Licenciatura em Engenharia Informática e de Computadores
Unidade Curricular de Computação na Nuvem

Projeto Final CN2223TF

Autores - Grupo 14

Gonçalo Silva - 47255
António Antunes - 47280

Docente: José Simão

Semestre de Verão 2022/2023

1 junho, 2023

Índice

Introdução

Este capítulo introdutório, irá apresentar o trabalho final realizado no âmbito da disciplina de Computação em Nuvem, juntamente com a estrutura do relatório.

Neste projeto, designado por *CN2223TF*, o nosso grupo teve a oportunidade de aplicar e consolidar os conhecimentos adquiridos ao longo do semestre sobre os serviços disponibilizados pela *Google Cloud Platform (GCP)*, nomeadamente o *Cloud Storage*, *Pub/Sub*, *Firestore*, *Compute Engine*, *Cloud Functions*, *Vision API* e *Static Maps AP*I.

Objetivos

O principal objetivo foi planear e desenvolver um sistema distribuído para submissão e execução de tarefas de computação na nuvem, com requisitos de elasticidade, utilizando de forma integrada os serviços da GCP para armazenamento, processamento e comunicação.
Este sistema tem como objetivos:

- Processar imagens para verificar a existência de monumentos ou locais famosos (landmarks).
- Obter informações sobre os landmarks detetados numa imagem, tais como, o seu nome, localização geográfica (latitude e longitude), e o respetivo nível de certeza associado à identificação (valor entre 0 e 1)
- Obter uma imagem correspondente a um mapa da zona do monumento identificado.
- Obter os nomes de todas as imagens onde houve identificação de monumento com um grau de certeza acima de ***t***
- Garantir elasticidade do sistema, aumentando ou diminuindo a sua capacidade de processamento de imagens.

Arquitetura do Sistema

Na figura seguinte está apresentada a arquitetura utilizada neste sistema, tanto como as interações entre cada um dos componentes.

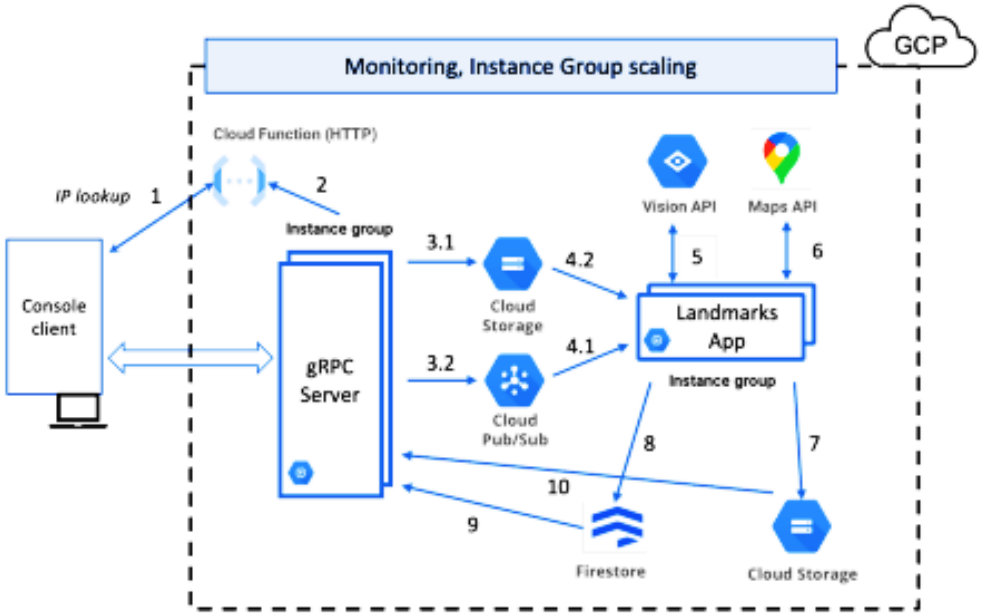

Figura 1- Arquitetura do Sistema

Fluxo de operações

A função de pesquisa, utilizada pela aplicação cliente para obter os endereços IP dos servidores gRPC, é implementada como uma Cloud Function. Essa função obtém os endereços IP das máquinas virtuais que fazem parte do grupo de instâncias. A aplicação cliente escolhe aleatoriamente um IP da lista retornada pela função e, em caso de falha na conexão com o servidor gRPC usando o IP selecionado, ela tenta outro IP ou realiza novamente a pesquisa para atualizar a lista de IPs e estabelecer uma nova conexão.

Após o envio de uma imagem, esta é armazenada no Cloud Storage e é retornado à aplicação cliente um identificador único para consultas futuras. Em seguida, uma mensagem contendo o identificador do pedido, o nome do bucket e do blob é enviada para um tópico Pub/Sub. Essa mensagem é processada por um worker responsável pela análise de imagem, que recebe o nome do bucket e do blob para processamento. O worker interage com o serviço Vision API para identificação de locais e com o serviço de mapas para obtenção de mapas estáticos.

Após o processamento da imagem, os mapas dos locais identificados são armazenados no Cloud Storage e as informações relevantes do pedido e do resultado da análise são armazenadas no Firestore.

A qualquer momento, a aplicação cliente pode solicitar informações sobre as imagens submetidas ao servidor gRPC usando o descritor do pedido. Para retornar essas informações, o servidor gRPC consulta o Firestore e/ou o Cloud Storage.

Contrato do servidor

A comunicação entre o cliente e o servidor é feita através de chamadas Remote Procedure Call (RPC) com gRPC. Estas chamadas seguem um contrato específico, definido pelo servidor, que estabelece os pedidos e respostas associadas a cada operação. Este contrato fornece uma estrutura predefinida para a troca de informações entre as partes envolvidas, permitindo uma interação eficiente e consistente entre o cliente e o servidor.

O contrato define as seguintes operações realizadas em protobuf:

```
service CN2223TFService {
  rpc imageSubmit (stream ImageUploadRequest) returns (IdentifierResponse);
  rpc getLandmarks (IdentifierRequest) returns (LandmarksResponse);
  rpc getStaticMap (IdentifierRequest) returns (stream StaticMapResponse);
  rpc getImagesByConfidence (ConfidenceRequest) returns (FilteredImagesResponse);
}
```

Figura 2 - Operações definidas no contrato protobuf

imageSubmit

A operação ***imageSubmit*** tem como objetivo o carregamento de uma imagem para que seja efetuada a deteção de monumentos ou locais famosos (Landmarks). O cliente envia uma imagem para o servidor e recebe como resposta um identificador único para posteriormente obter os landmarks identificados e outras informações.

Quanto à comunicação entre cliente e servidor, decidimos utilizar chamadas que envolvem streaming do cliente, ou seja, o cliente envia uma sequência de mensagens ao servidor e apenas recebe uma resposta do mesmo.

getLandmarks

A operação ***getLandmarks*** recebe como parâmetro o identificador retornado pela operação descrita anteriormente, e o servidor com esse identificador vai retornar uma lista com todos os landmarks e o respetivo grau de certeza de cada um. Para simplificar a interação entre o cliente e o servidor, foi adotado um modelo de comunicação unária. Neste modelo, o cliente envia um pedido ao servidor e este responde com uma única resposta contendo a lista de nomes dos landmarks encontrados na imagem.

getStaticMap

Na operação ***getStaticMap*** o cliente insere novamente o identificador da imagem e recebe do servidor uma imagem com o mapa estático de uma das localizações identificadas. Para que o servidor não tenha que enviar os bytes da imagem que representa o mapa estático na sua totalidade em apenas 1 mensagem, optamos por utilizar chamadas com stream do servidor.

getAllByConfidenceLevel

A operação ***getAllByConfidenceLevel*** tem como objetivo obter os nomes de todas as fotos onde houve identificação de um monumento com um grau de certeza acima de t (por exemplo, acima de 0,6) e o respetivo nome do local identificado, recebendo os parâmetros necessários para fazer esta filtragem, como tal, optamos por utilizar um tipo de chamadas unária visto que apenas retornamos uma lista com as imagens e o respetivo landmark que estão dentro do critério.

Servidor

Uma vez que o contrato esteja definido, o servidor irá implementar as 4 operações estabelecidas pelo contrato. O servidor gRPC utiliza vários serviços GCP tais como *Firestore*, *Cloud Storage* e *Pub/Sub*.

O servidor responde às perguntas dos clientes e atribui tarefas de deteção de landmarks aos workers, publicando mensagens no *Pub/Sub* para que estes possam consumi-las, ele também faz upload de imagens para o *Google Cloud Storage* e usa o *Firestore* como um base de dados para armazenar e consultar informações sobre landmarks detetados em imagens processadas pelo servidor.

imageSubmit

Nesta primeira operação o cliente fornece o path da imagem ao servidor, um exemplo de uma imagem está demonstrado na figura seguinte.

Figura 3 - imagem fornecida pelo cliente

Depois da imagem ser submetida pelo cliente, o servidor vai começar por gerar um identificador de forma aleatória (adicionando um randomUUID ao nome da imagem que o cliente forneceu), de seguida guarda a imagem com esse identificador único no *Storage.* Além disso, o servidor encaminha a tarefa de deteção para os workers, enviando uma mensagem que contém o id gerado para um tópico denominado *landmarks-detection* no serviço *Pub/Sub*. Os workers, ao receberem a tarefa, executarão todo o processo mencionado na Secção 2.3.

No final é retornado para o cliente o identificador gerado, para que possam ser realizadas as restantes operações.

getLandmarks

O papel do servidor nesta operação já referida anteriormente, consiste em retornar a informação pretendida sobre todos os landmarks encontrados numa dada imagem, para isso é realizada uma query à coleção *"images"* para acessar a imagem pretendida e de seguida, obter os documentos presentes na coleção "landmarks", para cada um destes documentos que iremos obter será extraído os campos local, latitude, longitude e confidence, obtendo assim todos os dados pretendidos da *Firestore.*

getStaticMap

De modo a obter uma imagem do mapa estático da imagem correspondente ao identificador que o cliente forneceu, o servidor obtém o blob que contém o mapa da imagem, pesquisando no bucket o nome do blob da imagem adicionando o prefixo *"static_map_".* Um exemplo desta imagem esta apresentado na figura 4.

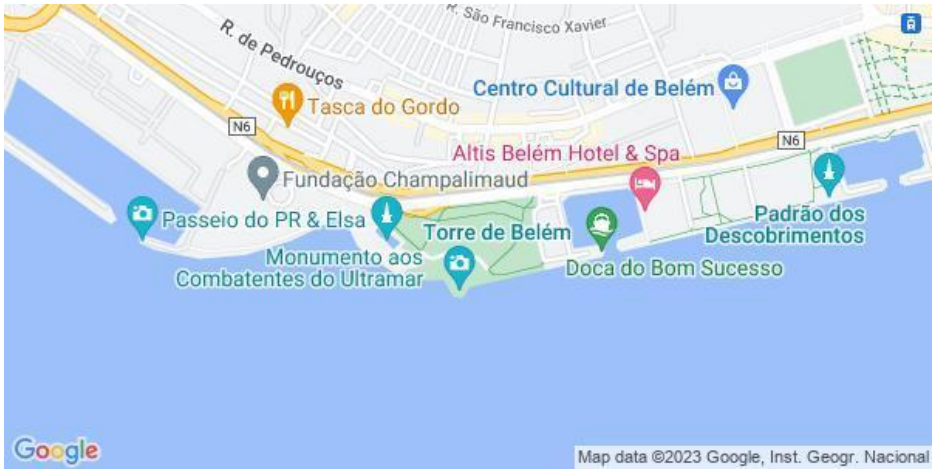

Figura 4 - Mapa estático retornado pelo servidor

getAllByConfidenceLevel

A operação ***getAllByConfidenceLevel*** tem como objetivo obter os nomes de todas as fotos onde houve identificação de um monumento com um grau de certeza acima de t (por exemplo, acima de 0,6) e o respetivo nome do local identificado, recebendo os parâmetros necessários para fazer esta filtragem.

Worker

A aplicação worker (Landmarks App), atua sobre uma subscrição de *Pub/Sub*, recebendo todas as mensagens associadas ao tópico landmarks-detection que reflete um pedido de processamento de imagem pelo servidor.

Um worker é uma entidade responsável por executar as ordens recebidas ao ler mensagens do tópico " *landmarks-detection* " no serviço Google *Pub/Sub*.
Neste trabalho, todos os workers estão associados à mesma subscrição, isso significa que, se um worker consumir uma mensagem, os demais workers não poderão consumi-la (padrão work-queue). Desta forma, o trabalho é distribuído entre os vários workers, garantindo que cada um deles execute uma tarefa diferente. Cada worker existente, ao receber uma imagem, tem como objetivos:

- Efetuar a deteção dos landmarks presentes numa imagem através do uso da *Google Vision API.*
- Gerar uma nova imagem que representa o mapa estático para um landmark encontrado na imagem e guardá-lo no *Storage*.

- Guardar informações(nome do local, latitude, longitude e nível de certeza) sobre os landmarks detetados numa imagem no *FireStore.*

Em primeiro lugar, quando um worker consome uma mensagem da sua subscription, recebe um ID que corresponde à imagem a ser processada. Esse ID é uma referência para a imagem armazenada no *Storage*. Para obter o conteúdo da imagem, é possível utilizar um URI presente no *Storage*, que é formado pela combinação do nome do bucket e o nome do blob, por exemplo: gs://<bucketName>/<blobName>.

Com isto, é realizada uma consulta ao serviço *Google Vision API*, cujo objetivo é obter informações sobre todos os landmarks detetados na imagem e com base nessas informações, o worker irá criar uma imagem com o prefixo *"static_map_"* e salvá-la no serviço *Cloud Storage*, no mesmo bucket em que a imagem original foi guardada. Além disso, as informações sobre os landmarks detetados na imagem serão armazenadas no *Firestore*.

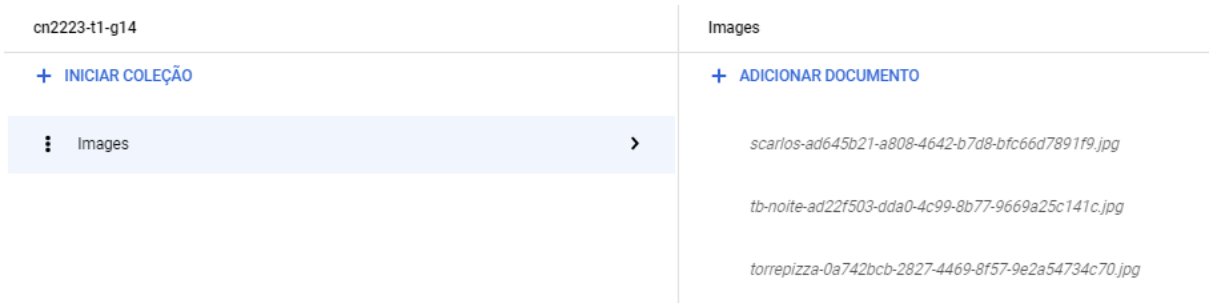

Figura 5 - Lista de imagens na FireStore

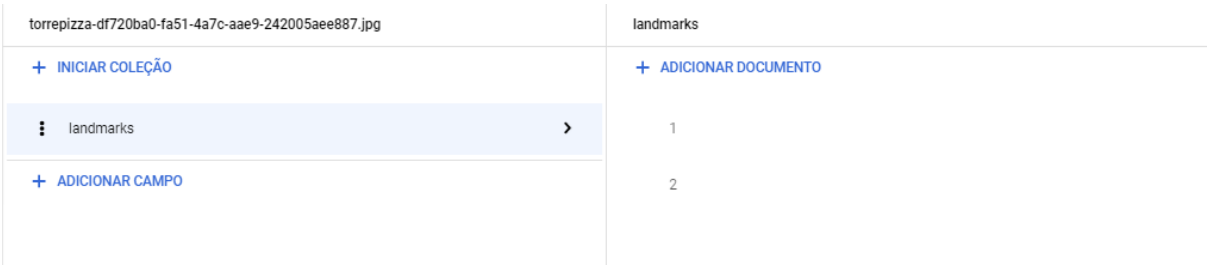

Figura 6 - Lista de landmarks contidos em uma imagem

Figura 7 - Landmark de uma imagem

Cliente

O cliente, tal como o servidor, implementa as operações do contrato que implementamos. Essas funções representam os 4 tipos de pedidos que o nosso cliente pode fazer ao servidor.

Observers

Para cada operação implementada do contrato é necessário termos um observer que fique a escuta de pedidos. O objetivo do observer e lidar com as respostas assíncronas enviadas pelo servidor para o cliente. Desta forma, o observador permite ao cliente interagir e responder às respostas e eventos assíncronos do servidor de maneira apropriada. É nos observers que são mostrados ao cliente informações e dados que resultam da resposta do servidor. De seguida, seguem-se algumas figuras que representam a resposta fornecida ao cliente em cada uma das operações que este implementa.

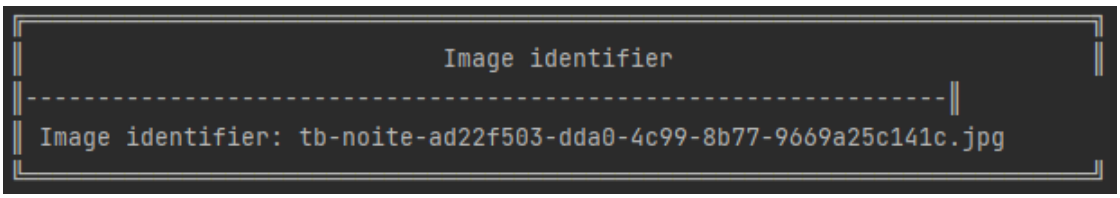

Figura 8 - Resposta da operação **ImageSubmit**

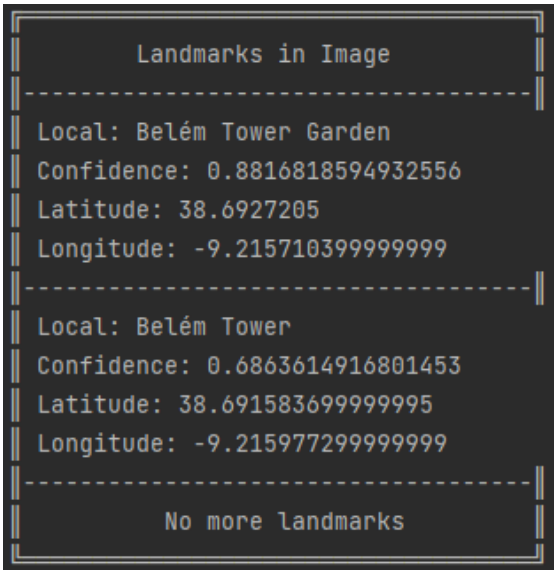

Figura 10 - resposta da operação **getLandmarks**

Figura 9 - resposta da operação **getStaticMap**

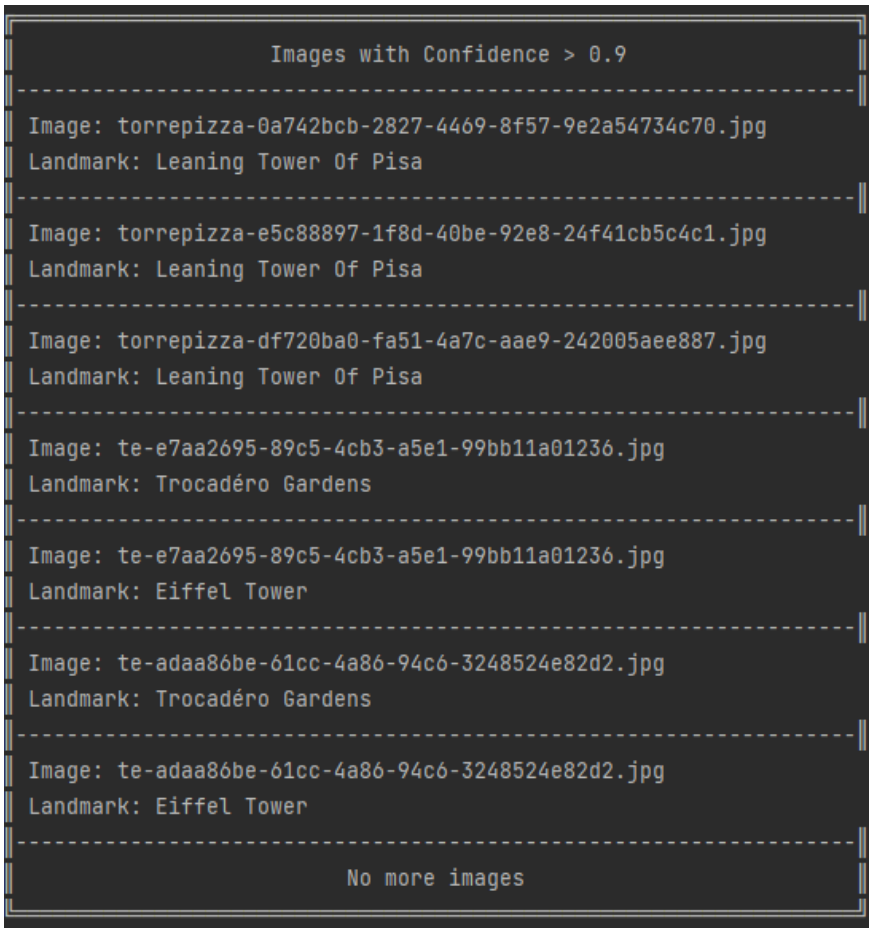

Figura 11 - resposta da operação getAllByConfidence

LookUp Function

Para que o cliente possa se comunicar com um dos servidores disponíveis, tem de primeiro saber os seus endereços IP. Para tal recorre ao serviço Lookup, que lhe devolverá todos os endereços associados aos servidores, dos quais o cliente escolherá um aleatoriamente. Isto permitirá distribuir a carga de solicitações de clientes entre os vários servidores disponíveis.

Para a implementação desse serviço, foi criada uma Cloud Function que funciona como um trigger HTTP e implementa o serviço Lookup, usado pela aplicação para obter os IPs das diferentes máquinas de um *Instance Group*. Para cada máquina virtual encontrada, que se apresente no estado Running, é extraído o seu endereço IP e adicionado a uma lista. No final esta lista conterá todos os IP's dos servidores gRPC e esta será enviada para o cliente.

Conclusão

Por fim, podemos concluir que ao longo do desenvolvimento deste trabalho o grupo conseguiu aplicar a aprofundar todos os tópicos lecionados ao longo do semestre para que fosse possível desenvolver um sistema capaz de detetar monumentos e locais famosos que faz uso de diversos serviços da plataforma Google Cloud mencionados ao longo do trabalho.

Em geral, a execução deste trabalho teve um resultado satisfatório, visto que todos os objetivos foram alcançados com sucesso, o que nos permitiu compreender a arquitetura dos sistemas distribuídos e as suas diferentes utilizações quanto a padrões de comunicação e interação.