fordev

Release 1.0.5

Matheus Felipe

26 jul. 2023

Index

CAPÍTULO 1

Instalação

Use `pip` e `virtualenv` para instalar **fordev** em seu ambiente. Isso irá separar as dependências do módulo fordev dos módulos instalados globalmente em seu sistema.

1.1 Instalação com pip

Crie e entre no diretório do seu projeto:

```
$ mkdir my_project && cd my_project
```

Crie e ative o ambiente virtual:

```
$ virtualenv venv && source venv/Scripts/activate
```

Nota: Em ambiente linux, use `source venv/bin/activate` para ativar o ambiente virtual.

Instale com `pip`:

```
$ pip install fordev
```

Pronto, pode começar o trabalho ;)

CAPÍTULO 2

Demo

Aqui você encontra uma pequena demonstração de como utilizar o pacote **fordev** no shell interativo.

Vamos socilitar a geração de dados randômicos de uma pessoa com as seguintes características:

- **É do sexo masculino**
- **Tem 25 anos de idade**
- **E mora no estado de SP**

```
>>> from fordev.generators import people
>>> people(sex='M', age=25, uf_code='SP')
{
    'altura': '1,90',
    'bairro': 'Jardim Maria Amélia',
    'celular': '(12) 98401-5301',
    'cep': '12318-110',
    'cidade': 'Jacareí',
    'cor': 'laranja',
    'cpf': '061.632.758-70',
    'data_nasc': '06/12/1995',
    'email': 'bentoyagolorenzogoncalves-72@alcastro.com.br',
    'endereco': 'Rua José Benedito de Oliveira',
    'estado': 'SP',
    'idade': 25,
    'mae': 'Tereza Melissa Priscila',
    'nome': 'Bento Yago Lorenzo Gonçalves',
    'numero': 760,
    'pai': 'Sérgio Guilherme Erick Gonçalves',
    'peso': 88,
    'rg': '23.920.314-8',
    'senha': 'ErOKUUyoml',
    'sexo': 'Masculino',
    'signo': 'Sagitário',
```

(continua na próxima página)

(continuação da página anterior)

```
    'telefone_fixo': '(12) 2844-9806',
    'tipo_sanguineo': 'AB+'
}
```

CAPÍTULO 3

Referência

Aqui você encontra a referência completa do pacote **fordev**, podendo ser usado como referência de uso ou referência de desenvolvimento.

3.1 fordev

Fordev é um pacote que mapeia o site 4devs.com.br via scraping e disponibiliza os geradoradores e validadores de dados como uma interface Python.

Exemplo

Gere dados de 1 pessoa:

```
>>> from fordev.generators import people
>>> people(sex='M', age=25, uf_code='SP')
{
    'altura': '1,90',
    'bairro': 'Jardim Maria Amélia',
    'celular': '(12) 98401-5301',
    'cep': '12318-110',
    'cidade': 'Jacareí',
    'cor': 'laranja',
    'cpf': '061.632.758-70',
    'data_nasc': '06/12/1995',
    'email': 'bentoyagolorenzogoncalves-72@alcastro.com.br',
    'endereco': 'Rua José Benedito de Oliveira',
    'estado': 'SP',
    'idade': 25,
    'mae': 'Tereza Melissa Priscila',
    'nome': 'Bento Yago Lorenzo Gonçalves',
```

(continua na próxima página)

(continuação da página anterior)

```
    'numero': 760,
    'pai': 'Sérgio Guilherme Erick Gonçalves',
    'peso': 88,
    'rg': '23.920.314-8',
    'senha': 'ErOKUUyoml',
    'sexo': 'Masculino',
    'signo': 'Sagitário',
    'telefone_fixo': '(12) 2844-9806',
    'tipo_sanguineo': 'AB+'
}
```

Funcionalidades:

Os módulos e funções internos que contém a API para uso são:

- `fordev.generators` - **O módulo que contém a API para geração de dados.**
  - `certificate(...)` - Gerador de certidões de nascimento, casamento e óbito;
  - `cnh(...)` - Gerador de CNH (Carteira Nacional de Habilitação);
  - `bank_account(...)` - Gerador de contas bancárias;
  - `cpf(...)` - Gerador de CPF (Cadastro de Pessoas Físicas);
  - `pis_pasep(...)` - Gerador de PIS/PASEP (Programa de Integração Social e Programa de Formação do Patrimônio do Servidor Público);
  - `renavam(...)` - Gerador de RENAVAM (Registro Nacional de Veículos Automotores);
  - `vehicle(...)` - Gerador de veículos;
  - `vehicle_brand(...)` - Gerador de marca de veículos;
  - `vehicle_plate(...)` - Gerador de placa de veículos;
  - `cnpj(...)` - Gerador de CNPJ (Cadastro Nacional da Pessoa Jurídica);
  - `rg(...)` - Gerador de RG (Registro Geral) emitido por SSP-SP;
  - `state_registration(...)` - Gerador de Inscrições Estaduais válidas para todos os estados;
  - `voter_title(...)` - Gerador de título de eleitor;
  - `credit_card(...)` - Gerador de dados de cartão de crédito;
  - `people(...)` - Gerador de dados de pessoas (Nome, RG, CPF, CEP e Endereço);
  - `company(...)` - Gerador de dados de empresa (Nome, Razão Social, Inscrição Estadual, CNPJ, CEP e Endereço);
  - `uf(...)` - Gerador de código de UF (Unidade Federativa);
  - `city(...)` - Gerador de cidades do brasil por estado selecionado.
- `fordev.validators` - **O módulo que contém a API para validação de dados.**
  - `is_valid_credit_card(...)` - Verifica se o código de cartão de crédito passado é válido;
  - `is_valid_bank_account(...)` - Verifica se os dados da conta bancária passado é válido;
  - `is_valid_certificate(...)` - Verifica se o código de certidão passado é válido;
  - `is_valid_cnh(...)` - Verifica se o código do CNH passado é válido;
  - `is_valid_cnpj(...)` - Verifica se o código do cnpj passado é válido;

- `is_valid_cpf(...)` - Verifica se o código do cpf passado é válido;
- `is_valid_pis_pasep(...)` - Verifica se o código do PIS/PASEP passado é válido;
- `is_valid_renavam(...)` - Verifica se o código do RENAVAM passado é válido;
- `is_valid_rg(...)` - Verifica se o código do RG passado é válido;
- `is_valid_voter_title(...)` - Verifica se o código do Título de Eleitor passado é válido;
- `is_valid_state_registration(...)` - Verifica se o código da Inscrição Estadual passado é válido.

Todos os demais módulos internos são os responsáveis por manter o pacote **fordev** funcional. Apenas os consulte/use se deseja compreender seu funcionamento e/ou contribuir com o projeto.

Obtenha mais detalhes dos módulos internos na seção abaixo:

3.2 fordev.generators

Este módulo coleta dados aleatórios gerados pelo site 4Devs e disponíbiliza uma API simples para uso.

Use a função `help()` para mais informações:

```
>>> from fordev import generators
>>> help(generators)
Help on module fordev.generators in fordev:

NAME
    fordev.generators

DESCRIPTION
(...)
```

3.2.1 Parâmetros Comuns

Muitas funções do módulo `fordev.generators` contém parâmetros em comum, são eles:

`uf_code: str`
Recebe o Código da **Unidade Federativa** para geração do dado.

Caso não saiba o que é ou não conheça o do estado que necessita, obtenha mais informações em: https://pt.wikipedia.org/wiki/Subdivis%C3%B5es_do_Brasil

`formatting: bool`
Se receber o valor `True`, retorna os dados formatados. Se receber o valor `False`, retorna os dados sem formatação.

`data_only: bool`
Se receber o valor `True`, retorna somente os dados em texto puro. Se receber o valor `False`, retorna um dicionário contendo uma chave `msg` e `data` ou `error` contendo valores correspondentes à nomenclatura de suas chaves.

Sendo assim, sempre que os encontrar, utilize conforme o descrito acima.

3.2.2 Docs de todas funções

fordev.generators.**certificate**(*type_: str = 'I'*, *formatting: bool = True*, *data_only: bool = True*) → str

Gere o código de certidões (birth, wedding, religious wedding and death) aleatórias.

Parâmetros

type – O tipo da certidão para geração do código.

Consulte a doc para verificar as opções suportadas: https://fordev.rtfd.io/pt_BR/latest/fordev/generators.html

Certidões suportadas

`I` = **Indifferent** (Indiferente) <*Padrão*>

`B` = **Birth** (Nascimento)

`W` = **Wedding** (Casamento)

`R` = **Religious** Wedding (Casamento Religioso)

`D` = **Death (Morte)**

Nota: Os tipos de certidões devem ser passados para o parâmetro `type_`.

Exemplo:

```
>>> from fordev.generators import certificate
>>> certificate(type_='B')
```

---

fordev.generators.**cnh**(*data_only: bool = True*) → str

Geração aleatória de CNH (Carteira Nacional de Habilitação).

fordev.generators.**bank_account**(*bank: int = 0*, *uf_code: str = ''*, *data_only: bool = True*) → dict

Gere dados de conta bancária.

Parâmetros

bank – Recebe um valor númerico de 0 a 5 que representa a bandeira do banco da conta bancária a ser gerada.

Consulte a doc para verificar as opções suportadas: https://fordev.rtfd.io/pt_BR/latest/fordev/generators.html

Bandeiras suportadas

`0` = **Aleatório** <*Padrão*>

`1` = **Banco do Brasil**

`2` = **Bradesco**

`3` = **Citibank**

`4` = **Itaú**

`5` = **Santander**

Nota: O valor númerico que representa a bandeira do banco deve ser passada para o parâmetro `bank`.

Exemplo:

```
>>> from fordev.generators import bank_account
>>> bank_account(bank=2)  # Banco Bradesco
```

---

fordev.generators.**cpf**(*uf_code: str = ''*, *formatting: bool = True*, *data_only: bool = True*) → str

Gere o código de um CPF(Cadastro de Pessoas Físicas) aleatório.

fordev.generators.**pis_pasep**(*formatting: bool = True*, *data_only: bool = True*) → str

Gere o código do PIS/PASEP aleatório.

fordev.generators.**renavam**(*data_only: bool = True*) → str

Gere o código do RENAVAM(Registro Nacional de Veículos Automotores) aleatório.

fordev.generators.**vehicle**(*brand_code: int = 0*, *uf_code: str = ''*, *formatting: bool = True*, *data_only: bool = True*) → dict

Gere dados de veículo aleatório.

Parâmetros

brand – Recebe um valor númerico de 0 a 87 que representa a marca do carro para geração dos dados aleatórios.

Consulte a doc para verificar as opções suportadas: https://fordev.rtfd.io/pt_BR/latest/fordev/generators.html

Marcas suportadas

0 = **Aleatório** <*Padrão*>

1 = **Acura**

2 = **Agrale**

3 = **Alfa Romeo**

4 = **AM Gen**

5 = **Asia Motors**

6 = **ASTON MARTIN**

7 = **Audi**

8 = **BMW**

9 = **BRM**

10 = **Buggy**

11 = **Bugre**

12 = **Cadillac**

13 = **CBT Jipe**

14 = **CHANA**

15 = **CHANGAN**

16 = **CHERY**

17 = **Chrysler**

18 = **Citroen**

19 = **Cross Lander**

20 = **Daewoo**

21 = **Daihatsu**

22 = **Dodge**

23 = **EFFA**

24 = **Engesa**

25 = **Envemo**

26 = **Ferrari**

27 = **Fiat**

28 = **Fibravan**

29 = **Ford**

30 = **FOTON**

31 = **Fyber**

32 = **GEELY**

33 = **GM - Chevrolet**

34 = **GREAT WALL**

35 = **Gurgel**

36 = **HAFEI**

37 = **Honda**

38 = **Hyundai**

39 = **Isuzu**

40 = **JAC**

41 = **Jaguar**

42 = **Jeep**

43 = **JINBEI**

44 = **JPX**

45 = **Kia Motors**

46 = **Lada**

47 = **LAMBORGHINI**

48 = **Land Rover**

49 = **Lexus**

50 = **LIFAN**

51 = **LOBINI**

52 = **Lotus**

53 = **Mahindra**

54 = **Maserati**

55 = **Matra**

56 = **Mazda**

57 = **Mercedes-Benz**

58 = **Mercury**

59 = **MG**

60 = **MINI**

61 = **Mitsubishi**

62 = **Miura**

63 = **Nissan**

64 = **Peugeot**

65 = **Plymouth**

66 = **Pontiac**

67 = **Porsche**

68 = **RAM**

69 = **RELY**

70 = **Renault**

71 = **Rolls-Royce**

72 = **Rover**

73 = **Saab**

74 = **Saturn**

75 = **Seat**

76 = **SHINERAY**

77 = **smart**

78 = **SSANGYONG**

79 = **Subaru**

80 = **Suzuki**

81 = **TAC**

82 = **Toyota**

83 = **Troller**

84 = **Volvo**

85 = **VW - VolksWagen**

86 = **Wake**

87 = **Walk**

Nota: O valor númerico que representa a marca do veículo deve ser passada para o parâmetro `brand_code`.

Exemplo:

```
>>> from fordev.generators import vehicle
>>> vehicle(brand_code=26)  # Ferrari
```

---

fordev.generators.**vehicle_brand**(*n: int = 1*, *data_only: bool = True*) → list

Obtenha o nome de marca(s) de veículo(s).

Parâmetros

n – Recebe o número de marcas de veículos a ser gerado. O valor mínimo é 1 e o máximo é 87.

fordev.generators.**vehicle_plate**(*uf_code: str = ''*, *formatting: bool = True*, *data_only: bool = True*) → str

Gere o código da placa de veículo aleatório.

fordev.generators.**cnpj**(*formatting: bool = True*, *data_only: bool = True*) → str

Gere o código do CNPJ(Cadastro Nacional da Pessoa Jurídica) aleatório.

fordev.generators.**rg**(*formatting: bool = True*, *data_only: bool = True*) → str

Gere o código do RG(Registro Geral) aleatório, emitido por SSP-SP.

fordev.generators.**state_registration**(*uf_code: str = 'SP'*, *formatting: bool = True*, *data_only: bool = True*) → str

Gere o código de registro de estado aleatório.

fordev.generators.**voter_title**(*uf_code: str*, *data_only: bool = True*) → str

Gere o código do título de eleitor aleatório, conforme o UF especificado.

fordev.generators.**credit_card**(*bank: int = 0*, *formatting: bool = True*, *data_only: bool = True*) → dict

Gere dados de cartão de crédito aleatório.

Parâmetros

bank – Recebe um valor númerico de 0 a 10 representando a bandeira do cartão de crédito a ser gerado.

Consulte a doc para verificar as opções suportadas: https://fordev.rtfd.io/pt_BR/latest/fordev/generators.html

Bandeiras suportadas

`0` = **Aleatório** <*Padrão*>

`1` = **MasterCard**

`2` = **Visa 16 Dígitos**

`3` = **American Express**

`4` = **Diners Club**

`5` = **Discover**

`6` = **enRoute**

`7` = **JCB**

`8` = **Voyager**

`9` = **HiperCard**

`10` = **Aura**

Nota: O valor númerico que representa a bandeira do cartão de crédito deve ser passada para o parâmetro `bank`.

Exemplo:

```
>>> from fordev.generators import credit_card
>>> credit_card(bank=3)  # American Express
```

---

fordev.generators.**people**(*n: int = 1*, *sex: str = 'R'*, *age: int = 0*, *uf_code: str = ''*, *formatting: bool = True*, *data_only: bool = True*) → str

Gere dados de pessoa(s) aleatório(s)

Parâmetros

- **n** – O número de pessoas a ter dados gerados. O mínimo é 1 e o máximo é 30.
- **sex** – Uma string representando o sexo da pessoa para geração dos dados.

  Consulte a doc para verificar as opções suportadas: https://fordev.rtfd.io/pt_BR/latest/fordev/generators.html
- **age** – A idade da pessoa para geração dos dados. A idade mínima é 18 e a máxima é 80.

Opções de sexo

`R` = **Random** <*Padrão*>

`M` = **Male**

`F` = **Feminine**

Nota: A opção de sexo deve ser passada para o parâmetro `sex`.

Exemplo:

```
>>> from fordev.generators import people
>>> people(sex='M')  # Male (Masculino)
```

---

fordev.generators.**company**(*uf_code: str = 'SP'*, *age: int = 1*, *formatting: bool = True*, *data_only: bool = True*) → dict

Gere dados de companhia (empresa/organização) aleatório.

Parâmetros

age – Representa o tempo de existência da companhia (a idade da companhia).

fordev.generators.**uf**(*n: int = 1*, *data_only: bool = True*) → list

Gere o código da UF(Unidade Federativa) aleatório.

Parâmetros

n – O número de UF's para geração do dado. O mínimo é 1 e o máximo é 27.

fordev.generators.**city**(*uf_code: str = 'SP'*, *data_only: bool = True*) → list

Obtenha as cidades do UF especificado.

3.3 fordev.validators

Este módulo válida os dados utilizando o site 4Devs e disponibiliza uma API simples para uso.

Use a função `help()` para mais informações:

```
>>> from fordev import validators
>>> help(validators)
Help on module fordev.validators in fordev:

NAME
    fordev.validators

DESCRIPTION
(...)
```

3.3.1 Parâmetros Comuns

Muitas funções do módulo `fordev.validators` contém parâmetros em comum, são eles:

`data_only: bool`
: Se receber o valor `True`, retorna somente os dados em texto puro. Se receber o valor `False`, retorna um dicionário contendo uma chave `msg` e `data` ou `error` contendo valores correspondentes à nomenclatura de suas chaves.

Sendo assim, sempre que o encontrar, utilize conforme o descrito acima.

3.3.2 Docs de todas funções

`fordev.validators.`**`is_valid_credit_card`**(*flag: int*, *credit_card_code: str*, *data_only: bool = True*) → bool

Verifique se o código do cartão de crédito é válido.

Parâmetros

- **`flag`** – A bandeira do cartão de crédito que deseja validar o código.

  Consulte a doc para verificar as opções suportadas: https://fordev.rtfd.io/pt_BR/latest/fordev/generators.html

- **`credit_card_code`** – O código do cartão de crédito para verificação.

Bandeiras suportadas

1 = **MasterCard**

2 = **Visa 16 Dígitos**

3 = **Visa Electron**

4 = **American Express**

5 = **Diners Club**

6 = **Discover**

7 = **enRoute**

8 = **JCB**

9 = **Maestro**

10 = **Solo**

11 = **Switch**

12 = **Laser**

Nota: O valor númerico que representa a bandeira do cartão de crédito deve ser passada para o parâmetro `flag`.

Exemplo:

```
>>> from fordev.validators import is_valid_credit_card
>>> is_valid_credit_card(flag=3)  # Visa Electron
```

---

fordev.validators.**is_valid_bank_account**(*bank: int*, *agency: str*, *account: str*, *data_only: bool = True*) → bool

Verifique se os dados da conta bancária são válidos.

Parâmetros

- **bank** – A bandeira do banco da conta bancária que deseja validar os dados.

  Consulte a doc para verificar as opções suportadas: https://fordev.rtfd.io/pt_BR/latest/fordev/generators.html

- **agency** – O código da agência bancária para verificação.
- **account** – O código da conta bancária para verificação.

Bandeiras suportadas

1 = **Banco do Brasil**

2 = **Bradesco**

3 = **Citibank**

4 = **Itaú**

5 = **Santander**

Nota: O valor númerico que representa a bandeira do banco deve ser passada para o parâmetro bank.

Exemplo:

```
>>> from fordev.validators import is_valid_bank_account
>>> is_valid_bank_account(bank=4)  # Itaú
```

---

fordev.validators.**is_valid_certificate**(*certificate_code: str*, *data_only: bool = True*) → bool

Verifique se o código da Certidão (birth, wedding, religious wedding and death) é válido.

Parâmetros

certificate_code – O código da certidão para verificação.

fordev.validators.**is_valid_cnh**(*cnh_code: str*, *data_only: bool = True*) → bool

Verifique se o código da CNH é válido.

Parâmetros

cnh_code – O código da CNH para verificação.

fordev.validators.**is_valid_cnpj**(*cnpj_code: str*, *data_only: bool = True*) → bool

Verifique se o código do CNPJ é válido.

Parâmetros
cnpj_code – O código CNPJ para verificação.

fordev.validators.**is_valid_cpf**(*cpf_code: str*, *data_only: bool = True*) → bool

Verifique se o código do CPF é válido.

Parâmetros
cpf_code – O código do CPF para verificação.

fordev.validators.**is_valid_pis_pasep**(*pis_pasep_code: str*, *data_only: bool = True*) → bool

Verifique se o código do PIS/PASEP é válido.

Parâmetros
pis_pasep_code – O código PIS/PASEP para verificação.

fordev.validators.**is_valid_renavam**(*renavam_code: str*, *data_only: bool = True*) → bool

Verifique se o código do RENAVAM é válido.

Parâmetros
renavam_code – O código do RENAVAM para verificação.

fordev.validators.**is_valid_rg**(*rg_code: str*, *data_only: bool = True*) → bool

Verifique se o código do RG é válido.

Parâmetros
rg_code – O código do RG para verificação.

fordev.validators.**is_valid_voter_title**(*voter_title_code: str*, *data_only: bool = True*) → bool

Verifique se o código do título de eleitor é válido.

Parâmetros
voter_title_code – O código do título de eleitor para verificação.

fordev.validators.**is_valid_state_registration**(*uf_code: str*, *state_registration_code: str*, *data_only: bool = True*) → bool

Verifique se o código do registro estadual é válido.

Parâmetros

- **uf_code** – O código UF(Unidade Federativa) do estado que pertence o registro estadual.

  Mais informações: https://pt.wikipedia.org/wiki/Subdivis%C3%B5es_do_Brasil
- **state_registration_code** – O código do registro estadual para verificação.

3.4 fordev.core

Este módulo é o core para criar e manipular requests para a API do site 4devs.

fordev.core.**_create_headers**(*content_length: int*, *referer: str*) → dict

Gere o header para ser enviado em requests HTTP para o site 4devs.

Parâmetros

- **content_length** – Indica o tamanho do entity-body, em bytes, enviados no header para o destinatário.

- **referer** – Referência a ação a ser executada pela API do site 4devs. Pode-se interpretar como o endpoint do serviço a ser disponibilizado.

fordev.core.**fordev_request**(*content_length: int*, *referer: str*, *payload: dict*) → dict

Cria uma request HTTP a API do site 4devs e retorna seu conteúdo em formato de dicionário.

Parâmetros

- **content_length** – Indica o tamanho do entity-body, em bytes, enviados no header para o destinatário.
- **referer** – Referência a ação a ser executada pela API do site 4devs. Pode-se interpretar como o endpoint do serviço a ser disponibilizado.
- **payload** – Um dicionário de dados contendo a ação e outros dados solicitados pela API.

3.5 fordev.filters

Este módulo é usado para filtrar os dados contidos na estrutura HTML retornada pela API do site 4devs e converter para uma estrutura de dicionário Python.

fordev.filters.**data_format**(*data_only: bool*, *data_dict: dict*) → dict

Filtra os dados conforme especificado.

fordev.filters.**filter_bank_account_info**(*html: str*) → dict

Filtra dados de conta bancária contidos na estrutura HTML.

Parâmetros

html – Uma estrutra HTML contendo dados de conta bancária retornados pela API do site 4devs.

fordev.filters.**filter_vehicle_info**(*html: str*) → dict

Filtra dados de veículo contido na estrutura HTML.

Parâmetros

html – Uma estrutra HTML contendo dados de veículo retornados pela API do site 4devs.

fordev.filters.**filter_credit_card_info**(*html: str*) → dict

Filtra dados de cartão de crédito contidos na estrutura HTML.

Parâmetros

html – Uma estrutra HTML contendo dados de cartão de crédito retornados pela API do site 4devs.

fordev.filters.**filter_company_info**(*html: str*) → dict

Filtra dados de companhia (empresa/organização) contidos na estrutura HTML.

Parâmetros

html – Uma estrutra HTML contendo dados de companhia (empresa/organização) retornados pela API do site 4devs.

fordev.filters.**filter_city_name**(*html: str*) → list

Filtra dados de cidade contidos na estrutura HTML.

Parâmetros

html – Uma estrutra HTML contendo dados de uma cidade retornados pela API do site 4devs.

CAPÍTULO 4

Contribuições

Toda contribuição é super bem-vinda!

Abaixo mostro com o que você pode contribuir:

- Encontrou algum bug, quer propor uma nova funcionalidade ou conversar sobre o projeto? Abra uma Issue e descreve seu caso.
- Existe uma issue aberta e você quer resolve-la, quer implementar uma nova funcionalidade ou melhorar a documentação? Faça suas adições e me envie um *Pull Request*
- Gostou do projeto, mas não quer ou ainda não consegue contribuir com ele? Considere deixar uma estrela no Github.

Obrigado pelo interesse em colaborar de alguma forma com o projeto

CAPÍTULO 5

Aviso Legal

Nota: O aviso abaixo é uma adaptação para utilização no repositório. Confira os termos de uso oficial do site 4Devs em: Termos de Uso

Todo os dados são gerados de forma randômica, respeitando as regras de criação de cada tipo de dado.

Todo os dados gerados são para fins informativos e utilizados para auxiliar estudantes, programadores, analistas e testadores no desenvolvimento de softwares que necessitem de tais dados. Não devem ser considerados completos, atualizados, e não se destinam a ser utilizado no lugar de uma consulta jurídica, médica, financeira, ou de qualquer outro profissional. Todo e qualquer risco da utilização dos dados disponibilizados atráves do módulo **Fordev** é assumido pelo próprio usuário.

CAPÍTULO 6

Licença

Fordev utiliza a *licença MIT* em todo seu código, confira suas condições abaixo:

```
MIT License

Copyright (c) 2020 Matheus Felipe <matheusfelipeog@protonmail.com> or <github.com/
↪matheusfelipeog>

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

Está é a documentação oficial e completa do módulo **Fordev**, aqui você encontrará exemplos e uma explicação individual de cada função geradora e validadora de dados disponibilizados e mapeados no site 4Devs.

CAPÍTULO 7

Objetivo

O site 4Devs disponibiliza diversas funcionalidades muito úteis para um desenvolvedor utilizar em seus projetos que necessitam de dados randômicos válidos e outras peculiaridades, tais como: dados dos principais documentos pessoais do brasil (CPF, CNPJ, CNH etc), dados bancários, dados de cartões de crédito, dados completos de pessoas (nome, idade, documentos, endereço etc) e muitos outros geradores de dados. Porém, até o momento, não possui uma interface/API pública para utiliza-los diretamente no código da aplicação em desenvolvimento, assim, sendo necessário ir buscar tais dados diretamente no site.

Fordev foi construído para resolver esse problema, disponibilizando um módulo de fácil uso que mapeia todo o site 4Devs usando técnicas de scraping, de modo que seja possível obter todos recursos disponíveis no site em um módulo Python.

CAPÍTULO 8

Índices e tabela

- genindex
- modindex
- search

Índice de Módulos Python

f

Índice

Símbolos

B

C

D

F

I

M

P

R

S

U

V