Áreas: TI & Computação	**UC:** Sistemas Distribuídos e Mobile	
Professor: Gustavo Fernandes	Trabalho Prático (A3)	
Grupo:	Data Apresentação e Entrega: **19/06/2023**	Valor: **20 pontos**

PROJETO

Instruções

1. A turma deve ser dividida em grupos de **no máximo** 3 alunos. O grupo deve criar um repositório no github **para disponibilizar o código fonte.**
2. Deve ser entregue o código funcional no repositório e este mesmo código apresentado para o professor. Se a aplicação apresentada não condiz com o código no github, a avaliação será nula, assim como apresentação de programas que não executam.
3. Vale ressaltar que não será admitida **cópia de trabalhos**, de espécie alguma. Entendam **não admitir cópia** como sendo **nota zero para ambos os grupos**.
4. No dia **19/06/2023** os grupos deverão estar presentes e todos os integrantes devem comparecer no dia. O grupo apresenta a aplicação para o professor e passará por perguntas sobre o código. O aluno que não demonstrar conhecimento sobre o trabalho será penalizado na nota.

Bom trabalho a todos!

Objetivo

O objetivo deste trabalho é desenvolver uma aplicação (CRUD) de **Tarefas do dia para Fazer**.

Requisitos do trabalho:

- A aplicação deve conter o **backend Flask** e o banco de dados em containers distintos.
- Uma tarefa possui, **pelo menos**: Um título, uma descrição e o horário limite para completar a tarefa.
- Uma tarefa atrasada, deve ter a cor do horário limite alterada para chamar a atenção do usuário na lista de tarefas.
- Os dados no banco alimentam o "frontend" da aplicação e são manipulados pela API.
- O frontend deve ser desenvolvido usando **templates em Jinja**, *para facilitar o trabalho*. Consulte os repositórios https://github.com/gustavoafernandes/flask-blog e https://github.com/gustavoafernandes/flask-blog-v2 para melhor entendimento dessa engine. Portanto, a própria API entrega as Views para o usuário.
- A aplicação deve permitir ao usuário Cadastrar uma tarefa, Ler uma tarefa específica, Excluir uma tarefa e Listar todas as tarefas.
- Crie uma Interface amigável e de fácil utilização.
- Além do código fonte da API, devem constar também no github um arquivo .sql contendo os comandos SQL para criar o banco e as tabelas necessárias, os templates e todos os arquivos estáticos (html, css, js, imagens, etc) e o arquivo Dockerfile usado para criar a imagem Docker da API. Todos **organizados** na estrutura do projeto.
- O professor poderá fazer testes clonando seu repositório e replicando a aplicação para validar seu trabalho. Nenhuma modificação de código será realizada pelo professor, portanto seu repositório deve entregar todos os arquivos para uma aplicação funcional.
- **Deixar o repositório público.**
- No *Readme* do repositório, adicionar:
  - Objetivo do Sistema
  - Colaboradores: Adicionar os integrantes do grupo, com link para o respectivo github de cada componente.

Pontos extras – Requisitos:

(+4 pontos) Criar o frontend por algum (micro)framework ou biblioteca. Isso permite um novo nó na aplicação, que deve, também, estar em seu próprio container.

(+2 pontos) Complementar a aplicação com a edição de uma tarefa, permitindo ao usuário editar o conteúdo (descrição, título e/ou horário limite) de uma tarefa que ainda não foi realizada.

(+2 pontos) Diferenciar uma tarefa feita de uma tarefa ainda sem fazer. Pode usar uma simples *tag* ou até mesmo cores distintas para diferenciar o *status* da tarefa.