Trabalho POO 2

Aluno: Joao Lucas Saraiva

Turma: CC4M

01 - Cite pelo menos três linguagens de programação nativamente orientadas a objeto. Quais as suas impressões ao utilizar Typescript frente a elas (vantagens e desvantagens).

R: Três linguagens de programação nativamente orientadas a objeto são:

1. **Java:** Uma linguagem de programação popular, conhecida por sua portabilidade e ampla adoção em desenvolvimento empresarial.
2. **C#:** Desenvolvida pela Microsoft, é amplamente utilizada no desenvolvimento de aplicativos Windows e na criação de aplicações .NET.
3. **Python:** Embora Python seja uma linguagem multiparadigma, suporta programação orientada a objetos de maneira robusta e flexível.

**Impressões sobre TypeScript frente a essas linguagens:**

* **Vantagens:**
  + **Tipagem Estática:** TypeScript fornece tipagem estática, o que pode ajudar a identificar erros em tempo de desenvolvimento.
  + **Superset do JavaScript:** TypeScript é um superset do JavaScript, permitindo a utilização de código JavaScript existente e a adoção gradual.
  + **Interfaces e Tipos Avançados:** Oferece recursos avançados como interfaces e tipos que proporcionam maior robustez ao desenvolvimento.
* **Desvantagens:**
  + **Curva de Aprendizado:** Pode ter uma curva de aprendizado para quem não está familiarizado com tipagem estática.
  + **Overhead de Tipagem:** Adicionar tipos pode aumentar a quantidade de código, potencialmente afetando a legibilidade.
  + **Menos Flexibilidade Dinâmica:** A natureza tipada estática pode ser menos flexível em comparação com linguagens dinâmicas.

02 - O que um padrão de projeto orientado a objeto?

RUm padrão de projeto é uma solução geral para um problema recorrente no desenvolvimento de software. Ele não é um código específico, mas uma descrição ou modelo de como resolver um problema que pode ser adaptado para diversas situações. Padrões de projeto orientados a objeto ajudam a criar designs flexíveis e reutilizáveis.

Exemplos de padrões de projeto incluem o padrão de Observador, Factory Method, Singleton, e muitos outros. Cada padrão aborda um tipo específico de problema e fornece uma solução testada e comumente aceita.

03 - Descreva as principais funções de um ORM (*Object Relational Mapping*).

As principais funções de um ORM (Object Relational Mapping) são:

Mapeamento de Objetos para Tabelas: O ORM mapeia objetos de uma linguagem de programação para tabelas em um banco de dados relacional. Cada classe de objeto é associada a uma tabela e suas propriedades correspondem às colunas da tabela.

Gerenciamento de Conexões e Transações: O ORM cuida da abertura e fechamento de conexões com o banco de dados e permite a execução de transações de maneira eficiente, garantindo a consistência dos dados.

Consulta de Dados Simplificada: Oferece uma linguagem de consulta orientada a objetos que permite aos desenvolvedores recuperar dados do banco de dados usando uma sintaxe familiar, em vez de escrever consultas SQL manualmente.

Mapeamento de Relacionamentos: Permite definir e mapear relacionamentos entre objetos, como associações um-para-muitos, muitos-para-muitos e assim por diante, tornando a manipulação de dados relacionais mais intuitiva.

Geração de Esquema Automático: Pode criar automaticamente o esquema do banco de dados com base nas definições das classes de objetos, economizando tempo e esforço no gerenciamento de esquemas.

Cache de Objetos: Mantém um cache de objetos em memória para evitar consultas excessivas ao banco de dados, melhorando o desempenho e a eficiência das operações de leitura.

Suporte a Herança e Polimorfismo: Permite o mapeamento de hierarquias de classes e suporta conceitos de herança e polimorfismo de forma natural.

Validação de Dados: Pode incluir recursos de validação de dados, garantindo que apenas dados válidos sejam armazenados no banco de dados.

Portabilidade de Banco de Dados: Geralmente é compatível com diferentes sistemas de gerenciamento de bancos de dados, permitindo que o código seja mais portátil entre diferentes plataformas de banco de dados.

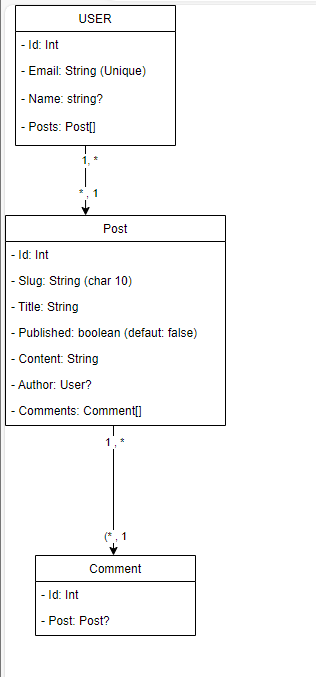
Facilita a Manutenção do Código: Abstrai a complexidade das interações com o banco de dados, tornando o código mais limpo, organizado e de fácil manutenção.

No geral, um ORM simplifica a interação entre um aplicativo orientado a objetos e um banco de dados relacional, tornando o desenvolvimento de software mais produtivo e menos propenso a erros.

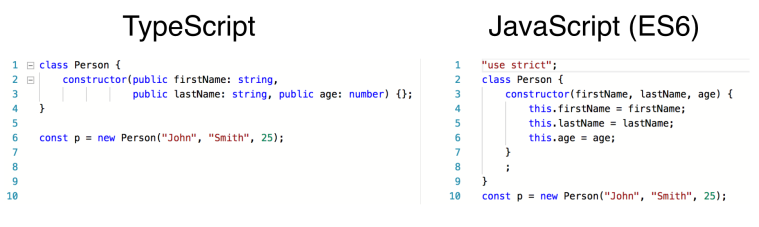
04 - Analise a imagem abaixo, com um modelo do Prisma:

Desenhe um diagrama de classes da UML válido para o diagrama.

R:



05 - Analise a imagem a seguir e descreva as vantagens de se utilizar Typescript frente à linguagem Javascript para programação orientada a objetos:

  
Vantagens de usar TypeScript em relação ao JavaScript no exemplo dado:

* **Tipagem Estática Explícita:** No TypeScript, a tipagem estática é explicitamente declarada, como pode ser visto na definição do construtor da classe "Person." Isso torna claro quais tipos de dados são esperados para cada parâmetro, tornando a tipagem mais segura e fácil de entender.
* **Detecção de Erros em Tempo de Compilação:** O TypeScript verifica os tipos em tempo de compilação, o que significa que quaisquer erros de tipo, como passar um tipo incorreto de argumento para o construtor, são identificados antes mesmo de o código ser executado. Isso ajuda a evitar erros difíceis de depurar.
* **Intellisense e Autocompletar:** Em ambientes de desenvolvimento como o Visual Studio Code, o TypeScript oferece Intellisense e autocompletar, fornecendo sugestões e informações sobre tipos de dados durante a escrita do código, o que melhora a produtividade.
* **Manutenção Simplificada:** Com a tipagem estática, a manutenção do código é facilitada, pois você tem um entendimento claro da estrutura do código e das expectativas de tipos.
* **Documentação Autogerada:** O TypeScript permite adicionar anotações de tipo, que podem ser usadas para gerar automaticamente documentação mais rica e descritiva, ajudando a documentar o código de maneira eficaz.
* **Maior Clareza e Legibilidade:** A tipagem explícita no TypeScript torna o código mais claro e legível, facilitando a compreensão, especialmente quando várias pessoas colaboram no projeto.
* **Segurança e Redução de Erros:** A detecção de erros em tempo de compilação no TypeScript ajuda a evitar bugs comuns, tornando o código mais seguro e robusto.
* **Facilita a Refatoração:** O TypeScript oferece suporte sólido para refatoração de código, permitindo que você faça alterações com confiança, sabendo que a tipagem está correta.
* **Adoção Gradual:** Você pode começar a usar TypeScript gradualmente em projetos JavaScript existentes, introduzindo tipos em partes específicas do código, facilitando a transição.