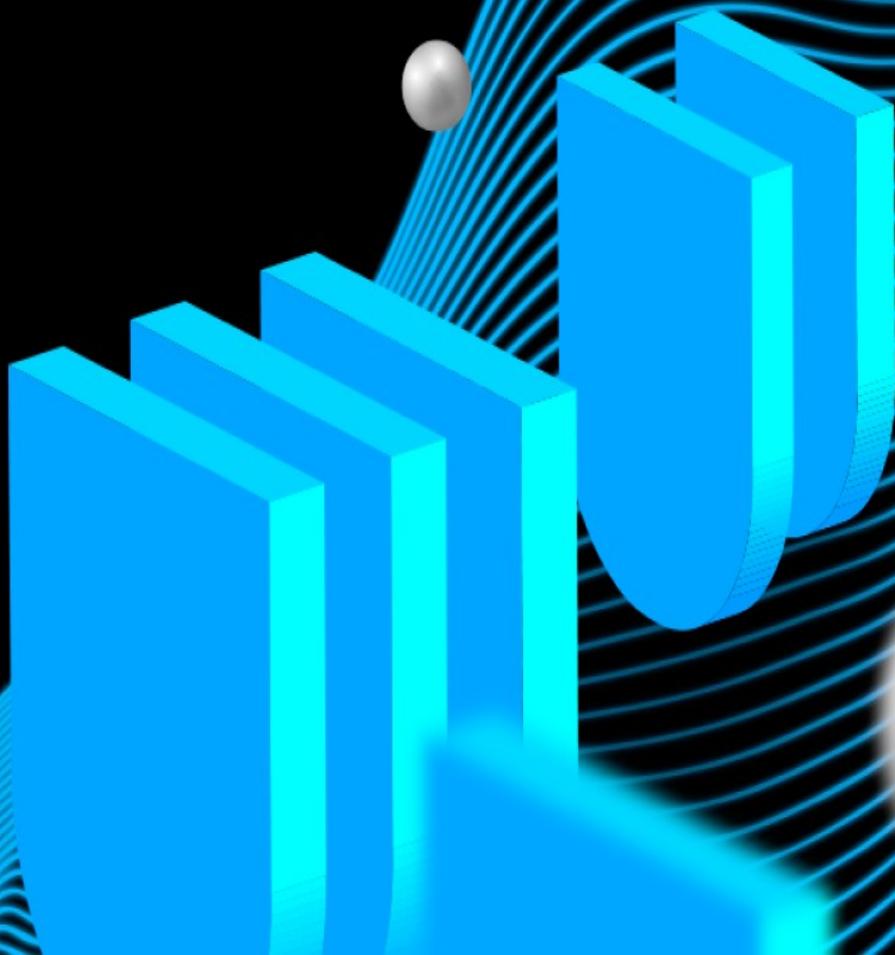




UNITAU

Universidade de Taubaté





Introdução



Introdução



“Desenvolvimento de aplicações e sistemas para dispositivos móveis, por vezes utilizado apenas como desenvolvimento mobile é toda atividade e processos acerca do desenvolvimento de software para dispositivos móveis (handheld) como computadores de bolso, PDAs, smartphone, telefone celular, console portátil e Ultra Mobile PC combinado com tecnologias como GPS, TV portátil, touch, consoles, navegador de Internet, WAP, leitores de áudio, vídeo e texto, entre outros.”

“Os principais sistemas operacionais do mercado atual são o Android, iOS, BlackBerry, HP webOS, Symbian OS, Bada da Samsung, e Windows Mobile”



Distribuição de Aplicativos

“Para adquirir uma aplicação mobile, geralmente as empresas o fazem a partir de web stores (lojas na internet) onde garantem a distribuição através de licenças ou através de compras.”

“Eles estão disponíveis através de plataformas de distribuição de aplicações que são normalmente operadas pelo proprietário do sistema operacional móvel, como App Store, Google Play, BlackBerry App World, entre outros.”

Aplicações Nativas



A divisão do mercado em diferentes sistemas operacionais com diferentes versões torna complexo o desenvolvimento de aplicações nativas, que são escritas na linguagem nativa do sistema operacional como **Objective-C/Swift** para **iOS** e **Java/Kotlin** para **Android**.

Aplicações nativas podem ser otimizadas para o sistema operacional ao qual se destinam. Podem ter melhor performance e utilizar melhor os recursos do sistema operacional.

Aplicações Nativas



Porém, desenvolver o mesmo aplicativo de forma nativa para diferentes sistemas operacionais, torna o desenvolvimento mais caro e mais complexo pois é necessário manter códigos diferentes para cada plataforma, geralmente com diferentes equipes de desenvolvimento.

Além disso, pode haver diferença no comportamento e aparência da aplicação de uma plataforma para outra.

Desenvolvimento Cross-Plataforma



Para diminuir os custos e tempo de desenvolvimento, surgiram ferramentas para desenvolvimento cross-plataforma para aplicações mobile.

Algumas ferramentas de desenvolvimento utilizam bibliotecas nativas para criar uma API unificada para os diferentes sistemas operacionais.

Essas ferramentas ainda precisam de código dependente da plataforma e as aplicações podem ter aparência diferente de uma plataforma para outra.

Desenvolvimento Cross-Plataforma



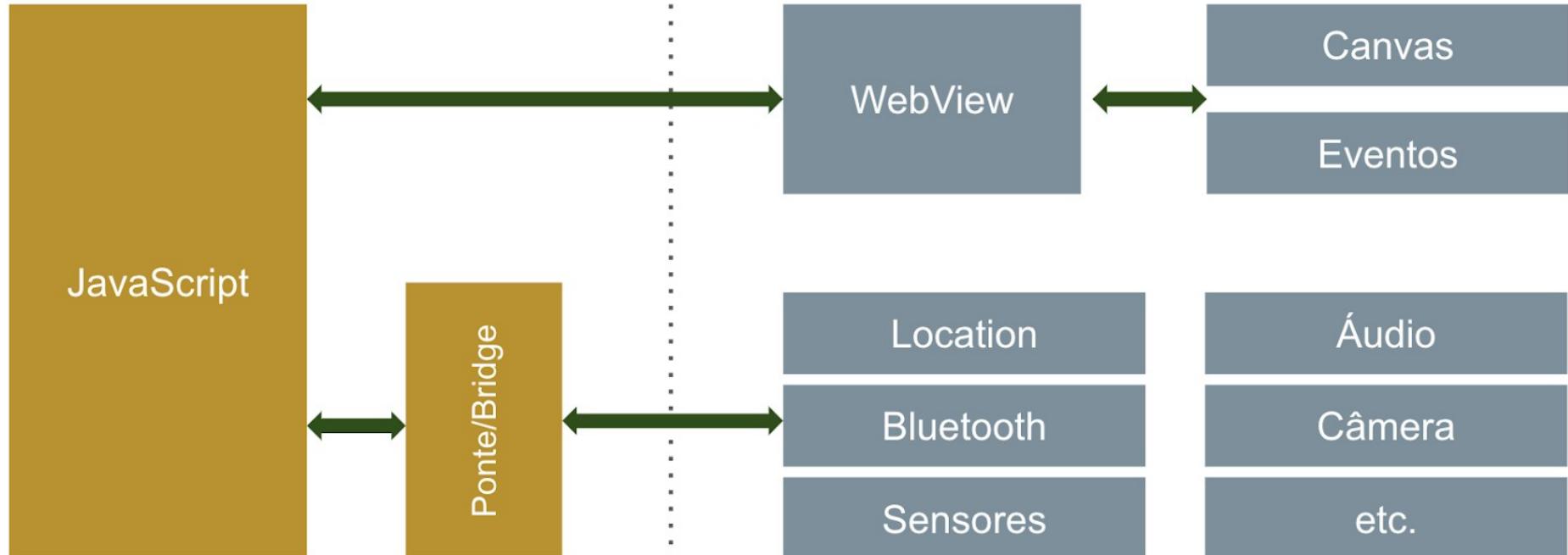
Outra opção são ferramentas que executam a aplicação em uma webview, gerando a aplicação usando tecnologias web como **HTML**, **CSS**, **JavaScript**. Como são aplicações executadas em um browser, tem uma aparência mais padronizada, mas podem comprometer a performance da aplicação.

Desenvolvimento Cross-Plataforma



Sua Aplicação

Plataformas



React Native



Uma dos mais modernos framework de desenvolvimento cross-plataforma é o **React Native** do **FaceBook**.

Utiliza JavaScript para desenvolver a aplicação com um bridge para o lado nativo do sistema operacional.

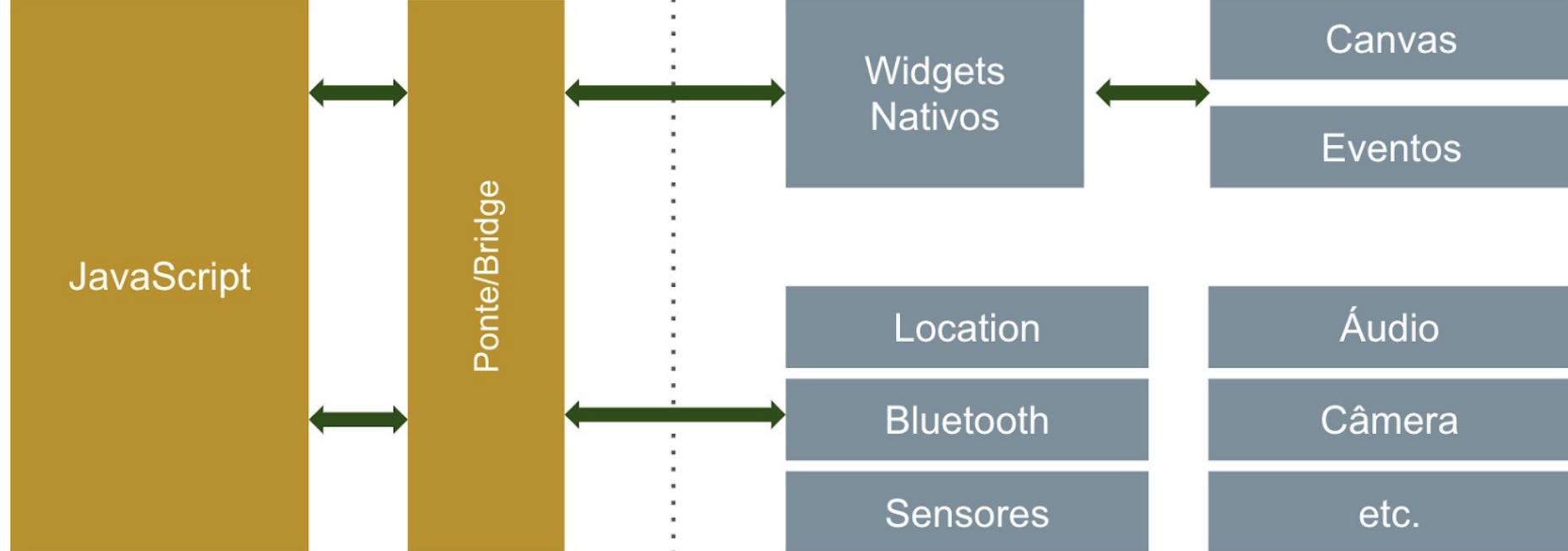
É um framework muito utilizada mas o bridge para os widgets nativos pode levar a problemas de performance.

React Native



Sua Aplicação

Plataformas



Flutter



O **Flutter** é um framework de desenvolvimento cross-plataforma do **Google** que utiliza a linguagem **Dart**.

Tem um motor de renderização próprio, de alta performance, e não utiliza os widgets nativos do sistema operacional.

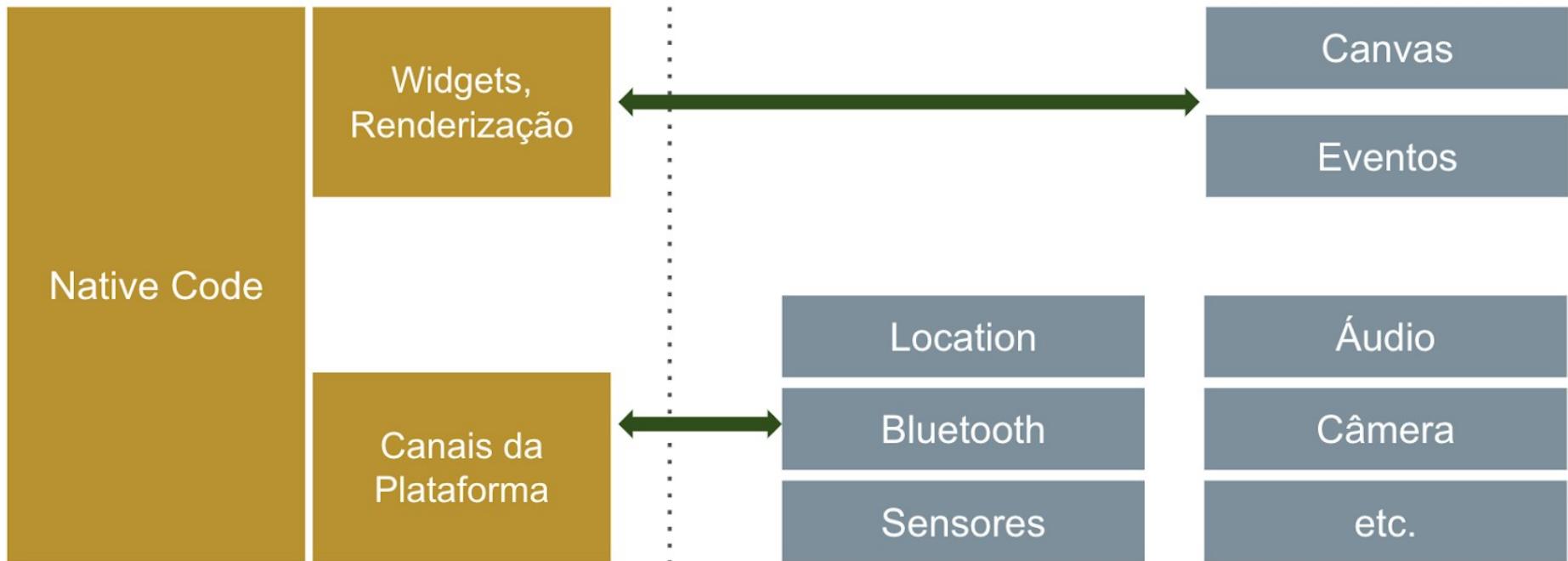
O código da aplicação é compilado “ahead of time” para código nativo para a plataforma, obtendo uma melhor performance para a aplicação.

Flutter



Sua Aplicação

Plataformas





Linguagens Compiladas

“Linguagem compilada é uma linguagem de programação em que o código fonte, nessa linguagem, é executado diretamente pelo sistema operacional ou pelo processador, após ser traduzido por meio de um processo chamado compilação, usando um programa de computador chamado compilador, para uma linguagem de baixo nível, como linguagem de montagem ou código de máquina.”



Linguagens Interpretadas

“Linguagem interpretada é uma linguagem de programação em que o código fonte nessa linguagem é executado por um programa de computador chamado interpretador, que em seguida é executado pelo sistema operacional ou processador. Mesmo que um código em uma linguagem passe pelo processo de compilação, a linguagem pode ser considerada interpretada se o programa resultante não for executado diretamente pelo sistema operacional ou processador. Um exemplo disso é o Bytecode, que é um tipo de linguagem interpretada que passa pelo processo de compilação e, em seguida, é executado por uma máquina virtual, cuja sintaxe é similar a código de máquina e cada comando ocupa 1 byte.”

Compilação Ahead Of Time (AOT)



“O ato de transformar o código-fonte em código de máquina é chamado de "compilação". Quando todo o código é transformado de uma só vez antes de atingir as plataformas que o executam, o processo é chamado de Compilação AOT (Ahead-Of-Time) . Ele difere do processo JIT (Just-In-Time).

Ou seja é um processo de compilação que ocorre antes da execução do aplicativo e não durante, como ocorre no processo JIT. O processo AOT compila uma linguagem de alto nível (C, C++, etc.) ou um bytecode (código intermediário. Ex: Java) em código de máquina nativo (dependente do sistema) para que o arquivo binário resultante, seja executado nativamente, ajudando a eliminar sobrecargas de compilação durante a execução.”

Compilação Just In Time (JIT)



“Na computação, a compilação just-in-time (JIT) (também tradução dinâmica ou compilações em tempo de execução) é uma forma de executar código de computador que envolve a compilação durante a execução de um programa - em tempo de execução - em vez de antes da execução. [...] Uma implementação comum de compilação JIT é primeiro ter compilação AOT para bytecode (código de máquina virtual), conhecido como compilação de bytecode, e então ter compilação JIT para código de máquina (compilação dinâmica), em vez de interpretação do bytecode. [...] Os compiladores JIT traduzem continuamente, como ocorre com os interpretadores, mas o armazenamento em cache do código compilado minimiza o atraso na execução futura do mesmo código durante uma determinada execução.”

Bibliografia



O material a seguir tem como referência:

- CLOW, M. **Learn Goggle Flutter Fast: 65 Example Apps.** [S.I.]: Independently Published, 2019. ISBN 9781092297370
- NAPOLI, M. L. **Beginning Flutter: A Hands On Guide to App Developement.** [S.I.]: John Wiley & Sons, 2019. ISBN 9781119550877