Trabalho Final de Computação na Nuvem

G06

Licenciatura em Engenharia de Informática, Redes e Telecomunicações
Área Departamental de Eng. Eletrónica e Telecomunicações e Computadores
Instituto Superior de Engenharia de Lisboa

Trabalho realizado por: André Santos, nº48705
Ema Tavares, nº 48704
Fábio Santos, nº 48716

Docente: Fernanda Passos

Maio e Junho de 2023

Conteúdo

Introdução

O sistema CN2223TF tem como objetivo desenvolver uma solução de computação em nuvem utilizando os serviços integrados da Google Cloud Platform. O sistema é projetado para processar imagens e detectar monumentos ou locais famosos. Além disso, oferece elasticidade ao ajustar dinamicamente sua capacidade de processamento com base na carga de trabalho.

O sistema disponibiliza as suas funcionalidades por meio de uma interface gRPC, que permite aos clientes enviem ficheiros de imagem para detecção de monumentos. O sistema armazena esses arquivos no Cloud Storage e retorna um identificador único do pedido para posterior consulta de resultados. Os clientes podem obter uma lista de monumentos identificados, juntamente com suas coordenadas geográficas e níveis de confiança associados. Além disso, os clientes podem obter um mapa estático correspondente ao monumento localizado na imagem submetida, ou então obter todos os monumentos identificados pelo servidor com um grau de certeza superior a um valor determinado.

A arquitetura do sistema envolve vários componentes e serviços fornecidos pela Google Cloud Platform. O Cloud Storage é usado para armazenar as imagens e os mapas estáticos, enquanto o Firestore armazena informações relevantes sobre o processamento das imagens e os locais identificados. O Pub/Sub possibilita a troca de mensagens desacopladas entre os componentes do sistema, e o Compute Engine hospeda réplicas do servidor gRPC e instâncias da aplicação para identificação de monumentos. A Vision API é utilizada para a detecção de monumentos nas imagens, o Static Maps API é usado para obter imagens de mapas com base em coordenadas de latitude e longitude e o Cloud Function é usado para a aplicação cliente descobrir os ips das várias instâncias de servidores gRPC, para que esta possa ser estabelecida a ligação.

Após o envio das imagens, estas são armazenadas no Cloud Storage, e um identificador único é retornado para o cliente. Em seguida, uma mensagem é enviada para um tópico Pub/Sub com o identificador e informações de armazenamento para o processamento da detecção de monumentos. As Landmark Apps inscritos no tópico analisam as imagens usando a Vision API e recuperam mapas estáticos usando a Static Maps API. Os mapas processados são armazenados no Cloud Storage, e as informações relevantes são armazenadas no Firestore. O servidor gRPC acessa o Firestore e o Cloud Storage para fornecer informações sobre as imagens enviadas aos clientes.

Implementação do contrato

No presente trabalho começamos por implementar o contrato que vai ser implementado pela aplicação servidor e cliente. Foram definidos alguns métodos para que seja possível realizar várias operações como pedido no enunciado. As operações disponibilizadas pelo contrato são:

- Fazer o envio de uma imagem de modo a serem detetados os objetos presentes;
- Obter os nomes dos pontos de referência identificados na imagem, localização geográfica e nível de certeza (0 a 1);
- Obter o mapa da imagem submetida;
- Obter os nomes de todos os blobs onde houve identificação de monumento com um grau de certeza acima de t e o respetivo nome do local identificado.

Todas as operações realizadas pelo servidor encontram-se no ficheiro CN2223TF.proto e têm as seguintes assinaturas:

```
service CN2223TF {
  rpc uploadPhoto(stream Block) returns (BlobIdentifier);
  rpc getLandmarks(BlobIdentifier) returns (LandMarksResult);
  rpc getMapImage(BlobIdentifier) returns (stream MapResult);
  rpc getAccurateLandmarks (Accuracy) returns (AccuracyResult);
}
```

Figura 1 - Assinaturas dos métodos com as operações a realizar

A função **uploadPhoto** define que o cliente deve enviar streams de Block, composto por um *blobName, data e dataType* e que o servidor deve retornar um BlobIdentifier que é composto por um id.

A função **getLandmarks** define que o cliente deve enviar um BlobIdentifier, e que o servidor deve retornar um LandMarksResult, tendo este como atributo, um array de LandmarkElement, que por sua vez tens os atributos de *name, latitude, longitude e accuracy*.

A função **getMapImage** define que o cliente deve enviar um BlobIdentifier e que o servidor deve retornar streams de MapResult, compostas por bytes de mapas.

Por último a função **getAccurateLandmarks** define que o cliente deve enviar um Accuracy que é composto pelo atributo *accuracy* e que o servidor deve retornar um AccuracyResult que é composto por um array de LandMarkAccuracyResult, que por sua vez é composto pelos atributos de *name e imageName*.

Implementação do Lookup

O Cloud Functions permite que através de um pedido get para um determinado endpoint HTTP, que seja executado código na Google Cloud, desta forma quando uma aplicação cliente é executada, de forma que esta possa estabelecer uma ligação com o servidor, é necessário saber o endereço IP do servidor.

Nesta componente do sistema foi implementado uma *Google Cloud Function* que permite obter os endereços ip das várias instâncias de máquinas virutais a correr na *Google Cloud,* que se localizam no servidor “europe-west9-a” e que pertençam á instância “instance-group-server”, deste modo garantimos que os endereços ip que são retornados pela função correspondem apenas a endereços ip de máquinas servidoras que estão a desempenhar o papel de servidores gRPC.

Implementação do cliente

Quando a aplicação do cliente é executada, esta necessita de saber o endereço ip do servidor gRPC, e para tal vai fazer uso da Cloud Functions como foi explicado no tópico anterior.

É de notar que a função *lookup* retorna uma lista de *VMs* que se encontram a correr no *instance group* a que diz respeito às réplicas do servidor *gRPC*, pelo que a aplicação cliente ao receber a lista de endereços ip vai escolher um endereço de forma aleatória, de forma a evitar que haja congestionamento de algum dos servidores.

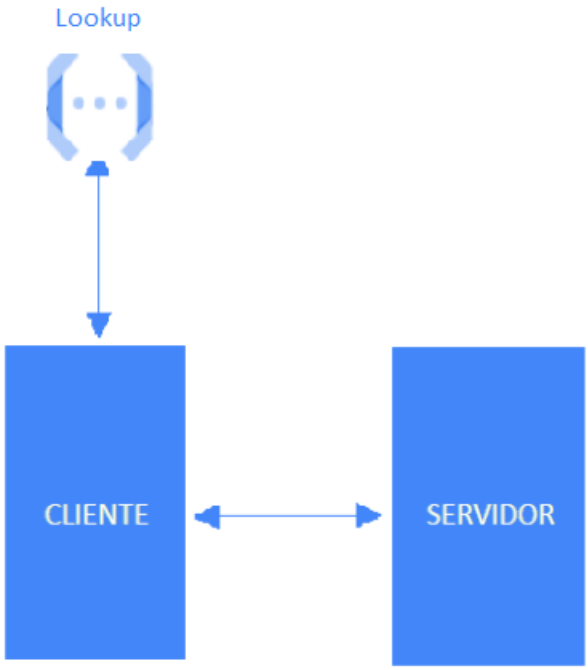

Figura 2 - Diagrama do cliente

Deste modo, observemos as funções necessárias para execução do cliente.

uploadPhoto

Esta função pede ao cliente que introduza o nome da imagem que pretende submeter para o servidor. Uma vez que o envio da imagem para o servidor é feito em streams, a imagem está a ser decomposta em Blocks, com um tamanho máximo de 1Mb. Para tal é usado um stub não bloquante que a cada chamada do método onNext envia um Block que leva no atributo dataos bytes da imagem e no fim chama o método onCompleted para informar o servidor que já concluiu a submissão da imagem.

Como retorno recebe um identificador do pedido, que servirá para futuras consultas de informações associadas á imagem que o pedido identifica

getResults

Como o nome indica, esta função tem como objetivo obter os resultados de uma imagem, para tal solicita ao cliente um id de pedido para uma foto que tenha sido previamente submetida, de forma que o servidor possa ir consultar a sua base de dados e enviar como resposta os monumentos presentes na imagem, a localização e o grau de certeza associado ao mesmo.

getMap

Esta função solicita ao cliente o id de um pedido previamente feito, de forma que possa enviar para o cliente uma imagem correspondente ao mapa do monumento presente na imagem associada ao pedido submetido. Caso a imagem tenha algum ponto de referência é enviado o mapa, mas na imagem correspondente ao identificador do pedido não exista o cliente é avisado que a imagem não contém pontos de referência.

photosNameWithScoreBiggerThan

Nesta função é pedido ao cliente que indique um grau de incerteza para o qual pretende obter todos os blobs que contenham pontos de referência com um valor de certeza maior que X. Para tal os clientes apenas têm de enviar o grau de certeza, recebendo um array de com o nome dos vários pontos de referência, bem como o nome de cada.

Implementação do Servidor

Como pode-se verificar na figura abaixo, o servidor vai atender aos pedidos do cliente, para tal vai fazer uso de vários serviços, como o gRPC para a receber os pedidos e enviar as respostas, o Cloud Storage para fazer armazenamento da imagem submetida pelo cliente, Pub/Sub para poder enviar uma mensagem para o tópico de Pub/Sub em que a Landmarks APP está subscrita, e o Firestore para poder aceder às várias informações geradas pela Landmarks APP.

De forma que o sistema tenha uma maior disponibilidade e garantia de elasticidade, foi criada uma instance group com réplicas do servidor, que podem ter uma dimensão máxima de 3 e uma mínima de 1 máquina virtual a correr, consoante seja necessário, uma vez que o sistema pode estar a necessitar de mais ou menos recursos consoante o número de pedidos gerados pelas aplicações clientes.

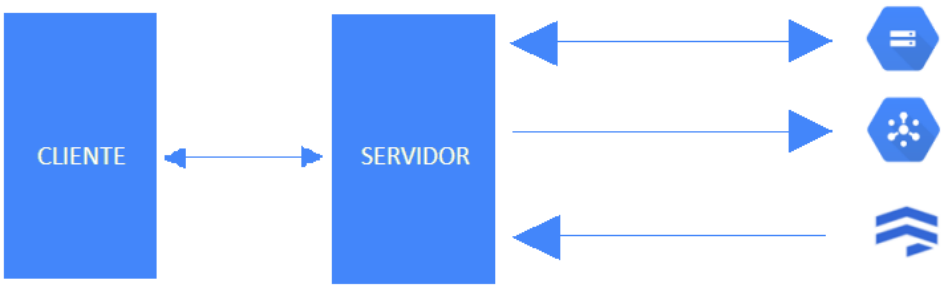

Figura 3 - Diagrama da implementação do servidor

De forma a poder correr com sucesso a aplicação do servidor é necessário definir nas variáveis do sistema a variável GOOGLE APPLICATION CREDENTIALS, apontando para o ficheiro que contém a chave da conta de serviço com as roles de *Cloud datastore Owner, Storage Admin e Pub/Sub Admin.*

Deste modo, observemos as funções necessárias para execução do servidor.

uploadPhoto

Essa função utiliza um stream de cliente e retorna um StreamObserver que permite que o cliente envie dados e finalize a operação. O servidor recebe um StreamObserver como parâmetro e usa um objeto para facilitar a escrita dos dados no armazenamento. A cada chamada do metodo onNext do StreamObserver o cliente vai guardar os bytes recebidos de bloco, no fim da passagem de todos os blocos o cliente chama o metodo onComplete e o servidor gera um identificador único do pedido e faz o upload dos bytes da imagem para o Cloud Storage. Ao finalizar o envio do sdados, o servidor publica no tópico de Pub/Sub uma mensagem, contendo o id do pedido, o blob onde a imagem está localizada e o bucket da mesma, de forma que a Landmarks APP possa ser despoletada e começar a processar a imagem. Ao finalizar o envio da mensagem Pub/Sub, o servidor informa ao cliente que a operação foi concluída enviando o identificador do pedido.

getLandmarks

Esta operação que recebe como parâmetro um identificador do blob e um StreamObserver vai consultar o Firestore de modo a obter as informações de uma certa imagem que foi enviada pela função anterior. Quando o servidor faz o pedido ao Firestore recebe um documento, e no campo *landmarks* recebe um array de pontos de referência detetados, como tal terá que adicionar cada elemento detetato á variável do LandMarksResult que é depois enviada aquando da chamada do método onNext() do StreamObserver recebido do cliente.

getMapImage

Esta operação recebe um identificador do blob e um StreamObserver como parâmetro, ele lida com o processo de obtenção de um mapa a partir do identificador do pedido. O processo começa por chamar o serviço Firestore para obter o nome do blob do mapa e consequentemente faz um pedido á Cloud Storage para fazer o download do blob do mapa para depois poder enviar em stream de MapResult para o cliente os bytes do mapa, como o envio dos dados é feito em stream, os bytes do mapa estão a ser repartidos em blocos de 1Mb e enviados a cada chamada do método onNext() do StreamObserver recebido como parâmetro do cliente. Após o envio de todos os blocos de MapResult é chamado o método onCompleted() do StreamObserver de forma a avisar o cliente sobre a conclusão do envio dos dados.

getAccurateLandmarks

O método getAccurateLandmarks recebe um objeto Accuracy e um StreamObserver como parâmetros. Ele lida com o processo de obtenção de nomes de fotos com grau de certeza maior que um determinado valorespecificado no atributo *accuracy* do objeto Accuracy recebido como parâmetro. Da maneira que os documentos Firestore foram armazenadas não foi possível fazer uma query para obter apenas os documentos com um valor de grau de certeza superior ao valor especificado, pelo que foi necessário obter todos os documentos, para depois o servidor percorrer os vários documentos e os vários elementos dos arrays presentes no campo *landmarks* dos documentos.

Depois de fazer a filtragem dos documentos, estes foram adicionados á variável de AccuracyResult que foi depois enviada para o cliente.

Implementação do Landmarks App

A aplicação *LandMarksApp*, responsável por detetar objetos em imagens, atua como um *subscriber* de um tópico de *pubsub*, recebendo todas as mensagens associadas a esse tópico, as quais representam solicitações de processamento de imagem feitas pelo servidor.

Tal como na aplicação foi necessário definir a variável de ambiente GOOGLE APPLICATION CREDENTIALS, para correr a aplicação do Landmarks App é necessário que esta referencie o documento json que contém a chave da conta de serviço com as permissões de *Cloud datastore Owner, Storage Admin, Pub/Sub Admin* e ainda de VisionAI Admin. Foi também necessário definir uma variável no código que contém a chave api da library Maps Static API que teve de ser ativada através da consola web do Google Cloud Platform.

De forma que o sistema tenha uma maior disponibilidade e garantia de elasticidade, foi criada uma instance group com réplicas da aplicação Landmarks App, que podem ter uma dimensão máxima de 2 e uma mínima de 0 máquinas virtuais a correr, consoante seja necessário, uma vez que o sistema pode estar a necessitar de mais ou menos recursos consoante o número de mensagens Pub/Sub enviadas pelo servidor gRPC.

Quando a aplicação é executada, esta vai criar caso necessário o tópico de Pub/Sub e se necessário uma subscrição de forma que possa criar então um susbcriber na subscrição, permitindo assim receber as mensagens publicadas no tópico Pub/Sub que despoletam a o arranque do processamento da imagem presente no blob.

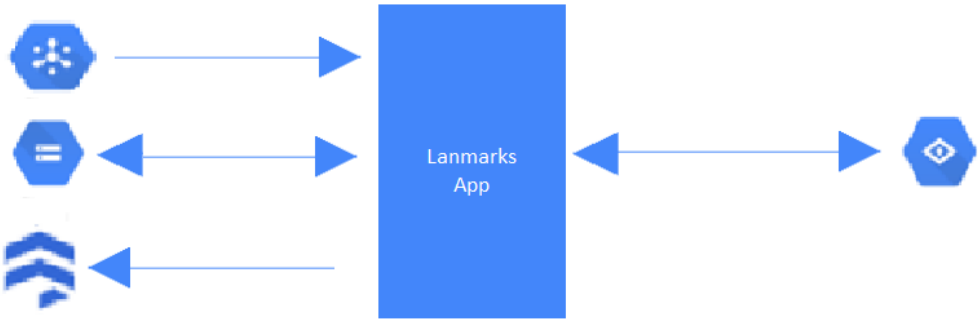

Figura 4 - Diagrama do Landmarks App

Deste modo, observemos as funções necessárias para execução do servidor.

detectLandmarksGcs

Nesta função, que recebe como parâmetros os vários campos recebidos na mensagem PubSub, como o identificador do pedido, o nome do blob e o nome do bucket onde está armazenada a imagem no Cloud Storage, está a ser criado um AnnotateImaegRequest da imagem a analisar, que é depois submetido á Vision API. Com os resultados obtidos da Vision API está a invocada a função

getStaticMapSaveImage e retornada uma classe DetectLandmarksGcs que contém os bytes do mapa e os vários pontos de referência identificados.

getStaticMapSaveImage

Nesta função que recebe como parâmetros a latitude, longitude e chave api, estão a ser feitos pedidos á api MapsStaticApi, passando as coordenadas recebidas como parâmetro, o resultado do retorno dos pedidos é os bytes que correspondem ao mapa estático da localização do ponto de referência identificado pela VisionApi.

Conclusão

A conclusão do projeto evidenciou a capacidade do grupo em desenvolver um sistema capaz de detetar objetos em imagens, gerar imagens processadas e armazená-las. Durante o processo, foram aplicados os conhecimentos teórico-práticos adquiridos ao longo do semestre de 22/23, abordando os serviços disponíveis na plataforma de nuvem escolhida, a *Google Cloud Platform*.

Como resultado da execução satisfatória do trabalho, o grupo obteve um maior entendimento sobre as características das arquiteturas de sistemas distribuídos, assim como as diversas aplicações de padrões de comunicação e interação. Além disso, foi possível explorar as potencialidades e competências da computação em nuvem, compreendendo a importância da sincronização e coordenação em ambientes distribuídos.

Esses resultados demonstram a aplicação dos conceitos estudados e a capacidade do grupo em enfrentar desafios práticos relacionados à computação em nuvem e sistemas distribuídos.