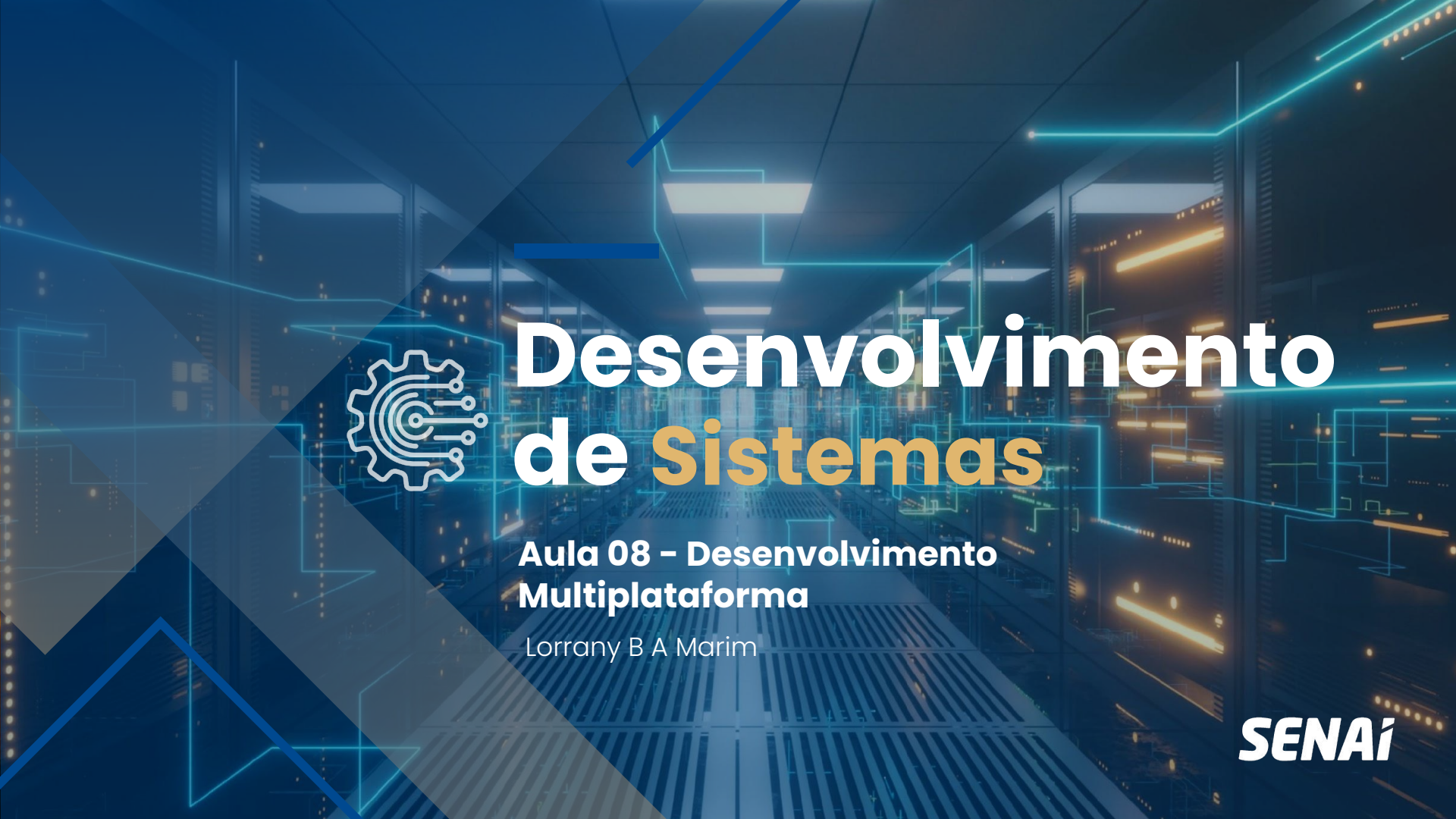


Desenvolvimento de Sistemas

Aula 08 - Desenvolvimento Multiplataforma

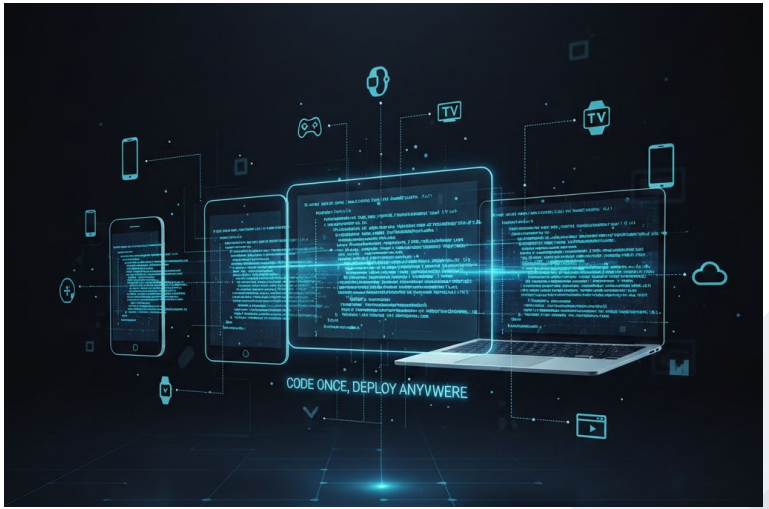
Lorrany B A Marim

SENAI

O Desafio da Portabilidade

Um código, muitos destinos

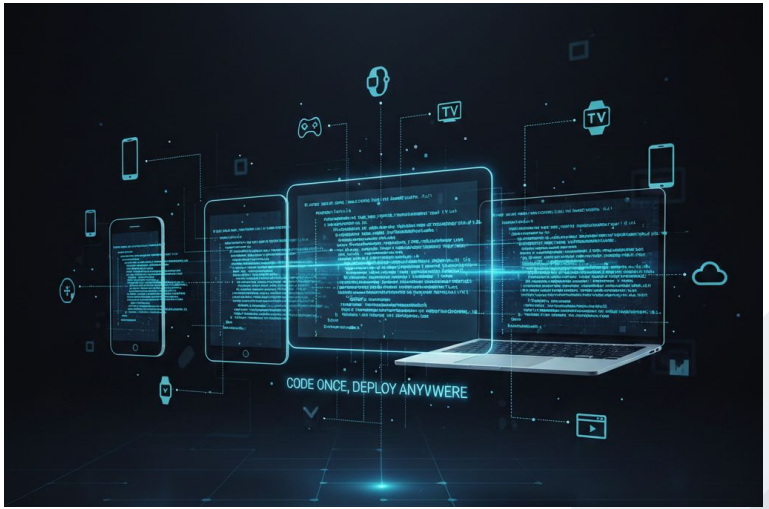
- **Definição:** Portabilidade é a capacidade de um software ser transferido de uma plataforma (sistema operacional ou hardware) para outra.
- **O Problema:** Diferentes sistemas (Windows, Android, iOS) possuem interfaces de programação (APIs) incompatíveis. Um software que depende muito de uma plataforma específica é difícil de portar.



O Desafio da Portabilidade

Um código, muitos destinos

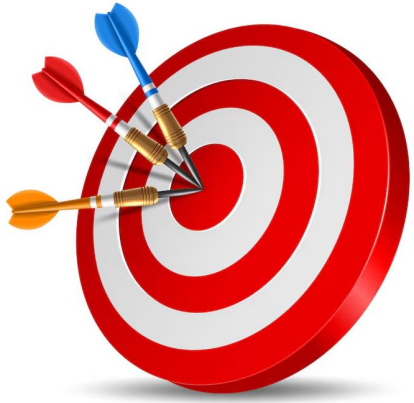
- **A Necessidade:** No mercado atual, espera-se que o sistema funcione em celulares, desktops e na web. Criar um código para cada um é caro e difícil de manter.
- **Solução:** Usar camadas de abstração (Frameworks) para mediar o nosso código e o sistema operacional.



Desenvolvimento Nativo

Foco total na plataforma

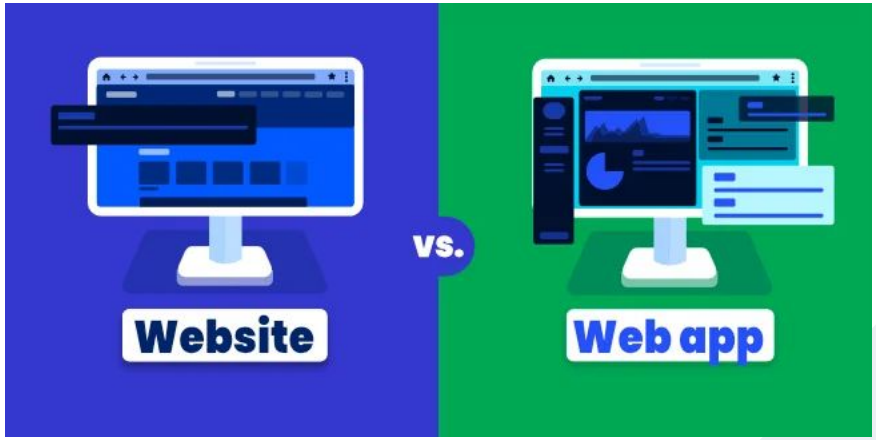
- **Conceito:** O software é escrito na linguagem oficial do sistema (ex: Swift para iOS, Kotlin para Android).
- **Vantagem:** Acesso total ao hardware e desempenho máximo, pois não há camadas extras de tradução.
- **Desvantagem:**
  - **Acoplamento:** O código fica dependente da plataforma. Se mudar o sistema, precisa reescrever.
  - **Custo:** É necessário manter múltiplas bases de código (uma para Android, uma para iOS, etc.), duplicando o esforço de desenvolvimento e testes.



Desenvolvimento Híbrido (Webview)

A Web dentro do App

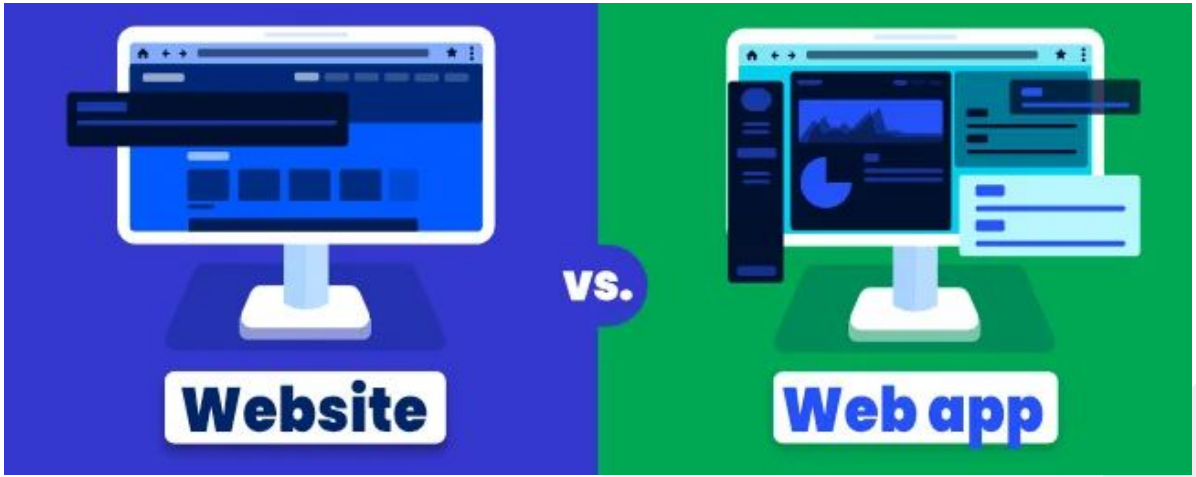
- **Conceito:** O aplicativo é, na verdade, um site (HTML, CSS e JS) rodando dentro de uma "casca" nativa (um componente de navegador embutido).
- **Funcionamento:** Usa-se um "Adapter" (Adaptador) para que o código Web consiga conversar com recursos nativos (câmera, GPS).



Desenvolvimento Híbrido (Webview)

A Web dentro do App

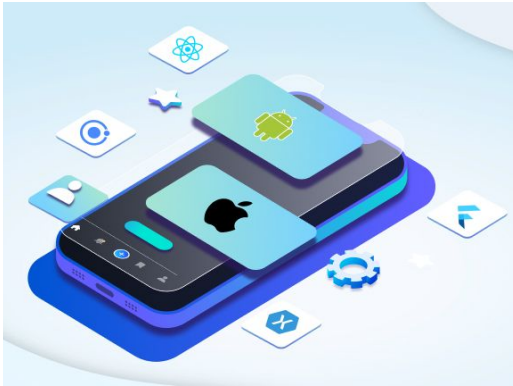
- **Ferramentas:** Ionic, Cordova.
- **Trade-off (Troca):** Ganha-se em velocidade de desenvolvimento (um código para tudo), mas perde-se em desempenho e a interface pode não parecer "natural" ao usuário.



Cross-Platform (Nativo)

A ponte entre mundos

- **Conceito:** Escreve-se em uma linguagem única (como JavaScript ou Dart), mas o Framework renderiza componentes visuais nativos ou desenha diretamente na tela (como num jogo), sem usar HTML.
- **Arquitetura (Padrão Bridge):** Essas ferramentas utilizam o conceito do padrão de projeto *Bridge* (Ponte). Elas separam a abstração (seu código) da implementação (o sistema operacional), permitindo que variem independentemente.
- **Resultado:** Desempenho muito próximo do nativo com uma única base de código.



Tecnologias Mobile (React e Flutter)

- **Flutter (Google):**
  - Usa a linguagem Dart.
  - **Diferencial:** Em vez de usar componentes do sistema, ele desenha cada pixel na tela (como uma engine de jogos). Isso garante que o app tenha a mesma aparência em qualquer versão do Android ou iOS.
  - É um *Framework* completo, ditando a arquitetura da aplicação.

Tecnologias Mobile (React e Flutter)

- **React Native (Facebook):**
  - Usa JavaScript.
  - **Filosofia:** "Aprenda uma vez, escreva em qualquer lugar".
  - Usa uma "Ponte" para comunicar o JavaScript com os componentes nativos do celular (botões, listas) em tempo de execução.

React Native (Via Expo)

O **Expo** é a forma mais simples e recomendada de iniciar com React Native, pois facilita a configuração do ambiente.

O que baixar:

1. **Node.js (LTS):** O motor que executa o JavaScript no computador.
2. **Git:** Para controle de versão e download de dependências.
3. **VS Code:** O editor de código mais utilizado.

React Native (Via Expo)

Como parametrizar (Windows):

- Após instalar o **Node.js**, abra o terminal (PowerShell ou CMD) e verifique se foi instalado corretamente digitando: `node -v` e `npm -v`.
- Para testar o app no celular físico, baixe o aplicativo **Expo Go** na Play Store ou App Store.

Como iniciar um projeto:

1. No terminal, navegue até a pasta onde deseja salvar o projeto.
2. Execute: `npx create-expo-app@latest MeuProjeto`
3. Entre na pasta: `cd MeuProjeto`
4. Inicie o servidor: `npx expo start`
5. Escaneie o **QR Code** que aparecerá no terminal usando o app Expo Go no seu celular.

React Native (Via Expo)

Exemplo Simples (App.js):

```
import { StyleSheet, Text, View } from 'react-native';
export default function App() {
  return (
    <View style={styles.container}>
      <Text style={styles.text}>Olá, Mundo com React
Native!</Text>
    </View>
  );
}
const styles = StyleSheet.create({
  container: { flex: 1, justifyContent: 'center', alignItems:
'center', backgroundColor: '#fff' },
  text: { fontSize: 20, fontWeight: 'bold', color: '#0081C9' },
});
```

Flutter

O Flutter é conhecido por sua alta performance e visual consistente entre plataformas.

O que baixar:

1. **Flutter SDK:** O pacote de ferramentas do Flutter (baixe o .zip no site oficial).
2. **Git para Windows.**
3. **Android Studio:** Necessário para instalar o SDK do Android e emuladores.

Como parametrizar (Windows):

1. Extração: Extraia o Flutter SDK em `C:\src\flutter` (evite a pasta Arquivos de Programas).

Flutter

Como parametrizar (Windows):

2. Variáveis de Ambiente:

- Pesquise no Windows por "Variáveis de Ambiente".
- Em "Variáveis de Usuário", edite a variável `Path`.
- Clique em "Novo" e adicione o caminho: `C:\src\flutter\bin`.

3. Validação:

- Abra o terminal e digite `flutter doctor`. Ele dirá o que falta configurar (como aceitar licenças do Android).

Como iniciar um projeto:

1. No terminal, digite: `flutter create meu_projeto`
2. Entre na pasta: `cd meu_projeto`
3. Para rodar (com um celular ou emulador conectado): `flutter run`

Flutter

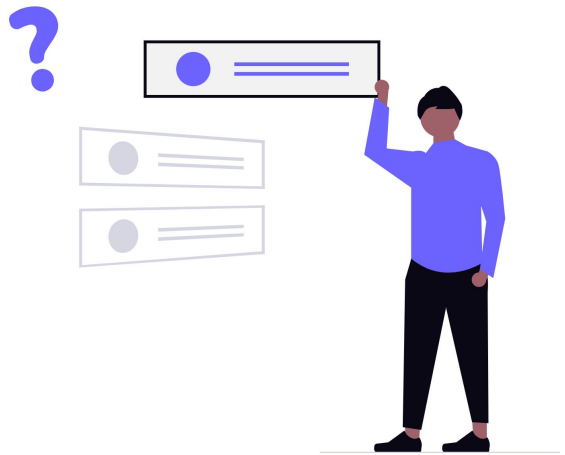
Exemplo Simples (main.dart):

```dart
import 'package:flutter/material.dart';

void main() => runApp(MaterialApp(home: Home()));

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: Text(
          'Olá, Mundo com Flutter!',
          style: TextStyle(fontSize: 20, color: Colors.blue),
        ),
      ),
    );
  }
}
```

React ou Flutter?

- **React Native:** Se você já conhece **Web (React/JS)**. É excelente para apps que dependem muito de interface e APIs.

- **Flutter:** Se você quer **performance máxima** e total controle sobre o design visual (UI) de forma nativa e rápida.



Tecnologias Desktop (Electron)

Aplicações Desktop com tecnologia Web

- **O que é:** Um framework que permite criar aplicações para Windows, Linux e Mac usando tecnologias Web (HTML, CSS, JS).

- **Funcionamento:** Ele empacota um navegador (Chromium) e um servidor (Node.js) dentro do aplicativo.

- **Exemplos:** VS Code, WhatsApp Desktop, Discord.

- **Reuso:** Permite que desenvolvedores Web criem software Desktop sem aprender C++ ou C#, aplicando o princípio de reutilização de conhecimento.

Qual Escolher?

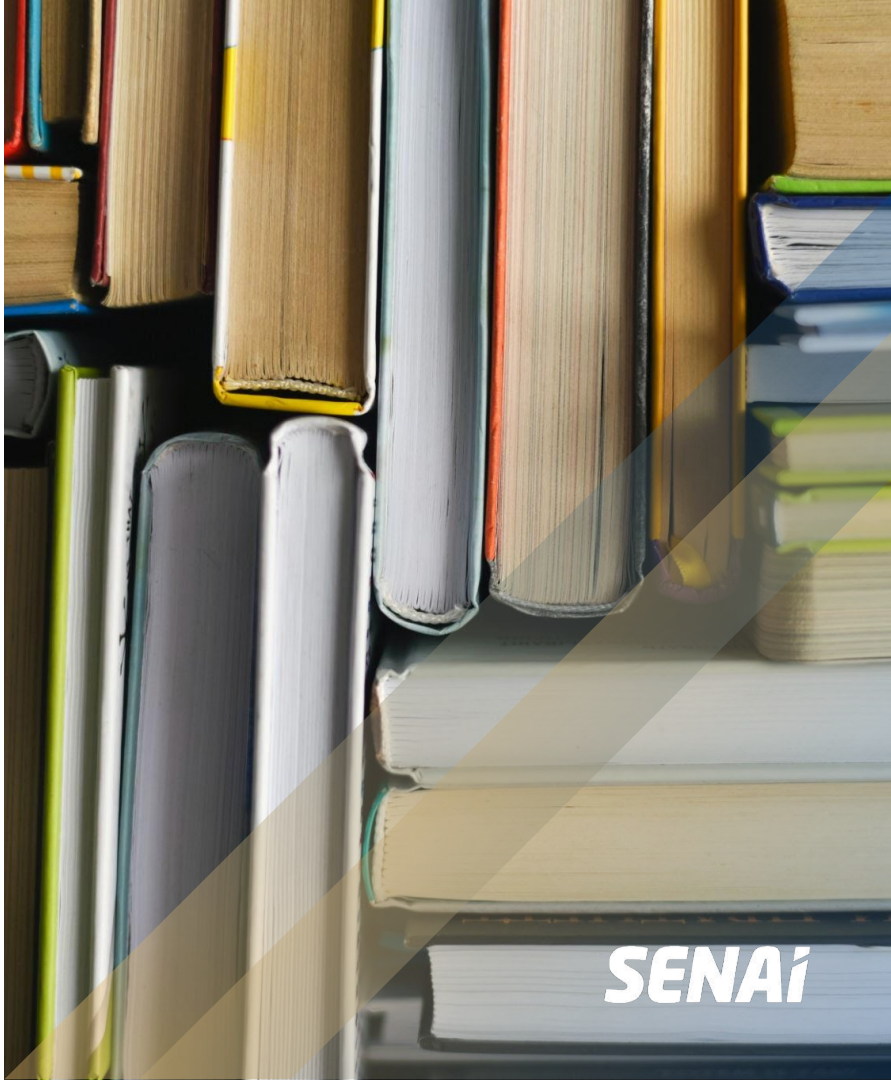
Não existe Bala de Prata

- **Analise o contexto:**
  - Precisa de desempenho extremo ou uso intensivo de hardware específico?
    Vá de Nativo.
  - Precisa validar uma ideia rápido (MVP) em Android e iOS?
    Vá de Cross-platform (Flutter/React Native).
  - Já tem uma equipe Web e quer ir para o Desktop?
    Vá de Electron.

- **Recapitulando:** Frameworks e padrões de projeto (como Bridge e Adapter) existem para resolver o problema da complexidade e da mudança constante das plataformas.

Fontes Utilizadas

- *Downey, A. B. "Pense em Python".*
- *Gamma, E. et al. "Padrões de Projeto: Soluções Reutilizáveis de Software Orientado a Objetos".*
- *Danjou, J. "Python Levado a Sério".*
- *Valente, M. T. "Engenharia de Software Moderna".*
- *Preece, J. et al. "Design de Interação".*




Obrigado!
SENAI