PROJETO GERENCIADOR DE TAREFAS EM PYTHON

Por:

Arthur Almeida de Souza

Lucas da Silva Nascimento

ESTRUTURA DO PROJETO

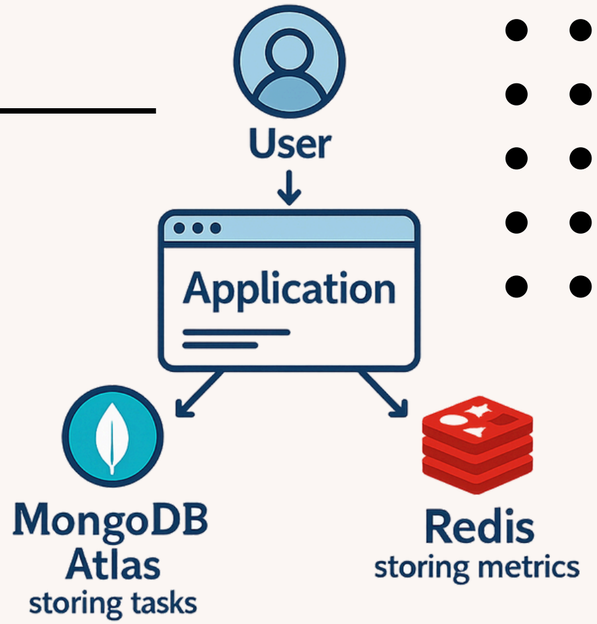
Arquivo main.py e func.py

redis

CONEXÕES

MongoDb: Banco de dados principal

Redis: Banco (in-memory) gravar e ler métricas




```python
# Conexão com MongoDB Atlas
cliente = MongoClient(
    "mongodb+srv://ArthurSouza:senhaFicticia@cluster0.dqda2cv.mongodb.net/"
    "?retryWrites=true&w=majority&appName=Cluster0"
)
banco = cliente.get_database("GerenciadorTarefas")
tarefas = banco.get_collection("ListaTarefas")

```

Utilizamos uma string de conexão para comunicar com o MongoDb

```python
redis_client = Redis(host='localhost', port=6379, db=0)
DEFAULT_USER_ID = "13"  # User id pré-definido
user_id = DEFAULT_USER_ID

```

Utilizando usuário fixo para questões de ser uma demonstração

CRIAÇÃO DE TAREFAS

Utilizando dicionário para armazenar os dados

```python
def adicionar_tarefa(colecao, titulo, descricao, tags):
    nova_tarefa = {
        "titulo": titulo,
        "descricao": descricao,
        "criado_em": datetime.datetime.now(),
        "status": "pendente",
        "tags": tags,
        "comentarios": []
    }

```

Função do Mongo Insert_One para adicionar o dicionário nova_tarefa

```
colecao.insert_one(nova_tarefa)
```

UTILIZANDO REDIS

```python
redis_client.hincrby(f"user:{user_id}:tasks:status", "pendente", 1)

data_hoje = datetime.datetime.now().strftime("%Y-%m-%d")
redis_client.hincrby(f"user:{user_id}:tasks:created", data_hoje, 1)

```

Parâmetros do hincrby:

Name

Field

Increment

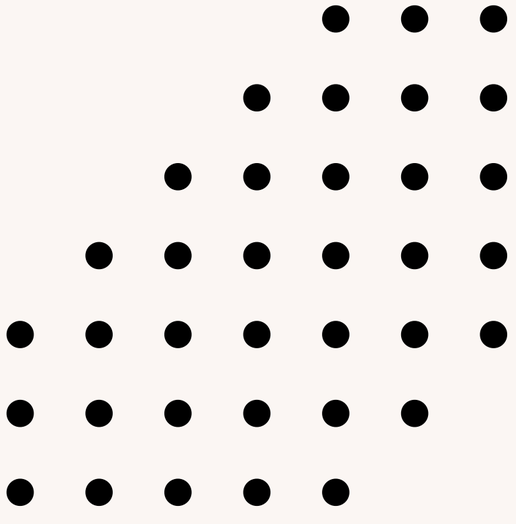
```python
for tag in tags:
    redis_client.zincrby(f"user:{user_id}:tags:top", 1, tag)
```

Parâmetros do zincrby:

Sorted set key

Incremento

Membro



MODIFICAR TAREFA – MONGO

Verificar se tarefa existe
armazenar dados antigos
Atualizar no banco
Armazenar dados novos

```python
def modificar_tarefa(colecao, id_tarefa, dados_novos):
    tarefa_atual = colecao.find_one({"_id": ObjectId(id_tarefa)})
    if not tarefa_atual:
        return None

    status_antigo = tarefa_atual.get("status", "pendente")
    tags_antigas = tarefa_atual.get("tags", [])

    resultado = colecao.update_one(
        {"_id": ObjectId(id_tarefa)},
        {"$set": dados_novos}
    )
    if resultado.modified_count == 0:
        return None

    novo_status = dados_novos.get("status", status_antigo)
    novas_tags = dados_novos.get("tags", tags_antigas)
```

MODIFICAR TAREFA – REDIS

```python
if novo_status != status_antigo:
    redis_client.hincrby(f"user:{user_id}:tasks:status", status_antigo, -1)
    redis_client.hincrby(f"user:{user_id}:tasks:status", novo_status, 1)

    if novo_status == "concluida":
        data_hoje = datetime.datetime.now().strftime("%Y-%m-%d")
        redis_client.hincrby(
            f"user:{user_id}:tasks:completed",data_hoje,1)

        criado_em = tarefa_atual.get("criado_em")
        if isinstance(criado_em, datetime.datetime):
            tempo_conclusao = (datetime.datetime.now() - criado_em).total_seconds()
            redis_client.hincrbyfloat(
                f"user:{user_id}:stats:productivity","soma_tempo_conclusao",tempo_conclusao)
            redis_client.hincrby(
                f"user:{user_id}:stats:productivity","num_tarefas_concluidas",1)
```

Comparando as variáveis armazenadas para atualização

Atualizando métrica de tarefas concluídas hoje!

Comparando a diferença do tempo criado e finalizado para ter a métrica de tempo de conclusão

Atualizando tags novas e armazenando no top de tags

```python
if set(novas_tags) != set(tags_antigas):
    for tag in novas_tags:
        redis_client.zincrby(f"user:{user_id}:tags:top", 1, tag)
return resultado
```

REMOVER TAREFA – REDIS E MONGO

```python
def remover_tarefa(colecao, id_tarefa):
    tarefa = colecao.find_one({"_id": ObjectId(id_tarefa)})
    if not tarefa:
        return None

    status_atual = tarefa.get("status", "pendente")
    resultado = colecao.delete_one({"_id": ObjectId(id_tarefa)})

    if resultado.deleted_count > 0:
        redis_client.hincrby(f"user:{user_id}:tasks:status", status_atual,

    return resultado
```

Identificar tarefa e verificar se existe receber status atual deletar no banco

Verificar se já tem alguma tarefa deletada e atualiza o status para status concluída no Redis

OUTRAS FUNÇÕES

Cria um dicionário com texto e data de criação e atualiza no banco

```python
def inserir_comentario(colecao, id_tarefa, texto_comentario):
    comentario = {
        "texto": texto_comentario,
        "data": datetime.datetime.now()
    }
    return colecao.update_one(
        {"_id": ObjectId(id_tarefa)},
        {"$push": {"comentarios": comentario}}
    )
```

```python
def listar_tarefas(colecao, status=None, tags=None):
    filtros = {}
    if status:
        filtros["status"] = status
    if tags:
        filtros["tags"] = {"$in": tags}
    return list(colecao.find(filtros))
```

MÉTRICAS PEDIDAS

```python
print("\nContadores de status:")
status_hash = redis_client.hgetall(f"user:{user_id}:tasks:status")
if not status_hash:
    print("  (nenhum dado encontrado)")
else:
    for campo, valor in status_hash.items():
        print(f"  {campo.decode()}: {int(valor)}")
```

hgettall - Recupera todos os campos e valores do hash de uma só vez retorna dicionário objetos do tipo bytes

decode - Transforma aquele objeto bytes em uma string como “pendente”

```python
print("\nTarefas concluídas por dia:")
chave_completed = f"user:{user_id}:tasks:completed"
data_hoje = datetime.datetime.now().strftime("%Y-%m-%d")
valor_concluidas = redis_client.hget(chave_completed, data_hoje)
qtd_concluidas = int(valor_concluidas) if valor_concluidas else 0
print(f"  Hoje ({data_hoje}): {qtd_concluidas}")
```

hgett - recupera um valor único

Linha 5 - Caso tenha algo em valor concluídas, ele já será convertido de bytes para inteiro

zrevrange - Intervalo de início e fim, definido em posições, irá listar as 10 tags mais utilizadas

```python
print("\nRanking de tags (top 10):")
top_tags = redis_client.zrevrange(f"user:{user_id}:tags:top", 0, 9, withscores=True)
if not top_tags:
    print("  (nenhum dado encontrado)")
else:
    for tag, score in top_tags:
        print(f"  {tag.decode()}: {int(score)}")
```

MÉTRICAS PEDIDAS

```python
stats = redis_client.hgetall(f"user:{user_id}:stats:productivity")
soma_tempo = float(stats.get(b"soma_tempo_conclusao", 0))
num_concluidas = int(stats.get(b"num_tarefas_concluidas", 0))
tempo_medio = soma_tempo / num_concluidas if num_concluidas else 0
print(f"  Tempo médio de conclusão (segundos): {tempo_medio:.2f}")
```

stats - Dicionário Python com chaves do tipo bytes.
.get(key, default) - Tenta retornar stats[key].
b- literal de bytes

hget- recupera
um valor único

```python
chave_created = f"user:{user_id}:tasks:created"
valor_criadas = redis_client.hget(chave_created, data_hoje)
qtd_criadas = int(valor_criadas) if valor_criadas else 0
print(f"  Tarefas criadas hoje: {qtd_criadas}")
```

MÉTRICAS PEDIDAS

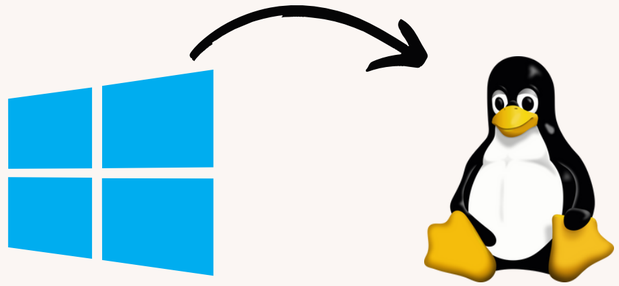
```python
hoje = datetime.datetime.now().date()
total_concluidas_semana = 0
total_criadas_semana = 0
for i in range(7):
    dia = (hoje - datetime.timedelta(days=i)).strftime("%Y-%m-%d")
    # soma concluídas deste dia
    qc = redis_client.hget(chave_completed, dia)
    total_concluidas_semana += int(qc) if qc else 0
    # soma criadas deste dia
    qc2 = redis_client.hget(chave_created, dia)
    total_criadas_semana += int(qc2) if qc2 else 0

if total_criadas_semana:
    taxa_semanal = (total_concluidas_semana / total_criadas_semana) * 100
else:
    taxa_semanal = 0.0
print(f"  Taxa de conclusão últimos 7 dias: {taxa_semanal:.2f}%")
```

Lógica para obter a taxa de conclusão semanal utilizando datetime e método hget para ober chaves únicas como
concluídas
Criadas
retornando total de concluídas e dividindo por criadas na semana

OBSERVAÇÕES DO REDIS

Redis é desenvolvido originalmente para Linux. O código-fonte usa APIs e dependências que naturalmente rodam em kernel Linux.



WSL

Subsistema windows que roda um linux.


```
redis-cli
127.0.0.1:6379>
DEL chave1 chave2 chave3
DEL chave1 chave2 chave3

FLUSHDB
127.0.0.1:6379> QUIT
```

Exemplo prático de como acessar o redis via CMD. (Tendo que ter instalado o WSL e configurado)