A1 – APLICANDO CONHECIMENTO

COMPONENTE CURRICULAR:	DESENVOLVIMENTO DE SISTEMAS II
INTEGRANTES DO GRUPO:	JOÃO PEDRO LIMA LUSTOSA AMORIM
RA:	10289920

1. **Padrões arquiteturais**

   Escolha 4 padrões arquiteturais apresentados em aula (listados abaixo) e realize uma síntese, explicando-o.

   R:

   - **Client/server:** Este tipo de estrutura é um modelo de arquitetura de rede no qual os computadores se dividem em dois grupos: os **servidores** e os **clientes**. Os computadores do tipo servidores fornecem serviços ou recursos solicitados pelos clientes, podendo estes serem projetados especificamente para esta atividade. A comunicação entre os dois é feita através de protocolos de rede, geralmente realizada pela *internet* ou por uma rede privada, sendo assim um modelo altamente escalável, flexível e seguro, permitindo que as empresas distribuam seus serviços e recursos em vários dispositivos e plataformas. A estrutura ***cliente/server*** permite que as tarefas sejam distribuídas entre o cliente e o servidor, sendo esta comunicação baseada em trocas de mensagens, podendo estas serem síncronas ou assíncronas, e não necessariamente ocorre somente entre um cliente ou um servidor, é possível adicionar vários clientes e vários servidores. Este tipo de estrutura se torna eficaz e segura devido ao fato do servidor ser responsável por gerenciar os recursos compartilhados, sendo assim confiável por também fornecer recursos de controle de acesso para os clientes determinando ao servidor a gerência de acesso à recursos específicos para diferentes usuários ou grupos de usuários e garantindo a reutilização segura de códigos em diferentes aplicativos e serviços.
   - **Service-Oriented Architecture (SOA):** A arquitetura orientada a serviços (SOA) é um método de desenvolvimento de software que utiliza componentes de software

chamados de serviços para criar aplicações de negócios. Cada serviço fornece um recurso de negócios e todos eles também podem se comunicar entre si em diferentes plataformas e linguagens. Os desenvolvedores usam SOA para reutilizar serviços em sistemas diferentes ou combinar vários serviços independentes para realizar tarefas complexas. Este método poupa tempo e tem baixo custo-benefício, tornando-se mais rápido para a entrada no mercado. Devida a alta eficiência, tarefas como atualizar, criar ou depurar pequenos serviços se tornam mais fáceis sem afetar a funcionalidade geral do processo de negócios, sendo mais fácil de se adaptar ao avanço tecnológico.

- **P2P (*Peer-to-peer*):** esta estrutura é uma arquitetura de redes em que cada par ou nó coopera entre si para prover serviços um ao outro, sem a necessidade de um servidor central. Todos os pares são clientes e servidores. A estrutura P2P, diferentemente da estrutura cliente/server, se conecta através de uma rede onde a informação é distribuída e está sempre ativa, não requer clientes e servidores fixos e não há nós com funções especiais, todos eles são iguais ou podem ter a mesma função.
- **Pipes-and-filters (PF):** nesta arquitetura os componentes computacionais são filtros que agem como transdutores que recebem uma entrada, transformam de acordo com um ou mais algoritmos e geram uma saída para um canal de comunicação. Os filtros devem ser componentes independentes a fim de se adaptar em qualquer posição para obtermos resultados diferentes. Como todos os componentes utilizam a mesma interface, eles podem ser compostos em diferentes pipes. Por ser uma estrutura flexível, novos filtros podem ser adicionados, outros existentes podem ser omitidos ou também é possível encadear de formas diferentes dentro de uma nova sequência sem precisarmos alterar os filtros.

2. **Estilos arquiteturais**

Conforme apresentando em aula, existem diversos estilos arquiteturais. Dentre os listados a seguir, escolha 4, pesquise sobre e elabore uma síntese e forneça exemplos de aplicações que podem ser beneficiadas por estes padrões: Database-centric, Monolithic application, Layered, Pipes and filters, Event-driven, Publish-subscribe, Plug-ins.

R:

- **Database-centric:** este estilo de arquitetura tem o banco de dados como o centro da aplicação e a partir dele, gera demandas, inovações, busca recursos e alcança clientes. Empresas centradas em dados tratam os dados com ativos permanentes e compartilhados, as aplicações e algoritmos mudam mais rápido e são temporários. Aplicações com sistemas de ERP que utilizam grandes volumes de dados centralizados são beneficiadas pelo *Database-centric*.
- **Layered:** este tipo de arquitetura é dividida em camadas distintas e independentes destinadas a uma função específica. Cada camada se comunica apenas com as camadas adjacentes, seguindo uma estrutura hierárquica. Essa abordagem facilita a manutenção, a escalabilidade e a reutilização de códigos. É possível encontrar este tipo de arquitetura nas aplicações webs como e-commerces e aplicações bancárias online.
- **Event-Driven:** este tipo de arquitetura é feito para capturar, comunicar e processar eventos entre serviços desacoplados, podendo permanecerem assíncronos enquanto compartilham informações e realizam tarefas. É possível encontrar este tipo de arquitetura em sistemas de IoT (*Internet of Things*) como dispositivos de detectação de temperatura ou assistentes como as Echo-Dots.
- **Publish-Subscribe:** este tipo de padrão permite a um aplicativo anunciar eventos para vários consumidores de seu interesse assincronamente, sem acoplar os remetentes aos destinatários. Os consumidores se inscrevem para receber notificações de eventos específicos, promovendo um sistema desacoplado. Um exemplo desse tipo de padrão são os serviços de push nos quais um usuário (por exemplo no WhatsApp ou Instagram) recebe notificação todas as vezes que um perfil ao qual ele está inscrito faz uma publicação.

3. **Princípios arquiteturais**

Conforme apresentado em aula, estivem alguns princípios arquiteturais em processos de desenvolvimento de software, sendo eles o Solid, economics e least. Elabora uma síntese de cada um.

R: SOLID é um acrônimo que representa os cinco princípios da programação orientada a objetos: **SRP** (*Single Responsibility Principle*), **OCP** (*Open/Closed*

Principle), **LSP** (*Liskov Substitution Principle*), **ISP** (*Interface Segregation Principle*), **DIP** (*Dependency Inversion Principle*). Esses princípios são a base de vários padrões de projetos e tornam software mais evolutivos, de fácil manutenção e adaptabilidade, tornando-o assim fácil de ser atualizado. A falta do SOLID acarreta em problemas como repetição de código, falta de coesão ou padronização no código, rigidez e fragilidade, aumentando assim o número de falhas, dificulta a execução e criação de testes e por consequências destes fatos, não é possível reutilizar o código em outros sistemas.

O princípio *Economics* ou princípio do custo de oportunidade parte da escolha na qual o foco principal é respeitar o limite orçamentário para o desenvolvimento de uma aplicação ou software. Este princípio se divide entre Microeconomia no qual o foco está no comportamento de agentes individuais e na formação de preços no mercado, e a Macroeconomia que estuda a economia como um todo.

Least Privilege ou princípio da Regra do Mínimo sugere uma abordagem na qual um usuário deve ter acesso apenas ao que é absolutamente necessário para desempenhar suas responsabilidades e nada mais. Quanto maior o acesso do cliente aos ambientes de desenvolvimento do sistema ou da aplicação, maior o impacto negativo caso ele se torne uma ameaça ou se sua conta for comprometida. Um sistema de TI onde os usuários têm permissões mínimas necessárias para realizar suas tarefas.