PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Enzo Barcelos Rios Ferreira
Igor Miranda Santos
João Paulo de Sales Pimenta

CARACTERÍSTICAS DE REPOSITÓRIOS POPULARES NO GITHUB:
Uma investigação sobre maturidade, colaboração, frequência de atualização e uso de linguagens em projetos open-source

Belo Horizonte
2025

SUMÁRIO

1 INTRODUÇÃO

O desenvolvimento de software *open-source* tem desempenhado um papel fundamental na evolução da tecnologia, permitindo a colaboração entre desenvolvedores e empresas ao redor do mundo. Neste relatório, analisamos os 1.000 repositórios mais populares do *GitHub*, com base no número de estrelas, para entender melhor suas características e padrões de desenvolvimento.

A pesquisa busca responder perguntas como: repositórios populares tendem a ser mais antigos e maduros? Eles recebem muitas contribuições externas? São frequentemente atualizados e lançam novas versões com regularidade? Além disso, investigamos se esses projetos utilizam as linguagens de programação mais populares e qual a taxa de fechamento de *issues*.

Para responder a essas questões, realizamos a coleta de dados via *GraphQL* e aplicamos métricas específicas para cada aspecto analisado. Os resultados permitem compreender melhor como esses projetos evoluem e quais fatores podem influenciar sua popularidade e manutenção ao longo do tempo.

1.1 Objetivos

1.1.1 Objetivo geral

Analisar as características dos repositórios *open-source* mais populares do *GitHub* para identificar padrões relacionados à sua maturidade, contribuição externa, frequência de atualização e uso de linguagens de programação.

1.1.2 Objetivos específicos

- Determinar a idade média dos repositórios populares e avaliar se projetos mais antigos tendem a ser mais bem avaliados.
- Analisar a quantidade de contribuições externas recebidas por esses repositórios, considerando o número de *pull requests* aceitas.
- Verificar a frequência com que esses repositórios lançam novas versões (*releases*).
- Avaliar a regularidade das atualizações, analisando o tempo decorrido desde a última modificação.

- Identificar as linguagens de programação mais utilizadas nos repositórios populares.
- Examinar a taxa de fechamento de *issues* para entender a manutenção e gerenciamento dos projetos.
- Comparar como essas características variam de acordo com a linguagem de programação utilizada nos repositórios.

2 METODOLOGIA

2.1 Coleta de Dados

A coleta de dados foi realizada por meio da **API do GitHub**, utilizando consultas *GraphQL* para obter informações detalhadas sobre repositórios populares. A query *GraphQL* foi construída para buscar os 10 repositórios mais populares por vez, com paginação para coletar dados de 1000 repositórios. Os campos incluídos na query foram:

- **name**: Nome do repositório.
- **createdAt**: Data de criação.
- **pullRequests**: Número de pull requests aceitas.
- **releases**: Número de releases.
- **updatedAt**: Data da última atualização.
- **primaryLanguage**: Linguagem primária.
- **issues**: Número de issues abertas e fechadas.

A paginação foi implementada utilizando o cursor (**after_cursor**) para buscar os próximos conjuntos de repositórios até atingir o total de 1000 repositórios. A automatização da coleta foi realizada por meio de scripts em **Python**, utilizando a biblioteca *requests* para fazer chamadas à API. Os dados brutos foram armazenados em um arquivo CSV (resultados.csv).

2.2 Processamento de Dados

Os dados coletados foram processados utilizando a biblioteca **Pandas** em Python. As etapas de processamento incluíram:

- **Conversão de Datas**: As colunas de datas (*createdAt* e *updatedAt*) foram convertidas para o formato *datetime* para facilitar cálculos.
- **Cálculo de Métricas**:
    - **Idade do Repositório**: Calculada como a diferença entre a data atual e a data de criação, em anos.
    - **Dias desde a Última Atualização**: Calculada como a diferença entre a data atual e a data da última atualização, em dias.
    - **Taxa de Fechamento de *Issues***: Calculada como a razão entre o número de *issues* fechadas e o total de *issues* (fechadas + abertas).

Os dados processados foram armazenados em um novo arquivo CSV (resultados_processados.csv), contendo as métricas calculadas.

2.3 Análise de Dados

Antes da análise, os dados passaram por uma etapa de pré-processamento para garantir a qualidade e a consistência. As seguintes ações foram realizadas:

- **Limpeza de dados:** Remoção de registros incompletos ou duplicados.
- **Transformação de dados:** Conversão de colunas para tipos de dados adequados (e.g., datas para o formato *datetime*).
- **Normalização:** Padronização de nomes de colunas e valores para facilitar a análise.

A análise descritiva foi realizada para resumir e descrever as características principais dos dados. Foram utilizadas as seguintes métricas e técnicas:

- **Medidas de tendência central:** Cálculo da mediana para variáveis numéricas, como idade do repositório, número de *pull requests* aceitas, total de *releases*, dias desde a última atualização e taxa de fechamento de *issues*.
- **Contagem de categorias:** Frequência de linguagens de programação primárias nos repositórios analisados.
- **Visualização de dados:** Geração de tabelas para facilitar a interpretação dos resultados.

As seguintes ferramentas e tecnologias foram utilizadas para a análise de dados:

- **Linguagem de programação:** Python.
- Bibliotecas: *Pandas* (para manipulação de dados), *NumPy* (para cálculos numéricos)

Para cada uma das questões de pesquisa (RQs), foram definidas métricas específicas:

1. **RQ 01 - Idade dos repositórios:** Calculada a partir da diferença entre a data atual e a data de criação do repositório.
2. **RQ 02 - Contribuição externa:** Quantificada pelo número total de *pull requests* aceitas.
3. **RQ 03 - Frequência de *releases*:** Calculada pelo número total de *releases* publicadas.
4. **RQ 04 - Atualização recente:** Determinada pelo número de dias desde a última atualização no repositório.
5. **RQ 05 - Linguagens de programação:** Contagem da linguagem primária de cada repositório.

6. **RQ 06 - Taxa de fechamento de *issues*:** Calculada pela razão entre o número de *issues* fechadas e o total de *issues*.

Os dados processados foram armazenados em um novo arquivo CSV (resultados_analisados.csv), contendo as medianas calculadas.

3 APRESENTAÇÃO E DISCUSSÃO DOS RESULTADOS

RQ 01. Sistemas populares são maduros/antigos?

Métrica: Idade do repositório (calculado a partir da data de sua criação)
Hipótese: Espera-se que sistemas populares sejam maduros, ou seja, tenham uma idade considerável, pois sistemas mais antigos têm mais tempo para ganhar popularidade e contribuições.

- **Mediana da idade dos repositórios:** 8.27 anos

RQ 02. Sistemas populares recebem muita contribuição externa?

Métrica: Total de *pull requests* aceitas
Hipótese: Espera-se que sistemas populares recebam muitas contribuições externas, refletindo uma comunidade ativa e engajada.
Resultado:

- **Mediana de *pull requests* aceitas:** 613.5

RQ 03. Sistemas populares lançam releases com frequência?

Métrica: Total de *releases*
Hipótese: Espera-se que sistemas populares lancem releases com frequência, indicando uma manutenção ativa e evolução contínua do projeto.
Resultado:

- **Mediana de *releases*:** 33.0

RQ 04. Sistemas populares são atualizados com frequência?

Métrica: Tempo até a última atualização (calculado a partir da data de última atualização)
Hipótese: Espera-se que sistemas populares sejam atualizados com frequência, refletindo uma manutenção contínua e resposta rápida a problemas e novas funcionalidades.
Resultado:

- **Mediana de dias desde a última atualização:** 0 dias

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

Métrica: Linguagem primária de cada um desses repositórios

Hipótese: Espera-se que sistemas populares sejam escritos em linguagens de programação populares, como *JavaScript, Python e Java.*
Resultado:

- **Contagem por linguagem primária:**
  - **Python:** 172 repositórios
  - **TypeScript:** 146 repositórios
  - **JavaScript:** 143 repositórios
  - **Go:** 73 repositórios
  - **Java:** 52 repositórios
  - **C++:** 51 repositórios
  - **Rust**: 41 repositórios
  - **C**: 26 repositórios
  - **Shell**: 23 repositórios
  - **Jupyter Notebook:** 21 repositórios
  - **N/A:** 104 repositórios
  - **Outras linguagens:** 148 repositórios

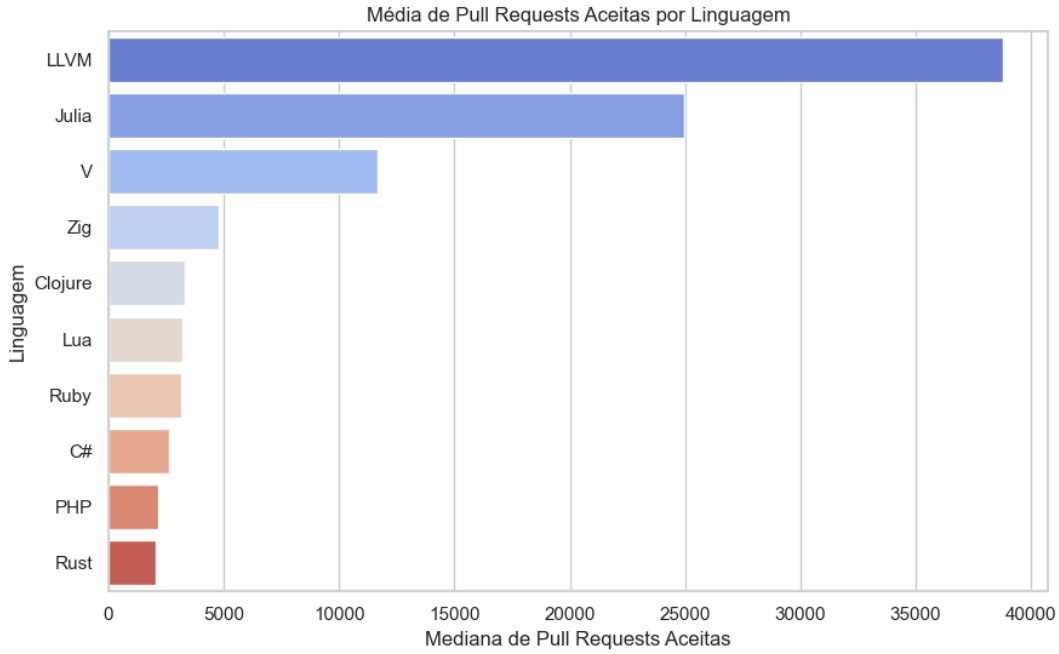
RQ 06. Sistemas populares possuem um alto percentual de issues fechadas?

Métrica: Razão entre número de *issues* fechadas pelo total de *issues*
Hipótese: Espera-se que sistemas populares tenham um alto percentual de *issues* fechadas, indicando uma boa gestão de problemas e uma comunidade ativa na resolução de *issues*.
Resultado:

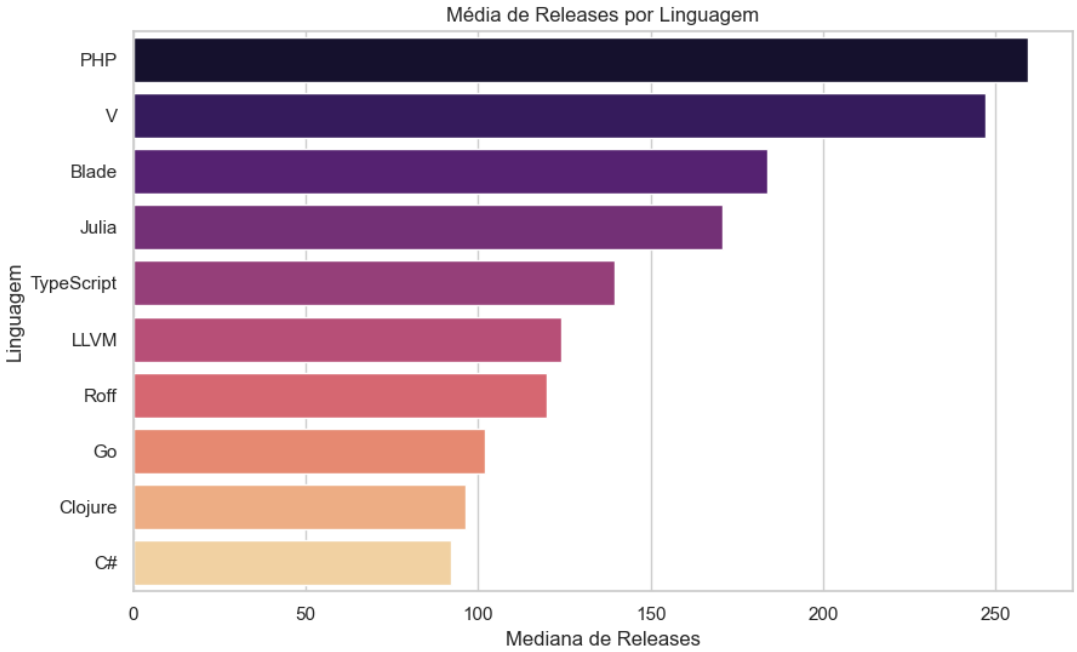
- **Mediana da taxa de fechamento de issues:** 0.99 (99%)

4 VISUALIZAÇÃO E ANÁLISE DOS DADOS

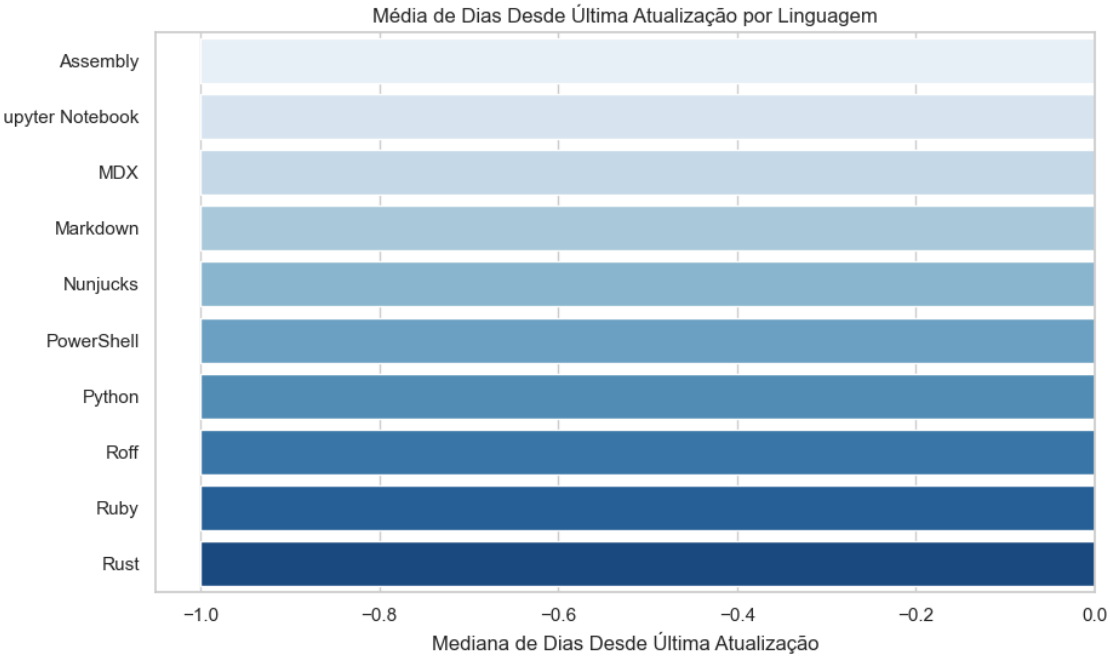
4.1 Contribuição externa por linguagem: Total de Pull Requests



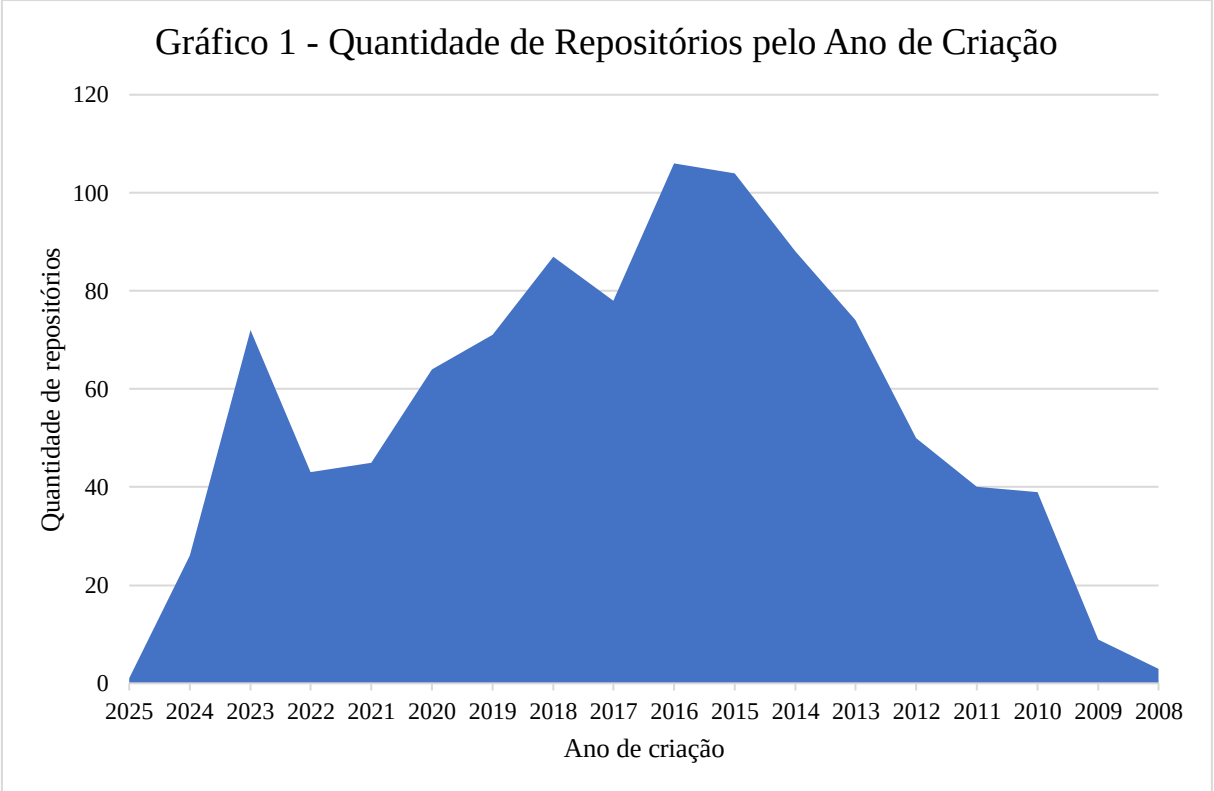
4.2 Lançamento de releases por linguagem



4.3 Frequência de atualização por linguagem



4.4 Quantidade de Repositórios pelo Ano de Criação

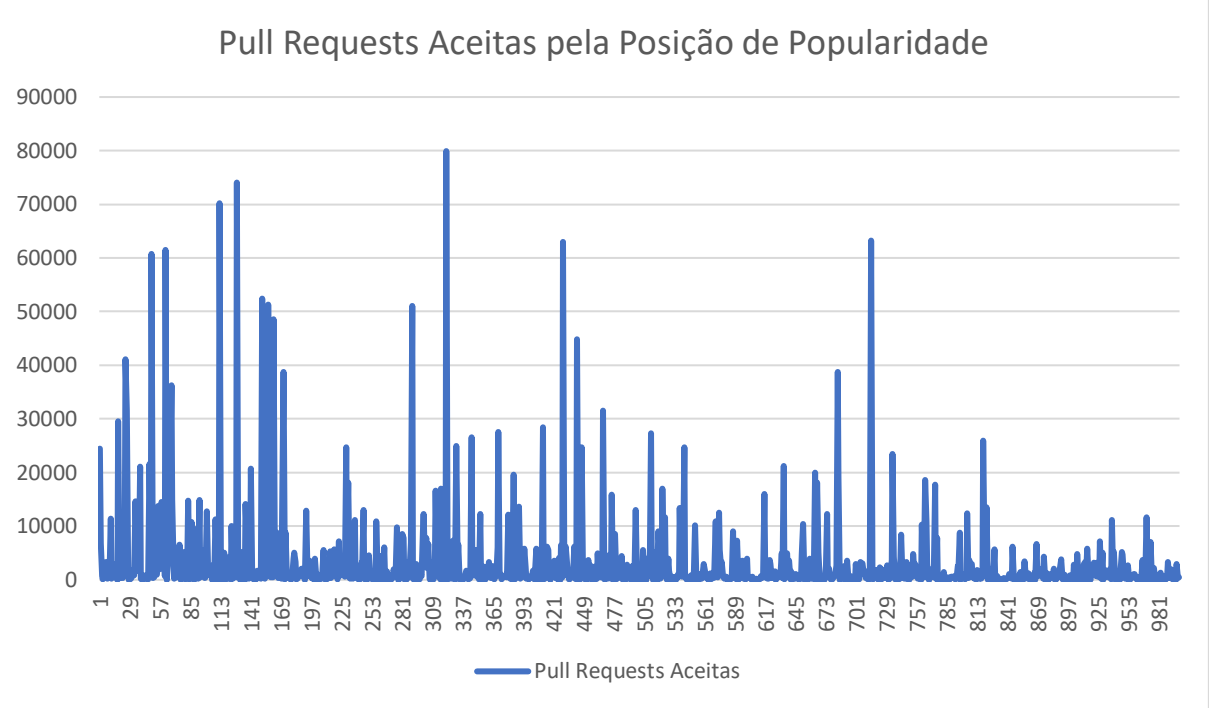


De acordo com o gráfico, a maioria dos repositórios populares foi criada entre 2014 e 2018, o que significa que eles têm entre 5 e 10 anos de idade.

Repositórios mais antigos (criados antes de 2010) são menos comuns, o que sugere que a popularidade não está diretamente ligada a sistemas extremamente antigos, mas sim a sistemas que tiveram tempo suficiente para amadurecer e ganhar notoriedade (cerca de 5 a 10 anos).

A queda no número de repositórios criados após 2019 pode indicar que leva tempo para que um repositório se torne popular, e sistemas mais recentes ainda estão em processo de ganhar popularidade.

4.5 Pull Requests Aceitas pela Popularidade



Há uma relação positiva entre a popularidade de um repositório e o número de pull requests aceitas. Repositórios extremamente populares, como *kubernetes*, *tensorflow*, e *flutter*, têm um número muito alto de PRs aceitas, refletindo uma comunidade grande e ativa.

Repositórios medianamente populares têm um número moderado de PRs, enquanto repositórios menos populares têm um número baixo de PRs.

A popularidade de um repositório parece estar diretamente relacionada ao engajamento da comunidade, medido pelo número de PRs aceitas.

Repositórios mais antigos e estabelecidos tendem a ter mais PRs aceitas, pois tiveram mais tempo para acumular contribuições e construir uma comunidade.

5 CONCLUSÃO

Análise dos repositórios populares no GitHub permitiu identificar características importantes sobre sua evolução e manutenção. Projetos bem-sucedidos tendem a ter:

Maturidade: A maioria dos repositórios populares tem entre 5 e 10 anos.

Comunidade ativa: Um alto número de *pull requests* aceitas indica engajamento.

Atualizações frequentes: Projetos populares são constantemente mantidos.

Uso de linguagens populares: *Python*, *TypeScript* e *JavaScript* dominam.

Boa gestão de *issues*: Alta taxa de fechamento de issues reflete eficiência na resolução de problemas.

Esses fatores são fundamentais para que um repositório se torne e permaneça popular no GitHub. A pesquisa reforça a importância da gestão ativa da comunidade, manutenção contínua e uso de tecnologias amplamente adotadas para garantir o sucesso de projetos *open-source*.

6 GLOSSÁRIO

API (Application Programming Interface)

Interface de programação que permite a comunicação entre diferentes sistemas.

CSV (Comma-Separated Values)

Formato de arquivo utilizado para armazenar dados tabulares separados por vírgulas.

GitHub

Plataforma de hospedagem de código-fonte e controle de versão baseado no Git.

GraphQL

Linguagem de consulta desenvolvida pelo Facebook para APIs mais eficientes e flexíveis.

Issues

Relatórios de problemas ou sugestões de melhoria dentro de um repositório.

Open-source

Modelo de desenvolvimento de software em que o código-fonte é aberto para visualização, modificação e distribuição.

Pull Request

Solicitação para incorporar alterações no código-fonte de um repositório.

Release

Versão oficial de um software disponibilizada ao público.

Repository (Repositório)

Local onde o código-fonte de um projeto é armazenado e gerenciado.

Star (Estrela)

Indicador de popularidade de um repositório no GitHub, baseado na quantidade de usuários que o marcaram como favorito.