**PROJETO INTEGRADOR 3º SEMESTRE 2023**

**Grupo:**

**Daniel de Godoy Carolino**

**Carlos De Gasperi**

**Chiara Maneo**

**GitHub projeto:**

https://github.com/DanielCarolino89/Projeto-Interdisciplinar-3Semestre

**Bando de Dados Não Relacional**

Nosso projeto em PYTHON utilizamos o framework DJANGO juntamente com o banco não relacional MONGODB.

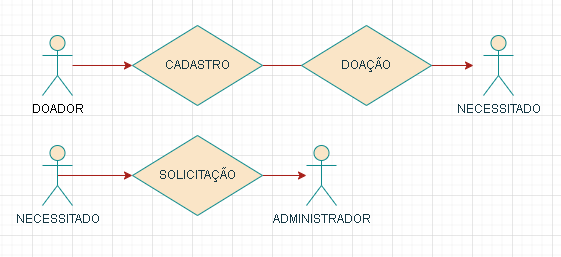
A escolha do banco devido ao modelo de dados flexível para lidar com dados de diferentes formatos e estruturas.

**Projeto:**

Nosso projeto foi baseado no tema da ODS2, doações de alimentos para pessoas que necessitam. Alimentos que estão ainda bons para consumo mais não estão “apresentáveis” para a venda. Pessoas Físicas, Jurídicas e Empresas fazem o cadastro e no programa informa o que vai doar, a quantidade e a data de validade de cada produto; e pessoas que precisam de ajuda fazem o cadastro que gera um arquivo para administração do site.

**História de relacionamentos:**

O usuário entra no site e efetua o cadastro, fazendo o cadastro é redirecionado para página de inserção de alimentos, podendo adicionar o tipo de item como cesta básica, tomate, arroz e etc, a quantidade em unidade ou quilograma e a data possível de vencimento. Nas configurações de usuário ele pode alterar, nickname, e-mail ou senha; também na mesma aba alterar os dados dele. Em relatório de doações exibe valor total de doações, a média. Selecionando o alimento doado exibe a quantidade de produtos com o mesmo nome, quantidade de produtos doados com mesmo nome.



**Dados coletados para inserir ao banco:**

**Pessoas a doar:**

Username, email, password, CNPJ, empresa, cpf, rg, orgao, nome, idade, endereço, número, bairro, estado, telefone, celular, sobre, encontrou, declaração.

**Pessoas que precisam de ajuda:**

nome, idade, endereço, número, bairro, estado, telefone, celular, renda, pessoas, sobre, encontrou, declaração.

**Doações:**

Item, quantidade, medida, vencimento.

**Código:**

**CRUD:**

|  |
| --- |
| **class ConexaoDATABASE:**  **def getConnection(self):**  return pymongo.MongoClient("mongodb://localhost:27017/odsdatabase")  **class CRUD\_PessoaCadastro:**  def \_\_init\_\_(self,connection) -> None:  self.client = connection.getConnection()  self.db = self.client['odsdatabase']    **def insert(self, \*\*kwargs):**  data = {}  collection = self.db['cd\_PessoaCadastro']  for key, value in kwargs.items():  data[key] = value    collection.insert\_one(data)  **def delete(self, \*\*kwargs):**  collection = self.db['cd\_PessoaCadastro']  query = {}  for key, value in kwargs.items():  query[key] = value  collection.delete\_one(query)    **def update(self, \*\*kwargs):**  collection = self.db['cd\_PessoaCadastro']  query = {}  update\_data = {}  for key, value in kwargs.items():  if key == "\_id":  query[key] = value  else:  update\_data[key] = value  collection.update\_one(query, {"$set": update\_data})    **def view(self, \*\*kwargs):**  collection = self.db['cd\_PessoaCadastro']  query = {}  for key, value in kwargs.items():  query[key] = value  documents = collection.find(query)  for document in documents:  print(document)  **class CRUD\_PessoaApoio:**  def \_\_init\_\_(self,connection) -> None:  self.client = connection.getConnection()  self.db = self.client['odsdatabase']    **def insert(self, \*\*kwargs):**  data = {}  collection = self.db['cd\_PessoaApoio']  for key, value in kwargs.items():  data[key] = value    collection.insert\_one(data)  **def delete(self, \*\*kwargs):**  collection = self.db['cd\_PessoaApoio']  query = {}  for key, value in kwargs.items():  query[key] = value  collection.delete\_one(query)    **def update(self, \*\*kwargs):**  collection = self.db['cd\_PessoaApoio']  query = {}  update\_data = {}  for key, value in kwargs.items():  if key == "\_id":  query[key] = value  else:  update\_data[key] = value  collection.update\_one(query, {"$set": update\_data})    **def view(self, \*\*kwargs):**  collection = self.db['cd\_PessoaApoio']  query = {}  for key, value in kwargs.items():  query[key] = value  documents = collection.find(query)  for document in documents:  print(document)    **class CRUD\_ItemsDoacao:**  def \_\_init\_\_(self,connection) -> None:  self.client = connection.getConnection()  self.db = self.client['odsdatabase']    **def insert(self, \*\*kwargs):**  data = {}  collection = self.db['cd\_ItemsDoacao']  for key, value in kwargs.items():  data[key] = value    collection.insert\_one(data)  **def delete(self, \*\*kwargs):**  collection = self.db['cd\_ItemsDoacao']  query = {}  for key, value in kwargs.items():  query[key] = value  collection.delete\_one(query)    **def update(self, \*\*kwargs):**  collection = self.db['cd\_ItemsDoacao']  query = {}  update\_data = {}  for key, value in kwargs.items():  if key == "\_id":  query[key] = value  else:  update\_data[key] = value  collection.update\_one(query, {"$set": update\_data})    **def view(self, \*\*kwargs):**  collection = self.db['cd\_ItemsDoacao']  query = {}  for key, value in kwargs.items():  query[key] = value  documents = collection.find(query)  for document in documents:  print(document) |

**PARA CONSULTAS BDNR:**

|  |
| --- |
| **def Total\_doacoes(self):**  collection = self.db['cd\_ItemsDoacao']  pipeline = [  {  '$group': {  '\_id': None,  'totalDoacoes': {'$sum': 1}  }  }  ]  result = next(collection.aggregate(pipeline), None)  total\_doacoes = result['totalDoacoes'] if result else 0  return total\_doacoes  **def doacoes\_por\_item(self, item):**  collection = self.db['cd\_ItemsDoacao']  pipeline = [  {  '$match': {'item': item}  },  {  '$group': {  '\_id': None,  'doacoesItem': {'$sum': 1}  }  }  ]  result = next(collection.aggregate(pipeline), None)  doacoes\_item = result['doacoesItem'] if result else 0  return doacoes\_item  **def quantidade\_total\_por\_item(self, item):**  collection = self.db['cd\_ItemsDoacao']  cursor = collection.aggregate([  {  '$match': {'item': item}  },  {  '$group': {  '\_id': {'item': '$item'},  'totalQuantidade': {  '$sum': {  '$convert': {  'input': '$quantidade',  'to': 'int',  'onError': 0,  'onNull': 0  }  }  }  }  }  ])  result = next(cursor, None)  return result['totalQuantidade'] if result else 0  **def porcentagem\_total\_doacoes(self):**  collection = self.db['cd\_ItemsDoacao']  pipeline = [  {  '$group': {  '\_id': None,  'totalDoacoes': {'$sum': 1}  }  }  ]  result = next(collection.aggregate(pipeline), None)  total\_doacoes = result['totalDoacoes'] if result else 0  porcentagem\_doacoes = total\_doacoes \* 0.10 if total\_doacoes > 0 else 0  return porcentagem\_doacoes  **def valor\_medio\_total(self):**  collection = self.db['cd\_ItemsDoacao']  pipeline = [  {  '$group': {  '\_id': None,  'totalDoacoes': {'$sum': 1}  }  }  ]  result = next(collection.aggregate(pipeline), None)  total\_doacoes = result['totalDoacoes'] if result else 0  porcentagem\_doacoes = total\_doacoes \* 0.10 if total\_doacoes > 0 else 0  return porcentagem\_doacoes |