DESENVOLVIMENTO WEB – FRONT END

AULA 1

Prof. Mauricio Antonio Ferste

CONVERSA INICIAL

Este é o primeiro de seis capítulos destinados a tratar da WEB. Basicamente, é um contexto muito extenso, povoado de várias e instigantes tecnologias, e tudo graças ao Senhor Tim Berners-Lee, que em 1990 criou o *WWW*.

Figura 1 – Sr. Tim Berners-Lee

Créditos: drserg/Shutterstock.

O conceito inicial da proposta é bem simples. Em resumo, para ser um sistema “web”:

- deve possuir um identificador único (endereço, URL);
- a estrutura de apresentação deve ser escrita em HTML*;
- a transmissão de dados pela rede deve respeitar o protocolo HTTP.

Neste contexto, Tim também escreveu o primeiro editor/navegador de páginas web (“WorldWideWeb.app”) e o primeiro servidor web (“httpd”). No final de 1990, a primeira página da web foi servida na internet aberta e, em 1991, pessoas de fora do CERN foram convidadas a ingressar nessa nova comunidade da web. À medida que a web começou a crescer, Tim percebeu que seu

verdadeiro potencial só seria liberado se alguém, em qualquer lugar, pudesse usá-la sem pagar uma taxa ou ter que pedir permissão. Ele explica: “Se a tecnologia fosse proprietária e estivesse sob meu controle total, provavelmente não teria decolado. Você não pode propor que algo seja um espaço universal e ao mesmo tempo manter o controle dele”. Assim, Tim e outros defenderam a garantia de que o CERN concordaria em disponibilizar o código subjacente sem royalties, para sempre. Esta decisão foi anunciada em abril de 1993 e desencadeou uma onda global de criatividade, colaboração e inovação nunca antes vista. Em 2003, as empresas que desenvolviam novos padrões da web se comprometeram com uma política de isenção de royalties para seu trabalho. Em 2014, ano em que comemoramos o 25º aniversário da web, quase duas em cada cinco pessoas em todo o mundo a usavam. Tim mudou-se do CERN para o Massachusetts Institute of Technology em 1994 para fundar o World Wide Web Consortium (W3C), uma comunidade internacional dedicada ao desenvolvimento de padrões abertos da web. Ele continua sendo o Diretor do W3C até hoje.

Então, chegando aos dias atuais, podemos tratar da WEB e suas evoluções.

Importante!

Trataremos de várias tecnologias WEB, algumas mais, outras menos. Este estudo pressupõe que o leitor já teve algum contato com HTML, CSS e JavaScript e, embora façamos explicações quando necessário, para HTML e CSS elas serão superficiais, mais a nível de revisão.

Principalmente, vamos entender aqui, nesta primeira unidade, o contexto de desenvolvimento baseado em frameworks. Não veremos códigos ainda; a ideia é tratar de entendimento teórico sobre arquitetura e ferramentas que envolvem o desenvolvimento.

Vamos lá!

TEMA 1 – INTRODUÇÃO AO FRAMEWORK: O QUE É, TERMINOLOGIA E TERMOS

Um framework vem com vários recursos extraordinários e geralmente já possui um fluxo de trabalho ou estrutura a seguir, explicado. É algo muito mais abstrato do que uma biblioteca. Isso realmente confunde muita gente. Por

exemplo, o jQuery já foi chamado de *biblioteca*, mas não é chamado assim há muito tempo. Os frameworks têm um foco mais amplo do que as bibliotecas. Caso contrário, uma estrutura pode ser criada a partir de uma coleção de modelos, APIs e até bibliotecas. Por exemplo, o Angular, um framework JavaScript para aplicações, consiste em bibliotecas para animações, requisições HTTP, testes, renderização de formulários, reatividade, roteamento e muitas outras funcionalidades.

1.1 Definições

Um framework é um conjunto de ferramentas, bibliotecas e padrões que fornecem uma estrutura para o desenvolvimento de software. Um framework fornece um conjunto de funcionalidades pré-construídas que podem ser utilizadas para acelerar o desenvolvimento de um software. Geralmente, um framework tem uma arquitetura bem definida e uma documentação extensa, que ajuda o desenvolvedor a entender como as diferentes partes do sistema se interconectam e a desenvolver mais rapidamente.

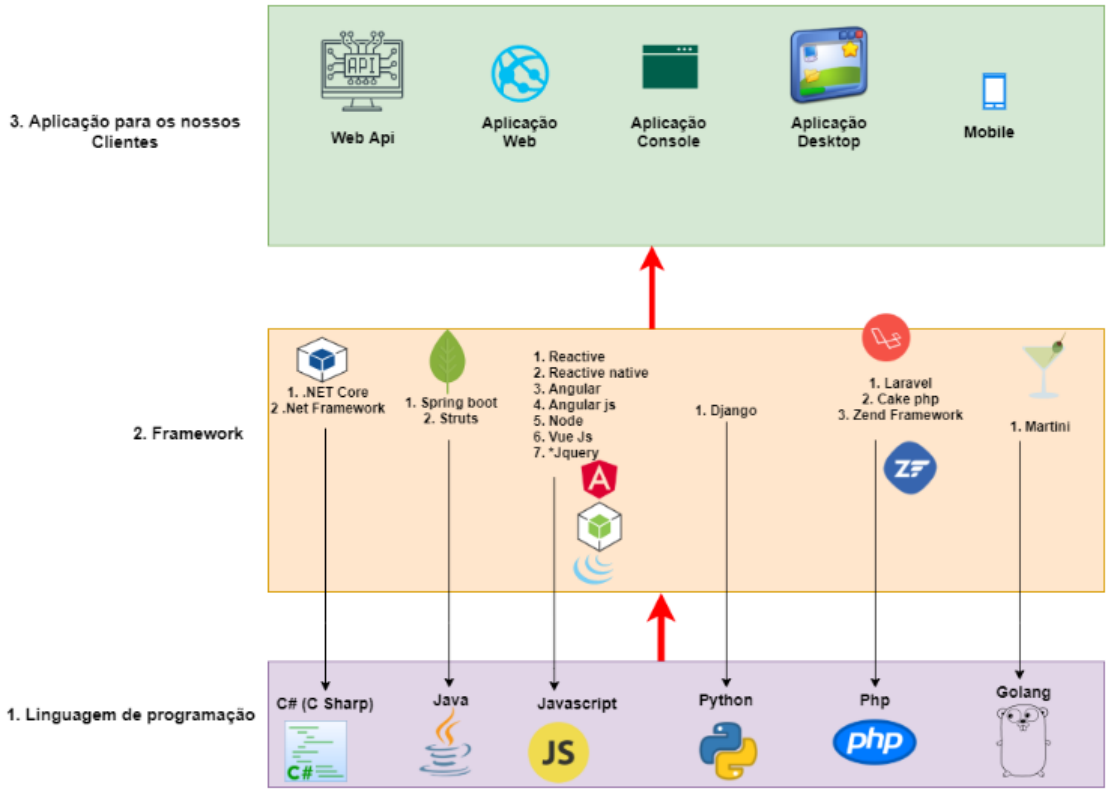
1.2 Exemplos de frameworks

Existem diversos frameworks e vamos explorar apropriadamente vários deles que são focados no desenvolvimento web. Entretanto, antes disso, vamos aprofundar um pouco o entendimento do que são para explanar sobre como são usados. Existem os mais diversos, mas, para efeito de entendimento, iniciamos com os citados abaixo como exemplos de frameworks de sucesso que podem servir de inspiração.

Importante!

Veremos a seguir, na Figura 2, o início da análise de alguns frameworks. Entenda que todos eles têm características em comum, como serem MVC e facilitarem acesso ao banco, entre outras características desejáveis.

Figura 2 – Explanação sobre os diferentes níveis de desenvolvimento, do mais básico que é a linguagem pura, passando pela camada de framework, chegando até a aplicação de frameworks



Fonte: Bianchi, 2023.

Segundo a Figura 2, pode-se entender que existem vários frameworks em várias plataformas. Vamos explanar um pouco o detalhamento de todos eles para que se tenha uma ideia de como e onde melhor utilizá-los.

- **.NET**

O primeiro deles é o .NET Framework, que é um framework de software desenvolvido pela Microsoft, que inclui uma grande biblioteca de código pré-construído e APIs (Interfaces de Programação de Aplicativos) que podem ser usadas pelos desenvolvedores para criar uma ampla gama de aplicativos para Windows. O .NET Framework fornece um ambiente de tempo de execução que permite que aplicativos sejam executados em qualquer máquina Windows que tenha o framework instalado. C# e o .NET Framework são frequentemente usados juntos para criar aplicativos Windows poderosos. C# é a linguagem de programação usada para escrever o código, e o .NET Framework fornece o ambiente de tempo de execução e as bibliotecas de código pré-construídas que

facilitam para os desenvolvedores a criação de aplicativos de forma rápida e eficiente. Nos últimos anos, a Microsoft lançou uma nova versão do .NET Framework chamada *.NET Core*, que é um framework multiplataforma que pode ser usado para desenvolver aplicativos para Windows, Linux e macOS. C# também pode ser usado com o .NET Core para criar aplicativos modernos e multiplataforma.

- **Spring Boot e Java Struts**

Java Struts e Spring Boot são frameworks web para Java que ajudam os desenvolvedores a criar aplicações web de forma rápida e fácil. Struts é um framework de código aberto que foi desenvolvido pela Apache Software Foundation. Ele fornece um conjunto de ferramentas e componentes para construir aplicações web baseadas na arquitetura Model-View-Controller (MVC) (<https://struts.apache.org>).

O Struts é amplamente utilizado há anos e possui uma grande comunidade de desenvolvedores que contribuem para seu desenvolvimento. O Spring Boot, por outro lado, é um framework mais novo que foi desenvolvido pela Pivotal Software. Ele também é de código aberto e é construído em cima do Spring Framework. O Spring Boot simplifica o desenvolvimento de aplicações web fornecendo um conjunto de ferramentas e convenções para configurar e implantar aplicações web. Uma das principais diferenças entre os dois frameworks é sua abordagem ao desenvolvimento. O Struts é mais focado em fornecer um conjunto de ferramentas e componentes que os desenvolvedores podem usar para construir suas aplicações, enquanto o Spring Boot adota uma abordagem mais opinativa, fornecendo um conjunto de convenções e padrões que tornam mais fácil para os desenvolvedores começar rapidamente (<https://spring.io>).

- **Django**

Django é um framework web de alto nível para Python que permite o desenvolvimento rápido de aplicações web seguras e escaláveis. É um framework de código aberto que segue o padrão Model-View-Controller (MVC) e é projetado para ajudar os desenvolvedores a evitar tarefas comuns e repetitivas no desenvolvimento web, como a criação de formulários HTML e a manipulação de sessões de usuário. Django é conhecido por sua facilidade de uso e produtividade. Ele fornece muitas ferramentas e recursos úteis, como um

ORM (Object-Relational Mapping) para trabalhar com bancos de dados, uma interface de administração pronta para uso que pode ser personalizada para atender às necessidades do projeto, autenticação e gerenciamento de permissões de usuário e muito mais. O Django também é altamente modular e permite que os desenvolvedores personalizem e estendam suas funcionalidades com facilidade. Além disso, a comunidade Django é grande e ativa, com muitos pacotes e bibliotecas disponíveis para ajudar no desenvolvimento de aplicações web. Em resumo, o Django é um framework Python poderoso e flexível para desenvolvimento web, com uma ampla gama de recursos e uma comunidade ativa de desenvolvedores. Ele é amplamente utilizado por empresas de todos os tamanhos para construir aplicações web escaláveis e robustas (<https://www.djangoproject.com>).

- **Laravel**

Laravel é um framework de desenvolvimento web em PHP de código aberto que segue o padrão Model-View-Controller (MVC). É conhecido por sua simplicidade, elegância e facilidade de uso, tornando-se um dos frameworks PHP mais populares atualmente. Laravel oferece um grande número de recursos e ferramentas úteis para desenvolvedores PHP, como roteamento, autenticação, gerenciamento de sessões, migrações de banco de dados, ORM (Object-Relational Mapping), sistema de cache e muito mais. Ele também inclui um sistema de templating chamado *Blade*, que permite aos desenvolvedores criar facilmente a interface do usuário de suas aplicações. Outra característica importante do Laravel é o seu sistema de gerenciamento de dependências chamado *Composer*. Isso torna a instalação e o gerenciamento de bibliotecas e pacotes PHP de terceiros muito mais fáceis. O Laravel também é altamente modular e permite que os desenvolvedores personalizem e estendam suas funcionalidades com facilidade. Além disso, a comunidade Laravel é grande e ativa, com muitos pacotes e bibliotecas disponíveis para ajudar no desenvolvimento de aplicações web. Em resumo, o Laravel é um framework PHP moderno, elegante e fácil de usar, com uma ampla gama de recursos e uma comunidade ativa de desenvolvedores. É uma ótima opção para o desenvolvimento de aplicações web escaláveis e robustas (<https://laravel.com>).

Considerando, então, que conhecemos alguns frameworks importantes, vamos agora aprofundar suas principais características e derivar para os

frameworks conhecidos para a web exclusivamente, em especial focados em linguagens Javascript e Typescript.

TEMA 2 – PRINCIPAIS FERRAMENTAS PARA FRONT-END E FRAMEWORKS: CARACTERÍSTICAS TÉCNICAS

Existem várias ferramentas e frameworks disponíveis para front-end, cada um com suas próprias características técnicas. Aqui estão algumas das principais ferramentas e frameworks:

- HTML (Hypertext Markup Language): é a linguagem padrão para criar documentos da web e fornece a estrutura básica de uma página da web.
- CSS (Cascading Style Sheets): é usado para adicionar estilos visuais a uma página da web, como cores, fontes, tamanhos e posicionamento.
- JavaScript: é uma linguagem de programação que permite criar interatividade em páginas da web, como menus de navegação, botões de rolagem e animações.

O modelo MVC (Model-View-Controller) é uma arquitetura de software comumente utilizada no desenvolvimento web, que separa a aplicação em três componentes principais: o Model (Modelo), o View (Visão) e o Controller (Controlador). O modelo é responsável pela manipulação de dados, a visão é responsável pela apresentação dos dados na interface do usuário, e o controlador é responsável pela interação entre o modelo e a visão.

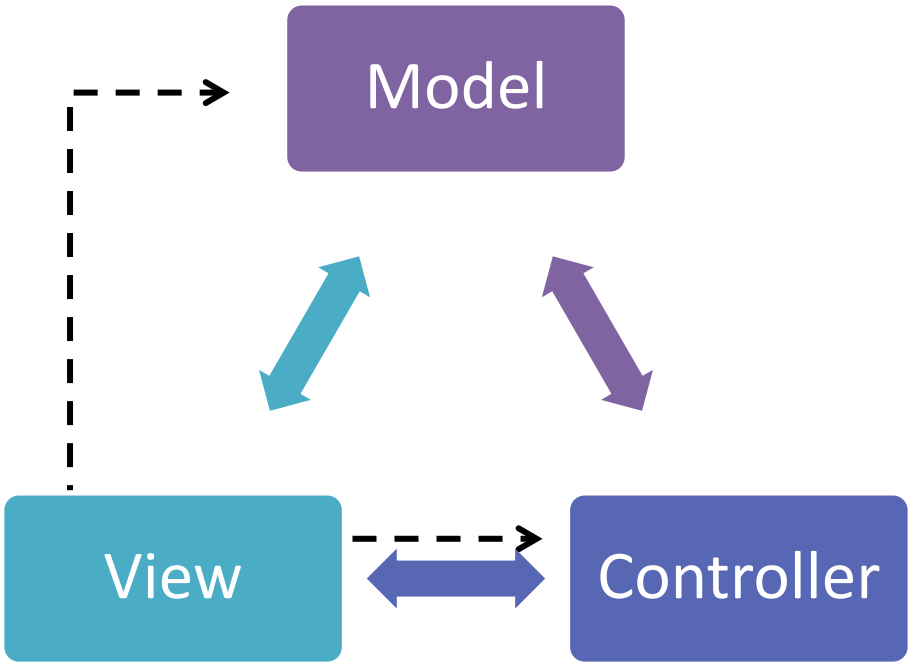
No desenvolvimento front-end, o padrão MVC é comumente usado em frameworks de JavaScript, como o AngularJS, ReactJS e Vue.js. Cada um desses frameworks implementa o padrão MVC de maneira um pouco diferente, mas todos eles compartilham a ideia fundamental de separar as responsabilidades da aplicação em três componentes distintos.

O modelo, no contexto de front-end, geralmente se refere à manipulação de dados e lógica de negócios, que pode incluir validação de entrada do usuário, chamadas de *API* e *gerenciamento de estado de aplicação*. A visão é responsável pela apresentação dos dados na interface do usuário. Isso inclui a renderização de componentes, a atualização da interface do usuário em resposta a alterações de estado e a manipulação de eventos de usuário. O controlador é responsável pela interação entre o modelo e a visão, permitindo que os usuários interajam com a interface do usuário e atualizem os dados na

aplicação. Ele também pode ser responsável pelo gerenciamento de rotas e do fluxo de navegação na aplicação.

Em resumo, o padrão MVC é uma abordagem comum no desenvolvimento front-end que divide a aplicação em três componentes distintos: modelo, visão e controlador. Isso ajuda a tornar a aplicação mais fácil de entender, modificar e manter ao longo do tempo (<https://www.w3schools.in/mvc-architecture>).

Figura 3 – Modelo conceitual em alto nível sobre o que é o MVC, como previsto em padrões de projeto



Fonte: elaborado por Ferste, 2023, com base em Gamma et al., 2000.

Embora este trabalho não seja destinado ao estudo de arquitetura, é importante detalhar um pouco mais, pois existem variações e implementações ligeiramente diferentes que são usadas na comunidade de desenvolvimento. Aqui estão algumas das variações mais comuns:

1. MVVM (Model-View-ViewModel): o MVVM é um padrão que se baseia no MVC, mas com uma separação mais clara entre a visão e o modelo. O ViewModel atua como um intermediário entre a visão e o modelo, gerenciando a lógica de apresentação de dados e atualização de estados.
2. MVP (Model-View-Presenter): o MVP é outro padrão que se baseia no MVC, mas com um foco maior na separação da lógica de apresentação

de dados da visão. O Presenter é responsável por gerenciar a lógica de apresentação de dados e atualização de estados, enquanto o modelo é responsável pela manipulação de dados e lógica de negócios.

3. Flux: arquitetura de gerenciamento de estado que se concentra em manter um estado centralizado para toda a aplicação. Ele segue um padrão unidirecional de fluxo de dados, com a visão disparando ações que são capturadas pelo Dispatcher, que atualiza o estado da aplicação. Em seguida, o estado atualizado é propagado para a visão, que se atualiza em resposta.
4. Redux: variação do padrão Flux que é amplamente utilizada em aplicações React. Ele também mantém um estado centralizado para toda a aplicação, mas usa um único store para gerenciar o estado da aplicação. As ações são enviadas para o store, que atualiza o estado da aplicação em resposta.

Essas são apenas algumas das variações do padrão MVC que são comumente usadas no desenvolvimento web em JavaScript. Cada uma dessas variações tem suas próprias vantagens e desvantagens, e a escolha de uma delas depende das necessidades específicas da aplicação. Veremos agora alguns dos principais frameworks que nos interessam para WEB em Javascript e Typescript.

Importante!

Abaixo seguem os frameworks que nos interessam no contexto de WEB, pois adotam HTML5 + CSS3 + Javascript + Typescript, são componentizáveis e utilizados até para Mobile.

2.1 React

O React é uma biblioteca JavaScript de código aberto para construir interfaces de usuário. Ele permite criar componentes reutilizáveis e gerenciar o estado de uma aplicação de forma eficiente.

Algumas características do React incluem:

- Componentes: o React é baseado em um sistema de componentes, que são blocos reutilizáveis de código que representam partes da interface do usuário. Os componentes do React são criados usando JavaScript e possuem uma estrutura clara de entrada e saída de dados.

- Virtual DOM: o React usa um DOM virtual, que é uma representação leve do DOM real. O uso do DOM virtual ajuda a aumentar o desempenho do aplicativo, pois evita a atualização desnecessária do DOM real.
- JSX: o React permite a criação de componentes usando uma sintaxe especial chamada *JSX*. O JSX permite que os desenvolvedores escrevam código HTML e JavaScript em conjunto, tornando mais fácil a criação de interfaces de usuário complexas.
- Ferramentas de desenvolvimento: o React vem com um conjunto de ferramentas de desenvolvimento integradas, incluindo o Create React App, que simplifica o processo de criação de um novo aplicativo React.
- Comunidade: o React possui uma grande comunidade de desenvolvedores e é continuamente atualizado para fornecer recursos avançados e melhorias de desempenho (<https://react.dev>).

Figura 4 – Conceito de *React*: basicamente tudo é componentizável, onde cada componente é uma div /div

```
function Video ({video}){
return (
    <div>
    <thumbnail video = {video} />
        < a href= {video.url}>
        <h3> {video.title} </h3>
        <p> {video.description} </p>
        </a>
        < LikeButton video = {video} > />
    </div>
    );
}
```

Fonte: React, 2023.

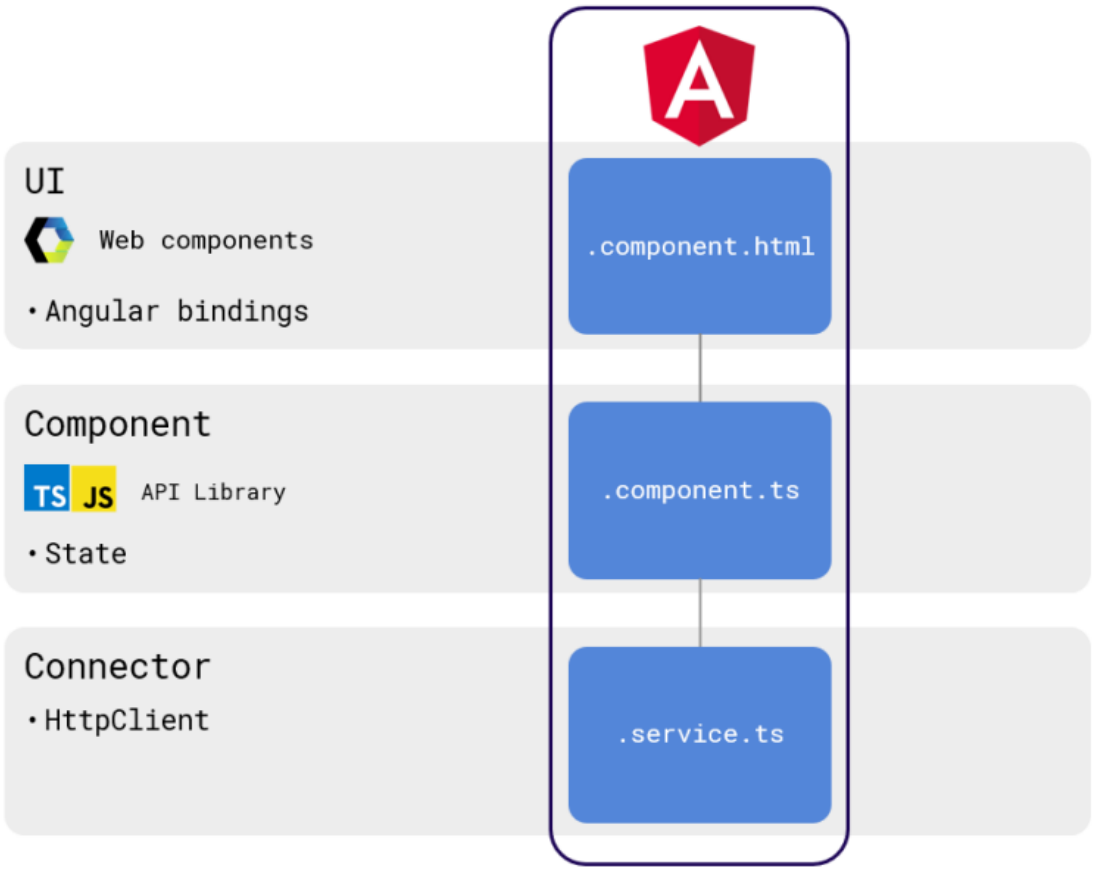
O React é usado em muitos projetos de grande escala, incluindo o Facebook, Instagram, Airbnb e Netflix. Ele é amplamente reconhecido por sua flexibilidade, eficiência e facilidade de uso, tornando-o uma excelente escolha para o desenvolvimento de aplicativos da web modernos e escaláveis (<https://www.react.dev>).

2.2 Angular

Angular é um framework JavaScript de código aberto para desenvolvimento de aplicativos da web. Ele oferece recursos avançados, como injeção de dependência, vinculação de dados e roteamento, além de ser altamente escalável. Algumas características do Angular incluem:

- Componentes: o Angular é baseado em um sistema de componentes, que são blocos reutilizáveis de código que representam partes da interface do usuário. Os componentes do Angular são criados usando TypeScript e possuem uma estrutura clara de entrada e saída de dados.
- Data Binding: o Angular fornece vários tipos de vinculação de dados, como vinculação unidirecional e bidirecional, que ajudam a manter o modelo e a visualização sincronizados.
- Injeção de Dependência: o Angular possui um poderoso sistema de injeção de dependência que ajuda a gerenciar a criação de objetos e sua dependência com outros objetos.
- Diretivas: o Angular fornece um conjunto de diretivas que podem ser usadas para manipular o DOM, controlar o comportamento de um componente e adicionar recursos interativos à interface do usuário.
- Ferramentas de desenvolvimento: o Angular vem com um conjunto de ferramentas de desenvolvimento integradas, incluindo o Angular CLI, que simplifica o processo de criação de um novo aplicativo Angular (<https://angular.io>).

Figura 5 – Arquitetura básica do Angular, baseada no conceito de *Single Web Page*, ou uma página única que pode ser atualizada (este conceito será explorado futuramente)



Fonte: Angular, 2023.

O Angular é usado em muitos projetos de grande escala, incluindo o Google Ads e o Google Fiber. Ele também possui uma grande comunidade de desenvolvedores e é continuamente atualizado para fornecer recursos avançados e melhorias de desempenho. O Angular é uma excelente escolha para criar aplicativos da web escaláveis e de alta qualidade.

Importante!

Angular tem duas principais vertentes, AngularJS, que é a primeira versão, e Angular, da segunda versão em diante. Neste sentido, entenda:

• AngularJS (<https://angularjs.org>): NÃO USAR, Legado, antigo, evitar, não usar, útil somente para quem desenvolveu nesta plataforma.

• Angular (<https://angular.io>): atual, versão 2 em diante usar sempre este, sem o JS no final, não usar qualquer referência ao primeiro, pois é muito diferente!

2.3 Vue

Vue é um framework JavaScript progressivo e de código aberto para a criação de interfaces de usuário. Ele foi criado por Evan You em 2014 e desde então se tornou uma das principais ferramentas para desenvolvimento de front-end.

Algumas características do Vue.js incluem:

- Data Binding: o Vue.js permite a vinculação de dados bidirecional, o que significa que os dados exibidos na interface do usuário são atualizados automaticamente quando há mudanças nos dados do modelo.
- Componentes: o Vue.js permite criar componentes reutilizáveis que podem ser facilmente integrados em diferentes partes de um aplicativo.
- Diretivas: o Vue.js fornece um conjunto de diretivas integradas que podem ser usadas para manipular o DOM, controlar o comportamento de um componente e adicionar recursos interativos à interface do usuário.
- Template Syntax: a sintaxe de modelo do Vue.js é intuitiva e fácil de entender, tornando mais fácil para os desenvolvedores criar interfaces de usuário complexas sem ter que lidar com muita complexidade.
- Ferramentas de desenvolvimento: o Vue.js vem com um conjunto de ferramentas de desenvolvimento integradas, incluindo Vue CLI, que simplifica o processo de criação de um novo aplicativo Vue.js.

Vue.js é usado em muitos projetos de grande escala, incluindo o Adobe Portfolio e o Alibaba. Ele também possui uma grande comunidade de desenvolvedores e é continuamente atualizado para fornecer recursos avançados e melhorias de desempenho (<https://vuejs.org>).

2.4 Bootstrap

Bootstrap oferece uma ampla gama de recursos, como modelos, componentes e ferramentas de construção, que ajudam a acelerar o processo de desenvolvimento.

Algumas características do Bootstrap incluem:

- Grid System: o Bootstrap fornece um sistema de grid responsivo que ajuda a criar layouts de página flexíveis e escaláveis que se adaptam a diferentes tamanhos de tela.

- Componentes: o Bootstrap oferece um conjunto de componentes predefinidos, como botões, formulários, navegação, alertas, modais e muito mais. Esses componentes são fáceis de usar e podem ser personalizados de acordo com as necessidades do projeto.
- Tipografia: o Bootstrap vem com uma variedade de estilos de tipografia predefinidos que podem ser facilmente personalizados.
- Estilos de CSS: o Bootstrap vem com uma variedade de estilos de CSS predefinidos que podem ser usados para personalizar a aparência do site, como cores, fontes, espaçamento, bordas e muito mais.
- Ferramentas de desenvolvimento: o Bootstrap vem com uma variedade de ferramentas de desenvolvimento integradas, como Sass e Less, que ajudam a personalizar ainda mais a aparência do site.

Bootstrap é usado em muitos projetos de grande escala, incluindo o Airbnb e o Spotify. Ele também possui uma grande comunidade de desenvolvedores que criam recursos e plugins para estender sua funcionalidade. Bootstrap é uma excelente opção para quem deseja criar sites responsivos e móveis de forma rápida e eficiente, sem precisar reinventar a roda (<https://getbootstrap.com>).

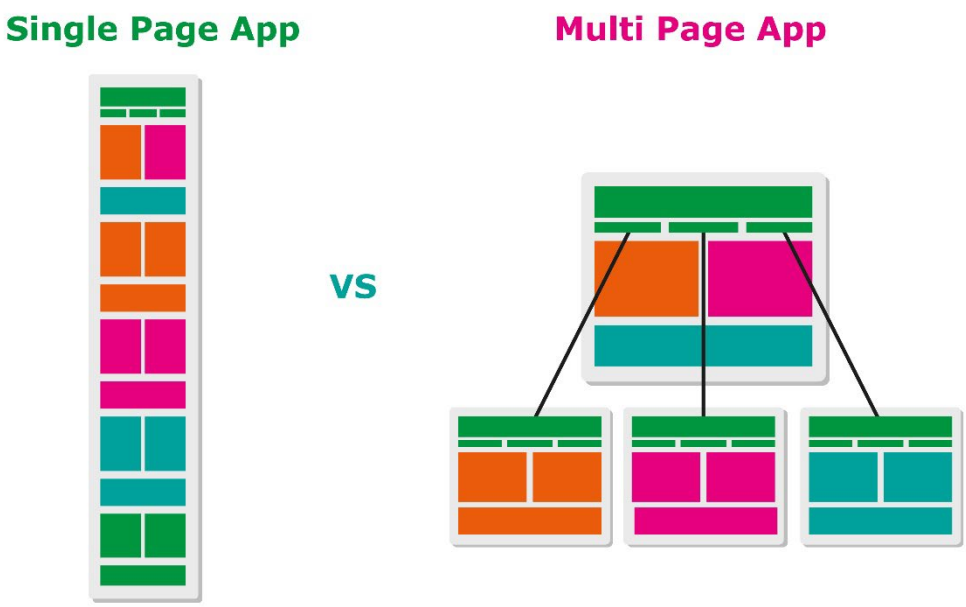
2.5 Materialize e Sass

Materialize é um framework de design responsivo que se concentra em fornecer componentes e estilos de material design do Google. Ele oferece uma variedade de recursos úteis, como um sistema de grid responsivo e componentes personalizados.

Sass é uma linguagem de folha de estilo que ajuda a escrever CSS mais eficiente e organizado. Ele oferece recursos como variáveis, aninhamento de seletores e mixins, que simplificam o processo de desenvolvimento de CSS.

Essas são apenas algumas das principais ferramentas e frameworks disponíveis para front-end. Cada uma delas tem suas próprias características técnicas e pode ser usada para atender a diferentes necessidades de desenvolvimento.

A principal diferença entre SPA (Single Web Page Application) e MPA (Multiple Web Page Application) está na forma como a navegação e as atualizações de conteúdo são tratadas. Enquanto as SPAs carregam uma única página inicial e atualizam dinamicamente o conteúdo, as MPAs têm várias páginas HTML independentes e dependem de solicitações ao servidor para carregar uma nova página. Ambas as abordagens têm suas vantagens e são adequadas para diferentes tipos de aplicações web, dependendo dos requisitos e da experiência do usuário desejada.



Crédito: Davi Souza.

3.1 Desenvolvimento Single Web Page *versus* Multiple Web Page

As múltiplas páginas da web nativas são uma abordagem tradicional e amplamente utilizada no desenvolvimento web. Essa estratégia consiste em criar várias páginas HTML independentes, cada uma contendo seu próprio conteúdo, design e funcionalidade. Embora essa abordagem possa apresentar desafios, também oferece vantagens significativas.

Uma das principais vantagens de utilizar várias páginas da web nativas é a organização. Dividir o conteúdo em páginas separadas permite uma melhor organização e estruturação do site. Cada página pode ser dedicada a um tópico ou seção específica, o que facilita a navegação e a localização de informações pelos usuários. Além disso, a capacidade de indexação dos mecanismos de pesquisa é aprimorada, pois cada página pode ser rastreada e indexada individualmente, aumentando a visibilidade nos resultados de pesquisa.

Outra vantagem é o desempenho. Ao carregar apenas o conteúdo necessário para cada página, os tempos de carregamento podem ser reduzidos, melhorando a experiência do usuário. Os navegadores podem lidar de forma mais eficiente com o carregamento de páginas individuais em comparação com sites de página única mais complexos. Além disso, a divisão do conteúdo em várias páginas pode permitir o uso de técnicas de caching, em que o navegador armazena em cache as páginas visitadas anteriormente, acelerando ainda mais o carregamento subsequente.

A flexibilidade é outro benefício das múltiplas páginas nativas. Cada página pode ter um layout, design e funcionalidade específicos, adaptados às necessidades da seção correspondente do site. Isso permite personalização e otimização mais detalhadas, garantindo que cada página atenda aos requisitos específicos. Além disso, a manutenção do código é facilitada, pois as alterações podem ser feitas de forma isolada em cada página.

No entanto, o uso de múltiplas páginas nativas também apresenta algumas desvantagens. A manutenção pode se tornar mais complexa, pois as alterações precisam ser feitas em várias páginas, aumentando o tempo necessário e a chance de erros. O carregamento repetitivo de elementos comuns em todas as páginas também pode afetar o tempo de carregamento geral do site. Além disso, a navegação entre as páginas pode ser menos fluida, especialmente se houver muitas páginas e os usuários precisarem clicar em vários links para acessar o conteúdo desejado.

É importante considerar cuidadosamente as necessidades e os objetivos do projeto ao decidir se as múltiplas páginas da web nativas são a abordagem mais adequada. Em alguns casos, outras soluções, como frameworks de aplicativos web ou abordagens de carregamento dinâmico de conteúdo, podem ser consideradas para melhorar a experiência do usuário e a eficiência do desenvolvimento.

3.2 Single Web Page

Single Web Page (ou página web única) é um tipo de site que contém todo o seu conteúdo em uma única página, sem a necessidade de navegar por várias páginas para obter informações ou executar ações. Em vez disso, o usuário percorre o conteúdo na mesma página usando a rolagem ou interações com elementos da página. Esses sites geralmente usam tecnologias como JavaScript, AJAX e jQuery para criar uma experiência de usuário interativa e atraente, além de fornecer transições suaves entre as seções da página.

As páginas web únicas são comuns em sites de portfólio, aplicativos web e produtos digitais, onde o objetivo é fornecer ao usuário informações e recursos de maneira clara e fácil de usar, sem distrações ou interrupções. Eles também são populares em sites móveis, onde a velocidade e a simplicidade são essenciais para uma experiência do usuário positiva.

Existem diferentes opções para criar uma Single Web Page, cada uma com suas próprias vantagens e desvantagens. Algumas das opções mais comuns incluem:

- Frameworks JavaScript: frameworks como Angular, React e Vue.js oferecem ferramentas e bibliotecas para criar uma Single Web Page responsiva e interativa.
- Sites geradores de página única: existem várias ferramentas on-line que permitem criar rapidamente uma Single Web Page usando modelos pré-construídos. Exemplos incluem Carrd, Unbounce e Leadpages.
- Scroll-based animations: Essa opção é baseada no uso de bibliotecas de animação, como o ScrollMagic, para criar animações em diferentes partes da página conforme o usuário rola para baixo.
- Parallax scrolling: a parallax scrolling é uma técnica de animação que cria a ilusão de profundidade na página. Isso é alcançado movendo diferentes camadas de elementos a diferentes velocidades enquanto o usuário rola pela página.
- Landing pages: landing pages são páginas projetadas para orientar o usuário a realizar uma ação específica, como se inscrever em um serviço, comprar um produto ou preencher um formulário.
- One-page portfolios: uma one-page portfolio é uma Single Web Page que apresenta todo o conteúdo do portfólio de um indivíduo ou empresa. Essa

opção é comum para profissionais criativos, como designers, artistas e fotógrafos.

Cada uma dessas opções pode ser personalizada para atender às necessidades de um projeto específico, com base em requisitos de design, funcionalidade e orçamento. Ao escolher entre um framework web e o desenvolvimento nativo para criar uma única página web, há alguns fatores importantes a serem considerados, como tempo de desenvolvimento, recursos necessários, escalabilidade e eficiência.

Um framework web, como o Angular, permite que você desenvolva rapidamente aplicativos web complexos com uma curva de aprendizado relativamente baixa. Eles também oferecem muitas bibliotecas e recursos pré-construídos que podem ajudar a economizar tempo e esforço.

Por outro lado, o desenvolvimento nativo oferece um alto nível de eficiência e controle, especialmente quando se trata de otimização de desempenho. No entanto, pode levar mais tempo para desenvolver um aplicativo web dessa forma e pode ser necessário mais recursos para executar o aplicativo. Dito isso, para destacar a eficiência do Angular em uma única página web, você pode aproveitar ao máximo os recursos e bibliotecas que o Angular oferece, como módulos e serviços, para criar um aplicativo web escalável e fácil de manter. Angular também tem uma grande comunidade de desenvolvedores, o que significa que você pode encontrar muitos recursos e soluções para problemas comuns em fóruns e comunidades on-line.

No entanto, é importante lembrar que cada projeto é único e requer uma abordagem personalizada. Considere suas necessidades específicas de desenvolvimento e os recursos disponíveis ao escolher entre um framework web e o desenvolvimento nativo para sua única página web.

Embora as Single Web Pages possam ter muitas vantagens, elas também apresentam algumas desvantagens que precisam ser consideradas antes de optar por esse tipo de site. Algumas das desvantagens mais comuns são:

- Dificuldades de SEO: como toda a informação está concentrada em uma única página, pode ser mais difícil para os motores de busca indexar todo o conteúdo. Além disso, é mais difícil otimizar cada seção individualmente para palavras-chave específicas.

- Navegação limitada: uma Single Web Page pode ter uma navegação limitada, o que pode dificultar a localização de informações específicas. Além disso, se o usuário precisar voltar a uma seção anterior, ele terá que rolar toda a página novamente.
- Problemas de responsividade: Single Web Pages podem ser mais difíceis de adaptar a diferentes tamanhos de tela e dispositivos móveis, o que pode prejudicar a experiência do usuário em smartphones e tablets.
- Dificuldades de manutenção: se a Single Web Page for atualizada regularmente, pode ser difícil manter a coerência visual e garantir que todas as seções funcionem corretamente.

É importante pesar essas desvantagens em relação aos benefícios antes de optar por uma Single Web Page. Para alguns projetos, as vantagens podem superar as desvantagens, enquanto para outros projetos, uma abordagem mais tradicional pode ser mais apropriada (Frain, 2022).

3.3 Qual o melhor?

A escolha entre uma Single Web Page (página única) e uma multiple web page (múltiplas páginas) depende das necessidades e objetivos específicos do projeto.

Uma Single Web Page oferece uma experiência mais fluida para o usuário, pois todo o conteúdo é carregado em uma única página, evitando interrupções durante a navegação. Isso é especialmente útil para aplicativos ou sites que fornecem informações ou funcionalidades contínuas, como aplicativos de mensagens ou ferramentas de produtividade. Além disso, o desenvolvimento e a manutenção podem ser mais rápidos e simples, já que todo o código e conteúdo estão em um único local. No entanto, o desempenho pode ser afetado se houver uma grande quantidade de conteúdo ou recursos pesados a serem carregados inicialmente.

Por outro lado, as múltiplas páginas oferecem uma melhor organização e escalabilidade para projetos com muitos conteúdos distintos. Cada página pode ser dedicada a um tópico ou seção específica, facilitando a navegação e a localização de informações. Isso é particularmente útil em sites maiores, como blogs, lojas on-line ou sites de notícias, onde os usuários podem acessar diferentes partes do site de acordo com suas necessidades. Além disso, a

indexação dos mecanismos de pesquisa é aprimorada, pois cada página pode ser rastreada e indexada individualmente. No entanto, a manutenção pode se tornar mais complexa, já que as alterações precisam ser feitas em várias páginas diferentes.

Em resumo, uma Single Web Page é adequada para aplicativos ou sites que exigem uma experiência de usuário contínua e fluída, enquanto as múltiplas páginas são mais adequadas para sites com muitas seções ou conteúdos distintos. A escolha depende do tipo de projeto, das necessidades dos usuários e das preferências de desenvolvimento e manutenção. É importante avaliar cuidadosamente as características e requisitos específicos antes de decidir qual abordagem utilizar.

TEMA 4 – FERRAMENTAS DE PROTOTIPAGEM

Ferramentas de prototipagem são programas ou aplicativos usados para criar protótipos interativos de produtos digitais, como websites, aplicativos móveis, softwares e outros sistemas de software. Essas ferramentas são usadas para visualizar e testar a interface do usuário, fluxos de navegação e funcionalidades antes de iniciar a fase de desenvolvimento real.

As ferramentas de prototipagem permitem que designers, desenvolvedores e outros membros da equipe trabalhem juntos para criar protótipos interativos de alta fidelidade em um ambiente de colaboração. Além disso, essas ferramentas geralmente possuem recursos que permitem a criação de animações, transições, interações e outras funcionalidades que podem ser testadas pelos usuários para obter feedback valioso.

Existem muitas ferramentas de prototipagem disponíveis, desde programas de desktop até aplicativos baseados em nuvem (Norman, 2023).

Algumas das ferramentas de prototipagem mais populares incluem Figma, Sketch, Adobe XD, Axure RP, InVision, Marvel App e Balsamiq. Cada ferramenta tem suas próprias características e recursos, e a escolha da ferramenta de prototipagem depende das necessidades do projeto e das preferências pessoais da equipe.

4.1 Figma

Figma é uma ferramenta de design e prototipagem baseada em nuvem, usada para criar protótipos de aplicativos, interfaces de usuário e outros projetos digitais. Foi lançado em 2016 e tornou-se popular rapidamente entre designers, desenvolvedores e equipes de produtos devido à sua facilidade de uso, colaboração em tempo real e recursos avançados de design.

Uma das principais vantagens do Figma é que permite que as equipes colaborem em tempo real, permitindo que várias pessoas trabalhem em um projeto ao mesmo tempo. Além disso, tem uma interface intuitiva e fácil de usar, com uma ampla gama de recursos para a criação de protótipos interativos, como animações, transições e interações. Também é possível criar estilos compartilhados para garantir consistência visual em todo o projeto, além de oferecer suporte para vários formatos de arquivo, incluindo imagens, ícones, gráficos vetoriais e arquivos de fonte. Outro destaque é a capacidade de criar apresentações de design para compartilhar com a equipe ou clientes, bem como integrações com outras ferramentas, como Slack, Trello e GitHub.

O Figma oferece uma versão gratuita para indivíduos e equipes pequenas, bem como uma versão paga com recursos adicionais para equipes maiores e projetos mais complexos. Com sua facilidade de uso e recursos avançados de prototipagem, o Figma é uma das ferramentas mais populares entre os profissionais de design e desenvolvimento de produtos (<https://www.figma.com>).

4.2 Sketch

Sketch é uma ferramenta de design gráfico e prototipagem popular para interfaces de usuário, aplicativos móveis e web design. Foi lançado em 2010 e é conhecido por sua facilidade de uso, recursos avançados de design e capacidade de colaboração em equipe.

Algumas das características do Sketch incluem:

- ferramentas avançadas de vetor e pixel para criação de designs de alta qualidade;
- biblioteca de componentes reutilizáveis que pode ser compartilhada entre projetos;

- capacidade de criar estilos compartilhados para manter a consistência visual em todo o projeto;
- suporte para plugins que adicionam recursos adicionais à ferramenta;
- integrações com outras ferramentas populares, como InVision, Zeplin e Marvel App.

Uma das vantagens do Sketch é que ele foi desenvolvido especificamente para designers de interface do usuário e, portanto, possui recursos avançados para esse tipo de design. Além disso, oferece suporte para múltiplas pranchetas, o que facilita a criação de designs para diferentes telas e dispositivos.

O Sketch é uma ferramenta paga com preços acessíveis, o que o torna uma escolha popular para designers independentes, startups e pequenas empresas. Com sua facilidade de uso e recursos avançados de prototipagem, o Sketch é amplamente utilizado por profissionais de design e desenvolvimento de produtos em todo o mundo (<https://www.sketch.com>).

4.3 Balsamiq

Balsamiq é uma ferramenta de prototipagem de interface do usuário (UI) e user experience (UX) que permite aos designers criar wireframes rápidos e simples para seus projetos. Foi lançado em 2008 e é conhecido por sua simplicidade e facilidade de uso.

Algumas das características do Balsamiq incluem:

- biblioteca de elementos de interface do usuário pré-desenhados para adicionar rapidamente ao wireframe;
- ferramentas de desenho vetorial para criar novos elementos de interface do usuário;
- capacidade de criar múltiplas versões de um wireframe para iterar rapidamente em ideias de design;
- suporte para colaboração em equipe e feedback de stakeholders;
- exportação de arquivos em diferentes formatos para compartilhamento e desenvolvimento.

Uma das principais vantagens do Balsamiq é sua simplicidade e facilidade de uso. Ele é projetado para ser uma ferramenta rápida e fácil para criar wireframes, permitindo que os designers criem protótipos simples e eficazes em

pouco tempo. Isso pode economizar tempo e dinheiro em projetos, especialmente para equipes menores ou projetos com prazos apertados.

O Balsamiq é uma ferramenta paga, mas oferece uma versão de avaliação gratuita por 30 dias. É uma escolha popular entre os designers e desenvolvedores que precisam de uma ferramenta de prototipagem rápida e eficaz para seus projetos (<https://balsamiq.com>).

TEMA 5 – FERRAMENTAS DE DESENVOLVIMENTO

Existem várias ferramentas de desenvolvimento que podem ser úteis para desenvolvedores, incluindo:

- Insomnia: uma ferramenta de teste de API que permite enviar solicitações HTTP e HTTPS e visualizar as respostas;
- Angular CLI: uma interface de linha de comando (CLI) para criar aplicativos Angular. Ela automatiza muitas tarefas de desenvolvimento, como a criação de componentes e serviços;
- WSL (Windows Subsystem for Linux): uma camada de compatibilidade que permite executar o Linux e suas ferramentas de linha de comando em um sistema operacional Windows;
- Chocolatey: um gerenciador de pacotes para o Windows que permite instalar e gerenciar software de forma automatizada;
- NVM (Node Version Manager): uma ferramenta que permite instalar e gerenciar várias versões do Node.js em um mesmo ambiente;
- NPM/Yarn: gerenciadores de pacotes JavaScript que permitem instalar e gerenciar bibliotecas e dependências de projetos;
- Node.js: um ambiente de tempo de execução JavaScript que permite executar código JavaScript do lado do servidor.

Essas ferramentas podem ajudar os desenvolvedores a automatizar tarefas, gerenciar dependências, testar e depurar código e executar código em diferentes ambientes. Existem muitas outras ferramentas disponíveis para desenvolvedores, dependendo do tipo de projeto e tecnologias utilizadas. É importante que os desenvolvedores se mantenham atualizados sobre as ferramentas disponíveis e escolham as mais adequadas para suas necessidades.

5.1 Insomnia ou outro programa para acesso a API

Para desenvolver, necessitamos de programas para acessar uma API exposta. Um exemplo é o Insomnia, que é uma ferramenta de teste de API que permite enviar solicitações HTTP e HTTPS e visualizar as respostas. Ela é usada para testar e depurar APIs RESTful e outras APIs baseadas em HTTP. O Insomnia suporta vários tipos de solicitações, como GET, POST, PUT, DELETE e PATCH, e permite definir cabeçalhos personalizados, parâmetros de consulta e payloads de solicitação.

O Insomnia é uma ferramenta gratuita e de código aberto disponível para Windows, Mac e Linux. Ele possui uma interface gráfica de usuário amigável que torna fácil criar solicitações, definir variáveis de ambiente, testar diferentes cenários de solicitação e compartilhar suas coleções de solicitações com outras pessoas.

Algumas das características do Insomnia incluem:

- suporte à autenticação básica, OAuth 1.0a e 2.0 e JSON Web Tokens (JWTs);
- criação e organização de coleções de solicitações para diferentes APIs.
- interface gráfica de usuário amigável e fácil de usar;
- suporte para importar e exportar solicitações em vários formatos, como cURL, Postman e Swagger;
- monitoramento de solicitações em tempo real para depurar problemas de conexão e latência.

O Insomnia é uma ferramenta popular entre desenvolvedores que trabalham com APIs RESTful e outras APIs baseadas em HTTP, pois permite testar e depurar solicitações rapidamente. Com a interface amigável e as várias funcionalidades que ele oferece, é uma das ferramentas de teste de API mais populares atualmente (<https://insomnia.rest>).

5.2 Angular CLI

Angular CLI (Command Line Interface) é uma interface de linha de comando que facilita a criação, desenvolvimento e teste de aplicativos Angular. Ele é um conjunto de ferramentas que ajuda os desenvolvedores a automatizar

tarefas repetitivas, como a criação de componentes e serviços, a configuração de roteamento e a execução de testes.

Algumas das funcionalidades mais importantes do Angular CLI incluem:

- Criação de um novo projeto Angular: o Angular CLI facilita a criação de um novo projeto Angular com todas as configurações básicas, como arquivos de configuração, estrutura de pastas e dependências.
- Geração de componentes, serviços, diretivas e outros artefatos: o Angular CLI permite gerar facilmente vários artefatos Angular usando uma única linha de comando. Ele também pode gerar automaticamente testes para esses artefatos.
- Servidor de desenvolvimento: o Angular CLI possui um servidor de desenvolvimento que permite executar o aplicativo localmente e ver as mudanças em tempo real no navegador.
- Compilação para produção: o Angular CLI possui um processo de compilação otimizado para a produção, que gera arquivos minificados e concatenados para o menor tempo de carregamento possível.
- Testes: o Angular CLI integra testes unitários e de integração, que podem ser executados facilmente usando uma única linha de comando.

O Angular CLI é uma ferramenta essencial para desenvolvedores que trabalham com Angular, pois permite aumentar a produtividade e facilitar o processo de desenvolvimento. Com o Angular CLI, é possível criar e gerenciar projetos Angular de maneira mais rápida e eficiente (<https://angular.io/cli>).

FINALIZANDO

Concluímos aqui o entendimento básico do que é a estrutura e frameworks. Eles fornecem componentes reutilizáveis, bibliotecas e padrões de projeto que aceleram o processo de desenvolvimento e aumentam a produtividade. Com um framework, os desenvolvedores podem se concentrar mais na lógica de negócio e nas funcionalidades específicas da aplicação, em vez de perder tempo desenvolvendo componentes básicos do zero. Isso resulta em uma economia de tempo e recursos significativa.

Além disso, o conhecimento sobre infraestrutura web é crucial para o desenvolvimento eficiente e bem-sucedido de aplicações. Compreender os conceitos de *hospedagem*, *servidores*, *balanceamento de carga*,

armazenamento de dados e *gerenciamento de tráfego* é essencial para tomar decisões adequadas durante o processo de desenvolvimento. Um bom entendimento da infraestrutura web permite otimizar o desempenho da aplicação, garantir sua disponibilidade e escalabilidade, além de facilitar a integração com outros sistemas e serviços.

Em resumo, os frameworks web e o conhecimento sobre infraestrutura são fundamentais para o desenvolvimento de aplicações e sites modernos. Eles proporcionam eficiência, segurança, escalabilidade e permitem a criação de aplicações robustas e de alta qualidade. Portanto, investir no aprendizado desses elementos é essencial para os desenvolvedores e equipes de desenvolvimento que desejam criar aplicações web bem-sucedidas.

REFERÊNCIAS

ANGULAR. Disponível em: <https://www.angular.io>. Acesso em: 3 ago. 2023.

APACHE. Disponível em: <https://struts.apache.org>. Acesso em: 3 ago. 2023.

BALSAMIQ. Disponível em: <https://balsamiq.com>. Acesso em: 3 ago. 2023.

BETRYBE. Disponível em: <https://blog.betrybe.com/framework-de-programacao/angular>. Acesso em: 3 ago. 2023.

BIACHI L. **O que é aplicação, framework e linguagem de programação qual a diferença?** O que e qual devo escolher quando estou iniciando minha carreira? Disponível em: <https://pt.linkedin.com/pulse/o-que-%C3%A9-aplica%C3%A7%C3%A3o-framework-e-linguagem-de-qual-diferen%C3%A7a-bianch>. Acesso em: 3 ago. 2023.

BOOTSTRAP. Disponível em: <https://getbootstrap.com>. Acesso em: 3 ago. 2023.

DJANGO. Disponível em: <https://www.djangoproject.com>. Acesso em: 3 ago. 2023.

FRAIN, B. **Responsive Web Design with HTML5 and CSS - Fourth Edition**: Build future-proof responsive websites using the latest HTML5 and CSS techniques. Packt, 2022.

FIGMA. Disponível em: <https://www.figma.com>. Acesso em: 3 ago. 2023.

GAMMA, E. et al. **Padrões de projetos**: soluções reutilizáveis de software orientados a objetos. São Paulo> Bookman, 200. *E-book*.

GENEXUS. Disponível em: <https://training.genexus.com/pt/aprendizagem/pdf/arquitetura-de-aplicativo-angular-pdf>. Acesso em: 3 ago. 2023.

GOULÃO, M. **Component-Based Software Engineering**: a Quantitative Approach. Universidade Nova de Lisboa, 2008.

INSOMNIA. Disponível em: <https://insomnia.rest>. Acesso em: 3 ago. 2023.

LARAVEL. Disponível em: <https://laravel.com>. Acesso em: 3 ago. 2023.

LEARN. Disponível em: <https://learn.microsoft.com/pt-br/windows/wsl/tutorials/wsl-vscode>. Acesso em: 3 ago. 2023.

NORMAN, D. A. **Design for a Better World**: Meaningful, Sustainable, Humanity Centered. The MIT Press, Cambridge, Massachusetts, 2023.

NPM. **Node Package Manager**. Disponível em: <https://www.npmjs.com>. Acesso em: 3 ago. 2023.

REACT. Disponível em: <https://www.react.dev>. Acesso em: 3 ago. 2023.

SCHMIDT, D. C.; WALLNAU, K. **"Component-Based Software Engineering**: Putting the Pieces Together" por Clemens Szyperski. Addison Wesley, 2001.

SKETCH. Disponível em: <https://www.sketch.com>. Acesso em: 3 ago. 2023.

SPRING. Disponível em: <https://spring.io>. Acesso em: 3 ago. 2023.

TSCONFIG. Disponível em: <https://www.typescriptlang.org/docs/handbook/tsconfig-json.html>. Acesso em: 3 ago. 2023.

TYPESCRIPT. Disponível em: <https://www.typescriptlang.org>. Acesso em: 3 ago. 2023.

VUEJS. Disponível em: <https://vuejs.org>. Acesso em: 3 ago. 2023.

WEBFOUNDATION. **World Wide Web Foundation**. Disponível em: <https://webfoundation.org>. Acesso em: 3 ago. 2023.

W3C. Disponível em: <https://www.w3schools.in/mvc-architecture>. Acesso em: 3 ago. 2023.