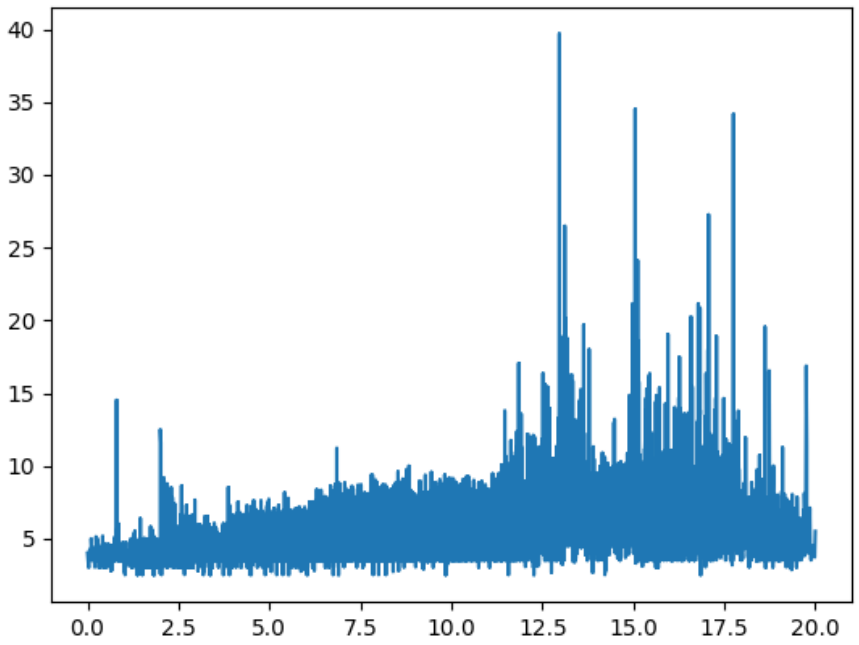
Medições do SLA

Nome do Serviço 1: Criar Jogador

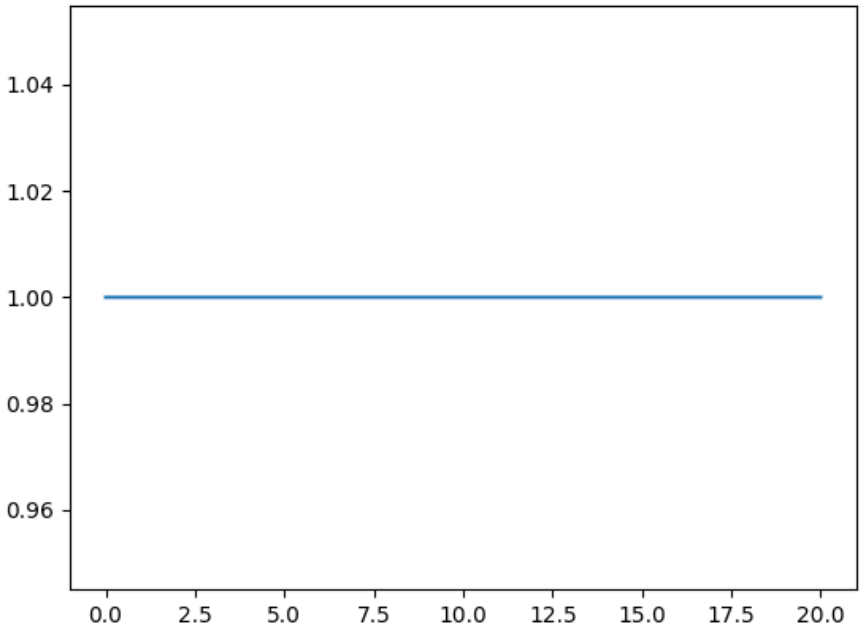
- Tipo de operação: inserção
- Arquivos envolvidos:
  https://github.com/celiofcj/fever-dunk/blob/mongodb/src/main/java/com/feverdunk/site/models/Jogador.java
  https://github.com/celiofcj/fever-dunk/blob/mongodb/src/main/java/com/feverdunk/site/controller/JogadorController.java
  https://github.com/celiofcj/fever-dunk/blob/mongodb/src/main/java/com/feverdunk/site/service/JogadorService.java
  https://github.com/celiofcj/fever-dunk/blob/mongodb/src/main/java/com/feverdunk/site/repository/JogadorRepository.java
- Arquivos com o código fonte de medição SLA:
  https://github.com/celiofcj/fever-dunk/blob/mongodb/sla/jogador_post.js
- Descrição das configurações:
  Ryzen 5 4500
  16GB DDR4 3200 Mhz

Medição 1

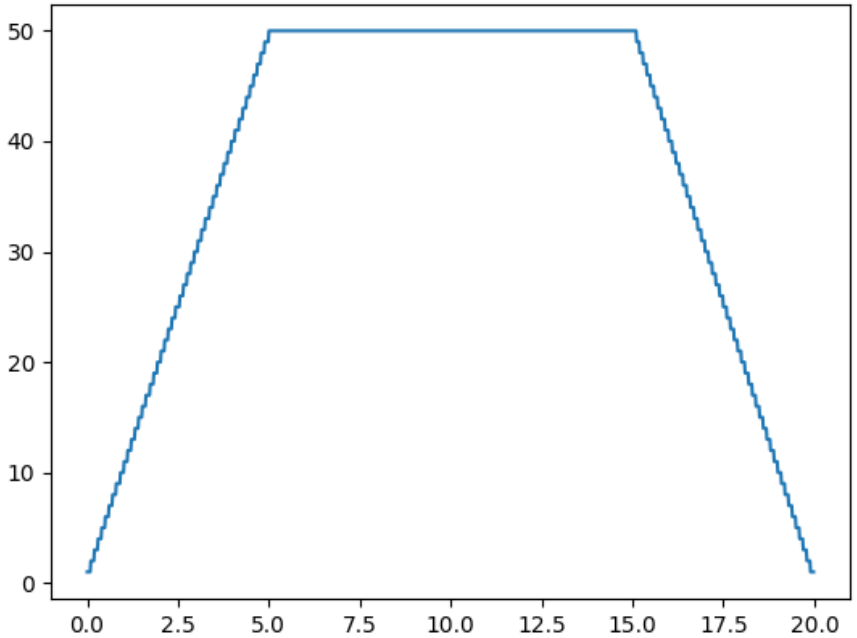
- Data da medição: 16/11/2023
- Testes de carga(SLA):



Latência



Vazão



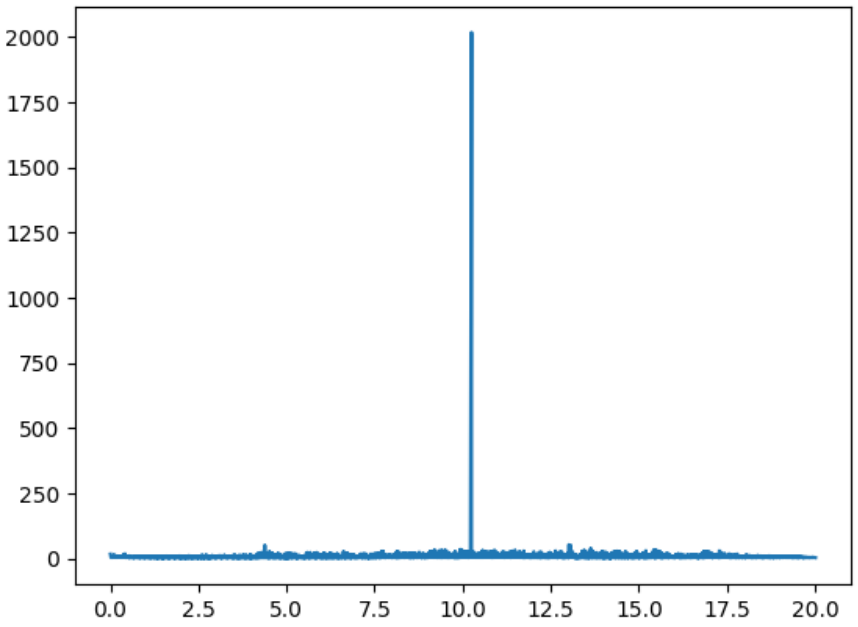
Concorrência

- Potenciais gargalos: O código presente nos arquivos especificamente na função “Post” que foi utilizado para a realização do serviço não possui complexidade muito alta nem

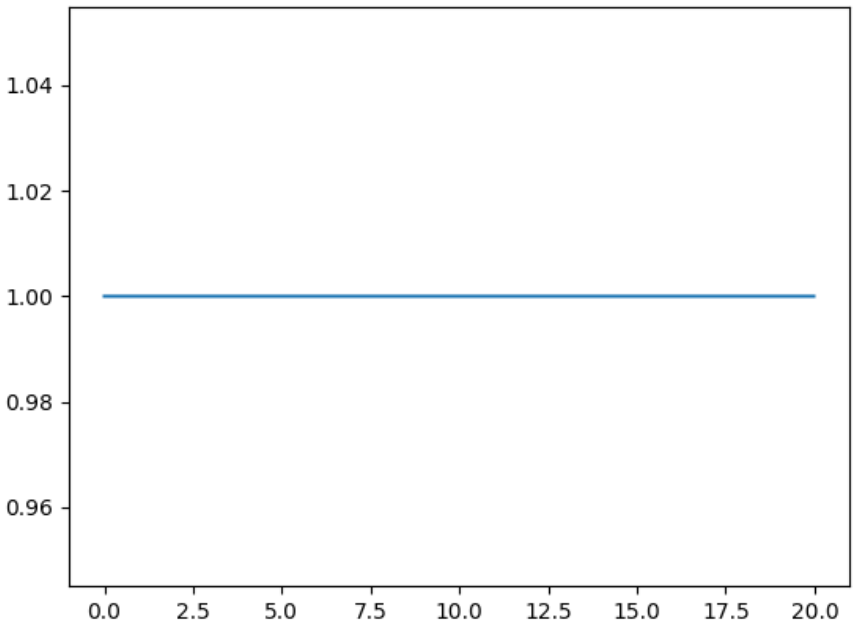
mesmo um vasto uso de memória. Desse modo, notamos que os potenciais gargalos provavelmente estariam ligados diretamente a inserção dos dados no banco ou na autenticação dos dados do jogador.

Medição 2

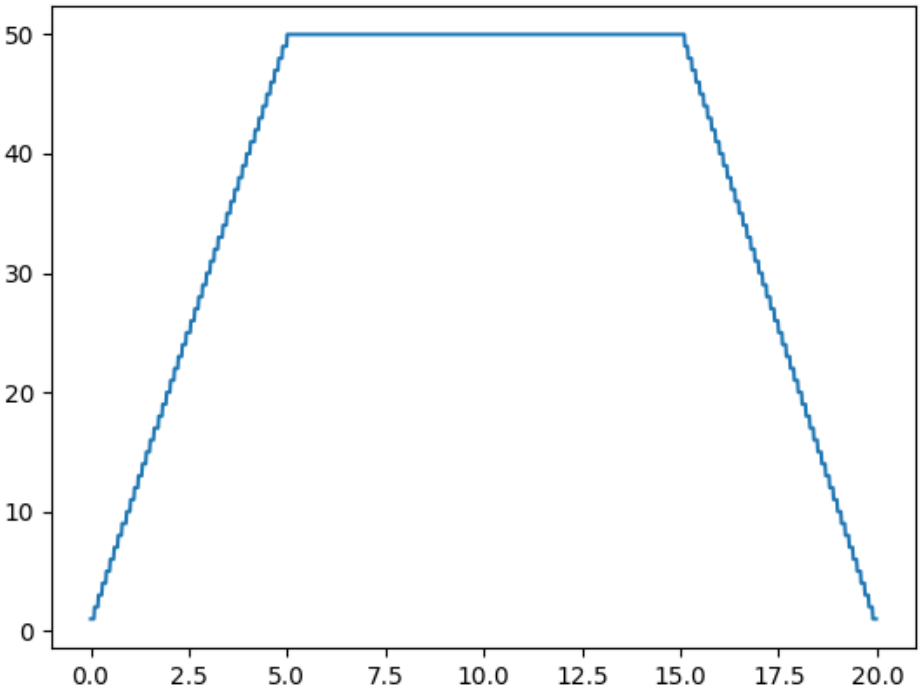
- Data da medição: 30/11/2023
- Testes de carga(SLA):



Latência



Vazão



Concorrência

- **Melhorias e Otimizações:** Realizamos testes de carga detalhados com o objetivo de detectar eventuais pontos críticos no sistema. Embora tenhamos

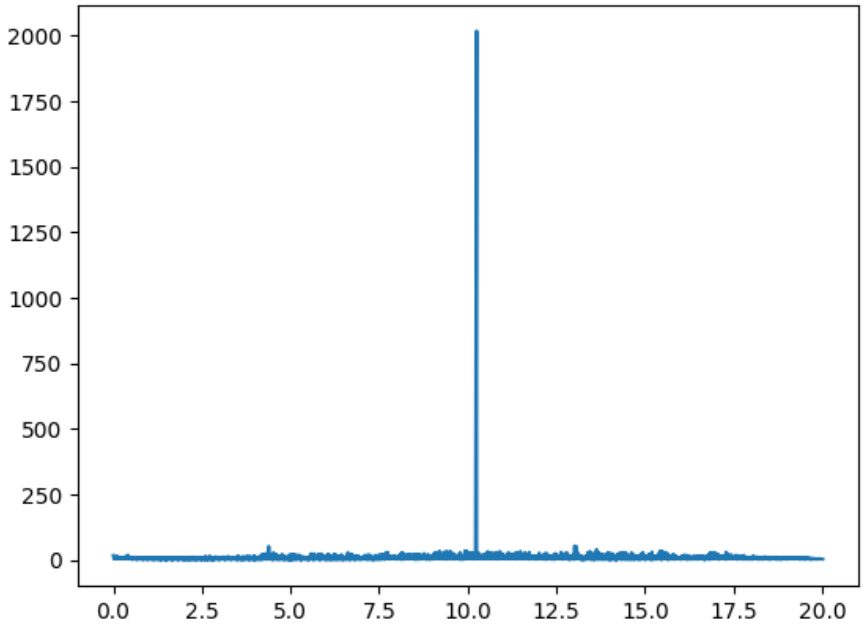
tentado aprimorar a aplicação feverDunk em resposta aos gargalos identificados anteriormente, nossos esforços não foram bem-sucedidos. Diante dessa constatação, optamos por preservar o desenvolvimento inicial, mantendo o código anterior devido à sua simplicidade, que se alia a um desempenho consistente. A simplicidade não apenas simplifica a manutenção, mas também atende às exigências dos usuários.

Nome do Serviço 2: Criar desempenho

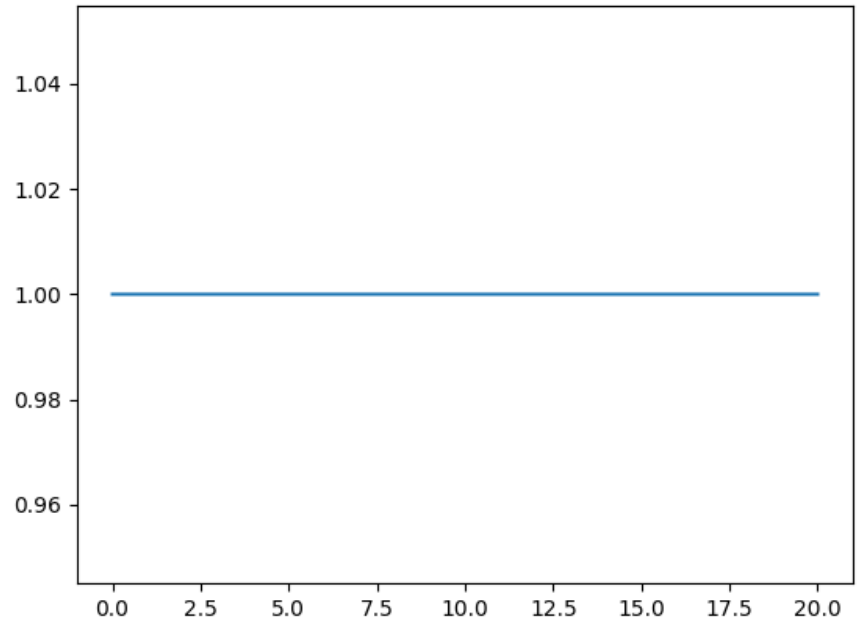
- Tipo de operação: inserção
- Arquivos envolvidos:
  https://github.com/celiofcj/fever-dunk/blob/mongodb/src/main/java/com/feverdunk/site/models/Desempenho.java
  https://github.com/celiofcj/fever-dunk/blob/mongodb/src/main/java/com/feverdunk/site/controller/DesempenhoController.java
  https://github.com/celiofcj/fever-dunk/blob/mongodb/src/main/java/com/feverdunk/site/service/DesempenhoService.java
  https://github.com/celiofcj/fever-dunk/blob/mongodb/src/main/java/com/feverdunk/site/repository/DesempenhoRepository.java
- Arquivos com o código fonte de medição SLA:
  https://github.com/celiofcj/fever-dunk/blob/mongodb/sla/desempenho_put.js
- Descrição das configurações:
  Ryzen 5 4500
  16GB DDR4 3200 Mhz

Medição 1

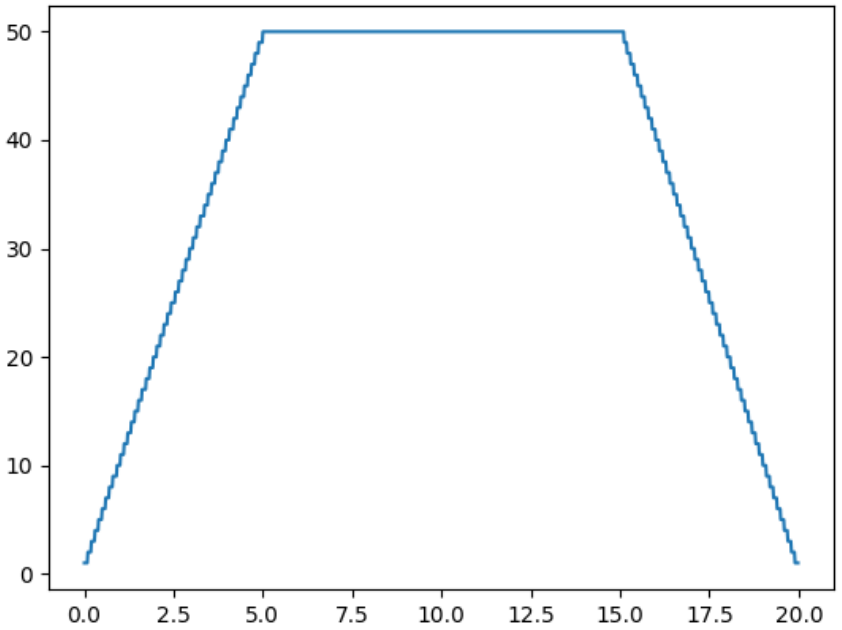
- Data da medição: 16/11/2023
- Testes de carga(SLA):



Latência



Vazão

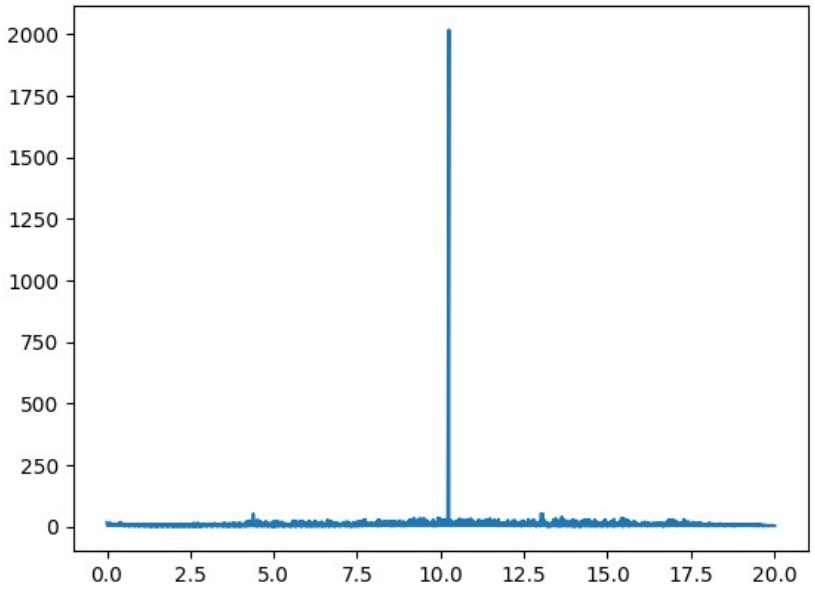


Concorrência

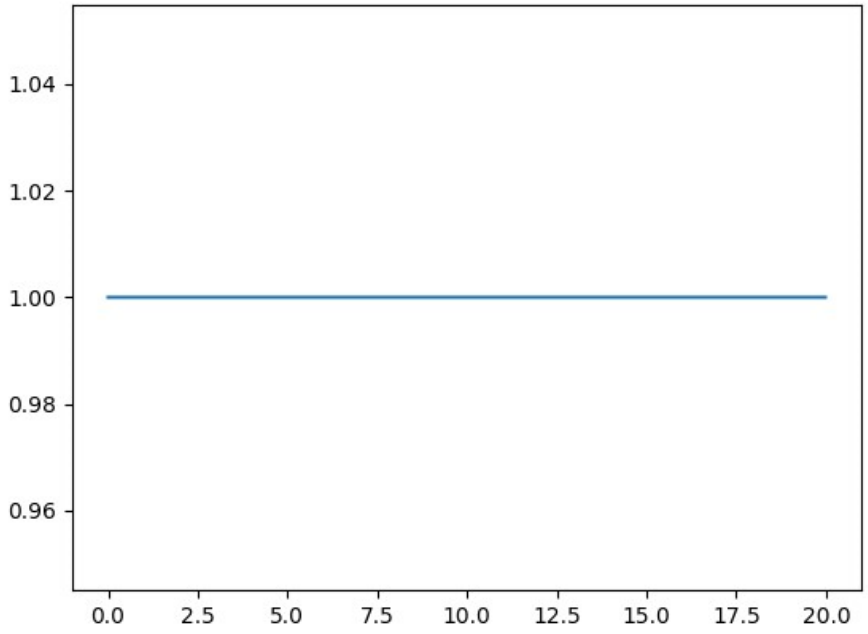
- Potenciais gargalos: O código presente nos arquivos especificamente na função “Put” que foi utilizado para a realização do serviço não possui complexidade muito alta nem mesmo um vasto uso de memória. Desse modo, notamos que os potenciais gargalos provavelmente estariam ligados diretamente a inserção dos dados no banco ou na autenticação dos dados do desempenho.

Medição 2

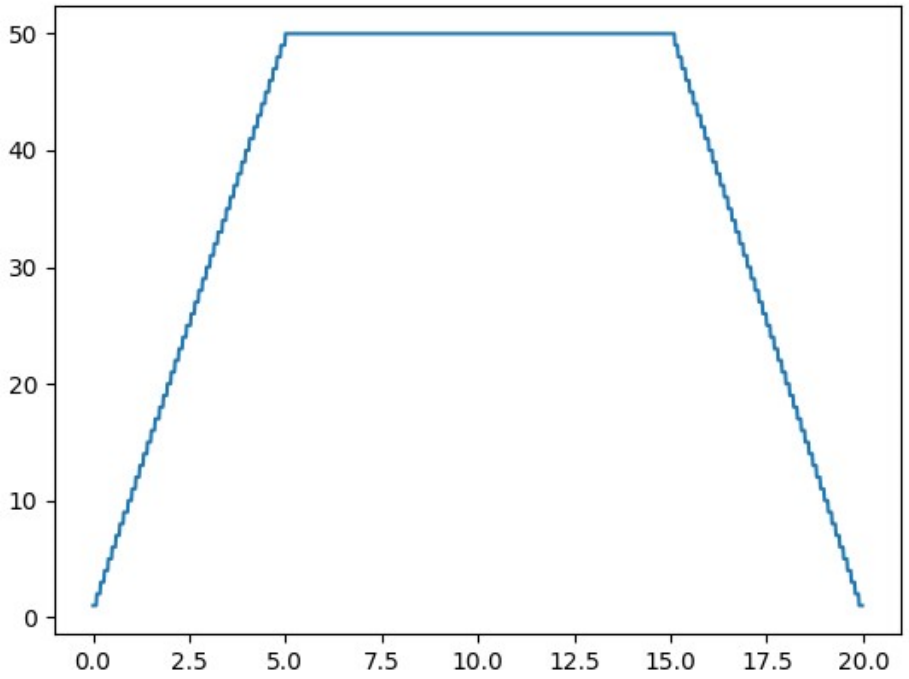
- Data da medição: 30/11/2023
- Testes de carga(SLA):



Latência



Vazão



Concorrência

- **Melhorias e Otimizações:** Realizamos testes de carga detalhados com o objetivo de detectar eventuais pontos críticos no sistema. Embora tenhamos tentado aprimorar a aplicação feverDunk em resposta aos gargalos identificados anteriormente, nossos esforços não foram bem-sucedidos. Diante dessa constatação, optamos por preservar o desenvolvimento inicial, mantendo o código anterior devido à sua simplicidade, que se alia a um desempenho consistente. A simplicidade não apenas simplifica a manutenção, mas também atende às exigências dos usuários.