**1 – O que a camada lógica de negocio engloba? Como ela se relaciona com as demais camadas? O que tem a ver os modelos construídos na fase de análise?**

A camada de lógica de negócio engloba o conjunto de classes que vai realizar toda a lógica do sistema de informação. As demais camadas são derivadas ou dependentes dessa camada e, portanto, é interessante iniciar o projeto dos componentes da arquitetura do sistema por ela. Os modelos construídos na fase de análise são os principais insumos para o projeto dessa camada, em especial o modelo conceitual estrutural e o modelo de casos de uso.

**2 – Com relação à escolha do padrão para organização da lógica de domínio, cite RESUMIDAMENTE do quê depende a escolha do padrão. Comente RESUMIDAMENTE cada um dos 4 padrões avaliados em sala**

A escolha do padrão depende da complexidade da logica de domínio.

Os quatro tipos de padrão:

Transaction Script

* Desenvolvimento Rapido
* Menos Camadas de abstração

Table Module

* Complexidade intermediaria
* Funciona bem em frameworks orientadas ao Record Set.

Domain Model

* Puramente orientado aos objetos
* Mais complexidade mas e mais flexível

Server Layer

* Interface funcional com as camadas superiores
* Útil quando combinado com domain model , ou table module

**3 – O que é o Padrão DDD? Qual sua ideia básica? Como é na prática? O que ele têm a ver com a modelagem do domínio real?**

Trata-se de uma metodologia de design de software que tem um foco no que está acontecendo no domínio da aplicação. Seu foco é na modelagem das entidades principais de negócio usando a linguagem adequada daquele domínio para facilitar a manutenção, extensão e entendimento. Em outras palavras, e como o nome sugere, o design é centrado na lógica de negócios (domínio) do software.

É a construção do software a partir da modelagem do domínio real como classes de domínio que, por sua vez, possuirão relacionamentos entre elas.

**4 – Cite e descreva 5 alterações básicas a serem incorporadas em um diagrama de classes do CDP**

Alteração de informações relativas a tipos de dados de atributos: Na fase de análise, é comum especificar tipos de dados gerais para atributos. Na fase de projeto, contudo, os atributos devem ser mapeados em variáveis de um tipo de dados provido pela linguagem de implementação.

Adição de navegabilidades nas associações: Na fase de análise, as associações são consideradas navegáveis em todas as direções. O mesmo não ocorre na fase de projeto, quando se pode definir que certas associações são navegáveis apenas em um sentido, indicando que apenas um dos objetos terá uma referência para o outro (ou para coleções de objetos, no caso de associações com multiplicidade).

Adição de informações de visibilidade de atributos e associações: De maneira geral, na fase de análise não se especifica a visibilidade de atributos e associações. As associações são tipicamente consideradas navegáveis e visíveis em todas as direções. Já os atributos são considerados públicos. Porém essa não é uma boa estratégia para a fase de projeto.

Adição de métodos às classes: Muitas vezes, as classes de um diagrama de classes de análise não têm informação acerca das suas operações. Mesmo quando elas têm essa informação, ela pode ser insuficiente, tendo em vista que é no projeto que se decide efetivamente como abordar a distribuição de responsabilidades para a realização de funcionalidades.

Eliminação de classes associativas: Caso o diagrama de classes de análise contenha classes associativas, recomenda-se substituí-las por classes normais, criando novas associações. Isso é importante, pois as linguagens de programação não têm construtores capazes de implementar diretamente esses elementos de modelo.

**5 – Cite e descreva as situações de quando não se deve usar o DDD.**

1 - Às vezes só é necessário um CRUD DDD não é uma solução para tudo. A maioria dos sistemas possui uma boa parte composta por cadastros básicos (CRUD) e não seria adequado usar DDD para isso. O DDD deve ajudar na modelagem das classes mais importantes e mais centrais do sistema de forma e diminuir a complexidade e ajudar na manutenção das mesmas, afinal este é o objetivo dos princípios de orientação a objetos.

2 - Compartilhando dados com outros sistemas Rotinas de integração que recebem ou disponibilizam dados para outros sistemas não devem ser "inteligentes". Muitos desenvolvedores acabam modelando suas classes de negócios tentando resolver as questões internas do sistema e, ao mesmo tempo, pensando em como essas classes serão expostas para outros sistemas. Padrões como DTO (Data Transfer Object) que usam objetos "burros" são mais adequados para isso.

**6 – O que é uma Camada de Persistência de Objetos ?**

Conceitualmente, uma Camada de Persistência de Objetos é uma biblioteca que permite a realização do processo de persistência (isto é, o armazenamento e manutenção do estado de objetos em algum meio não-volátil, como um banco de dados) de forma transparente.

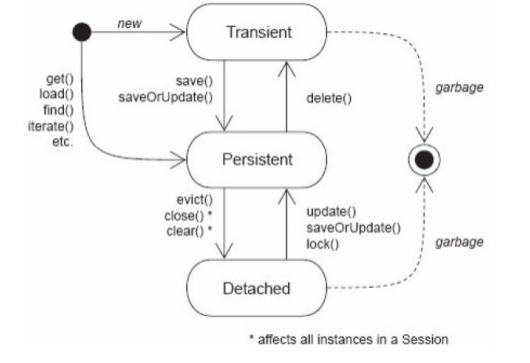
**8 – O que são OIDs? O que é recomendado quando ocorre o mapeamento?**

Os OIDs são identificadores únicos que representam um objeto, em linguagens de programação, existentes, este objeto é implícito e criado quando ocorre a criação de um novo objeto, já em um banco de dados relacional cabe ao desenvolvedor a responsabilidade desta criação. Quando ocorre o mapeamento, recomenda-se armazenar no banco de dados relacional o identificador do objeto como chave primária, ou qualquer outro atributo do objeto que possa identifica-lo como único

**9 – O que é o Hibernate? Qual suporte ele oferece em relação a mapeamentos?**

O Hibernate é um framework de mapeamento objeto relacional para aplicações Java, ou seja, é uma ferramenta para mapear classes Java em tabelas do banco de dados e viceversa. É bastante poderoso e dá suporte ao mapeamento de associações entre objetos, herança, polimorfismo, composição e coleções.

**10 – Demonstre um diagrama com O ciclo de vida de um objeto persistente**



**11 – Descreva a função (resumidamente) de cada camada MVC.**

• Modelo: – Representa a lógica do negócio, – Armazena as entidades do sistema, – Pode interagir com a base de dados,

• Visão: – Responsável por exibir as informações ao usuário

• Controle: – Interliga a Visão e o Modelo

**12 – Descreva cada camada do Three-Tier Architecture**

Camada de apresentação : Contem toda a interfaçe gráfica e permite a interação com o usurário por meio dos seus serviços disponíveis ao usuário

Camada logica: Contem toda a logica do negocio, bem como a logica de transações

Camada dos Dados : Contem os dados que são manipulados pela aplicação , bem como o acesso a dado, atualização e persistência destes .

**13 - Sobre o JSP: O que ele é? Como ele é considerado? O que ele suporta? Podem ser recompilados?**

JSP é um template de uma página Web que usa Java para gerar HTML dinamicamente, JSP é considerado server-side e roda em um objeto chamado container, que os transformam em Servlets, Suporta a parte de rede, acesso a banco de dados e threads do Java, São recompilados automaticamente quando necessário

**14 – O que são Servlets? Possuem interface com o usuário? Podem responder requisições HTML? O que tem a ver com o JSP? Cite 4 vantagens dos Servlets**

São componentes server-side escritos em Java e independentes de plataforma e de protocolos:

• Não possuem interface com o usuário pois são executados dentro do servidor Web,

• Podem responder requisições HTML ou construir páginas HTML dinamicamente,

• Lembrando que internamente, no servidor Web, todo JSP é traduzido em um Servlet