Sistema Task Manager

Objetivo

Desenvolver um **Sistema de Gerenciamento de Tarefas** simples em Python, aplicando:

- Arquitetura em camadas
- Testes automatizados com pytest
- Separação de responsabilidades

O que o sistema faz?

- Criar tarefas com título, descrição, prioridade e prazo
- Listar todas as tarefas
- Buscar tarefa por ID
- Atualizar status da tarefa
- Deletar tarefas

Estrutura do Projeto

```
task_manager/
  task.py
  storage.py
  repository.py
  service.py                                    (Bônus)
tests/
  test_task.py
  test_repository.py
requirements.txt
README.md
```

Tecnologias

- Python 3.8+
- pytest
- pytest-mock

1 Arquivo: task.py

O que implementar

1. Enum Priority

```
from enum import IntEnum

class Priority(IntEnum):
 BAIXA = 1
 MEDIA = 2
 ALTA = 3
```

2. Enum Status

```
from enum import Enum

class Status(Enum):
 PENDENTE = "pendente"
 EM_PROGRESSO = "em_progresso"
 CONCLUIDA = "concluida"
```

3. Classe Task Atributos:

- `id`: int ou None
- `titulo`: str
- `descricao`: str
- `prioridade`: Priority
- `prazo`: datetime
- `status`: Status (padrão: PENDENTE)

Métodos obrigatórios:

- `validar()`: Verifica se título tem 3+ caracteres e prazo não é passado. Lança ValueError se inválido.

Testes (test_task.py)

Criar pelo menos 3 testes:

1. Teste de criação válida
2. Teste de título inválido (deve lançar ValueError)
3. Teste de prazo no passado (deve lançar ValueError)

Exemplo:

```
import pytest
from datetime import datetime, timedelta
from task_manager.task import Task, Priority, Status

def test_task_valida():
 prazo = datetime.now() + timedelta(days=1)
 task = Task(None, "Estudar", "Python", Priority.ALTA, prazo)
 task.validar()  # Nao deve lancar erro
 assert task.titulo == "Estudar"

def test_titulo_curto_invalido():
 prazo = datetime.now() + timedelta(days=1)
 task = Task(None, "AB", "Desc", Priority.BAIXA, prazo)
 with pytest.raises(ValueError):
  task.validar()
```

2 Arquivo: storage.py

Classe InMemoryStorage

Armazena dados em memória (dicionário). **Atributo:**

- `_data`: dict vazio no __init__

Métodos obrigatórios:

- `add(id, item)`: Adiciona item com chave id
- `get(id)`: Retorna item ou None
- `get_all()`: Retorna lista com todos os valores
- `delete(id)`: Remove item, retorna True/False
- `clear()`: Limpa tudo

Exemplo:

```
class InMemoryStorage:
 def __init__(self):
  self._data = {}

 def add(self, id, item):
  self._data[id] = item

 def get(self, id):
  return self._data.get(id)

 # ... outros metodos
```

3 Arquivo: repository.py

Classe TaskRepository

Gerencia persistência de tarefas.
Atributos:

- `storage`: InMemoryStorage (recebe no construtor)
- `_next_id`: int (começa em 1)

Métodos obrigatórios:

- `save(task)`: Atribui ID, salva no storage, retorna task
- `find_by_id(id)`: Busca e retorna task ou None
- `find_all()`: Retorna lista de todas as tasks
- `delete(id)`: Remove task

Exemplo do método save:

```
def save(self, task):
 task.id = self._next_id
 self._next_id += 1
 self.storage.add(task.id, task)
 return task
```

Testes (test_repository.py)

Criar pelo menos 4 testes usando **mocks**:

1. Teste save atribui ID
2. Teste save chama storage.add

3. Teste find_by_id chama storage.get
4. Teste find_all retorna lista

Exemplo com mock:

```
def test_save_atribui_id(mocker):
 mock_storage = mocker.Mock()
 repo = TaskRepository(mock_storage)

 task = Task(None, "Teste", "Desc",
 Priority.BAIXA, datetime.now())

 resultado = repo.save(task)

 assert resultado.id == 1
 mock_storage.add.assert_called_once()
```

4 Arquivo: service.py (BÔNUS)

Classe TaskService

Atributo:

- `repository`: TaskRepository (recebe no construtor)

Métodos:

- `criar_tarefa(...)`: Cria, valida e salva
- `listar_todas()`: Retorna todas
- `atualizar_status(id, status)`: Atualiza status

5 Arquivos de Configuração

requirements.txt

```
pytest==7.4.0
pytest-mock==3.11.1
```

README.md

Incluir:

- Título e descrição

- Como instalar: `pip install -r requirements.txt`
- Como testar: `pytest -v`
- Estrutura do projeto

6 Checklist

- Task com Priority e Status
- Método validar() funcionando
- InMemoryStorage com 5 métodos
- TaskRepository com 4 métodos
- Testes de Task (3+)
- Testes de Repository com mocks (4+)
- README.md completo
- (Bônus) TaskService

7 Dicas

1. Implemente uma classe por vez
2. Teste cada classe antes de ir para a próxima
3. Use `pytest -v` para ver detalhes dos testes
4. Consulte a documentação do pytest quando tiver dúvidas

8 Como Executar

Instalar dependências:

```
pip install -r requirements.txt
```

Executar testes:

```
pytest -v
```

Executar com cobertura (opcional):

```
pytest --cov=task_manager
```

9 Exemplo de Uso

```
from datetime import datetime, timedelta
from task_manager.task import Task, Priority
from task_manager.storage import InMemoryStorage
from task_manager.repository import TaskRepository

# Criar componentes
storage = InMemoryStorage()
repo = TaskRepository(storage)

# Criar tarefa
prazo = datetime.now() + timedelta(days=5)
task = Task(None, "Estudar", "Python", Priority.ALTA, prazo)
task.validar()

# Salvar
task_salva = repo.save(task)
print(f"ID: {task_salva.id}")

# Buscar
encontrada = repo.find_by_id(1)
print(f"Titulo: {encontrada.titulo}")
```