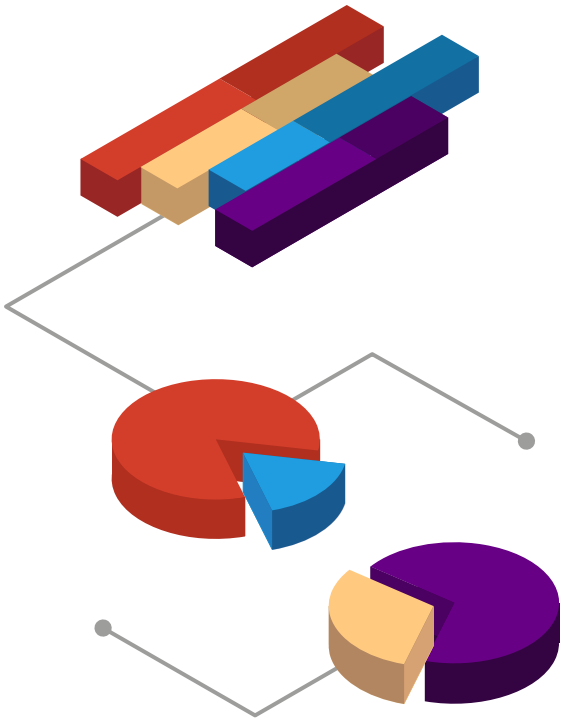
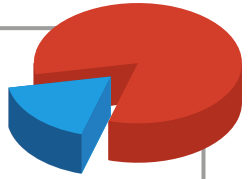
PUC Minas

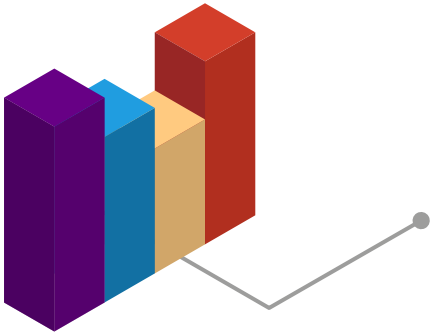
Análise de Características de Repositórios Populares do GitHub



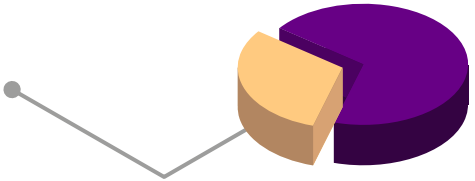
Ana Carolina Corrêa, Caio Elias, Henrique Diniz e Maria Clara Santos



Sumário



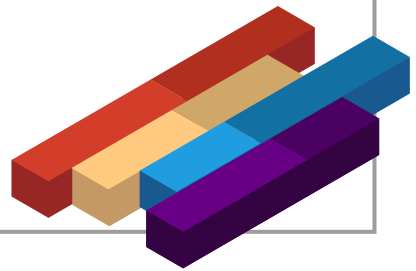
01



Introdução & Objetivo

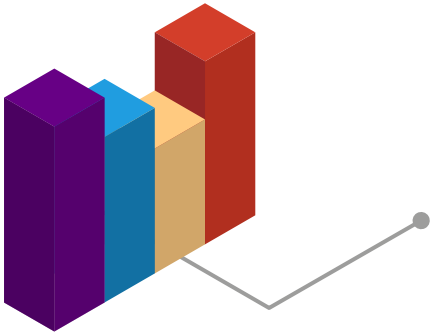
Introdução & Objetivo

Este estudo tem como objetivo **analisar as principais características dos repositórios mais populares no GitHub**, uma das maiores plataformas de hospedagem de código-fonte do mundo. Fatores como idade, contribuição externa, frequência de releases e atualizações, bem como a linguagem de programação utilizada, e a manutenção de issues, são investigados para identificar **padrões que possam explicar a popularidade** desses repositórios. A análise baseia-se em uma amostra de 1000 repositórios, cujos dados foram coletados através da API GraphQL do GitHub.

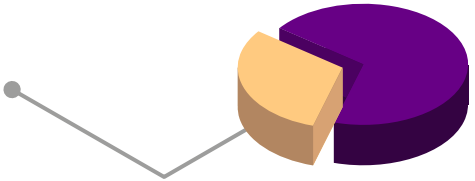


Hipóteses

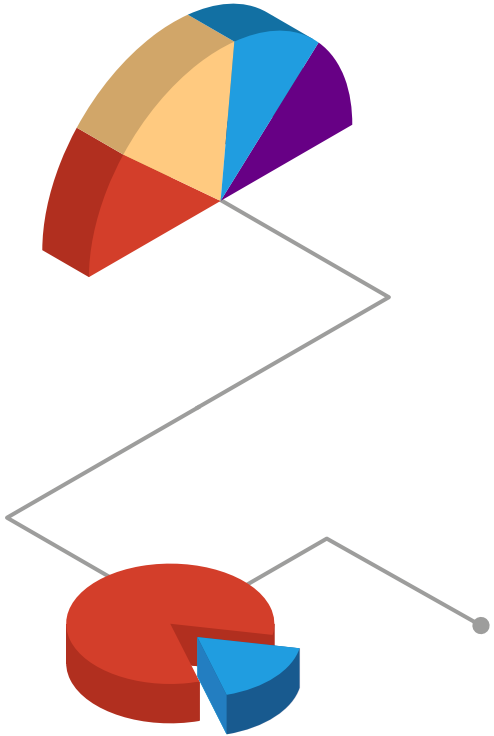
RQs	Hipóteses	Possíveis respostas
RQ 01: Maturidade dos Sistemas	Sistemas populares são maduros/antigos?	Sim, idade média superior a cinco anos
RQ 02: Contribuição Externa	Sistemas populares recebem muita contribuição externa?	Sim, recebem um número significativo de contribuições externas na forma de pull requests aceitas
RQ 03: Frequência de Releases	Sistemas populares lançam releases com frequência?	Eles podem ser caracterizados por uma média de pelo menos uma release a cada três meses
RQ 04: Frequência de Atualizações	Sistemas populares são atualizados com frequência?	Sim, sugerindo uma manutenção ativa (atualizações a cada mês ou em intervalos ainda menores)
RQ 05: Linguagem de Programação	Sistemas populares são escritos nas linguagens mais populares?	Sim, a maioria em JavaScript, Python, e Java
RQ 06: Manutenção de Issues	Sistemas populares possuem um alto percentual de issues fechadas?	Sim, mais de 75% das issues



02



Metodologia



Coleta e processamento de dados utilizando a API GraphQL do GitHub

2.1 Coleta de dados

2.1.1 Seleção dos Repositórios

Popularidade: repositórios com mais de uma estrela atribuída por usuários, aplicando filtros específicos na query.

2.1.2 Paginação

Em cada consulta, um número limitado de repositórios (20 por vez) foi coletado, utilizando o cursor fornecido pela API (continua a busca a partir do último repositório retornado).

2.1.3 Atributos Coletados

Idade do Repositório: Data de criação do repositório.
Contribuição Externa: Número total de pull requests aceitas.
Frequência de Releases: Número total de releases publicadas.
Frequência de Atualizações: Data da última atualização do repositório.
Linguagem de Programação: Linguagem principal utilizada no repositório.
Manutenção de Issues: Número total de issues abertas e fechadas.

2.2 Processamento e Análise dos Dados

2.2.1 Cálculo de Métricas

Para cada questão de pesquisa, calculamos a mediana das métricas coletadas.

2.2.2 Análise de Linguagens de Programação

Classificamos e contamos a frequência das linguagens de programação, identificando quais são as mais comuns entre os projetos analisados.

2.2.3 Razão de Issues Fechadas

Calculamos a razão entre o número de issues fechadas e o total de issues para avaliar a eficácia da manutenção nos repositórios.

2.3 Ferramentas Utilizadas

Python

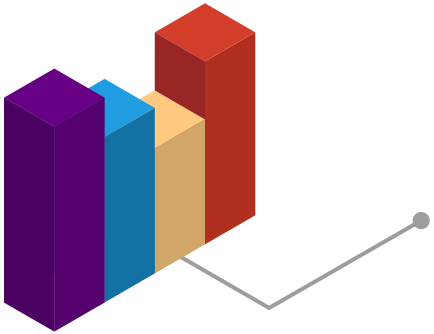
A coleta e análise dos dados. Bibliotecas como requests foram usadas para interagir com a API do GitHub, enquanto pandas e statistics foram empregadas para processamento e análise dos dados.

API GraphQL do GitHub

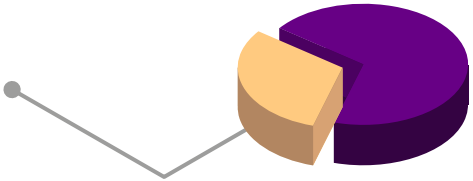
Flexibilidade e capacidade de retornar exatamente os dados necessários, facilitando a coleta de informações precisas e detalhadas dos repositórios.

2.4 Salvamento e Organização dos Dados

Todos os dados coletados foram armazenados em arquivos CSV para facilitar a manipulação e análise subsequente. Os resultados intermediários e as análises foram organizados em tabelas.



03



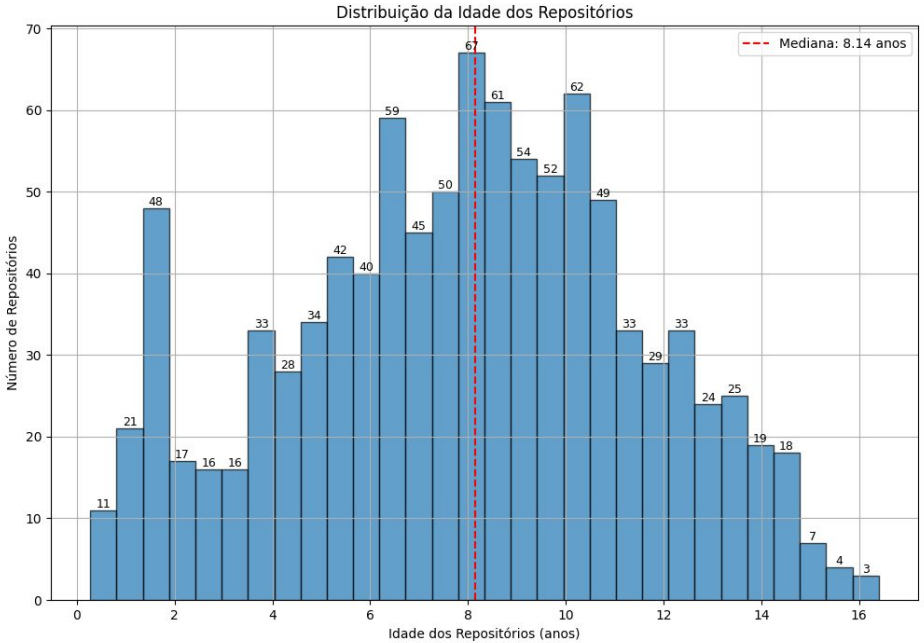
Resultados

Resultados

1 Sistemas populares são maduros/antigos?

A mediana da idade dos repositórios analisados foi de 8,1 anos. Este resultado sugere que os repositórios populares tendem a ser relativamente maduros, com uma história de desenvolvimento que se estende por vários anos.

Isso corrobora a hipótese de que projetos com maior tempo de existência têm mais oportunidades de se tornarem populares, devido ao tempo para amadurecer e ganhar tração na comunidade de desenvolvedores.

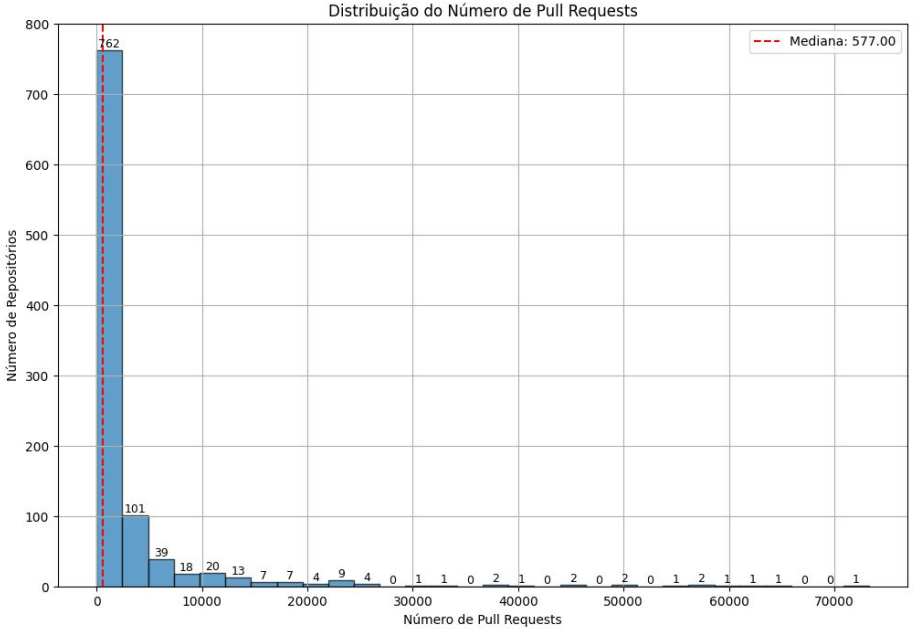


Resultados

2 Sistemas populares recebem muita contribuição externa?

A mediana do total de pull requests aceitas foi de 577. Esse número indica um alto nível de contribuição externa, sugerindo que os repositórios populares atraem muitas colaborações da comunidade.

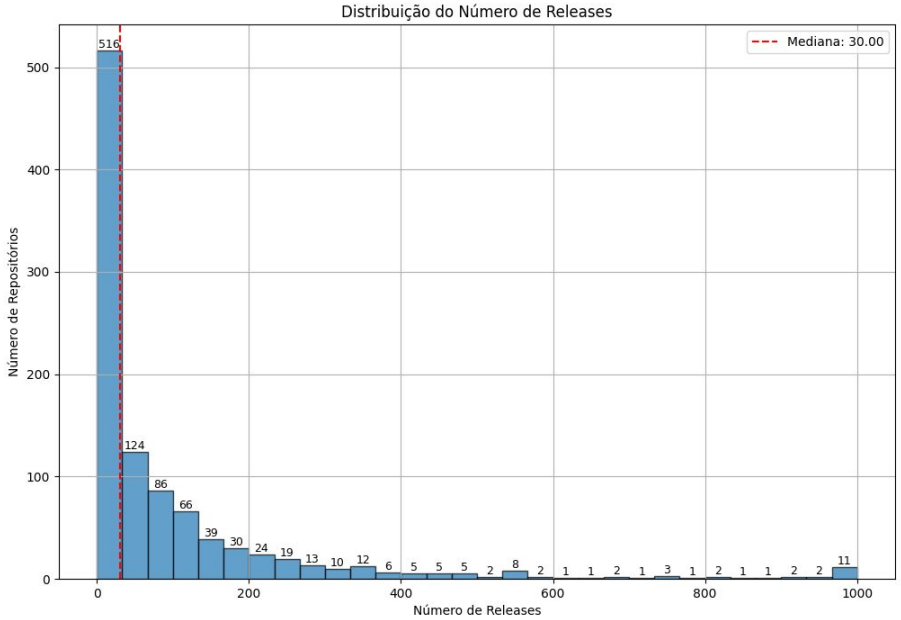
Esse dado reforça a ideia de que projetos populares tendem a ser sustentados por uma comunidade ativa e engajada.



Resultados

3 Sistemas populares lançam releases com frequência?

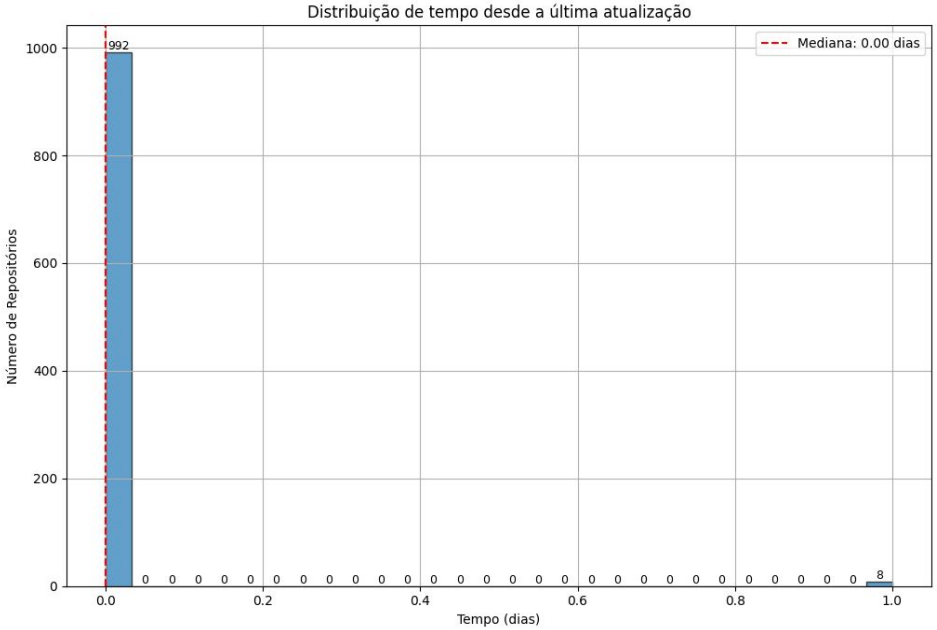
A mediana do total de releases foi de 30. Isso sugere que os repositórios populares mantêm um ciclo de lançamento de novas versões razoavelmente frequente, o que pode ser um indicador de um desenvolvimento contínuo e um compromisso com a melhoria e a manutenção do software.



Resultados

Sistemas populares são atualizados com frequência?

A mediana do tempo desde a última atualização foi de 0 dias, indicando que muitos dos repositórios populares são atualizados constantemente. Esse dado é consistente com a prática de desenvolvimento ágil e contínuo, onde as atualizações e melhorias são feitas regularmente, mantendo o software relevante e em constante evolução.

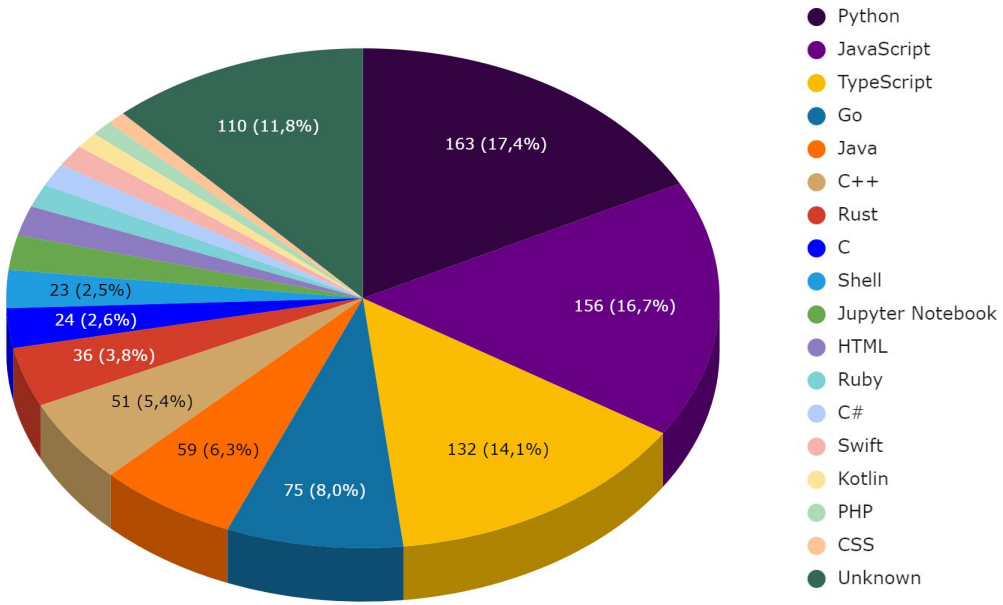


Resultados

5 Sistemas populares são escritos nas linguagens mais populares?

Linguagens amplamente adotadas e versáteis, como Python, JavaScript e TypeScript, dominam entre os projetos populares.

O número significativo de repositórios marcados como "Unknown" (110) pode refletir a presença de projetos que não especificam uma linguagem principal ou que utilizam uma variedade de linguagens.

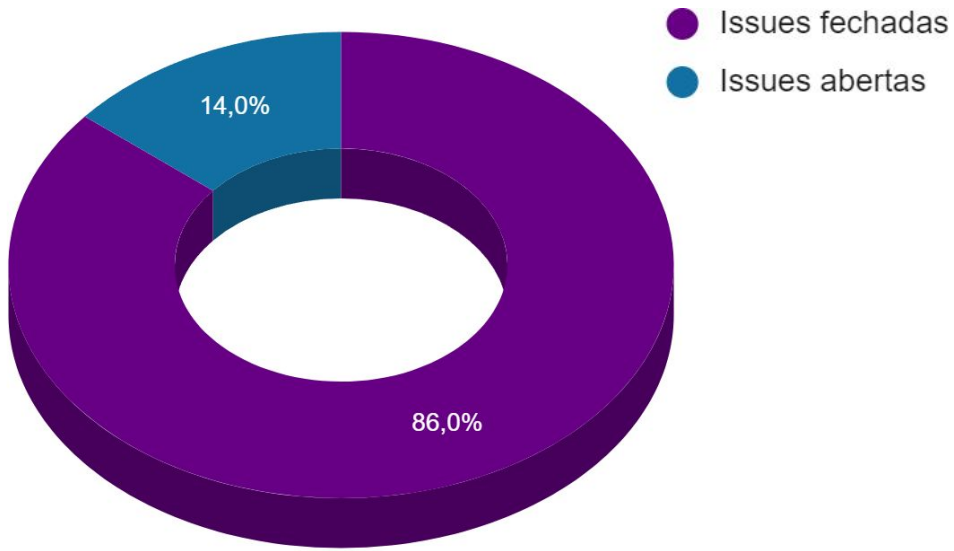


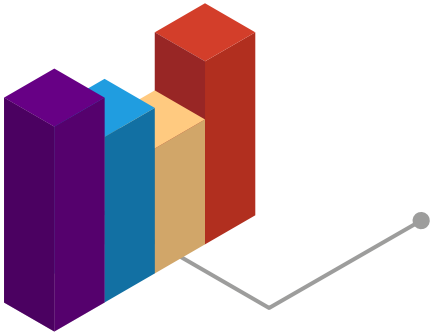
Resultados

6 Sistemas populares possuem um alto percentual de issues fechadas?

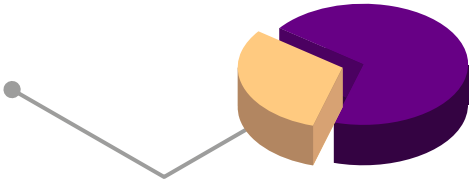
A mediana da razão de issues fechadas foi de 0,86. Isso indica que, em média, 86% das issues nos repositórios populares são fechadas, sugerindo uma boa gestão e manutenção dos problemas relatados.

Esse alto percentual de issues fechadas pode ser visto como um sinal de um projeto bem mantido e com uma equipe de desenvolvedores responsiva.





04



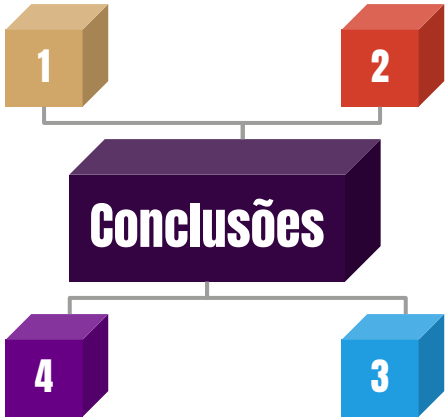
Considerações finais

Considerações finais

Maturidade, colaboração ativa, e manutenção rigorosa são fundamentais para o sucesso no GitHub.

Importância da Maturidade

Projetos mais antigos têm mais tempo para amadurecer e ganhar tração na comunidade



Colaboração e Ciclo de Desenvolvimento

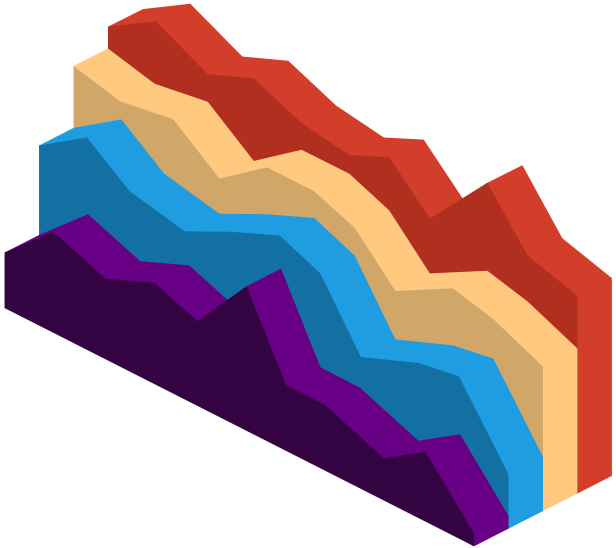
Ciclo de desenvolvimento contínuo mantém o software atualizado e funcional

Atualizações Constantes

Manter o projeto relevante é essencial para atrair e manter o interesse da comunidade

Diversidade nas Linguagens

A presença de repositórios com múltiplas linguagens ou sem especificação clara indica a diversidade tecnológica



Obrigado!

Ana Carolina Corrêa, Caio Elias,
Henrique Diniz e Maria Clara Santos