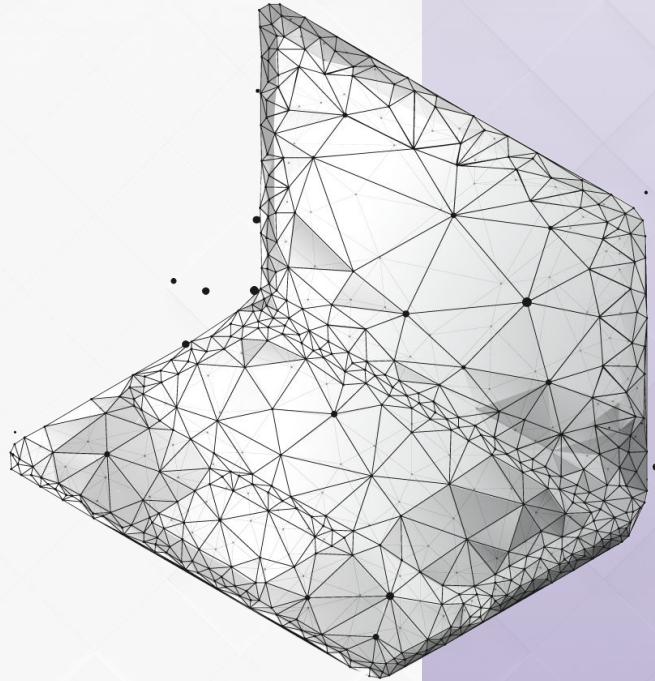


Parte 2



Compreendendo o Stack Tecnológico





Overview

- Esta seção se concentra em explicar o que é uma stack tecnológica, stacks e frameworks comuns usados por startups, e como as escolhas em torno da stack tecnológica afetam escalabilidade, crescimento do time, velocidade de desenvolvimento e outras áreas críticas.
- O objetivo é equipar os analistas de capital de risco com conhecimento suficiente para fazer as perguntas certas sobre a arquitetura de tecnologia de uma empresa e identificar potenciais riscos ou pontos fortes.





Revisitando a última aula



feedbacks da
última aula:

3.9



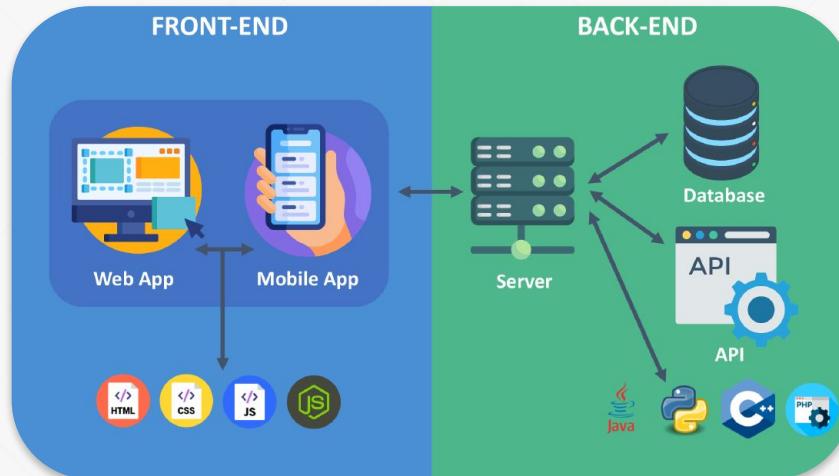
Erratas e
esclarecimentos



Discussão sobre
startups que se
encaixam em cada
tipo de empresa de
tecnologia (**dever de
casa**)



⚙️ O que é um stack tecnológico?

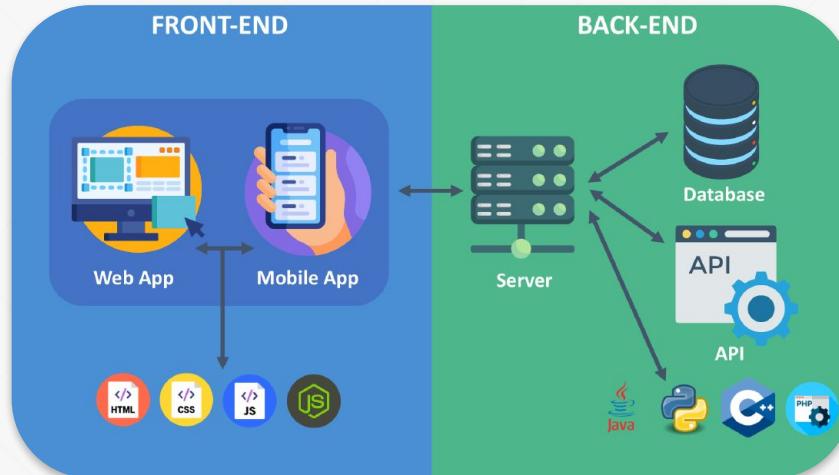


Um stack tecnológico refere-se à combinação de tecnologias, frameworks e ferramentas que uma empresa usa para construir e executar suas aplicações. Geralmente, é dividido em várias camadas, cada uma desempenhando um papel crítico na entrega do produto final aos usuários. **Estas camadas incluem:**

Front-end

- O **lado do cliente** da aplicação, responsável pelo que os usuários interagem diretamente.
- Inclui linguagens e frameworks como HTML, CSS, JavaScript, React, Angular e Vue.js.
- Determina a **experiência do usuário (UX)** e a responsividade do produto.

⚙️ O que é um stack tecnológico?

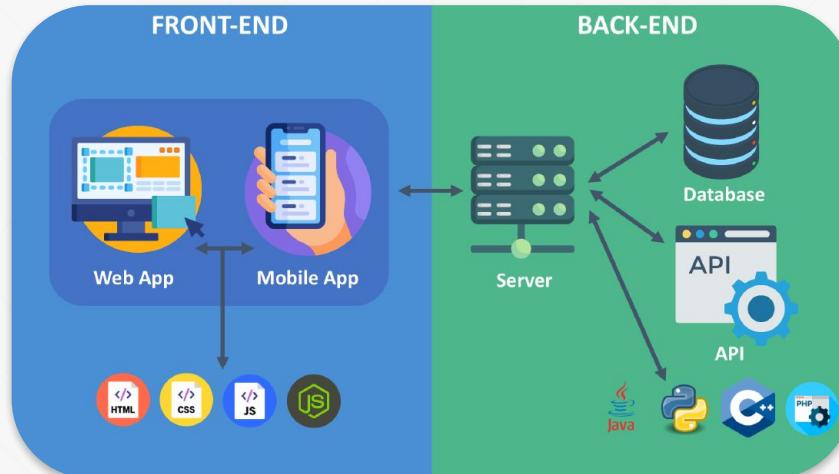


Um stack tecnológico refere-se à combinação de tecnologias, frameworks e ferramentas que uma empresa usa para construir e executar suas aplicações. Geralmente, é dividido em várias camadas, cada uma desempenhando um papel crítico na entrega do produto final aos usuários. **Estas camadas incluem:**

Back-end

- O **lado do servidor** da aplicação, gerenciando lógica, processamento de dados e comunicação com o banco de dados.
- Inclui linguagens como Node.js, Python, Ruby, PHP e Java, além de frameworks como Express.js, Django e Rails.
- Potencializa a **funcionalidade central da aplicação** (ex: autenticação de usuários, manipulação de APIs, etc.).

⚙️ O que é um stack tecnológico?



Um stack tecnológico refere-se à combinação de tecnologias, frameworks e ferramentas que uma empresa usa para construir e executar suas aplicações. Geralmente, é dividido em várias camadas, cada uma desempenhando um papel crítico na entrega do produto final aos usuários. **Estas camadas incluem:**

Mobile

- O desenvolvimento de **aplicativos móveis** frequentemente tem seu próprio stack dedicado, como:
 - Nativo (ex: Swift para iOS, Kotlin para Android)
 - Frameworks multiplataforma (ex: React Native, Flutter)
- Permite que as empresas desenvolvam aplicações que funcionem perfeitamente em **dispositivos móveis**.

Banco de dados

O sistema utilizado para armazenar e recuperar dados para a aplicação.



Bancos de Dados SQL:

Bancos de dados relacionais como PostgreSQL, MySQL (**dados estruturados, forte consistência**).



Bancos de Dados NoSQL:

MongoDB, Cassandra (dados não estruturados ou semi-estruturados, **esquema flexível, escalabilidade**).



BDs de Propósito específico:

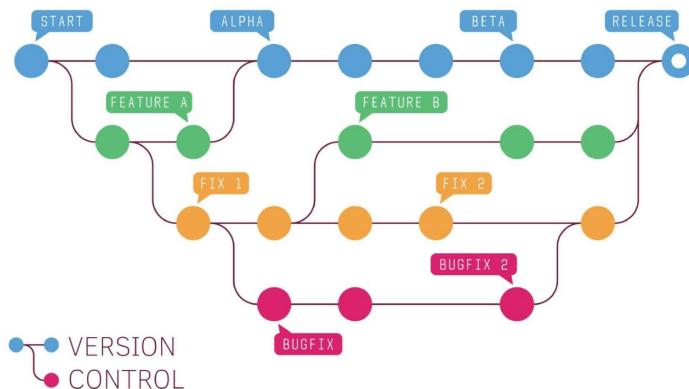
- Grafos (Neo4J)
- Busca (ElasticSearch)
- Caching (Redis)

Ferramentas de Testes

- **Testes unitários:** usando frameworks como Jest para JavaScript, PyTest para Python
- **Testes de integração:** ferramentas como Selenium para testes de interface do usuário
- **Testes funcionais:** verificam se o sistema está funcionando conforme as especificações e requisitos do usuário, geralmente usando ferramentas como Cucumber ou Behave
- **Testes de carga:** utilizando ferramentas como Apache JMeter ou Gatling para simular tráfego pesado



Versionamento de código



- **Sistemas de controle de versão** como Git são essenciais para **rastrear mudanças no código** ao longo do tempo
- Plataformas de hospedagem de código como GitHub, GitLab ou Bitbucket facilitam a **colaboração entre desenvolvedores**
- **Ferramentas de revisão de código** integradas ajudam a manter a qualidade e consistência do código



Containers e “dockerização”

- Tecnologias de containerização como Docker permitem empacotar aplicações e suas dependências em **unidades isoladas e portáteis**
- Facilita a **implantação consistente** em diferentes ambientes, desde desenvolvimento até produção
- Orquestração de containers com Kubernetes permite gerenciar e escalar aplicações containerizadas de forma eficiente



Ferramentas de BI e análise de dados

- Ferramentas como Tableau, Power BI ou Looker são utilizadas para **visualização e análise de dados**
- Plataformas de business intelligence ajudam as empresas a tomar **decisões baseadas em dados**
- Integração com fontes de dados diversas permite uma **visão holística do desempenho do negócio**



Pipeline de Dados

- Envolve ferramentas e frameworks utilizados para **extrair, transformar e carregar dados** (processos ETL).
- Tecnologias incluem Hadoop, Apache Spark, Kafka e Airflow.
- Essencial para empresas que utilizam big data, aprendizado de máquina e análise de dados em tempo real.



Google Cloud

Infraestrutura em Nuvem

- A plataforma onde a aplicação é hospedada, oferecendo serviços como computação, armazenamento e rede.
- Principais provedores: AWS (Amazon Web Services), Microsoft Azure, Google Cloud.
- A infraestrutura em nuvem permite **escalabilidade e flexibilidade**, reduzindo a necessidade de servidores físicos internos.



Porque isso importa para vocês

- Compreender os componentes básicos de um stack tecnológico ajuda os analistas a fazer perguntas informadas sobre a tecnologia que a startup está usando e como os componentes se encaixam para atender às necessidades do negócio.
- O stack tecnológico é um reflexo direto de como uma empresa aborda a experiência do usuário, o desempenho e a escalabilidade.





Para interação

O quanto você acha que a **stack tecnológica** escolhida pode **influenciar** no **sucesso ou fracasso** de uma tech startup?



**Não
influencia
nada**

**Influencia
muito**

Stacks tecnológicos e frameworks comuns usados por startups

Diferentes startups, dependendo de seu produto e necessidades de escalabilidade, usarão diferentes combinações de tecnologias para criar sua stack tecnológica.



Fullstack Frameworks Modernos (Next.js, Remix, Nuxt, Sveltekit, etc)

- Combinam frontend e backend em um único framework, **simplificando o desenvolvimento**
- Oferecem recursos avançados como renderização do lado do servidor (SSR) e **hidratação progressiva**
- **Casos de Uso Comuns:** Startups que buscam desenvolvimento rápido e eficiente de aplicações web modernas e escaláveis





Content Frameworks Modernos (Gatsby, Astro, etc)

- Frameworks focados em **geração de conteúdo estático**, otimizados para performance e SEO
- Oferecem recursos avançados como geração de páginas em tempo de compilação (SSG) e integração com sistemas de gerenciamento de conteúdo (CMS)
- **Casos de Uso Comuns:** Blogs, sites de marketing, documentação de produtos e outros sites com conteúdo frequentemente atualizado



Dev Stack Clássicos (MEAN, MERN, MEVN, JAMstack, LAMP, etc)

- Estruturas de desenvolvimentos flexíveis, que dão espaço para **customização** para encaixar em quase qualquer contexto
- Essas estruturas combinam diferentes tecnologias para criar um stack completo, como:
 - **MEAN**: MongoDB, Express.js, Angular, Node.js
 - **MERN**: MongoDB, Express.js, React, Node.js
 - **MEVN**: MongoDB, Express.js, Vue.js, Node.js
 - **JAMstack**: JavaScript, APIs, Markup
 - **LAMP**: Linux, Apache, MySQL, PHP
- **Casos de Uso Comuns**: Startups que precisam de flexibilidade para adaptar seu stack conforme crescem e evoluem suas necessidades tecnológicas.





Fullstack Frameworks Clássicos (Django, Rails, Laravel, Asp.net, Spring, etc)

- Frameworks fullstack clássicos oferecem ferramentas e convenções para desenvolver **aplicações web completas**.
- Seguem uma abordagem integrada e opinativa, priorizando convenções sobre configurações.
- Oferecem um conjunto abrangente de ferramentas e bibliotecas integradas, facilitando o desenvolvimento rápido de aplicações complexas
- Geralmente seguem o padrão MVC (Model-View-Controller), promovendo uma estrutura organizada e de fácil manutenção
- **Casos de Uso Comuns:** Startups que buscam desenvolver aplicações web robustas, especialmente quando a produtividade e a consistência do código são prioridades, em detrimento da escalabilidade



Frameworks Mobile (React Native, Flutter, NativeScript, etc)



- Frameworks para desenvolvimento de **aplicativos móveis multiplataforma**
- Permitem criar aplicativos para iOS e Android usando uma única base de código
- **Casos de Uso Comuns:** Startups que desejam lançar rapidamente em várias plataformas móveis com recursos limitados



NativeScript



No-code / low-code stacks (Bubble, Adalo, Glide, Webflow, Wix, Airtable, Zapier, Make, etc)

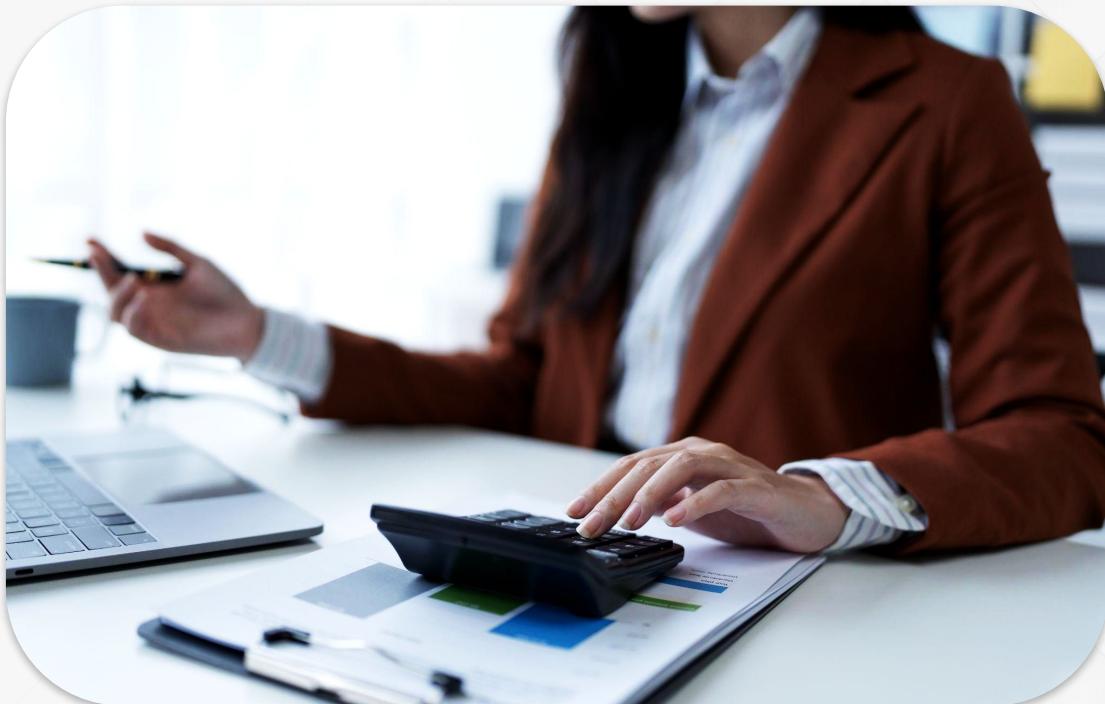
- Plataformas que permitem criar aplicações **sem (ou com mínimo) conhecimento de programação**
- Aceleram o desenvolvimento de protótipos e MVPs, permitindo validar ideias rapidamente
- **Casos de Uso Comuns:** Startups em estágio inicial, empreendedores não-técnicos, e empresas que buscam criar soluções internas rapidamente





Porque isso importa para vocês

- Compreender o stack usado pela startup fornece insights sobre sua velocidade de desenvolvimento, escalabilidade futura e potenciais restrições.
- Saber quais tecnologias são populares na indústria ajuda os analistas a avaliar se a empresa está usando frameworks atualizados e bem suportados ou soluções mais antigas e menos escaláveis.



Um overview de linguagens e frameworks

Linguagem/ Framework	Overview	Paradigma de Programação	Principais Qualidades	Principais Fraquezas	Coisas Únicas	Principais Casos de Uso
 ERLANG	Erlang é uma linguagem funcional projetada para sistemas distribuídos, enquanto Elixir é um framework moderno que roda sobre a máquina virtual Erlang.	Funcional e concorrente	Alta escalabilidade, robustez, suporte a concorrência leve	Curva de aprendizado íngreme, menor adoção no mercado	BEAM para execução concorrente, imutabilidade de dados	Sistemas de telecomunicações aplicações em tempo real
	PHP é uma linguagem de script amplamente utilizada para desenvolvimento web, com kphp sendo uma versão otimizada que compila PHP em código nativo.	Imperativo e orientado a objetos	Facilidade de uso, ampla adoção na web, boa integração com bancos de dados	Segurança limitada, desempenho inferior em aplicações complexas	Extensa comunidade e suporte, flexibilidade com outras tecnologias	Desenvolvimento de sites dinâmicos, ecommerce
	Ruby on Rails é um framework web que facilita o desenvolvimento ágil de aplicações através do princípio "Convention over Configuration".	Orientado a objetos e MVC	Alta velocidade no desenvolvimento, boas práticas como DRY	Desempenho mais lento, flexibilidade limitada	Sistema integrado de ORM (ActiveRecord)	Startups, aplicações web complexas

Um overview de linguagens e frameworks

Linguagem/ Framework	Overview	Paradigma de Programação	Principais Qualidades	Principais Fraquezas	Coisas Únicas	Principais Casos de Uso
	Node.js é um ambiente de execução JavaScript no lado do servidor que permite construir aplicações escaláveis e rápidas.	Assíncrono e orientado a eventos	Alta performance em operações I/O, unificação do desenvolvimento front-end e back-end	Curva de aprendizado para callbacks, desafios em aplicações complexas	Uso do JavaScript em ambos os lados da aplicação	Aplicações em tempo real, API RESTful
	.NET é uma plataforma da Microsoft que suporta várias linguagens, sendo C# a mais popular para construir aplicações robustas.	Orientado a objetos	Alto desempenho, suporte robusto para aplicações empresariais	Dependência do ecossistema Microsoft, curva de aprendizado para novos desenvolvedores	Integração com ferramentas Microsoft, suporte a múltiplas linguagens	Aplicações empresariais, desenvolvimento web com ASP.NET Core
	Java é uma linguagem orientada a objetos amplamente utilizada em ambientes corporativos e aplicativos móveis.	Orientado a objetos	Portabilidade através da JVM, robustez e segurança	Verbosidade da linguagem, desempenho inferior a linguagens compiladas nativamente	Ampla adoção em sistemas corporativos e aplicativos Android	Desenvolvimento de aplicativos empresariais, aplicativos móveis para Android

Um overview de linguagens e frameworks

Linguagem/ Framework	Overview	Paradigma de Programação	Principais Qualidades	Principais Fraquezas	Coisas Únicas	Principais Casos de Uso
 python ™	Python é uma linguagem versátil conhecida por sua simplicidade e legibilidade, amplamente usada em diversas áreas.	Multiparadigma	Sintaxe simples e legível, grande variedade de bibliotecas	Desempenho inferior a linguagens compiladas, gerenciamento de memória desafiador em grandes aplicações	Comunidade ativa com pacotes opensource	Desenvolvimento web com Django/Flask, análise de dados

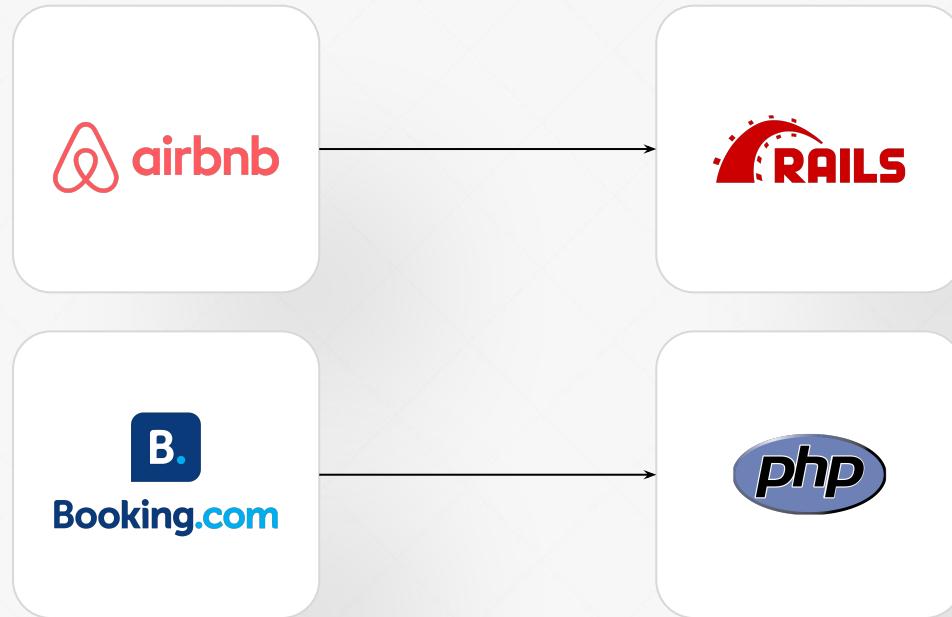
⚙️ Conhecendo stacks de grandes empresas



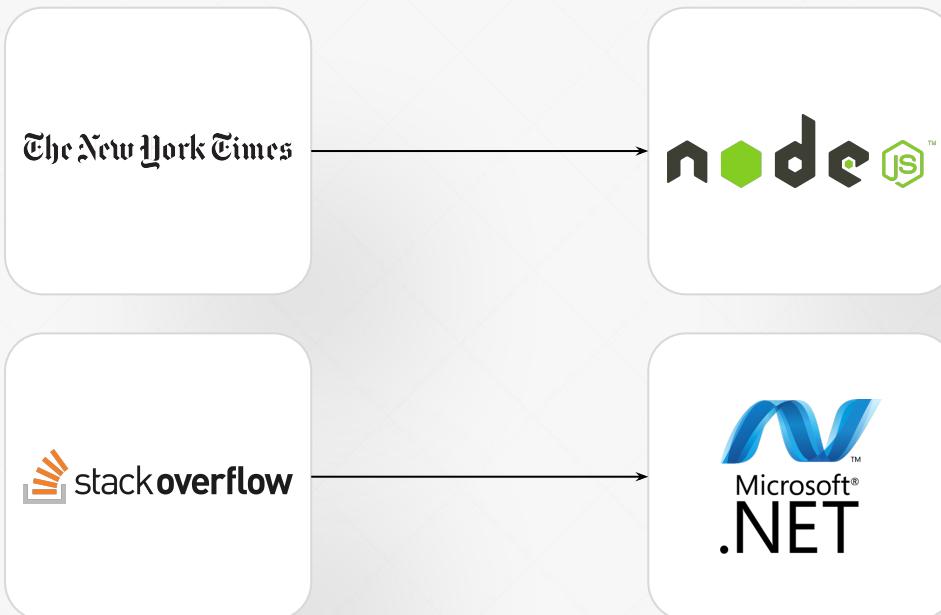
WhatsApp tinha 55 funcionários quando foi vendida por \$19bi

Telegram tem 30 funcionários e vale \$30bi

⚙️ Conhecendo stacks de grandes empresas



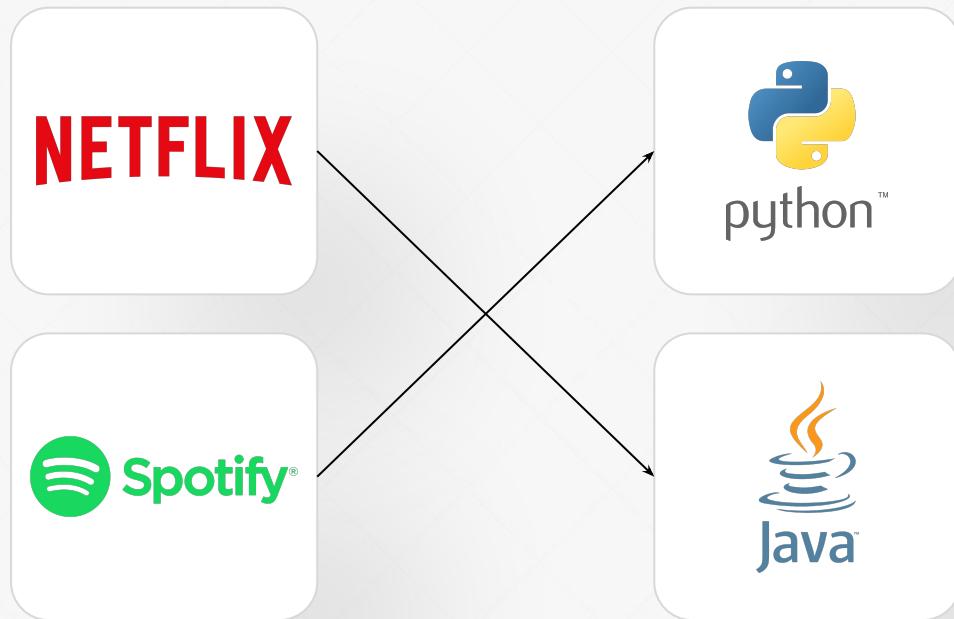
⚙️ Conhecendo stacks de grandes empresas



⚙️ Conhecendo stacks de grandes empresas



⚙️ Conhecendo stacks de grandes empresas



Como a escolha da stack tecnológica afeta uma startup

- Velocidade de desenvolvimento



Linguagens e frameworks: ✓

Escolher um framework que permita **desenvolvimento rápido** pode fazer uma grande diferença para uma startup. Alguns frameworks (como React e Node.js) são projetados para ajudar os desenvolvedores a avançar rapidamente, com um **rico ecossistema de bibliotecas e ferramentas**.

Facilidade de manutenção: ✓

Escolhas ruins em stacks tecnológicos podem resultar em **débitos técnicos**, onde o desenvolvimento rápido leva a problemas de longo prazo em manutenção e escalabilidade.

Testes, docker e outras ferramentas de desenvolvimento: ✓

Utilizar ferramentas como testes automatizados e Docker para padronização de ambientes de desenvolvimento pode evitar erros, melhorar a colaboração entre times e **garantir um desenvolvimento mais ágil e confiável**.

Como a escolha da stack tecnológica afeta uma startup

- Crescimento, desenvolvimento e flexibilidade de time



Padronização de tecnologias: ✓

A padronização de tecnologias melhora a consistência do código e a **colaboração na equipe**. Isso envolve convenções de codificação, ferramentas de linting e formatação automática, e padrões de arquitetura. Essa abordagem acelera o desenvolvimento, **reduz erros e facilita a manutenção** a longo prazo.

Facilidade de aprendizado: ✓

A escolha de tecnologias que sejam fáceis de aprender e bem documentadas facilita a **integração de novos desenvolvedores** e promove a **flexibilidade da equipe**. Ferramentas e frameworks com uma curva de aprendizado suave permitem que a equipe **cresça rapidamente e mantenha uma produtividade elevada**, mesmo com novos membros.

Facilidade de Contratação: ✓

Adotar tecnologias populares e amplamente conhecidas pode **facilitar a contratação** de novos membros. Frameworks e linguagens com comunidades fortes e ampla documentação atraem mais candidatos, acelerando o processo de recrutamento e facilitando a integração de novos desenvolvedores na equipe.

Como a escolha da stack tecnológica afeta uma startup

- Crescimento, desenvolvimento e flexibilidade de time



Documentação técnica:

Uma documentação técnica clara é crucial para o crescimento da equipe, facilitando a **integração de novos membros** e mantendo a **consistência do código**. Ela reduz o **tempo de onboarding**, serve como referência e ajuda a manter práticas de desenvolvimento consistentes, mesmo com rotatividade de equipe.

Facilidade de manutenção:

O débito técnico deve ser monitorado e gerenciadoativamente. Acúmulos de problemas de código mal projetado ou não escalável podem **desacelerar o desenvolvimento no futuro**, prejudicando a flexibilidade da equipe e a capacidade de escalar rapidamente.

Testes, docker e outras ferramentas de desenvolvimento:

Testes automatizados, Docker e ferramentas de DevOps são cruciais para **qualidade, consistência e eficiência** no desenvolvimento. Testes unitários, de integração e end-to-end detectam problemas cedo, reduzindo o tempo de depuração. Docker cria ambientes consistentes, evitando o **"funciona na minha máquina"**. Ferramentas de CI/CD automatizam build, teste e implantação, **acelerando o ciclo** e minimizando erros humanos.

Como a escolha da stack tecnológica afeta uma startup

- Escalabilidade



Linguagens e frameworks:

Certas linguagens e frameworks são mais escaláveis que outros. Go e Rust se destacam em sistemas de alta concorrência. Node.js (Express) e Django (Python) oferecem boa escalabilidade horizontal. A escolha deve considerar tanto o **desempenho atual** quanto as **necessidades futuras** da startup.

Bancos de dados:

A escolha do banco de dados impacta diretamente a **escalabilidade**. Bancos de dados relacionais como PostgreSQL oferecem consistência forte, mas podem enfrentar desafios de escalabilidade em cargas muito altas. Bancos de dados NoSQL como MongoDB ou Cassandra são projetados para escalar horizontalmente, mas podem sacrificar algumas garantias de consistência.

Testes, docker e outras ferramentas de desenvolvimento:

Ferramentas como Git, CI/CD e Docker são cruciais para a **escalabilidade do desenvolvimento**. Elas facilitam colaboração, integração contínua e consistência entre ambientes. Testes automatizados (unitários, integração, e2e) são essenciais para manter a qualidade do código conforme o projeto cresce.

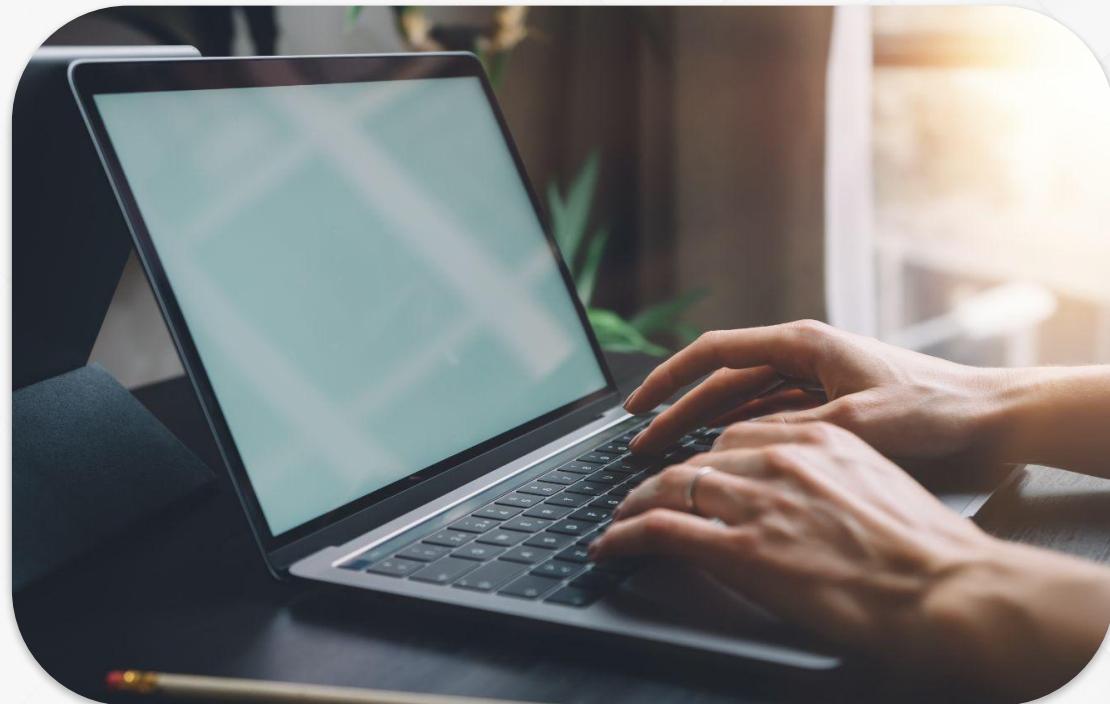
Débitos técnicos:

Débito técnico precisa ser constantemente monitorado em projetos que visam escalar. Escolhas tecnológicas ou soluções temporárias podem resultar em **grandes custos futuros**, impactando a capacidade de o sistema suportar maior número de usuários ou dados sem perda de desempenho ou confiabilidade.



Porque isso importa para vocês

- Os analistas devem entender os trade-offs entre desenvolvimento rápido e preocupações de escalabilidade ou segurança a longo prazo.
- Ser capaz de avaliar se uma startup fez escolhas sábias em seu stack tecnológico fornece aos analistas insights sobre a capacidade da empresa de lidar com o crescimento futuro e potenciais riscos de segurança.



Técnicas e perguntas para avaliar uma startup e stack tecnológica

- O principal é entender se as escolhas de tecnologia possuem um racional coerente e que se encaixam com o atual momento e conjunto de problemas da startup.
- Escolhas sem pé nem cabeça podem indicar fundadores técnicos que não possuem experiência o suficiente para prever e estar conscientes de problemas futuros, mesmo que as decisões atuais criem esses problemas.



Técnicas e perguntas para avaliar uma startup e stack tecnológica



- Por exemplo:

Startups no **early stage**

precisam validar o seu produto e encontrar seu mercado, então o principal aqui é velocidade dos ciclos de desenvolvimento e o quanto rápido o time consegue colocar um (ou múltiplos) protótipo, MVP ou funcionalidade na mão dos seus clientes, principalmente se chegar a hora de fazer um pivot completo.

Startups no **mid stage**

precisam crescer rápido, ganhar share de mercado, adquirir e reter clientes, então o principal aqui é o gerenciamento de complexidade sem perder velocidade, quanto rápido a startup consegue contratar e embarcar seus talentos, rotacionar times, escalar sua infraestrutura.

Startups no **late stage**

precisam escalar com qualidade, maximizar uptime, otimizar partes críticas do sistema, continuar evoluindo o produto em uma velocidade competitiva, porém em uma grande escala organizacional, onde governança e segurança se tornam muito importantes.

[Acessar perguntas de inspeção](#)





Análise de Tecnologias

Também é importante inspecionar se as escolhas de tecnologia, mesmo com founders conscientes, são boas escolhas. Para isso é importante analisar as escolhas de stack de tecnologia para as partes mais importantes do sistema.





Analisando Linguagens e DBs:

- Bom encaixe com domínio do problema
- Boa documentação
- Facilidade de aprendizado
- Popularidade e comunidade
 - Facilidade de contratação
 - Custo médio de mão de obra
 - Diversidade e maturidade de ferramentas e bibliotecas
 - Uso e patrocínio de grandes empresas
- Outros aspectos: Performance, segurança, tolerância a erros



Analisando frameworks, bibliotecas e ferramentas:

- Licença
- Boa documentação
- Facilidade de aprendizado
- Popularidade e comunidade
- Uso e patrocínio de grandes empresas
- Atualização e segurança

Algumas ferramentas

<https://www.thoughtworks.com/radar>

- Verificar se tecnologia ainda é recomendada

<https://survey.stackoverflow.co/2024/technology/>

- Verificar popularidade e adoção das tecnologias

<https://2023.stateofjs.com/en-US>

- Verificar popularidade e adoção das tecnologias

State of Web Development

[**State of Web Development.pdf**](#)



<https://github.com/>

- Verificar quantidade de estrelas, forks, issues, PRs
- Verificar data e última atualização de versão e frequências de atualização
- Verificar contribuidores e dependências



Para interação

- O quanto você acha que a stack tecnológica escolhida pode influenciar no sucesso ou fracasso de uma tech startup?



Não
influencia
nada

Influencia
muito



Porque isso importa para vocês

- Os analistas devem ser capazes de reconhecer quando uma stack tecnológica está bem estruturada para o crescimento futuro e quando pode apresentar riscos ocultos, como débitos técnicos ou dificuldades para escalar.
- Identificar sinais de alerta na stack tecnológica pode prevenir problemas futuros onerosos, especialmente em startups de alto crescimento.

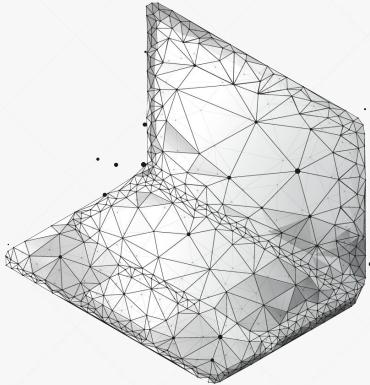




Exercício para casa

- Buscar startups **do seu portfólio investido ou não e investigar as escolhas de tecnologia**, por que elas foram feitas e se tiveram impacto positivo ou negativo para o crescimento da startup.
- Analisar uma startup do seu portfólio investido ou não com as perguntas de **inspeção e análise de tecnologias**, e evidenciar os **green, yellow e red flags**, e a qualidade das tecnologias escolhidas.





SBC School



Resultados

Ao final desta seção, os analistas de capital de risco terão uma compreensão sólida do que é uma stack tecnológica, quais diferentes componentes estão envolvidos e como as escolhas feitas por uma startup em relação a sua stack impactam escalabilidade, crescimento de time e velocidade de desenvolvimento.

Os analistas também serão capazes de reconhecer sinais de alerta em uma stack tecnológica mal planejado, o que pode afetar a viabilidade a longo prazo da empresa.

Bibliografia

<https://stackshare.io/>