### **Sistema de Gerenciamento de Tarefas com Notificações Automáticas**

#### **Descrição:**

A aplicação será um sistema de gerenciamento de tarefas que permita ao usuário criar, atualizar e excluir tarefas. Além disso, o sistema enviará notificações automáticas sobre prazos por meio de uma API de mensageira (por exemplo, Telegram ou Twilio para SMS) e armazenará os dados das tarefas em um banco de dados. A aplicação será implementada com microserviços que se comunicarão por meio de containers Docker, e será possível hospedar partes da aplicação em uma instância de nuvem (como AWS, Azure ou Google Cloud).

#### **Tecnologias Utilizadas:**

1. **Backend em Python (Flask ou FastAPI)**: Responsável pelo gerenciamento das tarefas e pela integração com o banco de dados e o sistema de notificações. Python será usado para criar as rotas da API.
2. **Frontend em React.js (ou outra biblioteca JS)**: Interface de usuário para permitir a interação com as tarefas de forma visual. Os usuários poderão adicionar, editar e excluir tarefas diretamente no frontend.
3. **Banco de Dados (MongoDB ou PostgreSQL)**: Usado para armazenar as informações das tarefas dos usuários, incluindo prazos, descrições e status (completo/incompleto).
4. **Mensageira (Twilio ou Telegram API)**: Para enviar notificações aos usuários sobre prazos de tarefas próximos. Você pode integrar o Twilio para enviar SMS ou criar um bot no Telegram que envie as notificações.
5. **Containers (Docker)**: A aplicação será containerizada com Docker, permitindo que cada microserviço seja executado de forma independente. Um container será usado para o frontend, outro para o backend, e um terceiro para o banco de dados.
6. **Cloud Services (AWS/Google Cloud/Azure)**: Uma parte da aplicação, como o banco de dados ou a API de mensageira, pode ser hospedada em uma instância de cloud para garantir a escalabilidade e a disponibilidade.
7. **Automação (CI/CD)**: Automatizar o processo de deploy e testes utilizando ferramentas de integração contínua, como GitHub Actions, Travis CI, ou Jenkins.

#### **Funcionalidades:**

1. **CRUD de Tarefas**: Os usuários podem criar, editar, visualizar e excluir tarefas.
2. **Notificações Automáticas**: As tarefas com prazos receberão lembretes automáticos enviados via mensageira.
3. **Autenticação de Usuários**: Implementar autenticação básica (como JWT ou OAuth) para permitir o gerenciamento de tarefas pessoais.
4. **Deploy Automatizado**: Utilizando automação de CI/CD para deploy contínuo das novas funcionalidades.
5. **Containers com Docker**: Todos os serviços serão executados em containers independentes para facilitar o desenvolvimento, a escalabilidade e a manutenção.

#### **Pontos que atendem os critérios:**

* **Uso de repositório de código**: O projeto pode ser hospedado no GitHub ou GitLab.
* **Mensageira**: Integração com um serviço de mensagens (como Twilio ou Telegram) para envio de lembretes automáticos.
* **Containers e microserviços**: Uso de Docker para containerização dos serviços.
* **Banco de Dados**: Utilização de um banco de dados (MongoDB ou PostgreSQL) para armazenar as tarefas.
* **Serviços na nuvem**: Utilização de uma instância de nuvem para hospedar a aplicação ou parte dela (por exemplo, uma base de dados gerenciada em AWS RDS).