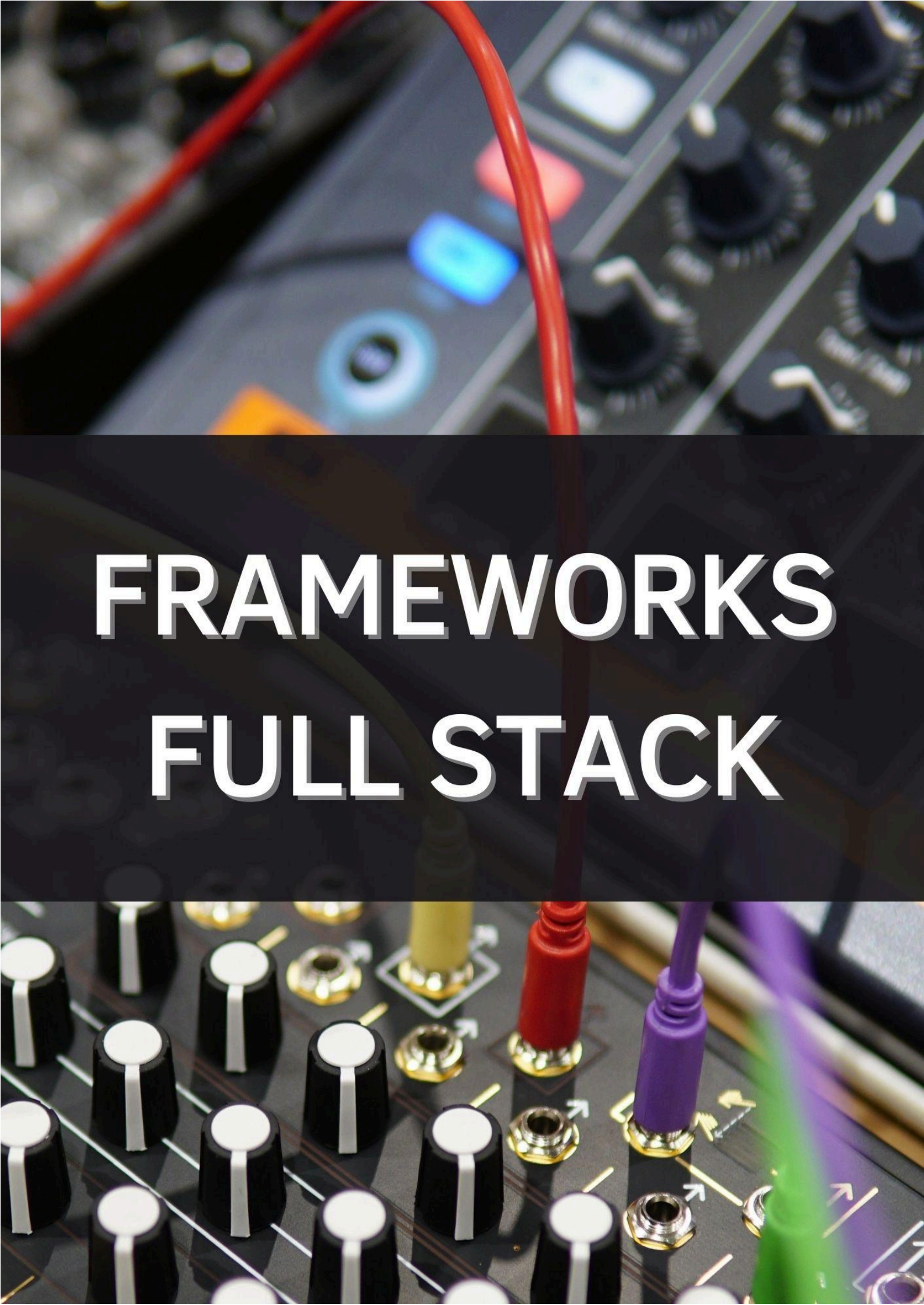


FRAMEWORKS FULL STACK

Texto base

1

Apresentação do Curso, explicação do que é Framework Full Stack e Configuração do ambiente de desenvolvimento

Prof. André Nascimento Maia
Prof. Caio Nascimento Maia

Resumo

Desenvolver aplicações web modernas utilizando framework full-stack garante grande produtividade, maior foco no problema em questão e maior flexibilidade para adicionar e remover pequenas funcionalidades durante o desenvolvimento. Nesta aula, entenderemos os conceitos sobre os termos framework full-stack e apresentaremos algumas ferramentas relevantes para essa prática, também ajudaremos o aluno a preparar o ambiente de desenvolvimento para a criação de aplicações web full-stack.

1.1. O Curso de Framework Full Stack

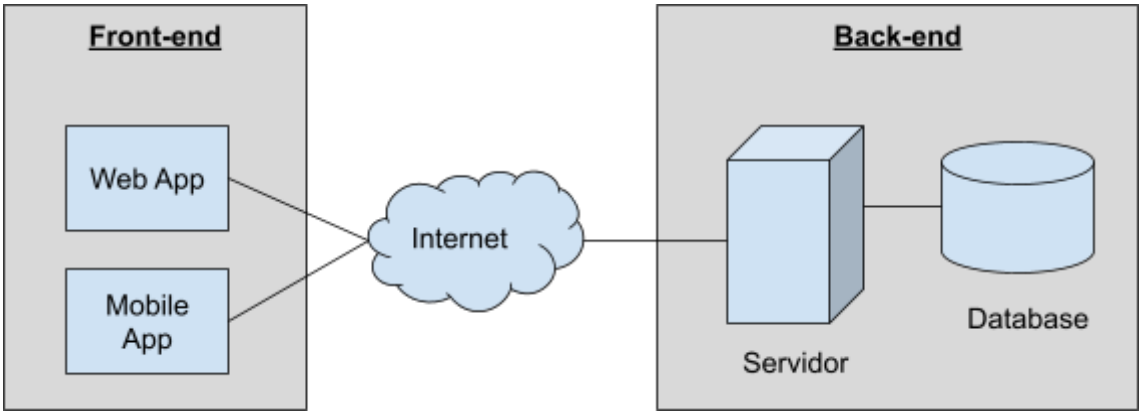
É comum vermos por todos os lugares sistemas computacionais com o objetivo de solucionar diversos problemas comuns, desde os problemas mais simples, como criar e manter anotações sobre qualquer assunto como o Google Keep e Microsoft To Do, até permitir pagamentos por diversos meios de pagamento on-line como os serviços PagSeguro e PayPal.

Além dos serviços abertos ao público, empresas de pequeno, médio e grande porte necessitam de sistemas computacionais para suportar seus processos, operações e garantir sua eficiência.

Em ambos os casos, seja na construção de software como serviço público ou privado com o objetivo de solucionar do mais simples ao mais complexo dos problemas da sociedade, todo desenvolvimento de software utiliza-se de padrões, estratégias, modelos, processos e técnicas para garantir assertividade, produtividade e flexibilidade durante todo o processo.

Um dos modelos mais utilizados para a construção de software é o modelo conhecido como cliente-servidor (ver Figura 1.1). Este é um modelo desenvolvido pela Xerox na década de 70 (SARAF, Pranay D.; BARTERE, Mahip M.; LOKULWAR, Prasad P., 2021).

Figura 1.1: Arquitetura cliente- servidor



Fonte: do autor, 2021

Front-ends são as aplicações da camada cliente do modelo cliente-servidor. Front-end é tudo o que fica exposto para o usuário e pode ser considerado o canal onde a empresa entrega os seus serviços para os seus clientes. Alguns exemplos são: aplicativos *mobile iOS e Android*, aplicativos Web, Web Sites, APIs públicas, arquivos, relatórios.

Back-end é a camada servidor do modelo cliente-servidor. São todas as aplicações que não ficam expostas diretamente aos usuários. São os processos internos do serviço, como APIs internas, rotinas de processamentos de dados em lote, bancos de dados, filas de processamento, monitoramento, alarmes de monitoramento.

Quando unidos os dois conceitos de back-end e front-end para criação de uma aplicação, essa aplicação é chamada de *aplicação full-stack*, cujo a ideia principal é definir que a aplicação tem uma *pilha de tecnologias completa* para o seu desenvolvimento (ZAMMETTI, 2020).

Durante o desenvolvimento de software, os *Frameworks* são amplamente utilizados. A base de todo *framework* é um conjunto de códigos comuns com funções genéricas que combinadas permitem a criação de operações básicas como: ler o conteúdo de arquivos, conectar-se a um servidor através da internet, executar comandos em um banco de dados, entre outras várias aplicações. Para todas as linguagens de programação e tecnologias existentes é possível encontrar diversos *frameworks* para auxiliar a resolver todo tipo de problema de forma produtiva. São exemplos:

- React.js: *Framework* para criação de aplicações web do lado do cliente;
- Flask: É um *framework* para a criação de aplicações web do lado do servidor;

Dadas as definições para *full-stack* e aplicações *full-stack*, o que seria então um *framework full-stack?*

Quando nos referimos a um *framework full-stack*, estamos nos referindo a um conjunto de *frameworks* utilizados para criação de aplicações *full-stack*, com o objetivo de aumentarmos a nossa produtividade juntando diversas partes de código através dos *frameworks* utilizados.

Nesta disciplina, o nosso objetivo principal é aprender a resolver problemas utilizando uma solução baseada em software utilizando *frameworks* para construção de aplicações *full-stack*.

Utilizaremos como base o problema de vendas on-line de canecas personalizadas, criando um e-commerce fictício que chamaremos de Impact Commerce.

Começaremos pelo desenho e entendimento da solução de software, com o objetivo principal de entender o todo em um alto nível antes de implementarmos qualquer trecho necessário para a solução. Em seguida, aprofundaremos na solução para o *front-end*, discutindo aspectos importantes para todo *front-end* e levando em consideração os requisitos entendidos e levantados no início da disciplina para a nossa solução. Por fim, será implementada uma solução de *back-end*, discutindo tudo o que é necessário para que o que foi desenhado para que a solução funcione de acordo com os requisitos funcionais e não funcionais desejados, desde implementação de APIs REST até o conceito de *Continuous Delivery* utilizando *Containers Dockers* e nuvens públicas para um ciclo de desenvolvimento de software completo do desenvolvimento local até o ambiente de produção.

Em cada uma das lições que serão distribuídas ao longo do curso, discutiremos conceitos teóricos que envolvem cada uma das fases e também indicaremos práticas para experimentar os conceitos apresentados. A ideia principal é entendermos os conceitos teóricos em nível de profundidade necessários para aplicação prática e resolução de problemas reais, como o problema-tema discutido durante a disciplina, a construção do e-commerce.

1.2. Full-stack moderna para desenvolvimento de software

Todo desenvolvimento de software exige a utilização de ferramentas, tecnologias e frameworks para garantir a produtividade e assertividade. Para solucionarmos problemas baseados em software, vamos utilizar um desenvolvimento *full-stack* moderno utilizando *frameworks* específicos para cada trecho da solução, ferramentas para nos auxiliar na produção de código, linguagens de programação adequadas para resolver cada tipo de problema e infraestrutura necessária para o ambiente de produção e desenvolvimento. Em detalhes, na tabela 1.1, temos as principais tecnologias que fazem parte da nossa pilha moderna completa para desenvolvimento de aplicações utilizadas.

Tabela 1.1. Pilha moderna completa para desenvolvimento de aplicações

Categoria	Front-end	Back-end
Framework	Next.js	Flask
Ferramentas para desenvolvimento	NPM e Visual Studio Code	Terminal e Visual Studio Code
Linguagem de programação	Javascript	Python
Infraestrutura	Vercel	Heroku
Persistência	—	PostgreSQL
Protocolos de comunicação	HTTP	APIs REST

Fonte: do autor, 2021

Esta stack é considerada moderna já que utiliza tecnologias extremamente difundidas pela comunidade de desenvolvimento de software, além de garantirem simplicidade, grande produtividade e flexibilidade para construção de diversas soluções diferentes utilizando a mesma *stack*.

React.js é um dos frameworks mais utilizados da atualidade para o desenvolvimento de aplicações web e também de aplicações *mobile*, utilizando sua versão chamada React Native. Porém, para aumentarmos nossa produtividade e utilizamos uma série de configurações, ajustes, padrões e recomendações utilizadas por diversas empresas, utilizaremos Next.js que é um framework sobre React.js para criação do front-end do e-commerce. O Next.js oferece uma das melhores experiências para desenvolvimento de aplicações web do mercado, implementando por padrão, funcionalidades como: renderização estática e dinâmica, construção inteligente do código compilado para o ambiente de produção, pré-carregamento de rotas para melhorar a experiência do cliente. A escolha de Next.js auxilia na escolha de todos as outras categorias de front-end, porque Next.js é criado sobre React.js, que é escrito em Javascript e justifica a escolha das linguagens de programação Javascript ou TypeScript.

Utilizaremos Python como linguagem de programação para o back-end, e um dos seus principais frameworks chamado Flask para a criação de APIs REST. Será necessário persistir os dados para entendimento e processamento, então, utilizaremos o banco de dados relacional PostgreSQL. Ambas as escolhas se encaixam muito bem com a Heroku, infraestrutura escolhida, que é um tipo de serviço também conhecido como Platform as a Service (PaaS). Os serviços de PaaS nos permitem utilizar uma infraestrutura flexível e escalável sem nos preocuparmos com manutenção e configuração dessa infraestrutura que é oferecida como um serviço.

1.3. Configuração do ambiente de desenvolvimento

Para desenvolvermos aplicações utilizando uma pilha moderna completa, devemos ter o ambiente de desenvolvimento local configurado adequadamente.

Os pré-requisitos são:

1. **Sistema operacional com base Linux ou Windows**. Em nossos exemplos utilizaremos MacOS, que é um sistema operacional baseado em Linux e muito utilizado atualmente para desenvolvimento de software pelas grandes empresas de tecnologia, dado que permite grande produtividade e estabilidade para os desenvolvedores. Porém, em qualquer sistema operacional Linux ou Windows, existem instalações equivalentes.
2. **Conexão com a Internet**. Todos os pacotes de instalação e configuração do ambiente atual, serão baixados da internet via links fornecidos durante a lição.

Os objetivos para o *front-end* são:

1. **Configurar o Node.js**. O Node.js é um motor para execução de código da linguagem Javascript construído sobre o motor de execução V8 Javascript do Chrome. Utilizaremos o Node.js com ambiente base para produção de código da camada de *front-end*.
2. **Instalar ferramentas de NPM, React.js e Next.js no ambiente de Node.js**. Essas serão as ferramentas que utilizaremos para o desenvolvimento do *front-end*. Elas nos auxiliaram na pré-configuração do *framework*, construção de código, gerenciamento de dependências, validação de código, geração de código utilizando os componentes dos *frameworks*.

Os objetivos para o *backend-end* são:

1. **Instalar linguagem de programação e *runtime* Python**. Utilizaremos a linguagem de programação python na versão maior ou superior a 3.7 para desenvolvimento do serviço de *back-end.*
2. **Configurar ambiente de containers Docker**. Os containers docker nos permitem simular ambientes computacionais segregados em uma única máquina de desenvolvimento. Nos auxiliará na instalação de bancos de dados PostgreSQL, por exemplo.

Os objetivos em comum:

1. **Instalar Visual Studio Code**. Este será a interface de desenvolvimento (IDE) utilizada durante toda a disciplina e nos exemplos. O aluno que preferir utilizar outra IDE por estar mais familiarizado, não haverá problemas.
2. **Instalar e configurar Git e conta no GitHub**. Todo código será versionado em repositórios Git. Inclusive os exemplos serão disponibilizados em repositórios Git, então é necessário que os alunos tenham familiaridade com esta ferramenta. Os repositórios também serão disponibilizados através do Github, para facilitar o processo de *Continuous Delivery* nos serviços Vercel e Heroku.

Referências

SARAF, Pranay D.; BARTERE, Mahip M.; LOKULWAR, Prasad P. A Review on Evolution of Architectures, Services, and Applications in Computing Towards Edge Computing. In: **International Conference on Innovative Computing and Communications.** Springer, Singapore, 2022. p. 733-744.

ZAMMETTI, Frank. **Modern Full-Stack Development:** Using TypeScript, React, Node. js, Webpack, and Docker. Apress, 2020.

React: Uma biblioteca JavaScript para criar interfaces de usuário. Disponível em: https://pt-br.reactjs.org/. Acesso em: 10 dez. 2021.

Flask: Web Development, one drop at a time. Disponível em: https://flask.palletsprojects.com/en/2.0.x/. Acesso em: 10 Dez de 2021.

Node.js: a JavaScript runtime built on Chrome's V8 JavaScript Engine. Disponível em: https://nodejs.org/en/. Acesso em: 10 Dez de 2021.