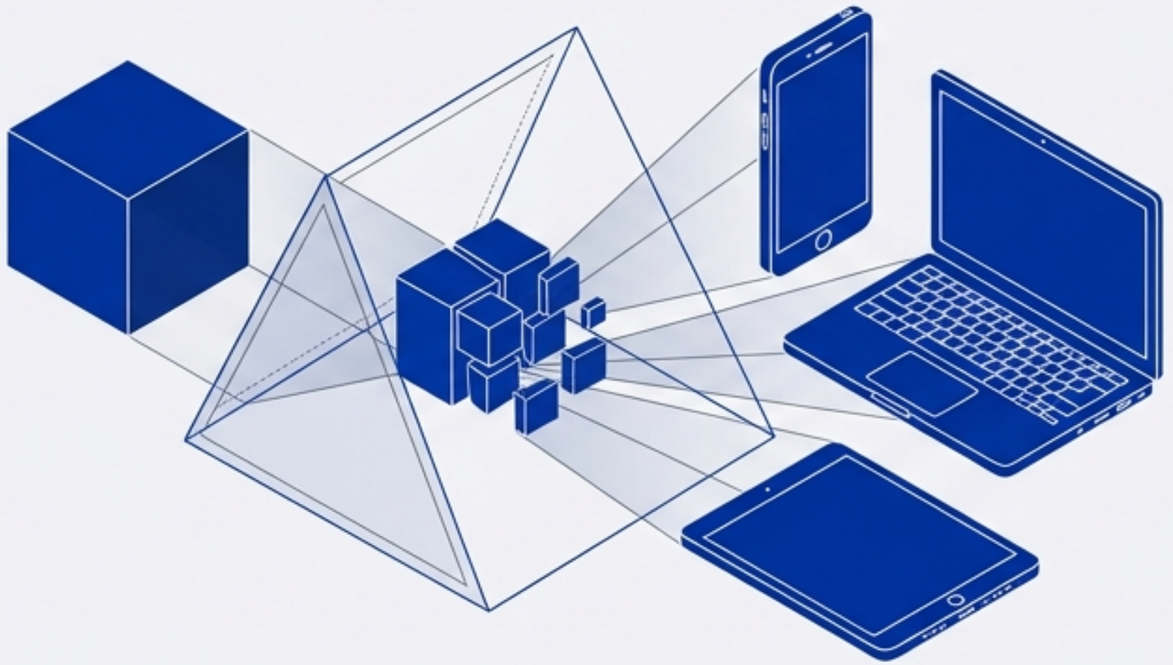
Desenvolvimento de Software Multiplataforma

A arte de fazer um software “falar várias línguas tecnológicas” sem perder a identidade.



DEFINIÇÃO

A prática de criar aplicações capazes de funcionar em diferentes sistemas operacionais (Windows, Linux, Android, iOS) com o mínimo de adaptações.

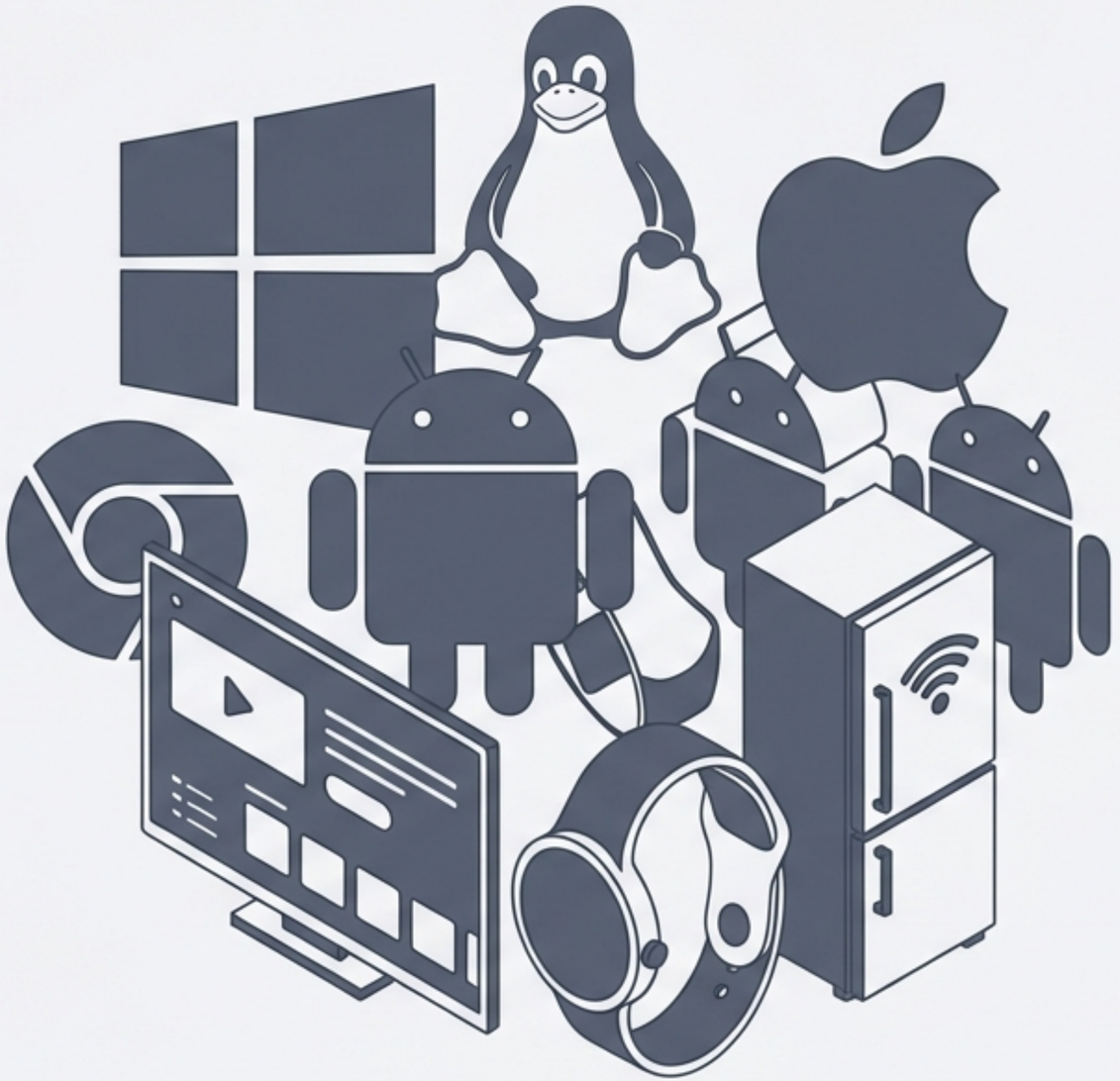
O OBJETIVO

Uma única base de código. Múltiplas experiências.

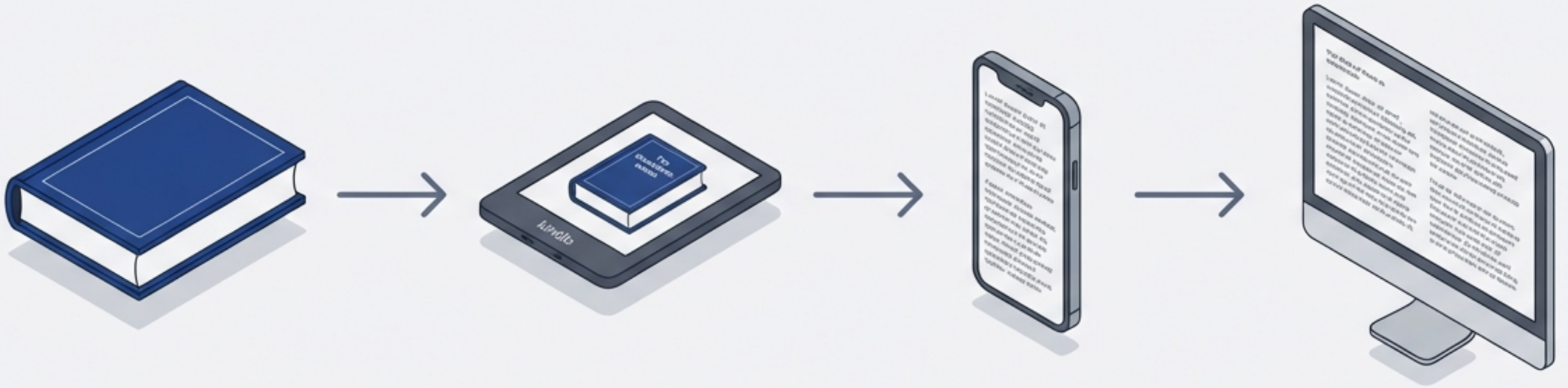
O Mundo Digital é Fragmentado

- Custo altíssimo de desenvolvimento.
- Tempo de entrega duplicado ou triplicado.
- Manutenção transformada em pesadelo logístico.

O desenvolvimento nativo exclusivo para cada plataforma não é escalável para todas as empresas.



Escreva uma vez, execute em vários lugares



A Metáfora do Livro: O conteúdo (código) permanece o mesmo, apenas o suporte (device) muda.

1

Redução de Custos e Tempo

2

Manutenção Centralizada

3

Padronização da Experiência

Os Três Pilares da Execução

1. Aplicações Web

O navegador atua como o "intérprete universal" do código.

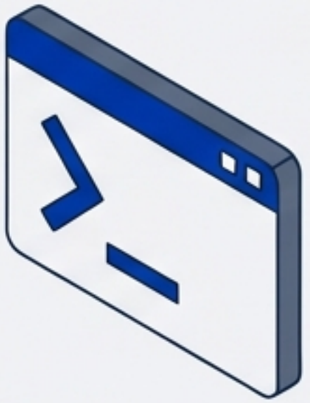
`HTML, CSS, JS, React, Vue, Angular`

2. Frameworks Multiplataforma

Tradutores automáticos que geram aplicativos nativos a partir de um código único.

`React Native, Flutter, Electron, .NET MAUI`

3. Linguagens Suportadas



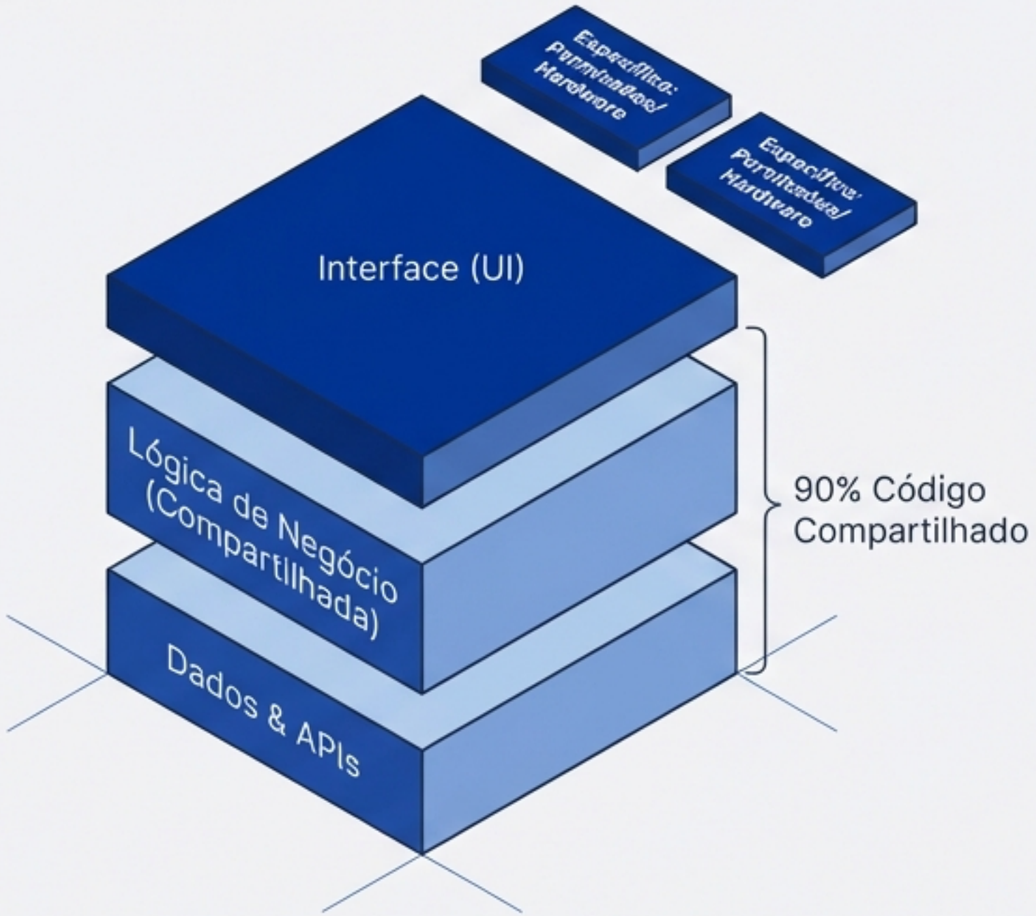
Portabilidade garantida através de Máquinas Virtuais (VMs) ou Runtimes.

`Java, Python, C#`

Arquitetura: A Defesa Contra o Caos

Sem uma arquitetura definida, o código vira um "espaguete interplanetário".

ARQUITETURA IDEAL



SEM ARQUITETURA

O Grande Debate: Nativo vs. Multiplataforma

NATIVO

Código exclusivo para cada sistema.

- 🥇 Melhor performance possível
- 🥇 Experiência de UX totalmente adaptada
- 💸 Maior custo (Equipes separadas)
- 💸 Menor produtividade

MULTIPLATAFORMA

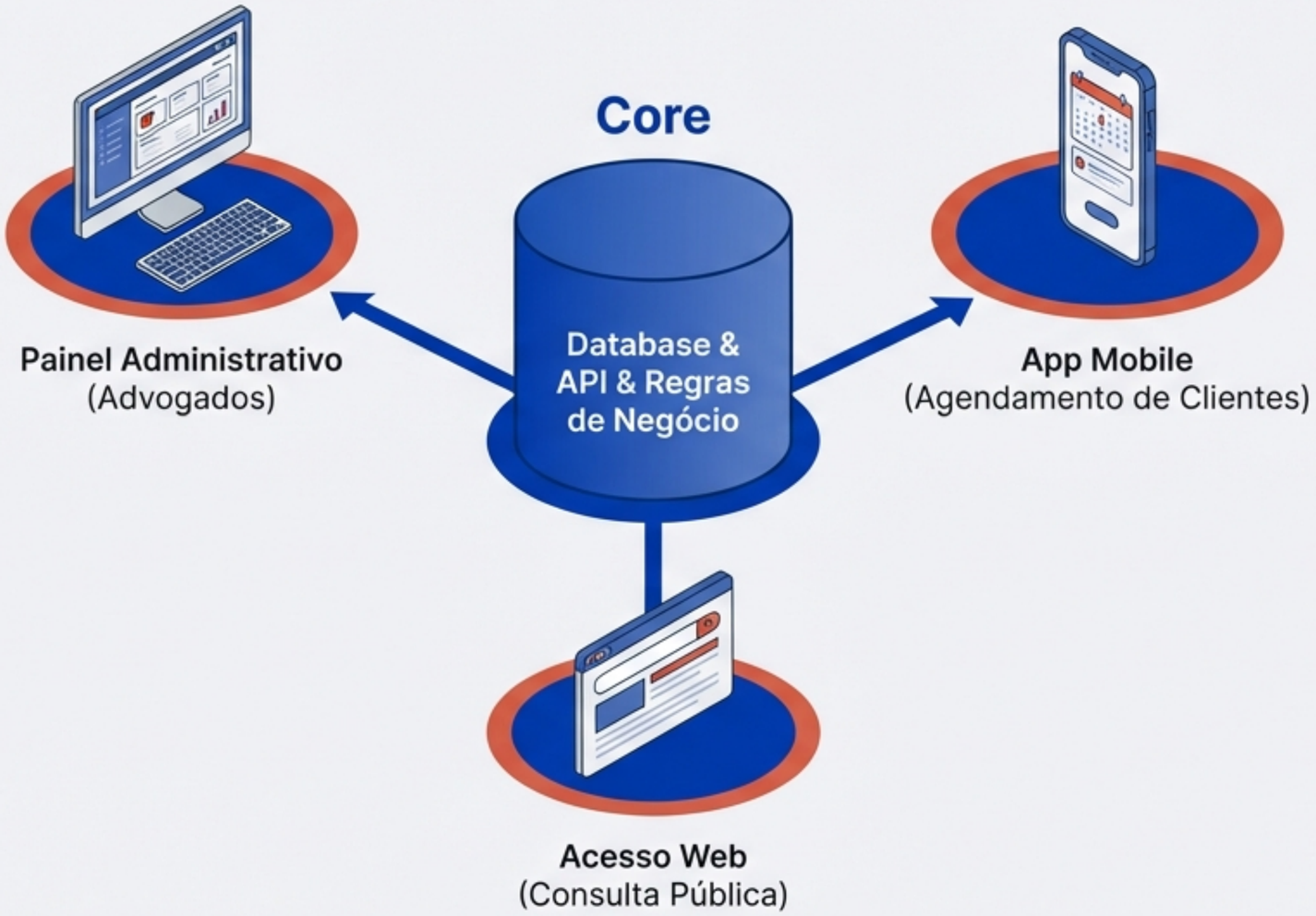
Código compartilhado e unificado.

- 🚀 Melhor produtividade (Time-to-market)
- 💰 Menor custo operacional
- ⚖️ Experiência padronizada
- 📉 Performance levemente inferior (em alguns casos)

Conclusão: Cada abordagem tem seu lugar estratégico dependendo do recurso e prazo disponível.

Aplicação no Mundo Real

Cenário: Sistema de gestão para escritório de advocacia.

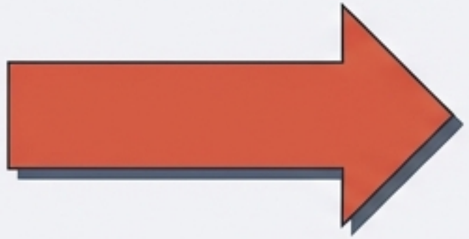


Eficiência Real: Tudo roda sob a mesma base de dados, a mesma regra de negócio e a mesma API. O código do ‘Core’ é escrito apenas uma vez.

A Missão Acadêmica: De Programador a Arquiteto

Plano de Ensino e Objetivos da Disciplina (80h)

Programador



Arquiteto de Software

De: Fazer código
Para: Pensamento **Arquitetural**

De: Tela única
Para: Visão **Sistêmica**

De: Framework
Para: Mentalidade de **Portabilidade**

Objetivo Principal: Capacitar o estudante a **projetar**, desenvolver e **implantar** aplicações modernas com **reutilização de código**.

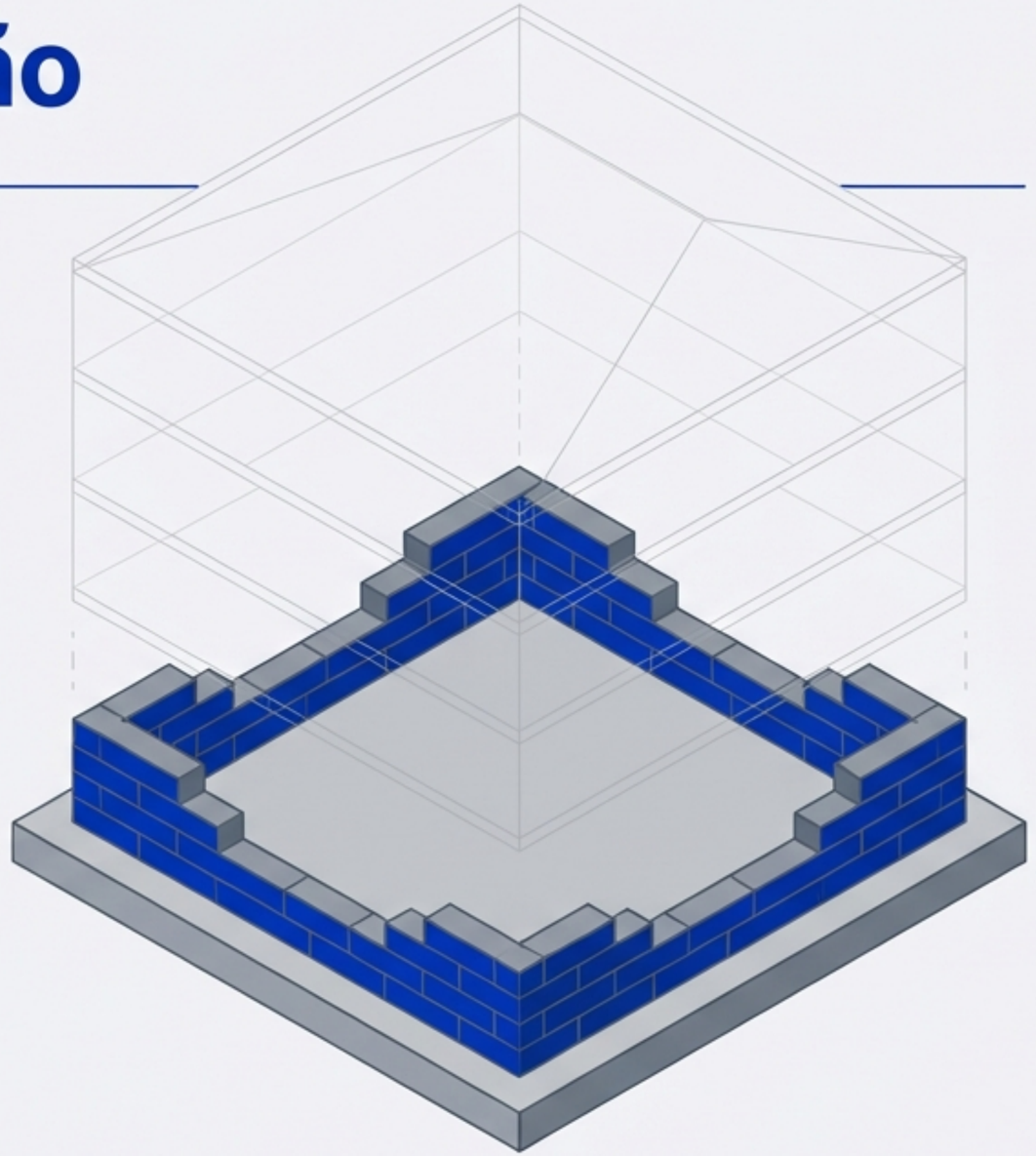
O Roadmap: A Fundação

Unidade I: Fundamentos

- Conceitos de responsividade
- Arquiteturas modernas (SPA, cliente-servidor)
- Introdução a APIs REST

Unidade II: Desenvolvimento Web

- HTML5, CSS3, JS Moderno e Manipulação de DOM
- Consumo de APIs com `fetch`
- Introdução a PWA (Manifest, Service Workers)
- Armazenamento local (LocalStorage)



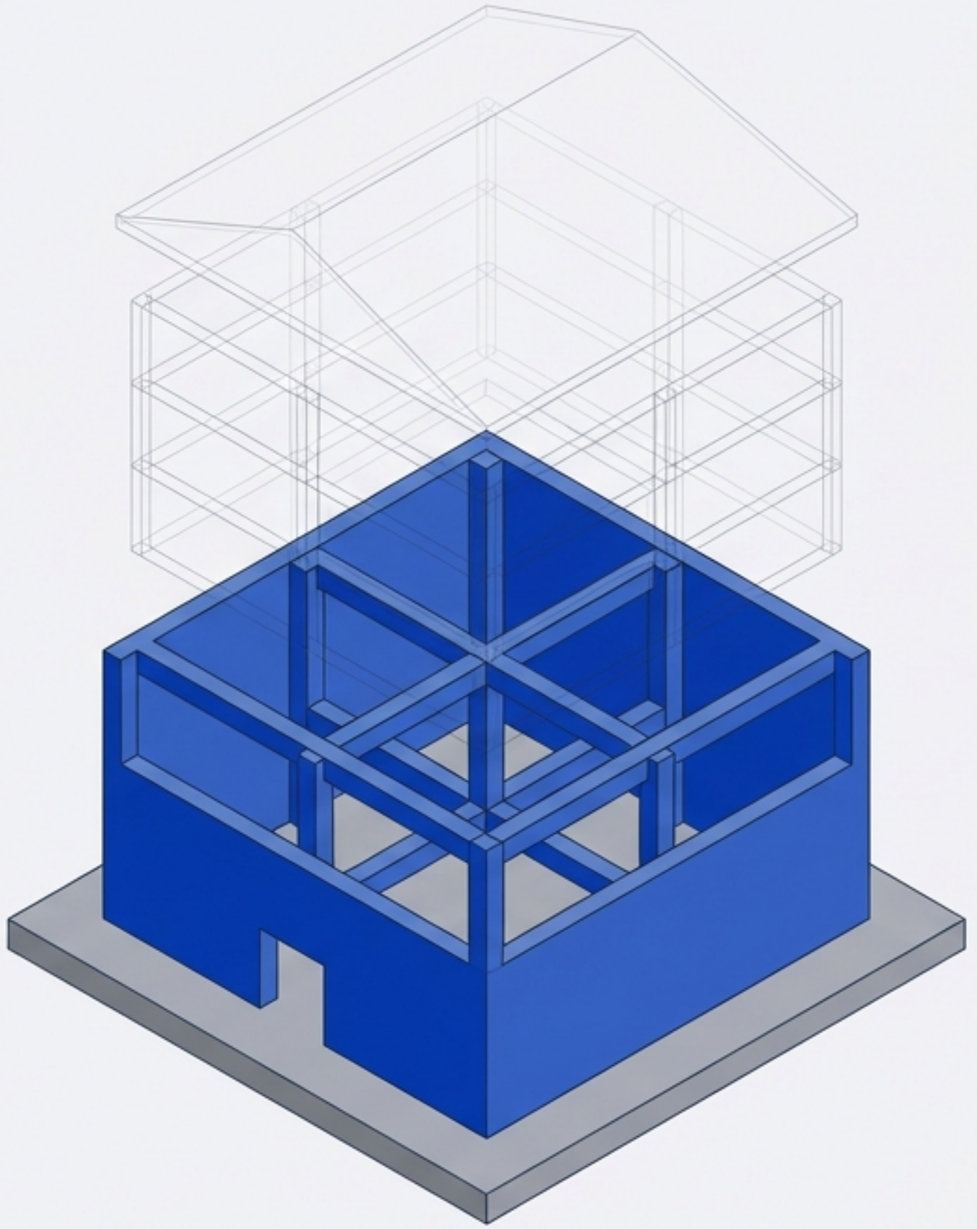
O Roadmap: A Construção

Unidade III: Mobile Híbrido

- Tecnologias: **React Native**, **Flutter** ou **Cordova**
- Componentização e Navegação entre telas
- Gerenciamento de estado
- Build e geração de APK

Unidade IV: Integração e Backend

- Noções com **Node.js** e **Express**
- Rotas, controladores e conexão com Banco de Dados
- **JSON** como padrão de comunicação de comunicação



O Roadmap: A Entrega

Unidade V: Deploy e Publicação

- Deploy Web (Vercel, Netlify ou servidor próprio)
- Publicação de aplicativos mobile
- Versionamento com `Git` e `GitHub`
- Integração Contínua (`CI/CD`) básica
- Boas práticas de segurança

Metodologia: Aprendizagem Baseada em Projetos

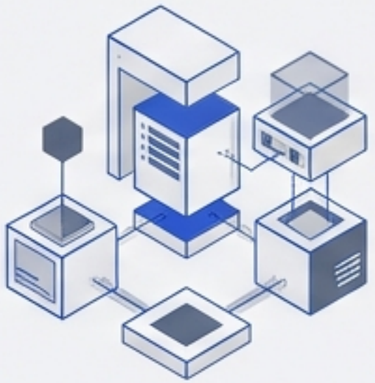
- Aulas expositivas **dialogadas** & **Sala de aula invertida**
- Simulação de **ambiente profissional**
- **Desafio**: Criar uma aplicação multiplataforma completa (Web + Mobile) com backend integrado.

Ferramentas do Ofício

IDE Principal

Runtime

Chrome/Edge/Firefox

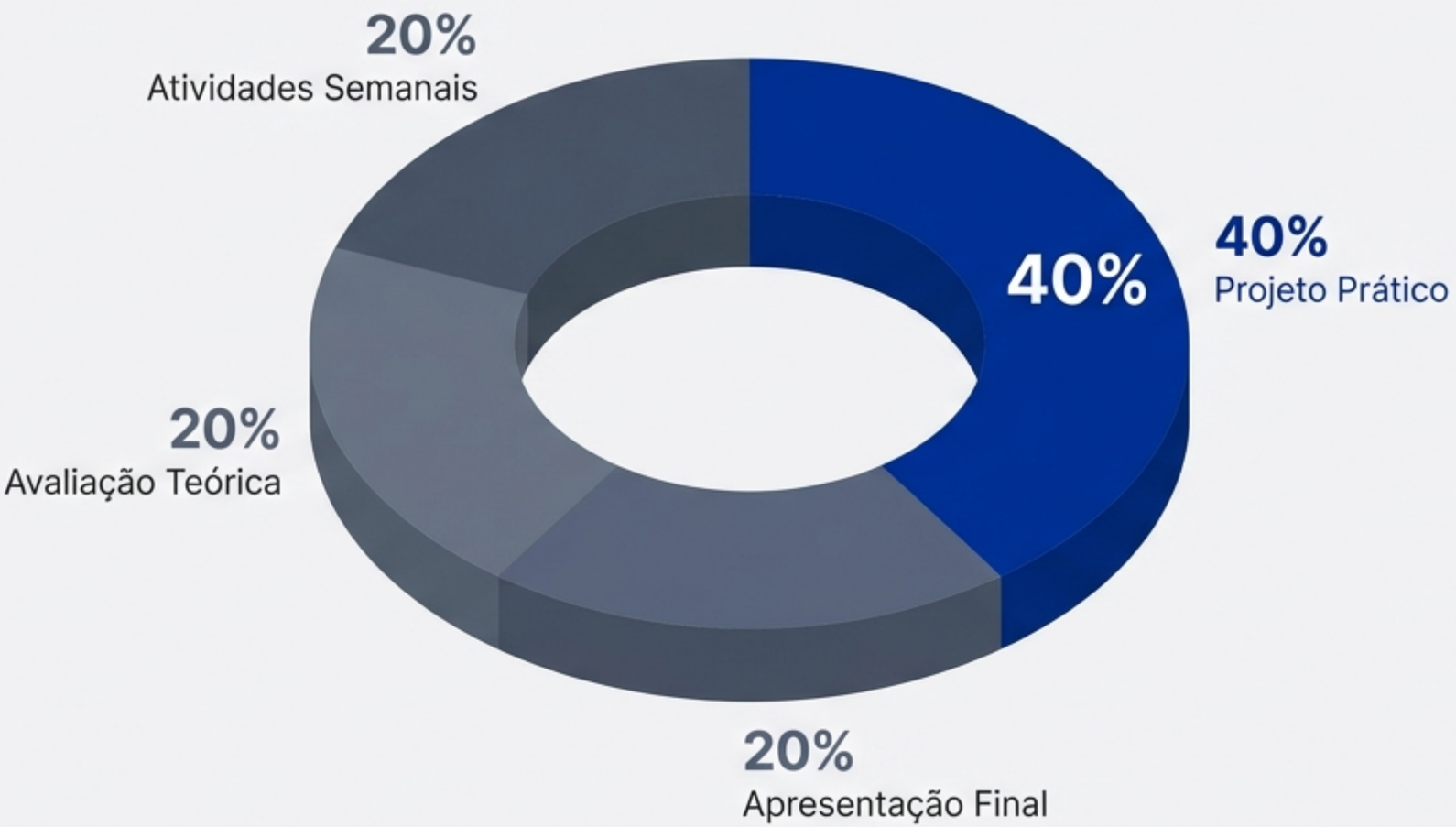


Emulador / Mobile

Versionamento

Frameworks

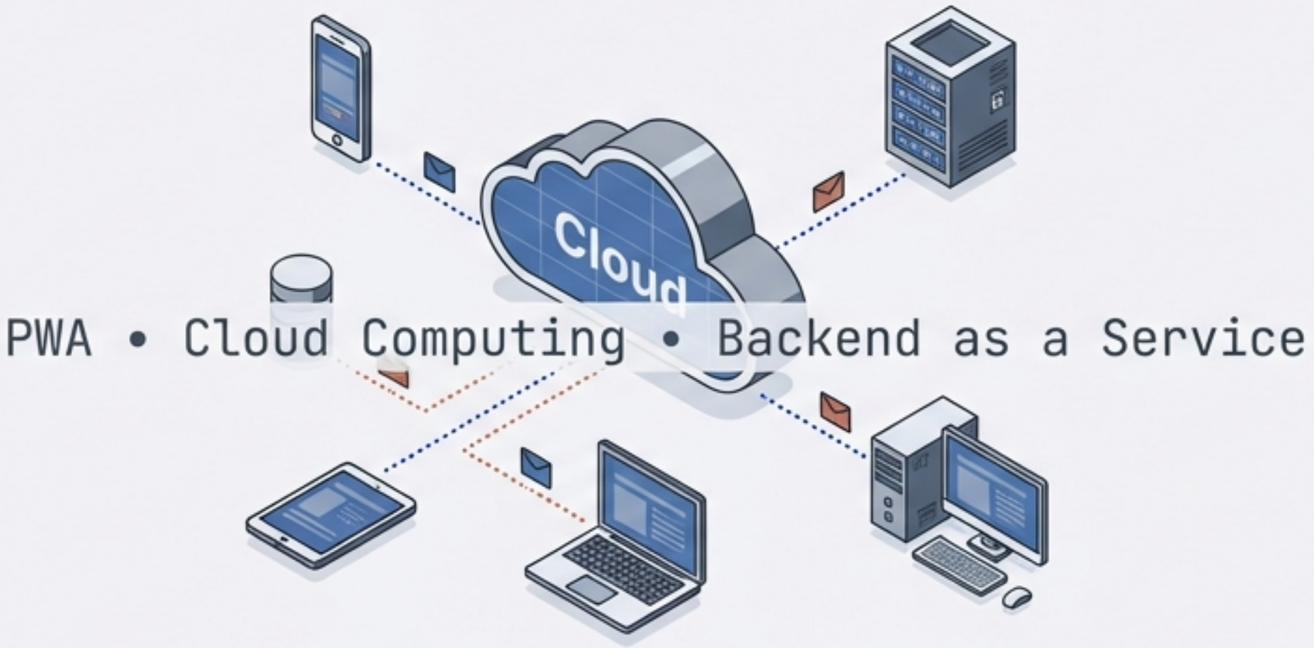
Avaliação e Métricas de Sucesso



Critérios de Qualidade

- Organização do código
- Integração entre camadas
- Interface e Usabilidade
- Documentação técnica

O Futuro é Onipresente



“Desenvolvimento de Software Multiplataforma é a abordagem estratégica de projetar aplicações capazes de operar em diferentes sistemas a partir de uma base de código compartilhada.”

EFICIÊNCIA • ESCALABILIDADE • CONSISTÊNCIA