Relatório Final SCC – Projeto 2

1. Introdução

Este projeto teve como objetivo reimplementar o sistema TuKano com base no projeto base fornecido para o desenvolvimento do Projeto 1. Esta abordagem permitiu aproveitar a estrutura base sem as alterações introduzidas na solução PaaS do Projeto 1, adaptando-a diretamente para uma arquitetura baseada em infraestrutura como serviço (IaaS). Utilizaram-se Docker e Kubernetes para gerir os serviços e implementaram-se novas soluções para o armazenamento de dados, autenticação de sessões e gestão de blobs, ao substituir os serviços PaaS previamente usados.

Inicialmente, foi usado o projeto 1 como base para o projeto 2, mas após alguns problemas, foi decidido começar novamente, colocando diretamente apenas algumas classes já feitas durante a primeira tentativa de implementação.

---

2. Implementação em Azure

A implementação no Azure Kubernetes Service envolveu a criação de containers para cada um dos serviços do Tukano e a configuração de YAMLs do Kubernetes para orquestração.

2.1 Reimplementação a partir do Projeto Base

Para este projeto, utilizámos o código base inicial (antes de qualquer alteração relacionada com o Projeto 1). Esta decisão garantiu maior flexibilidade e evitou dependências de serviços PaaS, como o Azure Blob Storage, Cosmos DB ou Azure Cache for Redis, utilizados no Projeto 1. Desta forma, construímos as soluções IaaS diretamente sobre o código base, e asseguramos uma integração simplificada com Docker e Kubernetes.

2.2 Persistent Volumes para Armazenamento de Blobs

No Projeto 1, o Azure Blob Storage era utilizado como serviço PaaS para armazenar os vídeos e outros ficheiros. No entanto, no contexto do Projeto 2, implementou-se uma solução IaaS através de Persistent Volumes no Kubernetes.

Um Persistent Volume Claim foi configurado para disponibilizar um espaço de armazenamento persistente que podia ser montado diretamente no contentor do serviço de

blobs. Este armazenamento garantiu que os dados permanecessem intactos, mesmo em casos de falha reinício do contentor.

O diretório de armazenamento foi configurado como um ponto único para todos os ficheiros carregados pelos utilizadores. Esta abordagem assegurou consistência e simplicidade, e manteve o sistema independente de serviços externos. O PVC foi dimensionado para lidar com o volume de dados esperado, ao permitir ainda escalabilidade caso fosse necessário.

2.3 PostgreSQL para Gestão de Dados

Substituímos o Cosmos DB, utilizado no Projeto 1, por uma instância de PostgreSQL gerida no AKS. Esta base de dados foi escolhida por ser relacional, ao garantir total compatibilidade com as queries SQL desenvolvidas no projeto inicial.

A configuração incluiu a criação de uma instância única de PostgreSQL no cluster, com credenciais protegidas e associadas ao sistema de autenticação do Kubernetes. A base de dados foi configurada com tabelas para armazenar 'users', 'shorts', 'likes' e 'followers'.

Durante a migração, verificámos a necessidade de adaptar a camada de acesso a dados para garantir uma integração eficiente entre o código da aplicação e a base de dados. Optámos por manter a estrutura relacional para facilitar consultas complexas e assegurar a consistência dos dados em operações como criação, edição e eliminação.

2.4 Autenticação de Sessão

No contexto da segurança, implementou-se um sistema de autenticação de sessão no serviço de blobs. Este sistema assegura que apenas utilizadores autenticados podem realizar operações como upload e download.

A solução baseou-se na geração de tokens únicos associados a cada sessão. Estes tokens eram validados a cada pedido feito ao serviço de blobs, garantindo que apenas utilizadores com permissões válidas conseguiam aceder aos dados. A implementação do sistema de autenticação exigiu alterações na camada de lógica da aplicação, com o objetivo de verificar os tokens antes de qualquer operação.

2.5 Gestão com Kubernetes

A gestão de todo o sistema foi realizada utilizando Kubernetes, através de uma combinação de imagens Docker personalizadas e ficheiros de configuração YAML. Cada componente foi configurado de forma independente para garantir modularidade e facilitar a manutenção.

Orquestração dos Componentes:

- Serviço de Blobs: Configurado para utilizar um endereço interno no cluster (ClusterIP), acessível apenas pelas restantes componentes do sistema. Este serviço dependia do PVC para armazenamento.
- Tukano Application: A aplicação principal foi configurada como um serviço separado, responsável por gerir as operações entre os utilizadores, a base de dados e os vídeos.
- PostgreSQL: A base de dados foi gerida num contentor isolado, com acesso limitado para reforçar a segurança e evitar interferências externas.

Escalabilidade e Tolerância a Falhas:

Com a utilização de Kubernetes, foi possível configurar réplicas para os serviços mais críticos, garantindo alta disponibilidade e distribuição equilibrada da carga. Em caso de falha de um pod, o Kubernetes reiniciava automaticamente a instância, assegurando a continuidade do sistema.

Gestão de Recursos:

Os serviços foram configurados com limites de CPU e memória para evitar consumo excessivo de recursos no cluster. Esta configuração garantiu uma utilização eficiente da infraestrutura e preveniu possíveis sobrecargas.

---

3. Testes de Desempenho

Os testes foram realizados de forma manual, dado que não utilizámos a ferramenta Artillery para simular cargas. Os testes envolveram interações diretas com os serviços expostos pelo Kubernetes, utilizando ferramentas como Postman, e testes manuais de autenticação.

Cenários Testados

- Upload e Download de Blobs: Validámos o armazenamento e recuperação de blobs através do serviço.
- Gestão de Utilizadores e Shorts: Confirmámos a funcionalidade de criação, edição e remoção de dados armazenados no PostgreSQL.
- Autenticação de Sessões: Verificámos que as operações em blobs requeriam tokens de autenticação válidos.

---

4. Conclusão

A migração do sistema Tukano para uma arquitetura baseada em Kubernetes foi um desafio difícil que resultou na modernização da infraestrutura do sistema. Substituímos serviços PaaS por alternativas baseadas em containers e com a utilização de Kubernetes.

Embora os testes tenham sido manuais, conseguimos confirmar a correta integração dos serviços e o funcionamento das principais funcionalidades do sistema. Assim, acreditamos que o desempenho é compatível com os requisitos e objetivos do projeto.

Realizado por:

André Santos, 71802

Tiago Santos, 71921