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

Eloquent ORM (Object-Relational Mapping) é uma implementação de ORM que permite trabalhar com bancos de dados de uma maneira orientada a objetos. Nesse modelo a a necessidade de cada tabela do banco de dados possui uma "Model" correspondente .

Prós : Sua sintaxe é simples e intuitiva, oferece uma facilidade de contrução de relacionamentos , possui mecanismos de tratamento de consultas na camada de Modelo como Mutators, Collections e Factories.

Contra : A implementação do Eloquent requer um abstração claro sobre o que se espera do resultado, pois corre-se o risco de prejudicar a performance da aplicação tendo em vista seu alto consumo de memória.

Query Builder permite construir consultas SQL de forma programática, oferece uma maneira mais direta de interagir com o banco de dados sem a necessidade de lançar mão da camada de modelos.

Prós: Possui alto desempenho, uma vez que sua interação é direta ao banco de dados, há uma flexibilização nas instruções SQL, possui a segurança de uma consulta mais controlada e precisa e consome pouco recursos de servidor.

Contra : Requer um código mais simples que pode resultar num código mais “verboso”, não orientado a objetos, as queries devem ser escritas de forma mais completa e precisa, requer maior conhecimento de instruções SQL, sua aplicação pode ser mais demorada que ocasiona uma certa queda de produtividade.

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.

Para garantir a segurança dos dados de uma aplicação podemos lançar mão de validação de entrada para garantir a veracidade dos dados fornecidos pelos usuários antes de serem processados ou armazenado . Uma prática bastante utilizada é sempre possuir uma autenticação em duas faces lançando mão do mecanismo OAUTH2 nos caso de requisições REST e manter as o controle CSRF em todas as requisições WEB.

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 camada intermediária que processa uma requisição antes da mesma acessar a camada de controle. Eles desempenham um papel crucial no no controle de acesso, permitindo que possa adicionadas funcionalidades como autenticação, verificação de permissões, logging, proteção contra CSRF (Cross-Site Request Forgery).
Poderia aplicar uma validação na classe de controle ***app/Http/Auth/AuthenticatedSessionController.php*** da camada middleware se um usuário tem permissão a uma determinada rota, consultando uma valor atribuído na tabela users, como por exemplo acrescentar um campo “type_user” por exemplo.

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?

As migrations são de extrema importância no desenvolvimento de um projeto, uma vês que ela gerencia a criação, alteração e exclusão da estrutura das tabelas de um banco de dados de forma programática e controlada por versão ajudando a manter o controle da estrutura das tabelas.

As boas práticas de se criar uma migration abrange desde a criação da nomenclatura do arquivo que deve ser de forma descritiva e com um padrão consistente para facilitar a identificação do propósito de cada uma, sutilizar sempre as migrations para alteração das estruturas de tabelas evitando a criação, exclusão ou alteração de um campo de forma despadronizada, que pode levar a uma falha na execução da aplicação , testar as migration localmente antes de aplicar migrations em um ambiente de produção ou compartilhá-las com a equipe, teste-as completamente em seu ambiente de desenvolvimento para garantir que elas funcionam conforme esperado.

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

_____PESQUISADO_____

Uma transação é um conjunto de operações SQL que são executadas como uma única unidade de trabalho. As transações garantem que todas as operações sejam completadas com sucesso ou nenhuma delas seja aplicada, mantendo a integridade dos dados. No SQL Server, você inicia uma transação com o comando BEGIN TRANSACTION e a finaliza com COMMIT (para confirmar as alterações) ou ROLLBACK (para desfazer todas as alterações desde o início da transação).

Savepoints são pontos intermediários definidos dentro de uma transação, que permitem reverter parte de uma transação sem afetar todo o conjunto de operações. Eles são criados

usando o comando SAVE TRANSACTION e permitem fazer rollback para um savepoint específico sem reverter toda a transação.

Sendo bem honesto, não possuo nenhuma familiaridade com SQL SERVER, ao longo dos anos que vivenciei e vivencio em desenvolvimento, posso dizer entre 1998 à 2005 e de 2019 aos dias atuais nunca deparei-me com um projeto com SGBD MSSQL SERVER.

Já desenvolvi projetos com PostgreSQL, MySQL, MariaDB, SQLITE e mais recentemente com INFLUXDB para telemetria.

Por esse motivo não me sinto à vontade em descrever um exemplo de forma honesta.