Comparação avaliativa entre os frameworks Ionic e Flutter

Lucas Kurtz Moreira* Anubis Rossetto †

2020

Resumo

Este artigo apresenta uma avaliação comparativa entre os frameworks Ionic e Flutter para o desenvolvimento utilizando a abordagem Progressive Web App (PWA). Dessa forma, foi desenvolvida uma aplicação de gerenciamento de tarefas em ambos os frameworks, incorporando os serviços da plataforma Google Firebase para armazenamento e hospedagem das aplicações. Após o desenvolvimento da aplicação foi realizada uma avaliação utilizando a ferramenta Lighthouse que fornece uma auditoria quanto a vários aspectos, tais como, acessibilidade, desempenho, otimização de mecanismo de busca (SEO), melhores práticas e PWAs.

Palavras-chave: PWA. Ionic. Flutter. Lighthouse.

Introdução

Recentemente há um avanço importante no desenvolvimento de aplicações web para dispositivos móveis que permite o emprego de recursos encontrados anteriormente apenas em aplicativos nativos. Uma das abordagens de desenvolvimento que tem permitido esse avanço é denominada Progressive Web Apps (PWA), ou seja, uma evolução híbrida entre aplicações web (sites) e aplicações móveis. Assim, combinam recursos oferecidos pelos mais modernos navegadores, com vantagens de uso em smartphones. PWAs permitem fornecer aplicações independentes de plataforma uma

*<lucaskurtz96@gmail.com>
†<anubisrossetto@ifsul.edu.br>

vez que rodam no browser, além de prover atualização em background, suporte off-line, atalho em tela inicial do dispositivo e ter uma aparência de um aplicativo (app) instalável.

Há também uma tendência de crescimento nesse tipo de aplicação uma vez que se tem observado que os usuários de dispositivos móveis baixam poucas aplicações e dessas o uso fica restrito na grande parte em poucas aplicações. Algumas pesquisas indicam que em médias os usuários interagem de forma efetiva com apenas 5 aplicativos principais instalados em seus dispositivos, enquanto os restantes costumam ser removidos ou muito pouco utilizados. Também, as redes limitadas e de qualidade baixa, desestimulam os usuários a fazer downloads de aplicativos. Ainda há os usuários que evitam acessar a loja de aplicativos para fazer buscas, downloads e testes.

Existem vários frameworks disponíveis para desenvolvimento tanto de aplicações móveis quanto web apps. Dois desses frameworks que tem se destacado são: Ionic e Flutter. Esse projeto busca investigar aspectos de performance de PWA geradas a partir desses dois frameworks.

O framework para o desenvolvimento de um aplicativo móvel pode ter um impacto no esforço de desenvolvimento necessário para criar a aplicação e na qualidade e desempenho percebidos da aplicação gerada. O objetivo ao escolher um framework é maximizar a reutilização da base de código do projeto e minimizar o esforço total necessário para suportar múltiplas plataformas enquanto ainda produz um produto final responsivo, atraente, fácil de usar e consistente entre plataformas (GONSALVES, 2019).

Assim, tem-se a seguinte questão de pesquisa: Qual o impacto da escolha de um framework de desenvolvimento no desempenho e na qualidade percebida das PWAs geradas? Dessa forma, o objetivo é comparar dois frameworks para desenvolvimento de PWA a partir da implementação de uma aplicação, verificando o impacto da escolha de uma estrutura de desenvolvimento no desempenho e qualidade da aplicação.

PWA é um produto da evolução conjunta dos navegadores móveis e da funcionalidade dos aplicativos nativos, como notificações por push, navegação por GPS e outras funções nativas que foram mescladas. O desenvolvimento de um aplicativo PWA pode ser várias vezes mais barato e mais rápido que as versões nativas, o que pode ser benéfico para pequenas empresas. Isso também pode ser benéfico para o usuário; pois não ocupa espaço de hardware do dispositivo na memória do telefone e não é necessário fazer o download do aplicativo nas lojas de aplicativos (KUBRAK, 2017).

PWAs também possuem a possibilidade de ter operação offline, utilizando Service Worker que vai gerenciar a comunicação entre a aplicação e o servidor, e permite salvar dados importantes em cache para a execução da aplicação.

Os dois frameworks escolhidos permitem gerar a aplicação como um PWA. O Ionic foi escolhido por ser um dos frameworks mais maduros disponíveis, possui documentação bem ampla e é muito difundido como ferramenta para desenvolvimento móvel. Já o Flutter é um framework novo da Google, muito jovem e ainda não foi

amplamente incluído na pesquisa de desenvolvimento de aplicativos para dispositivos móveis, mas tem atraído muito interesse dos desenvolvedores e tem perspectivas de crescimento

Para os testes das aplicações foi utilizada a ferramenta de testes LightHouse, ele executa uma série de testes na página e gera um relatório sobre o seu desempenho. Utiliza como principais critérios de avaliação o desempenho, PWA, acessibilidade, melhores práticas e SEO.

O artigo está organizado da seguinte forma: na Seção 1 são descritas as tecnologias para o desenvolvimento de um Progressive Web App, juntamente com suas características ; na Seção 2 é detalhada metodologia para avaliação comparativa dos frameworks, envolvendo também aspectos do desenvolvimento da aplicação e a descrição das funcionalidades; na seção 3 é apresentada a avaliação dos frameworks a partir da ferramenta Lighthouse e após estão as considerações finais, juntamente com o planejamento das melhorias futuras.

1 TECNOLOGIAS PARA A CRIAÇÃO DE PWAs

Nesta seção são discutidos os conceitos envolvidos no desenvolvimento usando a abordagem PWA, juntamente com as tecnologias envolvidas para o desenvolvimento da aplicação, além do mecanismo que viabiliza a execução de uma aplicação Web offline .

1.1 Progressive Web App

Progressive Web App (PWA) é considerada uma nova metodologia de desenvolvimento móvel, ou ainda um novo "tipo" de aplicativo que combina os melhores recursos oferecidos pela maioria dos navegadores com os benefícios de experiência móvel. De acordo com Russel (2015) as características de uma aplicação PWA são:

- *Responsive* (Responsiva): Se adapta a qualquer tipo de tela;
- *Connectivity independent* (Independente de conectividade): Permite utilizar offline, com dados armazenados em cache pelo Service Workers;
- *App-like-interactions* (Igual a um aplicativo nativo): Utilização e interação igual a de um aplicativo nativo, porque é compilado em Shell;
- *Fresh*: Atualizada pelo processo de atualização do Service Workers;
- *Safe*(Segura): Obrigatório o uso do protocolo HTTPS e é vinculado ao TLS (requisito do service workers) para impedir espionagem;
- *Discoverable* (Detectável): São identificados como "aplicativos", graças ao escopo de registro do W3C *manifests* e service worker, permitindo que os mecanismos de pesquisa os encontrem;
- *Re-engageable* (Reengajável): Pode acessar as UIs de reengajamento do sistema operacional. Ex: Notificações via push;

- *Installable* (Instalável): Na tela inicial por meio de prompts fornecidos pelo navegador, permitindo que os usuários mantenham os aplicativos que consideram mais úteis, sem o incômodo de uma loja de aplicativos;
- *Linkable* (Vinculável): Significa que eles tem atrito zero, instalação zero e fácil de compartilhar através de URL.

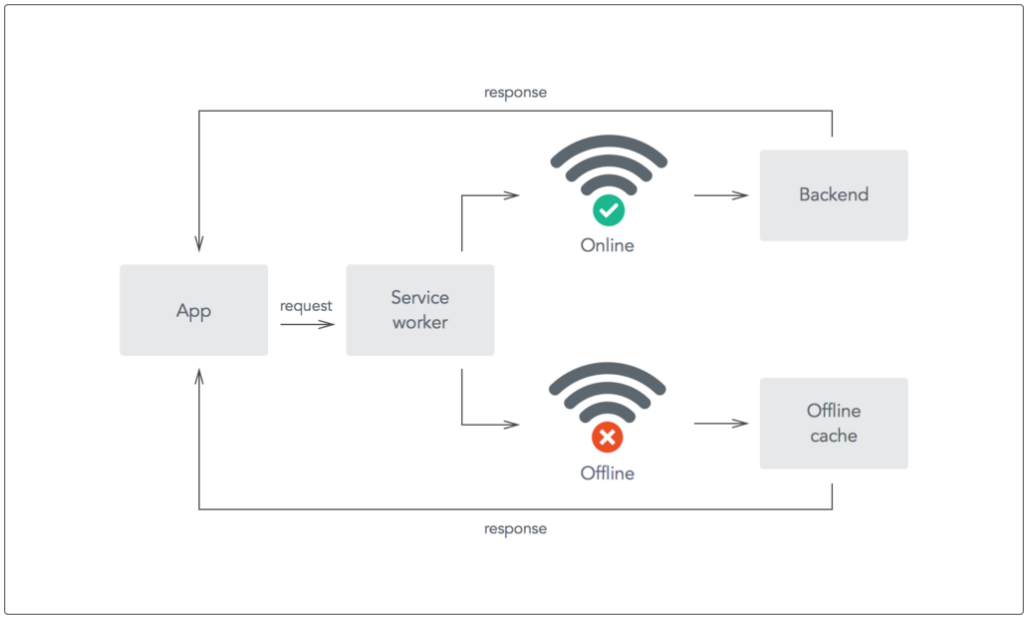
1.2 Service Worker

O service worker (SW) é um script que o navegador executa em segundo plano, separado da página web, abrindo a porta para recursos que não precisam de uma página da web ou de interação com usuário(DEVELOPERS, 2019b). Atualmente, os SWs já incluem recursos como notificações push e sincronização em segundo plano. A tendência é que no futuro eles poderão dar suporte a outras coisas, como sincronização periódica ou geofencing. A razão pela qual essa API é interessante é o fato que ela permite oferecer suporte a experiências offline, fornecendo aos desenvolvedores controle total sobre a experiência. Segundo Developers (2019b) as características importantes dos service workers são:

- Ele é um JavaScript Worker, portanto, não consegue acessar o DOM diretamente. Em vez disso, um service worker pode se comunicar com as páginas que controla respondendo a mensagens enviadas pela interface postMessage. Essas páginas podem manipular o DOM, se necessário;
- O service worker é um proxy de rede programável, o que permite controlar como as solicitações de rede da página são gerenciadas;
- Ele é encerrado quando ocioso e reiniciado quando necessário novamente. Isso significa que não se pode confiar no estado global dentro dos gerenciadores *onfetch* e *onmessage* de um service worker. Para informações que devem ser mantidas e reutilizadas entre reinícios, os service workers podem acessar a IndexedDB API.

A Figura 1 mostra o ciclo de funcionamento do Service Workers, onde o aplicativo faz uma solicitação ao Service Worker, sendo que quando tem conexão, busca no *backend*, e quando estiver offline, busca no cache.

Figura 1 – Funcionamento do Service Worker



Fonte: (DONALDSON, 2017)

1.3 Ionic Framework

O Ionic Framework é um kit de ferramentas de interface do usuário de código aberto para a criação de aplicativos móveis e de desktop de alta qualidade e desempenho, usando tecnologias da web (HTML, CSS e JavaScript)(FRAMEWORK, 2019).

Como o Ionic é focado no desenvolvimento de aplicativos da web, desenvolvidos usando HTML5, CSS e Java Script, isso permite que sejam executados nos navegadores da Web usando um único código base. Porém para executar em dispositivos móveis, não suporta muitas funções, incluindo câmera, geolocalização e suporte offline. Dessa forma, foram introduzidas as aplicações móveis híbridas, que são essencialmente aplicativos incorporados em um container nativo, permitindo que as páginas HTML5 sejam exibidas no dispositivo móvel com o mesmo comportamento dos aplicativos nativos. Uma vez que o framework Ionic trabalha com essa abordagem de desenvolvimento, também permite gerar um PWA a partir da mesma codificação(BLACKBERRY, 2017). Já para acessar alguns recursos do dispositivo móvel é possível utilizar o Capacitor em conjunto com o Ionic.

A estrutura Ionic baseia-se na biblioteca Angular e concentra-se na criação rápida de aplicativos da Web por meio de uma grande variedade de componentes predefinidos para o design e interações da interface do usuário. Assim, concentra-se na experiência do usuário *front-end* ou na interação do usuário com a interface dos aplicativos (controles, interações, gestos, animações). Tem aprendizagem fácil e se integra bem com outras bibliotecas ou estruturas, como Angular, e também pode ser usado de forma independente sem *front-end* usando *script*.

1.4 Flutter Framework

O Flutter é um SDK *open source* para desenvolvimento de aplicativos móveis criado pelo Google, cujo objetivo é desenvolver aplicativos para as plataformas Android e iOS.

O Flutter é uma plataforma nova e promissora que atraiu a atenção de grandes empresas que já lançaram seus aplicativos utilizando a plataforma. Muito da expectativa sobre o Flutter se deve a sua simplicidade em comparação ao desenvolvimento de aplicações Web e também por causa de sua performance em comparação com aplicativos nativos.

De acordo com Medium (2019), a alta performance e produtividade no Flutter são obtidos utilizando várias técnicas conforme relacionadas abaixo:

- O Flutter não utiliza JavaScript. Dart é a linguagem de programação. Ele compila para código binário, e é por isso que ele é executado com o desempenho nativo de Objective-C, Swift, Java ou Kotlin;
- Flutter não utiliza componentes de UI nativos. Isso pode parecer estranho no começo, no entanto, como os componentes são implementados no próprio Flutter, não há camada de comunicação entre a *view* e seu código. Devido a isso, as aplicações atingem a melhor velocidade. Então, botões, texto, elementos de mídia, *backgrounds* são todos desenhados pelo mecanismo de gráficos do Flutter. Além disso, o pacote do aplicativo Flutter "Olá, Mundo" é bem pequeno: iOS = 2.5Mb e Android = 4Mb;
- O Flutter utiliza uma abordagem declarativa, inspirada no framework web React, para construir sua UI baseada em *widgets* (chamados "componentes" no mundo web). Para obter mais de *widgets*, eles são renderizados apenas quando necessário, geralmente quando seu estado foi alterado (assim como o Virtual DOM faz para WEB);
- O framework integrou o Hot-reload, típico da web, mas ainda ausente em plataformas nativas. Isso permite que a estrutura Flutter reconstrua automaticamente a árvore (*tree*) de *widgets*, permitindo que o desenvolvedor visualize rapidamente os efeitos de suas alterações.

1.5 Firebase

Firebase é uma plataforma de desenvolvimento mobile e web com foco em ser um back-end completo e de fácil usabilidade. Ela oferece funcionalidades como análises, bancos de dados, mensagens e relatórios de falhas para que o desenvolvedor possa se concentrar nos usuários e sua aplicação, sem que precise gerenciar a infraestrutura Firebase (2019b apud FERNANDES, 2019, p. 22). A plataforma Firebase possuir diversos serviços, entre eles alguns que serão usados na implementação desse projeto são abordados a seguir.

- Cloud Firestore

O Cloud Firestore é um banco de dados flexível e escalonável para desenvolvimento de dispositivos móveis, Web e servidores a partir do Firebase e do Google Cloud Platform. Como o Firebase Realtime Database, ele mantém seus dados em sincronia em aplicativos cliente por meio de listeners em tempo real. Além disso, oferece suporte off-line para dispositivos móveis e Web para que você possa criar aplicativos responsivos que funcionem independentemente da latência da rede ou da conectividade com a Internet. O Cloud Firestore também oferece integração perfeita com outros produtos do Firebase e do Google Cloud Platform, incluindo o Cloud Functions (FIREBASE, 2019a).

- Firebase Hosting

  O Firebase Hosting é um recurso de hospedagem de conteúdo da Web de nível de produção para desenvolvedores. Com um único comando, é possível implantar apps da Web rapidamente e exibir conteúdo estático e dinâmico a uma rede de distribuição de conteúdo (CDN) global(FIREBASE, 2019c).

2 METODOLOGIA

Para alcançar o objetivo proposto nesse trabalho, foram desenvolvidas a mesma aplicação em ambas as ferramentas. A aplicação proposta está relacionada ao gerenciamento de tarefas do dia-a-dia do usuário. Assim, a ideia da aplicação é que após o usuário estar logado, ele possa gerenciar suas tarefas: visualizar a listagem, incluir novas tarefas, excluir tarefas, ou ainda marcar como realizada. Embora a aplicação proposta seja simples, o objetivo principal é avaliar o PWA gerado pelos frameworks sem muita interferência do programador, usando as configurações padrão das ferramentas. Para o armazenamento dos dados foi utilizado o Cloud Firestore da plataforma Firebase. Tendo uma aplicação PWA e usando os motores de sincronização do Firestore é possível propiciar a operação offline da aplicação. Além disso, as duas aplicações geradas foram hospedadas no Hosting do Firebase.

2.1 Requisitos funcionais

Os seguintes requisitos funcionais foram definidos para a aplicação:

- Cadastrar usuário: O usuário pode se cadastrar através de email e senha ou cadastrar por meio de sua conta Facebook ou Google;

- Manter tarefas: permitir consultar, incluir, alterar e excluir tarefas com possibilidade de marcar a localização (latitude e longitude);

- Login: Logar na aplicação.

2.2 Requisitos não funcionais

A seguir estão listados os requisitos nao funcionais definidos para aplicação:

- Utilizar o Firebase Firestore para armazenar os dados das tarefas;

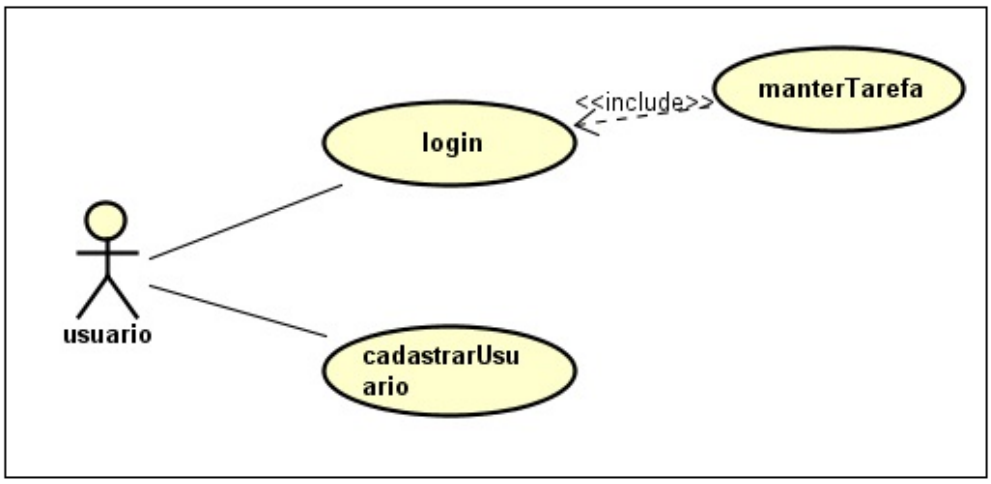
- Hospedar a aplicação PWA no Firebase;

- Permitir fazer o cadastro de usuário utilizando a conta do Google ou Facebook;

- Possibilitar operação offline da aplicação com sincronização dos dados quando a conexão é retomada.

A Figura 2 apresenta o diagrama de casos de uso da aplicação PWA proposta, com base nos requisitos funcionais da aplicação. O diagrama mostra a interação dos atores do sistema com os casos de uso. Três casos de uso são previstos: cadastro do usuário com e-mail e senha; fazer login; e manter tarefa (listar, incluir, alterar e excluir).

Figura 2 – Diagrama de Casos de Uso



Fonte: Do autor

2.3 Persistência de dados

Para persistência dos dados foi utilizado o Cloud Firestore, sendo o mesmo projeto para as duas aplicações geradas (Ionic e Flutter). Isso significa que as duas aplicações usam a mesma base de dados.

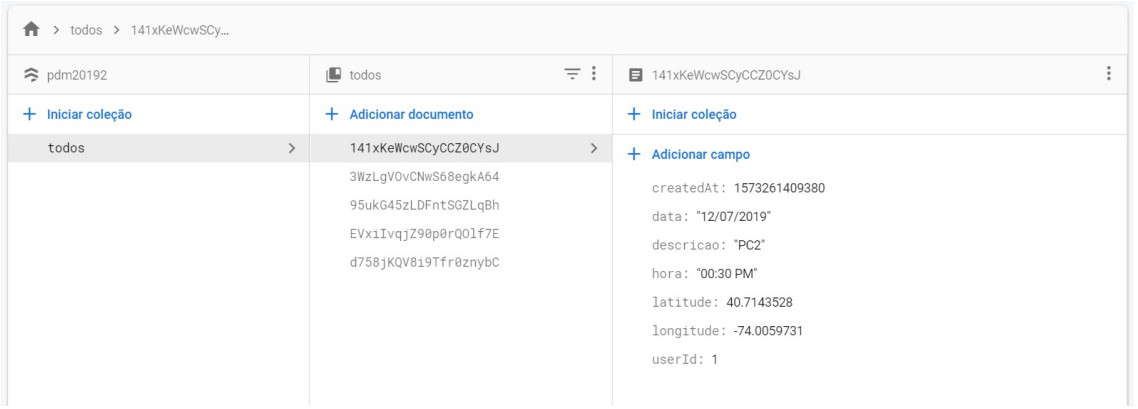
Seguindo o modelo de dados NoSQL do Cloud Firestore, o armazenamento ocorre em documentos que contêm mapeamentos de campos para valores. Esses documentos são armazenados em coleções, que são contêineres de documentos para organizar dados e criar consultas. Os documentos são compatíveis com muitos tipos de dados diferentes, desde strings e números simples a objetos complexos e aninhados. Também é possível criar subcoleções dentro dos documentos e criar estruturas de dados hierárquicas que podem ser escalonadas à medida que o banco de dados cresce. O modelo de dados do Cloud Firestore é compatível com qualquer estrutura de dados que funcione melhor para seu app (FIREBASE, 2019a).

Para o projeto de gerenciamento de tarefas apenas uma única coleção foi criada: *todos*, que pode ter a sua estrutura visualizada na Figura 3. Essa coleção possui por sua vez documentos que podem conter os seguintes campos de dados:

- *createdAt*: timestamp com a data de inclusão;
- *data* : data para realização da tarefa;
- *hora* : hora para realização da tarefa;
- *descricao* : descrição da tarefa;
- *latitude* : latitude da localização da tarefa;
- *longitude* : longitude da localização da tarefa;
- *user* : usuário que lançou a tarefa.

Cabe destacar que os campos latitude e longitude foram incluídos prevendo a possibilidade do usuário indicar um local de realização da tarefa, como uma informação opcional.

Figura 3 – Estrutura da coleção *todos* no Cloud Firestore

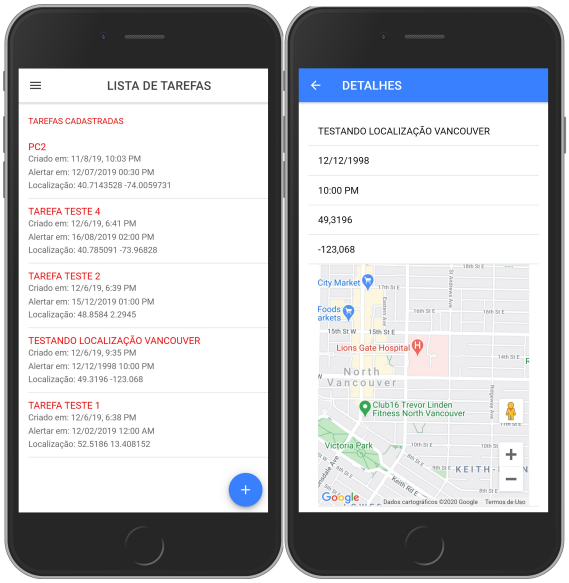


Fonte: Do autor

2.4 Detalhamento das aplicações

A Figura 4 apresenta as duas telas da aplicação desenvolvida com o framework Ionic. A primeira tela apresenta a listagem de tarefas do usuário. Essa tela também possui um botão com o sinal de adição que quando clicado direciona para a tela de edição. O usuário também pode fazer a exclusão de uma tarefa deslizando o item da tarefa na tela de listagem. Nesse caso aparece um botão para efetuar a exclusão. Já a segunda tela mostra um formulário onde é possível fazer a inclusão e alteração dos dados da tarefa. Nessa tela há um botão para efetivar a gravação dos dados. Um mapa também é apresentado nessa tela para que o usuário possa indicar o local da tarefa ou visualizar a tarefa quando em uma situação de alteração. Quanto a localização da tarefa, é facultado ao usuário sua seleção.

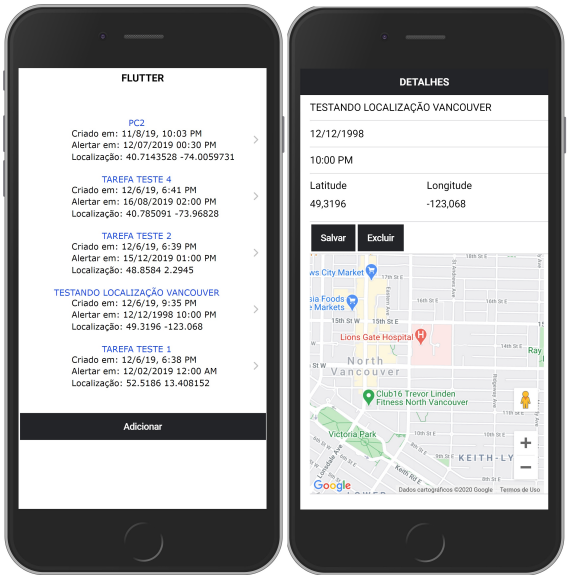
Figura 4 – Aplicação Ionic



Fonte: Do autor

Já a Figura 5 ilustra as telas da aplicação gerada a partir do framework Flutter. Da mesma forma que a aplicação do Ionic, são visualizadas a esquerda a tela com a listagem das tarefas e na direita a tela que contém um formulário para manipulação dos dados. Seguiu-se assim a mesma lógica nas duas aplicações com pequenas modificações. A tela de listagem apresenta um botão com o label "Adicionar" que direciona para segunda tela, o formulário de edição. A página de edição apresenta os campos, o mapa e os botões "Salvar"e "Excluir". Ou seja, nesse caso a opção de excluir fica na tela de edição.

Figura 5 – Aplicação Flutter



Fonte: Do autor

2.5 Limitações quanto ao desenvolvimento

No processo de desenvolvimento foi incluída a parte de login do usuário como previsto nos requisitos funcionais. No entanto, a ferramenta Lighthouse, usada para realização da avaliação, considera a página inicial da aplicação para os testes que realiza. Nesse caso, considerou-se que a página que lista as tarefas seria mais interessante para efetivação dos testes e foi retirada a página de login.

3 AVALIAÇÃO

Para comparar a aplicação desenvolvida nos frameworks foi conduzida uma avaliação com a ferramenta Lighthouse.

O LightHouse é uma ferramenta automatizada de código aberto cujo objetivo é oferecer uma auditoria abrangente de todos os aspectos da qualidade de um aplicativo da web (DEVELOPERS, 2019a). Essa ferramenta executa uma série de testes na página e gera um relatório sobre o desempenho nesses testes, apresentando as falhas e os indicadores para o aprimoramento do aplicativo (RENKEL, 2019). Pode ser executado como uma extensão do Google Chrome ou por linha de comando (NodeJS).

A extensão do Lighthouse ajuda na comparação de sites com certas medidas e objetivos que o Google avalia como importantes, especialmente para PWAs. Embora a extensão funcione em sites não PWA também, ela foi projetada para ajudar a otimizar sites para melhor velocidade de renderização, tempo para a primeira interação do usuário e conformidade geral com o objetivo do Google para PWAs e o futuro da experiência na web.

A seguir são apresentados os itens de análise da auditoria do LightHouse:

- Desempenho: quão rápido está o desempenho do seu site (por exemplo, com que rapidez as páginas estão sendo carregadas, etc.)?
- Aplicativo da Web progressivo: o seu site fornece uma experiência moderna e semelhante a um aplicativo para os visitantes?
- Acessibilidade: quão acessíveis são as suas páginas da Web? Quais são as suas vulnerabilidades?
- Melhores práticas: o seu site está de acordo com as práticas recomendadas do Google?
- SEO: você precisa revisar ou melhorar o seu SEO para ajudar o seu site a ter melhor classificação?

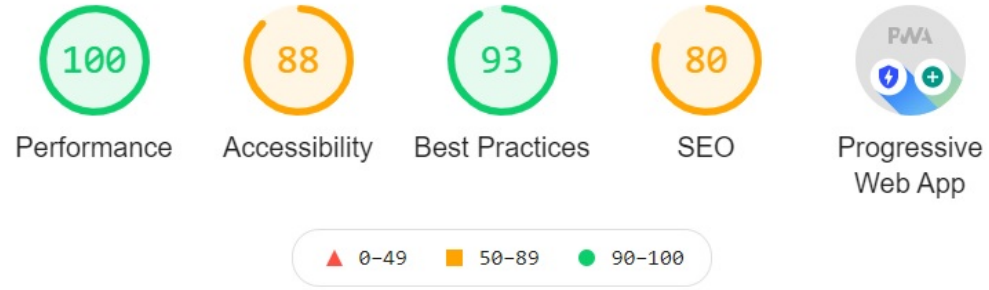
Com relação especificamente ao critério PWA, a ferramenta audita os seguintes itens:

- Registra um service worker;
- Responde com um código 200 quando estiver off-line;
- Contém algum conteúdo quando o JavaScript não está disponível;
- Usa https;
- O usuário pode ser solicitado a instalar o aplicativo da web;
- Configurado para uma tela inicial personalizada;
- A barra de endereço corresponde às cores da marca;
- Tem uma tag <meta name = viewport> com largura ou escala inicial;
- O conteúdo é dimensionado corretamente para a viewport;
- Redirecionar o tráfego http para https;
- O carregamento de páginas é rápido o suficiente em 3G;

3.1 Avaliação aplicação Ionic

Ao submeter a aplicação desenvolvida em Ionic a avaliação do Lighthouse, obteve-se as pontuações conforme apresentadas na Figura 6. Avalia-se tal resultado como positivo, já que em todos os critérios se obteve uma boa pontuação.

Figura 6 – Resultado da Avaliação com Lighthouse para aplicação IONIC



Fonte: Do autor

Com relação ao critério acessibilidade que obteve 88 pontos, a ferramenta apontou alguns problemas e fez sugestões de melhorias que são apresentadas a seguir:

- As cores de fundo e de primeiro plano não têm uma taxa de contraste suficiente: Para corrigir utilizar cores de contraste mais forte no primeiro plano;
- user-scalable = "no"é usado no elemento meta name= "viewport"ou o atributo de escala máxima é menor que 5: Utilizar formas de zoom para usuários com baixa visão.

Cabe destacar também que no Ionic a pontuação da categoria SEO foi 80, e o relatório indicou um problema:

- O documento não tem uma meta descrição: Incluir meta description nos resultados de pesquisa para resumir concisamente o conteúdo da página.

Já o relatório gerado exclusivamente para a categoria PWA, obteve os resultados mostrados na Figura 7. A avaliação considerou a aplicação rápida e confiável, instalável e cumprindo a maioria dos itens de otimização. No entanto, dois itens ficaram prejudicados na avaliação e ajustes foram recomendados:

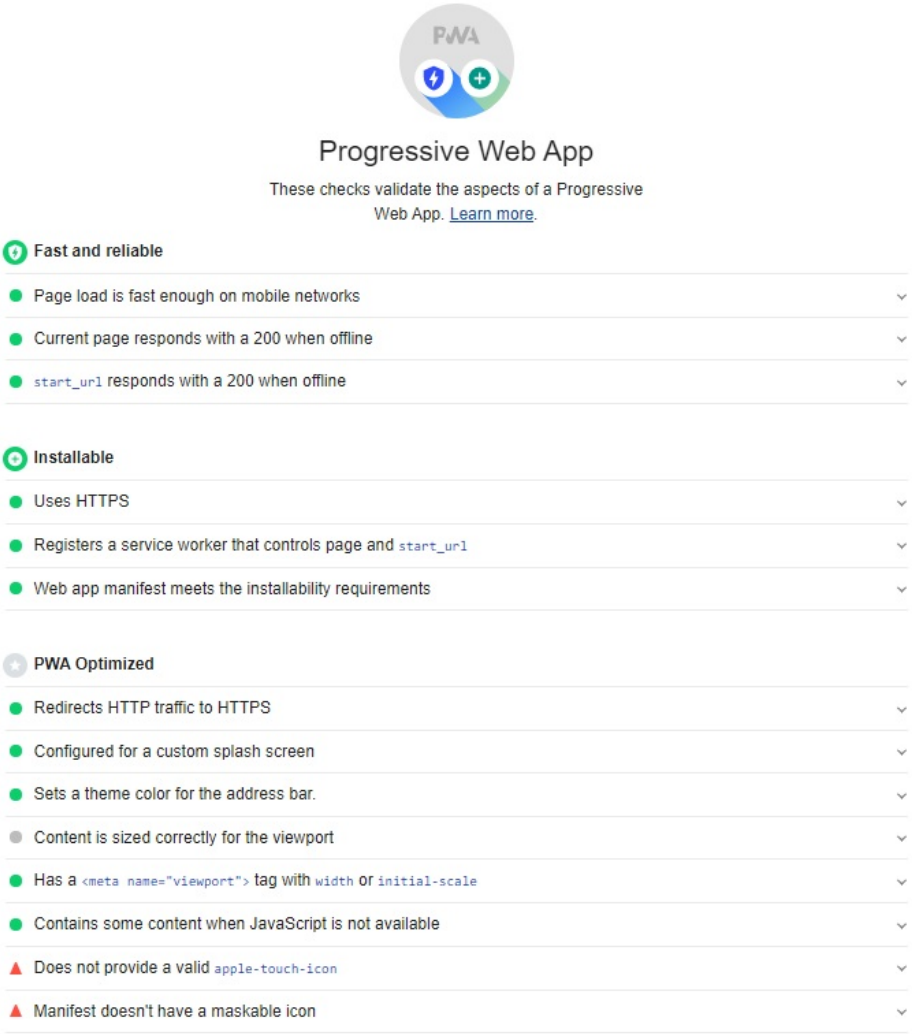
- Does not provide a valid apple-touch-icon;

  Para uma aparência ideal no iOS quando os usuários adicionam um aplicativo PWA à tela inicial, defina um "ícone de toque da Apple". Ele deve apontar para um PNG quadrado não transparente de 192px (ou 180px).

- Manifest doesn't have a maskable icon;

  Pode ser corrigido com um ícone mascarado que garante que a imagem preencha toda a forma sem ser letterbox ao instalar o aplicativo em um dispositivo.

Figura 7 – Avaliação PWA da aplicação IONIC

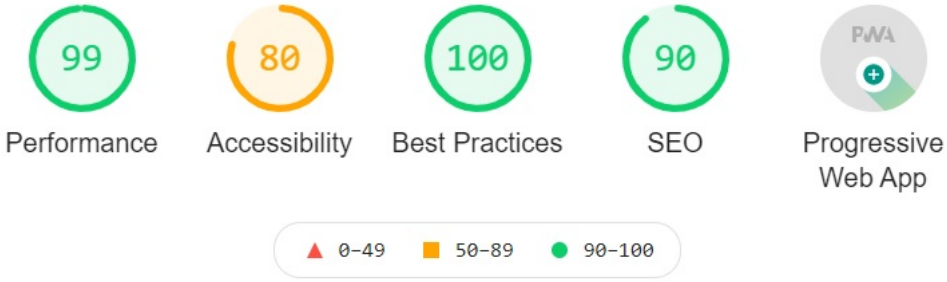


Fonte: Do autor

3.2 Avaliação aplicação Flutter

A segunda aplicação, que foi criada em Flutter, obteve as pontuações mostradas na Figura 8. Essa aplicação também alcançou um bom resultado.

Figura 8 – Avaliação da aplicação desenvolvida com o framework FLUTTER



Fonte: Do autor

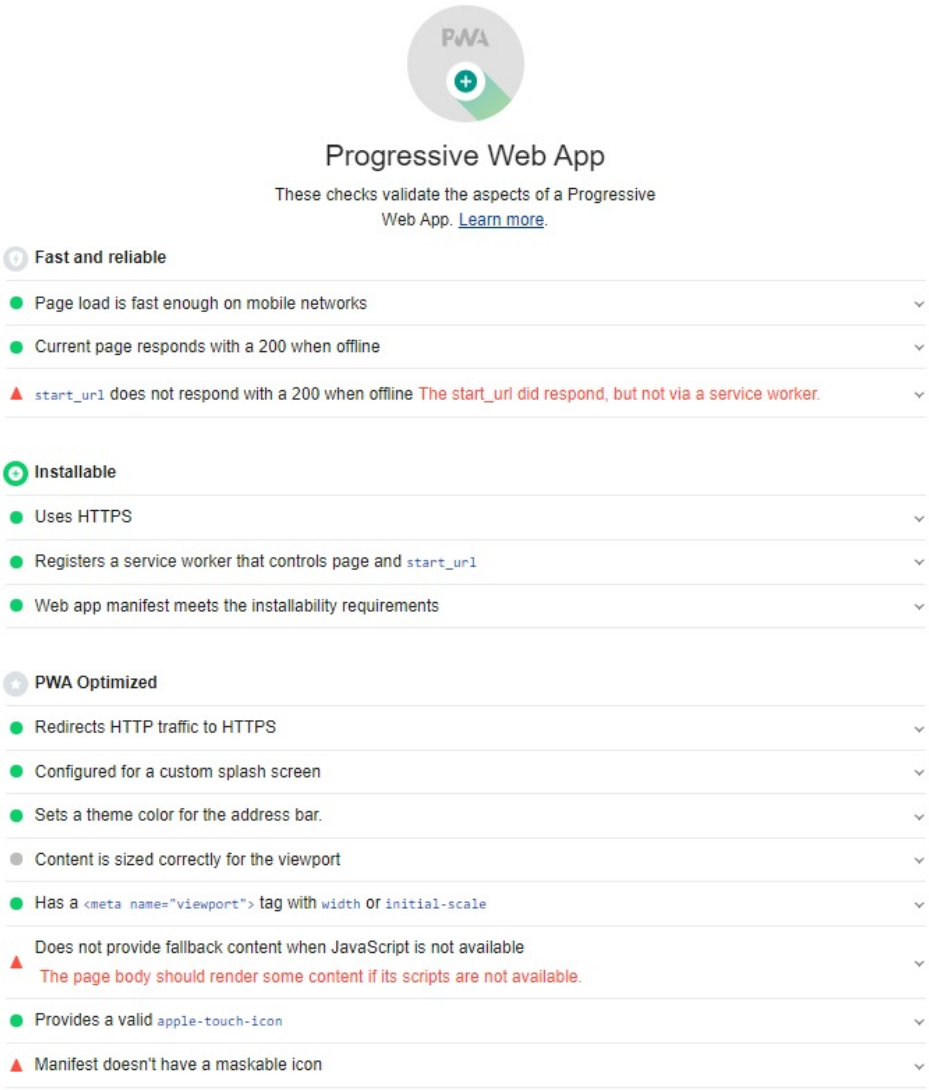
O único critério que ficou abaixo de 90 pontos foi o de acessibilidade que recebeu alguns apontamentos de erros e as recomendações relatadas a seguir:

- atributos aria- * não têm valores válidos: Corrigir setando valores válidos no atribulo ARIA, para funcionamento de tecnologias assistivas, como leitores de tela;
- elemento <html> não tem um atributo [lang]: Adicionar um atributo lang, pois se não tiver a página presume que está no idioma padrão que o usuário escolheu;
- user-scalable = "no"é usado no elemento meta name = "viewport"ou o atributo de escala máxima é menor que 5: Utilizar formas de zoom para usuários com baixa visão.

Conforme verificado no relatório gerado para a categoria PWA, a aplicação Flutter alcançou os resultados mostrados na Figura 9. Para essa aplicação, o relatório apontou um problema relacionado a velocidade de resposta e dois problemas vinculados a otimização. Esses apontamentos e as recomendações de correção são listados a seguir:

- start url does not respond with a 200 when offline: Pode ser corrigido com Workbox que é a abordagem recomendada para adicionar service workers a sites porque automatiza muitos padrões, torna mais fácil seguir as práticas recomendadas e evita bugs sutis que são comuns ao usar a API ServiceWorker de baixo nível diretamente;
- Does not provide fallback content when JavaScript is not available: Para corrigir, o aplicativo deve exibir algum conteúdo quando o JavaScript está desabilitado, mesmo que seja apenas um aviso ao usuário de que o JavaScript é necessário para usar o aplicativo;
- Manifest doesn't have a maskable icon: Pode ser corrigido com um ícone mascarado que garante que a imagem preencha toda a forma sem ser letterbox ao instalar o aplicativo em um dispositivo.

Figura 9 – Avaliação PWA da aplicação FLUTTER



Fonte: Do autor

Considerações finais

Há uma tendência que PWAs venham a substituir grande parte dos aplicativos nativos, tendo em vista que os navegadores começam a ter recursos de sistemas operacionais que hospedam outros aplicativos. PWAs tem como fundamento que essas aplicações sejam aprimoradas gradualmente, considerando os recursos que estão sendo incorporados aos navegadores.

Frameworks, que são ferramentas geralmente utilizados para o desenvolvimento de aplicações, encapsulam detalhes de implementação voláteis através de seus pontos de extensão, interfaces estáveis e bem definidas, aumentando a modularidade da aplicação. Eles incentivam o reuso e aumentam a produtividade dos desenvolvedores (PUC-RIO, 2006).

Assim, esse trabalho propôs avaliar o impacto da escolha de um framework no resultado final de uma aplicação PWA quanto aos aspectos de desempenho e qualidade. Para fazer a comparação foi desenvolvida uma aplicação de gerenciamento de tarefas nos frameworks Ionic e Flutter e utilizada a ferramenta Lighthouse para a avaliação.

Conforme apresentado na Seção 3, os resultados da avaliação explicitados nos relatórios gerados pelo Lighthouse não evidenciaram grande diferença nos critérios da ferramenta. Nas duas aplicações foi verificada a necessidade de alguns ajustes para melhorar os aspectos de acessibilidade (2 itens na aplicação Ionic e 3 itens na Flutter). Para a aplicação IONIC também ocorreu um apontamento de melhoria quanto ao critério de otimização no mecanismos de busca (SEO).

Na avaliação quanto a auditoria dos itens de PWA, alguns pontos de melhoria foram relatados, porém a aplicação Ionic obteve um desempenho um pouco melhor, pois na aplicação Flutter ocorreu uma fragilidade quanto a rapidez de resposta.

Cabe destacar que o desenvolvimento nos frameworks primou pela utilização das configurações padrão das ferramentas quanto aos aspectos de PWA, justamente para poder avaliar o que os frameworks já prevem para atender os requisitos de qualidade, conforme avaliação do Lighthouse. Assim, após avaliação com o Lighthouse e observando que alguns apontamentos de correção seriam simples de serem realizados, optou-se em manter as aplicações como geradas inicialmente e mostrar os resultados gerados pelo Lighthouse.

Além da avaliação do Lighthouse, cabe destacar alguns pontos observados pelo autor durante o processo de desenvolvimento. Em especial com relação ao acesso a documentação, onde observou-se uma possibilidade muito maior de conteúdo para o framework Ionic. Esse último, possui mais documentação na internet, o que o torna mais fácil e rápido corrigir erros ou encontrar diferentes alternativas para implementação. Já o Flutter possui pouca documentação, tornando muito difícil encontrar soluções. Ainda, observou-se que, como o framework é relativamente recente, contém muitos bugs e novas versões são disponibilizadas com muita frequência, o que dificultou a implementação em alguns momentos.

Tem-se como proposta de melhorias futuras a possibilidade de corrigir os erros que apontados nas aplicações geradas de maneira padrão pelos frameworks e verificar como será a avaliação com a ferramenta LightHouse. Além disso, também vislumbra-se incluir a utilização de API na página inicial (como Google Maps) e averiguar se isso interferirá na avaliação.

Evaluative comparison between Ionic and Flutter frameworks

Lucas Kurtz Moreira* Anubis Rossetto †

2020

Abstract

This article presents a comparative assessment between the Ionic and Flutter frameworks for development using a Progressive Web App (PWA) approach. Adding more features to web apps progressively to have an experience and function as native mobile apps is a recent concept that has been gaining adherents. In this way, a task management application was developed in both frameworks, incorporating the services of the Google Firebase platform for storage and hosting of applications. After the development of the applications, an evaluation was carried out using a tool that provides an audit on several aspects, such as, accessibility, performance, search engine optimization (SEO), best practices and PWA.

Keywords: PWA. Ionic. Flutter. Lighthouse.

Referências

BLACKBERRY, O. Cross platform development (hybrid vs native). *Journal of Applied Technology and Innovation (eISSN: 2600-7304)*, v. 1, n. 2, p. 67–78, 2017. Citado na página 5.

*<lucaskurtz96@gmail.com>
†<anubisrossetto@ifsul.edu.br>

DEVELOPERS, G. *Lighthouse Scoring Guide*. 2019. Disponível em: <https://developers.google.com/web/tools/lighthouse/v3/scoring>. Acesso em: 13 nov 2019. Citado na página 11.

DEVELOPERS, G. *Service Workers: an Introduction*. 2019. Disponível em: <https://developers.google.com/web/fundamentals/primers/service-workers>. Acesso em: 23 out 2019. Citado na página 4.

DONALDSON, S. *Service Workers in the Browser (Not Clerks The Movie)*. 2017. Disponível em: <https://developers.redhat.com/blog/2017/03/30/service-workers-in-the-browser-not-clerks-the-movie/>. Acesso em: 08 nov 2019. Citado na página 5.

FERNANDES, M. F. G. *Udyat: um aplicativo híbrido com o intuito de assegurar o usuário no meio em que vive. Trabalho de conclusao de curso IFSUL, Passo Fundo*. 2019. Citado na página 6.

FIREBASE, G. D. *Cloud Firestore*. 2019. Disponível em: <https://firebase.google.com/docs/firestore/?hl=pt-br>. Acesso em: 18 nov 2019. Citado 2 vezes nas páginas 7 e 8.

FIREBASE, G. D. *Firebase*. 2019. Disponível em: <https://firebase.google.com/>. Acesso em: 16 mai 2019. Citado na página 6.

FIREBASE, G. D. *Firebase Hosting*. 2019. Disponível em: <https://firebase.google.com/docs/hosting/?hl=pt-br>. Acesso em: 18 nov 2019. Citado na página 7.

FRAMEWORK, I. *What is Ionic Framework?* 2019. Disponível em: <https://ionicframework.com/docs/intro>. Acesso em: 24 out 2019. Citado na página 5.

GONSALVES, M. Evaluating the mobile development frameworks apache cordova and flutter and their impact on the development process and application characteristics. 2019. Citado na página 2.

KUBRAK, V. *What are the advantages and disadvantages of PWA over Native Apps?* 2017. Disponível em: <https://pwapps.cloud/advantages-and-disadvantages-of-pwa-over-native-apps>. Acesso em: 19 nov 2019. Citado na página 2.

MEDIUM. *Introdução ao Flutter: O Básico*. 2019. Disponível em: <https://medium.com/flutter-angola/introdu%C3%A7%C3%A3o-ao-flutter-o-b%C3%A1sico-f8c8302be95c>. Acesso em: 24 out 2019. Citado na página 6.

PUC-RIO. *Frameworks: conceitos gerais*. 2006. Disponível em: <https://www.maxwell.vrac.puc-rio.br/8623/8623_3.PDF>. Acesso em: 30 nov 2020. Citado na página 16.

RENKEL, G. *Google LightHouse*. 2019. Disponível em: <https://www.secnet.com.br/blog/google-lighthouse>. Acesso em: 07 nov 2019. Citado na página 11.

RUSSEL, A. *Progressive Web Apps Escaping Tabs Without Losing Our Soul*. 2015. Disponível em: <https://infrequently.org/2015/06/progressive-apps-escaping-tabs-without-losing-our-soul>. Acesso em: 12 out 2017. Citado na página 3.