Trabalho 01 - MongoDB

(BD Orientado a Documentos)

> **Disciplina**: Banco de Dados NoSQL
> **Prof**.: Camilo Barreto
> **Período**: 4
> **Curso**: Tec. Sistemas para Internet
> **IFTM - Campus Uberlândia Centro**
>
> **Informações**:
> **- Data de publicação**: 06/10/2023
> **- Data de entrega: 20/10/2023**
> **- Local de entrega**: Class Room
> **- Formato de entrega**: github (detalhes no final do documento).
> **- Pontos: 15 pts**

Acesso a Documentação do MongoDB:
- **Documentação Geral**: https://www.mongodb.com/docs/
- **Guides**: https://www.mongodb.com/docs/guides/
- **Getting Started**: https://www.mongodb.com/docs/manual/tutorial/getting-started/
- **JSON Schema**: https://www.mongodb.com/docs/manual/reference/operator/query/jsonSchema/

Ferramentas:
- **Gerar JSON:** http://www.objgen.com/json

Enunciado:

> Este trabalho envolve a modelagem e criação de um banco de dados para um sistema de gerenciamento de tarefas (To-Do List) usando MongoDB.

Um sistema de gerenciamento de tarefas, também conhecido como **To-Do List** (lista de tarefas), é uma aplicação ou ferramenta que permite que os usuários criem, organizem e acompanhem as tarefas que precisam ser realizadas. O objetivo principal de um sistema de gerenciamento de tarefas é ajudar as pessoas a manterem o controle de suas responsabilidades e prioridades.

Funcionalidades comuns em sistemas de gerenciamento de tarefas:

- **Criação de Tarefas:** Os usuários podem adicionar tarefas à lista, especificando detalhes como título, descrição, data de vencimento e prioridade.
- **Organização:** As tarefas podem ser organizadas em categorias, projetos ou listas, permitindo que os usuários classifiquem e agrupem suas responsabilidades de acordo com diferentes critérios.
- **Definição de Prioridade:** Os usuários geralmente podem atribuir uma prioridade a cada tarefa, indicando quais são mais importantes ou urgentes.
- **Data de Vencimento:** Muitos sistemas permitem definir uma data de vencimento para cada tarefa, ajudando os usuários a acompanhar prazos.
- **Marcação de Conclusão:** Os usuários podem marcar tarefas como concluídas quando terminam, movendo-as para uma seção de tarefas concluídas ou simplesmente marcando-as como feitas.

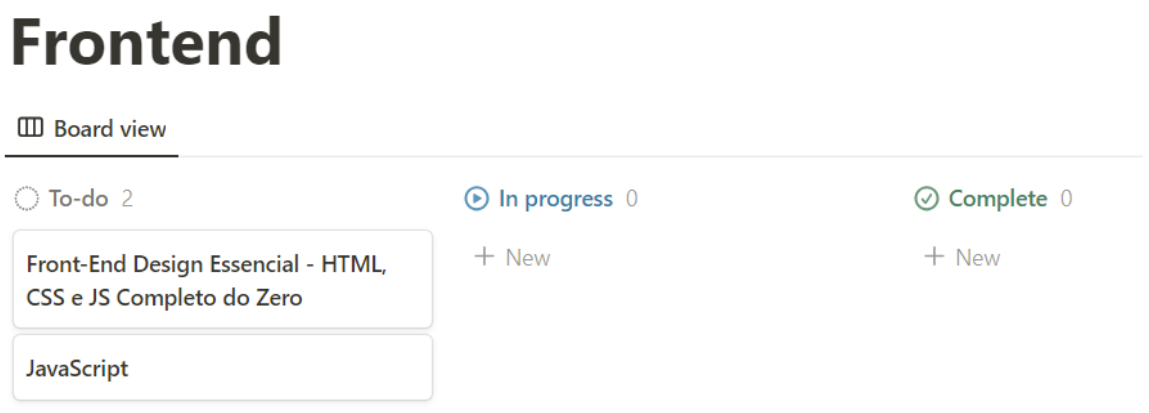
Desta forma, uma tarefa pode ter, mas não somente isto, os seguintes campos:

1. **Título;**
2. **Descrição;**
3. **Data de início;**
4. **Data de vencimento;**
5. **Prioridade;**
6. **Status Atual (To-do, In-Progress e Complete)**

É interessante também que uma tarefa tenha:

1. Atribuição de um usuário (quem irá fazer a tarefa), e;
2. Relação com outra tarefa (relacionamento opcional);

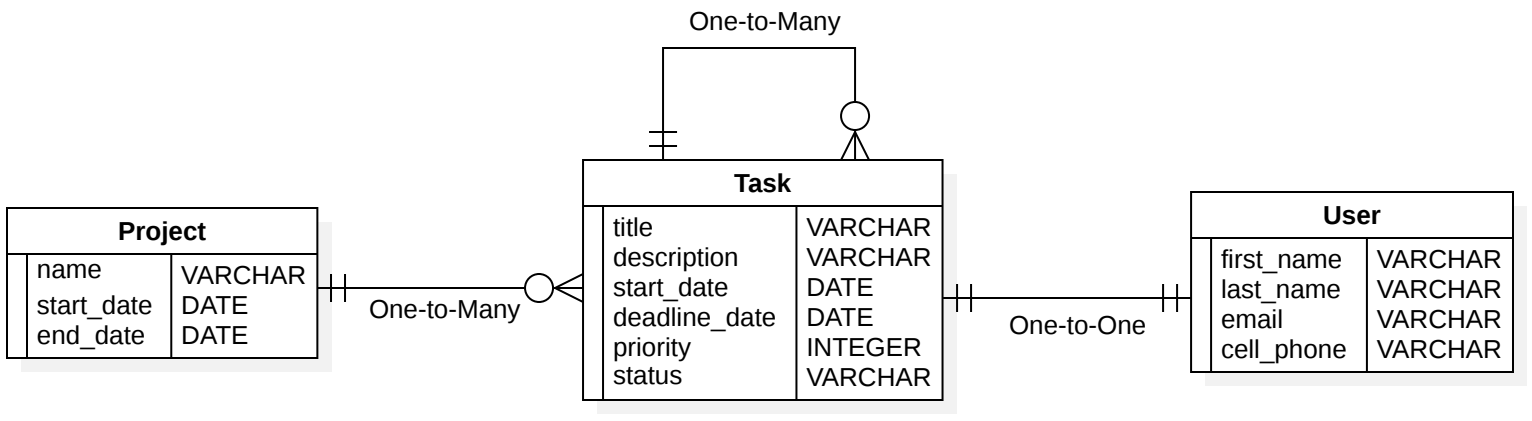
Um exemplo de To-Do list é o aplicativo Notion, que possibilita o gerenciamento de tarefas através de uma interface de usuário:



Pode-se observar que o gerenciamento está dividido em três agrupamentos baseado no “status” de cada tarefa. Desta forma, pode-se mudar o status de “To-Do” para “In-Progress” apenas arrastando uma tarefa para o lado.

Em uma análise baseada em Banco de Dados Relacionais, podemos exemplificar uma parte do sistema com o seguinte diagrama ER:

- **Tabela Project**: entidade principal contendo as informações do projeto e quais as tarefas relacionadas a ele;
- **Tabela Task**: entidade contendo as informações sobre uma tarefa específica do desenvolvimento de um projeto;
- **User**: entidade contendo dados do usuário atrelado a uma tarefa de um projeto;



💡 Considere esse exemplo para a implementação do trabalho!

A seguir serão descritos os passos para a implementação de um banco de dados para o sistema To-Do List.

1️⃣ Passo 1: Modelagem do Banco de Dados em MongoDB

Descrição:

Você deve projetar o esquema do banco de dados para o sistema de gerenciamento de tarefas. Considere os campos necessários para armazenar informações sobre cada tarefa, como título, descrição, data de criação, data de conclusão, status, etc (tudo que foi informado anteriormente). Utilize o diagrama apresentado acima para modelar o banco de dados, inclua mais campos caso achar necessário.

Considerações:

- Modele o banco de dados da melhor forma. Pode-se optar por utilizar documentos embarcados ou referenciados. Irei avaliar os critérios que você utilizou.
- Faça isso utilizando as melhores práticas de implementação;

Entrega:

- Documento com a modelagem do Banco de Dados apresentando o(s) arquivo(s) JSON;
- Descrição dos campos escolhidos: detalhamento de cada chave escolhida para compor um documento da coleção de projetos;

Exemplo resumido de representação de um documento:

```
{
  "name": "Project X",
  "startDate": "10/10/2023",
  "endDate": "10/10/2023",
  "tasks": [
     {
         "title": "XYZ"
         ....
         "user": {
              .....
          }
     }
  ]
}
```

2️⃣ Passo 2: Validação de Dados

Descrição:

Utilize a funcionalidade de validação de esquema do MongoDB para garantir que os dados inseridos no banco de dados estejam corretos. Defina regras de validação para campos obrigatórios, tipos de dados e outras restrições relevantes.

Entrega:

- Um documento com o arquivo JSON do Schema produzido baseando-se no “Passo 1”;

3️⃣ Passo 3: Criação do Banco de Dados e Coleção

Descrição:

Agora que o esquema de banco de dados foi projetado, crie um banco de dados MongoDB para armazenar os documentos relacionados ao sistema To-Do List.

Entrega:

- Scripts utilizados no processo de criação, como:
  - Comandos do Mongo Shell;
- Queries para criação, modificação, etc;

4️⃣ Passo 4: Operações de CR~~UD~~

Descrição:

Implemente as operações CR~~UD~~ (**Create**, **Read**, ~~Update, Delete~~) para gerenciar as tarefas no banco de dados.

- **Criar e Ler;**

Considerações:

- Construa e insira no banco de dados 30 documentos baseado na sua modelagem de banco de dados;

- Utilize ferramentas para produzir os documentos, como, por exemplo, o ChatGPT;
  Prompt para o ChatGPT:

```
Poderia criar um dataset com 30 documentos embarcados "tasks", baseando-se no
exemplo abaixo?

{
  "name": "Project X",
  "startDate": "10/10/2023",
  "endDate": "10/10/2023",
  "tasks": [
     {
         "title": "XYZ"
         ....
         "user": {
              .....
          }
     }
  ]
}
```

Provavelmente ele te fornecerá todos os documentos em etapas;

- Verifique se todos os documentos foram inseridos e que nenhum teve problemas com a validação;

Buscas:

- Crie 15 queries para busca de projetos, tarefas e usuários;
  - Implemente as queries mais utilizadas em sistemas deste tipo, como, por exemplo:
    - Todas as tarefas de um projeto que estejam com status “To-Do” (a fazer);
    - Todas as tarefas de um usuário que estejam com o status “To-Do” (a fazer);
    - Todas as tarefas com o prazo de entrega vencido.
    - Etc….

Entrega:

- Queries utilizadas no processo de inserção e seus resultados;
- Queries utilizadas no processo de busca e seus resultados;

5️⃣ Passo 5: Operações de Update

Implemente as operações de **Update** para gerenciar as tarefas no banco de dados.

Considerações:

- Implemente 5 queries para atualizar os documentos da coleção, por exemplo:
  - Alterar quem será o usuário responsável pela tarefa;
  - Dilatar o prazo de entrega da tarefa;
  - Mudar o status de “In Progress” para “Complete”;
  - Etc.

Entrega:

- Queries utilizadas no processo de atualização e seus resultados;

6️⃣ Passo 6: Indexação e Otimização

Descrição:

Crie **índices** no banco de dados para melhorar o desempenho das consultas. Crie índices para campos frequentemente usados, como data de criação e status da tarefa. Teste a diferença de desempenho antes e depois da indexação.

Entrega:

- Estatísticas do Banco de Dados antes da otimização;
- Estatísticas do Banco de Dados depois da otimização;

🏁 Entrega do Trabalho

> 💡 **A entrega do trabalho prático sobre o Sistema de Gerenciamento de Tarefas (To-Do List) será realizada por meio do GitHub!**
>
> **Crie um repositório no GitHub para o seu trabalho prático. Certifique-se de que o repositório esteja definido como "público" para que o professor possa acessá-lo.**

Estrutura do Repositório

1. Dentro do repositório, organize os arquivos e resultados de cada etapa em diretórios específicos. Cada diretório deve ser nomeado de acordo com o passo correspondente, por exemplo, "Passo1", "Passo2", e assim por diante.
2. Os documentos textuais, como relatórios, descrições de etapas ou qualquer outra documentação, devem ser escritos em formato Markdown (.md) para facilitar a leitura diretamente na plataforma do GitHub.

⚠️ Envie o link do trabalho finalizado na área deste trabalho no classroom!

Bom trabalho!