1. Quais problemas podem ocorrer durante a transformação do modelo de domínio em código?

R: Sem um modelo de domínio bem definido e documentado, a transformação em código pode não refletir completamente as necessidades do negócio, o que pode gerar retrabalho. Além disso, se houver entidades replicadas ou mal definidas, isso também pode gerar problemas na implantação.

2. Descreva pelo menos três preceitos do MDD.

R: O MDD tem três preceitos principais: 1) os modelos de software são a fonte primária de informações para o desenvolvimento de software, 2) a geração automática de código é preconizada para reduzir erros e aumentar a produtividade dos desenvolvedores, e 3) os modelos podem ser reutilizados para economizar tempo e esforço e estabelecer padrões para o desenvolvimento de software.

3. Explique os relacionamentos entre os principais padrões do Domain-Driven Design (DDD) representados na figura do slide “Blocos de Construção do MDD” apresentado na aula.

R: Esses padrões do DDD estão relacionados e são frequentemente usados em conjunto para criar modelos de domínio eficazes e flexíveis. Por exemplo, um repositório pode ser usado para armazenar e recuperar entidades, um serviço de domínio pode ser usado para executar operações complexas que envolvem várias entidades e agregados, e um agregado pode ser usado para agrupar entidades relacionadas e garantir que as alterações nessas entidades sejam feitas de maneira consistente.

4. Em uma arquitetura em camadas, cada camada depende predominantemente da camada imediatamente inferior. No entanto, na figura do slide “Arquitetura em Camadas” apresentado em aula existem setas entre as seguintes camadas: User Interface -> Domain, User Interface -> Infrastructure e Application -> Infrastructure. Dê um exemplo de dependência que cada uma dessas setas estaria representando.

R:

User Interface -> Domain: A camada de interface do usuário depende da camada de domínio para acessar as regras de negócios e funcionalidades específicas do domínio. Por exemplo, em um sistema de e-commerce, a interface do usuário depende da camada de domínio para acessar as regras de negócios relacionadas à venda de produtos, como verificação de estoque, cálculo de preço e validação de pagamento.

User Interface -> Infrastructure: A camada de interface do usuário depende da camada de infraestrutura para acessar serviços externos, como bancos de dados e APIs de terceiros. Por exemplo, em um aplicativo de previsão do tempo, a interface do usuário depende da camada de infraestrutura para acessar dados meteorológicos de um serviço externo.

Application -> Infrastructure: A camada de aplicação depende da camada de infraestrutura para acessar serviços externos e recursos compartilhados, como bancos de dados e sistemas de cache. Por exemplo, em um sistema de gerenciamento de estoque, a camada de aplicação depende da camada de infraestrutura para acessar um banco de dados compartilhado que armazena informações sobre estoque e pedidos

5. Na aula foram dados alguns exemplos de entidade (pessoa, conta bancária, aeroporto). Dê outros três exemplos de entidade, indicando seus possíveis atributos identificadores.

R:

Pedido de compra: Um pedido de compra pode ter atributos identificadores como número do pedido, data do pedido e o nome do cliente que fez o pedido.

Paciente em um hospital: Um paciente em um hospital pode ter atributos identificadores como nome, número de identificação do paciente e número do quarto onde ele está hospedado.

Produto em um estoque: Um produto em um estoque pode ter atributos identificadores como código do produto, descrição do produto e quantidade de estoque disponível.

6. Descreva três diferenças entre entidades e value objects.

R:

Identidade única: Entidades possuem identidade única, enquanto Value Objects não possuem identidade única

Mutabilidade: Entidades são mutáveis, enquanto Value Objects são imutáveis.

Participação em Agregados: Entidades são geralmente o elemento central dos Agregados, enquanto os Value Objects são usados para representar propriedades de uma Entidade ou Agregado

7. Por que serviços não podem ser tratados como entidades e value objects?

R: Serviços são conceitos que descrevem comportamentos ou ações que podem ser realizados em um sistema, enquanto Entidades e Value Objects são representados por objetos que possuem estado e comportamento. Por essa razão, os serviços não podem ser tratados como Entidades ou Value Objects, pois não possuem identidade única nem são intercambiáveis, e não possuem estado interno.

8. Como os aggregates podem ajudar na integridade dos dados e na aplicação das invariantes?

R: Os Agregados agrupam Entidades e Value Objects relacionados em uma unidade coesa e consistente, permitindo que suas operações sejam gerenciadas em um único ponto de controle. Eles ajudam a garantir a integridade dos dados e a aplicação das invariantes por meio do encapsulamento, transações atômicas e restrições de acesso.

9. Explique a relação entre aggregates e factories.

R: As Factories são responsáveis por criar instâncias de objetos, incluindo Entidades e Value Objects que fazem parte de um Agregado, e garantir que essas instâncias pertençam ao Agregado correto e respeitem as invariantes de negócio. As Factories podem ser usadas para criar instâncias de Agregados inteiros, garantindo que todos os objetos necessários para o Agregado estejam presentes e corretamente configurados.

10. O que factories e repositories têm em comum? No que eles diferem?

R: As Factories e Repositories são padrões comuns na engenharia de software para lidar com a criação e armazenamento de objetos. As Factories criam novas instâncias de objetos, enquanto os Repositories armazenam e recuperam objetos de um sistema de armazenamento. Factories geralmente são usadas para criar objetos temporários ou gerenciar a criação de objetos complexos, enquanto os Repositories são usados para armazenar objetos permanentemente e recuperá-los quando necessário.