# Documentação Automatizada de APIs com Tutoriais Gerados a Partir do Stack Overflow

**Adriano Mendonça Rocha**


Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2016

**Adriano Mendonça Rocha**

# Documentação Automatizada de APIs com Tutoriais Gerados a Partir do Stack Overflow

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Engenharia de Software

Orientador: Prof. Dr. Marcelo de Almeida Maia

Uberlândia
2016

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

R672d
2016

Rocha, Adriano Mendonça, 1986-
Documentação automatizada de APIs com tutoriais gerados a partir do Stack Overflow / Adriano Mendonça Rocha. - 2016.
79 f. : il.

Orientador: Marcelo de Almeida Maia.
Dissertação (mestrado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação.
Inclui bibliografia.

1. Computação - Teses. 2. Software - Desenvolvimento - Teses. 3. Interface de programação de aplicações - Teses. 4. Software de aplicação - Teses. I. Maia, Marcelo de Almeida. II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU: 681.3

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Os abaixo assinados, por meio deste, certificam que leram e recomendam para a Faculdade de Computação a aceitação da dissertação intitulada **"Documentação Automatizada de APIs com Tutoriais Gerados a Partir do Stack Overflow"** por **Adriano Mendonça Rocha** como parte dos requisitos exigidos para a obtenção do título de **Mestre em Ciência da Computação.**

Uberlândia, 5 de agosto de 2016

Orientador: _____
Prof. Dr. Marcelo de Almeida Maia
Universidade Federal de Uberlândia

Banca Examinadora:

_____
Prof. Dr. Marcelo Keese Albertini
Universidade Federal de Uberlândia

_____
Prof. Dr. Marco Tulio Valente
Universidade Federal de Minas Gerais

*Este trabalho é dedicado ao meu pai Iermak (in memoriam), à minha esposa Carolina e ao meu filho Luiz Felipe.*

# Agradecimentos

Primeiramente agradeço a Deus por ter me dado saúde e sabedoria para desenvolver este trabalho.

À minha esposa Carolina pela força, apoio e compreensão que depositou em mim durante esta jornada.

Ao meu orientador Prof. Dr. Marcelo de Almeida Maia, por ter acreditado no meu potencial em desenvolver este trabalho.

*"A persistência é o caminho do êxito."*
*(Charles Chaplin)*

# Resumo

Uma das maneiras mais comuns de reuso de software é por meio de APIs. Porém, um dos principais desafios para o uso efetivo de uma API é o acesso a uma documentação de fácil compreensão. Vários trabalhos propuseram alternativas para tornar a documentação de APIs mais compreensível, ou até mais detalhada. Entretanto, estes trabalhos ainda não levaram em consideração a complexidade de entendimento dos exemplos para tornar estas documentações adaptáveis a diferentes níveis de experiência de desenvolvedores. Neste trabalho desenvolvemos e avaliamos quatro metodologias diferentes para gerar tutoriais para APIs a partir do conteúdo do Stack Overflow e organizá-los conforme a complexidade de entendimento. As metodologias foram avaliadas por meio de tutoriais gerados para a API Swing. Foi conduzido um *survey* para avaliar oito diferentes características dos tutoriais gerados. O resultado geral da avaliação dos tutoriais foi positivo nas diversas características, mostrando a viabilidade para o uso de tutoriais gerados automaticamente. Além disso, o uso de critérios para apresentação dos elementos do tutorial por ordem de complexidade, a separação do tutorial em partes básica e avançada, a natureza de tutorial para os *posts* selecionados e existência de código-fonte didático tiveram resultados significantemente diferentes em relação à metodologia de geração escolhida. Um segundo estudo comparou o uso da documentação oficial da API Android com o uso do tutorial proposto neste trabalho. Foi realizado um experimento controlado com alunos que tiveram um primeiro contato com o desenvolvimento Android, onde estes desenvolveram duas tarefas básicas de programação. Os resultados deste experimento mostraram que na maioria dos casos, os alunos tiveram melhores desempenhos nas tarefas quando utilizaram o tutorial proposto. Os principais motivos do baixo desempenho dos alunos nas tarefas utilizando a documentação oficial da API foram devido à falta de exemplos de uso nesta documentação, além de sua difícil utilização.

**Palavras-chave:** Redocumentação de APIs. Mineração de Repositórios. Stack Overflow.

# Abstract

One of the most common forms of reuse is through API usage. However, one of the main challenges to effective usage is an accessible and easy to understand documentation. Several papers have proposed alternatives to make more understandable API documentation, or even more detailed. However, these studies have not taken into account the complexity of understanding of the examples to make these documentations adaptable to different levels of experience of developers. In this work we developed and evaluated four different methodologies to generate tutorials for APIs from the contents of Stack Overflow and organizing them according to the complexity of understanding. The methodologies were evaluated through tutorials generated for the Swing API. A survey was conducted to evaluate eight different features of the generated tutorials. The overall outcome of the tutorials was positive on several characteristics, showing the feasibility of the use of tutorials generated automatically. In addition, the use of criteria for presentation of tutorial elements in order of complexity, the separation of the tutorial in basic and advanced parts, the nature of tutorial to the selected posts and existence of didactic source had significantly different results regarding a chosen generation methodology. A second study compared the official documentation of the Android API and tutorial generated by the best methodology of the previous study. A controlled experiment was conducted with students who had a first contact with the Android development. In the experiment these students developed two tasks, one using the official documentation of Android and using the generated tutorial. The results of this experiment showed that in most cases, the students had the best performance in tasks when they used the tutorial proposed in this work. The main reasons for the poor performance of students in tasks using the official API documentation were due to lack of usage examples, as well as its difficult use.

**Keywords:** API Redocumentation. Repositories Mining. Stack Overflow.

# Lista de ilustrações

# Lista de tabelas

# Sumário

# Capítulo 1

# Introdução

O reuso de software tem sido indicado durante os processos de desenvolvimento e manutenção de software, pois oferece benefícios tais como: redução do tempo e custos no desenvolvimento; produtos de melhor qualidade e confiabilidade; uso eficaz de especialistas; etc. (SOMMERVILLE, 2007). Uma das maneiras mais comuns de reuso é por meio de APIs (*Application Programming Interfaces*), que é um conjunto de padrões de programação que permite aos desenvolvedores utilizarem suas funcionalidades sem a necessidade de entender seus detalhes de implementação. Porém durante atividades de reuso de software, desenvolvedores geralmente encontram dificuldades para compreender a documentação da funcionalidade que se deseja reutilizar (SUBRAMANIAN; INOZEMTSEVA; HOLMES, 2014). Atualmente existem várias APIs que podem ser utilizadas durante o desenvolvimento de software, mas geralmente os desenvolvedores têm dificuldades ao utilizarem essas APIs por falta de uma documentação de fácil compreensão (ROBILLARD, 2009), (XIE; PEI, 2006). Uma das alternativas para buscar um melhor entendimento sobre uma dada API é recorrer a serviços de busca na Internet por conteúdo que possa ajudar o desenvolvedor a entender as funcionalidades da API de forma intuitiva (trechos simples de código que explique de forma clara uma certa funcionalidade) (PARNIN et al., 2012).

O Stack Overflow (SO) é um *site* de perguntas e respostas (*posts*) relacionadas ao desenvolvimento de software onde muitos desenvolvedores buscam informações que possam auxiliá-los durante o processo de desenvolvimento ou manutenção de software. Os desenvolvedores podem enviar dúvidas relacionadas à programação e, geralmente, os mais experientes no assunto enviam melhores respostas, de forma a esclarecer as dúvidas dos desenvolvedores menos experientes no assunto. Todo o conteúdo fica disponível para que essas dúvidas e suas respectivas respostas possam ser acessadas pela comunidade. Treude et al. (TREUDE et al., 2012) mostraram que o SO possui mais de 12 milhões de visitantes e 135 milhões de visualizações de páginas todo mês, isto faz com que este *site* seja uma fonte importante de informações que os desenvolvedores podem utilizar no processo de desenvolvimento ou manutenção de software. Os desenvolvedores têm acessado o SO em busca de informações que possam melhorar o entendimento das APIs. Um

problema destes *sites* é que possuem um grande volume de informação, dificultando o acesso ao conteúdo que realmente ajude os desenvolvedores a esclarecerem suas dúvidas (PONZANELLI; BACCHELLI; LANZA, 2013). O SO oferece mecanismos de busca, tais como: barra de pesquisa; filtragem por *tags*, usuários, entre outros. Estes mecanismos auxiliam os desenvolvedores a encontrarem suas respostas, mas mesmo assim ainda retornam muito conteúdo, e no caso da barra de pesquisa, dependendo da formulação da pesquisa feita pelo usuário, esta pode excluir *posts* que poderiam ajudar no entendimento da API. Vários trabalhos (HENB; MONPERRUS; MEZINI, 2012), (KIM et al., 2009), (MONTANDON et al., 2013), (SOUZA; CAMPOS; MAIA, 2014a), (STYLOS; MYERS; YANG, 2009) foram propostos para tornar a documentação de APIs mais compreensível, ou até mais detalhada, entretanto ainda não têm levado em consideração a complexidade de entendimento dos exemplos destas documentações, podendo o desenvolvedor perder um tempo considerável até encontrar a informação desejada.

Souza et al. (SOUZA; CAMPOS; MAIA, 2014a) propuseram uma abordagem para encontrar temas de uma certa API no SO e, através destes temas, construir um livro de receitas para a API. Esta abordagem é baseada na filtragem do conteúdo do SO, selecionando tópicos específicos através de um classificador para questões do tipo (*"How-to-do"*)(CAMPOS; MAIA, 2014),(SOUZA; CAMPOS; MAIA, 2014b). Esta abordagem não leva em consideração a complexidade de entendimento dos exemplos do livro de receitas, ou seja, pode acontecer de um capítulo do livro de receitas seja formado apenas com exemplos de difícil compreensão. Outro problema é que não há uma separação entre capítulos mais complexos e capítulos mais básicos, de caráter introdutório.

Para solucionar estas limitações, neste trabalho foram desenvolvidas e analisadas quatro diferentes metodologias para filtrar os *posts* do SO e organizá-los conforme a complexidade de entendimento. Desta forma, um desenvolvedor ainda inexperiente com uma certa API, pode começar com a leitura de um *post* mais simples (complexidade de entendimento menor) e, depois de entender estes mais simples, pode continuar com outros de complexidade mais alta.

O principal objetivo deste trabalho é construir um tutorial para uma certa API a partir de informações presentes no SO, com exemplos (*posts* do SO) que possuem grau de complexidade diferente, dos mais simples aos mais complexos.

O resto desta dissertação está organizado como a seguir. No Capítulo 2 é apresentado o ferramental utilizado, onde serão abordadas as principais ferramentas utilizadas neste trabalho, tais como: Mallet (*Machine Learning for Language Toolkit*) e LambdaMART. O Capítulo 3 apresenta as quatro metodologias propostas neste trabalho para geração de tutoriais a partir do conteúdo do SO. No Capítulo 4 é apresentado o procedimento adotado na avaliação e comparação destas metodologias, além de apresentar o procedimento para a comparação de um tutorial gerado por uma das metodologias propostas com a documentação oficial para uma dada API. No Capítulo 5 são apresentados e discutidos

os resultados obtidos. O Capítulo 6 apresenta os trabalhos relacionados e no Capítulo 7 é apresentada a conclusão da dissertação.

# Capítulo 2

# Ferramental Utilizado

Nesse capítulo serão apresentadas as principais ferramentas utilizadas no trabalho: Mallet (*Machine Learning for Language Toolkit*) e LambdaMART.

## 2.1 Mallet

Mallet é uma ferramenta para o processamento de linguagem natural estatística, classificação de documentos, clusterização, modelagem em tópicos, extração de informações e outras aplicações de aprendizado de máquina para texto (MCCALLUM, 2016). Neste trabalho o Mallet foi utilizado para a classificação de documentos e a modelagem em tópicos.

### 2.1.1 Classificador de Documentos

Um classificador de documentos é um algoritmo que faz a rotulagem dos documentos para um conjunto fixo de classes, tais como, "tutorial" *versus* "não-tutorial", com base em um conjunto de exemplos de treinamento. O Mallet possui implementações de vários algoritmos de classificação, como: Naïve Bayes, Máxima Entropia e Árvores de Decisão. Além disso, o Mallet fornece ferramentas para avaliar os classificadores (MCCALLUM, 2016).

### 2.1.2 Modelagem em Tópicos

A modelagem em tópicos fornece uma maneira simples de analisar grandes volumes de texto e organizar os documentos em tópicos. Um tópico consiste em um conjunto de palavras que frequentemente ocorrem em conjunto em um texto (MCCALLUM, 2016). Segundo (STEYVERS; GRIFFITHS, 2007), a modelagem em tópicos é baseada na ideia de que os documentos são misturas de tópicos, em que um tópico é uma distribuição de probabilidade sobre palavras.

Esta ferramenta tem como entrada um conjunto de documentos (ou *corpus*) e procura por padrões, extraindo tópicos a partir do corpus. Neste trabalho, um documento é formado pelo conteúdo textual obtido em um dado *post* do Stack Overflow.

Um exemplo de utilização da ferramenta de modelagem em tópicos é mostrado a seguir. Foi dado como entrada 10 documentos (corpus) contendo informações sobre os temas: Alan Turing, Automobile, Beethoven, Biology, City, Da Vinci, Einstein, Orange, War e Water. Logo após o processamento destes documentos, a ferramenta gerou os seguintes tópicos:

- 0 water earth food billion people air land produce world precipitation vapor hydrogen runoff agriculture irrigation access safe transpiration evaporation
- 1 einstein theory physics years school paper maric relativity albert wrote father zurich professor called german teaching year papers moved
- 2 city cities food population flaherty trade large scale cost density pre town development higher protection square bairoch output neolithic
- 3 biology life organisms study natural evolutionary basic examines cell evolution human studied term bc der molecular theory modern work
- 4 beethoven born johann family court bonn began age mother german piano van march published life years birth including father
- 5 turing work computer alan machine time park bletchley bombe enigma british death computing government program manchester worked design working
- 6 orange citrus sweet oranges fruit trees tree grown seeds cultivars cultivated mandarin bitter sinensis china introduced spanish south green
- 7 war world warfare societies needed united number century men states human violence history statue people conflicts past raids conflict
- 8 benz engine daimler automobile built powered car company maybach patent automobiles model dmg combustion steam vehicles vehicle internal design
- 9 leonardo nb vinci verrocchio piero milan ludovico ser painting da father workshop painted florence renaissance di life vasari medici

Se analisarmos cada tópico visualmente podemos relacioná-los a cada tema. Por exemplo, o tópico 0 está relacionado com Water, pois apareceram vários termos relacionados com água neste tópico, tais como, water, vapor, hydrogen, irrigatioan, evaporation, transpiration, etc. O tópico 1 está relacionado com Einstein, pois aparecem os termos: einstein, theory, physics, relativity, albert, professor, german, etc. Dados os temas e os tópicos gerados, facilmente uma pessoa relacionaria cada tema a um tópico. Nesse caso é fácil

analisar, pois temos apenas 10 documentos e 10 tópicos, quando se trata de analisar vários documentos, como é o caso deste trabalho, já não é mais viável analisar os resultados visualmente.

A ferramenta de modelagem em tópicos tem uma função que gera uma tabela contendo os tópicos mais influentes para cada documento que foi dado como entrada. A Tabela 1 mostra o resultado desta função para o exemplo dos 10 temas analisados nos parágrafos anteriores.

A primeira coluna da Tabela 1 indica o número do documento que foi dado como entrada à ferramenta de modelagem em tópicos. Um conjunto de documentos pode ser dado como entrada ao mesmo tempo, basta indicar à ferramenta o diretório onde o conjunto de documentos está localizado.

A segunda coluna da Tabela 1 indica o diretório juntamente com o nome do documento que foi dado como entrada à ferramenta, por exemplo, o documento contendo as informações do tema Einstein é encontrado em: file:/C:/mallet/entrada_dados/05/Einstein.txt.

A terceira coluna da Tabela 1 indica o tópico mais influente em cada documento que foi dado como entrada. Por exemplo, para o documento contendo as informações sobre o tema Alan Turing o tópico 5 é o mais influente, a próxima coluna da tabela contém um valor que representa o quanto este tópico é influente, se multiplicarmos este valor por 100, vamos obter a porcentagem de influência do tópico, neste caso 68,03%.

A quinta coluna da Tabela 1 indica o segundo tópico mais influente em cada documento que foi dado como entrada. Por exemplo, para o documento que contém as informações sobre o tema Alan Turing o tópico 1 é o segundo mais influente, a próxima coluna da tabela contém um valor que representa o quanto este tópico é influente, se multiplicarmos este valor por 100, vamos obter a porcentagem de influência do tópico, neste caso 11,53%.

A ferramenta de modelagem em tópicos fornece o terceiro, o quarto, o quinto, o sexto, o sétimo, o oitavo, o nono e o décimo tópico mais influente. Por simplificação a Tabela 1 indica somente o primeiro e o segundo tópico mais influente. A quantidade de tópicos que aparecem na tabela gerada pela ferramenta de modelagem em tópicos depende do número de tópicos definido pelo usuário antes da execução da ferramenta. Se a ferramenta fosse configurada para gerar 20 tópicos, então apareceriam na tabela os 20 tópicos com seus respectivos valores de influência para cada documento.

## 2.2 Ranqueador LambdaMART

Neste trabalho foi utilizado o algoritmo de ranqueamento LambdaMART que é uma versão *boosted tree* do LambdaRank, que é baseado no RankNet. Os algoritmos RankNet, LambdaRank e LambdaMART têm provado serem muito bem sucedidos na resolução de problemas de ranqueamento do mundo real (BURGES, 2010). Estes algoritmos também são utilizados em desafios de aprendizagem para ranquear (CHAPELLE; CHANG; LIU,

Tabela 1 – Influência dos tópicos em cada documento.

| **Nº** | **documento** | **1º tópico mais influente** | **Valor de influência do 1º tópico mais influente.** | **2º tópico mais influente** | **Valor de influência do 2º tópico mais influente.** |
|---|---|---|---|---|---|
| 0 | file:/C:/mallet/entrada/Alan%20Turing.txt | 5 | 0.6806... | 1 | 0.1153... |
| 1 | file:/C:/mallet/entrada/Automobile.txt | 8 | 0.9974... | 4 | 0.0011... |
| 2 | file:/C:/mallet/entrada/Beethoven.txt | 4 | 0.9994... | 2 | 0.0000... |
| 3 | file:/C:/mallet/entrada/Biology.txt | 3 | 0.9994... | 4 | 0.0000... |
| 4 | file:/C:/mallet/entrada/City.txt | 2 | 0.9994... | 4 | 0.0000... |
| 5 | file:/C:/mallet/entrada/Da%20Vinci.txt | 9 | 0.9385... | 4 | 0.0473... |
| 6 | file:/C:/mallet/entrada/Einstein.txt | 1 | 0.9181... | 4 | 0.0815... |
| 7 | file:/C:/mallet/entrada/Orange.txt | 6 | 0.9993... | 4 | 0.0000... |
| 8 | file:/C:/mallet/entrada/War.txt | 7 | 0.9154... | 2 | 0.0631... |
| 9 | file:/C:/mallet/entrada/Water.txt | 0 | 0.9577... | 2 | 0.0379... |

2011). LambdaMART é significativamente mais rápido nas fases de treinamento e de teste do que outros algoritmos do estado da arte (WU et al., 2010).

Foi utilizado neste trabalho a biblioteca RankLib que contém os algoritmos mais populares de aprendizagem para ranquear, incluindo o algoritmo LambdaMART. Para treinar este algoritmo primeiramente é necessário definir os atributos que serão dados como entrada. Um atributo é uma característica do elemento a ser ranqueado, por exemplo, se considerarmos os *posts* do Stack Overflow como elementos a serem ranqueados, poderíamos definir como características: a quantidade de linhas, a sua pontuação, quantidade de linhas de código-fonte, etc. Uma vez definido os atributos, o próximo passo é elaborar o conjunto de treinamento para o algoritmo. Para elaborar este conjunto é necessário obter exemplos de elementos ranqueados juntamente com seus atributos. Para treinar o algoritmo são dados os atributos do elemento e sua ordem. Os valores dos atributos devem ser representados através de número real, variando entre 0 e 1. A Tabela 2 ilustra um exemplo de conjunto de treinamento com M elementos com seus respectivos atributos e suas respectivas ordens.

Tabela 2 – Exemplo de um conjunto de treinamento para o algoritmo de ranqueamento.

| **Ordem** | **Atributo 1** | **Atributo 2** | **Atributo 3** | **...** | **Atributo N** |
|---|---|---|---|---|---|
| 1 | 0.13 | 0.00 | 0.05 | ... | 0.08 |
| 2 | 0.25 | 0.10 | 0.00 | ... | 0.43 |
| 3 | 0.41 | 0.28 | 0.17 | ... | 0.34 |
| ... | ... | ... | ... | ... | ... |
| M | 0.68 | 0.37 | 0.23 | ... | 0.59 |

Depois de treinado o algoritmo, basta executá-lo sobre os elementos que se deseja ranquear.

Capítulo 3

# A Abordagem de Construção

Neste trabalho foram desenvolvidas quatro metodologias diferentes para gerar tutoriais para uma dada API. Para cada metodologia foi dado um nome e uma sigla conforme suas principais características. A Tabela 3 mostra as principais características de cada metodologia com suas respectivas siglas.

Tabela 3 – Principais características de filtragem das metodologias e suas siglas.

| **Sigla** | **Características** |
|---|---|
| FHCBAC | **F**iltros + ***H****ow to* + **C**omplexidade + Divisão **B**ásico e **A**vançado + **C**obertura API. |
| GP | **G**oogle **P**uro. |
| GFC | **G**oogle + **F**iltros + **C**omplexidade. |
| GFHCBAC | **G**oogle + **F**iltros + ***H****ow to* + **C**omplexidade + Divisão **B**ásico e **A**vançado + **C**obertura API. |

As metodologias propostas implementaram diferentes características para permitir a comparação do benefício entre incluir ou não a característica. Por exemplo, na metodologia GP não foram implementados os filtros propostos e nem o algoritmo de ranqueamento por complexidade para avaliar a falta que estes mecanismos podem fazer.

Para cada metodologia foi desenvolvida uma sequência de etapas para construir um tutorial para uma dada API, a partir do banco de dados disponível do Stack Overflow.

Na Metodologia FHCBAC são aplicados vários filtros desenvolvidos neste trabalho e algoritmos já estabelecidos tais como: classificador de textos *Mallet* para filtrar os posts do tipo *"How-to-do"*, LDA (*Latent Dirichlet Allocation*) - modelagem de tópicos para gerar os capítulos do tutorial e LambdaMart para ranquear os *posts* conforme sua complexidade.

Na Metodologia GP utiliza-se o LDA para definir os capítulos do tutorial para uma

certa API, depois é utilizado o mecanismo de pesquisa do Google com as palavras chaves de cada capítulo, e os $n$ primeiros resultados ranqueados pelo Google são utilizados para montar cada capítulo do tutorial.

A Metodologia GFC é híbrida, pois combina etapas da Metodologia FHCBAC com etapas da Metodologia GP. Nesta metodologia é utilizado o LDA para definir os capítulos do tutorial para uma certa API, depois é utilizado o mecanismo de pesquisa do Google com as palavras chaves de cada capítulo (etapas da Metodologia GP), em seguida são aplicados os filtros e algoritmos desenvolvidos na Metodologia FHCBAC, e finalmente é montado o tutorial.

A Metodologia GFHCBAC também é híbrida, pois combina etapas da Metodologia FHCBAC com etapas da Metodologia GP. As diferenças são as seguintes: a primeira etapa consiste em gerar o tutorial para uma certa API utilizando a Metodologia FHCBAC; aplica-se o mecanismo de pesquisa do Google com as palavras chaves de cada capítulo combinadas com a palavra *"how to create"* filtrando apenas os resultados do site *"stackoverflow.com"*, a fim de encontrar os *posts* relacionados à "criação" de objetos, por exemplo, em um capítulo relacionado à tabelas um *post* de criação interessante seria *"How to instantiate an empty table?"*. Em seguida são aplicados os filtros e os algoritmos desenvolvidos na Metodologia FHCBAC nos *posts* retornados pelo Google e depois eles são incluídos no tutorial.

## 3.1 Metodologia FHCBAC

A seguir serão detalhadas as principais etapas da Metodologia FHCBAC para a geração de um tutorial para uma dada API "X", conforme a Figura 1.

**Etapa 1:** Filtro 1 - Seleção dos *posts* que possuem a *tag* da API "X" e que possuem resposta aceita pelo usuário no banco de dados do Stack Overflow.

**Etapa 2:** Filtro 2 - Aplicação do classificador Mallet nos *posts* filtrados na Etapa 1 a fim de selecionar apenas os *posts* do tipo *"How-to-do"*. O objetivo desta etapa é identificar entre os *posts* previamente selecionados aqueles que possuem um tipo especial de questão (*"How-to-do"*), pois apenas esse tipo de pergunta é adequado para a construção de um tutorial, uma vez que está relacionada em como implementar funcionalidades no *software* utilizando uma tecnologia ou uma API.

Existem outros tipos de questões propostas em (NASEHI et al., 2012) que não estão relacionadas com tutoriais e que não deveriam ser consideradas, como: *Debug/corrective, Need-to-know* e *Seeking-different-solution.*

- ***Debug/corrective:*** Está relacionado com problemas no código em desenvolvimento, como: erros de compilação, comportamento inesperado e erros em tempo de execução.

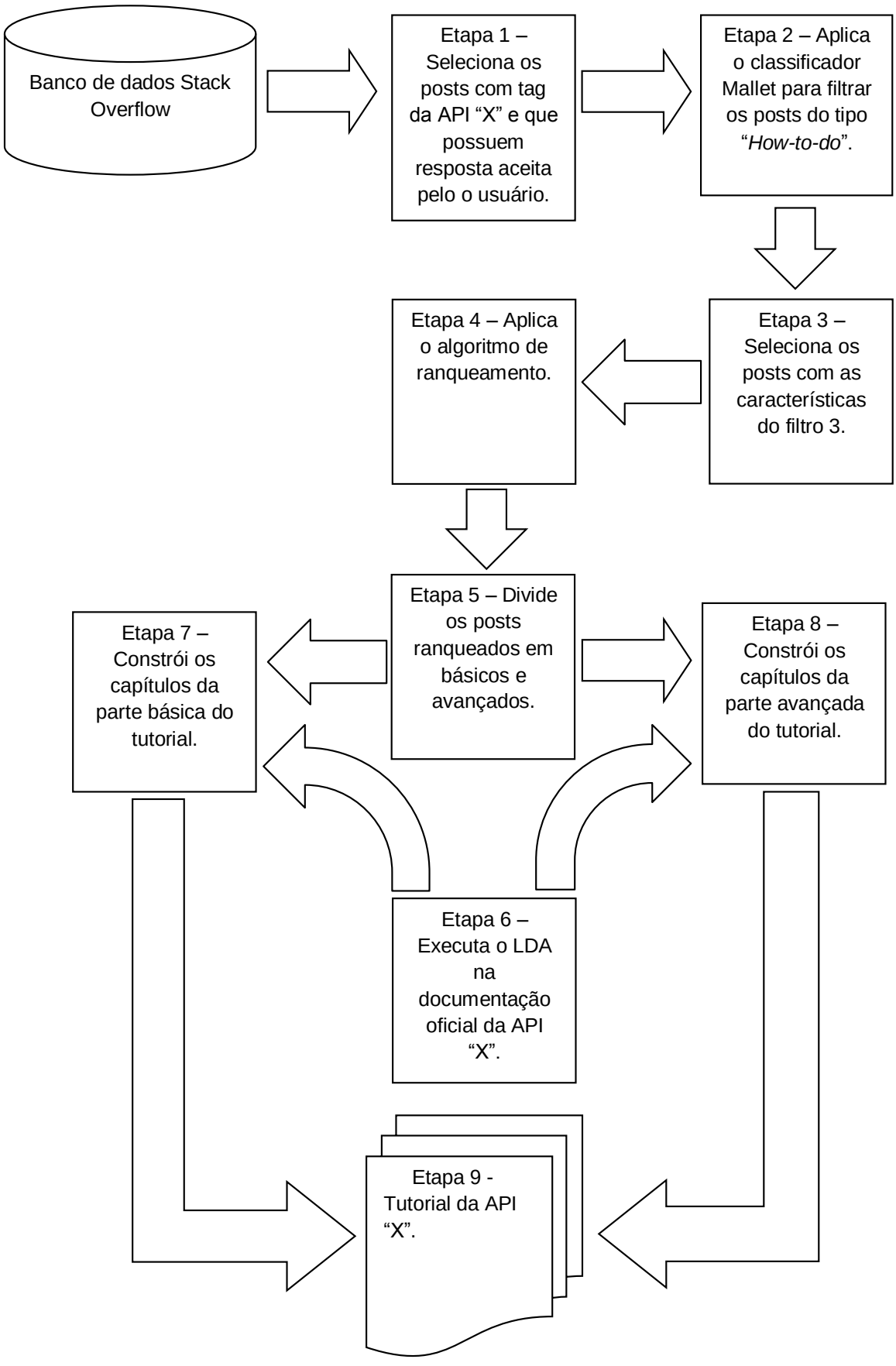


Figura 1 – Principais etapas para gerar um tutorial para API "X", utilizando a metodologia FHCBAC.

- ***Need-to-know:*** Está relacionado a perguntas que dizem respeito à possibilidade ou disponibilidade de fazer algo, por exemplo, a presença de uma funcionalidade em uma API ou em uma linguagem de programação.

- ***Seeking-different-solution:*** Está relacionado a desenvolvedores que têm um código que está funcionando, mas procuram outras soluções para fazer a mesma tarefa.

O classificador do Mallet foi treinado utilizando um conjunto de dados de treinamento com 100 exemplos aleatórios de *posts* do tipo *"How-to-do"* e 100 exemplos aleatórios envolvendo *posts "Debug/corrective", "Need-to-know"* e *"Seeking-different-solution"*, conforme a classificação proposta por Nasehi et al. (NASEHI et al., 2012). Os *posts* do tipo *"How-to-do"* foram designados como grupo "Tutorial" e os demais tipos de *posts* foram designados como grupo "Não Tutorial". Para medir a acurácia do classificador após treinamento e teste, foi dado mais 100 exemplos aleatórios de *posts* do tipo *"How-to-do"* e mais 100 exemplos aleatórios envolvendo *posts "Debug/corrective", "Need-to-know"* e *"Seeking-different-solution"*, estes exemplos são diferentes dos exemplos usados na fase de treinamento e teste. Dentre os 200 exemplos, 187 foram classificados corretamente pelo classificador, obtendo uma acurácia de 93,5%.

Depois de treinado, o classificador é aplicado em todos os *posts* de uma determinada API. Sendo assim, espera-se que os posts classificados como "Tutorial" sejam do tipo *"How-to-do"* e portanto com uma natureza apropriada para participar de um tutorial.

**Etapa 3:** Filtro 3 - Seleção dos *posts* com as seguintes características:

1. O código fonte da resposta deve cobrir pelo menos um tipo da API (Classes, Interfaces, etc.). Considera-se que o código da resposta deve ilustrar como usar algum elemento da API.

2. No corpo da pergunta deve ter no máximo três linhas de código-fonte. Normalmente, perguntas com código extenso, estão solicitando ajuda acerca de algum problema no respectivo código.

3. No código da resposta deve ter no mínimo três linhas de código-fonte. Considera-se que três linhas seja um tamanho mínimo para que uma ilustração de uso da API seja significativa.

**Etapa 4:** Aplicação do algoritmo de ranqueamento LambdaMART nos *posts*. Nesta etapa, coleta-se um conjunto de atributos em cada *post*.

Foram definidos os seguintes atributos:

1. Quantidade de **if**s no código-fonte do post;

2. Quantidade de **for**s no código-fonte do post;

3. Quantidade de **while**s no código-fonte do post;

4. Quantidade de **switch**s no código-fonte do post;

5. Quantidade de linhas de código-fonte;

6. Pontuação do *post* no SO.

Depois de coletados os atributos dos *posts*, define-se um conjunto de dados de treinamento que será utilizado pelo algoritmo de ranqueamento. Este conjunto será formado por *posts* que possuem diferentes níveis de complexidade, conforme os valores de seus atributos. A seleção do conjunto de *posts* de treinamento é feita de maneira manual e, em seguida, é gerada uma matriz onde as linhas serão formadas pelos *posts* selecionados e as colunas serão formadas pelos valores de seus atributos. Em seguida, os *posts* são ranqueados de forma manual, atribuindo um valor para cada *post* conforme os valores dos seus atributos, por exemplo, um *post* X em que há várias estruturas de repetição é mais complexo do que um *post* Y que não tem estruturas de repetição, desta forma o valor atribuído ao *post* X é maior do que o atribuído a Y. O atributo *Pontuação do post no SO* é considerado para fins de seleção de *posts* de qualidade, este atributo não influencia na complexidade do *post*. Uma vez treinado o ranqueador, o próximo passo é ranquear todos *posts* filtrados.

**Etapa 5:** Divisão dos *posts* ranqueados em básicos e avançados. Seja $n$ o número de *posts* ranqueados. Define-se que os *posts* básicos são formados pelos os *posts* ranqueados da posição 1 até a posição $n/2$, e os *posts* avançados são formados pelos os *posts* ranqueados da posição $(n/2 + 1)$ até a posição $n$.

**Etapa 6:** Execução do algoritmo de modelagem de tópicos (LDA) na documentação oficial da API "X" para gerar os tópicos que representarão os capítulos do tutorial. É necessário definir um número de tópicos onde não haja muita repetição de assuntos nos tópicos e que abordem o máximo de assuntos da API. Para isto foi proposto o Algoritmo 1.

Na linha 1 do algoritmo a variável **numeroTopicos** recebe a quantidade inicial de tópicos que o LDA gerará, esta quantidade é informada pelo usuário. Nas linhas 2 e 3 são definidas duas variáveis de controle, **executarLDA** e **aumentarNumeroTopicos**, inicializadas com o valor "sim". Na linha 4 é definido a estrutura de repetição **while**, que repetirá enquanto a variável de controle **executarLDA** for igual a "sim". Na linha 5 a variável de controle **executarLDA** recebe o valor "nao", caso as condições das linhas 9 e 14 não sejam satisfeitas, a estrutura **while** da linha 4 é encerrada. Na linha 6 a estrutura de dados **topicos** recebe os tópicos gerados pelo LDA, a quantidade é definida pelo parâmetro **numeroTopicos** passado para a função **executaLDA()**. Na linha 7 a estrutura de dados **termosComunsParesTops** recebe o valor da quantidade de termos em comum para todos os pares de tópicos armazenados na estrutura de dados **topicos**. Por exemplo, a quantidade de termos em comuns dos pares tópico 1 e tópico 2, é armazenado em **termosComunsParesTops[1]**, a quantidade de termos em comuns dos pares

**Algoritmo 1** Algoritmo para encontrar um número adequado de tópicos para o LDA

```
1: numeroTopicos ← leiaNumeroTopicos();
2: executarLDA ← "sim";
3: aumentarNumeroTopicos ← "sim";
4: while executarLDA = "sim" do
5:    executarLDA ← "nao";
6:    topicos ← executaLDA(numeroTopicos);
7:    termosComunsParesTops ← calculeQuantTermosComunsPares(topicos);
8:    for i ← 1 to tamanho(termosComunsParesTops); do
9:       if termosComunsParesTops[i] ≥ 3 then
10:         numeroTopicos ← numeroTopicos − 1;
11:         executarLDA ← "sim";
12:         aumentarNumeroTopicos ← "nao";
13:      end if
14:      if aumentarNumeroTopicos = "sim" then
15:         numeroTopicos ← numeroTopicos + 10;
16:         executarLDA ← "sim";
17:      end if
18:   end for
19: end while
20: return numeroTopicos;
```

tópico 1 e tópico 3, é armazenado em **termosComunsParesTops[2]**, e assim por diante. Na linha 8 é definido a estrutura de repetição **for** que repetirá de 1 até a quantidade de elementos armazenados na estrutura de dados **termosComunsParesTops**. Na linha 9 é verificado através do comando **if** se a quantidade de termos em comuns dos pares de tópicos armazenados em **termosComunsParesTops[i]** é maior ou igual a 3. Este valor 3 é um *threshold* que foi definido após vários experimentos. Nos experimentos foi observado que quando há muitos termos em comuns entre os tópicos, há muita repetição de assunto da API entre os tópicos. Por exemplo, em um dos experimentos realizado na documentação oficial da API Swing, onde o LDA gerou 30 tópicos, entre estes, 4 abordavam o assunto "*Component*". Analisando estes 4 tópicos, foi observado que eles continham muitos termos em comuns. No mesmo experimento também foram constados outros pares de tópicos que continham muitos termos em comuns e que abordavam o mesmo tema. Então foi realizado outro experimento onde a quantidade de tópicos foi decrementada para 29, e foi observado que o número de termos em comuns entre os tópicos diminuiu, com consequência a quantidade de tópicos que abordam o mesmo assunto também. Realizando outros experimentos sempre decrementando o número de tópicos, foi observado que se houver dois ou mais pares de tópicos com até 2 termos em comuns é aceitável, pois analisando estes pares de tópicos, foi constatado que a maioria das vezes eles não abordam o mesmo assunto da API. Caso a condição da linha 9 seja satisfeita, então serão executados os comandos nas linhas 10, 11 e 12. Na linha 10 a variável **numeroTopicos** que representa a quantidade de tópicos que o LDA gerará será decrementada. Na linha 11

a variável de controle **executarLDA** recebe o valor "sim", isto faz com que o comando **while** da linha 4 seja executado novamente com o número de tópicos que o LDA gerará decrementado. Na linha 12 a variável de controle **aumentarNumeroTopicos** recebe o valor "nao", isto faz com que o comando da linha 14 não seja executado, pois este comando só é executado se na primeira execução do algoritmo, a condição da linha 9 não seja satisfeita, se isto ocorrer é porque provavelmente o número de tópicos que o LDA gerou não estão repetindo os assuntos da API, e pode acontecer desse número de tópicos não estar abordando os assuntos que um número maior de tópicos abordaria, sem repetir os assuntos, por isto é necessário executar o comando da linha 14, caso o algoritmo não chegue a um ponto que comece a repetir os assuntos da API nos tópicos. É necessário o algoritmo achar este ponto de repetição de assuntos da API nos tópicos, uma vez achado este ponto, então é feito um decremento no número de tópicos até achar um número que não repita assuntos da API e ao mesmo tempo abordem o máximo de assuntos da API possível. Na linha 13 é marcado o fim do escopo do comando **if** da linha 9. Na linha 14 o comando **if** verifica se a variável de controle **aumentarNumeroTopicos** é igual a "sim", caso isto ocorra é porque o algoritmo não achou uma quantidade de tópicos que provavelmente estão repetindo os assuntos da API entre si. Então é necessário aumentar o número de tópicos que o LDA gerará para achar este ponto de repetição de assuntos entre os tópicos. Este aumento no número de tópicos é feito na linha 15, onde é somado 10 ao valor atual da variável **numeroTopicos**. Este valor somado poderia ser qualquer outro valor, pois se o algoritmo não encontrar o ponto de repetição dos assuntos nos tópicos, o comando da linha 15 será executado novamente até que ache este ponto, então sempre será somado a **numeroTopicos** o valor definido na linha 15. Uma vez achado o ponto de repetição o algoritmo vai para etapa que decrementa o número de tópicos, até achar um número onde não há muita repetição de assuntos da API entre os tópicos. Na linha 16 a variável de controle **executarLDA** recebe o valor "sim", isto faz com que o comando **while** da linha 4 seja executado novamente com o número de tópicos que o LDA gerará aumentado. Na linha 17 é marcado o fim do escopo do comando **if** da linha 14. Na linha 18 é marcado o fim do escopo do comando **for** da linha 8. Na linha 19 é marcado o fim do escopo do comando **while** da linha 4. Na linha 20 é retornado o número de tópicos encontrado pelo algoritmo.

Depois de gerados os tópicos pelo o LDA, analisa-se a necessidade de clusterizar aqueles que ainda abordam o mesmo assunto, logo em seguida, o(s) tópico(s) e/ou cluster(s) são analisados para receber um nome. Cada tópico(s) e/ou cluster(s) representará um capítulo no tutorial.

**Etapa 7:** Constrói os capítulos da parte básica do tutorial, conforme o Algoritmo 2.

Na linha 1 do algoritmo é definido a estrutura de repetição **for** que repetirá de 1 até a quantidade de *posts* armazenados na estrutura de dados **postBasico**. Nesta estrutura estão armazenados todos os *posts* básicos obtidos na etapa 5 da metodologia em ordem

**Algoritmo 2** Algoritmo para formar capítulos Básicos do Tutorial

```
 1: for i ← 1 to postBasico.tamanho(); do
 2:    postBasico[i].selecionado ← "nao";
 3: end for
 4: for i ← 1 to capitulos.tamanho(); do
 5:    tiposAPI ← tiposAPICaps(capitulos[i]);
 6:    for j ← 1 to postBasico.tamanho(); do
 7:       for w ← 1 to tiposAPI.tamanho(); do
 8:          if postBasico[j].contemTipo(tiposAPI[w]) then
 9:             tiposAPI.removeTipo(tiposAPI[w]);
10:             if postBasico[j].selecionado = "nao" then
11:                capTutorial[i].incluiPost(postBasico[j]);
12:                postBasico[j].selecionado = "sim";
13:             end if
14:          end if
15:       end for
16:    end for
17: end for
```

crescente de complexidade, ou seja, o **postBasico[i]** é mais simples do que **postBasico[i+1]**. Na linha 2, para cada **postBasico[i]**, é atribuído o valor "nao" para o campo "selecionado". Este campo garante que o *post* será selecionado uma única vez. Na linha 3 é marcado o fim do escopo da estrutura **for** da linha 1. Na linha 4 é definido a estrutura de repetição **for** que repetirá de 1 até a quantidade de capítulos armazenados na estrutura de dados "capitulos". Nesta estrutura estão armazenados todos os capítulos obtidos na etapa 6 da metodologia. Cada capítulo possui os seus respectivos tipos da API "X" que foram clusterizados pelo LDA na etapa 6. Na linha 5 a estrutura de dados **tiposAPI** recebe todos os tipos da API "X" relacionados com o **capitulos[i]**. Na linha 6 a estrutura **for** é responsável por percorrer todos os *posts* básicos armazenados em **postBasico**. Na linha 7 a estrutura **for** é responsável por percorrer todos os tipos da API relacionados com o **capitulos[i]**. Como esta estrutura está dentro da estrutura **for** da linha 6, então para cada *post* percorrido, será percorrido todos os tipos da API armazenados na estrutura **tiposAPI**. Na linha 8 é verificado através do comando **if** se o *post* **postBasico[j]** contém o tipo da API **tiposAPI[w]**, se sim, na linha 9 o tipo da API **tiposAPI[w]** é removido da estrutura de dados **tiposAPI**, pois ele foi coberto pelo o **postBasico[j]**, isto evita que o algoritmo selecione muitos *posts* que cobrem os mesmos tipos da API. Na linha 10 é verificado através do comando **if** se o *post* **postBasico[j]** não foi selecionado, e se sim, então na linha 11 o *post* **postBasico[j]** é incluído no capítulo **capTutorial[i]**, em seguida, na linha 12, o *post* **postBasico[j]** é marcado como já selecionado, isto garante que o algoritmo não selecione *posts* repetidos em outros capítulos. A ideia geral do algoritmo é percorrer todos os *posts* básicos, seguindo a ordem de complexidade, dos mais simples para os mais complexos, e para cada *post*, é verificado se ele cobre pelo menos um tipo da

API, se sim, então ele é incluído no capítulo. Isto é feito para todos os capítulos obtidos na etapa 6 da metodologia. Na linha 13 é marcado o fim do escopo do comando **if** da linha 10. Na linha 14 é marcado o fim do escopo do comando **if** da linha 8. Na linha 15 é marcado o fim do escopo do comando **for** da linha 7. Na linha 16 é marcado o fim do escopo do comando **for** da linha 6. Na linha 17 é marcado o fim do escopo do comando **for** da linha 4.

**Etapa 8:** Construção dos capítulos da parte avançada do tutorial. Nesta etapa é usado o mesmo algoritmo da Etapa 7, a única diferença é que ao invés de o algoritmo executar com os *posts* básicos, ele executa com os *posts* avançados obtidos na Etapa 5 da metodologia.

**Etapa 9:** Montagem final do tutorial. Este algoritmo recebe como entrada os capítulos básicos e avançados retornados pelas Etapas 7 e 8. Para cada capítulo do tutorial o algoritmo cria um sumário geral contendo *links* para os outros capítulos. O algoritmo também cria para cada capítulo um sumário contendo *links* para todos os seus respectivos exemplos (*posts* selecionados na Etapa 7 ou 8). Estes sumários facilitam a navegabilidade pelo tutorial. No final desta etapa o algoritmo retorna o tutorial montado em um arquivo HTML, depois basta hospedar este arquivo em um servidor de hospedagem de página WEB. O usuário final acessará o tutorial online através de qualquer dispositivo que tenha acesso à internet e um navegador.

## 3.2 Metodologia GP

A seguir serão apresentadas as principais etapas da Metodologia GP para a geração de um tutorial para uma dada API "X". A Figura 2 ilustra estas etapas.

**Etapa 1:** Definição dos tópicos que organizarão o tutorial usando o algoritmo LDA. Nesta etapa é usado o mesmo algoritmo da Etapa 6 da Metodologia FHCBAC para definir um número balanceado de tópicos para o LDA.

**Etapa 2:** Definição dos capítulos do tutorial usando os tópicos definidos na etapa anterior. Nesta etapa é feita a mesma análise da Etapa 6 da Metodologia FHCBAC para clusterizar os tópicos que abordam o mesmo assunto da API, e em seguida, os tópico(s) e/ou cluster(s) são analisados para receber um nome. Cada tópico(s) e/ou cluster(s) representará um capítulo no tutorial.

**Etapa 3:** Utilização do mecanismo de pesquisa do Google, tendo como entrada as palavras extraídas dos tópico(s) e/ou cluster(s) gerados na Etapa 2 e palavras chaves como: o nome da API "X", *"how to"*, filtrando somente para o *site* stackoverflow.com. No final desta etapa, para cada capítulo do tutorial, o Google retornará a pesquisa com os *posts* do Stack Overflow ranqueados.

**Etapa 4:** Seleção dos $n$ primeiros resultados ranqueados pelo mecanismo de pesquisa do Google, para cada capítulo.

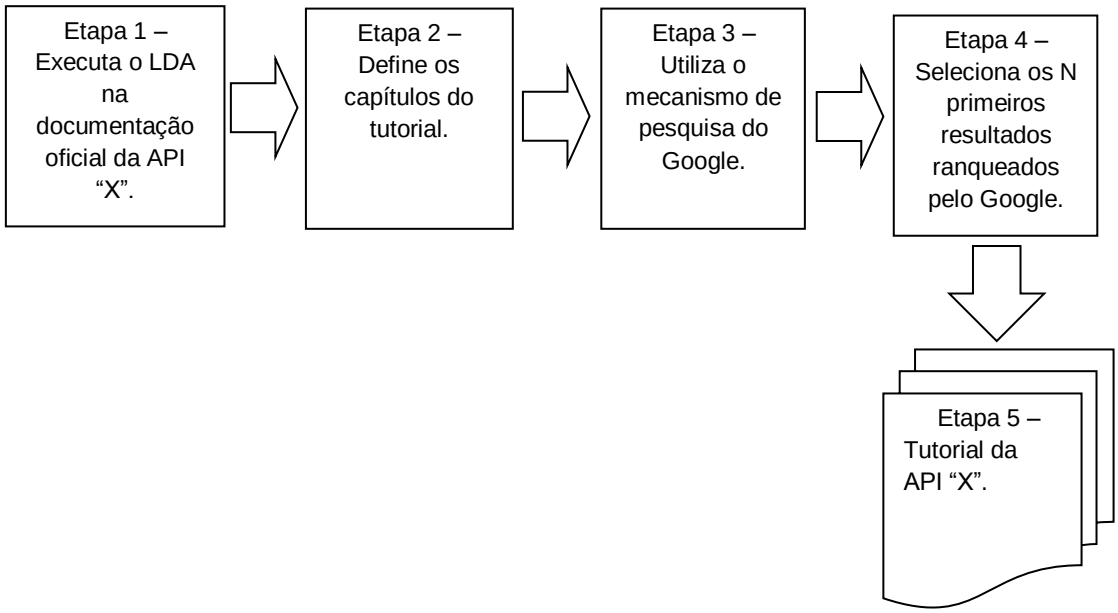


Figura 2 – Principais etapas para gerar um tutorial para API "X", utilizando a metodologia GP.

**Etapa 5:** Similar à Etapa 9 da Metodologia FHCBAC, com exceção que não há organização do tutorial em partes básica e avançada. Os *posts* utilizados para a montagem dos capítulos são aqueles obtidos na Etapa 4.

## 3.3 Metodologia GFC

A seguir serão apresentadas as principais etapas da Metodologia GFC para a geração de um tutorial para uma dada API "X". Esta metodologia é híbrida, pois combina algumas etapas da Metodologia FHCBAC com algumas etapas da Metodologia GP.

**Etapa 1:** Executa o LDA na documentação oficial da API "X". Nesta etapa é usado o mesmo algoritmo da Etapa 6 da Metodologia FHCBAC para encontrar um número de tópicos para o LDA.

**Etapa 2:** Usa os tópicos gerados pelo LDA na Etapa 1 para definir os capítulos do tutorial. Nesta etapa é feita a mesma análise da Etapa 6 da Metodologia FHCBAC para clusterizar os tópicos que abordam o mesmo assunto da API, e em seguida, os tópico(s) e/ou cluster(s) são analisados para receber um nome. Cada tópico(s) e/ou cluster(s) representará um capítulo no tutorial.

**Etapa 3:** Utiliza o mecanismo de pesquisa do Google, dando como entrada as palavras extraídas dos tópico(s) e/ou cluster(s) gerados na Etapa 2 e palavras chaves como: o nome da API "X", *"how to"* filtrando para o *site* stackoverflow.com. No final desta etapa, para

cada capítulo do tutorial, o Google retornará como resultado da pesquisa os *posts* do Stack Overflow ranqueados.

**Etapa 4:** Aplica o filtro desenvolvido neste trabalho para pegar os *posts* com as características:

1. No corpo da pergunta deve ter no máximo três linhas de código-fonte.
2. No código da resposta deve ter no mínimo três linhas de código-fonte.

**Etapa 5:** Ranqueia os *posts* através do algoritmo de ranqueamento por complexidade desenvolvido na Etapa 4 da Metodologia FHCBAC deste trabalho. Isto é feito para cada capítulo do tutorial.

**Etapa 6:** Seleciona os $n$ primeiros resultados ranqueados na Etapa 5. Isto é conduzido para cada capítulo do tutorial.

**Etapa 7:** Similar à Etapa 5 da Metodologia GP.

A Figura 3 ilustra as principais etapas da metodologia GFC.

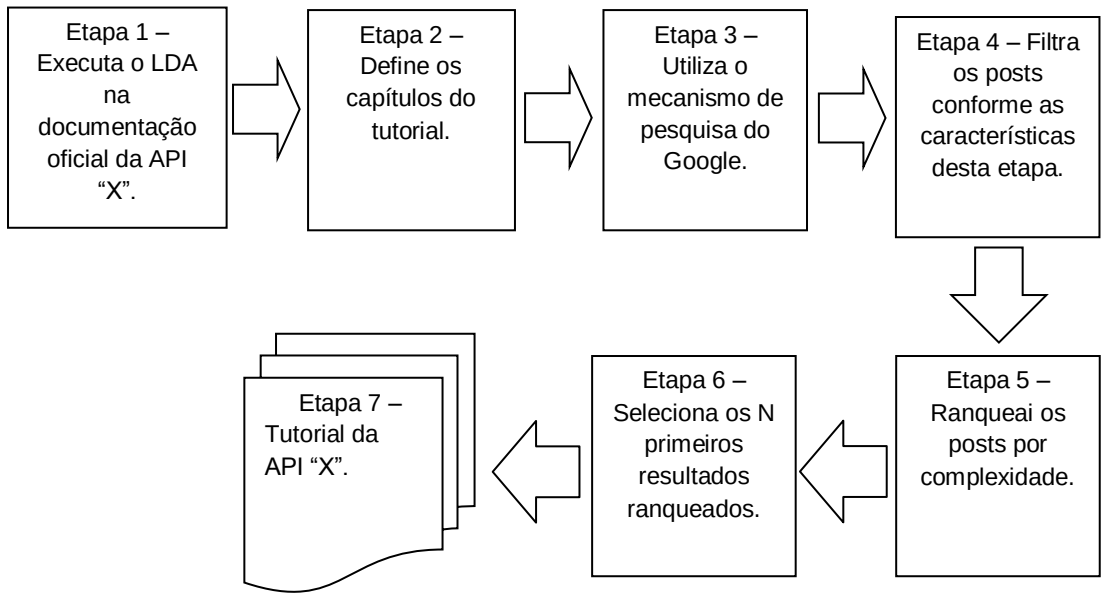


Figura 3 – Principais etapas para gerar um tutorial para API "X", utilizando a metodologia GFC.

## 3.4 Metodologia GFHCBAC

A seguir serão apresentadas as principais etapas da Metodologia GFHCBAC para a geração de um tutorial para uma dada API "X". Esta metodologia é híbrida, pois combina algumas etapas da Metodologia FHCBAC com algumas etapas da Metodologia GP.

**Etapa 1:** Gera o tutorial da API "X" utilizando a Metodologia FHCBAC.

**Etapa 2:** Utiliza o mecanismo de pesquisa do Google, dando como entrada as palavras chave extraídas dos capítulos do tutorial gerado na Etapa 1 e palavras chaves como: o nome da API "X", *"how to create"* e "Stack Overflow". No final desta etapa o Google retornará a pesquisa com os *posts* do Stack Overflow ranqueados, para cada capítulo.

**Etapa 3:** Aplica o filtro desenvolvido neste trabalho para pegar os *posts* com as características:

1. No corpo da pergunta deve ter no máximo três linhas de código-fonte.
2. No código da resposta deve ter no mínimo três linhas de código-fonte.

**Etapa 4:** Seleciona os $n$ primeiros *posts* de criação retornados pelo Google e os inclui no tutorial gerado na Etapa 1. Isto é feito para cada capítulo do tutorial.

**Etapa 5:** Executa o algoritmo desenvolvido no trabalho para montar o tutorial. Este algoritmo recebe como entrada o tutorial gerado pela Etapa 1 e os *posts* de criação de cada capítulo selecionados na Etapa 4. Os *posts* de criação são inseridos no início dos seus respectivos capítulos no tutorial, e os seus *links* também são inseridos no sumário do capítulo. O restante desta etapa é idêntico ao das demais abordagens.

A Figura 4 ilustra as principais etapas da metodologia GFHCBAC.

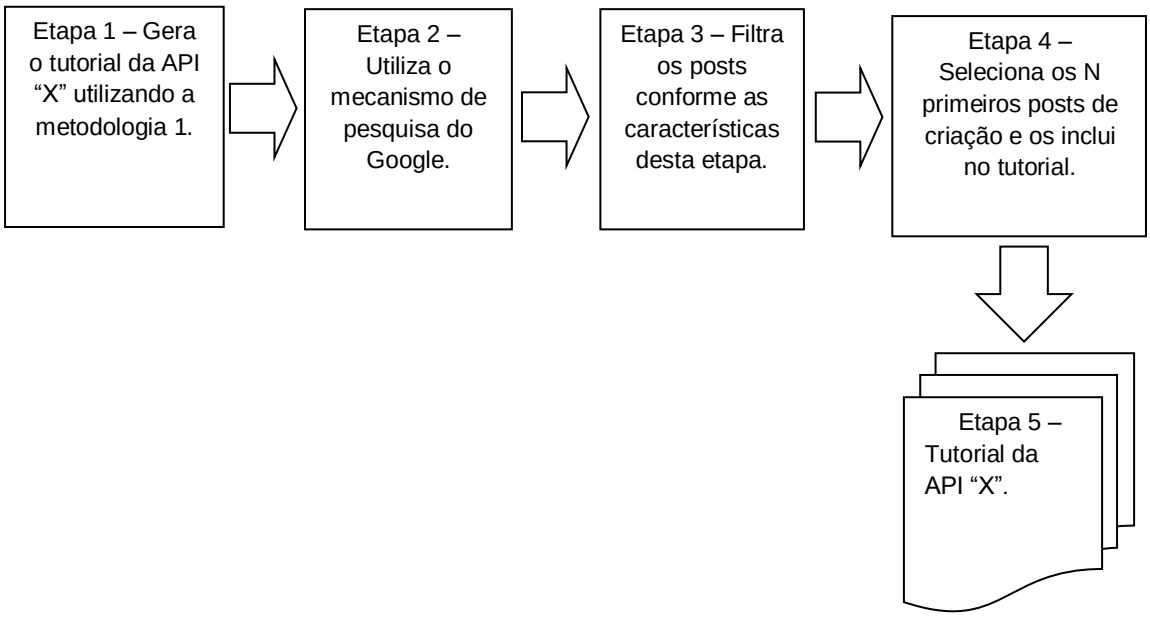


Figura 4 – Principais etapas para gerar um tutorial para API "X", utilizando a metodologia GFHCBAC.

Cabe ressaltar que os esforços para a geração dos tutoriais utilizando as metodologias propostas são mínimos, praticamente toda a abordagem é automática, a única etapa

manual é a da escolha dos nomes dos capítulos dos tutoriais analisando visualmente os termos dos tópicos gerados pelo LDA. Apesar de esta abordagem poder ser automatizada, preferimos não o fazer devido ao baixo esforço manual e a alta capacidade do ser humano escolher um bom título dado um conjunto de termos que define o tópico.

## 3.5 Características das Metodologias

A Tabela 4 mostra as principais características das metodologias desenvolvidas neste trabalho.

Tabela 4 – Principais características das metodologias.

| **Características** | **FHCBAC** | **GP** | **GFC** | **GFHCBAC** |
|---|---|---|---|---|
| Utiliza o LDA para definir os capítulos do tutorial | X | X | X | X |
| Classificador *"How-to"* | X | | | X |
| Filtro *"How-to"* do Google | | X | X | X |
| O código fonte da resposta do *post* deve ter pelo menos três linhas | X | | X | X |
| O código fonte da pergunta do *post* não pode ter mais do que três linhas | X | | X | X |
| Ranqueamento por complexidade | X | | X | X |
| Divisão dos capítulos em básicos e avançados | X | | | X |
| Os *posts* cobrem pelo menos um tipo da API | X | | | X |
| Utiliza o mecanismo de pesquisa do Google | | X | X | X |
| Seleciona *posts* de criação | | | | X |

Nas quatro metodologias é utilizado o LDA para definir os capítulos dos tutoriais. Com relação ao classificador *"How-to"* as metodologias FHCBAC e GFHCBAC o implementa. Na metodologia GP é utilizado o filtro do mecanismo de pesquisa do Google, através da formulação da pesquisa, onde é acrescentado a palavra *"How to"*, conforme a etapa 3 desta metodologia. Na metodologia GFC o filtro *"How to"* também é implementado utilizando o mecanismo de pesquisa do Google. Na metodologia GFHCBAC é utilizado o filtro *"How to"* do Google para os posts de "criação". Com relação às características **"O código fonte da resposta do post deve ter pelo menos três linhas"**, **"O código fonte da pergunta do post não pode ter mais do que três linhas"** e **"Ranqueamento por complexidade"**, estas são implementadas nas metodologias FHCBAC,GFC e GFHCBAC utilizando os mesmos algoritmos desenvolvidos neste trabalho, a metodologia GP não implementa estas características. As características **"Divisão dos capítulos em básicos e avançados"** e **"Os posts cobrem pelo menos um tipo da API"** são implementadas nas metodologias FHCBAC e GFHCBAC. As metodologias GP e GFC não implementam estas características. A característica **"Utiliza o mecanismo**

**de pesquisa do Google"** é implementada nas metodologias GP, GFC e GFHCBAC. A metodologia FHCBAC não implementa estas características. A característica **"Seleciona posts de criação"** é implementada somente na metodologia GFHCBAC. A metodologia GFHCBAC é a única que implementa todas as características da Tabela 4.

# Capítulo 4

# Desenho Experimental

Este capítulo tem como objetivo apresentar o procedimento adotado na avaliação e comparação das quatro metodologias desenvolvidas neste trabalho, além de apresentar o procedimento para a avaliação e comparação de um tutorial gerado pela metodologia GFHCBAC com a documentação oficial para a API Android.

## 4.1 Avaliação dos tutoriais gerados pelas quatro metodologias propostas

Foram utilizadas as quatro metodologias apresentadas anteriormente para gerar tutoriais para a API Swing. A escolha da API Swing tem vários pontos favoráveis para a avaliação das metodologias, apesar de a tecnologia desktop ter perdido espaço para tecnologias móveis e web, a popularidade do Swing entre novatos ainda é alta, é uma API conhecida que permite avaliar qualitativamente a qualidade do tutorial gerado, além de ser uma API bastante discutida no Stack Overflow. A escolha da API tem pouco impacto sob o ponto de vista de comparação entre as abordagens. As metodologias propostas neste trabalho foram desenhadas para gerar tutoriais não só para APIs, mas também para bibliotecas, linguagens de programação, ou qualquer tecnologia que contenha código-fonte como exemplos.

Cada tutorial recebeu um nome conforme as metodologias que o geram, FHCBAC: "Blue Version", GP: "Red Version", GFC: "Yellow Version" e GFHCBAC: "Green Version". Na Figura 5 é apresentada a tela inicial para o tutorial "Green Version". Na lateral esquerda, são apresentados os *links* de navegação para os capítulos, divididos em parte básica e avançada. Na parte superior central são apresentados os *links* para os elementos do Capítulo 1 (*Table*). Na parte inferior, já é apresentado o primeiro elemento do Capítulo 1, e os próximos elementos do capítulo também podem ser acessados por meio de rolagem. As quatro versões do tutorial da API Swing podem ser acessadas online[1].

[1] http://lascam.facom.ufu.br/APITutorial/SwingTutorial/

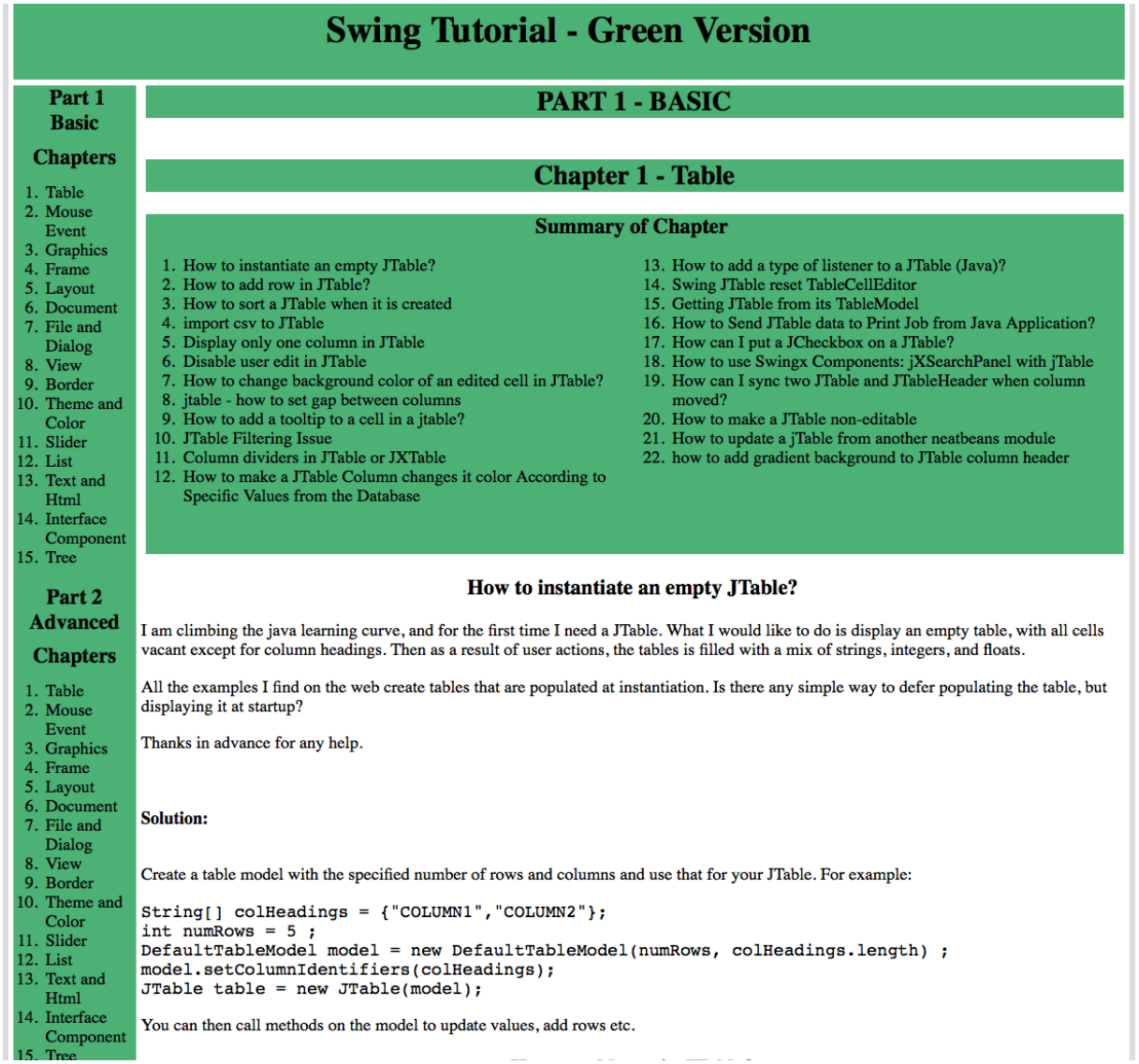


Figura 5 – Visão Inicial do Tutorial Swing gerado por GFHCBAC

Para avaliar estes tutoriais foi desenvolvido um formulário (Anexo A) online através do software Lime Survey. O link do formulário foi submetido a uma lista de e-mail do Programa de Pós-graduação em Ciência da Computação da UFU. O grupo possui cerca de 200 discentes. O link também foi enviado para 6 profissionais que trabalham na área de desenvolvimento de software. Ao todo 12 respondentes completaram a avaliação dos tutoriais.

Nas afirmações do questionário para avaliar os tutoriais foi utilizada a escala Likert em 5 níveis para medir o grau de concordância de cada afirmação que varia do extremo *Concordo fortemente* ao *Discordo fortemente*. Os participantes avaliaram os tutoriais conforme os seguintes critérios:

1. **Organização**. A afirmação solicitada para os participantes avaliarem foi: *O tutorial é organizado*. Os elementos adicionais indicados para o avaliador utilizar ao definir

o *score* desta afirmação foram:

- ❏ O sumário do tutorial está numerado.
- ❏ Os sumários dos capítulos possuem os títulos de todos os exemplos encontrados em um determinado capítulo.
- ❏ Os exemplos encontrados no tutorial estão organizados em pergunta seguido da solução.

2. **Intuitividade**. A afirmação solicitada para os participantes avaliarem foi: *O tutorial é intuitivo.* Os elementos adicionais indicados foram:
   - ❏ O tutorial pode ser utilizado sem nenhum tipo de treinamento.
   - ❏ Sua utilização é autoexplicativa.
3. **Navegabilidade**. A afirmação solicitada para os participantes avaliarem foi: *O tutorial possui uma navegabilidade adequada nos sumários.* Os elementos adicionais indicados foram:
   - ❏ O sumário do tutorial possui links para todos os capítulos.
   - ❏ Em cada capítulo, o sumário geral pode ser acessado, possibilitando a navegação para outros capítulos.
   - ❏ Cada capítulo possui seu próprio sumário com links para os exemplos do respectivo capítulo.
4. **Natureza didática do código-fonte**. A afirmação solicitada para os participantes avaliarem foi: *Os exemplos do tutorial possuem códigos-fonte didáticos.*
5. **Ordem de complexidade dos elementos do tutorial**. A afirmação solicitada para os participantes avaliarem foi: *O tutorial segue uma ordem crescente de complexidade dos exemplos dos capítulos, ou seja, os exemplos mais simples aparecem antes dos exemplos mais complexos.*
6. **Partes básica e avançada**. A afirmação solicitada para os participantes avaliarem foi: *A organização do tutorial com divisão em partes básica e avançada é adequada.*
7. **Natureza de tutorial**. A afirmação solicitada para os participantes avaliarem foi: *A natureza dos exemplos é compatível com um tutorial. Em outras palavras, os exemplos explicam como utilizar funcionalidades da API, e não, por exemplo, como corrigir erros de programação.*
8. **Coesão interna dos capítulos**. A afirmação solicitada para os participantes avaliarem foi: *Os exemplos encontrados nos capítulos estão relacionados com o respectivo nome do capítulo.*

Para cada item de avaliação, os participantes foram instruídos a avaliar comparativamente as quatro versões do tutorial, sendo que, se a versão X for melhor do que a versão Y atribuir um *score* maior para X em relação à Y.

A escolha do teste estatístico para análise das diferenças entre as avaliações levou em consideração uma série de características dos dados: 1) os resultados do *survey* são valores ordinais da escala Likert, 2) a amostra analisada não é grande e os resultados não permitem assumir normalidade dos dados, 3) cada avaliador avaliou os 4 tutoriais, o que caracteriza um experimento em blocos (avaliadores). Sendo assim, adotou-se o Teste de Friedman com análise *post-hoc* por ser um teste não-paramétrico que não requer suposição de normalidade da população e nem de tamanho amostral elevado. Além disso, suporta dados ordinais em bloco. Foi utilizada uma função utilitária baseada no pacote *coin* disponível na CRAN (*Comprehensive R Archive Network*)(HOTHORN et al., 2008).

## 4.2 Avaliação da comparação da documentação oficial da API Android com o tutorial gerado pela metodologia GFHCBAC

Para avaliar o tutorial gerado pela metodologia GFHCBAC em comparação com a documentação oficial de API, foi realizado um experimento envolvendo 8 alunos de graduação do curso de Sistemas de Informação da UFU. Para realizar o experimento foram desenvolvidas duas tarefas envolvendo a programação para o Android, o enunciado destas tarefas encontra-se no Anexo B desta dissertação. Os alunos nunca tiveram contato com a programação Android. O experimento foi realizado em duas etapas, na primeira etapa foi realizado um treinamento com os alunos para apresentar o ambiente de desenvolvimento para o Android e conceitos básicos. Na segunda etapa foi dado duas tarefas aos alunos. Para realizar estas tarefas os alunos foram organizados em dois grupos. Para a primeira tarefa o grupo A utilizou a documentação oficial do Android (Site Oficial) e o grupo B utilizou o tutorial gerado pela metodologia GFHCBAC (Green Version). Para realizar a segunda tarefa o grupo A utilizou o tutorial GFHCBAC e o grupo B utilizou a documentação oficial. O experimento foi realizado no laboratório de informática da UFU, onde todos os computadores possuem as mesmas configurações de hardware e software. Foi utilizado um software para gravar as telas dos computadores que os alunos utilizaram para desenvolver as tarefas. Foi cronometrado o tempo que cada aluno levou para realizar as tarefas. Para cada tarefa foi dado um tempo limite de 60 minutos. Ao final do experimento foi dado a cada aluno um questionário contendo 4 afirmações sobre as duas documentações utilizadas no experimento e duas questões abertas. Nas afirmações do questionário para avaliar as documentações foi utilizada a escala Likert em 5 níveis para medir o grau de concordância de cada afirmação que varia do extremo *Concordo fortemente* ao *Discordo*

*fortemente*. Os participantes avaliaram as documentações conforme os seguintes critérios: organização, intuitividade, navegabilidade e didática dos códigos-fonte. O questionário aplicado no experimento encontra-se no Anexo B desta dissertação.

# Capítulo 5

# Resultados e Discussão

Este capítulo tem como objetivo apresentar a avaliação dos tutoriais gerados pelas metodologias desenvolvidas neste trabalho. Também são apresentados os resultados da avaliação da comparação do tutorial gerado pela metodologia GFHCBAC com a documentação oficial para a API Android.

## 5.1 Resultados da avaliação dos tutoriais gerados pelas quatro metodologias propostas

Esta seção apresenta os resultados da avaliação dos tutoriais gerados pelas metodologias desenvolvidas neste trabalho. A Figura 6 apresenta os *boxplots* com os resultados de avaliação para cada critério. Para tornar a avaliação mais intuitiva foram colocados os pontos de avaliação com *jittering* em relação ao respectivo *score*, para evitar que todos os pontos ficassem na mesma posição, inviabilizando a respectiva apresentação.

A Tabela 5 mostra os resultados de comparação de diferença entre os *scores* das respectivas metodologias para cada critério. O resultado dos *p-values* menores que 0.05 para o teste de Friedman indica diferença estatisticamente significante entre as metodologias. Para identificar quais metodologias são responsáveis pelas diferenças, a Tabela 6 apresenta os resultados da análise *post-hoc* para o teste de Friedman.

Tabela 5 – Teste de Friedman para os *scores* de cada critério agrupados por metodologia com bloco por avaliadores

| Critério | p-value |
|---|---|
| **Básica-Avançada** | 0.002 |
| **Código didático** | 0.049 |
| Coesão do capítulo | 0.108 |
| **Complexidade** | 0.004 |
| Intuitividade | 0.472 |
| **Natureza de tutorial** | 0.008 |
| Navegabilidade | 0.146 |
| Organização | 0.437 |

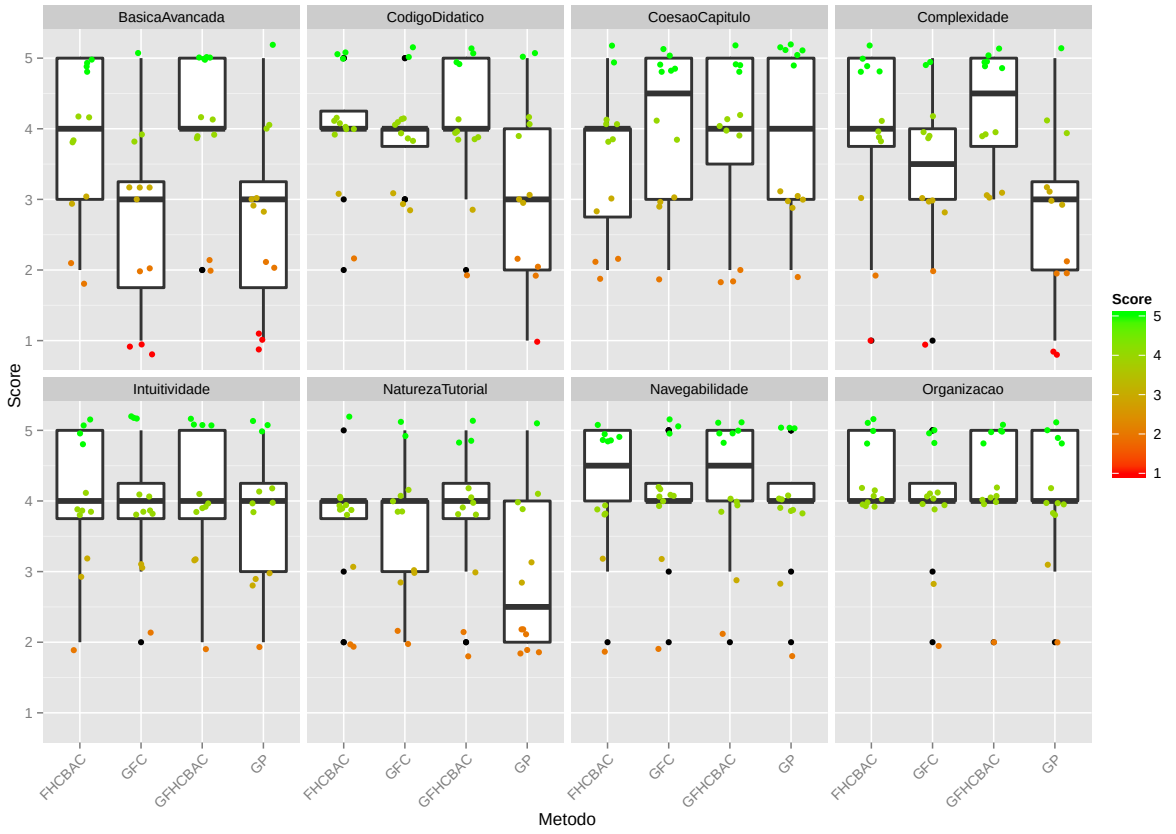


Figura 6 – Avaliação dos critérios de tutoriais. 5: Concordo Fortemente ... 1: Discordo Fortemente

Tabela 6 – Comparação entre metodologias por análise *post-hoc* para o Teste de Friedman

| Par de Metodologias | Complex. | Bas.Av. | Natur. | C.Did. |
|---|---|---|---|---|
| GFC - FHCBAC | 0.936 | **0.049** | 0.999 | 0.928 |
| GFHCBAC - FHCBAC | 0.861 | 0.733 | 0.891 | 0.983 |
| GP - FHCBAC | *0.052* | **0.049** | *0.060* | 0.122 |
| GFHCBAC - GFC | 0.516 | **0.001** | 0.839 | 0.761 |
| GP - GFC | 0.203 | 1.000 | 0.081 | 0.383 |
| GP - GFHCBAC | **0.004** | **0.002** | **0.007** | **0.049** |

**Organização**. Como se observa na Figura 6, os tutoriais gerados pelas quatro metodologias receberam um alto *score* para a organização, isto porque os quatro tutoriais estão organizados de forma semelhante. Como esperado, o Teste de Friedman não apontou diferença entre as metodologias. A razão para os altos *scores* deve-se aos sumários estarem numerados e possuírem os títulos de todos os exemplos encontrados em um determinado capítulo, além dos exemplos estarem organizados em pergunta seguida da solução.

**Intuitividade**. Este item de avaliação também apresentou altos *scores* para os quatro tutoriais, isto porque eles são autoexplicativos, o *layout* é bem simples e conta com *links* para os outros capítulos na tela inicial, conforme o usuário vai navegando pelos tutoriais, ele vai encontrando *links* que o leva direto para os exemplos oferecidos pelos tutoriais. Cabe ressaltar que houve avaliador que não concordou com a intuitividade e vários que

se posicionaram de maneira neutra.

**Navegabilidade**. Este item também apresentou altos *scores* para os quatro tutoriais, indicando que o fato deles possuírem dois tipos de sumário, um sumário geral que pode ser utilizado para acessar qualquer um dos capítulos, e um sumário local em cada capítulo que pode ser utilizado para acessar os exemplos do respectivo capítulo mostrou-se uma navegação adequada.

**Complexidade**. Dos quatro tutoriais gerados pelas metodologias, para este item de avaliação, a GFHCBAC foi a que obteve *scores* mais altos, podendo ser observado pela mediana na Figura 6, além de ser estatisticamente significativa a diferença em relação à GP. A explicação para o melhor desempenho da GFHCBAC é que esta metodologia possui uma etapa de ranqueamento dos exemplos, além de possuir exemplos simples de criação no começo de cada capítulo da parte básica do tutorial. A FHCBAC, a segunda com maior mediana, também possui uma etapa de ranqueamento dos exemplos. A diferença da avaliação em comparação com a GFHCBAC pode estar pelo fato da FHCBAC não gerar exemplos de criação no começo de cada capítulo básico do tutorial. O terceiro tutorial com maior mediana foi o gerado pela Metodologia GFC que utiliza o mecanismo de pesquisa do Google para ranquear os $n$ exemplos do tutorial, em uma etapa posterior também é aplicado um algoritmo de ranqueamento por complexidade. O tutorial gerado pela Metodologia GP foi o que teve a menor mediana, podendo ser explicado por esta metodologia somente utilizar o mecanismo de pesquisa do Google para ranquear os exemplos do tutorial. Como este mecanismo não leva em consideração a complexidade do exemplo, então não se pode esperar que estes exemplos estejam ranqueados por ordem de complexidade, podendo acontecer de um exemplo muito complexo aparecer antes de um exemplo muito simples, uma vez que o Google utiliza outras características para o ranqueamento.

**Partes básica e avançada**. Os tutoriais gerados pelas Metodologias GP e GFC apresentaram menores medianas entre os quatro, porque estas metodologias não dividem os capítulos dos tutoriais em partes básica e avançada. Mesmo que estas metodologias não implementam esta característica, ela foi considerada para efeitos de controle da avaliação, ou seja, para verificar se os avaliadores foram consistentes na avaliação dos tutoriais. A maioria dos avaliadores deu nota baixa para este quesito ou ficaram neutros para estas metodologias. Já as Metodologias FHCBAC e GFHCBAC dividem os capítulos em partes básica e avançada, explicando os melhores *scores*. Uma diferença estatisticamente significativa entre os grupos GP/GFC em relação aos grupos GFHCBAC/FHCBAC mostra que tal separação se mostrou efetiva. No entanto, alguns avaliadores discordaram, uma possível explicação seria por se tratar de um tutorial, eles acharam que não é adequado os tutoriais terem a parte avançada.

**Natureza de tutorial**. Os tutoriais gerados pelas Metodologias GFHCBAC e FHCBAC foram os que apresentaram maiores medianas, tendo o GFHCBAC sido estatistica-

mente maior que GP e FHCBAC com uma diferença *borderline* em relação à GP. Inclusive a GFC obteve uma diferença com p-value de 0.08 em relação à GP. Isto indica a adequação dos filtros aplicados em GFHCBAC, FHCBAC e GFC. Por exemplo, o filtro para selecionar apenas os exemplos do tipo *"How-to-do"*, o filtro que seleciona os exemplos com as características: *"no corpo da pergunta do exemplo deve ter no máximo três linhas de código fonte"*, *"no código da resposta do exemplo deve ter no mínimo três linhas de código fonte"* e *"o código fonte da resposta deve cobrir pelo menos um tipo da API(Classes, Interfaces, etc.)."*. Além de estas metodologias utilizarem o ranqueador por complexidade, que ranqueia os exemplos conforme sua complexidade, ou seja, os *posts* mais simples aparecem primeiro do que os *posts* mais complexos. Estas metodologias também dividem o tutorial em partes básica e avançada. A GFHCBAC teve um nível de concordância um pouco maior do que a FHCBAC, que pode ser explicado devido à GFHCBAC ser uma extensão da versão FHCBAC, com exemplos de criação no começo de cada capítulo. O tutorial gerado pela Metodologia GP obteve a menor mediana nos *scores*. Esta metodologia só utiliza o mecanismo de pesquisa do Google para ranquear os exemplos do tutorial, e como este mecanismo não leva em consideração a complexidade do conteúdo do exemplo, então os exemplos complexos estão misturados com os simples, sem nenhum ranqueamento por complexidade, o que indica uma deficiência na caracterização como tutorial. Além disso, não é aplicado nenhum dos filtros utilizados nas Metodologias GFHCBAC e FHCBAC. A GFC teve um nível de concordância intermediário entre as FHCBAC e GP, isto porque a GFC implementa alguns dos filtros utilizados nas Metodologias FHCBAC e GFHCBAC.

**Código-fonte didático**. Os tutoriais gerados pelas Metodologias GFHCBAC, FHCBAC e GFC, tiveram uma maior mediana em relação à GP. Entretanto, apenas GFHCBAC e GP se mostraram estatisticamente diferentes. Isto pode ser explicado pelo fato das metodologias que geram estes tutoriais implementarem filtros que tentam selecionar exemplos do tipo "Tutorial", conforme os critérios de cada filtro. A análise da Figura 6 indica que exemplos do tipo "Tutorial" são mais didáticos. O tutorial gerado pela Metodologia GP teve a mediana neutra, possivelmente por esta metodologia não implementar os filtros para selecionar exemplos do tipo "Tutorial", apenas utiliza o mecanismo de pesquisa do Google. Mesmo que este mecanismo não leve em consideração a complexidade dos exemplos, ou não tente identificar os exemplos do tipo "Tutorial", este mecanismo ainda retorna exemplos que ajudam os usuários a resolverem os seus problemas, mesmo que não seja da maneira mais didática possível.

**Coesão interna do capítulo**. Apesar de não ter havido diferença estatisticamente significativa entre os diversos tutoriais em relação à coesão dos capítulos, o tutorial gerado pela Metodologia GFC teve a maior mediana entre os quatro tutoriais. Esta metodologia utiliza o mecanismo de pesquisa do Google para buscar os exemplos do tutorial, sendo este mecanismo muito eficiente em buscar exemplos relacionados com o capítulo, uma vez que, para formular o texto da pesquisa, são considerados os termos retornados pelo LDA para

caracterização do capítulo. O tutorial gerado pela Metodologia GP também teve uma mediana alta, porém muitos avaliadores ficaram neutros em relação a este tutorial, onde sua metodologia também utiliza o mecanismo do Google para buscar os seus exemplos. Os tutoriais gerados pelas Metodologias FHCBAC e GFHCBAC também tiveram uma alta mediana, porém houve algumas avaliações negativas. Isto pode ser explicado devido a estas metodologias buscarem os exemplos para o tutorial considerando o conteúdo do código-fonte destes exemplos. Por exemplo, pode acontecer de um exemplo do tutorial que está relacionado ao capítulo "Tabela" estar em um capítulo relacionado à "Botão", se este exemplo tiver botões dentro de uma tabela.

### 5.1.1 Ameaças à Validade

Uma ameaça à validade externa é a amostra de avaliadores que participaram do estudo. Avaliadores de outras regiões ou países podem ser mais ou menos rigorosos na avaliação dos tutoriais. Em relação à validade interna, outros fatores que poderiam interferir em melhores resultados em geral para a abordagem GFHCABC são o nível de experiência dos avaliadores na API Swing ou a possibilidade de divergência de interpretação das afirmações. Para esta última, procurou-se mitigar por meio de afirmações simples e com auxílio à interpretação em casos onde identificou-se maior possibilidade de ambiguidade.

## 5.2 Resultados da avaliação da comparação da documentação oficial da API Android com o tutorial gerado pela metodologia GFHCBAC

Esta seção mostra os resultados da avaliação da comparação da documentação oficial da API Android com o tutorial gerado pela metodologia GFHCBAC para a mesma API. As Tabelas 7 e 8 mostram a relação do tempo gasto pelos os alunos nas tarefas 1 e 2 respectivamente.

Tabela 7 – Tabela de relação de tempo gasto na tarefa 1.

| **Aluno** | **Documentação Utilizada** | **Tempo para executar a tarefa** | **Finalizou a tarefa?** |
|---|---|---|---|
| Aluno 01 | Documentação Oficial | 60 minutos | Não |
| Aluno 02 | Tutorial GFHCBAC | 20 minutos | Sim |
| Aluno 03 | Documentação Oficial | 15 minutos | Sim |
| Aluno 04 | Tutorial GFHCBAC | 27 minutos | Sim |
| Aluno 05 | Documentação Oficial | 54 minutos | Sim |
| Aluno 06 | Tutorial GFHCBAC | 17 minutos | Sim |
| Aluno 07 | Documentação Oficial | 60 minutos | Não |
| Aluno 08 | Tutorial GFHCBAC | 26 minutos | Sim |

Tabela 8 – Tabela de relação de tempo gasto na tarefa 2.

| **Aluno** | **Documentação Utilizada** | **Tempo para executar a tarefa** | **Finalizou a tarefa?** |
|---|---|---|---|
| Aluno 01 | Tutorial GFHCBAC | 44 minutos | Sim |
| Aluno 02 | Documentação Oficial | 60 minutos | Não |
| Aluno 03 | Tutorial GFHCBAC | 8 minutos | Sim |
| Aluno 04 | Documentação Oficial | 60 minutos | Não |
| Aluno 05 | Tutorial GFHCBAC | 60 minutos | Não |
| Aluno 06 | Documentação Oficial | 52 minutos | Sim |
| Aluno 07 | Tutorial GFHCBAC | 17 minutos | Sim |
| Aluno 08 | Documentação Oficial | 60 minutos | Não |

Como se pode observar na Tabela 7 dos quatro alunos que executaram a tarefa 1 utilizando a documentação oficial, dois não conseguiram finalizar a tarefa, analisando os formulários que estes dois alunos responderam ao final do experimento, eles argumentaram que a organização da documentação oficial é confusa, contendo arquivos e artigos misturados, dificultando a navegação, além de sua difícil utilização. Dois alunos conseguiram finalizar a tarefa 1 utilizando a documentação oficial. Os quatro alunos que utilizaram o tutorial GFHCBAC conseguiram finalizar a tarefa 1, nos questionários que estes alunos responderam ao final do experimento, eles argumentaram que o tutorial possui exemplos práticos, tópicos sobre os assuntos explícitos e fácil utilização. Estes argumentos explicam o bom desempenho destes alunos ao executar a tarefa 1. Portanto os alunos que utilizaram o tutorial GFHCBAC tiveram melhores desempenho ao executar a tarefa 1 do que os alunos que utilizaram a documentação oficial.

A Tabela 8 mostra a relação do tempo gasto pelos alunos para executar a tarefa 2. Dos quatro alunos que executaram a tarefa 2 utilizando a documentação oficial apenas um conseguiu finalizar a tarefa. Com relação aos alunos que utilizaram o tutorial GFHCBAC para realizar a tarefa 2, dos quatro, três conseguiram realizar a tarefa. Para a tarefa 2 os alunos que utilizaram o tutorial GFHCBAC também tiveram melhor desempenho ao executar a tarefa do que os alunos que utilizaram a documentação oficial.

Ao fazermos uma comparação das Tabelas 7 e 8 levando em consideração cada aluno, podemos constatar que na maioria dos casos, os alunos tiveram melhores desempenhos nas tarefas quando eles utilizaram o tutorial GFHCBAC. Por exemplo, o Aluno 01 utilizando a documentação oficial não conseguiu finalizar a tarefa 1 em até 60 minutos, já utilizando o tutorial GFHCBAC ele conseguiu finalizar a tarefa 2 em 44 minutos. O Aluno 02 utilizando o tutorial GFHCBAC conseguiu finalizar a tarefa 1 em 20 minutos, já utilizando a documentação oficial ele não conseguiu finalizar a tarefa 2 em até 60 minutos. Para os outros alunos podemos fazer a mesma análise, com exceção do Aluno 05 que teve melhor desempenho utilizando a documentação oficial, neste caso ele conseguiu finalizar a tarefa 1 em 54 minutos utilizando a esta documentação, já utilizando o tutorial GFHCBAC ele não conseguiu finalizar a tarefa em até 60 minutos. Fazendo uma análise

do questionário respondido pelos alunos ao final do experimento, podemos notar que os principais motivos do baixo desempenho dos alunos nas tarefas utilizando a documentação oficial da API foram devido à falta de exemplos de uso nesta documentação, linguagem de difícil entendimento para quem está começando a utilizar a tecnologia, além de sua difícil utilização.

A Figura 7 mostra o resultado da *"afirmação 01 - A documentação é organizada."* para as duas documentações.

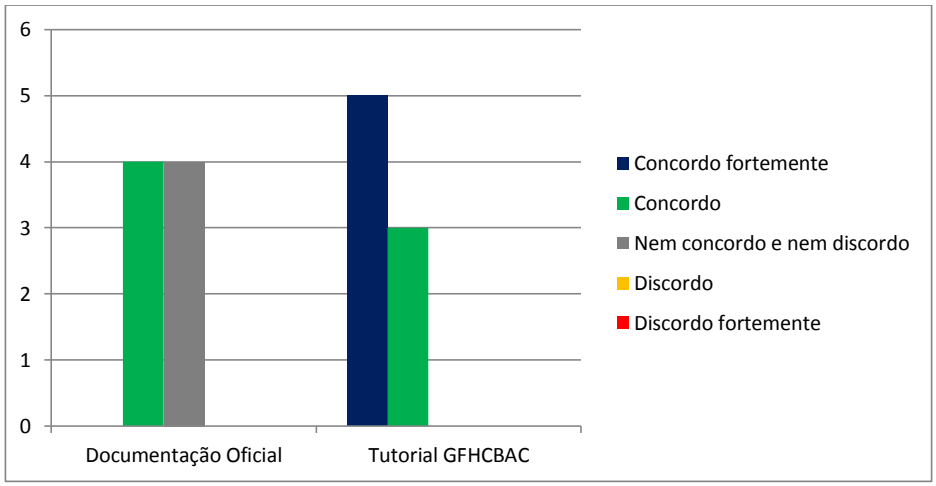


Figura 7 – Resultado da afirmação 01: A documentação é organizada.

Como podemos observar através da Figura 7 a documentação oficial e o tutorial GFHCBAC tiveram bons resultados, apesar de o tutorial ter tido uma avaliação melhor. Uma possível explicação seria pelo fato do tutorial estar organizado em partes básica e avançada, os exemplos estarem organizados em pergunta seguida da solução, além de ter sumários numerados com títulos de todos os exemplos encontrados em um determinado capítulo.

A Figura 8 mostra o resultado da *"afirmação 02 - A documentação é intuitiva."* para as duas documentações. A documentação oficial teve uma avaliação bem inferior em relação ao tutorial GFHCBAC para este quesito. Isto pode ser explicado pelo fato do tutorial ser mais simples, além de contar com *links* para os outros capítulos na tela inicial, conforme o usuário vai navegando pelo tutorial, ele vai encontrando *links* que o leva direto para os exemplos oferecidos pelo tutorial. Já a documentação oficial possui uma linguagem de difícil compreensão, falta de exemplos de uso e organização confusa o que contribui para a falta de intuitividade na documentação.

A Figura 9 mostra o resultado da *"Afirmação 03 - A documentação possui boa navegabilidade."* para as duas documentações. Como podemos observar o tutorial GFHCBAC teve uma boa avaliação, isto porque os sumários encontrados no tutorial facilitam a navegação do usuário. Já a avaliação para a documentação oficial ficou dividida, a metade

dos alunos concordaram que a documentação possui boa navegabilidade e a outra metade discordou.

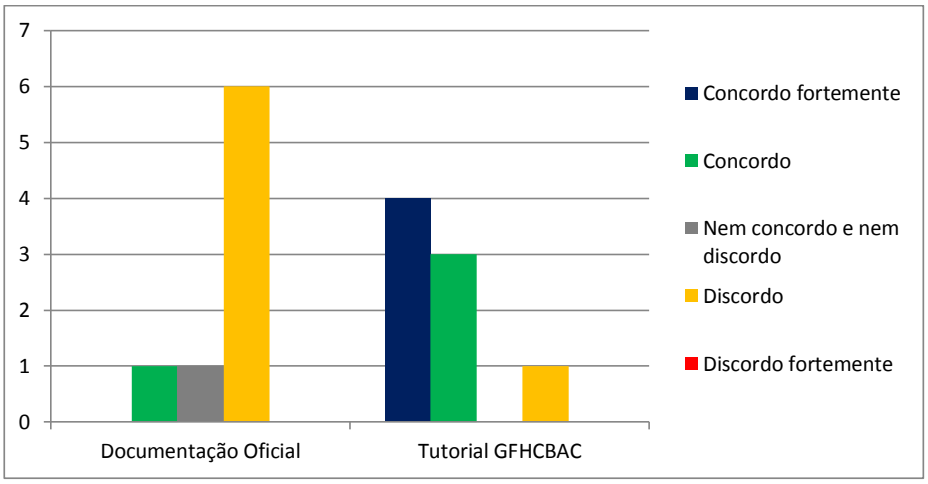


Figura 8 – Resultado da afirmação 02: A documentação é intuitiva.

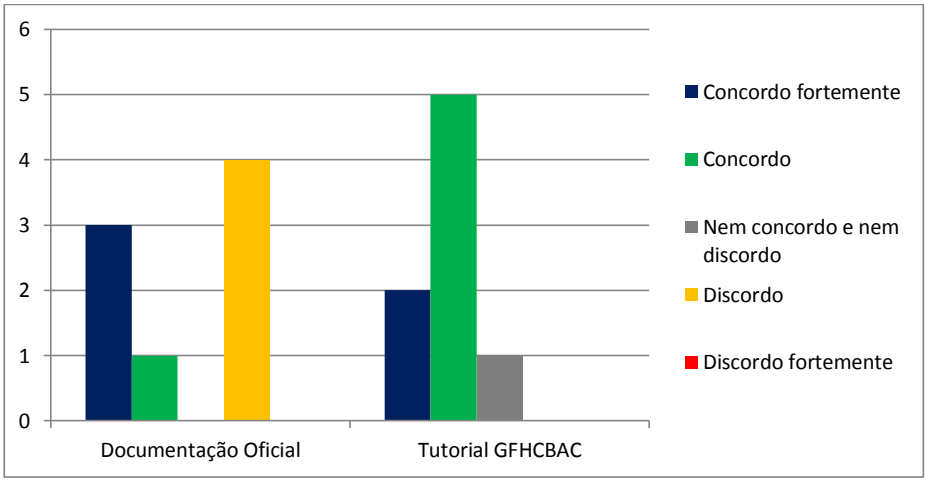


Figura 9 – Resultado da afirmação 03: A documentação possui boa navegabilidade.

A Figura 10 mostra o resultado da *"Afirmação 04 - Os exemplos da documentação possuem códigos-fonte didáticos."* para as duas documentações. O tutorial GFHCBAC também teve uma boa avaliação para este quesito, isto porque a metodologia que gera o tutorial implementa filtros que tentam selecionar exemplos do tipo "Tutorial" que são mais didáticos, conforme discutido na seção anterior. Além dos exemplos do tutorial estarem ranqueados conforme sua ordem de complexidade, ou seja, os exemplos mais simples são apresentados antes dos exemplos mais complexos. A documentação oficial não teve uma boa avaliação neste quesito, isto pode ser explicado pela falta de exemplos de uso mais simples.

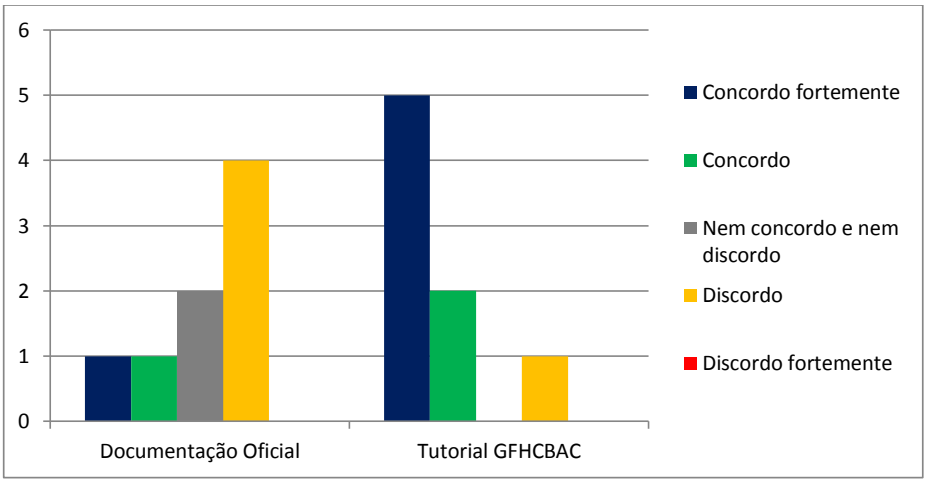


Figura 10 – Resultado da afirmação 04: Os exemplos da documentação possuem códigos-fonte didáticos.

A Tabela 9 mostra os principais pontos citados nas vantagens e desvantagens de cada uma das documentações utilizadas no experimento.

Tabela 9 – Principais pontos citados nas vantagens e desvantagens de cada documentação

| **Principais Pontos** | **Documentação Oficial** | **Tutorial GFHCBAC** |
|---|---|---|
| **Vantagens** | | |
| Barra de pesquisa | 2 | 0 |
| Fácil utilização | 0 | 3 |
| Tópicos dos assuntos explícitos | 0 | 2 |
| Exemplos práticos | 0 | 1 |
| Informativo | 0 | 1 |
| **Desvantagens** | | |
| Falta de exemplos com código-fonte | 5 | 0 |
| Difícil utilização | 4 | 0 |
| Sistema de busca pouco eficaz | 1 | 0 |
| Linguagem de difícil entendimento para quem está começando a utilizar a tecnologia | 1 | 0 |
| Falta de mais métodos do recurso buscado | 0 | 1 |

A Tabela 10 mostra as sugestões dos alunos que participaram do experimento para as duas documentações.

### 5.2.1 Ameaças à Validade

Uma ameaça à validade externa é a amostra de avaliadores que participaram do estudo. Avaliadores de outras regiões ou países podem ser mais ou menos rigorosos na avaliação dos tutoriais. Além disso, o número pequeno de avaliadores pode não refletir uma representatividade adequada da população de desenvolvedores novatos. Uma outra

ameaça é a realização do experimento somente com a API Android, apesar de ser uma plataforma amplamente utilizada, pode ser que resultados sejam diferentes para outras APIs. Em relação à validade interna, um fator que poderia interferir nos resultados em geral seria os diferentes perfis dos avaliadores em relação as suas experiências com a API Android, procurou-se mitigar este fator realizando um treinamento com todos os avaliadores antes de aplicar as tarefas do experimento para que suas experiências pudessem ser uniformizadas.

Tabela 10 – Sugestões dos alunos para as duas documentações avaliadas

| **Sugestões** | **Documentação Oficial** | **Tutorial GFHCBAC** |
|---|---|---|
| Inserir mais exemplos de uso | 5 | 0 |
| Melhorar a organização dos exemplos | 2 | 0 |
| Melhorar o sistema de busca | 1 | 0 |
| Suavizar a gramática da linguagem utilizada na documentação | 1 | 0 |
| Implementar barra de pesquisa | 0 | 2 |
| Inserir explicação do que as classes ou métodos fazem | 0 | 2 |
| Melhorar a interface | 0 | 2 |

Capítulo 6

# Trabalhos Relacionados

Vários autores têm trabalhado na questão de como elaborar uma documentação de uma API de forma mais simples e eficaz. Storey et al. (STOREY et al., 2010) mostraram a importância das mídias sociais (wikis, blogs, redes sociais, etc.) nas atividades da Engenharia de Software, que vão desde a engenharia de requisitos e desenvolvimento até testes e documentação. No artigo os autores citam a importância do SO como sendo um site de troca de informações e gerenciamento de trabalho colaborativo.

No trabalho de Treude et al. (TREUDE et al., 2012) foi discutido as oportunidades e os desafios que os desenvolvedores de software têm ao fazerem uso de informações disponibilizadas por sites da internet, como por exemplo, o SO. Os autores citaram que a maior parte das perguntas do SO, cerca de 90%, tem tempo médio de resposta de 11 minutos (MAMYKINA et al., 2011) (existem casos de usuários que recebem respostas em minutos, ou até mesmo em segundos após a sua postagem no site). O trabalho também mostrou que o SO possui mais de 12 milhões de visitantes e 135 milhões de visualizações de páginas todo mês (dados obtidos em 2012 pelos autores). Isto faz com que o SO seja uma fonte irresistível de informações que os desenvolvedores podem estar utilizando no processo de desenvolvimento ou manutenção de software.

O trabalho de Parnin et al. (PARNIN et al., 2012) mostra a importância das documentações de APIs feitas a partir de sites como o SO. Segundo os autores nas documentações oficiais das APIs, para uma certa funcionalidade há apenas um exemplo simples de código, sem um texto que explica melhor como utilizar a funcionalidade. Já nos sites como o SO há centenas de tópicos relacionados à funcionalidade que se deseje aprender. No trabalho também foi feito um estudo para viabilizar o uso do SO para obter exemplos e explicações sobre APIs. Os resultados mostraram que 87% das classes da API do Android são referenciadas no SO, o que motiva a construção de documentações a partir das informações contidas neste site.

Robillard (ROBILLARD, 2009) fez um estudo onde analisou as principais dificuldades que desenvolvedores de software têm ao aprender novas APIs. O estudo foi feito com 80 profissionais de software da Microsoft através de preenchimento de questionários e

realização de entrevistas. Entre estes profissionais, 78% disseram aprender sobre APIs lendo a documentação oficial, 55% através do uso de exemplos de código, 34% através de experimentação com a API, 30% através da leitura de artigos e 29% através de consulta a colegas de trabalho. Segundo o autor, os principais obstáculos na aprendizagem da API são os recursos disponíveis, como por exemplo, a documentação. Isto implica em esforços para tentar melhorar estes recursos.

Souza et al. (SOUZA; CAMPOS; MAIA, 2014a) propuseram uma abordagem semiautomática para construir livros de receitas para APIs através de informações disponíveis no SO. Esta abordagem utiliza a técnica de recuperação de informação LDA para encontrar temas de uma API, e através destes são identificados os potenciais capítulos do livro de receitas.

No trabalho de Head et al. (HEAD et al., 2015), os autores propuseram rotinas para linguagens específicas chamado Tutorons, que geram automaticamente contexto relevante, a pedido de micro explicações de código. Tutorons detecta código explicável em uma página web, em seguida faz uma análise no código e gera explicações em linguagem natural. Através de um estudo, os autores mostraram que as explicações geradas pelo Tutorons podem reduzir a necessidade de consultar a documentação em tarefas de modificação do código.

Treude e Robillard (TREUDE; ROBILLARD, 2016) apresentaram uma abordagem para aumentar automaticamente a documentação de um tipo particular de API com informações do Stack Overflow. No trabalho os autores apresentam SISE, uma nova abordagem baseada em aprendizagem de máquina que usa como características sentenças: sua formatação, sua pergunta, sua resposta, seus autores, *tags* e semelhança de uma frase que corresponde à documentação da API. Com o SISE, os autores alcançaram uma precisão de 0,64 e uma cobertura de 0,7 no conjunto de desenvolvimento. Em um estudo comparativo com oito desenvolvedores de software, eles relataram que o SISE resultou no maior número de frases que foram consideradas úteis para adicionar informações não encontradas na documentação da API.

Montandon et al. (MONTANDON et al., 2013) desenvolveram o APIMiner, que tem o objetivo de aumentar a documentação de APIs para Java com exemplos concretos de uso. Uma versão do APIMiner para a API Android possibilitou a extração de 73.732 exemplos de código-fonte a partir de 103 projetos *open-source*. Neste trabalho o objetivo foi melhorar a documentação já existente, e apenas para a linguagem Java.

Henb et al. (HENB; MONPERRUS; MEZINI, 2012) propuseram uma técnica semiautomática para construir um FAQs (*Frequently Asked Questions*) a partir de dados disponíveis em sites de discussão sobre software e fóruns.

Dekel e Herbsleb (DEKEL; HERBSLEB, 2009) implementaram um *plugin* chamado eMoose para o Eclipse que destaca no código-fonte métodos que estão sendo utilizados e que possuem diretivas (regras ou ressalvas existentes na documentação da API) na

sua documentação. O objetivo do trabalho é fazer com que os desenvolvedores possam examinar fragmentos de código-fonte consciente das diretivas associadas àquele trecho. Isto pode evitar erros de invocação, além de auxiliar os desenvolvedores na aprendizagem da API a partir de exemplos de código-fonte.

Kim et al. (KIM et al., 2009) apresentaram uma técnica para incrementar de forma automática a documentação de APIs com exemplos de código-fonte. Segundo os autores o conteúdo da maioria das documentações de APIs não possui código-fonte o suficiente. Os resultados mostraram que apenas 2% das classes (de mais de 27.000) de APIs dos Javadocs possuem exemplos de código. Isto faz com que desenvolvedores busquem na Internet código-fonte que os auxiliam na aprendizagem das APIs. Os autores alertam sobre a quantidade de tempo gasta no processo de encontrar um conteúdo na Internet que possa auxiliar os desenvolvedores no entendimento da funcionalidade desejada.

Long et al. (LONG; WANG; CAI, 2009) desenvolveram o Altair, uma ferramenta que gera automaticamente referências cruzadas de funções de APIs. A ferramenta realiza uma análise estática para extrair informações estruturais do código-fonte, e a partir desses dados calcula a sobreposição de pares, ou seja, encontram as duplas de funções que estão relacionadas entre si.

Dagenais e Robillard (DAGENAIS; ROBILLARD, 2014) propuseram AdDoc, uma técnica que detecta automaticamente os padrões de documentação, isto é, conjuntos coerentes de elementos de código que estão documentados juntos.

Kim et al. (KIM et al., 2013) propuseram um sistema de recomendação de exemplo de códigos que retorna documentos para API extraídos na Web. Os resultados da avaliação mostram que a abordagem fornece exemplos de código com alta precisão e aumenta a produtividade do programador.

Robillard e Chhetri (ROBILLARD; CHHETRI, 2014) propuseram detectar e recomendar fragmentos de documentação de API potencialmente importantes para um programador que já decidiu usar um dado elemento de API. Os autores categorizaram fragmentos de texto na documentação de API baseando se eles contêm informações que são indispensáveis, valiosas, ou nenhuma das opções. A partir dos fragmentos que contêm um conhecimento, eles extraíram padrões de palavras, e usaram esses padrões para encontrar automaticamente novos fragmentos que contêm conhecimentos semelhantes em uma documentação não analisada.

Chen e Zhang (CHEN; ZHANG, 2014) propuseram conectar a documentação oficial de API com a documentação informal através da captura do comportamento de desenvolvedores que navegam na Web.

Zhong e Su (ZHONG; SU, 2013) introduziram DOCREF, uma abordagem que combina técnicas de processamento de linguagem natural e análise de código para detectar e reportar inconsistências na documentação. Os autores utilizaram com sucesso o DOCREF para detectar mais de mil erros de documentação.

Pandita et al. (PANDITA et al., 2012) propuseram inferir especificações formais de texto de linguagem natural de documentações de API.

Petrosyan et al. (PETROSYAN; ROBILLARD; MORI, 2015) propuseram uma abordagem para descobrir seções de tutorial que explicam um determinado tipo de API. Os autores classificaram fragmentados de seções de tutorial usando classificação de textos supervisionado, baseado em características linguísticas e estruturais. Eles foram capazes de alcançar alta precisão e *recall* em diferentes tutoriais.

Capítulo 7

# Conclusão

Neste trabalho propomos quatro metodologias diferentes para a geração de tutoriais para uma dada API a partir do conteúdo do Stack Overflow. A principal contribuição em relação aos trabalhos relacionados é que este trabalho busca gerar uma documentação para uma dada API levando em consideração a complexidade de entendimento dos exemplos. Os resultados da avaliação dos tutoriais gerados pelas metodologias propostas neste trabalho mostraram a viabilidade para o uso de tutoriais gerados automaticamente.

Cada metodologia proposta implementou um grupo de características. Entre estas características estão filtros desenvolvidos neste trabalho, algoritmos de classificação, modelagem em tópicos e ranqueamento. As metodologias FHCBAC, GFC, GFHCBAC que utilizaram os filtros propostos tiveram resultados melhores do que a metodologia GP que utiliza apenas o mecanismo de pesquisa do Google. Este mecanismo não leva em consideração a complexidade dos exemplos relacionados ao desenvolvimento de software. Isto sugere a importância do desenvolvimento contínuo de filtros especializados para encontrar conteúdos relacionados ao desenvolvimento de software. Entre as quatro metodologias, a GFHCBAC foi a que teve os melhores resultados gerais. De certa forma isto é explicado pelo fato que ela implementa de alguma maneira as características das demais metodologias, ou seja, utiliza a metodologia FHCBAC para gerar um tutorial, depois utiliza o mecanismo de busca do Google para encontrar os exemplos de criação. Cabe ressaltar que esta metodologia não foi a mais eficiente em relação à coesão do capítulo, onde o uso nativo do Google obteve melhores resultados.

Os resultados da avaliação da comparação do tutorial gerado pela metodologia GFHCBAC com a documentação oficial para a API Android mostraram que na maioria dos casos, os alunos tiveram melhores desempenhos nas tarefas quando eles utilizaram o tutorial proposto neste trabalho. Os principais motivos do baixo desempenho dos alunos nas tarefas utilizando a documentação oficial da API foram devido à falta de exemplos de uso nesta documentação, linguagem de difícil compreensão, organização confusa e sua difícil utilização. Algumas desvantagens mencionadas a respeito do tutorial proposto foram a falta de explicação do que fazem as classes ou métodos encontrado nos exemplos

do tutorial, e a falta de uma barra de pesquisa no tutorial.

Os desenvolvedores, principalmente novatos, poderão utilizar os tutoriais gerados pelas metodologias propostas, principalmente pela metodologia GFHCBAC que se mostrou ser a mais didática nas avaliações, para aprender uma nova tecnologia (APIs, linguagens de programação, bibliotecas, etc.). Desenvolvedores experientes poderiam utilizar a parte avançada dos tutoriais para aprofundar seus conhecimentos. Como foi discutido, geralmente as documentações oficiais carecem de exemplos de uso das partes da API. Não é incomum, as documentações oficiais não possuírem exemplos simples contendo um código-fonte didático.

## 7.1 Trabalhos Futuros

Como trabalho futuro, sugere-se a criação de um portal de tutoriais de APIs que permita a avaliação em larga escala, tanto em diversidade de APIs quanto na diversidade de avaliadores. O usuário também será capaz de gerar seus próprios tutoriais através do portal, para isto bastará informar o nome da API em um campo de texto no portal. Também como trabalho futuro, pretendemos trabalhar nas limitações do trabalho desenvolvido nesta dissertação, para tentar lidar com estas limitações serão implementados nas metodologias de geração de tutoriais o seguinte:

**Utilizar técnicas de processamento de linguagem natural para encontrar exemplos mais simples e didáticos:** Uma das desvantagens identificadas nos resultados deste trabalho foi à questão das linguagens expressas na documentação oficial da API. Para um desenvolvedor iniciante, uma linguagem formal ou científica pode dificultar a aprendizagem de uma nova tecnologia (Linguagens de programação, APIs, etc.). Levando em consideração este problema, serão aplicadas técnicas de processamento de linguagem natural para encontrar exemplos mais simples e didáticos, facilitando o entendimento de desenvolvedores iniciantes.

**Inserir links nos tipos da API (Classes, Interfaces, etc.) encontrados no código fonte dos exemplos do tutorial:** Uma outra desvantagem identificada nos resultados deste trabalho foi à questão da falta de explicação do que os tipos da API fazem. Para resolver este problema serão inseridos links nos tipos da API encontrados no código-fonte dos exemplos do tutorial, desta maneira, quando os usuários clicarem nestes links, eles serão direcionados para a documentação oficial da API, obtendo mais fontes de informações sobre os tipos das APIs.

**Implementar uma barra de pesquisa no tutorial:** Na avaliação deste trabalho alguns avaliadores também questionaram a falta de uma barra de pesquisa no tutorial. Levando em consideração esta questão, será implementado uma barra de pesquisa nos tutoriais das APIs.

# Referências

BURGES, C. J. C. **From RankNet to LambdaRank to LambdaMART: An Overview**. [S.l.], 2010.

CAMPOS, E. C.; MAIA, M. de A. Automatic Categorization of Questions from Q&A Sites. In: **Proc. of the 29$^{th}$ Annual ACM Symposium on Applied Computing**. New York, NY, USA: ACM, 2014. (SAC '14), p. 641–643.

CHAPELLE, O.; CHANG, Y.; LIU, T. (Ed.). **Proceedings of the Yahoo! Learning to Rank Challenge**, v. 14 de **JMLR Proceedings**, (JMLR Proceedings, v. 14). [S.l.]: JMLR.org, 2011.

CHEN, C.; ZHANG, K. Who asked what: integrating crowdsourced FAQs into API documentation. In: JALOTE, P.; BRIAND, L. C.; HOEK, A. van der (Ed.). **ICSE Companion**. [S.l.]: ACM, 2014. p. 456–459.

DAGENAIS, B.; ROBILLARD, M. P. Using Traceability Links to Recommend Adaptive Changes for Documentation Evolution. **IEEE Trans. Software Eng.**, v. 40, n. 11, p. 1126–1146, 2014.

DEKEL, U.; HERBSLEB, J. D. Improving API Documentation Usability with Knowledge Pushing. In: **Proc. of the 31$^{st}$ Intl. Conf. on Software Engineering**. Washington, DC, USA: IEEE Computer Society, 2009. (ICSE '09), p. 320–330.

HEAD, A. et al. Tutorons: Generating context-relevant, on-demand explanations and demonstrations of online code. In: **Proc. of the 2015 IEEE Symposium on Visual Languages and Human-Centric Computing**. Atlanta, GA, USA: IEEE, 2015. (VL/HCC), p. 3–12.

HENB, S.; MONPERRUS, M.; MEZINI, M. Semi-automatically Extracting FAQs to Improve Accessibility of Software Development Knowledge. In: **Proc. of the 34$^{th}$ Intl. Conf. on Software Engineering**. Piscataway, NJ, USA: IEEE Press, 2012. (ICSE '12), p. 793–803.

HOTHORN, T. et al. Implementing a Class of Permutation Tests: The coin Package. **Journal of Statistical Software**, v. 28, n. 8, p. 1–23, 2008.

KIM, J. et al. Adding Examples into Java Documents. In: **Proc. of the 2009 IEEE/ACM Intl. Conf. on Automated Software Engineering**. Washington, DC, USA: IEEE Computer Society, 2009. (ASE '09), p. 540–544.

_____. Enriching Documents with Examples: A Corpus Mining Approach. **ACM Trans. Inf. Syst.**, ACM, New York, NY, USA, v. 31, n. 1, p. 1:1–1:27, jan. 2013.

LONG, F.; WANG, X.; CAI, Y. Api Hyperlinking via Structural Overlap. In: **Proc. of the the 7$^{th}$ Joint Meeting of the European Software Engineering Conf. and the ACM SIGSOFT Symposium on The Foundations of Software Engineering**. New York, NY, USA: ACM, 2009. (ESEC/FSE '09), p. 203–212.

MAMYKINA, L. et al. Design Lessons from the Fastest Q&A Site in the West. In: **Proc. of the SIGCHI Conf. on Human Factors in Computing Systems**. New York, NY, USA: ACM, 2011. (CHI '11), p. 2857–2866.

MCCALLUM, A. K. **MAchine Learning for LanguagE Toolkit**. 2016. Disponível em: <http://mallet.cs.umass.edu/>.

MONTANDON, J. E. et al. Documenting APIs with examples: Lessons learned with the APIMiner platform. In: **Proc. of the 20$^{th}$ Conf. on Reverse Engineering (WCRE)**. [S.l.: s.n.], 2013. p. 401–408.

NASEHI, S. M. et al. What Makes a Good Code Example? A Study of Programming Q&A in StackOverflow. In: **Proc. of the 2012 IEEE Intl. Conf. on Software Maintenance - ICSM'12**. Washington, DC, USA: IEEE Computer Society, 2012. p. 25–34.

PANDITA, R. et al. Inferring Method Specifications from Natural Language API Descriptions. In: **Proc. of the 34$^{th}$ Intl. Conf. on Software Engineering**. Piscataway, NJ, USA: IEEE Press, 2012. (ICSE '12), p. 815–825.

PARNIN, C. et al. Crowd Documentation: Exploring the Coverage and the Dynamics of API Discussions on Stack Overflow. In: . [S.l.]: Georgia Institute of Technology, Tech. Rep, 2012.

PETROSYAN, G.; ROBILLARD, M. P.; MORI, R. D. Discovering Information Explaining API Types Using Text Classification. In: **Proc. of the 37$^{th}$ Intl. Conf. on Software Engineering - Volume 1**. Piscataway, NJ, USA: IEEE Press, 2015. (ICSE '15), p. 869–879.

PONZANELLI, L.; BACCHELLI, A.; LANZA, M. Leveraging Crowd Knowledge for Software Comprehension and Development. In: **Proc. of the 17$^{th}$ on European Conference on Software Maintenance and Reengineering (CSMR)**. [S.l.: s.n.], 2013. p. 57–66.

ROBILLARD, M. P. What Makes APIs Hard to Learn? Answers from Developers. **IEEE Softw.**, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 26, n. 6, p. 27–34, nov. 2009.

ROBILLARD, M. P.; CHHETRI, Y. B. Recommending reference API documentation. **Empirical Software Engineering**, v. 20, n. 6, p. 1558–1586, 2014.

SOMMERVILLE, I. **Engenharia de Software**. 8$^{th}$. ed. São Paulo: Pearson Addison-Wesley, 2007.

SOUZA, L. B. L.; CAMPOS, E. C.; MAIA, M. A. On the Extraction of Cookbooks for APIs from the Crowd Knowledge. In: **Proc. of the 28$^{th}$ Brazilian Symposium on Software Engineering - SBES'2014**. Maceió, Brazil: [s.n.], 2014. (SBES'2014), p. 10pp.

SOUZA, L. B. L. de; CAMPOS, E. C.; MAIA, M. d. A. Ranking Crowd Knowledge to Assist Software Development. In: **Proc. of the 22$^{Nd}$ Intl. Conf. on Program Comprehension**. New York, NY, USA: ACM, 2014. (ICPC 2014), p. 72–82.

STEYVERS, M.; GRIFFITHS, T. Probabilistic Topic Models. Unpublished. 2007.

STOREY, M.-A. et al. The Impact of Social Media on Software Engineering Practices and Tools. In: **Proc. of the FSE/SDP Workshop on Future of Software Engineering Research**. New York, NY, USA: ACM, 2010. (FoSER '10), p. 359–364.

STYLOS, J.; MYERS, B. A.; YANG, Z. Jadeite: Improving API Documentation Using Usage Information. In: **CHI '09 Extended Abstracts on Human Factors in Computing Systems**. New York, NY, USA: ACM, 2009. (CHI EA '09), p. 4429–4434.

SUBRAMANIAN, S.; INOZEMTSEVA, L.; HOLMES, R. Live API Documentation. In: **Proc. of the 36$^{th}$ Intl. Conf. on Software Engineering**. New York, NY, USA: ACM, 2014. (ICSE 2014), p. 643–652.

TREUDE, C. et al. **Programming in a Socially Networked World: the Evolution of the Social Programmer**. 2012. The Future of Collaborative Software Development (FutureCSD).

TREUDE, C.; ROBILLARD, M. P. Augmenting API Documentation with Insights from Stack Overflow. In: **Proc. of the 38$^{st}$ Intl. Conf. on Software Engineering**. Austin, TX, USA: [s.n.], 2016. (ICSE '16).

WU, Q. et al. Adapting Boosting for Information Retrieval Measures. **Inf. Retr.**, Kluwer Academic Publishers, Hingham, MA, USA, v. 13, n. 3, p. 254–270, jun. 2010.

XIE, T.; PEI, J. MAPO: Mining API Usages from Open Source Repositories. In: **Proc. of the 2006 Intl. Workshop on Mining Software Repositories**. New York, NY, USA: ACM, 2006. (MSR '06), p. 54–57.

ZHONG, H.; SU, Z. Detecting API Documentation Errors. In: **Proc. of the 2013 ACM SIGPLAN Intl. Conf. on Object Oriented Programming Systems Languages & Applications**. New York, NY, USA: ACM, 2013. (OOPSLA '13), p. 803–816.

# Anexos

ANEXO **A**

# Formulário de Avaliação dos Tutoriais

**Afirmação 01) O tutorial é organizado.** Alguns critérios para avaliar a organização:

1. O sumário do tutorial está numerado.
2. Os sumários dos capítulos possuem os títulos de todos os exemplos encontrados em um determinado capítulo.
3. Os exemplos encontrados no tutorial estão organizados em pergunta seguido da solução.

( ) Concordo fortemente.
( ) Concordo.
( ) Nem concordo e nem discordo.
( ) Discordo.
( ) Discordo fortemente.

**Afirmação 02) O tutorial é intuitivo.** Alguns critérios para avaliar se o tutorial é intuitivo:

1. O tutorial pode ser utilizado sem nenhum tipo de treinamento.
2. Sua utilização é autoexplicativa.

( ) Concordo fortemente.
( ) Concordo.
( ) Nem concordo e nem discordo.
( ) Discordo.
( ) Discordo fortemente.

**Afirmação 03) O tutorial possui uma navegabilidade adequada nos sumários.** Alguns critérios para avaliar a navegabilidade:

1. O sumário do tutorial possui links para todos os capítulos.
2. Em cada capítulo o sumário geral pode ser acessado, possibilitando a navegação

para outros capítulos.

3. Cada capítulo possui seu próprio sumário com links para os exemplos do respectivo capítulo.

( ) Concordo fortemente.
( ) Concordo.
( ) Nem concordo e nem discordo.
( ) Discordo.
( ) Discordo fortemente.

**Afirmação 04) O tutorial segue uma ordem crescente de complexidade dos exemplos dos capítulos, ou seja, os exemplos mais simples aparecem antes dos exemplos mais complexos.**

( ) Concordo fortemente.
( ) Concordo.
( ) Nem concordo e nem discordo.
( ) Discordo.
( ) Discordo fortemente.

**Afirmação 05) A organização do tutorial com divisão em partes básica e avançada é adequada.**

( ) Concordo fortemente.
( ) Concordo.
( ) Nem concordo e nem discordo.
( ) Discordo.
( ) Discordo fortemente.

**Afirmação 06) A natureza dos exemplos é compatível com um tutorial. Em outras palavras, os exemplos explicam como utilizar funcionalidades da API, e não, por exemplo, como corrigir erros de programação.**

( ) Concordo fortemente.
( ) Concordo.
( ) Nem concordo e nem discordo.
( ) Discordo.
( ) Discordo fortemente.

**Afirmação 07) Os exemplos do tutorial possuem códigos-fonte didáticos.**

( ) Concordo fortemente.
( ) Concordo.
( ) Nem concordo e nem discordo.
( ) Discordo.
( ) Discordo fortemente.

**Afirmação 08) Os exemplos encontrados nos capítulos estão relacionados com o respectivo nome do capítulo.**

( ) Concordo fortemente.
( ) Concordo.
( ) Nem concordo e nem discordo.
( ) Discordo.
( ) Discordo fortemente.

**Escreva uma avaliação geral sobre o tutorial, explicando, por exemplo, o porquê você usaria ou não, recomendaria ou não, gostou ou não.**

**Sugestões.**

ANEXO B

# Documentos Utilizados no Segundo Experimento

## B.1 Tarefas Aplicadas Durante o Experimento

**Task 01)** Build an Android application that has a drop-down list (also known as spinner) containing the items: Abacaxi, Banana, Goiaba, Laranja and Maracujá. The figure below (Figura 11) illustrates what is asked in the exercise.

**Task 02)** Consider the following XML code defining an identifier for a Text View. **Layout XML file:** (O código XML encontra-se na Figura 12)

In the corresponding Java file, make a script to display the current date and time of the device using the Text View above. The figure below (Figura 13) illustrates what is asked in the exercise. Observe that when the application is executed the device time equals to the time in the Text View.

## B.2 Questionário de Avaliação da Documentação Oficial e do Tutorial gerado pela Metodologia GFHC-BAC

**1) A documentação é organizada.**
**Documentação Site Android**

( ) Concordo fortemente.
( ) Concordo.
( ) Nem concordo e nem discordo.
( ) Discordo.
( ) Discordo fortemente.

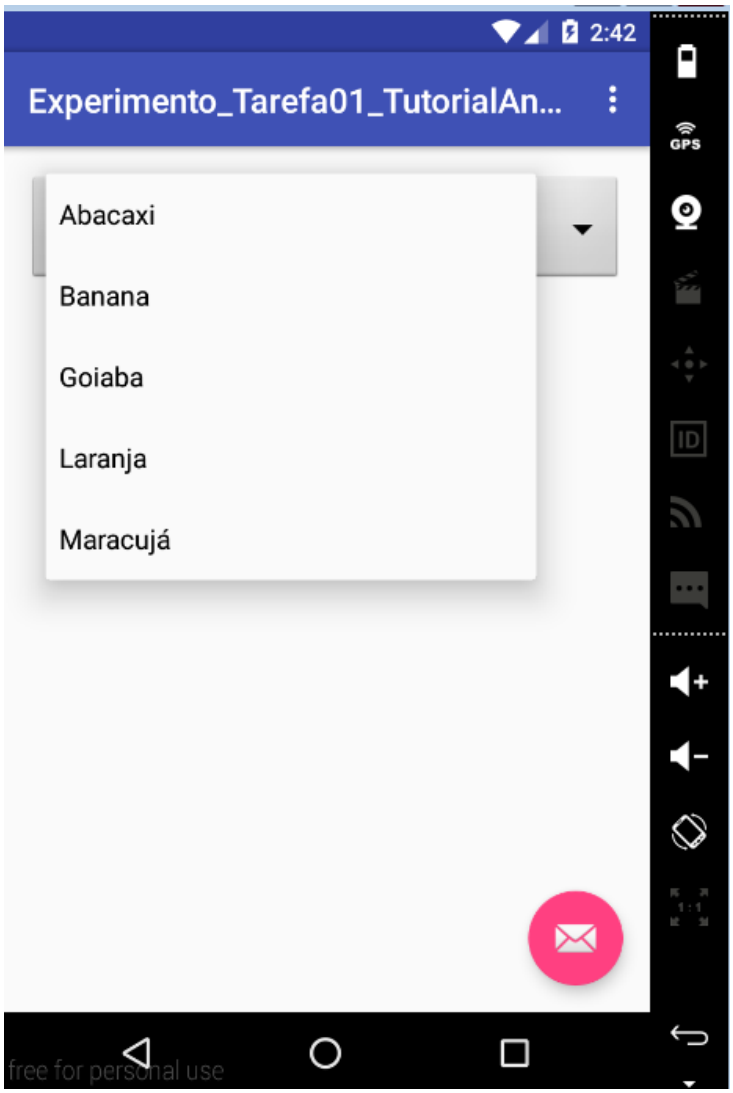


Figura 11 – Tarefa 01 aplicada no experimento.

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/textView01"
/>
```

Figura 12 – Código XML do arquivo de *Layout*

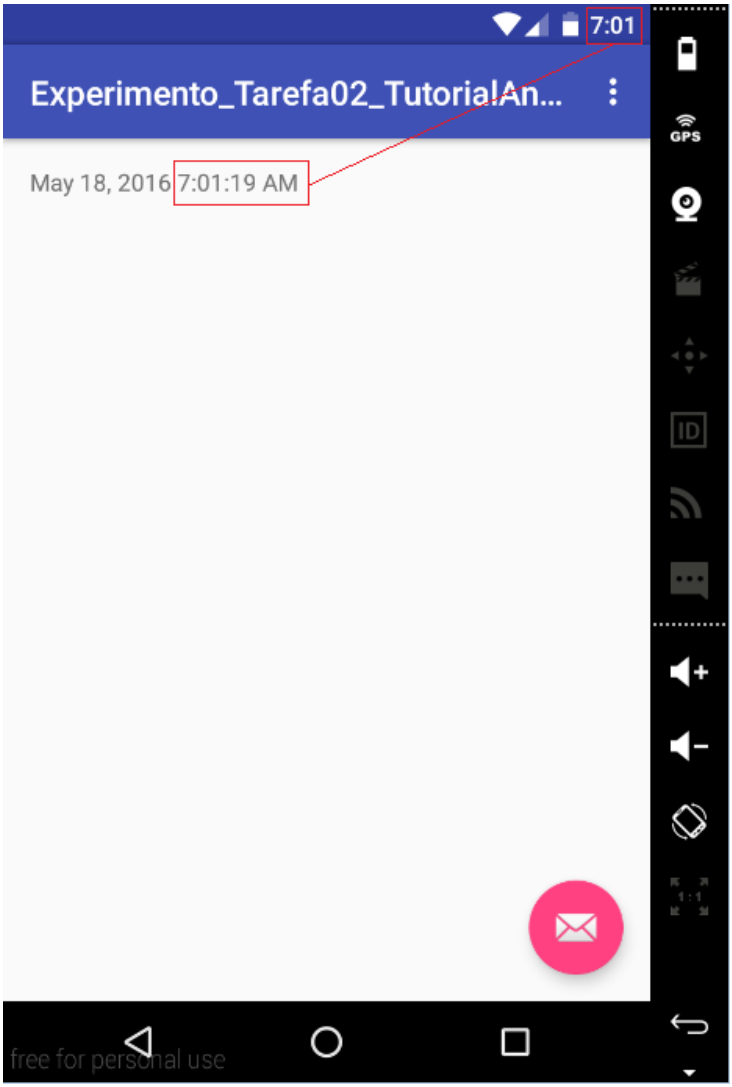


Figura 13 – Tarefa 02 aplicada no experimento.

**Documentação Green Version**

( ) Concordo fortemente.

( ) Concordo.

( ) Nem concordo e nem discordo.

( ) Discordo.

( ) Discordo fortemente.

**2) A documentação é intuitiva.** Alguns critérios para avaliar se a documentação é intuitiva.

1. Pode ser utilizada sem nenhum tipo de treinamento.
2. Sua utilização é autoexplicativa.

**Documentação Site Android**

( ) Concordo fortemente.

( ) Concordo.
( ) Nem concordo e nem discordo.
( ) Discordo.
( ) Discordo fortemente.

**Documentação Green Version**
( ) Concordo fortemente.
( ) Concordo.
( ) Nem concordo e nem discordo.
( ) Discordo.
( ) Discordo fortemente.

**3) A documentação possui boa navegabilidade.**
**Documentação Site Android**
( ) Concordo fortemente.
( ) Concordo.
( ) Nem concordo e nem discordo.
( ) Discordo.
( ) Discordo fortemente.

**Documentação Green Version**
( ) Concordo fortemente.
( ) Concordo.
( ) Nem concordo e nem discordo.
( ) Discordo.
( ) Discordo fortemente.

**4) Os exemplos da documentação possuem códigos-fonte didáticos.**
**Documentação Site Android**
( ) Concordo fortemente.
( ) Concordo.
( ) Nem concordo e nem discordo.
( ) Discordo.
( ) Discordo fortemente.

**Documentação Green Version**
( ) Concordo fortemente.
( ) Concordo.
( ) Nem concordo e nem discordo.

( ) Discordo.

( ) Discordo fortemente.

**5) Quais foram as maiores dificuldades encontradas na utilização da documentação e quais foram as vantagens?**

**Documentação Site Android**

**Documentação Green Version**

**6) Sugestões.**

**Documentação Site Android**

**Documentação Green Version**