Análise de READMEs de bibliotecas e frameworks open-source iOS escritos em Swift

Hilton Leite, Bruno Melo

Abstract

Este artigo apresenta uma análise de arquivos README.md de 85 repositórios de código aberto hospedados no GitHub. O principal objetivo foi mapear o estado da arte da escrita desses arquivos para bibliotecas e *frameworks* iOS escritos em Swift, e consequentemente prover um guia de boas práticas para escrita de READMEs para projetos deste tipo.

Keywords: iOS, Swift, README, Código aberto, Experiência do Desenvolvedor

1. Introdução

No contexto de software de código aberto, arquivos README são utilizados para documentar informações e detalhes específicos do projeto, entre eles: passos para realizar *build*, instalação, uso, e licença de distribuição. Devido ao seu conteúdo, e ao destaque que recebem nas plataformas de hospedagem de código-fonte, como o GitHub, READMEs são de grande importância para instruir novos usuários, e possíveis colaboradores.

No desenvolvimento de aplicativos móveis para a plataforma iOS, é muito comum a adição de bibliotecas e *frameworks* (posteriormente referidos apenas como bibliotecas) que facilitam o desenvolvimento, potencialmente o tornando mais rápido. Nesse ecossistema existem diversas bibliotecas que realizam funções similares, por isso, um README que consegue descrever de forma efetiva a instalação e uso da biblioteca pode ser a diferença entre a adesão da mesma, ou a procura de alguma concorrente similar.

Email addresses: hpbl@cin.ufpe.br (Hilton Leite), bhlvm@cin.ufpe.br (Bruno Melo)

Embora estudos recentes venham abordando a temática, ainda não temos guias definitivos sobre escrita de README, ou ferramentas difundidas de análise automática, que nos proporcionem um entendimento sistemáticos da estrutura e conteúdo dos mesmos. Também não se encontram pesquisas relacionadas específicas ao sistema iOS e a linguagem Swift, assim não levando em conta suas especificidades.

Com isso tivemos como objetivo realizar uma análise dos READMEs das bibliotecas iOS de código aberto mais populares, com o intuito de mapear o estado da arte da escrita desse tipo de documento.

2. Coleta dos Dados

A coleta dos repositórios foi realizada de forma automatizada através de consultas à API do GitHub. Na primeira consulta requisitamos os cem repositórios com mais estrelas, que foram etiquetados como "iOS", e escritos na linguagem Swift.

A partir da listagem destes repositórios, foi feito um filtro manual, de forma com que sobrassem apenas oitenta e cinco bibliotecas, excluindo assim repositórios de listagens, aplicativos, e outros que não se pertenciam ao escopo.

Com o nome das bibliotecas desejadas, outras requisições foram feitas para obter e URL de *download* do README de cada repositório, que foram utilizadas para adquirir os mesmos.

3. Definição das Análises

Onze desenvolvedores iOS, com diferentes níveis de experiência foram questionados sobre que tipo de informação eles gostariam de obter dos dados, e que tipo de perguntas sobre os mesmos seriam relevantes. Obtivemos quarenta e quatro respostas, que abrangiam diversos aspectos, e que em algumas ocasiões se repetiam.

Realizamos a categorização e agrupamento das respostas em treze categorias, de acordo com o seu conteúdo: contribuição, testes, status (*build*/ Integração contínua), gerenciamento de pacotes, projeto de exemplo, licenciamento/autoria, propósito, versionamento/compatibilidade, instalação, estrutura do arquivo, formatação, uso, dependências.

Inspirados nas respostas recebidas e nas categorias identificadas, geramos também novos questionamentos que julgamos relevantes, e classificamos os mesmos de acordo com as categorias.

Categoria	**Análise**
Estrutura do arquivo	Quais são os títulos de seções mais comuns?
	Qual o tamanho dos arquivos README?
Gerenciamento de pacotes	Quais são os gerenciadores de pacotes mais adotados?
Contribuição	Existe um guia de contribuição?
Licenciamento / Autoria	Qual licença é utilizada?
	Autores tem mais de um repositório popular?
Status (*build* / integração contínua)	O status de *build* está presente?
	O *build* está passando?

Table 1: Análises selecionadas.

Priorizamos então as análises de acordo com a relevância para o nosso objetivo e factibilidade de implementação programática, resultando em oito análises, englobando seis categorias, como presente na Tabela 1.

4. Análise dos Dados

As análises foram implementadas de forma programática, utilizando a linguagem Python. Para facilitar o *parsing* dos arquivos README, convertemos os mesmos para o formato HTML, utilizando a biblioteca Python-Markdown, e com os arquivos convertidos utilizamos a biblioteca Beautiful-Soup explorar as *tags* e o conteúdo textual.

4.1. Estrutura dos arquivos

Nas análises relacionadas à estrutura dos arquivos README, foram observadas questões relacionadas à forma em que eles são escritos, como as *tags* utilizadas, e extensão do arquivo.

4.1.1. Quais são os títulos de seções mais comuns?

A primeira análise realizada mediu quais títulos de sessões são mais comumente utilizados, evidenciando como esperado, que sessões de instalação, uso, licença, e gerenciamento de pacotes, estão entre as mais frequentes.

Porém chamou a atenção a atenção o fato de por oito vezes ter aparecido o título de seção "Get the Showroom App for iOS to give it a try". Verificamos que os repositórios que continham essa seção eram todos de um mesmo autor, justificando assim sua repetição, mas levantando questionamento sobre o quão comum é que autores tenham mais de um repositório popular.

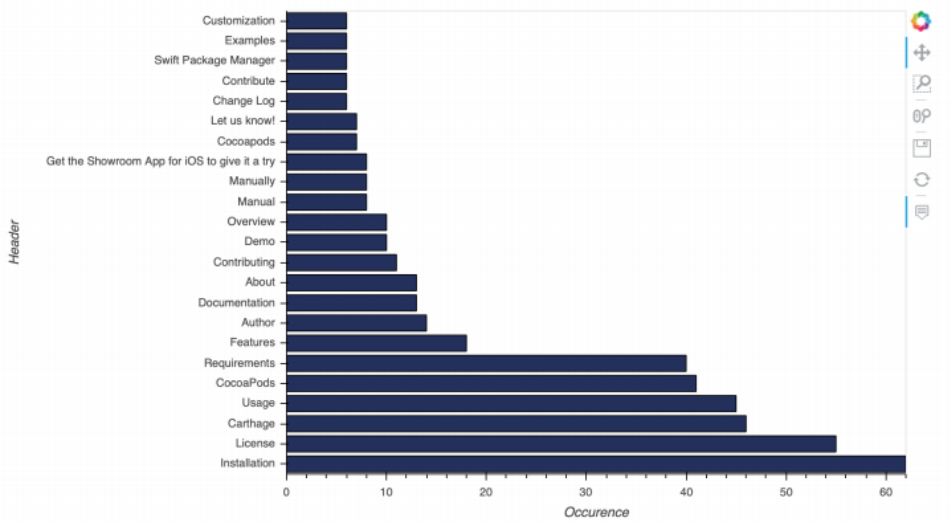

Figure 1: Títulos de seções mais comuns, que apareceram mais de 5 vezes.

Figure 2: Núvem de palavras dos títulos das sessões.

4.1.2. Qual o tamanho dos arquivos README?

Ao analisar a quantidade de linhas em cada arquivo, percebemos uma média de duzentas e dezenove linhas, com desvio padrão de cento e setenta

e um. Com essa informação temos um guia de tamanho esperado para um README de uma biblioteca iOS.

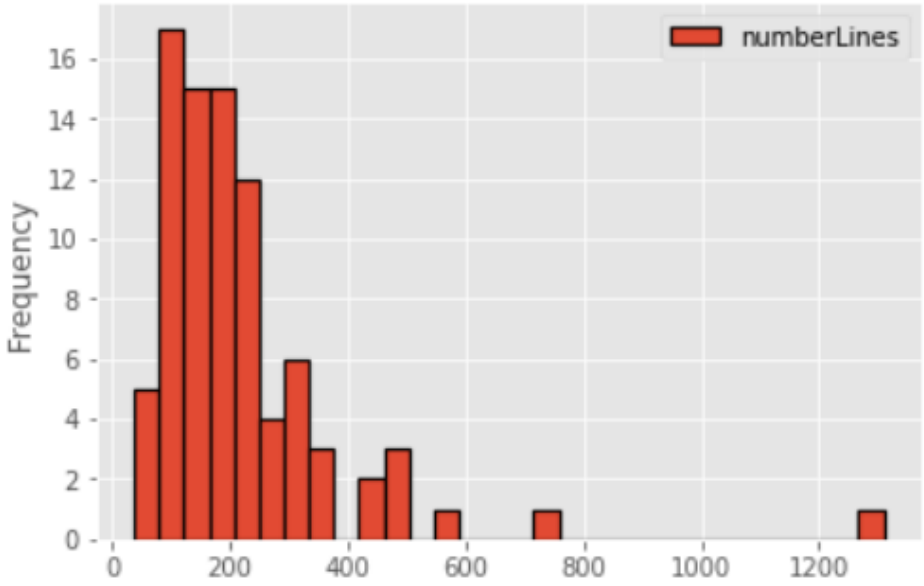

Figure 3: Tamanho dos arquivos README. Numero de linhas.

Verificamos individualmente a biblioteca com o menor número de linhas (trinta e oito), e descobrimos que se trata de um conjunto de componentes visuais modelados no estilo das interfaces gráficas dos anos noventa. O repositório tem uso bem direto e limitado, não necessitando de muito conteúdo no sue README.

Já o maior README, pertence a uma biblioteca de formulários, que tem diversos tipos de componente, sendo cada um documentado seu funcionamento com exemplos de código no próprio README. Tiramos a conclusão que quanto mais funcionalidades sua biblioteca tiver, maior vai ter que ser a documentação dela, podendo então gerar a opção de expor grande parte dessa documentação no README, o que tornará ele mais longo.

4.2. Gerenciamento de pacotes

É comum que bibliotecas sejam distribuídas através de gerenciadores de pacotes, que vão facilitar o processo de instalação, e lidar com as dependências necessárias do seu aplicativo.

4.2.1. Quais são os gerenciadores de pacotes mais adotados?

É pertinente saber quais gerenciadores de pacotes são mais utilizados, para que desenvolvedores de bibliotecas saibam por onde devem distribuir seus projetos proporcionando adesõa de forma mais fácil.

Como apontam os dados, o CocoaPods se consolida como o gerenciador de pacotes mais adotado, sendo mencionado em 94% dos READMEs, porém o Carthage também apresenta grande adoção, mencionado 80% das vezes. Observamos que o Swift Package Manager, embora seja embutido no próprio Swift, não foi tão bem aderido desde seu lançamento em 2016. Vale notar que CocoaPods e Carthage foram lançados respectivamente em 2011 e 2014, tendo então mais tempo para consolidação, melhorias, e adoção pelo público.

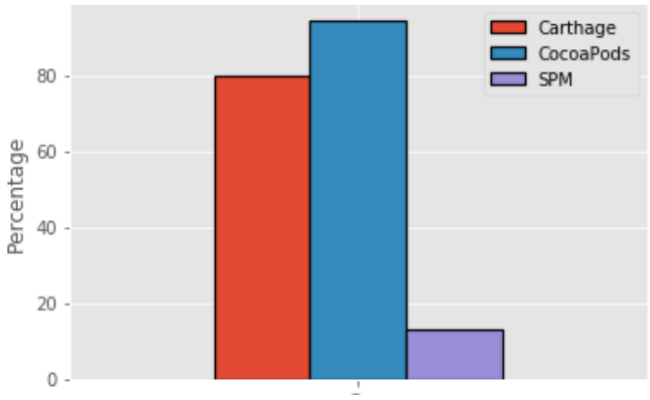

Figure 4: Gerenciadores de pacotes mais utilizados.

4.3. Contribuição

A maioria dos projetos de código aberto crescem e são mantidos através das contribuições de novos desenvolvedores, um guia efetivo de como contribuir para o repositório se torna então necessário.

4.3.1. Existe um guia de contribuição?

Para essa análise buscamos por menções a *contributing*, *contribution*, e *contribute*. Variações esperadas quando se fala sobre contribuições em inglês. Com isso descobrimos que mais da metade dos repositórios, 62.4% não mencionam contribuição. Dentre os que mencionam contribuição, apenas 27.1% tem uma sessão dedicada a contribuições no seu README.

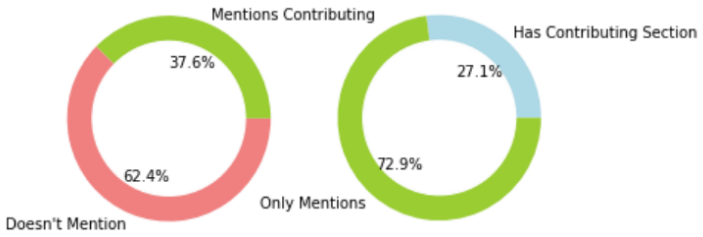

Figure 5: Menções a contribuição.

4.4. Licenciamento / Autoria

Dado que o propósito das bibliotecas é que sejam integradas ao código fonte de aplicações, e visto que aplicativos iOS muitas vezes tem como objetivo monetização, é necessário saber de que forma é licenciado o uso da biblioteca.

4.4.1. Qual licença é utilizada?

Utilizando uma listagem de licenças comuns de projetos de código aberto, buscamos por menções a estas dente os READMEs que mencionam licenciamento. A licença mais utilizada é a MIT, com uma diferença expressiva em relação a outras. Cerca de 7.5% dos repositórios que mencionam licenciamento não tiveram suas licenças identificadas. Análises futuras fazem-se necessárias para identificar o motivo.

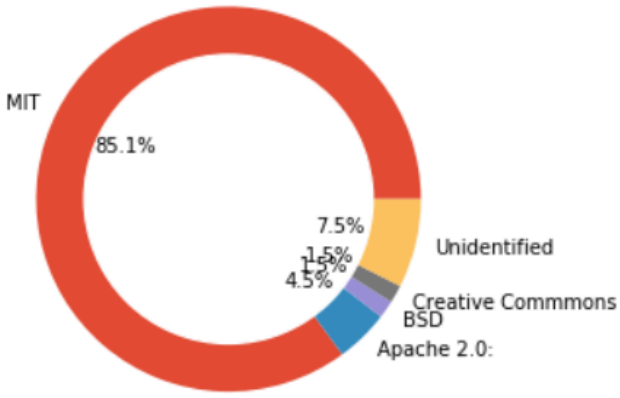

Figure 6: Licenças utilizadas por repositórios que mencionam licenciamento.

4.4.2. Autores tem mais de um repositório popular?

Como vimos na seção 4.1.1, existem autores com mais de um repositório dentre os mais populares, e verificamos que 27% das bibliotecas analisadas pertencem a apenas 5 autores.

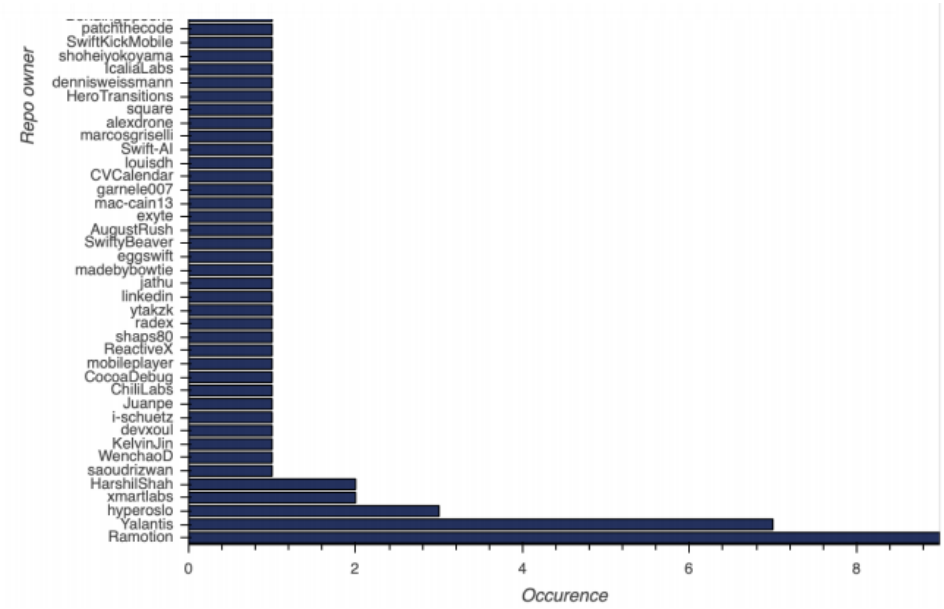

Figure 7: Quantos repostórios populares cada autor tem.

Se destacando entre os autores estão Ramotion, com nove bibliotecas, e Yalantis com sete. Ambos são contas de empresas que prestam serviços de desenvolvimento de aplicativos, e lançam bibliotecas de código aberto dos componentes que utilizam no desenvolvimento dos seus produtos.

4.5. Status (build/ integração contínua)

Uma informação que pode salvar tempo do usuário de bibliotecas, é saber se o status de *build* da mesma, pois evita que seja realizado o processo de instalação, para depois descobrir a informação.

A sinalização de um status de *build* mesmo que negativo, pode servir de indicativo que existe algum problema com o código, e chamar a atenção dos contribuidores para a necessidade de correções.

4.5.1. O status de build está presente?

Consideramos que as informações de de status de *build* devem refletir o estado atual do código-fonte, e mudar dinamicamente de acordo com o mesmo. Por isso restringimos nossa análise a bibliotecas que continham *badges* de *build* provindas de ferramentas de integração contínua.

Por questões de viabilidade técnica focamos na plataforma Travis CI, e descobrimos então que 49.41% dos READMEs possuem *badges* de status de *build* provindos desta plataforma.

4.5.2. O build está passando?

Coletando os *badges* de acordo com a URL embutida nos READMEs, podemos perceber que dos 42 coletados, 19% estão falhando, e que em dois casos não foi coneguida a informação através do *badge*.

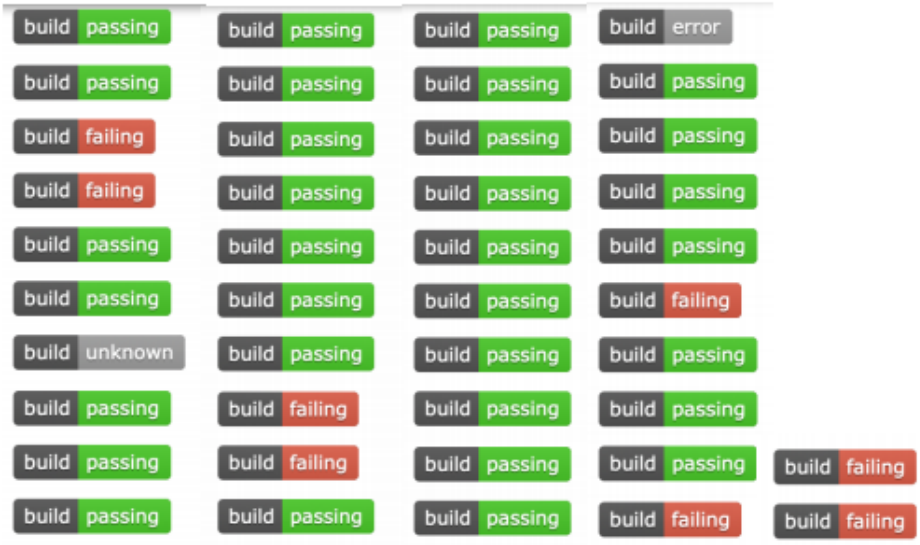

Figure 8: *Badges* de status de *build* do Travis CI.

5. Conclusões e Trabalhos Futuros

Neste trabalho foram realizadas parte das análises levantadas como relevantes pelos desenvolvedores iOS entrevistados. As análises apresentadas demonstram o potencial contido na análise sistemática de READMEs.

Propomos a realização de entrevistas com mais desenvolvedores, e autores de bibliotecas de código aberto, de forma a validar quais informações são de fato relevantes. Podendo então realizar novas análises mais profundas, que combinem forma e conteúdo dos arquivos, a fim de extrair um guia de boas práticas para escrita de READMEs.