on | (atributo | get_all_fields() | (método |
| zoop_wrapper.models.transaction.Transaction), | 65 | zoop_wrapper.models.base.BusinessOrIndividualModel), | 44 |
| expiration_month | (atributo | get_all_fields() | (método |
| zoop_wrapper.models.card.Card), | 53 | zoop_wrapper.models.base.ZoopObject), | 49 |
| expiration_month | (atributo | get_all_fields() | (método |
| zoop_wrapper.models.token.Token), | 60 | zoop_wrapper.models.token.Token), | 60 |
| expiration_year | (atributo | get_all_fields() | (método |
| zoop_wrapper.models.card.Card), | 53 | zoop_wrapper.models.transaction.Source), | 63 |
| expiration_year | (atributo | get_all_fields() | (método |
| zoop_wrapper.models.token.Token), | 60 | zoop_wrapper.models.transaction.Transaction), | 66 |
| F | | get_bank_account_non_required_fields() | (método |
| facebook | (atributo | de | classe |
| zoop_wrapper.models.base.SocialModel), | 47 | zoop_wrapper.models.token.Token), | 61 |
| fee_details | (atributo | get_bank_account_required_fields() | (método |
| zoop_wrapper.models.transaction.Transaction), | 65 | de | classe |
| fees | (atributo | zoop_wrapper.models.token.Token), | 61 |
| zoop_wrapper.models.transaction.Transaction), | 65 | get_bank_account_type() | (método |
| FieldError, | 29 | zoop_wrapper.models.token.Token), | 61 |
| FinancialModel | (classe | get_boleto_required_fields() | (método |
| em | zoop_wrapper.models.base), | de | classe |
| 45 | | zoop_wrapper.models.transaction.Transaction), | 66 |
| Fine | (classe | get_business_non_required_fields() | (método |
| em | zoop_wrapper.models.invoice), | de | classe |
| 55 | | zoop_wrapper.models.base.BusinessOrIndividualModel), | 44 |
| fingerprint | (atributo | get_business_required_fields() | (método |
| zoop_wrapper.models.bank_account.BankAccount), | 51 | de | classe |
| fingerprint | (atributo | zoop_wrapper.models.seller.Seller), | 58 |
| zoop_wrapper.models.card.Card), | 53 | get_business_required_fields() | (método |
| first4_digits | (atributo | de | classe |
| zoop_wrapper.models.card.Card), | 53 | zoop_wrapper.models.base.BusinessOrIndividualModel), | 44 |
| first_name | (atributo | get_business_non_required_fields() | (método |
| zoop_wrapper.models.base.Person), | 46 | de | classe |
| FIXED | (atributo | zoop_wrapper.models.seller.Seller), | 58 |
| zoop_wrapper.models.invoice.Discount), | 55 | get_business_required_fields() | (método |
| FIXED | (atributo | de | classe |
| zoop_wrapper.models.invoice.Fine), | 55 | zoop_wrapper.models.base.BusinessOrIndividualModel), | 44 |
| from_dict() | (método | get_business_required_fields() | (método |
| de | classe | de | classe |
| zoop_wrapper.models.base.ZoopObject), | 48 | zoop_wrapper.models.seller.Seller), | 58 |
| from_dict_or_instance() | (método | get_card_non_required_fields() | (método |
| de | classe | de | classe |
| zoop_wrapper.models.base.ZoopObject), | 48 | zoop_wrapper.models.token.Token), | 61 |
| full_name | (atributo | get_card_not_present_required_fields() | (método |
| zoop_wrapper.models.base.Person), | 46 | de | classe |
| full_name | (atributo | zoop_wrapper.models.transaction.Source), | 64 |
| zoop_wrapper.models.seller.Seller), | 58 | get_card_present_required_fields() | (método |
| G | | de | classe |
| gateway_authorizer | (atributo | zoop_wrapper.models.transaction.Source), | 64 |
| zoop_wrapper.models.transaction.Transaction), | 65 | get_card_required_fields() | (método |
| gatewayResponseTime | (atributo | de | classe |
| zoop_wrapper.models.transaction.History), | 62 | zoop_wrapper.models.token.Token), | 61 |
| | | get_card_required_fields() | (método |
| | | de | classe |
| | | zoop_wrapper.models.transaction.Transaction), | 66 |

`get_fixed_required_fields()` (método de classe `zoop_wrapper.models.invoice.BaseModeObject`), 54
`get_individual_non_required_fields()` (método de classe `zoop_wrapper.models.base.BusinessOrIndividualModel`), 44
`get_individual_non_required_fields()` (método de classe `zoop_wrapper.models.seller.Seller`), 58
`get_individual_required_fields()` (método de classe `zoop_wrapper.models.base.BusinessOrIndividualModel`), 44
`get_individual_required_fields()` (método de classe `zoop_wrapper.models.seller.Seller`), 58
`get_instance_from_data()` (no módulo `zoop_wrapper.models.utils`), 68
`get_logger()` (no módulo `zoop_wrapper.utils`), 30
`get_mode_required_fields_mapping()` (método `zoop_wrapper.models.invoice.BaseModeObject`), 54
`get_mode_required_fields_mapping()` (método `zoop_wrapper.models.invoice.Discount`), 55
`get_mode_required_fields_mapping()` (método `zoop_wrapper.models.invoice.Fine`), 55
`get_mode_required_fields_mapping()` (método `zoop_wrapper.models.invoice.Interest`), 56
`get_non_required_fields()` (método de classe `zoop_wrapper.models.bank_account.BankAccount`), 51
`get_non_required_fields()` (método de classe `zoop_wrapper.models.base.Address`), 43
`get_non_required_fields()` (método de classe `zoop_wrapper.models.base.FinancialModel`), 45
`get_non_required_fields()` (método de classe `zoop_wrapper.models.base.MarketPlaceModel`), 46
`get_non_required_fields()` (método de classe `zoop_wrapper.models.base.PaymentMethod`), 46
`get_non_required_fields()` (método de classe `zoop_wrapper.models.base.Person`), 46
`get_non_required_fields()` (método de classe `zoop_wrapper.models.base.ResourceModel`), 47
`get_non_required_fields()` (método de classe `zoop_wrapper.models.base.SocialModel`), 48
`get_non_required_fields()` (método de classe `zoop_wrapper.models.base.ZoopObject`), 49
`get_non_required_fields()` (método de classe `zoop_wrapper.models.buyer.Buyer`), 52
`get_non_required_fields()` (método de classe `zoop_wrapper.models.card.Card`), 53
`get_non_required_fields()` (método de classe `zoop_wrapper.models.invoice.BillingInstructions`), 54
`get_non_required_fields()` (método de classe `zoop_wrapper.models.invoice.Fine`), 55
`get_non_required_fields()` (método de classe `zoop_wrapper.models.invoice.Interest`), 56
`get_non_required_fields()` (método de classe `zoop_wrapper.models.invoice.Invoice`), 56
`get_non_required_fields()` (método de classe `zoop_wrapper.models.seller.Seller`), 58
`get_non_required_fields()` (método de classe `zoop_wrapper.models.token.Token`), 61
`get_non_required_fields()` (método de classe `zoop_wrapper.models.transaction.History`), 62
`get_non_required_fields()` (método de classe `zoop_wrapper.models.transaction.PointOfSale`), 63
`get_non_required_fields()` (método de classe `zoop_wrapper.models.transaction.Source`), 64
`get_non_required_fields()` (método de classe `zoop_wrapper.models.transaction.Transaction`), 67
`get_original_different_fields_mapping()` (método `zoop_wrapper.models.base.ZoopObject`), 49
`get_percentage_required_fields()` (método de classe `zoop_wrapper.models.invoice.BaseModeObject`), 54
`get_required_fields()` (método de classe `zoop_wrapper.models.bank_account.BankAccount`), 51
`get_required_fields()` (método de classe `zoop_wrapper.models.bank_account.BankAccountVerificationModel`), 52
`get_required_fields()` (método de classe `zoop_wrapper.models.base.Person`), 47
`get_required_fields()` (método de classe `zoop_wrapper.models.base.VerificationModel`), 48
`get_required_fields()` (método de classe `zoop_wrapper.models.base.ZoopObject`), 49
`get_required_fields()` (método de classe `zoop_wrapper.models.card.Card`), 53
`get_required_fields()` (método de classe `zoop_wrapper.models.card.CardVerificationChecklist`), 54
`get_required_fields()` (método de classe `zoop_wrapper.models.invoice.BaseModeObject`),

| | | | |
|-------------------------|--|-----------------------|--|
| 54 | | | |
| get_required_fields() | (método de classe | id (atributo | zoop_wrapper.models.transaction.History), |
| | zoop_wrapper.models.invoice.Discount), | | 62 |
| get_required_fields() | (método de classe | identification_number | (atributo |
| | zoop_wrapper.models.invoice.Invoice), | | zoop_wrapper.models.transaction.PointOfSale), |
| get_required_fields() | (método de classe | | 63 |
| | zoop_wrapper.models.seller.Seller), | IDENTIFIERS | (atributo |
| get_required_fields() | (método de classe | | zoop_wrapper.models.token.Token), |
| | zoop_wrapper.models.transaction.InstallmentPlan), | individual | (atributo |
| | 63 | | zoop_wrapper.models.transaction.Transaction), |
| get_required_fields() | (método de classe | | 65 |
| | zoop_wrapper.models.transaction.Source), | INDIVIDUAL_IDENTIFIER | (atributo |
| get_required_fields() | (método de classe | | zoop_wrapper.models.base.BusinessOrIndividualModel), |
| | zoop_wrapper.models.transaction.Transaction), | | 44 |
| | 67 | INDIVIDUAL_TYPE | (atributo |
| get_type() | (método | | zoop_wrapper.models.base.BusinessOrIndividualModel), |
| | zoop_wrapper.models.base.BusinessOrIndividualModel), | | 44 |
| | 44 | init_custom_fields() | (método |
| get_type_uri() | (método | | zoop_wrapper.models.bank_account.BankAccount), |
| | zoop_wrapper.models.base.BusinessOrIndividualModel), | | 51 |
| | 44 | init_custom_fields() | (método |
| get_validation_fields() | (método | | zoop_wrapper.models.base.BusinessOrIndividualModel), |
| | zoop_wrapper.models.base.BusinessOrIndividualModel), | | 44 |
| | 44 | init_custom_fields() | (método |
| get_validation_fields() | (método | | zoop_wrapper.models.base.PaymentMethod), |
| | zoop_wrapper.models.base.ZoopObject), | | 46 |
| | 49 | init_custom_fields() | (método |
| get_validation_fields() | (método | | zoop_wrapper.models.base.Person), |
| | zoop_wrapper.models.invoice.BaseModeObject), | init_custom_fields() | (método |
| | 54 | | zoop_wrapper.models.base.ZoopObject), |
| | | | 49 |
| get_validation_fields() | (método | init_custom_fields() | (método |
| | zoop_wrapper.models.token.Token), | | zoop_wrapper.models.card.Card), |
| get_validation_fields() | (método | | 53 |
| | zoop_wrapper.models.transaction.Source), | init_custom_fields() | (método |
| | 64 | | zoop_wrapper.models.invoice.BaseModeObject), |
| | | | 54 |
| get_validation_fields() | (método | init_custom_fields() | (método |
| | zoop_wrapper.models.transaction.Transaction), | | zoop_wrapper.models.invoice.BillingInstructions), |
| | 67 | | 55 |
| | | init_custom_fields() | (método |
| H | | | zoop_wrapper.models.invoice.Invoice), |
| history (atributo | zoop_wrapper.models.transaction.Transaction), | | 56 |
| | 65 | init_custom_fields() | (método |
| History (classe em | zoop_wrapper.models.seller.Seller), | | 58 |
| | 62 | init_custom_fields() | (método |
| holder_name | (atributo | | zoop_wrapper.models.token.Token), |
| | zoop_wrapper.models.bank_account.BankAccount), | init_custom_fields() | (método |
| | 51 | | zoop_wrapper.models.transaction.Source), |
| | | | 64 |
| holder_name | (atributo | init_custom_fields() | (método |
| | zoop_wrapper.models.card.Card), | | zoop_wrapper.models.transaction.Transaction), |
| holder_name | (atributo | | 67 |
| | zoop_wrapper.models.token.Token), | installment_plan | (atributo |
| | 59 | | zoop_wrapper.models.transaction.Transaction), |
| | | | 65 |
| I | | | |
| id (atributo | zoop_wrapper.models.base.ResourceModel), | | |

| | | | |
|--------------------------------|---|--|---|
| INSTALLMENT_PLAN_MODES | (atributo | zoop_wrapper.wrapper.seller.SellerWrapper), | |
| | zoop_wrapper.models.transaction.InstallmentPlan), | 38 | |
| 63 | | list_transactions() (método | |
| InstallmentPlan | (clase | em | zoop_wrapper.wrapper.transaction.TransactionWrapper), |
| | zoop_wrapper.models.transaction), | 63 | 42 |
| interest | (atributo | zoop_wrapper.models.invoice.BillingInstructions), | transactions_for_seller() (método |
| | 54 | | zoop_wrapper.wrapper.transaction.TransactionWrapper), |
| Interest | (clase | em | zoop_wrapper.models.invoice), |
| | 55 | | 42 |
| INTEREST_FREE_MODE | (atributo | location_latitude | (atributo |
| | zoop_wrapper.models.transaction.InstallmentPlan), | zoop_wrapper.models.transaction.Transaction), | |
| | 63 | | 65 |
| Invoice | (clase | em | zoop_wrapper.models.invoice), |
| | 56 | | location_longitude |
| InvoiceWrapper | (clase | em | zoop_wrapper.models.transaction.Transaction), |
| | zoop_wrapper.wrapper.invoice), | 36 | 65 |
| is_active | (atributo | zoop_wrapper.models.bank_account.BankAccount), | |
| | 51 | | M |
| is_active | (atributo | zoop_wrapper.models.card.Card), | make_data_copy_with_kwargs() (método es- |
| | 53 | | tático |
| is_mobile | (atributo | zoop_wrapper.models.seller.Seller), | zoop_wrapper.models.base.ZoopObject), |
| | 57 | | 50 |
| is_valid | (atributo | zoop_wrapper.models.card.Card), | marketplace_id |
| | 53 | | (atributo |
| is_value_empty() | (método | estático | zoop_wrapper.models.base.ZoopObject), |
| | 49 | | MARKETPLACE_ID |
| | | | (no |
| | | | módulo |
| | | | zoop_wrapper.constants), |
| | | | 29 |
| is_verified | (atributo | zoop_wrapper.models.bank_account.BankAccount), | MarketPlaceModel |
| | 51 | | (clase |
| | | | em |
| | | | zoop_wrapper.models.base), |
| | | | 45 |
| is_verified | (atributo | zoop_wrapper.models.card.Card), | merchant_code |
| | 53 | | (atributo |
| | | | zoop_wrapper.models.seller.Seller), |
| | | | 57 |
| | | | metadata |
| | | | (atributo |
| | | | zoop_wrapper.models.base.ResourceModel), |
| | | | 47 |
| L | | | MODES |
| | | | (atributo |
| | | | zoop_wrapper.models.invoice.BaseModeObject), |
| | | | 54 |
| last4_digits | (atributo | zoop_wrapper.models.bank_account.BankAccount), | MODES |
| | 51 | | (atributo |
| | | | zoop_wrapper.models.invoice.Discount), |
| | | | 55 |
| last4_digits | (atributo | zoop_wrapper.models.card.Card), | MODES |
| | 53 | | (atributo |
| | | | zoop_wrapper.models.invoice.Fine), |
| | | | 55 |
| last_name | (atributo | zoop_wrapper.models.base.Person), | MODES |
| | 46 | | (atributo |
| | | | zoop_wrapper.models.invoice.Interest), |
| | | | 56 |
| late_fee | (atributo | zoop_wrapper.models.invoice.BillingInstructions), | MONTHLY_PERCENTAGE |
| | 54 | | (atributo |
| | | | zoop_wrapper.models.invoice.Interest), |
| | | | 56 |
| line1 | (atributo | zoop_wrapper.models.base.Address), | N |
| | 43 | | |
| line2 | (atributo | zoop_wrapper.models.base.Address), | neighborhood |
| | 43 | | (atributo |
| line3 | (atributo | zoop_wrapper.models.base.Address), | zoop_wrapper.models.base.Address), |
| | 43 | | 43 |
| list_bank_accounts_by_seller() | (método | zoop_wrapper.wrapper.bank_account.BankAccountWrapper), | |
| | 33 | | O |
| list_buyers() | (método | zoop_wrapper.wrapper.buyer.BuyerWrapper), | on_behalf_of |
| | 34 | | (atributo |
| | | | zoop_wrapper.models.transaction.Transaction), |
| | | | 65 |
| list_seller_bank_accounts() | (método | zoop_wrapper.wrapper.seller.SellerWrapper), | operation_type |
| | 38 | | (atributo |
| | | | zoop_wrapper.models.transaction.History), |
| | | | 62 |
| list_sellers() | (método | | |

| | | |
|-------------------|---|--|
| original_amount | (atributo <i>zoop_wrapper.models.transaction.Transaction</i>), 65 | refunds (atributo <i>zoop_wrapper.models.transaction.Transaction</i>), 66 |
| owner | (atributo <i>zoop_wrapper.models.seller.Seller</i>), 58 | remove_bank_account () (método <i>zoop_wrapper.wrapper.bank_account.BankAccountWrapper</i>), 33 |
| P | | remove_buyer () (método <i>zoop_wrapper.wrapper.buyer.BuyerWrapper</i>), 34 |
| parse_errors () | (método <i>zoop_wrapper.exceptions.ValidationError</i>), 30 | remove_seller () (método <i>zoop_wrapper.wrapper.seller.SellerWrapper</i>), 38 |
| payment_method | (atributo <i>zoop_wrapper.models.transaction.Transaction</i>), 65 | RequestsWrapper (clase <i>em zoop_wrapper.wrapper.base</i>), 31 |
| payment_methods | (atributo <i>zoop_wrapper.models.base.FinancialModel</i>), 45 | RESOURCE (atributo <i>zoop_wrapper.models.bank_account.BankAccount</i>), 51 |
| payment_type | (atributo <i>zoop_wrapper.models.transaction.Transaction</i>), 66 | RESOURCE (atributo <i>zoop_wrapper.models.base.ResourceModel</i>), 47 |
| PAYMENT_TYPES | (atributo <i>zoop_wrapper.models.transaction.Transaction</i>), 66 | resource (atributo <i>zoop_wrapper.models.base.ResourceModel</i>), 47 |
| PaymentMethod | (clase <i>em zoop_wrapper.models.base</i>), 46 | RESOURCE (atributo <i>zoop_wrapper.models.buyer.Buyer</i>), 52 |
| PERCENTAGE | (atributo <i>zoop_wrapper.models.invoice.Discount</i>), 55 | RESOURCE (atributo <i>zoop_wrapper.models.card.Card</i>), 53 |
| PERCENTAGE | (atributo <i>zoop_wrapper.models.invoice.Fine</i>), 55 | RESOURCE (atributo <i>zoop_wrapper.models.invoice.Invoice</i>), 56 |
| Person | (clase <i>em zoop_wrapper.models.base</i>), 46 | RESOURCE (atributo <i>zoop_wrapper.models.seller.Seller</i>), 58 |
| phone_number | (atributo <i>zoop_wrapper.models.bank_account.BankAccount</i>), 51 | RESOURCE (atributo <i>zoop_wrapper.models.token.Token</i>), 60 |
| phone_number | (atributo <i>zoop_wrapper.models.base.Person</i>), 46 | RESOURCE (atributo <i>zoop_wrapper.models.transaction.Transaction</i>), 66 |
| point_of_sale | (atributo <i>zoop_wrapper.models.transaction.Transaction</i>), 66 | ResourceModel (clase <i>em zoop_wrapper.models.base</i>), 47 |
| PointOfSale | (clase <i>em zoop_wrapper.models.transaction</i>), 63 | response_code (atributo <i>zoop_wrapper.models.transaction.History</i>), 62 |
| postal_code | (atributo <i>zoop_wrapper.models.base.Address</i>), 43 | response_message (atributo <i>zoop_wrapper.models.transaction.History</i>), 62 |
| postal_code_check | (atributo <i>zoop_wrapper.models.base.VerificationModel</i>), 48 | retrieve_bank_account () (método <i>zoop_wrapper.wrapper.bank_account.BankAccountWrapper</i>), 34 |
| pre_authorization | (atributo <i>zoop_wrapper.models.transaction.Transaction</i>), 66 | retrieve_buyer () (método <i>zoop_wrapper.wrapper.buyer.BuyerWrapper</i>), 34 |
| R | | retrieve_card () (método <i>zoop_wrapper.wrapper.card.CardWrapper</i>), 36 |
| reference_id | (atributo <i>zoop_wrapper.models.transaction.Transaction</i>), 66 | retrieve_invoice () (método <i>zoop_wrapper.wrapper.invoice.InvoiceWrapper</i>), 36 |
| refunded | (atributo <i>zoop_wrapper.models.transaction.Transaction</i>), 66 | retrieve_seller () (método <i>zoop_wrapper.wrapper.seller.SellerWrapper</i>), 38 |
| | | retrieve_transaction () (método |

| | |
|--|--|
| <code>zoop_wrapper.wrapper.transaction.TransactionWrapper</code> , 43 | <code>status</code> (atributo <code>zoop_wrapper.models.transaction.Transaction</code>), 66 |
| <code>rewards</code> (atributo <code>zoop_wrapper.models.transaction.Transaction</code>), 66 | <code>transaction</code> (atributo <code>zoop_wrapper.models.transaction.History</code>), 62 |
| <code>routing_number</code> (atributo <code>zoop_wrapper.models.bank_account.BankAccount</code>), 51 | <code>transaction</code> (atributo <code>zoop_wrapper.models.transaction.History</code>), 62 |
| <code>routing_number</code> (atributo <code>zoop_wrapper.models.token.Token</code>), 60 | T |
| S | <code>taxpayer_id</code> (atributo <code>zoop_wrapper.models.base.BusinessOrIndividualModel</code>), 43 |
| <code>sales_receipt</code> (atributo <code>zoop_wrapper.models.transaction.Transaction</code>), 66 | <code>taxpayer_id</code> (atributo <code>zoop_wrapper.models.base.Person</code>), 46 |
| <code>SAVING_TYPE</code> (atributo <code>zoop_wrapper.models.bank_account.BankAccount</code>), 51 | <code>taxpayer_id</code> (atributo <code>zoop_wrapper.models.seller.Seller</code>), 57 |
| <code>search_business_seller()</code> (método <code>zoop_wrapper.wrapper.seller.SellerWrapper</code>), 38 | <code>taxpayer_id</code> (atributo <code>zoop_wrapper.models.token.Token</code>), 60 |
| <code>search_buyer()</code> (método <code>zoop_wrapper.wrapper.buyer.BuyerWrapper</code>), 35 | <code>terminal_code</code> (atributo <code>zoop_wrapper.models.seller.Seller</code>), 57 |
| <code>search_individual_seller()</code> (método <code>zoop_wrapper.wrapper.seller.SellerWrapper</code>), 38 | <code>to_dict()</code> (método <code>zoop_wrapper.exceptions.FieldError</code>), 29 |
| <code>security_code</code> (atributo <code>zoop_wrapper.models.token.Token</code>), 60 | <code>to_dict()</code> (método <code>zoop_wrapper.models.base.ZoopObject</code>), 50 |
| <code>security_code_check</code> (atributo <code>zoop_wrapper.models.card.CardVerificationCheckList</code>), 54 | <code>Token</code> (classe em <code>zoop_wrapper.models.token</code>), 59 |
| <code>security_code_check</code> (atributo <code>zoop_wrapper.models.invoice.Invoice</code>), 56 | <code>token_type</code> (atributo <code>zoop_wrapper.models.token.Token</code>), 59 |
| <code>Seller</code> (classe em <code>zoop_wrapper.models.seller</code>), 56 | <code>transaction</code> (atributo <code>zoop_wrapper.models.transaction.History</code>), 62 |
| <code>SellerWrapper</code> (classe em <code>zoop_wrapper.wrapper.seller</code>), 37 | <code>transaction</code> (classe em <code>zoop_wrapper.models.transaction</code>), 64 |
| <code>set_identifier()</code> (método <code>zoop_wrapper.models.base.BusinessOrIndividualModel</code>), 45 | <code>transaction_number</code> (atributo <code>zoop_wrapper.models.transaction.Transaction</code>), 66 |
| <code>show_profile_online</code> (atributo <code>zoop_wrapper.models.seller.Seller</code>), 57 | <code>TransactionWrapper</code> (classe em <code>zoop_wrapper.wrapper.transaction</code>), 40 |
| <code>SocialModel</code> (classe em <code>zoop_wrapper.models.base</code>), 47 | <code>twitter</code> (atributo <code>zoop_wrapper.models.base.SocialModel</code>), 47 |
| <code>Source</code> (classe em <code>zoop_wrapper.models.transaction</code>), 63 | <code>type</code> (atributo <code>zoop_wrapper.models.bank_account.BankAccount</code>), 51 |
| <code>SOURCE_TYPES</code> (atributo <code>zoop_wrapper.models.transaction.Source</code>), 63 | <code>type</code> (atributo <code>zoop_wrapper.models.seller.Seller</code>), 57 |
| <code>state</code> (atributo <code>zoop_wrapper.models.base.Address</code>), 43 | <code>type</code> (atributo <code>zoop_wrapper.models.token.Token</code>), 59 |
| <code>statement_descriptor</code> (atributo <code>zoop_wrapper.models.seller.Seller</code>), 57 | <code>TYPES</code> (atributo <code>zoop_wrapper.models.bank_account.BankAccount</code>), 51 |
| <code>statement_descriptor</code> (atributo <code>zoop_wrapper.models.transaction.Transaction</code>), 66 | <code>TYPES</code> (atributo <code>zoop_wrapper.models.token.Token</code>), 60 |
| | U |
| | <code>update_buyer()</code> (método <code>zoop_wrapper.wrapper.buyer.BuyerWrapper</code>), 35 |
| | <code>update_seller()</code> (método <code>zoop_wrapper.wrapper.seller.SellerWrapper</code>), 38 |

updated_at (atributo *zoop_wrapper.models.base.ResourceModel*), 47

URI (atributo *zoop_wrapper.models.base.BusinessOrIndividualModel*), 44

uri (atributo *zoop_wrapper.models.base.ResourceModel*), 47

used (atributo *zoop_wrapper.models.token.Token*), 59

V

validate_custom_fields() (método *zoop_wrapper.models.base.Person*), 47

validate_custom_fields() (método *zoop_wrapper.models.base.ZoopObject*), 50

validate_custom_fields() (método *zoop_wrapper.models.buyer.Buyer*), 52

validate_custom_fields() (método *zoop_wrapper.models.seller.Seller*), 59

validate_custom_fields() (método *zoop_wrapper.models.token.Token*), 62

validate_custom_fields() (método *zoop_wrapper.models.transaction.InstallmentPlan*), 63

validate_fields() (método *zoop_wrapper.models.base.ZoopObject*), 50

validate_identifiers() (método de classe *zoop_wrapper.models.base.BusinessOrIndividualModel*), 45

validate_type() (método de classe *zoop_wrapper.models.bank_account.BankAccount*), 52

ValidationError, 29

verification_checklist (atributo *zoop_wrapper.models.bank_account.BankAccount*), 51

verification_checklist (atributo *zoop_wrapper.models.card.Card*), 53

VerificationModel (classe em *zoop_wrapper.models.base*), 48

voided (atributo *zoop_wrapper.models.transaction.Transaction*), 66

W

website (atributo *zoop_wrapper.models.seller.Seller*), 57

WITH_INTEREST_MODE (atributo *zoop_wrapper.models.transaction.InstallmentPlan*), 63

Z

ZOOP_KEY (no módulo *zoop_wrapper.constants*), 29

zoop_wrapper.constants (módulo), 29

zoop_wrapper.exceptions (módulo), 29

zoop_wrapper.models.bank_account (módulo), 50

zoop_wrapper.models.base (módulo), 43

zoop_wrapper.models.buyer (módulo), 52

zoop_wrapper.models.card (módulo), 53

zoop_wrapper.models.invoice (módulo), 54

zoop_wrapper.models.seller (módulo), 56

zoop_wrapper.models.token (módulo), 59

zoop_wrapper.models.transaction (módulo), 62

zoop_wrapper.models.utils (módulo), 67

zoop_wrapper.response (módulo), 30

zoop_wrapper.utils (módulo), 30

zoop_wrapper.wrapper (módulo), 31

zoop_wrapper.wrapper.bank_account (módulo), 33

zoop_wrapper.wrapper.base (módulo), 31

zoop_wrapper.wrapper.buyer (módulo), 34

zoop_wrapper.wrapper.card (módulo), 35

zoop_wrapper.wrapper.invoice (módulo), 36

zoop_wrapper.wrapper.seller (módulo), 37

zoop_wrapper.wrapper.transaction (módulo), 40

ZoopObject (classe em *zoop_wrapper.models.base*), 48

ZoopResponse (classe em *zoop_wrapper.response*), 30

ZoopWrapper (classe em *zoop_wrapper.wrapper*), 31