**Layers de uma Aplicação Web**

Além do básico Client-side e Server-side, uma aplicação web também se divide em camadas (layers), são elas:

* Presentation Layer (PL)
* Data Service Layer (DSL)
* Business Logic Layer (BLL)
* Data Access Layer (DAL)

**Presentation Layer (PL)**

Esta é a camada que o usuário final vê e interage. É a interface, onde os dados estão sendo exibidos em tela de uma forma legível. Nesta camada é onde entram os dados fornecidos pelo usuário (por exemplo um input de texto) e são enviados para as camadas de dados.

**Data Service Layer (DSL)**

É a camada responsável por enviar os dados processados na Business Logic Layer para a Presentation Layer. Ela deve ser responsável também por garantir a segurança destes dados.

**Business Logic Layer (BLL)**

Esta camada garante a troca correta de dados e controla a funcionalidade da aplicação. É a camada que define a lógica para operações de negócios, regras de negócios e satisfaz as necessidades de negócios. Ela compreende os dois lados, ajustando informações de negócio em forma de dados e vice-versa.

**Data Access Layer (DAL)**

Nesta camada é onde ocorre a transação de informações e dados entre o serviço de persistência (Normalmente um banco de dados) com a camada de negócios. Somente esta camada deve ter acesso direto a qualquer SGDB.

Por vezes, podemos ver 3 camadas apenas, a Presentation Layer, a Business Layer (que compreende a DSL e a BLL) e a Persistence Layer (que é igual a DAL)

Uma aplicação web costuma usar uma arquitetura de 3 níveis, conforme citado logo acima.

**Tipos de Arquitetura Web**

As mais comuns são:

* Single-page web apps (SPA)
* Multi-page web apps
* Microservices architecture
* Serverless architecture
* Progressive web apps (PWA)

**Single-page App (SPA)**

SPAs carregam todo conteúdo necessário para seu funcionamento ao carregar a página pela primeira vez. Se baseiam em apenas uma página web, onde o conteúdo vai sendo alterado sem necessário de reloads ou navegações para outras páginas. O primeiro carregamente costuma ser mais lento, mas a navegação dentro da página costuma ser muito rápida e eficiente para o usuário. React, Vue, Angular costumam basear seus frameworks nessa arquitetura.

**Multi-page App**

É o mais comum em aplicações muito pesadas e grandes, onde cada página de conteúdo está separada e carrega suas necessidades quando é acessada. Você navega entre uma página e outra diretamente pelo browser.

**Microservices**

Tem a ideia de separar as diversas funcionalidades da aplicação em pequenos módulos e serviços, utilizando-os quando necessário e permitindo maior facilidade de modificar código de forma escopada sem prejudicar outras páginas ou módulos.

**Serverless**

Nesse formato, não é que a aplicação não use um servidor, mas sim que o servidor está hospedado em um serviço na nuvem, tal como AWS (Amazon) ou Azure (Microsoft), garantindo que se a aplicação é muito pesada e você não tem como manter seu próprio servidor, você delega a hospedagem para um terceiro.

**Progressive Web App (PWA)**

É uma forma de criar aplicações para dispositivos utilizando linguagem de front-end, porém apresentada de forma semelhante a um aplicativo móvel. Podem se beneficiar de APIs do browser que permitem acesso a conteúdos nativos de celulares, por exemplo, mas precisam ser corretamente configurados para serem passíveis de instalação.