

PROJETO TASK FLOW

Equipe: Flávio dos Santos, Eduarda Lopes, Jenifer Barreto

Resumo do Tema

O desenvolvimento de um sistema organizador de tarefas visa proporcionar uma solução eficaz para otimizar a gestão das atividades diárias do usuário, promovendo uma abordagem estruturada e funcional para o planejamento e acompanhamento de suas responsabilidades.

Este sistema atua diretamente no combate à procrastinação, à falha de memorização e à desorganização, ao proporcionar um ambiente intuitivo e de fácil acesso, onde o usuário pode não apenas criar e gerenciar suas tarefas, mas também priorizá-las, estabelecer prazos e monitorar seu progresso.

Ao integrar ferramentas de organização e lembretes, o sistema busca aumentar a produtividade, reduzir a sobrecarga cognitiva e, conseqüentemente, melhorar a eficiência nas tarefas cotidianas, favorecendo o alcance de metas e o desenvolvimento de hábitos positivos.

Público-Alvo

O público-alvo do sistema é, primordialmente, composto por estudantes universitários, um grupo que frequentemente enfrenta dificuldades com a gestão do tempo e organização das suas atividades diárias. O sistema foi projetado para atender às necessidades específicas desse público, visando aumentar sua produtividade e eficácia no cumprimento das tarefas acadêmicas. Além disso, a ferramenta pode se estender a outros indivíduos que busquem uma maneira eficiente de organizar suas responsabilidades cotidianas, como profissionais e professores.

Método de Pesquisa Utilizado

Para identificar as funcionalidades mais relevantes para o público-alvo, foi adotada a metodologia de pesquisa baseada em questionários online. Este método se mostrou eficaz para alcançar um público amplo e diversificado de estudantes universitários.

Utilizamos a plataforma Google Forms para facilitar a criação, distribuição e análise dos dados. Através dessa abordagem, foi possível coletar informações diretas sobre as preferências, dificuldades e expectativas dos usuários em relação à gestão de suas tarefas cotidianas. Os dados obtidos permitiram um entendimento aprofundado das necessidades do público-alvo, contribuindo para o desenvolvimento de funcionalidades que garantem a eficiência e a personalização do sistema.

Requisitos Identificados

Com base nas respostas obtidas nos questionários e nas validações dos protótipos, foram identificadas as funcionalidades principais que o sistema deveria oferecer:

- **Adicionar e Editar Tarefas:** Permitir ao usuário criar, modificar e excluir tarefas com facilidade.
- **Classificação por Categoria e Status:** A possibilidade de organizar as tarefas de acordo com sua natureza (por exemplo, acadêmicas, pessoais) e status (pendente, em progresso, concluída).
- **Lembretes e Notificações:** Alertas automáticos para garantir que o usuário não esqueça de seus compromissos e prazos.
- **Interface Amigável e Intuitiva:** Um design que facilite a navegação e a interação com o sistema, com um layout claro e simples.

Processo de Elicitação

No processo de elicitação, o método escolhido foi a prototipação, com a criação de protótipos rápidos para validação com o público-alvo. Após a análise das respostas dos questionários online, optamos por apresentar modelos de interface simplificados, com funcionalidades-chave, para obter feedback direto dos usuários.

Essa abordagem de prototipação permitiu a iteração rápida e a adaptação do design e das funcionalidades, garantindo que o sistema atendesse melhor às necessidades reais dos estudantes. A prototipação também possibilitou a identificação de ajustes e melhorias antes do desenvolvimento completo, minimizando riscos e otimizando o processo de criação.

Especificação Final dos Requisitos

A partir dos dados coletados e da validação dos protótipos, os requisitos finais foram definidos, levando em consideração as principais necessidades dos usuários:

- **Cadastro de Tarefas:** O sistema deve permitir que o usuário crie, edite e apague tarefas, atribuindo a elas categorias e prazos.
- **Gestão de Prioridades:** O usuário poderá marcar as tarefas com diferentes níveis de prioridade, facilitando a organização das atividades mais urgentes.
- **Notificações e Alertas:** Lembretes automáticos serão enviados antes dos prazos das tarefas, para garantir que o usuário esteja ciente de suas responsabilidades.

- **Interface Simples e Clara:** A interface gráfica deve ser intuitiva, com fácil navegação e recursos visuais que ajudem o usuário a compreender e utilizar o sistema de forma eficiente.

Tecnologias

Python: Linguagem de programação utilizada para a construção do sistema, devido à sua versatilidade, simplicidade e amplo suporte para bibliotecas de interfaces gráficas.

SQLite: Banco de dados leve e eficiente utilizado para armazenar as tarefas e informações do usuário, proporcionando uma solução simples para a persistência dos dados.

Bibliotecas: tkinter

Recursos e Restrições

Recursos

Disponíveis:

Flávio: Responsável pelo Frontend e design da interface, garantindo uma experiência de usuário fluida e agradável.

Eduarda: Responsável pelo Backend e documentação técnica, assegurando que a lógica do sistema esteja bem implementada e documentada.

Jenifer: Responsável pelos testes e pela documentação, realizando testes de funcionalidade e garantindo a qualidade do sistema.

Aplicação de Design de Software no Projeto

Utilizaremos o SRP no organizador de tarefas, separando as responsabilidades entre a interface gráfica e o gerenciamento de tarefas.

A interface gráfica deve ser responsável apenas pela apresentação e interação com o usuário, enquanto o gerenciamento das tarefas, como adição, remoção e armazenamento, deve ser feito por outra classe.

Ficando

assim:

Classe de Tarefa: Responsável por armazenar e manipular os dados das tarefas.

Classe de Interface Gráfica: Responsável apenas pela interação com o usuário.

Modelos de Design

Escolhemos MVC (Model-View-Controller) como padrão de design de software, pois ele separa as preocupações e responsabilidades do software em três componentes principais:

Model (Modelo): que é responsável pela lógica de negócios e pela manipulação dos dados. Ele armazena as informações e a lógica de como essas informações são alteradas.

View (Visão): cuida da apresentação visual para o usuário. Ele exibe os dados do Model e reflete qualquer alteração feita neles.

Controller (Controlador): atua como intermediário entre a View e o Model. Ele recebe entradas do usuário, processa essas entradas (interagindo com o Model) e atualiza a View.

Isso permite uma maior flexibilidade e modularidade do código, facilitando a manutenção e a evolução do software ao longo do tempo. Como o projeto já menciona o SRP, o MVC se encaixa perfeitamente, pois reforça essa separação de responsabilidades.

UML Caso de uso:

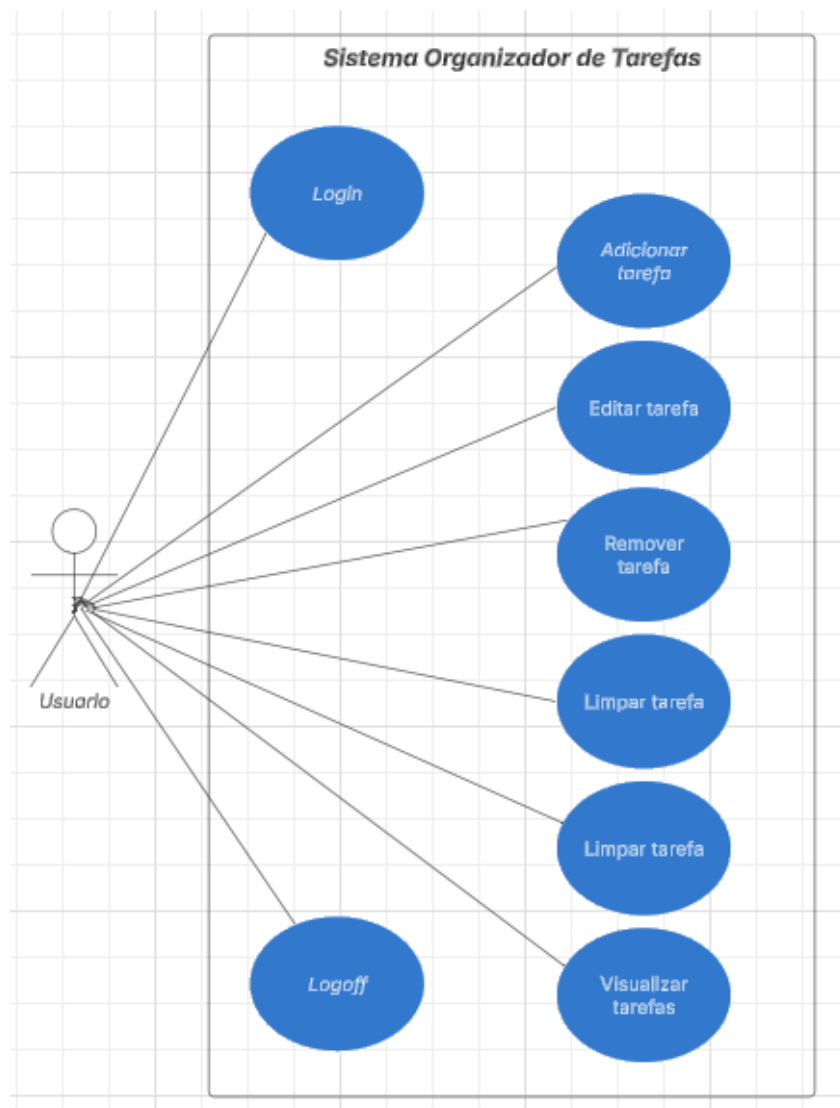
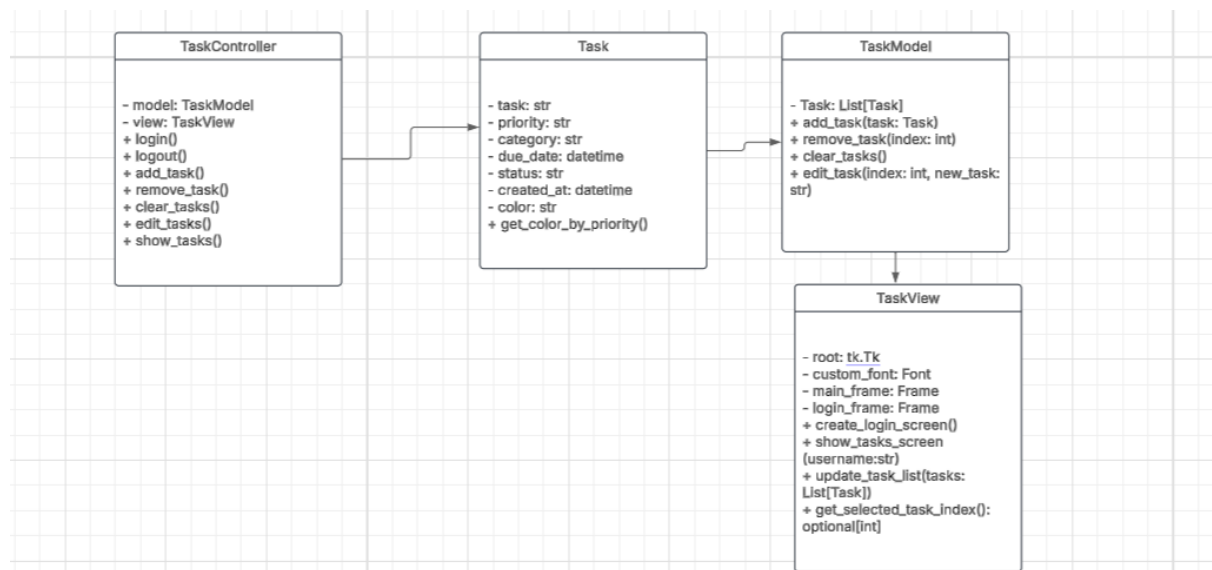


Diagrama de Classe:



LINK DO REPOSITÓRIO:

<https://github.com/FreitasLopes/TASKFLOW>

LINK DAS RESPOSTAS DO FORMULÁRIO:

[TASK FLOW \(respostas\).xlsx](#)

LINK LUCIDCHART – UML CASO DE USO:

https://lucid.app/lucidchart/8ffec047-f180-4d4d-a4fd-c8fb2ed5598b/edit?invitationId=inv_dd099546-a877-47f1-927b-0083e6ddf314

LINK LUCIDCHART – UML DIAGRAMA DE CLASSE:

https://lucid.app/lucidchart/040bd115-9e0c-4f0e-93b3-c884fdd90e56/edit?viewport_loc=-1401%2C-742%2C2462%2C942%2C0_0&invitationId=inv_46b5d148-d41a-4f3e-a619-94a17cdb62b1

