Gerenciador De Projetos Pessoais

Trabalho da disciplina de Engenharia de Software na graduação de Ciência de Dados, FGV/EMAp.

Membros:

```
Breno Azevedo
Cleomar Felipe
Gabriel Pastori
Gustavo Ramalho
Raul Lomonte
```

Introdução

Com a crescente necessidade de organização e eficiência em nosso cotidiano, nosso sistema surge como uma ferramenta indispensável para aqueles que buscam otimizar seu tempo e recursos. Esta plataforma permite que os usuários criem e administrem projetos, detalhando e acompanhando suas respectivas tarefas de maneira intuitiva. Além disso, com um dashboard informativo, os usuários têm uma visão completa de seus compromissos, enquanto as funcionalidades de etiquetas e filtragem facilitam a categorização e localização de projetos específicos, garantindo uma experiência de planejamento ágil e personalizada.

O diretório do projeto possui 3 pastas principais: 'docs', 'src', 'tests':

docs:

Contém o documento relacionado aos casos de uso escolhidos, estórias de usuários definidas e metáfora do sistema. Além disso, possui uma subpasta com todos os cinco diagramas solicitados: caso de uso, classe, componentes, pacotes e sequência.

PROF

src:

Contém a implementação do sistema de maneira geral, dividido em três subpastas: 'db': database, 'gui': estruturação visual do app e 'logic': parte lógica do app.

tests:

Onde foi realizado todos os testes referentes ao trabalho. Cada subpasta é definida como a funcionalidade que estamos testando.

Como Executar:

Para executar o programa, siga as instruções abaixo:

1. Certifique-se de ter o arquivo 'env', enviado em anexo, presente no diretório do projeto.

2. Certifique-se de instalar as dependências do projeto. Para isso, utilize o seguinte comando no terminal:

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

3. Utilize o seguinte comando no terminal:

```
python -m src.main
```

Planejamento, análise e design do sistema

Os casos de uso desse projeto estão detalhadamente documentados na pasta 'docs'. É importante ressaltar que eles foram criados no intuito de ter uma complexidade razoável, por exemplo, agregando tarefas de CRUD em um único caso de uso. Assim, foi possível criar todas as funcionalidades do sistema que desejavamos. Além disso, criamos as estórias de usuários e uma metáfora do sistema denominada "O Jardim de Projetos". Por fim, todos os diagramas solicitados foram criados, inclusive os dois extras - componentes e sequência.

Desenvolvimento

Para um melhor desenvolvimento do projeto, todos os commit's criados tentam ser o mais explicativos possíveis sobre o que é alterado. A elaboração da interface foi feita com TKinter e em todo o tempo fizemos refatoração do código. Além disso, utilizamos persistência em BD na nuvem.

Design Patterns

Singleton

Simple Factory

PROF

Mediator

Adapter

Utilizamos adapter para a funcionalidade de importar documentos. Inicialmente, é importado no formato json para o app, e o adapter permitiu a importação também de formatos txt e csv. Assim, o usuário possui uma gama maior de escolhas na hora de importar o seu documento.

Memento

Facade

Empregamos o padrão de design Facade para simplificar a interação com os 10 casos de uso complexos do sistema. Foi uma escolha interessante para encapsular a complexidade operacional, proporcionando uma interface unificada e simplificada. O fachade melhorou a legibilidade e manutenção do código,

além de também garantir uma arquitetura robusta e flexível, facilitando futuras expansões e adaptações do projeto.

Testes e qualidade de software

Para aprimorar a legibilidade e a qualidade do nosso código, implementamos várias práticas padrão. Primeiramente, todo o código é documentado conforme as diretrizes da PEP 257 (DocString), e organizado em módulos seguindo a PEP 8. Além disso, priorizamos o uso de identificadores significativos e a inclusão de type hints para maior clareza.

Para assegurar a integridade do código durante o processo de merge e versionamento, implementamos uma política de revisão obrigatória: nenhum push pode ser feito diretamente sem a aprovação de outro membro da equipe.

No que tange aos testes, adotamos o framework unittest. Para complementar, desenvolvemos mocks específicos dentro do próprio unittest, garantindo testes eficientes tanto para a lógica do código quanto para suas interfaces visuais. Como um adicional, utilizamos a ferramenta Sonar para gerar relatórios detalhados sobre a qualidade do nosso código, assegurando continuamente sua excelência. A ferramenta pylint também é utilizada para reforçar a aderência aos padrões de codificação.