Arquitetura de Software

- O que é?

De acordo com Camila Pessôa: É a “Estrutura fundamental ou esqueleto de um sistema de software, que define seus componentes, suas relações e seus princípios de projeto e evolução”. Em resumo, nada mais é do que um esboço daquilo que é essencial no nosso sistema, e vem como consequência dos **Requisitos de software.**

- Para que serve?

Considerando o tópico anterior, depois de definidas as nossas prioridades/requisitos, precisamos saber como vamos encaixá-las e delimitá-las no nosso projeto. Tal como um arquiteto faz um esboço de uma casa para saber os ambientes de convivência (sala de estar, cozinha), trabalho(Escritório), descanso (sala de estar e quartos), e cuidado pessoal (banheiro, lavabo, …), assim também o arquiteto de software deve fazer esse mesmo trabalho. Detalhe, não necessariamente quer dizer que o computador vai construir um quarto dentro de um computador. Na verdade, por meio de técnicas de programação (como a Programação Orientada a Objeto (**POO**), o uso de bibliotecas), poderemos fazer esse “desenho” do nosso sistema. A junção deste com outras técnicas, gerará uma Arquitetura Orientada a Serviços (SOA). Segundo Pessôa, “Sabemos que (a arquitetura de software) não se limita à design do software, ao código em si, bibliotecas, frameworks, hardware, metodologias ágeis ou princípios de desenvolvimento. No entanto, é interessante notar que todos esses elementos e escolhas são determinantes para definir uma boa arquitetura. […] funciona como um ‘tudo em todo lugar ao mesmo tempo’ organizado a fim de evitar caos.”

- Como faço?

Para aplicarmos, precisamos de **Padrões em arquitetura de software**, para facilitar nosso entendimento do código e evitar custos e desgastes desnecessários.

-> Arquitetura client-server:

1. *Client* (desktop): Responsável por manter a interface com o usuário e um ou outro código de aplicação. Delphi e VB são clientes do SQL, por exemplo.
2. Server (servidor): um banco de dados relacional. SQL é um exemplo. Aqui, existe a necessidade de, para evitar erros e procedimentos desnecessários, de usamos a arquitetura em camadas para evitar o acesso indevido e desnecessário do usuário a partes frágeis do sistema e manter 3 camadas de funcionamento simultâneo em que se precisa apenas de 1 dispositivo.

-> Layers (Arquitetura em camadas)

Na arquitetura em camadas, o sistema é desenvolvido com camadas que são independentes em termos de modificação, mas a camada anterior é dependente dos serviços e funcionalidades da camada atual. Ou seja, são parcialmente dependentes, pois o funcionamento de uma depende da próxima camada. É recomendada para o caso de manutenção em softwares que já existem ("Você pegou o bonde já andando"), ou no caso em que diferentes equipes desenvolvem de forma isolada cada parte do sistema.

-> MVC (mais famosa):

Na arquitetura Model-View-Controller (MVC), os sistemas são desenvolvidos nessas três camadas. O model estabelece regras de negócio e a interação entre o back-end e o banco de dados (recepção ou envio de dados). O view define e gerencia como os dados são apresentados ao usuário (parte ligada fortemente ao front-end). Por último, o controller é responsável por ser o nível intermediário entre o model e o view, ao receber dados do usuário, e os manipula e reage de acordo com a necessidade do programador.

-> SOA:

Na arquitetura orientada a serviço, divide-se o sistema em serviços independentes, e depois, criam-se interfaces para que os serviços interajam entre si. Em analogia, estamos preparando um prato de almoço. Onde os serviços são, por exemplo, a região da proteína, a região do carboidrato, e a fibrosa. A interface seria, portanto, a própria louça que abriga essas 3 regiões do seu prato de almoço.

-> Pipes and Filters:

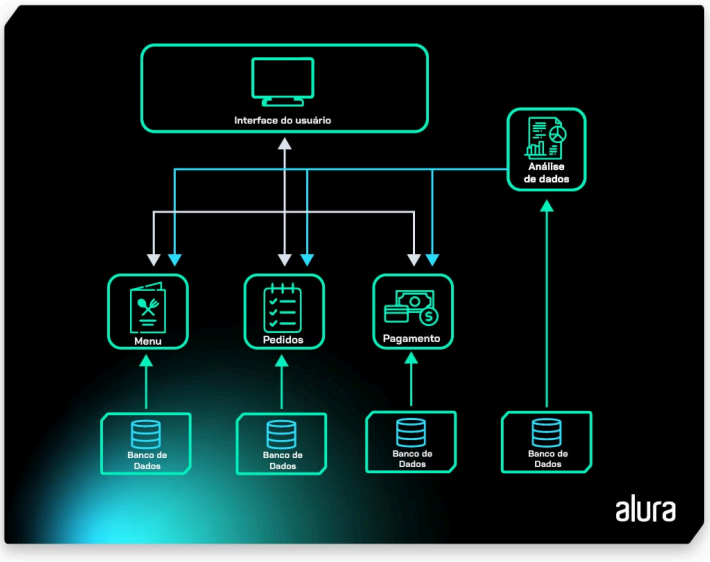
Nessa, o dado recebido pelo sistema e cada componente independente do sistema (chamado filtro), opera de forma específica dentro do filtro. Se pensar no caso de se passar um café, o dado é o pó de café, e o filtro, ele interage com o pó, de forma a transformar o pó em um café. Mas para que o pó seja transformado em café, é preciso que o nosso líquido seja conduzido por um duto. Podemos aplicá-lo, por exemplo, para criar um agente de IA que identifica e-mails que são SPAM, já que o agente recebe o email, o manipula até que o simplifiquemos a ponto criar parâmetros simples para identificar spam.

-> Monolítica:

Na arquitetura monolítica, criamos apenas um bloco onde os serviços são interdependentes, ou seja, sua manutenção é mais complexa. A interação entre serviços/métodos é feita de forma vertical. Pode haver modulação nesse tipo de sistema, para haver maior facilidade na manutenção do código. Exemplo famoso: Amazon Prime Video e Github.

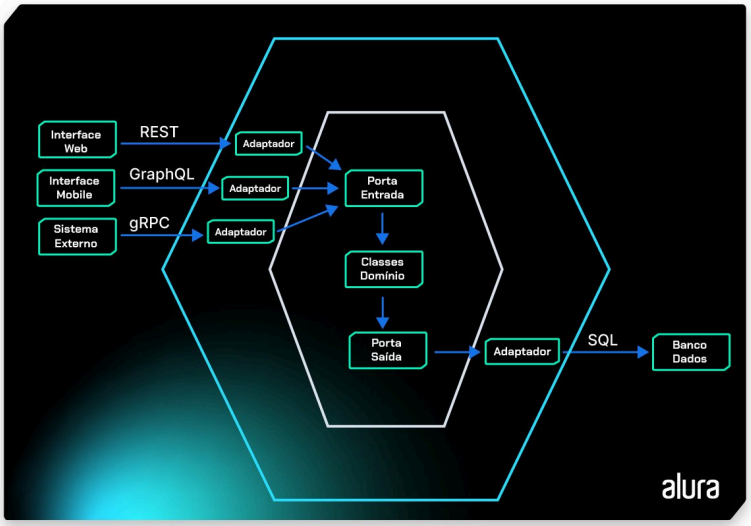
-> Baseada em microsserviços:

Considerada uma variante da arquitetura orientada a serviços, a arquitetura baseada em microsserviços é um tipo muito útil para arquitetura em que se busca uma independência maior em termos de serviços, e mais descentralizada do que a arquitetura monolítica.



-> Hexagonal:

A arquitetura hexagonal é uma arquitetura conhecida por seu desacoplamento e fácil manutenção, pois há grande descentralização dos serviços. Segundo Pessôa: “As portas são as interfaces que permitem que a lógica de de negócios se comunique com o mundo externo, e os adaptadores são responsáveis por implementar as portas.” Um ponto muito positivo dessa arquitetura é a possibilidade de se incrementar outras arquiteturas, devido ao grande desacoplamento. No entanto, a escritora supracitada diz que é preciso que tenhamos um núcleo bem definido para que a implementação dessa estrutura seja eficaz. Camila Pessôa ainda acrescenta que: “Essa arquitetura traz uma perspectiva diferente sobre a ordem das camadas de uma aplicação, em que normalmente temos a primeira camada como a interface e a última como um banco de dados para armazenamento. A ideia é entender front-end e back-end como camadas externas ao seu domínio!”



Fontes:

https://www.alura.com.br/artigos/padroes-arquiteturais-arquitetura-software-descomplicada?srsltid=AfmBOorMLZRV5h5qMfdL1TBjLXD9K4Ksb_zCpn3t_CB6sFm6s8JMTsLG