Atividade Artigo Sequilize (ORM)

-Qual o objetivo desse artigo?

O objetivo desse artigo é propagar informações para futuros desenvolvedores de aplicações que possuem características em comum como as especificadas nesse texto, para que possibilite o aumento da produtividade e facilite o desenvolvimento das mesmas, além de nivelar e padronizar a camada de acesso a dados do sistema.

-Qual metodologia os autores utilizaram?

Programação Orientada a Objetos (OOP)

Utiliza o conceito de objetos e classes para organizar e modularizar o código, facilitando a manutenção e a reutilização.

- O que é um CRUD?

CRUD é um acrônimo que representa as quatro operações básicas de persistência em um banco de dados: Create (criar), Read (ler), Update (atualizar) e Delete (excluir). Essas operações permitem a manipulação completa dos dados em uma aplicação, desde a inserção de novos registros até a leitura, atualização e remoção de informações existentes. CRUD é fundamental no desenvolvimento de aplicações que interagem com bancos de dados, sendo a base para a maioria dos sistemas de gerenciamento de informações.

-Explique o que é o NodeJS, Express, ORM e Sequilize de acordo com o artigo:

Node.js é um ambiente de execução JavaScript que permite rodar scripts do lado do servidor, fora do navegador. Ele é baseado no motor V8 do Google Chrome, que é extremamente rápido, e oferece uma arquitetura orientada a eventos e não bloqueante, ideal para aplicações de alta performance e escalabilidade, como servidores web e APIs.

Express.js é um framework minimalista para Node.js que facilita a construção de aplicações web e APIs, fornecendo ferramentas e funcionalidades como roteamento, middleware, e templates de forma eficiente e flexível.

ORM ou Object Relational Mapping é uma técnica de mapeamento objeto relacional que visa criar uma camada de mapeamento entre nosso modelo de objetos (aplicação) e nosso modelo relacional (banco de dados) de forma a abstrair o acesso ao mesmo.

Sequelize é um ORM (Object-Relational Mapping) para Node.js que facilita a interação com bancos de dados relacionais como MySQL, PostgreSQL, SQLite e MariaDB. Ele permite mapear tabelas do banco de dados em modelos JavaScript, o que simplifica operações CRUD (Create, Read, Update, Delete) e oferece recursos avançados como associação entre modelos, transações e validação de dados. O Sequelize ajuda a abstrair a complexidade do SQL, permitindo que os desenvolvedores trabalhem de forma mais eficiente com bancos de dados.

- Como o Sequilize pode gerar um CRUD?

O Sequelize pode gerar um CRUD (Create, Read, Update, Delete) ao definir modelos que representam as tabelas no banco de dados. A partir desses modelos, você pode utilizar os métodos do Sequelize, como create() para inserir registros, findAll() para buscar dados, update() para modificar registros existentes, e destroy() para deletar dados. Esses métodos abstraem as operações SQL, permitindo que você execute essas ações diretamente com JavaScript, facilitando o desenvolvimento de aplicações com persistência de dados.

-Qual a conclusão que os autores chegaram na escolha dessa tecnologia em resultados e discussão?

Os autores puderam concluir que com o uso de ORMS são gerados resultados concretos que reduzem tempo e consequentemente tempo, porém essas tecnologias possuem pontos fortes como (produtividade e manutenção) e um ponto fraco em especial (performance) em casos onde consultas maiores são geradas.

-Faça uma descrição do resultado obtido na aplicação do framework.

Ao escolher um ORM para seu projeto, é crucial considerar sua performance, pois um ORM lida diretamente com o banco de dados, envolvendo operações de leitura e escrita que podem ser caras. Consultas e gravações desorganizadas podem diminuir a eficiência, enquanto o uso de cache pode melhorar a performance, especialmente para consultas frequentes.

Embora a adoção de ORMs possa introduzir camadas adicionais e impacto na performance, um bom ORM bem configurado pode minimizar a diferença, que geralmente é de milésimos de segundos e não perceptível para os usuários finais. Mesmo com uma pequena queda na performance, o aumento na produtividade da equipe pode compensar esse custo.

Para grandes volumes de dados, como relatórios extensos ou importações massivas, o SQL nativo pode ser mais eficiente que o ORM, evitando a criação de milhares de objetos em memória e reduzindo o tempo de processamento. Em aplicações críticas onde milésimos de segundos são vitais, o acesso direto aos dados pode ser preferível.

Em resumo, é essencial avaliar os prós (produtividade e manutenção) e os contras (performance) do ORM e buscar um equilíbrio que melhor atenda às necessidades do projeto.

-O quê os autores conseguiram concluir com esse estudo realizado neste artigo?

O artigo aborda como os frameworks de ORM facilitam o desenvolvimento de aplicações ao simplificar o gerenciamento de bancos de dados relacionais. No entanto, destaca que é essencial também aprender SQL, especialmente para relatórios complexos que podem exigir SQL nativo. Além disso, sugere a importância de conhecer e se aprofundar no framework ORM escolhido, pois novos frameworks estão constantemente surgindo. O artigo foca em introduzir o conceito de ORM e suas características principais, prometendo detalhes adicionais e exemplos práticos em futuras edições sobre o Entity Framework e o NHibernate.

-Descreva sua experiência em ler um artigo científico e fazer um fichamento orientado pelas questões elaborado pelo professor; foi fácil compreender o artigo?; você acha que suas experiências com desenvolvimento de software é possível escrever um artigo semelhante com tecnologias e diferentes ideias (usar JavaScript com NodeJS e frameworks e padrões como fazemos no laboratório.

Ler o artigo foi uma tarefa muito produtiva e agregou muitas possibilidades na minha jornada, acredito que com o material que já possuímos poderemos sim iniciar artigos futuros com qualidade e coerência.