COMPARANDO O DESENVOLVIMENTO DE APLICATIVO PARA APARELHO MÓVEL UTILIZANDO REACT NATIVE E FLUTTER

Cesar Rocha Camilo

Prof. Dr. Thiago Schumacher Barcelos

Instituto Federal de Educação, Ciência e Tecnologia de São Paulo

Resumo

O desenvolvimento de aplicativos para aparelhos móveis está em crescimento constante, aquecendo o mercado de desenvolvimento de aplicativos. Este movimento provocou o surgimento de ferramentas que permitem gerar um aplicativo para diversas plataformas utilizando o mesmo código-fonte, mas qual *framework* provê melhor ambiente de desenvolvimento para programadores iniciantes e sem conhecimento em desenvolvimento de aplicativos? Este artigo apresenta um estudo de caso comparativo entre os *frameworks* React Native e Flutter em um experimento controlado realizado no time de Desenvolvimento do Hospital das Clínicas da Faculdade de Medicina da Universidade de São Paulo. Para responder à questão de pesquisa, o estudo se fundamenta em três pilares: tempo de execução, medido a partir do cronograma das atividades; métricas de software, aplicadas nos códigos gerados a partir dos *frameworks*, com intuito de medir a qualidade do código (Complexidade ciclomática, Linhas de execução e Taxa de manutenção); entrevista e questionário com os programadores envolvidos para analisar a sua percepção acerca dos *frameworks*.

Os resultados indicam que o Flutter é melhor para o desenvolvimento de aplicativos com menor complexidade de regras de negócio e para programadores iniciantes, enquanto que React Native é mais vantajoso para o desenvolvimento de aplicativos mais robustos.

Palavras-chave: Multiplataforma; React Native; Flutter.

1. Introdução

Com a difusão da internet e o crescimento do uso de dispositivos móveis, o desenvolvimento de aplicativos para plataforma *mobile* está em grande ascensão. Segundo artigo publicado no site Digitalks (GALVÃO, 2021), em 2017, quase 80% do acesso a internet foi realizado via plataformas *mobile*. Este número, em 2016, era de 68%, e em 2012 de apenas 40%, demonstrando que o uso da internet em aparelhos móveis dobrou nos últimos anos. Segundo (IBGE, 2020), o smartphone tornou-se o principal meio de acesso à internet no Brasil. Deste modo, ter um aplicativo se torna um canal de comunicação muito significativo para as empresas com seus clientes.

Os dispositivos móveis necessitam de um sistema operacional para funcionar e, entre as diferentes plataformas, as mais utilizadas são Android, Apple iOS e Windows Phone (BOULOS et al., 2011). Ao produzir um aplicativo, a meta é atingir o maior número possível de usuários, dessa forma, até alguns anos atrás se fazia necessário desenvolver o mesmo aplicativo para cada plataforma. Por conseguinte, desenvolver um aplicativo se tornava um projeto de alta complexidade e custo elevado (BERNARDES; MIYAKE, 2016), pois a equipe de desenvolvimento deveria dominar as linguagens nativas de cada plataforma. Por serem linguagens distintas, ao desenvolver um app, era necessário escrever o código individualmente para cada S.O. (SAMBASIVAN et al. 2011), criando um projeto para cada plataforma, dificultando o gerenciamento e manutenção do código, consequentemente, o custo do projeto se eleva, por ter mais profissionais envolvidos. Todo este movimento fomentou o surgimento de ferramentas para desenvolvimento de aplicativos multiplataforma. Estes permitem gerar aplicativos compatíveis com Android e iOS utilizando o mesmo código fonte, mas qual melhor *framework* para se desenvolver um aplicativo que possa ser utilizado em plataformas distintas? Esta foi a incerteza enfrentada pelo time de desenvolvimento do Hospital das Clínicas de São Paulo ao se deparar com o desafio de transformar um sistema web já existente em aplicativo para Android e iOS.

A fim de sanar esta dúvida, o time de desenvolvimento selecionou os *frameworks* React Native e Flutter para um teste prático. Foram selecionados dois programadores do time, ambos sem conhecimento prévio em desenvolvimento de app móveis, e a meta consistia em cada um desenvolver um app fictício utilizando um dos *frameworks*. Ao final do prazo estipulado, os resultados seriam comparados e analisados, a fim de apontar o *framework* mais adequado a ser utilizado pelo time. A análise dos dados coletados está norteada em responder às seguintes questões de pesquisa:

- Entre os *frameworks* utilizados no estudo, qual provê melhor ambiente de desenvolvimento para programadores iniciantes?
- Qual programador obteve maior velocidade de execução das tarefas previamente estabelecidas?
- Qual *framework* obteve melhores avaliações qualitativas nas métricas de software aplicadas?

2. Materiais e Métodos

Este estudo de caso, além de um experimento controlado, contou com uma entrevista com os programadores envolvidos e um questionário com perguntas focalizadas nas impressões sobre os *frameworks* e como medidas quantitativas. Foram extraídas métricas de software dos códigos gerados no experimento, juntamente com o cronograma de execução das tarefas, do qual foi obtida a medida do esforço de cada programador em horas.

Para o experimento, foi estabelecida uma meta: desenvolver uma interface que resgate e apresente os agendamentos e receitas do paciente, com opção de filtro por período e ordenação decrescente dos itens e com a premissa de utilizar as APIs já utilizadas no sistema web previamente existente.

Por não ter o conhecimento técnico em ambos *frameworks*, o time de desenvolvimento solicitou à sua diretoria que custeasse um curso de React Native e Flutter, para adquirir o mínimo de conhecimento necessário para iniciar o teste comparativo. Entre os membros do time foram selecionados dois programadores de nível pleno para realizar os cursos, na mesma instituição de ensino. Ambos programadores não possuíam conhecimento prévio quanto ao desenvolvimento mobile.

O questionário aplicado ao término do projeto é composto por questões com respostas de valor numérico definido de 1 a 10, sendo 1 classificado como ruim e 10 como bom. Em seguida, a entrevista foi conduzida com as mesmas indagações do questionário, com objetivo de coletar as impressões e percepções dos envolvidos durante a execução do projeto.

As métricas utilizadas foram Complexidade ciclomática (MCCABE, 1976), Linhas de execução do código-fonte (PRESSMAN, 2011) e Taxa de Manutenção (HARIPRASAD et al., 2017). Para implementar esta etapa, foi utilizado a ferramenta Plato v1.6.0 no React Native e Dart Code Metrics v2.4.0 para Flutter. Ambas as ferramentas foram analisadas para garantir as mesmas fórmulas de cálculo empregadas, certificando os resultados justos e equiparáveis.

Para mensurar o esforço empregado, as tarefas foram listadas no Product Backlog, seguindo a metodologia Scrum e posteriormente, foram decompostas em sub tarefas priorizadas em *sprints* durante a Sprint Planning. As tarefas foram registradas na ferramenta Trello, contendo a data de início e fim das tarefas. Os desenvolvedores registraram no Trello a conclusão de cada tarefa e com base nos *logs* da ferramenta, foi extraído o tempo em horas do esforço empregado em cada uma das tarefas.

3. Resultados e Discussão

Após a conclusão do curso, o projeto Teste foi apresentado aos programadores, por meio da apresentação dos requisitos e cronograma das atividades a serem realizadas. Também foi definido qual *framework* cada programador iria utilizar. A seguir, o projeto foi iniciado. Para fácil identificação neste trabalho, nomeamos o programador que utilizou React Native de Prog. R, e nomeamos de Prog. F o que utilizou Flutter. Durante a execução do projeto, reuniões diárias (Daily Scrum) ocorreram para o monitoramento das entregas. O autor deste trabalho participou do projeto no papel de Scrum Master e coordenador da equipe, facilitando e garantindo a execução dos eventos definidos na metodologia Scrum.

3.1 Cronograma de execução

Ao analisar o cronograma e as horas utilizadas por cada programador na finalização de cada tarefa, constatou-se que o desenvolvedor que utilizou Flutter registrou menos tempo de trabalho em comparação com o programador que utilizou React Native, sendo que este último não conseguiu atingir as expectativas de horas estipuladas no início do projeto, excedendo em 32 horas o cronograma. Os dados são apresentados na Tabela 1.

Tabela 1: Cronograma de execução

Sprint	React Native	Flutter	Horas estimadas
1	56	32	40
2	56	40	40
3	40	32	40
4	40	40	40
Total	192	144	160
Horas excedentes	32	0	-

3.2 Métricas

O *framework* Flutter obteve maior valor na métrica complexidade ciclomática (Tabela 2) em comparação com o RN, apresentando uma maior quantidade de caminhos independentes nas funções equivalentes desenvolvidas em RN. Isto pode ser observado na entrevista, quando o Prog. F afirma: "A primeira vez a linguagem do Dart tem semântica um pouco mais diferente, com bastante parênteses e chaves e pode acabar confundindo e levando um tempo até você se adaptar". Os dados apresentados na Tabela 2, somando à impressão do desenvolvedor, leva a crer que o código gerado no Flutter se torna mais complexo. Neste quesito, RN tem uma estrutura de código mais simples. No questionário (Tabela 4), isto fica evidenciado, quando Flutter obteve nota 6 no quesito Estrutura e componentes da linguagem e RN recebeu a nota 9.

Tabela 2: Dados obtidos com a aplicação da métrica Complexidade ciclomática. Fonte: o autor.

	ListaReceitas	DetalhaReceitas	ListaAgendas	DetalhaAgendas
Flutter	0	13	0	14
React Native	5	1	6	1

A quantidade de linhas geradas a partir do Flutter se excede em confrontação com seu concorrente, mas mesmo sendo maior em quantidade de linhas, a produtividade com Flutter se confirma superior em relação ao RN com a análise das horas de desenvolvimento apresentadas no cronograma. A métrica de Linhas de execução de código por módulo do projeto desenvolvido em cada *framework* pode ser visualizado na Tabela 3.

Tabela 3: Dados obtidos com a aplicação da métrica Linhas execução código. Fonte: o autor

	ListaReceitas	DetalhaReceitas	ListaAgendas	DetalhaAgendas
Flutter	71	170	71	195
React Native	88	55	140	53

A produtividade com o uso do Flutter foi avaliada com nota 9 no questionário, enquanto que RN obteve nota 8 na questão: “Classifique sua percepção acerca da produtividade utilizando este *framework*”. Os resultados desta e das demais questões do questionário estão disponíveis na Tabela 4.

Tabela 4: Resultados do questionário

Questão	**React Native**	**Flutter**
Qual o seu grau de satisfação acerca da documentação do Framework?	8	10
Classifique a facilidade de instalação e configuração do ambiente de desenvolvimento do Framework	6	10
Classifique a estrutura e componentes da linguagem utilizada pelo Framework (React: Javascript | Flutter: Dart)	9	6
Classifique a facilidade na codificação e fluidez do código utilizando o Framework.	8	10
Classifique sua percepção acerca da performance (velocidade e desempenho) do Framework.	8	10
Classifique sobre os estilos de componentes padrões disponíveis no Framework.	8	10
Classifique sua percepção acerca da produtividade utilizando este Framework.	8	9

A métrica taxa de manutenção (Tabela 5) utiliza os valores obtidos nas métricas complexidade ciclomática e quantidade de linhas de código em seu cálculo, sendo assim, Flutter registrou uma menor taxa de manutenção, expondo uma maior dificuldade em dar manutenção neste código.

Tabela 5: Dados obtidos com a aplicação da métrica Taxa de manutenção. Fonte: o autor.

	ListaReceitas	DetalhaReceitas	ListaAgendas	DetalhaAgendas
Flutter	37	24	38	21
React Native	42	44	41	44

4. Considerações Finais

Com a análise dos dados apresentados, é possível obter as respostas das questões introduzidas nos objetivos deste artigo:

Entre os *frameworks* utilizados no estudo, qual provê melhor ambiente de desenvolvimento para programadores iniciantes? R: Com base nos dados coletados, o *framework* Flutter obteve melhores resultados sobre o ambiente de desenvolvimento quando aplicados à programadores iniciantes, com pouca ou nenhuma experiência neste *framework*.

Qual programador obteve maior velocidade de execução das tarefas previamente estabelecidas? R: Com base no cronograma e horas de esforço utilizado para cada programador, o que utilizou o *framework* Flutter obteve melhor resultado em comparação com RN. O Prog. F utilizou 144 horas para finalizar todas as tarefas, em contrapartida, o Prog. R necessitou de 192 horas, excedendo em 32 o total estimado de 160 horas.

Qual *framework* obteve melhores avaliações qualitativas nas métricas de software aplicadas? R: Ainda que o Flutter tenha obtido melhores resultados nas questões prévias, o RN atingiu melhores resultados quando submetido às métricas de software, com melhor manutenibilidade e menor complexidade ciclomática, React Native consegue entregar as mesmas funcionalidades com menos linhas de código em comparação ao seu concorrente Flutter.

Isto posto, a hipótese de que o Flutter é melhor para o desenvolvimento de aplicativos com menor complexidade de regras de negócio e para programadores iniciantes é percebida. React Native obteve melhores resultados em todas as métricas de software aplicadas, porém o desenvolvedor utilizou mais horas para atingir os mesmos resultados, sendo assim, é possível indicar a hipótese de que RN é o melhor *framework* para aplicativos maiores e que irão sofrer mais alterações ao longo do tempo.

Com as devidas conclusões, o time de desenvolvimento do HC aderiu ao Flutter como *framework* para o desenvolvimento do Portal do Paciente app, devido sua maior velocidade de entrega e por este aplicativo ser de menor complexidade, não sendo um problema a taxa de manutenção do código gerado no Flutter ser inferior quando comparado com o código gerado no React Native.

5. Referências

BERNARDES, T. F.; MIYAKE, M. Y. Cross-platform Mobile Development Approaches: A Systematic Review. **Cross-platform Mobile Development Approaches: A Systematic Review**, [*s. l.*], v. 14, n. 4, p. 1892–1898, 2016. Disponível em: https://doi.org/10.1109/TLA.2016.7483531. Acessado em 6 abr. 2021.

BOULOS, M. *et al.* **How smartphones are changing the face of mobile and participatory healthcare: an overview, with example from eCAALYX**. [*S. l.*], 2011. Disponível em: <https://biomedical-engineering-online.biomedcentral.com/articles/10.1186/1475-925X-10-24>. Acesso em: 6 abr. 2021.

GALVÃO, D. **CRESCIMENTO DO MERCADO MOBILE ALTERA COMPORTAMENTO SOBRE USO DE APPS**. [*S. l.*], 2021. Disponível em: <https://digitalks.com.br/artigos/crescimento-do-mercado-mobile-altera-comportamento-sobre-uso-de-apps/>. Acesso em: 6 abr. 2021.

HARIPRASAD, T. et al. 2017 **International Conference on Trends in Electronics and Informatics (ICEI)**. In: 2017 INTERNATIONAL CONFERENCE ON TRENDS IN ELECTRONICS AND INFORMATICS (ICEI), 2017, Tirunelveli, India. Anais [...]. Tirunelveli, India: [s. n.], 2017. Disponível em: <https://doi.org/10.1109/ICOEI.2017.8300883>. Acesso em: 14 abr. 2021.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. **Acesso à internet e à televisão e posse de telefone móvel celular para uso pessoal 2018**. [*S. l.*]: Instituto Brasileiro de Geografia e Estatística, 2020. Disponível em: <https://biblioteca.ibge.gov.br/index.php/biblioteca-catalogo?view=detalhes&id=2101705>. Acesso em: 6 abr. 2021.

MCCABE, T. **A Complexity Measure. In: IEEE TRANSACTIONS ON SOFTWARE ENGINEERING**, 1976. Anais [...]. [S. l.: s. n.], 1976. Disponível em: <http://www.literateprogramming.com/mccabe.pdf>. Acesso em: 7 jun. 2021.

PRESSMAN, R. Engenharia de Software Uma Abordagem Profissional. 7. ed. [S. l.]: Mc Graw Hill & Bookman, 2011.

SAMBASIVAN, D. *et al.* **Generic framework for mobile application development**. [*S. l.*], [*s. d.*]. Disponível em: <https://ieeexplore.ieee.org/document/6113938>. Acesso em: 6 abr. 2021.