Relatório da Modelagem do Banco de Dados

Descrição (Linguagens e Metodologias usadas):

O “Connect Food” utilizará como linguagem de programação o “Python” e utiliza o banco de dados não relacional “MongoDB”. O padrão de projeto é aquele utilizado com padrão do framework “Django” o “MTV”.

Justificativa (Porque foram utilizadas tais tecnologias)

Python: Essa linguagem de programação é simplista em relação a conexão e manutenção de banco de dados, além de não ser verbosa e ter já estabelecida estruturas, como a lista.

MongoDB: O mongoDB é estritamente necessário tendo em vista que o volume de dados seria imenso já que qualquer pessoa poderia doar. Além disso com um banco de dados não relacional a chance de perder dados quando houver simultaneidade de ocorrências no banco é mínima.

Django: O Django será responsável pela padronização do projeto, além dele oferecer um servidor local para que o template do site seja feito.

Sobre o sistema: Um dos principais desafios do PI foi como inserir um banco de dados não relacional a um projeto que teria cadastros, login e entre outras possíveis tabelas, não faria sentido no âmbito do projeto, pois a conexão ficaria mais lenta e complexa, por isso, abrimos uma discussão no GitHub de elaboração de um projeto que seria mais eficaz num banco não relacional. Portanto tivemos a ideia de criar apenas uma coleção com os dados de receptor quanto de doador.

Por que usar o MongoDB? -> O projeto foi pensado para o mongo, a estrutura gerada em uma coleção é bem mais rápida que usar sistema de login e senha com várias tabelas, além de adaptar o sistema para que o usuário já tenha acesso a doações tornando-o mais prático tanto ao sistema quanto ao usuário

Sobre o banco de dados: Os dados obtidos serão:

Dados do doador, alimento e receber estão na mesma tabela:


```
{
  "nome": "Mercado Luz",
  "cpf": "48575845855",
  "cnpj": null,
  "email": "mercadoluz@gmail.com",
  "telefone": "196565656565",
  "endereco": "Av Leme, nº 456, Jardim Leme",
  "horario": "Das 15h à 18h",
  "alimento_id": 1,
  "categoria": "Fruta",
  "alimento": "Maçã",
  "quantidade": 60,
  "quant_medida": 80,
  "validade": {
    "$date": "2024-06-10T00:00:00.000Z"
  },
  "nome_recebedor": "Abrigo Luz",
  "quantidade_retirou": 5,
  "cnpj_recebedor": "121424244245",
  "email_recebedor": "abrigo@luz.com"
}
```


Quando o produto é doado os dados do receber fica como null, assim que o receber retira o produto é feito um update no banco acrescentando o dado do beneficiário. Se o receber retira toda a quantidade disponível do produto ele é excluído do banco. E se mais de uma instituição retira do mesmo produto, é feito um update em cima dos dados do receber anterior.

Fizemos 5 consultas com framework de aggregation para mostrar um relatório final sobre as doações e informações sobre o alimento:

```python
def resultados_agregados(request):
    manager = Doacao()

    # Consulta 1: Total de Alimentos por Categoria
    pipeline_categoria = [
        {"$group": {
            "_id": "$categoria",
            "total_alimentos": {"$sum": "$quantidade"}
        }},
        {"$sort": {"total_alimentos": -1}}
    ]
    total_alimentos_por_categoria = manager.executar_agregacao(pipeline_categoria)
    # Renomeando chave _id para categoria
    for item in total_alimentos_por_categoria:
        item['categoria'] = item.pop('_id')

    # Consulta 2: Doadores com Maior Quantidade de Doações
    pipeline_doadores = [
        {"$group": {
            "_id": "$nome",
            "total_doado": {"$sum": "$quantidade"}
        }},
        {"$sort": {"total_doado": -1}},
        {"$limit": 5}
    ]
    doadores_mais_doaram = manager.executar_agregacao(pipeline_doadores)
    # Renomeando chave _id para nome
    for item in doadores_mais_doaram:
        item['nome'] = item.pop('_id')

    # Consulta 3: Alimentos Próximos da Validade
    pipeline_validade = [
        {"$match": {
            "validade": {"$gte": datetime.now()}
        }},
        {"$sort": {"validade": 1}},
        {"$limit": 10}
    ]
    alimentos_proximos_validade = manager.executar_agregacao(pipeline_validade)
```

```python
# Consulta 4: Quantidade Total de Alimentos Disponíveis
pipeline_quantidade_total = [
    {"$group": {
        "_id": None,
        "quantidade_total": {"$sum": "$quantidade"}
    }}
]
quantidade_total_alimentos = manager.executar_agregacao(pipeline_quantidade_total)

# Consulta 5: Histórico de Retiradas por Receptor
pipeline_retiradas = [
    {"$match": {
        "nome_recebedor": {"$ne": None}
    }},
    {"$group": {
        "_id": "$nome_recebedor",
        "total_retirado": {"$sum": "$quantidade_retirou"}
    }},
    {"$sort": {"total_retirado": -1}}
]
historico_retiradas_receptor = manager.executar_agregacao(pipeline_retiradas)
# Renomeando chave _id para nome_recebedor
for item in historico_retiradas_receptor:
    item['nome_recebedor'] = item.pop('_id')
```