Universidade Federal do Mato Grosso do Sul
Faculdade de Computação
FACOM

Investigação e elaboração de um classificador para arquivos de configuração em projetos Open-source: Relatório

Paulo Henrique Horta Borges - RGA 2019.1904.003-0
Prof. Hudson Silva Borges

Novembro de 2024

1 Introdução

No desenvolvimento de software moderno, é extremamente comum o uso de aplicações externas para possibilitar um desenvolvimento mais fluido e ágil. Essas ferramentas incluem sistemas de gerenciamento de projetos (como gerenciamento de repositórios, por exemplo, Git), ferramentas que fornecem uma funcionalidade pronta (como frameworks de desenvolvimento de software, por exemplo React-JS), entre outras.

Com isso, é usual que essas ferramentas permitam a customização das funcionalidades entregues, possibilitando um grande leque de possibilidades sem que o desenvolvedor precise manter diferentes versões. Essa customização pode ser observada em projetos de desenvolvimento de software por meio do uso de arquivos de configuração, que são arquivos cujo conteúdo contém parâmetros que alteram sua funcionalidade.

Essa configuração é um alvo recente de pesquisas devido à possibilidade de problemas de desenvolvimento, funcionamento e gerenciamento do projeto. Um dos possíveis causadores dessa problemática é o erro de configuração ou de gerenciamento de configuração no projeto, exacerbado pelo grande volume de arquivos de configuração presentes em projetos grandes.

Para realizar estudos mais profundos no assunto, é necessário realizar primeiro mais estudos sobre como esses arquivos podem se apresentar, como identificá-los, para posteriormente analisar sua influência. Uma possível dificuldade para este estudo é a falta de padronização presentes nestes arquivos, o que dificulta sua identificação no projeto.

Dessa forma, esta atividade tem como objetivo a construção de um *dataset* rotulado de arquivos em projetos de desenvolvimento de software, categorizando-os em arquivos de configuração ou não. Com isso, é possível desenvolver ferramentas automáticas para identificar esses arquivos no repositório, facilitando a análise em estudos futuros.

2 Metodologia

Baseado no objetivo, foram propostas três etapas para a construção do *dataset*, sendo elas:

- Seleção de repositórios open-source;
- Obtenção das informações de todos os arquivos presentes na branch principal do projeto;
- Classificação manual de arquivos em "arquivos de configuração"e "não arquivos de configuração"

2.1 Seleção de projeto

Na primeira etapa, foi utilizada a plataforma GitHub para a seleção de projetos open-source, devido à sua ampla utilização em desenvolvimento colaborativo de software.

Em seguida, foi proposta a seleção de 50 projetos dentro da plataforma. O critério de seleção foi a presença do projeto na categoria 'Trending' da plataforma, que exibe repositórios populares por período de tempo (diário, semanal ou mensal).

Com isso, foram selecionados os seguintes repositórios: *amaro*, *anything-llm*, *GDevelop*, *Mermaid*, *p5.js*, *social-app*, *Kactus*, *Tailwind*, *TensorflowJS uBlock Amphion*, *ToolJet*, *Skyvern*, *jellyfin*, *eliza*, *you-get*, *llama-recipes*, *ViMusic*, *lerobot*, *termux-app*, *storm*, *memos*, *lobe-chat*, *PowerShell*, *screenpipe*, *formbricks*, *crawl4ai*, *videos*, *kamal*, *OpenBB*, *mindcraft*, *maybe*, *kestra*, *platform*, *drawdb*, *zerox*, *cocos-engine*, *co-tracker*, *EasySpider*, *hono*, *Pumpkin*, *tauri*, *hyperswitch*, *deskflow*, *citra*, *imgui*, *qBittorent*, *openai-dotnet*, *UmbracoCMS* e *Bruce*.

Dentre os repositórios escolhidos, foi observada a seguinte distribuição de linguagem de programação principal utilizada: 19 projetos *TypeScript/JavaScript*, 11 *Python*, 5 *C++*, 4 *Rust*, 4 *C#*, 2 *Java*, 2 *Ruby*, 1 *C*, 1 *Kotlin* e 1 *Go*.

Nos repositórios selecionados também é possível observar cerca de 1.104.735 estrelas no momento de coleta de dados, resultando em uma média de 22.094 estrelas por repositório analisado. Dentre esses repositórios, o com maior número de estrelas foi o repositório *tauri*, que é uma *framework* de desenvolvimento de aplicações na linguagem Rust, que possui cerca de 84500 estrelas na plataforma.

Um tópico de tendência observado nos projetos coletados durante o período foi o uso/envolvimento de Inteligência Artificial nos projetos. Dos 50 projetos analisados, 10 deles apresentavam foco majoritariamente no uso da tecnologia que, em sua maioria, utilizam as linguagens *Python* e *TypeScript/JavaScript*.Da mesma forma, foram observados também 8 projetos que focavam na facilitação de desenvolvimento de aplicações, como por exemplo as *frameworks* hono (framework de desenvolvimento da linguagem JavaScript) e ToolJet (framework que permite desenvolvimento simples de aplicações).

2.2 Extração de arquivos

Em seguida, foi realizada a extração de arquivos de cada um dos repositórios e suas respectivas informações, como por exemplo: Tamanho, Nome, Caminho e Data da primeira aparição do arquivo no projeto. Para a ex-

tração desses arquivos, foi utilizada a API REST da plataforma, obtendo-se os dados de forma automatizada pela utilização de um *script* desenvolvido na linguagem Python, resumindo-os em um arquivo JSON. Esse processo se deu nos períodos de 06/09/2024 até 16/09/2024 para os dez primeiros projetos e entre 17/10/2024 até 01/11/2024 para os 40 projetos subsequentes. Quando necessário a escolha de *branch*, foi favorecido a branch principal de desenvolvimento (usualmente nomeada *"dev"*, *"main"* ou *"master"*).

2.3 Classificação de arquivos de configuração

Por fim, para cada um dos projetos, deu-se início ao processo de classificação dos arquivos. Para isto, em primeira etapa foi realizada a extração do nome de cada um dos arquivos do projeto e exportado para um arquivo de texto por meio de um *script* na linguagem JavaScript.

Após exportado em arquivo de texto, os arquivos foram separados em "arquivos de configuração"e "não arquivos de configuração"nesse arquivo e, posteriormente, adicionou-se um novo arquivo JSON e uma planilha, contendo exclusivamente as informações dos arquivos de configuração, conforme mostrados pela figura 1.

file_path	is_config_file	file_size	first_appearance
.github/workflows/ci.yml	YES	1401	2023-01-20T09:09:37Z
.github/workflows/docker-publish.yml	YES	1365	2023-03-04T07:23:24Z
.gitignore	YES	59	2023-01-07T14:32:25Z

Figura 1: Amostra de planilha gerada para o projeto *Kamal*

2.4 Exportação de Resultados

Após realizada a classificação, foi realizado o desenvolvimento de um *script* na linguagem Python para a exportação automática dos resultados em planilhas. Com isso, realizaram-se análises sobre as características dos arquivos, verificando a porcentagem que atendia a determinados critérios e distinguindo os arquivos de configuração dos demais. Os critérios selecionados para análise foram:

- Tamanho do arquivo: Verificação de arquivos com tamanho dentro de intervalos especificos (100B, 500B, 1000B, 2000B, 5000B e 10000B);
- Extensão do arquivo: Verificação de arquivos com tamanho dentro de intervalos específicos (ou sem extensão);

Criteria	Number of Config Files	Number of Non-Config Files	Percentage of Config Files over Total
lock	121	2	98.3739837398374%
toml	224	8	96.55172413793103%
gradle	81	5	94.186046511162791%
properties	44	3	93.61702127659575%
conf	10	1	90.9090909090909%
config	10	3	76.923076923076923%
yml	707	244	74.34279705573081%
cjs	7	4	63.63636363636363%
In the root directory	451	355	55.955334987593055%

Figura 2: Exemplo dos critérios aplicados

- Caminho do arquivo: Verificação da porcentagem de arquivos de configuração presentes dentro e fora do diretório raiz.

Dessa forma, foi realizada a análise conforme os critérios apresentados, separando, em cada um dos critérios a quantidade de arquivos de configuração e sua porcentagem, conforme mostrados na Figura 2.

3 Resultados

No total, foram analisados 96044 arquivos em 50 projetos, com uma média de 1920 arquivos analisados por projeto. O projeto com maior número de arquivos foi o repositório *zerox*, com 49 arquivos, que é uma ferramenta OCR (Reconhecimento ótico de caracteres) para a obtenção de texto vindo de imagens. Do outro lado do espectro, foram observados 16531 arquivos no repositório ToolJet, que se trata de uma *framework* com o intuito de facilitar o desenvolvimento e o uso de aplicações e ferramentas, reduzindo a quantidade de código

Com esse estudo, também foi possível perceber uma alta concentração de arquivos de configuração em extensões específicas, como por exemplo "toml", ".conf"e "yml"nos projetos escolhidos. A extensão TOML é um padrão de extensão criado e adotado especificamente para a configuração de software, enquanto "yml"e ".conf"são padrões genérico que podem ser adotados para representar e são utilizados usualmente para representar fluxos de CI/CD e configurações presentes em sistemas operacionais UNIX.

Outro aspecto a se notar é a alta concentração de arquivos de configuração em projetos que incluem diversos exemplos de uso do próprio software. Nesses casos, é extremamente comum encontrar múltiplos "tsconfig.json". Um exemplo desse comportamento pode ser encontrado no repositório *platform*, que possui o mesmo intuito que o repositório zerox citado anteriormente e, possui 351 arquivos tsconfig.json para configuração da linguagem TypeScript no projeto.

Dessa forma, como os projetos foram obtidos na plataforma GitHub, observa-se em todos eles a presença de arquivos de configuração específicos da plataforma, que aumentam consideravelmente o número total de arquivos de configuração. Um exemplo é o uso do arquivo ".gitignore"para excluir arquivos sincronizados pela plataforma.

Outro exemplo de arquivo extremamente comum são os arquivos de configuração de fluxos de Integração Contínua (CI) e Entrega Contínua (CD). Esses arquivos são responsáveis pela padronização de procedimentos padrões (testes de qualidades, compilação do projeto, distribuição do projeto, etc), providenciando controle mais preciso sobre as *releases* do projeto.