ALGORITMO AVALIDOR DE CÓDIGO-FONTE RACKET

Paradigma de Programação Lógica e Funcional

Felipe Diniz Tomás

Ra: 110752

Definição do sistema

- O sistema é um **avaliador de código-fonte** em linguagem **racket**.
- Deve ser capaz de ler **múltiplos arquivos** de um diretório.
- Deve usar um **conjunto de métricas** de avaliação de código.
  - Definindo **pesos** para cada uma das métricas.
- Deve calcular uma **pontuação final** para cada arquivo.

Métricas

- As métricas foram definidas a partir de pesquisa, e que viabilizavam a implementação.

- Quantidade de linhas de código.
  - É interessante monitorar o **volume de código**, principalmente quando há comparação com outras implementações.
  - Foram consideradas apenas linhas de código, **desconsiderando linhas de comentários e linhas vazias**.

Métricas

- Quantidade de testes unitários.
  - Oferece uma análise sobre a **boa prática** e **coesão do código fonte**, já que indica que houveram testes de funções e portanto as mesmas estarão com seus resultados testados.

- Quantidade de definições (funções).
  - Contribui na análise da **complexidade do código**, já que um uso maior de funções denota-se uma maior fragmentação de funcionalidades aplicada ao código.

Métricas

- Quantidade de Requires (Imports).
  - Oferece uma análise do **uso de recurso externos e depências**.
  - Se um código possui muitos requires, denota-se que necessita de muitas bibliotecas externas ou arquivos externos (que podem ser arquivos do próprio autor ou de terceiros) para executar seu propósito.

- Quantidade de linhas maior que 80 caracteres.
  - Contribui na análise de **legibilidade do código** e é considerado uma **boa prática**.
  - A documentação do Racket destaca o uso de 80 caracteres como **vantajoso** em diversos aspectos.

Métricas

- Quantidade de comentários.
  - Além de ser uma **boa prática**, contribui na análise de **legibilidade do código e da lógica implementada**.
  - Muitos comentários indicam uma **preocupação do programador** em tornar seu **código mais legível e compreensível**, explicando o que determinada lógica empregue na implementação.

Pesos

- Foram definidos os pesos das métricas da seguinte forma:
  - Quantidade de **linhas de código**: 0,1 pontos por linha.
  - Quantidade de **testes unitários**: 0,4 pontos por teste.
  - Quantidade de **definições** (funções): 0,25 pontos por definição.
  - Quantidade de **requires**: 0,2 pontos por require.
  - Quantidade de **linhas maiores que 80 caracteres**: - 0,25 pontos por linha.
  - Quantidade de **comentários**: 0,3 pontos por linha.

Casos de teste

- Foi selecionado três códigos-fontes racket da internet, que implementam jogo ***flappy bird***.

- Exemplo – 1.rkt
  - https://github.com/Zhenya750/FlappyBird/
- Exemplo – 2.rkt
  - https://github.com/kkspeed/racket-flappy
- Exemplo – 3.rkt
  - https://github.com/Alexapostol2000/Flappy-Bird

Testes Unitários

- Foram implementados um total de 9 **testes** unitários.
  - sendo eles para as **funções que executam cada métrica**.
  - função da **pontuação final (pesos)**.
  - demais funções úteis.

Resultados

- Exemplo – 1.rkt

```
Nome do arquivo: "Exemplo - 1.rkt"

• Quantidade de linhas que contêm código:          177 linhas.
• Quantidade de testes unitários no código:        2 testes.
• Quantidade de definições no código:              24 definições.
• Quantidade de requires no código:                4 requires.
• Quantidade de linhas maiores que 80 caracteres: 0 linhas.
• Quantidade de comentários no código:             49 comentários.

Pontuação: 40 pontos
```

Resultados

- Exemplo – 2.rkt

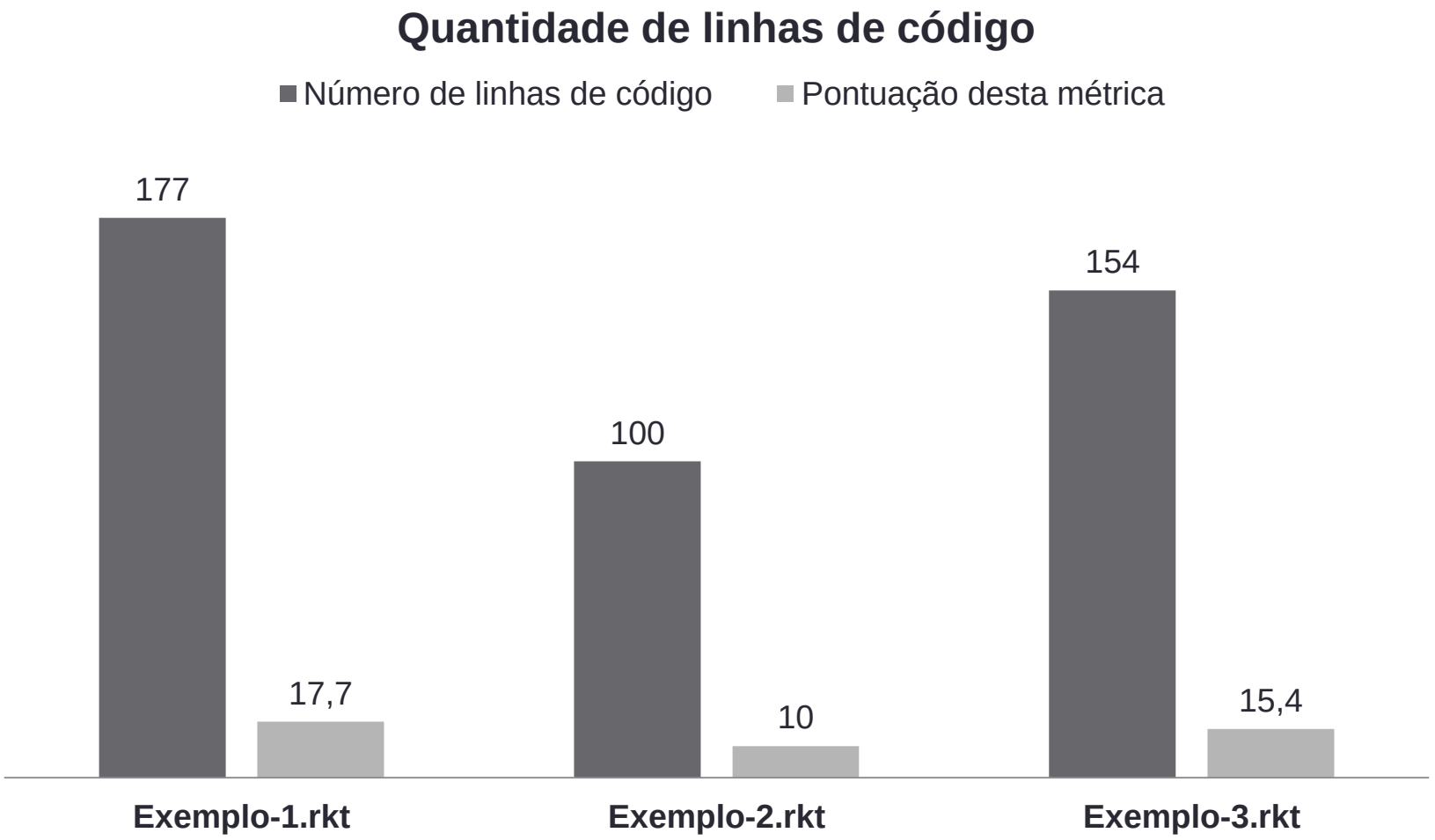
```
Nome do arquivo: "Exemplo - 2.rkt"

• Quantidade de linhas que contêm código:          100 linhas.
• Quantidade de testes unitários no código:        0 testes.
• Quantidade de definições no código:              32 definições.
• Quantidade de requires no código:                1 requires.
• Quantidade de linhas maiores que 80 caracteres: 2 linhas.
• Quantidade de comentários no código:             3 comentários.

Pontuação: 18.6 pontos
```

Resultados

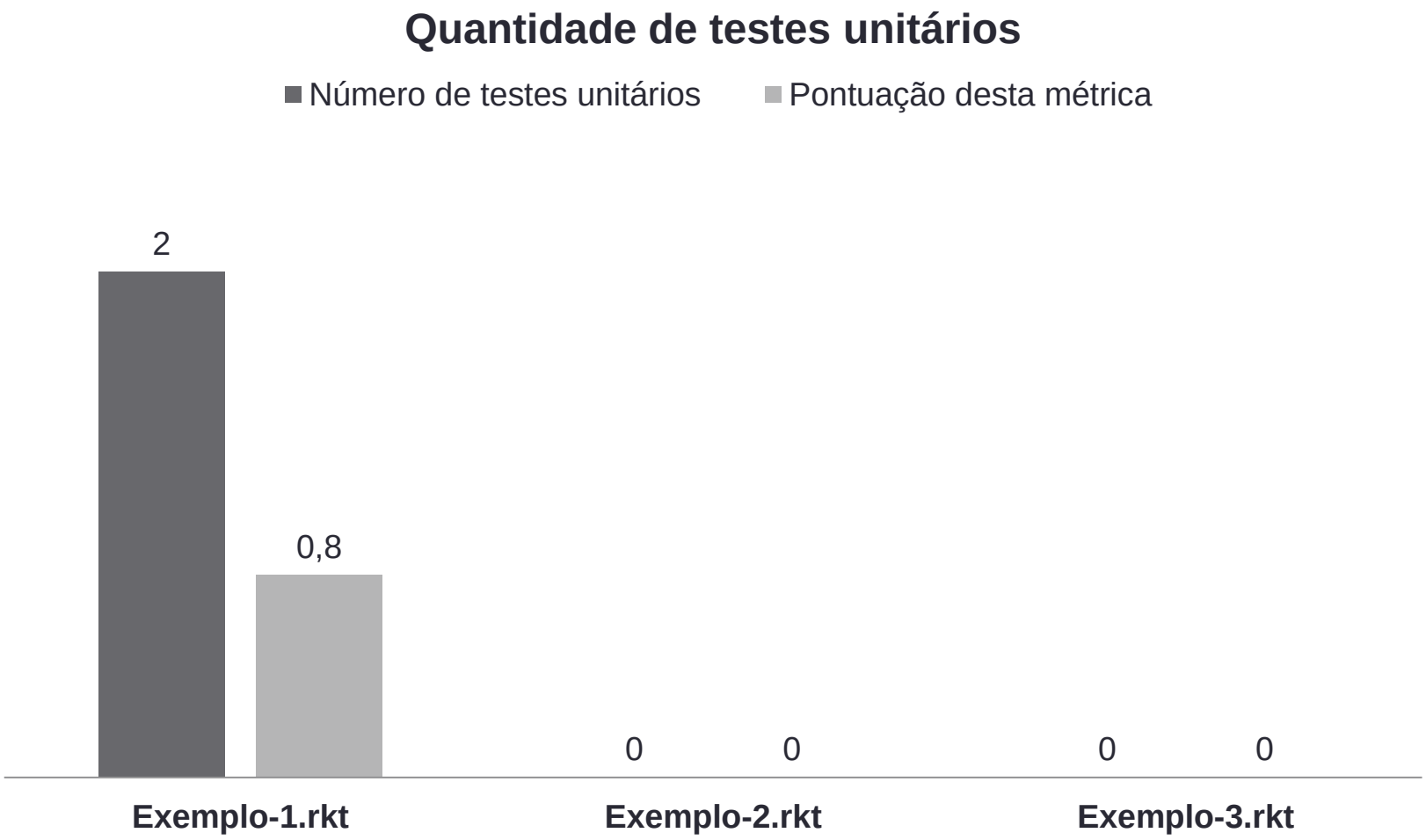
- Exemplo – 3.rkt

```
Nome do arquivo: "Exemplo - 3.rkt"

• Quantidade de linhas que contêm código:          154 linhas.
• Quantidade de testes unitários no código:        0 testes.
• Quantidade de definições no código:              51 definições.
• Quantidade de requires no código:                4 requires.
• Quantidade de linhas maiores que 80 caracteres: 41 linhas.
• Quantidade de comentários no código:             99 comentários.

Pontuação: 48.4 pontos
```

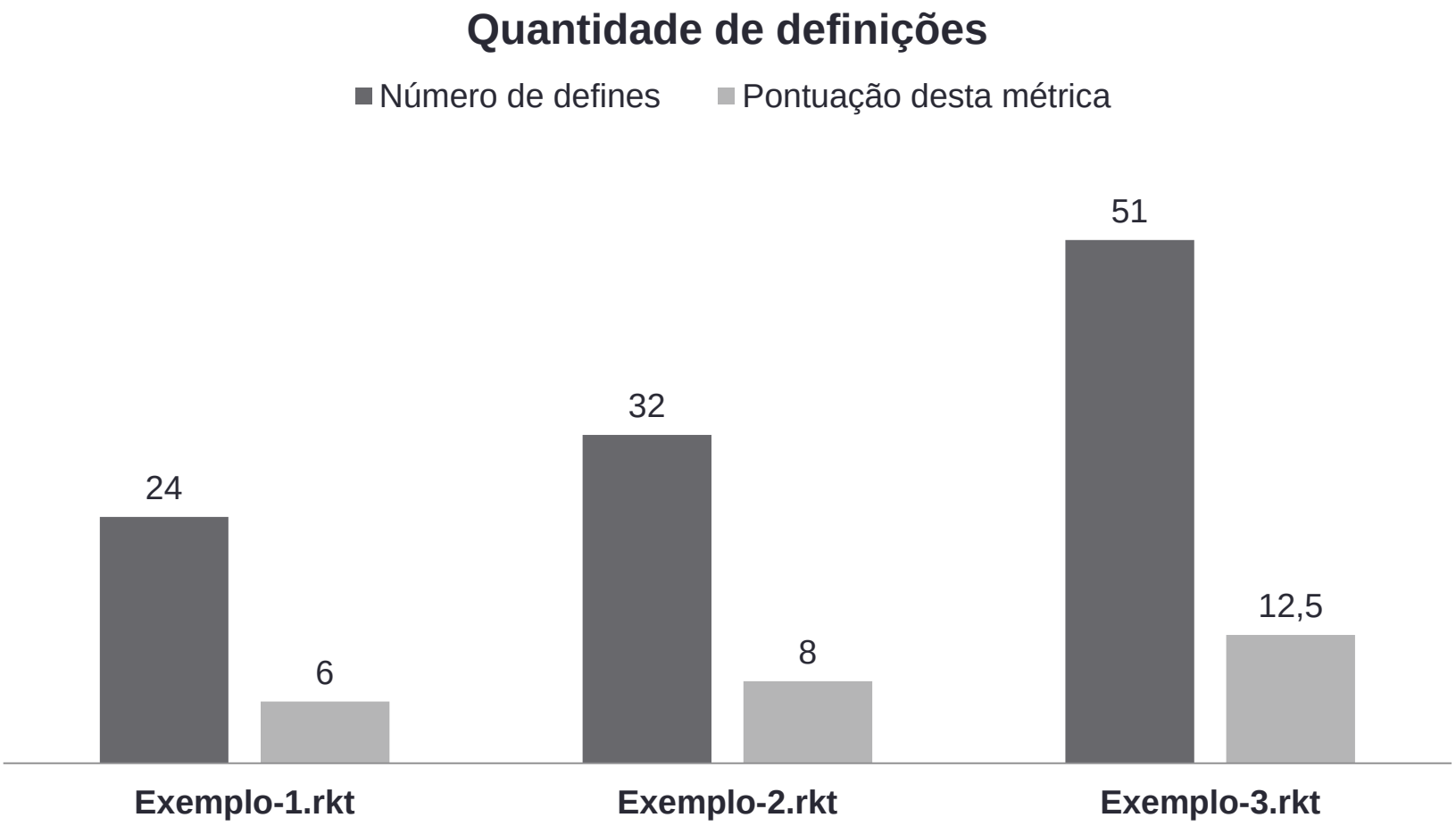
Análise dos resultados



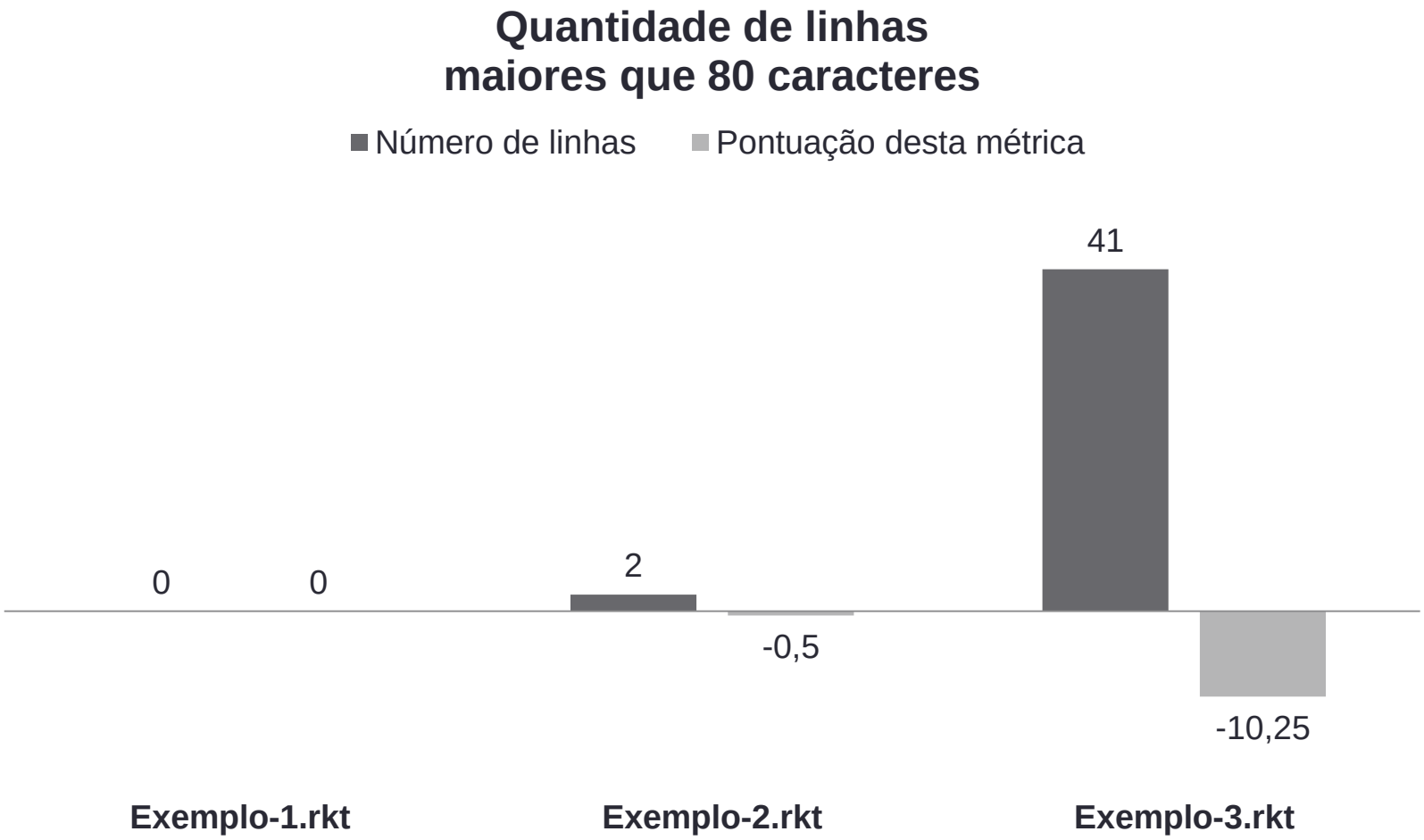
Análise dos resultados



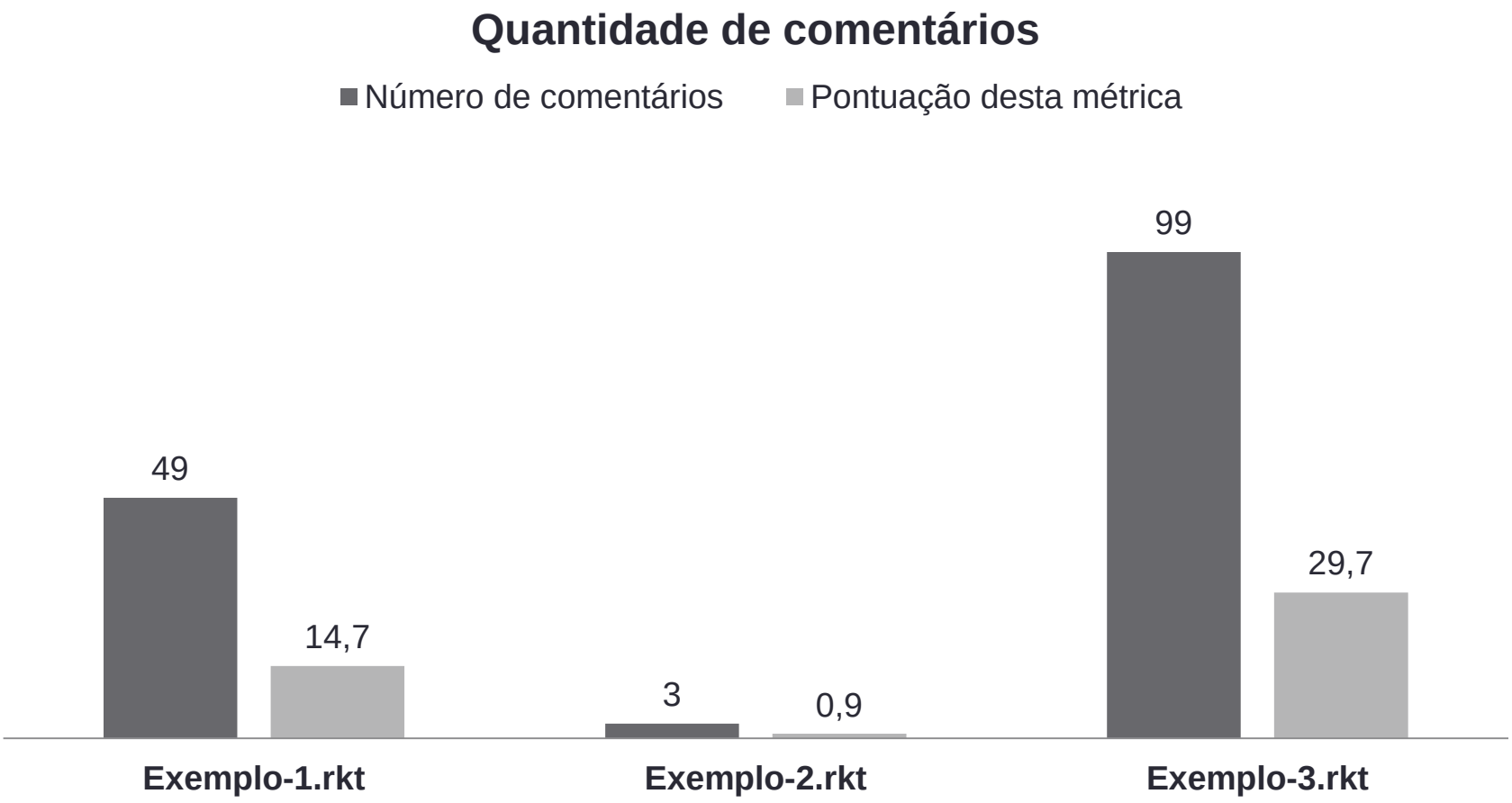
Análise dos resultados



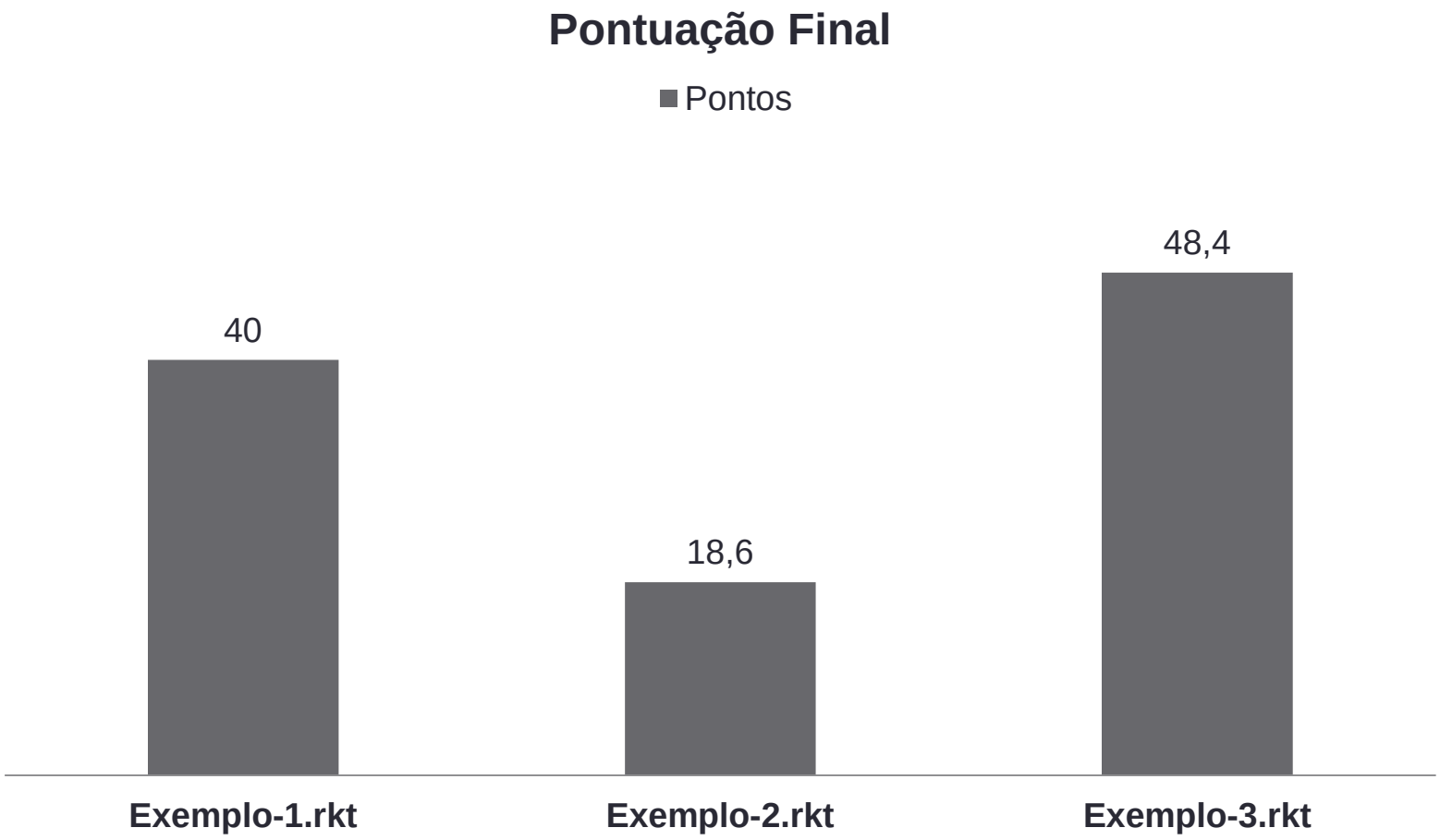
Análise dos resultados



Análise dos resultados



Pontuação final



Referências

- https://docs.racket-lang.org/

- https://github.com/Dinista/Basic-Racket-Code-Evaluator