Descrição da API: Todo App GraphQL com Spring Boot e MongoDB

A API Todo App é um sistema baseado em GraphQL, desenvolvido utilizando o framework Spring Boot e integrado ao banco de dados NoSQL MongoDB. Projetado para gerenciar tarefas e usuários, o aplicativo oferece uma interface eficiente e flexível para operações de criação, leitura, atualização e exclusão (CRUD).

Recursos Principais:

1. **Usuários:**
    - Adição de usuários com nome e e-mail.
    - Busca de todos os usuários cadastrados.
    - Busca de usuário por e-mail ou ID.
2. **Tarefas:**
    - Adição de tarefas associadas a usuários existentes.
    - Atualização de informações de tarefas, incluindo descrição.
    - Busca de todas as tarefas cadastradas ou por usuário e status.

Funcionalidades Adicionais:

- Suporte a CORS para facilitar a integração com aplicações front-end em diferentes domínios.
- Interface GraphiQL interativa para facilitar o teste e exploração da API.
- Configuração flexível do MongoDB, permitindo autenticação e personalização das credenciais.

Utilização:

- Acesse o endpoint `/graphql` para interagir com a API usando consultas e mutações GraphQL.
- Explore a documentação interativa do GraphiQL para entender a estrutura da API e testar operações.

Requisitos Técnicos:

- Java 8 ou superior.
- Ambiente de execução Spring Boot.
- Servidor MongoDB em execução.

Observação: Certifique-se de fornecer as credenciais corretas para o MongoDB no arquivo de configuração para garantir a integridade e segurança dos dados.

Tecnologias:

1. **Spring:**
    - **O que é:** Spring é um framework para o desenvolvimento de aplicativos Java. Ele fornece uma infraestrutura abrangente para o desenvolvimento de software empresarial, abordando desde a camada de acesso a dados até a criação de APIs RESTful.
    - **Recursos Principais:**

        - **Injeção de Dependência:** Promove o desacoplamento de componentes e facilita a modularidade do código.
        - **Spring Boot:** Uma extensão do Spring que simplifica o desenvolvimento, configuração e implementação de aplicativos Java.
2. **GraphQL:**
    - **O que é:** GraphQL é uma linguagem de consulta para APIs e um ambiente de execução para executar essas consultas com os dados existentes em sua API.
    - **Recursos Principais:**
        - **Consulta Flexível:** Os clientes solicitam apenas os dados necessários, evitando over-fetching e under-fetching de dados.
        - **Único Ponto de Extremidade:** Geralmente, uma única rota de endpoint para consultas, proporcionando eficiência na integração.
        - **Tipagem Forte:** Os tipos de dados são definidos explicitamente, proporcionando segurança e previsibilidade nas consultas.
3. **MongoDB:**
    - **O que é:** MongoDB é um banco de dados NoSQL, orientado a documentos, que fornece flexibilidade na modelagem de dados.
    - **Recursos Principais:**
        - **Banco de Dados NoSQL:** Lida com dados não estruturados ou semiestruturados.
        - **Escalabilidade Horizontal:** Capacidade de escalonar horizontalmente para lidar com grandes volumes de dados e tráfego.

Utilização da API

Acesso ao Endpoint GraphQL

A API é acessada por meio do Endpoint GraphQL, que geralmente está configurado em `/graphql`. Para interagir com a API, envie consultas GraphQL para esse endpoint usando métodos HTTP, como POST.

Exemplo de acesso com cURL:

Plain Text
bash
Plain Text
curl -X POST -H "Content-Type: application/json" -d '{"query": "{ allUsers { id name email } }"}' http://seu-domínio.com/graphql

Certifique-se de substituir `http://seu-domínio.com/graphql` pelo URL real do seu servidor GraphQL.

Exploração Interativa com GraphiQL

Para facilitar a exploração interativa da API, o GraphiQL pode ser habilitado. O GraphiQL é uma interface gráfica interativa para construir e testar consultas GraphQL.

Ao acessar o endpoint da API no navegador, você pode ser redirecionado automaticamente para o GraphiQL, onde poderá experimentar consultas, visualizar a documentação e obter respostas em tempo real.

Exemplo de URL para acessar o GraphiQL:

Plain Text
plaintext
Plain Text
http://seu-domínio.com/graphiql

Lembre-se de verificar a documentação da API para obter detalhes sobre os tipos de dados disponíveis, as consultas suportadas e as mutações disponíveis.

Exemplos de Consultas GraphQL

Consultar Todos os Usuários

Plain Text
graphql
Plain Text
```
query {
 allUsers {
   id
   name
   email
 }
}
```

Adicionar Novo Usuário

Plain Text
graphql
Plain Text
```
mutation {
 addUser(name: "Nome do Novo Usuário", email: "novousuario@email.com") {
   id
   name
   email
 }
}
```

Análise de Requisitos:

1. **Levantamento de Requisitos:** Compreensão das necessidades e expectativas dos usuários e da empresa.
2. **Especificação de Requisitos:** Documentação detalhada dos requisitos funcionais e não funcionais.

3. **Validação de Requisitos:** Verificação da consistência, completude e viabilidade dos requisitos com as partes interessadas.
4. **Priorização de Requisitos:** Definição de prioridades para garantir a entrega de funcionalidades essenciais primeiro.

Modelagem do Sistema:

Os diagramas de classe são essenciais na representação de um sistema de software, pois:

1. **Estrutura do Sistema:** Mostram a estrutura de classes e suas relações, proporcionando uma visão geral.
2. **Relacionamentos:** Destacam associações entre objetos, facilitando a compreensão das interações.
3. **Herança e Polimorfismo:** Representam hierarquias de classes, ajudando na aplicação desses conceitos.

Implementação em POO:

1. **Encapsulamento:** Oculta detalhes internos de uma classe, promovendo a modularidade e a proteção dos dados.
2. **Herança:** Permite criar novas classes reutilizando características de classes existentes, promovendo a hierarquia e a extensibilidade.
3. **Polimorfismo:** Permite que objetos de diferentes classes sejam tratados de maneira uniforme, proporcionando flexibilidade.

Uso de Interfaces e Abstrações:

Interfaces e abstrações são vitais na construção de um SGE, pois:

1. **Interfaces:** Definem contratos para implementações, facilitando a substituição e a extensão.
2. **Abstrações:** Ocultam detalhes de implementação, fornecendo uma visão simplificada.

Elas promovem flexibilidade, facilitam a manutenção e possibilitam a integração de diferentes módulos.

Estas práticas são incorporadas em sua aplicação para garantir um design robusto e modular.