PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Instituto de Ciências Exatas e Informática - Sistemas de Informação
Disciplina: Laboratório de Experimentação de Software
Professora: Aline Norberta de Brito

Integrantes
Matheus Andrade
Rafael Chequer
Samara Martins
Victor Fonseca

Relatório Final

Belo Horizonte
2023

Hipóteses Informais

RQ 01. Sistemas populares são maduros/antigos?

Repositórios tendem a acumular estrelas ao longo do tempo ao ganharem reconhecimento e usabilidade. Sendo assim acredita-se que a idade do repositório seja maior para sistemas com mais estrelas.

RQ 02. Sistemas populares recebem muita contribuição externa?

Por serem populares, esses sistemas possuem grande volume de usuários e visibilidade, sendo assim devem receber um volume significativo de contribuições externas. Assume-se que haja maior quantidade de pull requests solicitadas e aceitas.

RQ 03. Sistemas populares lançam releases com frequência?

Sistemas populares provavelmente possuem alta usabilidade, o que implica na necessidade de correções de bugs, inclusão de novas funcionalidades, na evolução do sistema como todo. Portanto, espera-se que o total de releases seja maior nesses sistemas.

RQ 04. Sistemas populares são atualizados com frequência?

Devido a uma base de usuários mais ativa e ampla a identificação e relato de erros tende a ser mais comum. Dessa maneira, é pressuposto, que sistemas populares tenham maior taxa de atualização.

RQ 05. Sistemas populares são escritos nas linguagens mais populares?

Linguagens mais populares possuem, obviamente, uma comunidade de desenvolvedores mais ampla e tornam-se mais acessíveis ao público. Sendo os repositórios mais populares reflexos da resposta da comunidade, espera-se que os mesmos tenham como linguagem principal linguagens mais populares.

RQ 06. Sistemas populares possuem um alto percentual de issues fechadas?

Tendo uma comunidade mais ativa e ampla, sistemas mais populares tendem a ter um percentual de issues fechadas maior devido a incidência de tentativas de resoluções dos problemas.

Metodologias

Aqui está uma descrição detalhada das metodologias utilizadas para responder a cada questão de pesquisa

RQ1: Sistemas populares são maduros/antigos?

- **Método:**
  - A data de criação do repositório foi comparada com a data atual para determinar a idade do repositório.
  - A biblioteca datetime do Python foi utilizada para calcular a diferença entre as datas.
  - Repositórios com mais de 1000 dias foram considerados "maduros/antigos".

- **Limitações:**
  - A idade do repositório é apenas um indicador de maturidade.
  - Repositórios mais novos podem ser considerados maduros se tiverem uma comunidade ativa e um código bem documentado.

RQ2: Sistemas populares recebem muita contribuição externa?

- **Método:**
  - O número total de pull requests aceitos foi utilizado como indicador de contribuição externa.
  - A API do GitHub foi utilizada para recuperar a quantidade de pull requests.

- **Limitações:**
  - O número de pull requests não é o único indicador de contribuição externa.
  - Forks e issues também podem ser considerados.

RQ3: Sistemas populares lançam releases com frequência?

- **Método:**
  - O número total de releases foi utilizado como indicador da frequência de releases.

- ❖ A API do GitHub foi utilizada para recuperar a quantidade de releases.

- **Limitações:**
- ❖ O número de releases não é o único indicador da frequência de releases.
- ❖ A importância das releases (maiores vs. menores) também pode ser considerada.

RQ4: Sistemas populares são atualizados com frequência?

- **Método:**
- ❖ A diferença entre a data da última atualização e a data atual foi utilizada como indicador da frequência de atualização.
- ❖ A biblioteca datetime do Python foi utilizada para calcular a diferença entre as datas.

- **Limitações:**
- ❖ A frequência de atualização pode ser afetada por diversos fatores, como a natureza do projeto e a atividade da comunidade.

RQ5: Sistemas populares são escritos nas linguagens mais populares?

- **Método:**
- ❖ A linguagem principal do repositório foi identificada.
- ❖ A popularidade da linguagem foi considerada com base em pesquisas e rankings online.

- **Limitações:**
- ❖ A popularidade da linguagem pode mudar com o tempo.
- ❖ A escolha da linguagem pode ser influenciada por diversos fatores, como a experiência dos desenvolvedores e as necessidades do projeto.

RQ6: Sistemas populares possuem um alto percentual de issues fechadas?

- **Método:**
- ❖ A razão entre o número de issues fechadas e o número total de issues foi utilizada como indicador da eficiência da comunidade em resolver problemas.
- ❖ A API do GitHub foi utilizada para recuperar o número de issues abertas e fechadas.

Análise dos Resultados de Pesquisa

Com base nos dados obtidos, podemos analisar as tendências gerais em relação às características de repositórios populares do GitHub

EXPERIMENTAÇÃO

MÉTRICAS	MÉDIA	MEDIANA	MODA
IDADE DO REPOSITÓRIO (DIAS)	2.821,91	2.873,00	2.784
TOTAL DE PULL REQUESTS ACEITOS	2.788,52	544,00	0
TOTAL DE RELEASES	84,63	25,5	0
TEMPO DESDE A ÚLTIMA ATUALIZAÇÃO (DIAS)	-0,65	-1,0	-1
LINGUAGEM PRINCIPAL	-	-	JAVA SCRIPT
PORCENTAGEM DE ISSUES FECHADAS	0,76	0,8545315327857441	0,0

RQ1: Sistemas populares são maduros/antigos?

A idade dos repositórios analisados variou consideravelmente, com alguns sendo bem estabelecidos (mais de 1000 dias) e outros relativamente novos. A média de idade foi de 2821.91 dias e a mediana de 2873.0 dias.

RQ2: Sistemas populares recebem muita contribuição externa?

A quantidade de pull requests aceitos variou entre os repositórios, com alguns tendo alto nível de contribuição externa (média de 2788.52) e outros menos (mediana de 544.5).

RQ3: Sistemas populares lançam releases com frequência?

O número de releases variou entre os repositórios, com alguns lançando novas versões com frequência (média de 84.63) e outros menos frequentemente (mediana de 25.5).

RQ4: Sistemas populares são atualizados com frequência?

A frequência de atualização variou entre os repositórios, com alguns sendo atualizados constantemente (média de -0.65 dias) e outros com atualizações menos recentes (mediana de -1.0 dia).

RQ5: Sistemas populares são escritos nas linguagens mais populares?

A linguagem de programação mais frequente foi JavaScript.

RQ6: Sistemas populares possuem um alto percentual de issues fechadas?

A proporção de issues fechadas variou entre os repositórios, com alguns tendo comunidades ativas que resolvem problemas rapidamente (média de 0.76) e outros com backlog maior de issues abertas (mediana de 0.8545).

Discussão dos Resultados

Nesta seção, aprofundaremos a análise dos resultados da pesquisa sobre repositórios populares do GitHub, comparando as hipóteses iniciais com os dados coletados e contextualizando-os com nuances e considerações adicionais.

RQ1: Idade do Repositório (dias)

Hipótese:

Repositórios populares tenderiam a ser mais antigos devido à acumulação de estrelas ao longo do tempo.

Resultados:

A idade dos repositórios variou consideravelmente, com uma média de 2821,91 dias, mediana de 2873 dias e moda de 2784 dias.

Observou-se uma ampla gama de idades, desde projetos iniciados recentemente até aqueles com mais de 10 anos de existência.

Conclusão:

A idade, por si só, não é um indicador definitivo de maturidade.

Repositórios mais novos podem ter comunidades ativas, código bem documentado e práticas de desenvolvimento modernas.

É importante considerar outros fatores, como a frequência de atualizações, a qualidade do código e o nível de engajamento da comunidade, ao avaliar a maturidade de um projeto.

RQ2: Total de Pull Requests Aceitos

Hipótese:

Devido à popularidade, espera-se um volume significativo de pull requests.

Resultados:

A quantidade de pull requests aceitos variou entre os repositórios, com uma média de 2788,52, mediana de 544,5 e moda de 0.

Alguns repositórios apresentaram milhares de pull requests, enquanto outros tiveram um número significativamente menor.

Conclusão:

O número de pull requests é um indicador da atividade da comunidade e do interesse no projeto.

No entanto, outros fatores, como a natureza do projeto, a rigidez do processo de revisão e a experiência dos colaboradores, também podem influenciar a quantidade de pull requests.

É importante analisar o contexto e a qualidade dos pull requests para avaliar o impacto real da contribuição externa.

RQ3: Total de Releases

Hipótese:

Sistemas populares, com alta usabilidade, tendem a ter releases frequentes para correções de bugs e novas funcionalidades.

Resultados:

O número de releases variou entre os repositórios, com uma média de 84,63, mediana de 25,5 e moda de 0.

Alguns repositórios lançaram novas versões com frequência, enquanto outros apresentaram um ritmo de lançamento mais lento.

Conclusão:

A frequência de releases pode ser um indicador da agilidade do projeto e da capacidade de entregar novas funcionalidades aos usuários.

No entanto, a importância das releases (maiores vs. menores) e o tipo de mudanças introduzidas também devem ser considerados.

Fatores como a natureza do projeto, a estabilidade do código e a necessidade de testes rigorosos podem influenciar o ritmo de lançamento

RQ4: Tempo Desde a Última Atualização (dias)

Hipótese:

Espera-se alta taxa de atualização devido à base de usuários ativa e ampla.

Resultados:

A frequência de atualização variou entre os repositórios, com uma média de -0,65 dias, mediana de -1,0 dia e moda de -1 dia.

A maioria dos repositórios apresentou atualizações recentes, evidenciando a atividade contínua dos desenvolvedores.

Conclusão:

A frequência de atualização pode ser um indicador da saúde do projeto e do compromisso dos desenvolvedores em mantê-lo atualizado.

No entanto, a natureza do projeto, a frequência de lançamento de novas funcionalidades e a necessidade de testes rigorosos podem influenciar a frequência de atualizações.

É importante analisar o tipo de mudanças realizadas nas atualizações para avaliar o seu impacto real no projeto.

RQ5: Linguagem Principal

Hipótese:

Espera-se que os repositórios utilizem linguagens como Python, JavaScript e TypeScript, devido à sua popularidade e comunidade ampla.

Resultados:

A linguagem mais frequente foi JavaScript

Observou-se uma diversidade de linguagens em uso, incluindo linguagens menos populares como Go, Rust e Ruby.

Conclusão:

A escolha da linguagem de programação é influenciada por diversos fatores, como a experiência dos desenvolvedores, as necessidades do projeto e a disponibilidade de bibliotecas e frameworks.

A popularidade da linguagem pode ser um indicador da comunidade disponível para suporte e desenvolvimento, mas não garante o sucesso do projeto.

É importante considerar as características específicas da linguagem escolhida e sua adequação ao projeto em questão.