Aplicações e Serviços de Computação em Nuvem

Grupo 21
Francisca Lemos pg52693
Nelson Almeida pg52697
Nuno Costa pg52698
Gonçalo Freitas pg53840
José Martins pg53968

December 2023

1 Introdução

No âmbito da UC Aplicações e Serviços de Computação em Nuvem foi proposto realizar uma análise abrangente da aplicação *Laravel.io*. O objetivo central deste trabalho prático foi aplicar conceitos lecionados ao longo do semestre, percorrendo as etapas de caracterização, instalação, análise, monitorização e avaliação da aplicação. Este relatório está estruturado em quatro etapas distintas, cada uma referente a uma fase específica do processo.

Na primeira etapa, explorou-se a arquitetura e os componentes fundamentais utilizados. Na segunda, concentramo-nos nas ferramentas utilizadas para a instalação e configuração automática da aplicação.
A terceira etapa do relatório é dedicada à apresentação e justificação das diversas ferramentas utilizadas para efeturar a monotorização da aplicação, incluindo a observação de métricas relevantes e, posteriormente, a visualização dos resultados obtidos. Por últimos, na quarta etapa, abordamos as ferramentas selecionadas para avaliar e testar a aplicação em diferentes cenários.

Este relatório tem como objetivo mostrar detalhadamente cada etapa do projeto, destacando as escolhas das ferramentas e metodologias usadas. Para além de apresentar os resultados obtidos, também abordaremos as conclusões que surgiram durante o desenvolvimento do projeto.

2 Descrição da Arquitetura

O trabalho recai sobre uma arquitetura monolítica distribuída, onde as principais componentes da aplicação são distribuídas em diferentes *pods*. Essa abordagem proporciona uma notável facilidade na escalabilidade do sistema, proporcionando uma base sólida para futuras expansões.

- **Aplicação:** O *pod* de aplicação constitui o núcleo funcional do sistema, incorporando módulos cruciais como Monitorização, Benchmarking, Frontend e Backend. Esta estrutura modular permite uma gestão eficiente de cada componente, facilitando a manutenção e evolução do sistema de forma independente.
- **Base de dados:** A escolha para a base de dados recaiu na utilização de uma imagem **mySql**, alinhada com as recomendações da aplicação. Esta decisão foi tomada com base na robustez e confiabilidade do MySQL, proporcionando um ambiente estável para armazenamento e recuperação de dados. A separação da base de dados em um *pod* dedicado contribui para a modularidade da arquitetura, facilitando a gestão e otimização do sistema de armazenamento.

3 Identificação das ferramentas e abordagem utilizada para a instalação e configuração automática da aplicação

A abordagem utilizada para a instalação automática das componentes da aplicação foi através da ferramenta *Ansible*, que permite automatizar o *deployment* de uma forma eficiente e simples usando o menor número de passos manuais possíveis. Explicando a instalação da aplicação, começou-se por criar dois containers:

- *MySQL*: Para a correta instalação da aplicação *Laravel.io*, a criação de uma base de dados torna-se imperativa, sendo a escolha recaída sobre o sistema de gestão de base de dados *MySQL*. O processo iniciou-se com a criação de um *Persistent Volume Claim* (PVC) para assegurar a persistência de dados, seguido pela configuração do serviço *MySQL*.
- *Laravel.io*: Este container foi instanciado a partir de uma imagem previamente gerada e disponível no *Docker Hub*. Semelhantemente ao procedimento para o *MySQL*, foram elaborados os arquivos de *deployment* e de criação de serviço.

Desta forma, foi possível realizar a distribuição eficiente dos diversos componentes da aplicação entre os containers, estabelecendo uma infraestrutura coesa e funcional. Este método automatizado contribui significativamente para a agilidade e consistência do processo de implementação da aplicação *Laravel.io*..

4 Ferramentas de monitorização, métricas e visualização

A monitorização de uma aplicação é um processo essencial que pode ser dividido em quatro etapas distintas: observação, recolha, análise e apresentação. Para facilitar esse processo, utilizamos o *Elastic Stack*. Este conjunto inclui serviços específicos para cada fase da monitorização, como *MetricBeats* para a observação de métricas, *Logstash* para a recolha e processamento de logs, *Elasticsearch* para a análise e armazenamento dos dados, e finalmente, *Kibana* para a apresentação visual e análise interativa dos resultados.

5 Ferramentas de avaliação e testes desenvolvidos

Para avaliar o desempenho de nossa aplicação no cluster, utilizamos o *JMeter* e o *Gatling* para simular a interação de vários utilizadores simultaneamente.

Optamos principalmente pela ferramenta *Gatling* devido aos testes específicos já disponíveis para este tipo de aplicações, como o *Soak test*, que simula um uso normal da aplicação e analisa o comportamente da mesma ao longo do tempo; o *Stress test*, que simula a recuperação da aplicação após falhas; e o *Capacity test*, que ajuda a entender como a nossa aplicação escala e monitoriza a performance em momentos de pior desempenho. Em seguida, analisamos os dados por meio das funcionalidades oferecidas por essas ferramentas.

5.1 Apresentação e análise dos resultados da avaliação experimental

Pela análise das imagens seguintes conclui-se que o nodo que contém a aplicação sofre no desempenho com o aumento de utilizadores simultaneos. Quanto ao nodo da base de dados decidimos replicar a BD para obter uma maior resiliência, como indicado na Figura 2.

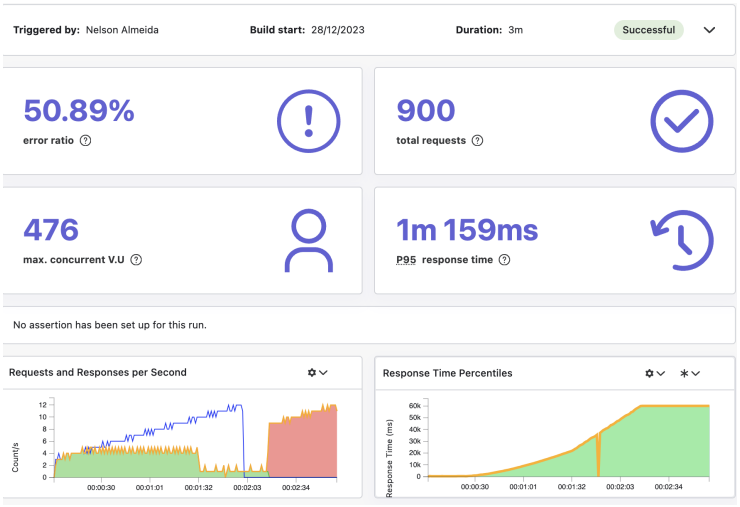

Figura 1: Início

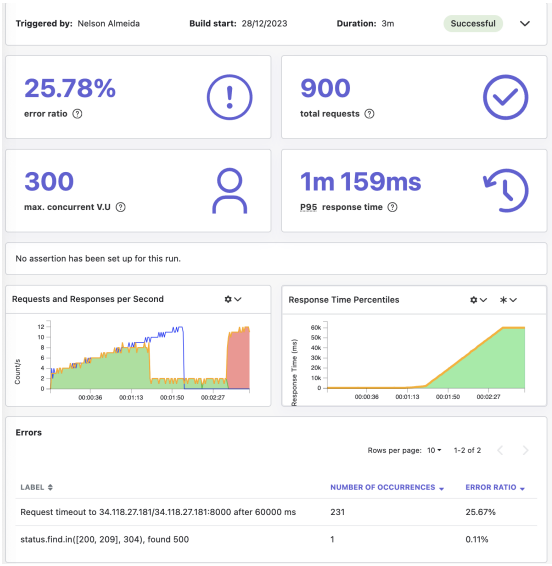

Figura 2: Replicar base de dados

Observamos um consumo elevado de memória RAM pela aplicação, como indicado na Figura 3. Em resposta a isso, realocamos recursos adicionais, resultando em melhorias significativas no desempenho (figura 4).

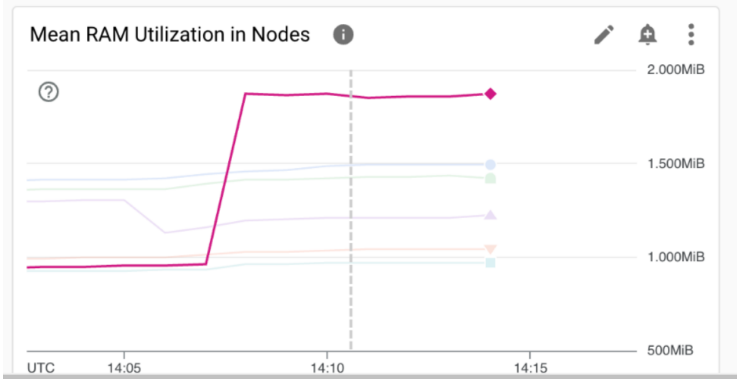

Figura 3: memória RAM consumida

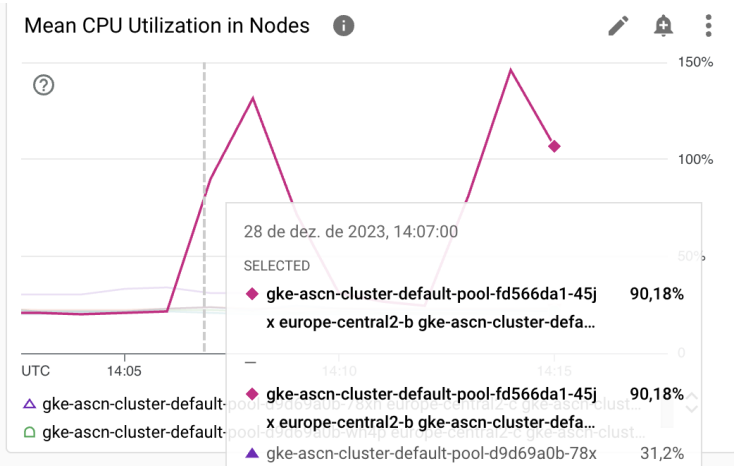

Figura 4: memória RAM consumida após realocação de recursos

Após isso, realizamos uns testes finais de carga através do *Gatling* para testar a escalabilidade final da aplicação (Figura 6,7,8).

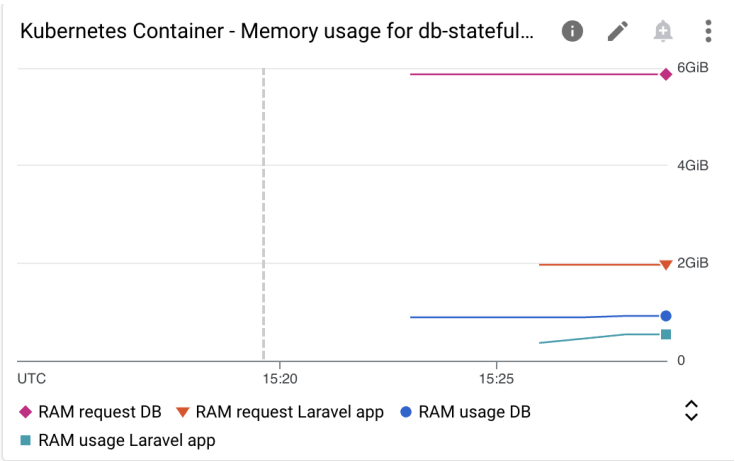

Figura 5: Resultado final da memória

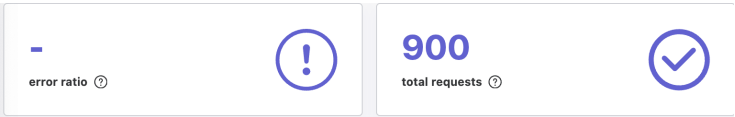

Figura 6: Total requests e error ratio final

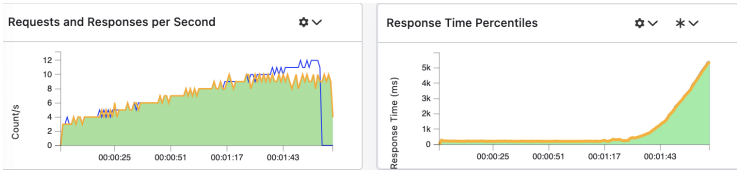

Figura 7: Soak test

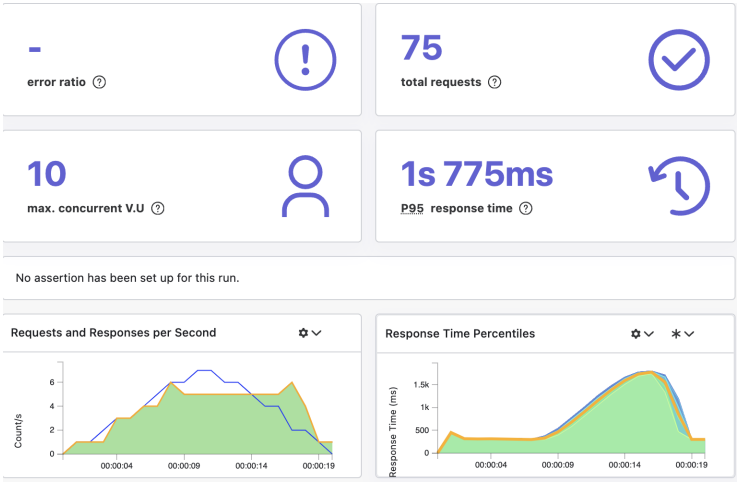

Figura 8: Stress test

6 Reflexão Final

6.1 Pontos fortes

- Apresentamos medidas de segurança tal como encriptação;
- Utilizamos ConfigMaps para uma definição mais clara e gerenciável das variáveis no nosso sistema;
- Implementação de variáveis de ambiente que permitem que escolhamos as configurações específicas que desejamos testar ou implementar.
- benchmark que nos da metricas da app tal como a response time e
- Bom balanceamento da carga entre diferentes pods.

6.2 Pontos a melhorar

- Monitorização sem o ElasticSearch
- Não realizamos testes tão aprofundados quanto gostariamos, o que nos dá uma menor perceção de possíveis falhas
- Não realizamos um cenário real elaborado para posteriormente observar o comportamento real da aplicação

7 Conclusão

Após a conclusão deste projeto, constatamos que alcançamos os objetivos propostos. Realizando testes de carga para várias situações diferentes e adaptando o sistema até este ficar o mínimo falível possível para o que se considerou serem cenários normais de uso. Este trabalho desempenhou um papel fundamental na consolidação dos conhecimentos adquiridos durante as aulas da respetiva UC. Mais ainda, acreditamos que representa uma mais valia para o mundo profissional que se segue, onde poderemos aplicar de maneira prática todas as aprendizagens adquiridas aqui.