Questões Teóricas

1. Explique a diferença entre Eloquent ORM e Query Builder no Laravel. Quais são os prós e contras de cada abordagem?

Ambas são ferramentas para facilitar as interações com o banco de dados, porém Eloquent possui sua própria semântica, que pode ser aprendida facilmente na documentação do framework, enquanto Query Builder usa mais os conceitos de SQL puro.

Uma das vantagens do Eloquent é o relacionamento mais simples entre as tabelas, enquanto no Query Builder isso precisaria ser feito manualmente. Por outro lado, para consultas mais complexas, o Query Builder pode ser mais viável.

2. Como você garantiria a segurança de uma aplicação Laravel ao lidar com entradas de usuários e dados sensíveis? Liste pelo menos três práticas recomendadas e explique cada uma delas.

1. **Hash de dados sensíveis**
   Armazenamento de senhas e informações sensíveis devem utilizar as funções já fornecidas pela linguagem para criptografia

2. **Validações**
   Através do parâmetro request, é possível utilizar uma gama de validações.

3. **Prevenção a Cross Site Request**
   Previne simulação de um formulário, vindo de um outro domínio, para utilizar essa prevenção, basta usar a diretiva csrf do Blade

3. Qual é o papel dos Middlewares no Laravel e como eles se integram ao pipeline de requisição? Dê um exemplo prático de como você criaria e aplicaria um Middleware personalizado para verificar se o usuário está ativo antes de permitir o acesso a uma rota específica.

Middlewares são interceptadores de requisições, em um Middleware é possível fazer validações, e verificar se a requisição solicitada pode ter seguimento, ou para manipular outros tipos de dados, como logs.

Para criar um Middleware, basta utilizar o comando php artisan make:middleware ValidarUsuarioAtivo. Com isso, aplicar a regra de negócio no método handle, e registrar o Middleware no arquivo de configuração. Além disso, também incluir na rota específica a indicação para o Middleware registrado.

4. Descreva como o Laravel gerencia migrations e como isso é útil para o desenvolvimento de aplicações. Quais são as melhores práticas ao criar e aplicar migrations?

O laravel já traz por padrão migration comuns, como users, failed_jobs, e mais, com métodos up() e down(), que servem para subir ou remover as tabelas a partir dos

comandos do artisan. É possível criar nossas próprias migrations para mantermos um histórico da estrutura de nossas tabelas, e também facilitar novas instalações. É uma boa prática manter as estruturas simples e legíveis, e evitar alterações como renomeamento de tabelas e colunas, pois pode afetar todo o sistema, se não for bem administrado.

5. Qual é a diferença entre transações e savepoints no SQL Server? Como você usaria transações em um ambiente Laravel?

Ambas são formas de realizar alterações no banco de dados de forma segura, podendo voltar ao estado anterior ao início da transação sem afetar o banco de dados. A transação, quando falha, reverte tudo que foi feito até o ponto de início, já o savepoint pode ser definido em pontos específicos do processo, para que, caso seja necessário o rollback, ele pode ser feito de maneira parcial.

Em laravel, é útil usar transações, quando muitas alterações são feitas usando Eloquent ou Query Builder de uma só vez, tanto em uma como em mais de uma tabela.