Tarefa: Desenvolvimento de um Sistema de Gerenciamento de Tarefas

Parte 1: Back-end (Laravel API)
Configuração do Projeto:

Crie um novo projeto Laravel utilizando o Composer.
Configure o banco de dados e as credenciais necessárias no arquivo .env.
Modelo e Migração:

Crie um modelo Task com os campos necessários (id, title, description, user_id).
Crie uma migração para a tabela tasks.
Controlador e Rotas:

Crie um controlador TaskController com métodos para listar, criar, atualizar e excluir tarefas.
Configure rotas nomeadas para cada uma das operações.
Autenticação:

Utilize o sistema de autenticação padrão do Laravel.
As rotas do TaskController devem exigir autenticação.
Consulta em Banco de Dados:

Na listagem de tarefas, retorne as tarefas associadas ao usuário autenticado.
Padrão de Projeto:

Utilize o padrão de projeto MVC.
Considere a separação de responsabilidades utilizando Service Classes ou Repositories.
Integração com Firebase:

Utilize a biblioteca do Firebase para notificar o front-end sobre alterações nas tarefas (opcional).
Parte 2: Front-end (Vue.js)
Configuração do Projeto:

Crie um projeto Vue.js utilizando o Vue CLI.
Componentes:

Crie componentes Vue para exibir a lista de tarefas, criar e editar uma tarefa.
Rotas:

Configure rotas nomeadas utilizando o Vue Router para navegação entre as páginas.
Autenticação:

Implemente uma página de login utilizando o Firebase Authentication.
Proteja as rotas no Vue Router para exigir autenticação.
Integração com Firebase:

Utilize a biblioteca do Firebase para receber notificações sobre alterações nas tarefas.
Consulta em Banco de Dados:

Consuma a API Laravel para exibir as tarefas associadas ao usuário autenticado.
Componentização:

Separe os componentes de Vue de forma a facilitar a reutilização.
Utilize props para passar dados entre componentes.
Critérios de Avaliação:
Estrutura de código organizada e seguindo os padrões recomendados.
Funcionalidades básicas (CRUD) implementadas corretamente.
Utilização de rotas nomeadas em ambas as partes.
Autenticação implementada tanto no back-end quanto no front-end.
Consulta em banco de dados de forma eficiente.
Uso de bibliotecas do Firebase para integração em ambas as partes.
Componentização no front-end para facilitar a manutenção e reutilização do código.
Este teste prático aborda uma variedade de conceitos importantes para um desenvolvedor fullstack, incluindo integração entre back-end e front-end, manipulação de banco de dados, autenticação e uso de bibliotecas externas. Certifique-se de fornecer uma documentação clara e instruções sobre como o projeto deve ser configurado e executado.