

Aluno: Gustavo Guerreiro

Turma: 302

Print da definição classe do trabalho:

```
medico.py  model.py X
BackEnd > model.py
1  """Este módulo contém a classe que representa o Usuario
2
3  Autor: Gustavo Guerreiro, Johannes Wachholz José e Lucas Vargas
4  """
5
6  from config import *
7
8  class Usuario(db.Model):
9      # atributos do usuário
10     id = db.Column(db.Integer, primary_key=True)
11     estado = db.Column(db.String(254))
12     cidade = db.Column(db.String(254))
13     endereco = db.Column(db.String(254))
14     complemento = db.Column(db.String(254))
15     cep = db.Column(db.String(254))
16     telefone = db.Column(db.String(254))
17     email = db.Column(db.String(254))
18     senha = db.Column(db.String(254))
19     data_surgimento = db.Column(db.String(254))
20
21     # driscriminador
22     type = db.Column(db.String(50))
23
24     # definições de mapeamento da classe mãe
25     __mapper_args__ = {
26         'polymorphic_identity': 'usuario',
27         'polymorphic_on': type
28     }
29
30     def __str__(self):
31         return f'id = {self.id}, {self.estado}, {self.cidade}, '+\
32             f'{self.endereco}, {self.complemento}, {self.cep}, '+\
33             f'{self.telefone}, {self.email}, {self.email}, '+\
34             f'{self.data_surgimento}'
35
36     def json(self):
37         return {
38             "id" : self.id,
```

```

def json(self):
    return {
        "id" : self.id,
        "estado" : self.estado,
        "cidade" : self.cidade,
        "endereco" : self.endereco,
        "complemento" : self.complemento,
        "cep" : self.cep,
        "telefone" : self.telefone,
        "email" : self.email,
        "senha" : self.senha,
        "data_surgimento" : self.data_surgimento
    }

```

Print do código de teste da classe do trabalho:

```

# para testar
if __name__ == "__main__":
    # comando para remover arquivo banco de dados caso já exista
    if os.path.exists(arquivobd):
        os.remove(arquivobd)

    # comando para criar tabela (classe usuario)
    db.create_all()

    # criar objetos (na memória, sem persistência)
    p1 = Usuario(estado="SC", cidade="Blumenau", endereco="endereço teste",
        complemento="Alto", cep="89037-255", telefone="992922070",
        email="lucasv@email.com", senha="123", data_surgimento="10/10/2020")
    p2 = Usuario(estado="PR", cidade="Itaporobo", endereco="endereço teste",
        complemento="vermelho", cep="32076-454", telefone="85920132",
        email="sabrino@email.com", senha="543", data_surgimento="21/40/2019")

    # para tornar os objetos persistentes
    db.session.add(p1)
    db.session.add(p2)
    db.session.commit()

    # exibir pessoa
    print(p1.json())
    print(p2)

```

Prints do código da definição da classe nova:

medico.py

BackEnd > medico.py

```
1  """Este módulo contém a classe que representa um médico
2
3  No arquivo encontra-se a implementação de uma classe: Medico.
4
5  Autor: Gustavo Guerreiro.
6  """
7  from config import *
8  from model import Usuario
9  from entidade import Entidade
10
11  class Medico(Usuario):
12      """Classe que representa um médico no site
13      """
14      # atributos exclusivos do médico
15      id = db.Column(db.Integer, db.ForeignKey('usuario.id'), primary_key=True)
16      nome_medico = db.Column(db.String(254))
17      area_atuacao = db.Column(db.String(254))
18      cpf_medico = db.Column(db.String(254))
19      sexo_medico = db.Column(db.String(254))
20
21      # atributo de chave estrangeira
22      id_entidade = db.Column(db.Integer, db.ForeignKey('entidade.id'), nullable=False)
23
24      # atributo de relacionamento
25      entidade = db.relationship("Entidade", foreign_keys=[id_entidade])
26
27      # definindo identidade polimórfica que ficará armazenada na classe pai
28      # no campo type
29      __mapper_args__ = {
30          'polymorphic_identity' : 'medico',
31      }
32
33      def __str__(self):
34          """Feita para fazer o print imprimir o dicionário contido em Medico
35
36          Returns:
37              (str): string adicionando a área de atuação do médico ao dicionário
38          """
```

```
32
33      def __str__(self):
34          """Feita para fazer o print imprimir o dicionário contido em Medico
35
36          Returns:
37              (str): string adicionando a área de atuação do médico ao dicionário
38          """
39          return super().__str__() + f', {self.nome_medico}, ' + \
40              f'{self.cpf_medico}, {self.sexo_medico}, ' + \
41              f'{self.area_atuacao}, {self.id_entidade}, {self.entidade}'
42
43      def json(self):
44          """Adiciona área_atuacao ao return do json do Usuario
45
46          Returns:
47              (dict): dicionário json com a área de atuação inclusa
48          """
49          return super().json() | {
50              "nome_medico" : self.nome_medico,
51              "area_atuacao" : self.area_atuacao,
52              "cpf_medico" : self.cpf_medico,
53              "sexo_medico" : self.sexo_medico,
54              "id_entidade" : self.id_entidade,
55              "entidade" : self.entidade.json(),
56          }
57
```

Print do código do teste:

```
57
58 # parte de teste da classe
59 if __name__ == "__main__":
60     # se um banco de dados já existir, o comando a seguir o apaga
61     if os.path.exists(arquivobd):
62         os.remove(arquivobd)
63
64     # comando cria uma tabela
65     db.create_all()
66
67     # criando instancia de entidade para fazer o teste do médico
68     entidade_teste = Entidade(estado="SC", cidade="Blumenau",
69     endereco="Alguma Casa", complemento="Alto", cep="89037-255",
70     telefone="992922070", email="lucasv@email.com", senha="123",
71     data_surgimento="10/10/2020", nome_fantasia="bananinha",
72     razao_social="coca-cola", numero_funcionarios="40",
73     tipo_instituicao="médica", cnpj="32323232")
74
75     # comandos para testar o módulo (criar objetos na memória sem persistência)
76     test1 = Medico(estado="RS", cidade="Porto Triste",
77     endereco="Algum Apartamento", complemento="Baixo", cep="55555-955",
78     telefone="47888888888", email="gustavog@email.com", senha="321",
79     data_surgimento="11/11/2011", nome_medico="Jorge", area_atuacao="Urologia",
80     sexo_medico="Masculino", cpf_medico="989.654.258-89", id_entidade="1")
81
82     # torna os objetos persistentes
83     db.session.add(entidade_teste)
84     db.session.add(test1)
85     db.session.commit()
86
87     # exibe o médico
88     print(test1.json())
89     print("-----")
90     # acessa o nome fantasia da entidade por meio do médico
91     print(test1.entidade.nome_fantasia)
```

Print do código rodando no terminal:

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Experimente a nova plataforma cruzada PowerShell https://aka.ms/pscore6

PS C:\Users\Usuário\Documents\Github\SitePlataformaMedica> & C:/Users/Usuário/AppData/Local/Programs/Python/Python39/python.exe c:/Users/Usuário/Documents/Github/SitePlataformaMedica/BackEnd/medico.py
{"id": 2, "estado": "RS", "cidade": "Porto Triste", "endereco": "Algum Apartamento", "complemento": "Baixo", "cep": "55555-955", "telefone": "47888888888", "email": "gustavog@email.com", "senha": "321", "data_surgimento": "11/11/2011", "nome_medico": "Jorge", "area_atuacao": "Urologia", "cpf_medico": "989.654.258-89", "sexo_medico": "Masculino", "id_entidade": 1, "entidade": {"id": 1, "estado": "SC", "cidade": "Blumenau", "endereco": "Alguma Casa", "complemento": "Alto", "cep": "89037-255", "telefone": "992922070", "email": "lucasv@email.com", "senha": "123", "data_surgimento": "10/10/2020", "nome_fantasia": "bananinha", "razao_social": "coca-cola", "numero_funcionarios": "40", "tipo_instituicao": "médica", "cnpj": "32323232"}}
-----
bananinha
PS C:\Users\Usuário\Documents\Github\SitePlataformaMedica>
```

Print código rodando json:

(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\Usuário>curl localhost:5000/listar_usuarios

```
[
  {
    "cep": "89037-255",
    "cidade": "Blumenau",
    "cnpj": 32323232,
    "complemento": "Alto",
    "data_surgimento": "10/10/2020",
    "email": "lucasv@email.com",
    "endereco": "Alguma Casa",
    "estado": "SC",
    "id": 1,
    "nome_fantasia": "bananinha",
    "numero_funcionarios": "40",
    "razao_social": "coca-cola",
    "senha": "123",
    "telefone": "992922070",
    "tipo_instituicao": "m\u00e9dica"
  },
  {
    "area_atuacao": "Urologia",
    "cep": "55555-955",
    "cidade": "Porto Triste",
    "complemento": "Baixo",
    "cpf_medico": "989.654.258-89",
    "data_surgimento": "11/11/2011",
    "email": "gustavog@email.com",
    "endereco": "Algum Apartamento",
    "entidade": {
      "cep": "89037-255",
      "cidade": "Blumenau",
      "cnpj": 32323232,
      "complemento": "Alto",
      "data_surgimento": "10/10/2020",
      "email": "lucasv@email.com",
      "endereco": "Alguma Casa",
      "estado": "SC",
      "id": 1,
      "nome_fantasia": "bananinha",
      "numero_funcionarios": "40",
      "razao_social": "coca-cola",
      "senha": "123",
      "telefone": "992922070",
      "tipo_instituicao": "m\u00e9dica"
    },
    "estado": "RS",
    "id": 2,
    "id_entidade": 1,
    "nome_medico": "Jorge",
    "senha": "321",
    "sexo_medico": "Masculino",
    "telefone": "47888888888"
  }
]
```

Link para o github: <https://github.com/GuerreiroG/SitePlataformaMedica>

O código fonte encontra-se na pasta "BackEnd"