Universidade Federal Rural de Pernambuco

Unidade Acadêmica de Belo Jardim - UABJ

Discente: Mª Clara Aquino, Nathan Cunha

Docente: Gabriel Silva Denini

Disciplina: Análise de Desempenho

Projeto Final

1. o projeto;

Desenvolvimento de uma plataforma completa para a comercialização de ativos digitais voltados para design gráfico, oferecendo modelos de venda por assinatura e compra individual, semelhante a plataformas como Freepik e Designi. O projeto abrange desde a concepção e design da interface do usuário até a implementação da infraestrutura backend, garantindo uma experiência fluida e segura para os usuários.

A plataforma conta com uma versão web, suportada por uma API robusta desenvolvida em Node.js, além de um aplicativo Android integrado a uma API em Flask. Também inclui a integração com sistemas de pagamento, armazenamento seguro de arquivos e recursos para gerenciamento eficiente de usuários e conteúdos.

2. O que o projeto vai avaliar:

- Quantas requisições cada API consegue fazer;
  - Vai ser utilizado o Apache Benchmark;
- Quanto tempo leva para carregar uma imagem do banco de dados na tela de home e na tela de produto;
  - vai ser utlizado as bibliotecas time de cada linguagem;
  - vai ser testado 10 vezes
- Como o serviço se comporta com muitos dados no banco;

3. Divisão de tarefas

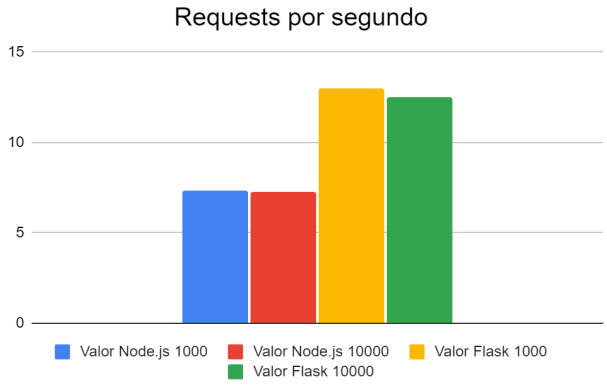
- Nathan Cunha: encarregado de fazer as avaliações na API desenvolvida em Node.js;
- Mª Clara Aquino: encarregada de fazer as avaliações na API desenvolvida em Flask;
- Atribuição dos dois: comparar os resultados obtidos e chegar a uma conclusão;

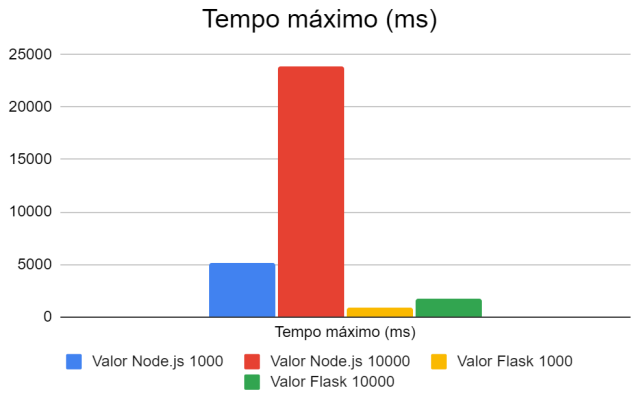
4. Resultados.

Primeiramente foi avaliado quanto tempo leva para abrir o mesmo produto. As requisições dentro das duas APIs foram as mesmas, para garantir maior confiabilidade nos resultados. Os testes foram realizados com diferentes cargas de requisições (1000 e 10000) para avaliar como cada framework se comporta sob pressão. Abaixo está uma análise detalhada dos dados:

Métrica	Valor Node.js 1000	Valor Node.js 10000	Valor Flask 1000	Valor Flask 1000
Requests por segundo	7.32 /sec	7.27 /sec	12.95 /sec	124.79 /sec
Tempo médio por request	1366.194 ms (1.36s)	13757.751 ms (1.37s)	772.222 ms	801.329 ms
Tempo máximo	5125 ms	23831 ms	890 ms	1737 ms
Percentil 50% (mediana)	1262 ms	12955 ms	762 ms	780 ms
Percentil 95%	1772 ms	16706 ms	775 ms	822 ms
Percentil 99%	4351 ms	20521 ms	793 ms	934 ms
Falhas	0	0	0	0

Nota-se que o Node.js mostrou um desempenho inferior em termos de requisições por segundo e tempos de resposta, especialmente sob alta carga. Isso pode ser um indicativo de que o Node.js, na configuração testada, não é o mais adequado para cenários de alta concorrência ou grande volume de requisições. já o Flask demonstrou um desempenho superior, com maior capacidade de processamento de requisições por segundo e tempos de resposta consistentes, mesmo sob carga pesada.

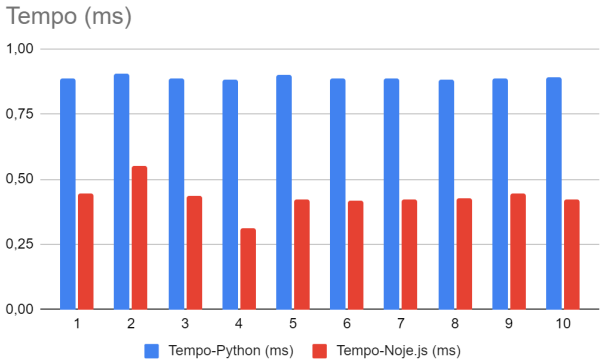




Depois foi feita a checagem de quanto tempo leva para carregar as imagens que aparecem na página de home. As requisições dentro das duas APIs foram as mesmas, para garantir maior confiabilidade nos resultados. Os testes foram realizados com diferentes cargas de requisições (1000 e 10000) para avaliar como cada framework se comporta sob pressão. Abaixo está uma análise detalhada dos dados:

Dados	Média	Mediana	Moda	Variância	Desvio-padrão
Tempo-Python	0,88945	0,88535	#N/A	0,0000599765	0,00816336668
Tempo-Node.js	0,43	0,42545	#N/A	0,002867464	0,05644529308

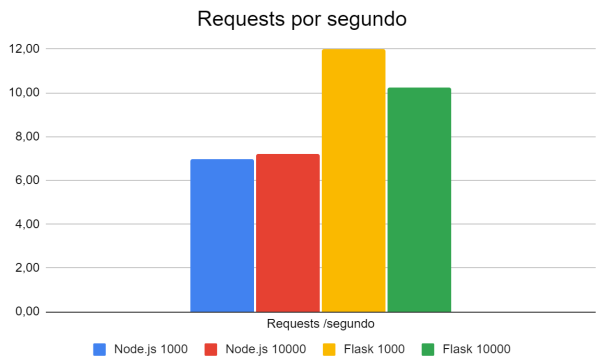
Observa-se que o Node.js apresentou um tempo médio de resposta significativamente menor, quase metade do tempo do Python. Isso indica que o Node.js é mais rápido na renderização da tela de home. A variância e o desvio-padrão do Python é muito menor que a do Node.js, indicando que os tempos de resposta do Python são mais consistentes. O Node.js, por outro lado, apresenta maior variabilidade nos tempos de resposta.
Apesar de ser mais rápido em termos de tempo médio e mediana, o Node.js apresenta maior variabilidade nos tempos de resposta, o que pode resultar em uma experiência menos consistente para o usuário. Embora mais lento, o Python oferece tempos de resposta mais consistentes, o que pode ser vantajoso para aplicações onde a previsibilidade é mais importante que a velocidade bruta.

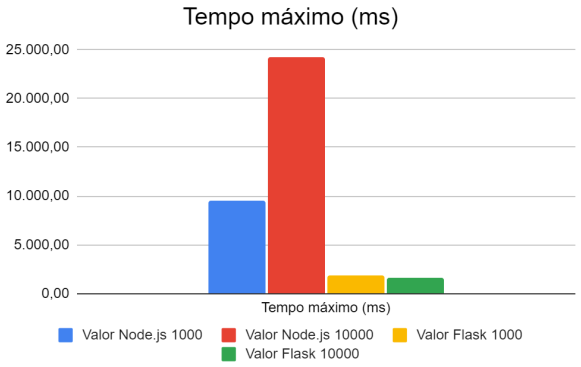


Posteriormente, foi testado com o banco de dados mais inflado de informações, para recorrer a realidade que a plataforma vai vivenciar. Os valores dos testes para mesma quantidade de requisições foram o seguinte:

Métrica	Node.js 1000	Node.js 10000	Valor Flask 1000	Valor Flask 10000
Requests /segundo	6,99	7,22	11.99	102.29
Tempo médio ms/request	1431,575	13846,305	834.051	977.610
Tempo máximo (ms)	9576	24227	1863	1606
Percentil 50% (mediana) (ms)	1303	13065	780	906
Percentil 95% (ms)	1561	18241	973	1280
Percentil 99% (ms)	6965	22992	1590	1404
Falhas	0	0	0	0

Nota-se que o Flask demonstrou um desempenho superior, com maior capacidade de processamento de requisições por segundo e tempos de resposta consistentes, mesmo sob carga pesada. Já o Node.js mostrou um desempenho inferior em termos de requisições por segundo e tempos de resposta, especialmente sob alta carga.





A análise de desempenho realizada nas APIs desenvolvidas em Node.js e Flask para a plataforma de comercialização de ativos digitais revelou diferenças significativas no comportamento de cada framework sob diferentes cargas de requisições. Os resultados indicam que o Flask apresenta um desempenho superior em termos de requisições por segundo (RPS) e tempos de resposta consistentes, mesmo sob carga pesada.

Por outro lado, o Node.js, embora tenha mostrado um tempo médio de resposta mais rápido em alguns cenários, apresentou maior variabilidade nos tempos de resposta e um desempenho inferior sob carga pesada.

Em relação ao carregamento de imagens na tela de home, o Node.js demonstrou ser mais rápido, mas com maior inconsistência nos tempos de resposta. Já o Flask, embora um pouco mais lento, ofereceu uma experiência mais previsível e estável, o que pode ser crucial para garantir uma boa experiência do usuário.