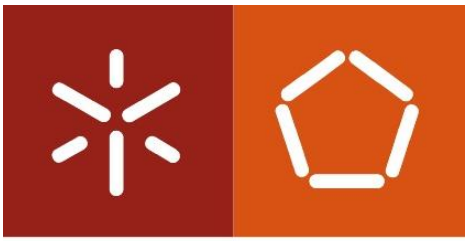


UNIVERSIDADE DO MINHO

DEPARTAMENTO DE INFORMÁTICA

Aplicações e Serviços de Computação em Nuvem

Duarte Oliveira `pg47157` Tiago Barata `pg47695`

Melânia Pereira `pg47520` António Guerra `pg47032`

11 de janeiro de 2022

Índice

1 Introdução

Este relatório procura analisar e compreender o software *open-source* **Wiki.js**, uma plataforma que permite desenvolver documentação de uma forma visualmente apelativa e intuitiva.

Serão abordados todos os tópicos escrutinados no enunciado deste projeto prático, tentando sempre primar pela objetividade e clareza, de maneira a transparecer mais facilmente todas as decisões que tomamos na elaboração deste trabalho. Será também feito o uso de diferentes imagens e esquemas que possam, de forma pertinente, justificar qualquer tipo de afirmação.

Finalmente, haverá uma abordagens as dificuldades associadas ao processo de desenvolvimento, procurando justifica-las e abordar de que forma poderíamos tê-las colmatado.

2 Wiki.js

O *wiki.js* é uma plataforma *open-source* que funciona virtualmente em qualquer plataforma, caracterizada também por deter diferentes atributos que a tornam um *software* de bastante interesse.

Em primeiro lugar, e graças à velocidade do *engine Node.js*, o *wiki.js* foi desenvolvido tendo em conta a *performance*.

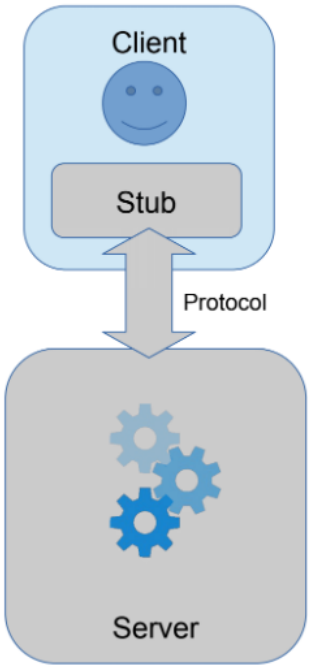
Para além disso, mas não menos relevante, é também um sistema escalável, utilizando de forma inteligente e conveniente todos os recursos disponíveis.

Finalmente, é um serviço que tem a si associado bastante documentação, toda ela bastante *user-friendly*, não só auxiliando o processo de instalação como na perceção de como utilizar diferentes ferramentas que nos são bastante úteis para a integração da plataforma da forma que mais nos conveio.

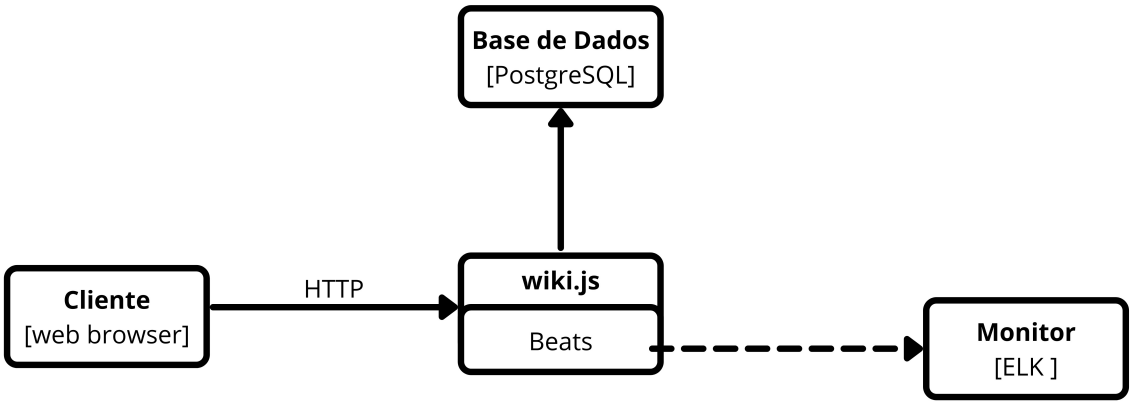
3 Arquitetura da Aplicação

Segundo a nossa interpretação e tendo em conta o conhecimento adquirido na vertente teórica desta disciplina - mais especificamente quando abordamos aplicações distribuídas - conseguimos enquadrar a *Wiki.js* como uma arquitetura relativamente semelhante à de **Cliente-Servidor**.

Sendo escrutinados e intelectualmente honestos, este foi, desde logo, um tema de algum debate entre o grupo durante algum tempo, visto que tivemos algumas dúvidas em definir a aplicação numa arquitetura distribuída.



4 Componentes



4.1 Aplicação

A aplicação contém diferentes características e módulos, que são integrados na aplicação *wiki.js* de forma a que esta tenha diferentes características de relevo importante para a interação com o utilizador. Entre elas destacam-se:

- Autenticação
- Comentários
- Search Engine
- Rendering
- Storage

A aplicação é aquela que oferece as funcionalidades com que o *User* interage a partir do *browser*. Estes componentes estão disponíveis e estão à mercê da utilização do utilizador.

4.2 Base de Dados

A base de dados é a componente que proporciona persistência ao serviço. Foi utilizado o *PostgreSQL* no desenvolvimento deste trabalho, por ser a mais recomendada. Foi usado o serviço *Cloud SQL* da *Google Cloud Platform* para alojar a base de dados.

A *Cloud SQL* disponibiliza a opção de criar instâncias que contém bases de dados às quais é possível conectar.

4.3 Monitor

Para monitorização foram usadas as *frameworks ElasticSearch* e *Kibana*, que serão discutidas e abordadas mais à frente.

4.4 Cliente

O cliente interage com a app por meio de um webBrowser, que conforme recomendado nos requisitos do *Wiki.js* foi o *Safari*.

5 Ferramentas e abordagem

De maneira a poder automatizar ao máximo a instalação da aplicação utilizou-se a ferramenta *open-source* **Ansible** cujas valências permitem que neste projeto se pudesse gerir e inferir automação a máquinas virtuais.

O *Ansible* permite que todo o processo de instalação de uma estrutura se torne bastante mais flexível visto que todas as instalações são feitas em *YAML*. Uma das grandes vantagens é o seu potencial de reutilização de *playbooks* - isto evita todo o mecanismo de uma reinstalação do zero de qualquer sistema. Para além disso, esta característica do *Ansible* permite que se utilizem módulos desenvolvidos por terceiros de maneira a estes poderem ser integrados numa panóplia de projetos, consoante a preferência.

Foram então usados inventários dinâmicos, de forma a minimizar ao máximo qualquer tipo de intervenção manual durante qualquer passo.

Foi empregue também a ferramenta *Docker*, também *open-source* que, conforme experienciado nas aulas práticas da disciplina, automatiza o *deployment* de aplicações em *containers* portáteis e auto-suficientes, ideais para que corram na *cloud*. Isto revelou-se importante no sentido em que tornou-se desnecessário instalar qualquer dependência do *Node.js* para o funcionamento, visto que já se encontra incluído na *Docker image*. Similar ao *Ansible*, o *Docker* também permite que se eliminem configurações repetitivas. Assim, a ideia é haver *deployment* de imagens que contém aplicações e as suas dependências.

O facto de termos tido contacto com esta ferramenta nas aulas práticas trouxe-nos bastante mais à vontade para a empregarmos neste projeto.

Para a instalação da aplicação decidiu seguir-se a guia de instalação da ferramenta *wiki.js* para o *docker*. Foi então criado um ficheiro ansible que dinamiza todos os passos de deployment da aplicação.

Para a criação de máquinas virtuais na *cloud* onde a aplicação será instalada, foram criadas através do *ansible* com recurso aos módulos da *google cloud*, nomeadamente `gcp_compute_instance`, `gcp_compute_instance_info`, `gcp_sql_instance` e `gcp_sql_instance_info`. Estes módulos forma usados para fazer a criação das máquinas virtuais e das bases de dados; os módulos de *info* foram usados para ser possível aceder aos endereços de IP dessas máquinas.

5.1 Serviços do Google Cloud Platform

A *Google Cloud Platform* é uma plataforma de grandes recursos, com um grande leque de serviços bastante robustos e fiáveis que permite a gestão de diferentes recursos computacionais consoante o objetivo que se esteja a empregar.

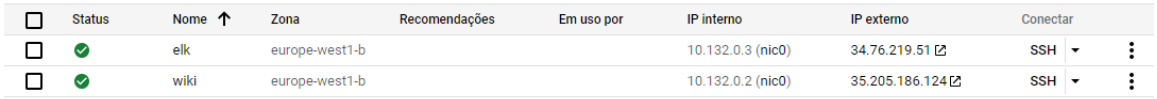
Dentro do ambiente do *GCP* encontramos diferentes ferramentas com as quais interagimos, a saber:

- Máquinas virtuais e respetivas instâncias

Definimos desde logo máquinas virtuais, tomando partido das definições da *Google Cloud Platform* do tipo *e2-medium*, com 2 **cores**, 4GB de RAM e 15 *GB* de disco.

Estes valores foram acertados tendo em conta os requisitos necessários ao funcionamento da aplicação, mas também consoante a nossa experiência ao longo do desenrolar do projeto.

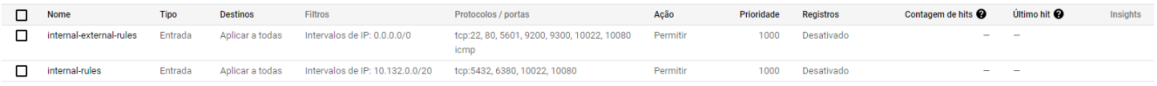
Teve-se que definir assim um disco para cada uma das instâncias e, para além disso, foi também crucial ter uma placa de rede associada a cada *VM*. Estas estruturas são obviamente necessários para o funcionamento distribuído na *cloud*.



Status	Nome	Zona	Recomendações	Em uso por	IP interno	IP externo	Conectar
✓	elk	europe-west1-b			10.132.0.3 (nic0)	34.76.219.51	SSH
✓	wiki	europe-west1-b			10.132.0.2 (nic0)	35.205.186.124	SSH

- Firewall

Tendo em conta as ferramentas de monitorização e a base de dados, foi necessário definir regras de *firewall* para permitir tráfego nos *ports* respetivos.



Nome	Tipo	Destinos	Filtros	Protocolos / portas	Ação	Prioridade	Registros	Contagem de hits	Último hit	Insights
internal-external-rules	Entrada	Aplicar a todas	Intervalos de IP: 0.0.0.0/0	tcp:22, 80, 5601, 9200, 9300, 10022, 10080 icmp	Permitir	1000	Desativado	–	–	
internal-rules	Entrada	Aplicar a todas	Intervalos de IP: 10.132.0.0/20	tcp:5432, 6380, 10022, 10080	Permitir	1000	Desativado	–	–	

5.2 Componentes replicados

Para a solução proposta, fez-se uma replicação da base de dados, que permite, em caso de falha da mesma, o sistema continuar a funcionar. O grupo considerou necessária apenas uma replicação, no entanto um simples comando de *loop* na criação da réplica, permite aumentar o número de instâncias.

6 Ferramentas de monitorização

Era necessário recolher dados e métricas relativas aos componentes aplicacionais e enviá-los para uma camada de monitorização para fazer controlo dessas mesmas métricas e tirar ilações relativas aos mesmos. Discriminam-se assim as ferramentas utilizadas para consumar este requisito:

- *ElasticSearch*

É uma ferramenta de análise de dados, permitindo receber e efetuar pesquisas sobre quaisquer tipo de métricas recebidas.

- *MetricBeats*

Esta ferramenta permite a monitorização da performance de cada componente da infraestrutura através da recolha de métricas, como o *CPU*, memória das instâncias ou até mesmo da base de dados.

- *Kibana*

Ferramenta que analisa grandes quantidades de dados em tempo real, dando a possibilidade de visualizar graficamente os dados obtidos, permitindo uma análise apelativa e intutiva.

De maneira recolhermos dados relativos aos diferentes componentes que constituem o *Wiki.js*, utilizamos

- Métricas de Performance do Sistema

Métricas associadas com o *CPU*, memória e utilização do disco. Este tipo de métricas caracterizam-se por identificar diferentes tipos de áreas de falha, evitando preventivamente que hajam problemas em grande escala, e levando a que hajam alterações ao sistema em si, evitando males maiores.

- Métricas de Performance da Aplicação

Este tipo de métricas está associado aos componentes da aplicação, inerentes ao sistema como por exemplo a Base de Dados. São dados úteis para controlar os processos que estão a ser executados nos servidores, e identificar quais deles estão a ter um impacto negativo na eficiência do sistema.

7 Conclusão

O desenvolvimento deste trabalho prático permitiu-nos adquirir bastante informação sobre como dar *deployment* de uma aplicação no mundo real, onde persiste a necessidade de garantir fiabilidade dos serviços e garantir a escalabilidade de uma aplicação.

Neste trabalho prático foi apresentada todo o esqueleto associado à implementação e automatização de cada camada implementada. Foi extremamente importante inserir e utilizar o *Ansible*, visto que a sua documentação se revelou extremamente útil. Sentimos alguns problemas técnicos, nomeadamente na criação de máquinas na *Google Cloud Plataform* com o uso de *Ansible* que, por motivos que desconhecemos, demoravam imenso tempo a serem geradas na plataforma. Houve alguns transtornos associados a isto, aos quais lamentamos, pois puseram em causa a qualidade do trabalho.

Existiram bastantes retrocessos e problemas ao longo deste trabalho, mas ficamos cientes do esforço que depositamos. Sabemos que existem falhas, nomeadamente nas últimas 2 fases. Não sendo desculpa, houve alguns problemas que nos afetaram diretamente, onde esperávamos ter as primeiras duas etapas realizadas muito mais rapidamente, mas infelizmente o tempo que alocamos à elaboração do trabalho foram quase exclusivamente consumidos pelas duas primeiras fases deste trabalho.