Desenvolvimento de tecnologias *cross-platform*: Ionic + Capacitor/Cordova

Simão Cunha[a93262] and Gonçalo Pereira[a93168]

Universidade do Minho - Campus de Gualtar, R. da Universidade, 4710-057 Braga Portugal

Tópicos de Desenvolvimento de Software (2022/2023)

Resumo O presente relatório refere-se ao ensaio escrito no âmbito da UC de Tópicos de Desenvolvimento de *Software*, onde começaremos com uma introdução acerca do desenvolvimento de *software* em *cross-platform*, com especial foco em Ionic. Iremos descrever esta *framework*, juntamente com Capacitor e Cordova. De seguida, iremos falar sobre a sua arquitetura, vantagens e desvantagens, *use-cases* e aplicabilidade. Terminaremos com uma secção de sistemas conhecidos construídos com a *framework* retratada ao longo deste artigo.

Keywords: Cross-Platform, Ionic, Capacitor, Cordova, Web, PWA, Android, iOS

1 Introdução

Quando se pretende desenvolver uma aplicação é importante responder a duas perguntas: Quem é o utilizador final? Como é que se chega ao maior número de utilizadores?. Claro que o ideal seria desenvolver a mesma aplicação para todas as plataformas chegando assim a todos os potenciais utilizadores. No entanto, desenvolver a mesma aplicação nativamente para duas ou três plataformas diferentes tem os seus custos, custos esses que nem todas as empresas conseguem suportar. Nestes casos, surge a possibilidade de desenvolver uma aplicação *cross-platform*, ou seja, após um único ciclo de desenvolvimento ter uma aplicação que funcione em plataformas diferentes - desta forma, por um custo menor, consegue-se atingir uma maior audiência. Este facto fez com o desenvolvimento multi-plataforma tenha ganho bastante popularidade ao longo dos anos.

Neste ensaio, iremos abordar diferentes aspetos das tecnologias *cross-platform* aplicadas ao desenvolvimento em dispositivos móveis onde iremos abordar em detalhe a *framework Ionic*.

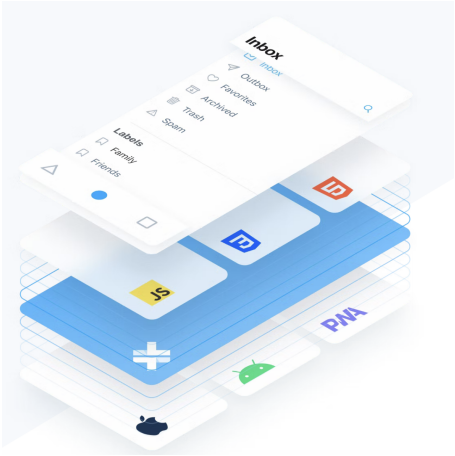

Figura 1: Aplicações híbridas - retirado de [1]

2 Ionic + Capacitor/Cordova

Ionic é uma *framework* de desenvolvimento de aplicações móveis híbridas, que combina tecnologias *web* (HTML, CSS e JavaScript) para construir aplicações que podem ser executados em várias plataformas, incluindo Android e iOS. Aplicações híbridas são essencialmente aplicações *web* mas que são instaladas como aplicações nativas sendo, no entanto, *renderizadas* pelo *engine* do *browser*. Ionic permite integrar ainda *frameworks* de *frontend*, suportando `Angular`, `React` ou `Vue`. Existem duas ferramentas que podem ser usadas na *Ionic*: Cordova e Capacitor.

Cordova é uma *framework* híbrida *open source* que permite aos *web developers* usar HTML, CSS, e JavaScript para criar uma aplicação nativa para uma variedade de plataformas *mobile*. Cordova renderiza a aplicação *web* numa nativa *WebView*, que não é mais do que um componente da aplicação (como um botão) que é usado para mostrar conteúdo *web* na aplicação nativa. Podemos pensar em *WebView* como um *web browser* sem os elementos de interface do utilizador, como por exemplo a barra de pesquisas ou a *nav bar*. A aplicação *web* a executar aqui dentro é como uma aplicação que poderia ser aberta num *browser mobile* - pode abrir páginas HTML adicionais, executar código JavaScript, executar ficheiros multimédia e comunicar com servidores remotos.

Além de ser uma *framework* usada para executar uma aplicação *web* numa aplicação nativa, Cordova também fornece API's em JavaScript que permite o acesso a várias funcionalidades do dispositivo, como o acesso aos contactos, através de *plugins*, que fazem a ponte entre a aplicação *web* em si com as diversas funcionalidades nativas do dispositivo. [2]

Capacitor é uma *framework open source* que executa aplicações Web de forma nativa em Android, iOS, etc, recorrendo à tecnologia de *Progressive Web Apps* e permitem aceder a SDK's e API's nativas em cada plataforma. É uma alternativa a Cordova, mas, para além de ter os mesmos benefícios em relação a *cross-platform*, também permite uma abordagem mais moderna ao desenvolvimento de aplicações tomando partido de Web API's e funcionalidades nativas mais recentes.

Através do uso de Capacitor, os desenvolvedores podem construir uma só aplicação usando um conjunto de API's independentemente da plataforma onde essa *app* está a executar, em vez de usarem tantas API's quanto o número de plataformas a considerar para a execução da aplicação. Por exemplo, o acesso à câmara do dispositivo irá ser feito através do mesmo código, independentemente do dispositivo correr em Android, iOS ou simplesmente na Web.

Eis algumas vantagens do Capacitor [3]:

- Suporte a Progressive Web App (PWA);
- Ferramenta CLI simples que é gerida por cada aplicação, ou seja, podemos ter várias versões de Capacitor em diferentes aplicações, pelo que não existe a necessidade de gerir dependências globais;
- Equipas de desenvolvimento nativo e de desenvolvimento *web* podem trabalhar de forma conjunta;
- Adoção de Capacitor forte e crescente;
- Capacitor é de fácil manutenção, contrariamente ao Cordova;
- Migração para Capacitor é facil - é compatível com Cordova.

Para desenvolver aplicações com Ionic, é necessário ter conhecimentos sólidos de tecnologias *web* como HTML, CSS e JavaScript. É recomendado conhecimento de alguma *framework* de *frontend*, `Angular`, `React` ou `Vue`, para facilitar o processo de desenvolvimento. Além disso, é recomendável ter conhecimentos em desenvolvimento móvel e familiaridade com o ambiente de desenvolvimento para as plataformas Android e iOS.

3 Arquitetura

Como já referido, normalmente usa-se Ionic em conjunto com ferramentas como Capacitor de forma a dar acesso às aplicações a funcionalidades nativas dos dispositivos. Capacitor encapsula funcionalidades nativas em código JavaScript pronto a ser usado pelas aplicações *web*.

Na figura abaixo é possível observar uma arquitetura de uma aplicação Ionic+Capacitor:

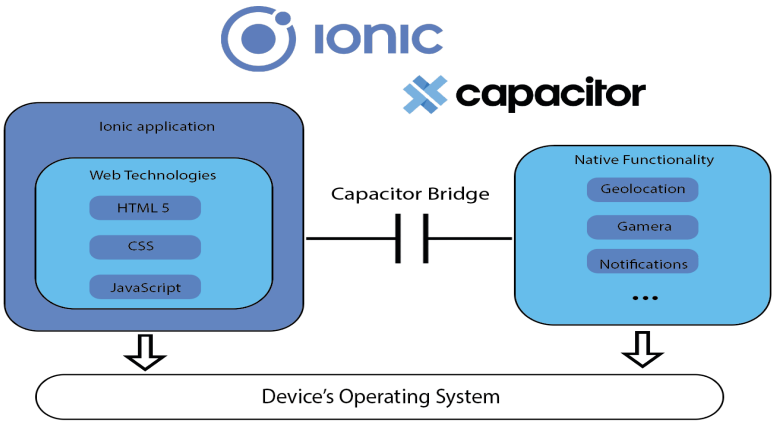

Figura 2: Arquitetura Ionic - adaptado de [4]

Analisando com mais detalhe a *Capacitor Bridge*, podemos ver na figura 3 o uso de *plugins*:

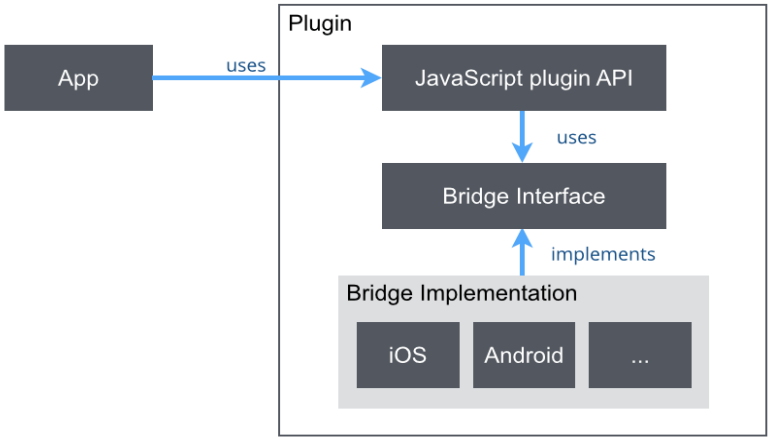

Figura 3: Exemplificação de um *plugin* - adaptado de [5]

4 Vantagens e Desvantagens

Ionic, como *framework*, segue a filosofia "code once, run everywhere", ou seja, após um ciclo de desenvolvimento, é possível ter uma aplicação a correr em qualquer plataforma, sendo uma solução de baixo custo e rápido em *deployment*.

Além disso, apresenta facilidades no que toca ao desenvolvimento, já que faz uso de tecnologias *web*, que fazem parte do conjunto de competências de qualquer desenvolvedor nos dias de hoje. Também tem a tarefa de testes facilitada, porque todos eles passam por um simples *browser* em vez de usar emuladores ou dispositivos físicos, pelo que as mudanças ao nível do UI são rápidas de se efetuarem.

Ionic permite, também, integrar ao nível do *frontend* as *frameworks* mais usadas como `Angular`, `React` ou `Vue`.

No que toca ao resultado final, é possível a criação e aplicações completamente interativas e, caso estejam bem construídas, são indistinguíveis de aplicações nativas.

No entanto, pelo facto da renderização ser feitas por *webviews* e por ter acesso a funcionalidades nativas apenas usando *native bridges*, Ionic apresenta um pior desempenho quando comparado com aplicações nativas. Além disso, *debugging*, otimização e segurança são áreas em que o desenvolvimento híbrido fica atrás do nativo. Outro aspeto negativo, é na incapacidade de desenvolver jogos, uma vez que não é possível desenvolver jogos 3D com Ionic já que usa apenas CSS para fazer animações gráficas.

5 *Use cases*

Nem sempre é óbvio por qual *framework* ou tecnologia devemos optar quando queremos desenvolver uma aplicação. Consideramos importante que seja feito um esforço para adequar as tecnologias aos requisitos da aplicação e às limitações impostas. [6]

Identificamos alguns *use cases* que para quem pretende usar Ionic:

(i) *Deploy* a uma aplicação para correr em iOS, Android, ter um *website* e até uma aplicação *desktop*;
(ii) Não ter tempo ou recursos para construir e para fazer manutenção a vários *codebases*;
(iii) Ter acesso a uma equipa de desenvolvimento de *software* cujo conhecimento é maioritariamente à base de *web development*;
(iv) Não ter acesso imediato a várias equipas capazes de programar em linguagens nativas e para *mobile* como Java, Kotlin, Swift, ...
(v) A aplicação não é muito dependente do *hardware* do dispositivo onde vai ser executado;

Por outro lado, a escolha de Ionic pode não ser benéfica em várias situações, tais como:

(i) Desenvolver apenas aplicações nativas. Nesta situação, poderá ser vantajoso utilizar outras tecnologias, como Flutter ou React Native;
(ii) Desenvolver uma aplicação complexa que requeira imensos recursos do *hardware* do dispositivo;

6 Aplicabilidade

Neste ensaio, já abordamos as limitações que Ionic pode ter mas também onde pode ser útil. Consideramos que esta *framework* pode ser usada no desenvolvimento de aplicações de diferentes géneros como redes sociais, entretenimento, utilitárias, produtividade, e-commerce, etc. No entanto, em jogos e plataformas de gestão - no geral em plataformas sensíveis - tanto no desempenho como na segurança entendemos que se deve optar por um desenvolvimento nativo, pois oferece melhores garantias.

Sistemas que beneficiam de Ionic	Redes sociais
	Entretenimento
	Utilitárias
	Produtividade
	E-Commerce
Sistemas que não beneficiam de Ionic	Jogos
	Plataformas de gestão
	Plataformas de trading

Tabela 1: Síntese do género de aplicações que beneficiam ou não com o uso de Ionic

7 Sistemas conhecidos construídos com Ionic + Capacitor/Cordova

Ionic foi usado para criar diversos sistemas *mobile*. Em [7], são descritos alguns deles:

7.1 `:86 400`

`:86 400`: é um banco australiano que usa Ionic como a plataforma para desenvolver *apps mobile* para Android e iOS. É possível lançar atualizações a cada 2-3 semanas por causa de Ionic assim como implementar correção de *bugs* críticos sem ter que esperar por aprovações da *app store*.

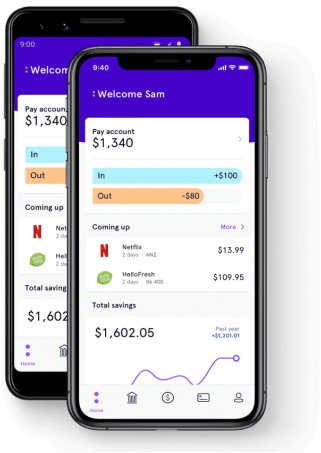

Figura 4: Aplicação *mobile* `:86 400` (retirado de [7])

7.2 `Shipt`

`Shipt` é um serviço de entrega de mercearia norte-americano cujos pedidos na *app mobile* escalaram muito rapidamente graças a Ionic e com uma abordagem *DevOps*. A aplicação foi construída com Ionic e serve membros em mais de 260 cidades com mais de 500 000 compradores. Graças ao *DevOps*, mantêm a aplicação atualizada com as novas *features* logo que estejam disponíveis para serem lançadas.

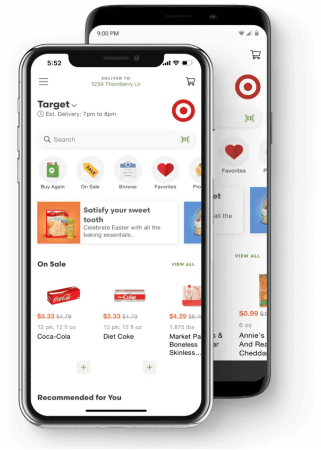

Figura 5: Aplicação *mobile* `Shipt` (retirado de [7])

7.3 `Untappd`

`Untappd` é uma rede-social norte americana para entusiastas em cerveja que foi construída com Ionic. Esta aplicação permite aos seus utilizadores encontrar, classificar e partilhar o que estão a beber.

Traduziram a sua UI original em Ionic, usando componentes como Toggle e Checkmarks para dar uma sensação nativa. Graças a Ionic, foram capazes em entrar no mercado de forma mais rápida.

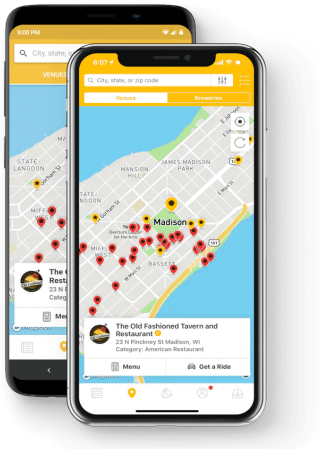

Figura 6: Aplicação *mobile* `Untappd` (retirado de [7])

7.4 `Burger King`

`Burger King` é uma cadeia de restaurantes *fast-food* norte americana. A sua aplicação foi construída com Ionic e Capacitor de forma a manter o *design* UI de forma consistente em todas as plataformas tal como por exemplo na visualização do menu que deve ser igual na aplicação *mobile*, na *web* ou mesmo dentro do restaurante. [8]

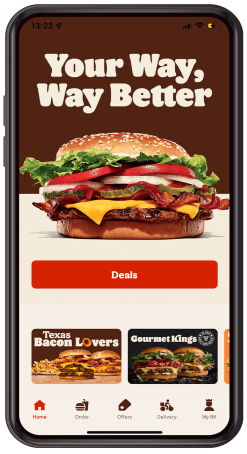

Figura 7: Aplicação *mobile* `Burger King`

7.5 Outros

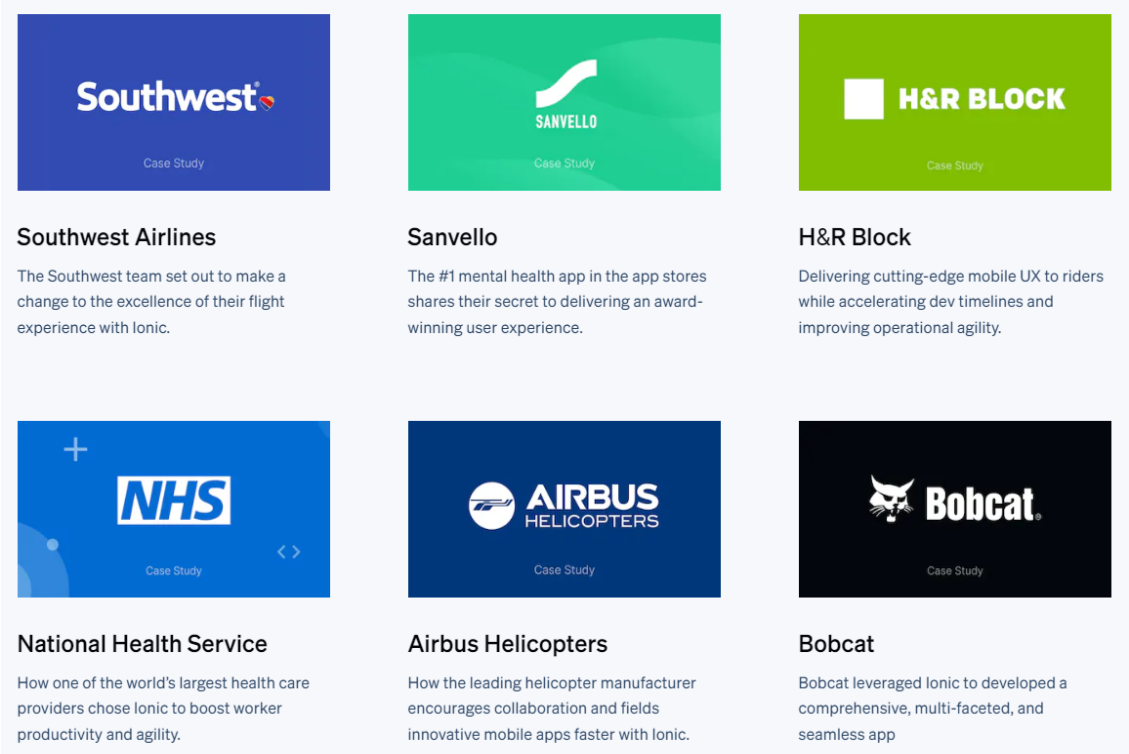

Figura 8: Outras entidades que construíram as suas aplicações com Ionic (retirado de [9])

Referências

1. Capacitor website.: https://capacitorjs.com/ (Consultado em mar. 2023)
2. Griffith C.: https://ionic.io/resources/articles/what-is-apache-cordova (Consultado em mar. 2023)
3. Lynch M.: https://ionic.io/resources/articles/capacitor-vs-cordova-modern-hybrid-app-development (Consultado em mar. 2023)
4. "The good and the bad of Ionic mobile development".: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-ionic-mobile-development/ (Consultado em mar. 2023)
5. Ripkens B.: https://blog.codecentric.de/ionic-angularjs-framework-on-the-rise (Consultado em mar. 2023)
6. Adam J.: https://kruschecompany.com/ionic-framework-hybrid-apps/ (Consultado em mar. 2023)
7. Romaniv T.: https://softjourn.com/insights/ionic-app-development-advantages-and-disadvantages (Consultado em mar. 2023)
8. Burger King study case: https://ionic.io/resources/articles/burger-king-design-system (Consultado em mar. 2023)
9. Outros casos de estudo: https://ionic.io/customers (Consultado em mar. 2023)