Computação na Nuvem

Licenciatura de Engenharia Informática, Redes e Telecomunicações

Departamento de Engenharia Eletrónica e Telecomunicações e de Computadores

2021/22

Verão

Trabalho Final

CN2122TF

Gustavo Campos (A47576) | Tiago Cebola (A47594) | Inês Sampaio (A47758)

Docente: José Simão

12 DE JUNHO DE 2022

Índice

Lista de Figuras iii

Lista de Abreviaturas, Acrónimos e Siglas iv

1. Introdução 1

2. Funcionalidades 3

3. Serviços do gRPC 5

3.1. Cloud Storage 5

3.2. Firestore 5

3.3. Pub/Sub 6

3.4. Compute Engine 6

3.5. Cloud Functions 6

3.6. Vision API 7

4. Arquitetura do Sistema 9

4.1. Contrato 9

4.2. gRPC Server 11

4.3. IpLookup Function 12

4.4. Client App 13

4.4.1. Virtual Machines IP Address 13

4.4.2. Submissão de uma Imagem 14

4.4.3. Obtenção do resultado do processamento 15

4.4.4. Obtenção do resultado do processamento 15

4.5. Detect Objects App (Worker) 15

4.5.1. Vision API 16

4.6. Monitor Function 18

5. Formato de dados 19

5.1. Mensagens Pub/Sub 19

5.2. Documento da Monitor Function 20
5.3. Documentos de deteção de objetos 20
6. Conclusão 23
7. Bibliografia 24

Lista de Figuras

Figura 1 - Arquitetura do CN2122TF 3
Figura 2 - RPCs do arquivo proto 9
Figura 3 - IpLookup Batch Script 13
Figura 4 - Monitor Function Batch Script 18
Figura 5 - Formato da mensagem Pub/Sub 19
Figura 6 - Formato do documento de monitorização 20
Figura 7 - Formato da coleção de deteção de imagens 20
Figura 8 - Formato da coleção "objects" 21

Lista de Abreviaturas, Acrónimos e Siglas

UC	Unidade Curricular
GCP	Google Cloud Platform
GCS	Google Cloud Storage
GCF	Google Cloud Firestore
gRPC	Google Remote Procedure Calls
ACL	Access Control List
URL	Uniform Resource Locator
API	Application Programming Interface
PUB	Publisher
SUB	Subscriber
HTTP	Hypertext Transfer Protocol
ACK	Acknowledged
JSON	JavaScript Object Notation

1. Introdução

Este relatório apresenta e discute o trabalho final da UC Computação na Nuvem e tem como principal objetivo realizar um sistema para submissão e execução de tarefas, com requisitos de elasticidade, utilizando de forma integrada os serviços da Google Cloud Platform para armazenamento, comunicação e computação, nomeadamente, Cloud Storage, Firestore, Pub/Sub, Compute Engine, Cloud Functions e Vision API.

Neste documento são apresentados os passos realizados para o desenvolvimento do sistema, tal como algumas das opções tomadas. Com esse intuito, o restante documento foi organizado segundo a seguinte estrutura:

- O capítulo 2 aborda em um contexto geral a arquitetura do trabalho e a interação entre os várias componentes.
- O capítulo 3 consiste em uma abordagem mais especifica sobre o papel de cada serviço disponibilizado pelas várias componentes e como estes funcionam.
- No capítulo 4 é apresentada a implementação de cada componente especificando as suas funcionalidades e interações.
- No último capítulo são apresentados os formatos dos dados utilizados para a comunicação no processamento das tarefas.

2. Funcionalidades

Neste capítulo apresentamos o sistema designado CN2122TF, com o objetivo de detetar múltiplos objetos como por exemplo bicicletas, cadeiras ou quadros, em ficheiros de imagem no formato (JPG, PNG, etc.) de forma a gerar novas imagens com anotações das zonas onde estão presentes os diversos objetos atribuindo aos mesmos um grau de certeza com que a API detetou o objeto.

O sistema possui elasticidade, podendo aumentar ou diminuir a capacidade de processamento de imagens conforme o número de instâncias de máquinas virtuais em execução, sendo esta um requisito dinâmico atribuído conforme a necessidade de maior ou menor recurso computacional. As funcionalidades do sistema estão disponíveis para as aplicações cliente através de uma interface gRPC com as seguintes operações:

- Submissão de um ficheiro do tipo imagem para deteção de objetos.
- Obtenção de uma nova imagem anotada com as zonas onde foram detetados os objetos, tal como a respetiva lista de nomes e o grau de certeza de cada objeto.
- Obter todos os nomes de ficheiros armazenados entre duas datas, que contêm um objeto com determinado nome e grau de certeza mínimo.

As diferentes interações entre os componentes do sistema estão representadas na Figura 1.

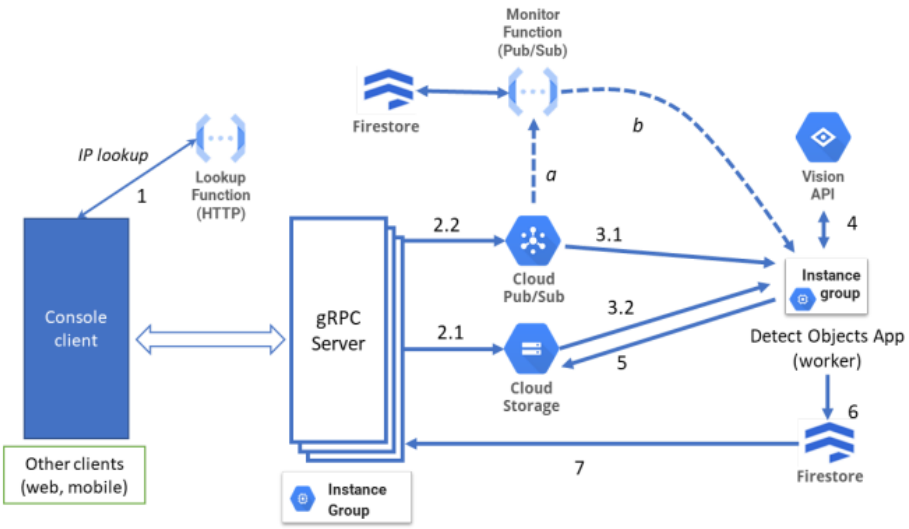

Figura 1 - Arquitetura do CN2122TF

3. Serviços do gRPC

Neste capítulo é dado a conhecer as funções disponibilizadas pelas APIs da Google Cloud Platform compreendendo os respetivos papeis e que de que forma irão desempenhar em um dado contexto do sistema.

3.1. Cloud Storage

A Cloud Storage é uma ferramenta escalonável dirigida ao armazenamento de dados não estruturados na nuvem. É baseado em coleções/agrupamentos (buckets) de objetos (blobs) de qualquer tipo.

Assim, é extremamente seguro utilizar esta infraestrutura para armazenar arquivos importantes, sendo possível aceder a partir de qualquer dispositivo com internet sempre que necessário. É de se notar que uma das suas mais valia encontra-se na alta disponibilidade através da replicação distribuída geograficamente além de que é tolerante a falhas de energia, hardware e humanas. Este serviço irá ser responsável por armazenar as imagens submetidas para posteriormente serem processadas tal como as imagens anotadas resultantes deste processamento.

3.2. Firestore

O Firestore é uma base de dados em nuvem NoSQL, onde os dados são armazenados como documentos de acordo com os requisitos da aplicação. Este serviço é considerado flexível e escalonável com o propósito de armazenar e sincronizar dados através de “realtime listeners” para o desenvolvimento de aplicações com um custo reduzido.

No desenvolvimento deste modulo, a qualquer momento, as aplicações cliente podem pedir ao servidor gRPC informações sobre os ficheiros submetidos, nomeadamente os objetos encontrados nas imagens submetidas sendo ainda possivel filtrar entre datas, nome e por grau de certeza.

3.3. Pub/Sub

O Pub/Sub é um serviço de mensagens assíncrono que implementa o padrão publish/subscribe, este mantêm estado sobre um conjunto de tópicos e de subscrições associada a cada tópico, separando os serviços de produção dos serviços de processamento de eventos. Outra característica deste é oferecer armazenamento de mensagens com alta durabilidade, além da entrega das mesmas em tempo real através de um desempenho consistente em larga escala e com alta disponibilidade.

Um intermediário Pub/Sub mantém estado sobre um conjunto de tópicos em união com os subscritores de cada tópico e cada subscritor pode subscrever a um ou mais tópicos. Esta componente na arquitetura do sistema tem a função de realizar a troca desacoplada de mensagens entre os componentes do sistema.

3.4. Compute Engine

O serviço Compute Engine é responsável por oferecer máquinas virtuais que são executadas em *data centers* e em redes globais de fibra da empresa Google. Todas as ferramentas e serviços disponíveis através do Compute Engine são funcionais no serviço em nuvem. Estas oferecem o balanceamento de carga, com o devido escalonamento das instâncias individuais para instâncias globais.

Este serviço é utilizado para alojar máquinas virtuais onde se executam os servidores gRPC e as aplicações de deteção e caraterísticas de imagens subscritas a um tópico.

3.5. Cloud Functions

As Cloud Functions são um ambiente de execução sem servidor para criar e conectar serviços em nuvem, ou seja, não existem servidores para provisionar, gerenciar, corrigir ou atualizá-las. As funções são dimensionadas automaticamente e são altamente disponíveis e tolerantes a falhas, assim podemos dizer que as funções da Cloud são relevantes na construção de aplicações *back-end* sem servidor, para realizar processamento de dados em tempo real e criar aplicativos inteligentes.

Este serviço é usado para implementar duas funções, uma que possui um tipo de *trigger* HTTP, ou seja, o seu desencadeamento é gerado por um pedido HTTP que irá realizar o *lpookup* dos endereços dos servidores gRPC disponíveis. A segunda função possui um *trigger* do tipo Topic que supervisiona o ritmo de pedidos de submissão de imagens que por sua vez é responsável por redimensionar o número de instâncias das VMs.

De forma a hospedar funções na Cloud é necessário utilizar um batch script que possui o nome da função, o tipo de acesso, o nome da classe java, o evento que ira desencadear a execução da função, a região de deployment, a diretoria JAR com as bibliotecas e dependências utilizadas e por fim a conta de serviço utilizada com as permissões necessárias para o acesso aos serviços.

3.6. Vision API

O serviço Vision API permite o acesso a deteção de visão que por sua vez inclui rotulagem de imagens, deteção de faces e pontos de referências, reconhecimento ótico de caracteres e marcação de conteúdo explicito seja facilmente integrado a uma aplicação.

No contexto da nossa aplicação este ira tratar de identificar os vários objetos em uma dada imagem de forma a realizar anotações acerca do nome e da localização do objeto na imagem.

4. Arquitetura do Sistema

Todas as operações de submissão e posterior consulta são disponibilizadas através de um servidor gRPC, o qual funciona como a fachada do sistema, isto é, a aplicação cliente não usa serviços da plataforma GCP. Para aumentar a disponibilidade e balanceamento de carga do sistema existem várias réplicas do servidor gRPC, cada uma a executar-se numa VM de um instance group.

As secções seguintes contêm os detalhes da implementação e esclarecimento de decisões tomadas em cada módulo do projeto, sendo divididas em 6 categorias: Contrato, gRPC Server, IpLookup Function, Client App, Detect Objects App e Monitor Function.

4.1. Contrato

O contrato define as funcionalidades do sistema que estarão disponíveis para as aplicações clientes de uma interface gRPC através de um arquivo do tipo .proto, para tal foram desenvolvidos 4 “métodos”, que tal como demonstra a Figura 2 possuem o prefixo rpc (Remote Procedure Call) onde auxiliara o compilador de protocol buffer a gerar código de interface de serviço. Estes métodos por sua vez tratam respetivamente por conectar-se a um servidor, submeter um ficheiro de imagem para deteção de objetos, obter um ficheiro submetido em conjunto com os nomes dos objetos encontrados e por fim obter uma lista de todos os ficheiros submetidos que estejam de acordo a uma série de requisitos propostos.

```
service CNImageService {

    // Tentativa de conexão ao servidor
    rpc isAlive(Void) returns (Text);

    // Submissão de um ficheiro imagem para deteção de objetos
    rpc SubmitImageDetectionObjectRequest(stream Img) returns (ReqId);

    // Retorno de um ficheiro imagem com a deteção e os nomes dos objetos
    rpc RequestImageDetectionObjectResult(ReqId) returns (ImgWithObjDetected);

    // Retorna os nomes dos ficheiros com o objeto referido
    rpc RequestFileNamesResult(FileReq) returns (FileResp);

}
```

Figura 2 - RPCs do arquivo proto

É de se notar que o contrato possui a especificidade da conotação “stream” que indica que o cliente ou o servidor poderão enviar múltiplos pedidos/respostas.

Para realizar estas operações forem desenvolvidos vários tipos de *message* que neste caso apenas possuem campos de tipos escalar como por exemplo string e int. As *messages* correspondem aos parâmetros recebidos e retornados, sendo, portanto, definidos os seguintes tipos:

- **Void:** Representa um tipo de dados sem conteúdo, ou seja, não possui campos

- **Text:** Representa o retorno do servidor ao cliente com a informação de sucesso ou insucesso da conexão na forma de um campo do tipo string.

- **ReqId:** Identificador representado por uma string, que é obtido pelo cliente após a realização do pedido de processamento da imagem e irá ser utilizado pelo mesmo com o intuito de receber os resultados deste processamento.

- **Img:** Representa uma estrutura com dois tipos de *message* incorporadas, nomeadamente Metadata, que possui o campo dado pelo nome da imagem submetida e outro correspondente ao tamanho da mesma e chunkOrMetadata que por sua vez representa o tipo de submissão realizada, ou seja, possui campos que podem ser do tipo Metadata de modo a dar a conhecer ao servidor a informação da imagem, ou do tipo bytes, que representa o conteúdo da imagem a ser guardado, este tipo de escolha é representado pelo prefixo *oneof*.
- **ImgWithObjDetected:** Representa o formato de retorno do resultado de pedido de submissão de uma imagem, onde constam os objetos presentes nessa imagem, representados pelos campos do nome, acompanhado do seu respetivo grau de certeza. A estes campos é aplicado o prefixo de *repeated* de forma a esclarecer que é uma lista de um dado tipo. Esta *message* possui ainda um campo que representa o URL da nova imagem para que o cliente possa acedê-la.

- **FileReq:** Identifica o pedido da lista de ficheiro de acordo com os parâmetros de pesquisa: data de início e fim, o nome do objeto e o grau de certeza mínimo com que seja feita a deteção.

- **FileResp:** Representa o retorno da lista com todos os URL dos ficheiros que estejam conforme a série de parâmetros de pesquisa do FileReq na forma de uma string.

4.2. gRPC Server

O servidor é responsável por implementar os métodos referenciados no contrato como “rpc”, sendo para isto necessário inicializar o acesso a serviços como a Firestore e o Cloud Storage. Estes serviços são inicializados através da variável de ambiente GOOGLE_APPLICATION_CREDENTIALS que contém a chave privada da conta de serviço do GCP. De forma a inicializar estes serviços são necessárias também outras informações, como o nome do bucket e o identificador do projeto respetivamente, que são inseridos a partir de macros com o valor definido por padrão.

O método “isAlive” trata apenas de criar um message do tipo Text para o cliente com a informação de que a conexão foi realizada com sucesso através da string “Server is alive” sendo esta posteriormente enviada para o cliente.

O método “SubmitImageDetectionObjectRequest” uma vez que recebe como parâmetro uma stream, é necessário que na sua invocação seja retornado um stream observer, que foi implementado de forma a interpretar cada mensagem enviada pelo cliente para esse mesmo observer. Esta implementação incorpora a inicialização de um documento na Firestore, de onde é obtido um identificador único, que será posteriormente utilizado não só para enviar ao cliente como identificador do pedido, como também para nomear o blob com a respetiva imagem guardada na cloud storage, evitando problemas no envio de ficheiros com o mesmo nome. O documento criado possui dois campos, nomeadamente o nome do bucket e do blob da imagem original.

No processo de criação do blob na cloud storage, é tido em conta que não é possível guardar diretamente ficheiros com um tamanho maior do que 1 Mbyte, assim sendo, nos casos em que isso acontece, é utilizado um “chunkingService” responsável por criar um canal para a storage, onde são escritos os vários bytes do ficheiro. Após a conclusão de ambos os processos anteriormente referidos, é publicada uma mensagem no tópico “detectionworkers” e informada a conclusão do método.

De forma a obter o resultado do processamento da imagem, foi implementado o método “RequestImageDetectionObjectResult” ao qual é passado como parâmetro o identificador do pedido, sendo este utilizado pelo servidor no acesso à Firestore, mais especificamente ao documento correspondente ao identificador. É criada a resposta do tipo ImgWithObjDetected, com os campos definidos no contrato, nomeadamente o link para o blob anotado e os objetos detetados com os seus respetivos graus de certeza.

Por fim, o método “RequestFileNamesResult” oferece ao cliente a possibilidade de obter uma lista com todos os links para os ficheiros anotados que se encontrem conforme os parâmetros de restrição, nomeadamente um intervalo de datas, o nome do objeto que pretende encontrar e o grau de certeza mínimo para esse mesmo objeto. Ao realizar as queries necessárias é construida uma lista que representa o FileResp composta pelos links para os ficheiros que passaram nos filtros sendo esta posteriormente enviada para o cliente.

4.3. IpLookup Function

Este módulo trata da obtenção dos endereços IP dos servidores gRPC, sendo o mesmo desenvolvido como uma Cloud Function, de forma a que posteriormente o cliente escolha um IP aleatório e, em caso de falha de ligação ao servidor gRPC através do IP escolhido, tente utilizar outro IP ou possua a oportunidade de repetir o processo de lookup para atualizar a lista e estabelecer uma nova ligação.

Para a concretização de tal, a classe Iplookup implementa *HttpFunction* que por sua vez possui um método default service, onde é possivel pedir e responder a pedidos HTTP. Através do *projectId* que é o identificador do projeto criado na Google Cloud Platform e da zona onde se encontra colocado o instance group, é possível obter a lista de todas as VMs que lhe pertencem. Caso as mesmas se encontrem em execução, ou seja, com o status “RUNNING”, é dado o prosseguimento a obtenção dos IPs através de um método da classe Instance denominado getNatIP, sendo este utilizado posteriormente para ser escrito no browser em um formato json em que a propriedade é sempre IpAddress e o valor desta corresponde ao IP obtido. De forma a hospedar esta função na cloud, foi executado o batch script demonstrado na Figura 3.

```
gcloud functions deploy Iplookup --allow-unauthenticated
--entry-point=lookupService.Iplookup --runtime=java11
--trigger-http --region=europe-west1 --source=target/deployment
--service-account=cn-2122-g03-allservices@cn2122-t1-g03.iam.gserviceaccount.com
```

Figura 3 - IpLookup Batch Script

4.4. Client App

A interface do cliente é constituída por um menu que lhe permite executar 3 funções: submeter uma imagem para deteção de objetos, obter os resultados dessa submissão e a obtenção de uma lista de ficheiro que estejam conforme uma série de parâmetros de pesquisa.

O cliente é ligado ao servidor a partir de uma classe ManagedChannel do tipo gRPC, onde através do IP e do porto do servidor consegue estabelecer um canal. Visto que existem operações onde é esperado que o servidor acabe a execução por completo e outras necessitam enviar vários pedidos como por exemplo os “chunks” das imagens que possuem uma dimensão elevada, são criados dois tipos de stub, nomeadamente o bloqueante que esperar a conclusão do servidor e o não bloqueante que é assíncrono.

De forma a garantir que o cliente conseguiu ter êxito na conexão ao servidor através do stub bloqueante é enviado uma message Text conforme definida pelo contrato onde o servidor irá escrever a mensagem de sucesso ou insucesso para o cliente.

Após todo este processo inicial a escolha da operação que o cliente deseja realizar é feita e para cada uma é ilustrado o seu processo.

4.4.1. Virtual Machines IP Address

As obtenções dos endereços IP das máquinas virtuais dão se a partir de um pedido HTTP do tipo GET ao URL da Cloud Function que hospeda a função IpLookup. Caso este tenha sucesso, ou seja, o status code seja 200, é criado um tipo Gson e uma Lista de JsonElement que vai ser preenchida através do body da resposta HTTP onde constam os vários IPs. A classe Gson realiza a serialização de uma lista de strings correspondente aos

vários IPs para o tipo JsonArray de forma a que seja possivel aceder aos vários elementos json e adicionar a lista previamente definida de JsonElement.

Após a obtenção de todos os elementos json, é utilizado uma função auxiliar que a partir de um intervalo entre 1 ao máximo de VMs conectadas no instance group que hospeda os servidores e gera um número aleatório que será utilizado para obter o valor da propriedade IpAddress do elemento json. Caso o status code seja do tipo 404 é retornado uma string com a informação de que o recurso não foi encontrado e por fim se este é do tipo 500 é retornado também uma string com a informação de que houve um erro por parte do servidor.

4.4.2. Submissão de uma Imagem

A função de submissão de imagem, necessita criar um ClientRequestStreamObserver que estende de StreamObserver onde o servidor irá escrever a sua resposta para o cliente, sendo este passado como callback na chamada do stub não bloqueante, que por sua vez retorna um StreamObserver onde o cliente envia os vários pedaços da imagem para o servidor.

O processo de obtenção do nome da imagem, o seu tamanho e envio da mesma é feito a partir da indicação do path onde se encontra, em que neste caso todas estão inseridas na pasta resources. De forma a passar esta informação para o servidor foi instanciado uma classe do tipo Metadata que conforme referido no capítulo 4.1 possui o nome da imagem e o seu respetivo tamanho e a classe do tipo Img que correspondente ao embrulho da informação do Metadata mais o tipo de envio a ser feito neste caso também Metadata.

O envio ao servidor é realizado ao chamadar o metodo onNext da stream passando lhe como argumento o valor retornado pela classe Img, para que posteriormente seja feita a leitura dos bytes do ficheiro em chunks, ou seja, um array de bytes de tamanho máximo correspondente a 1 Mbyte.

Para cada chunk lido do ficheiro é criado uma nova instância do tipo Img que faz set do tamanho do chunck e envia ao servidor até a leitura total da imagem seja concluída, no fim deste processo é enviado ao servidor a indicação que terminou o envio através da chamada a função onCompleted da stream.

4.4.3. Obtenção do resultado do processamento

A função que obtém o resultado do processamento da imagem requer um identificador do pedido de submissão feito anteriormente e retorna um ImgObjDetected que contém todas as informações acerca dos objetos detetados tal como referido anteriormente como o nome, o seu respetivo grau de certeza e o link da imagem anotada. Para tal o cliente introduz o identificador do pedido através da linha de comandos de forma a proceder a serialização desta string para o tipo definido pelo contrato ReqId.

É ilustrado na consola os valores dos campos referidos acima, contudo tendo em conta que a mesma retorna uma lista é necessário percorrer sobre a mesma de forma a obter o nome e o grau de certeza para o respetivo objeto.

4.4.4. Obtenção do resultado do processamento

Por último a função que retorna a lista de objetos encontrados conforme uma serie de parâmetros de pesquisa, dá-se de forma a que o cliente introduza na consola datas conforme o formato DD/MM/YYYY tanto para a de início como a de fim, o nome do objeto que deseja encontrar e o grau de certeza mínimo com que deseja que seja obtido. Ambas as datas sofrem um processo de verificação onde é comparado a escrita do cliente com o formato pedido, sendo que para cada vez que o mesmo escreve de forma errada, é feito um novo pedido de inserção da data. A partir de todos os campos preenchidos é criado o tipo definido pelo contrato FileReq de forma a ser passado como parâmetro a chamada da função do stub bloqueante requestFileNamesResult que por sua vez retorna o tipo FileResp com todas as informações desejadas caso seja satisfeito os critérios de pesquisa. É de se notar que neste caso o stub é bloqueante pois é somente feito um pedido com todas os critérios e o servidor somente responde no fim de executar todas as querys e encontrar ou não os documentos que satisfaçam as exigências.

4.5. Detect Objects App (Worker)

Este módulo é responsável por realizar o processamento das imagens submetidas por um cliente, guardando a informação na Firestore, sendo esta composta pelo URL que

corresponde à imagem anotada dentro do blob, a data em que esta mesma imagem foi processada, tal como todos os objetos encontrados com o seu respetivo grau de certeza.

O primeiro passo consiste em aceder ao bucket onde é realizado o armazenamento da imagem que o cliente deseja operar, e onde será inserida a imagem anotada com os objetos detetados, a designação desse bucket é “cn-2122-g03-trabalho_final”. É também colocado o blob da imagem anotada com acesso público.

O passo seguinte foi aceder à mensagem recebida a partir do tópico denominado “detectionworkers” no serviço de Pub/Sub, onde o worker se encontra subscrito a partir da subscrição “detectionworkers-sub”.

O processo de publicação de uma mensagem passa pelo acesso aos atributos da mensagem nomeadamente o bucket e o blot da imagem original, juntamente com o acesso ao documento da Firestore onde serão escritas as novas informações para os processamento da imagem e para tal é utilizado a Vision API descrita conforme a secção 4.5.1.

4.5.1. Vision API

A Vision API utiliza classes java que possuem o intuito de identificar e anotar objetos localizados, para tal disponibiliza um conjunto de classes e métodos apresentados abaixo:

- Classe ImageSource: Através do path fornecido como parâmetro obtém a referência para a imagem hospedada na cloud.
- Classe AnnotateImageRequest: Representa uma solicitação para executar tarefas que recebe o tipo de deteção e a imagem a ser processada, em que neste caso os recursos são do tipo OBJECT_LOCALIZATION.
- Classe BatchAnnotateImageRequest: Representa um agrupamento de solicitações de anotações de imagem em uma única chamada de serviço, que neste caso irá receber um single request, o retorno de AnnotateImageRequest.
- Classe ImageAnnotatorCliente: Representa o serviço que realiza a entidade de deteção de imagens de facto.
- Classe BatchAnnotateImagesResponse: Adquire do retorno da classe ImageAnnotatorCliente a solicitação de deteção da imagem.

- Classe AnnotateImageResponse: Adquire através da classe BatchAnnotateImagesResponse a resposta, ou seja, a imagem com os objetos detetados. Visto que esta pode receber vários pedidos a boa pratica é embrulhá-la numa lista em que caso esta seja vazia significa que não existe nenhum objeto detetado.
- Classe LocalizedObjectAnnotation: Representa as várias anotações feitas na imagem com caixas delimitadoras, que por sua vez disponibiliza métodos que indicam o nome do objeto, o grau de certeza e os vértices no eixo horizontal e vertical de onde se encontram o objeto.

A função annotateWithObject é a responsável por cada objeto detetado realizar uma customização gráfica na imagem da sua deteção nomeadamente em estilo 2D, em que é definido um polígono do tipo delimitador para a anotação da imagem detetada representado pela classe BoudingPoly. O nome do objeto é escrito a partir dos seus vértices sendo estes traçados do início do mesmo até ao fim do eixo horizontal onde este se encontra.

A classe NormalizedVertex representa um ponto 2D na imagem com as coordenadas dos vértices relativas a imagem original variando entre 0 e 1, em que a partir do tamanho da imagem e dos vértices da imagem original é possivel traçar os 4 pontos pertencentes a imagem.

Todas estas classes e funções foram chamadas no escopo da função detectLocalizedObjectGcs com o objetivo final de retornar uma lista de ObjectInfo, sendo esta a classe que contem o nome e o grau de certeza do objeto. É de se notar que ao realizar a escrita da imagem no bucket esta possui o prefixo “annotated-“ e também que possui uma ACL com a role READER que permite realizar o acesso publico ao URL da imagem.

No final desta etapa a lista é processada de forma a ser escrita no Firestore nomeadamente o link com a imagem anotada, a data de processamento, as informações iniciais e a lista que contem o nome e o grau de certeza dos objetos. De forma a que futuramente possa se realizar querys sobre os campos existentes no documento, foi decidido que a lista dos objetos seria armazenada numa nova coleção dentro do documento existente que por sua vez cria vários documentos com um identificador aleatório com os campos do nome e do grau de certeza.

4.6. Monitor Function

Esta função tem como objetivo o redimensionamento do número de instâncias de um grupo responsável pelos workers. Apesar de ser também desenvolvida como uma Cloud Function, esta não utiliza HTTP, uma vez que deverá ser invocada (*triggered*) pelo aparecimento de uma nova mensagem no tópico onde o servidor escreve as mensagens e para tal foi implementada a interface *BackgroundFunction*, que possui o método *accept*.

De forma a realizar este redimensionamento, é acedido a um documento da Firestore composto por um contador de pedidos e um tempo de referência. O contador de pedidos é incrementado no início da função sendo no final da execução desta atualizando o valor no documento da Firestore, ou seja, 0 no caso em que o tempo do pedido ultrapassa-se em 60 segundos o tempo de referência, sendo estes obtidos pelo método *currentTimeMillis* do próprio sistema. Estes valores são utilizados para realizar os cálculos do ritmo de pedidos dado pela seguinte fórmula:

$$Ritmo = \frac{Nº\ pedidos}{(Treq - Tref)}$$

Através destes cálculos é feita a verificação se o resultado obtido ultrapassa os limites do treshold do valor de referência neste caso 0.05, sendo o valor mínimo de 0.03 e máximo de 0.07. Caso isto aconteça é respetivamente decrementado ou incrementado o número de instâncias ligadas tendo sempre em conta a condição de que deverá existir no mínimo uma e no máximo quatro instâncias em execução. De forma a hospedar esta função na cloud, foi executado o batch script demonstado na Figura 4.

```
gcloud functions deploy Monitor --allow-unauthenticated
--entry-point=monitorService.DetectObjectsMonitorFunction
--runtime=java11 --trigger-topic detectionworkers
--region=europe-west1 --source=target/deployment
--service-account=cn-2122-g03-allservices@cn2122-t1-g03.iam.gserviceaccount.com
```

Figura 4 - Monitor Function Batch Script

5. Formato de dados

Neste capítulo é demonstrado os formatos dos dados utilizados na comunicação entre todos os componentes do sistema, mais especificamente as mensagens do serviço Pub/Sub e dos documentos criados na Firestore tanto para monitorização do ritmo de pedidos por parte dos clientes, como também para guardar a informação desses mesmos pedidos.

5.1. Mensagens Pub/Sub

As mensagens publicadas no serviço Pub/Sub tal como demonstrar a Figura 5 são compostas pelo corpo da mensagem que refere o identificador do pedido, que será utilizado para criar o novo blob, e por dois atributos, nomeadamente o nome do bucket e do blob original. Estes atributos são necessários para o subscritor que realiza a deteção da imagem conhecer a localização da imagem original que terá de aceder para realizar o respetivo processamento de deteção.

Figura 5 - Formato da mensagem Pub/Sub

5.2. Documento da Monitor Function

O documento da Firestore utilizado pela monitor function encontra-se incluído na coleção “SubmitedVMRequests” e é composto por apenas dois campos tal como indica a Figura 6. Estes campos são: o counter, que representa o número de pedidos, e o Tref, que indica o valor atual do tempo de referência a ter em conta pela função.

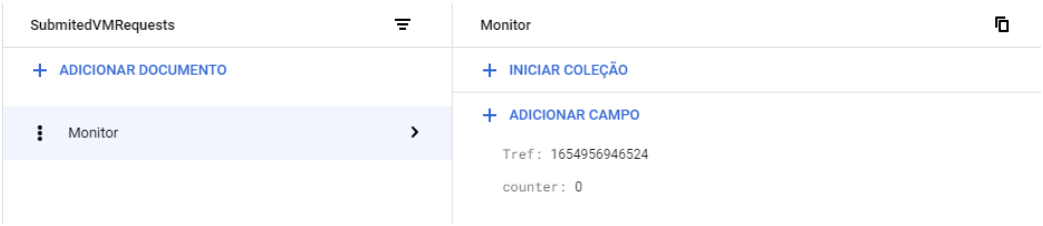

Figura 6 - Formato do documento de monitorização

5.3. Documentos de deteção de objetos

Uma vez que uma única query não permite pesquisar desigualdades em dois campos diferentes do documento, e como também não é possível pesquisar dentro de listas de tipos complexos, foi adotado o formato representado na Figura 7. Este documento é composto pelo link para a imagem anotada, acompanhado da data em que ocorreu o seu processamento, os dados da imagem original, nomeadamente os nomes do blob e do bucket em que está inserida, e por fim uma outra coleção denominada “objects”.

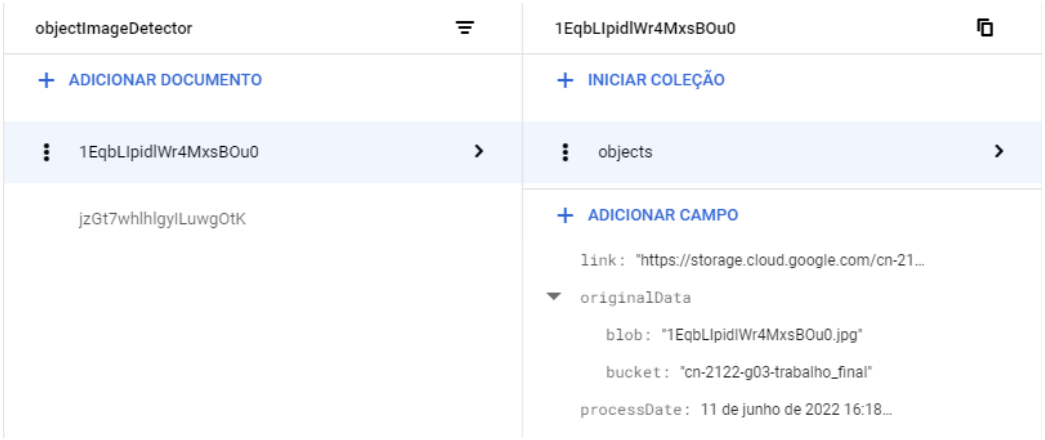

Figura 7 - Formato da coleção de deteção de imagens

Dentro da coleção “objects”, tal como é possível observar na Figura 8 encontram-se vários documentos, onde cada um corresponde a um objeto detetado na imagem e é composto por dois campos, o nome do respetivo objeto e o grau de certeza atribuido ao mesmo.

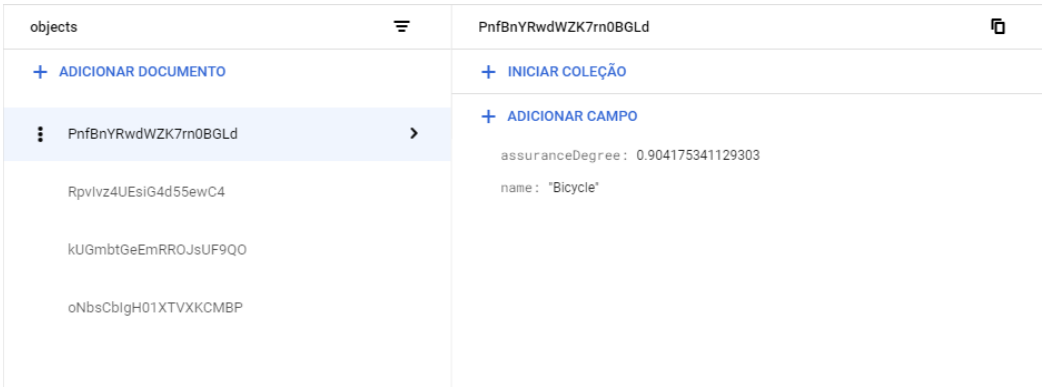

Figura 8 - Formato da coleção "objects"

6. Conclusão

No planeamento deste trabalho prevê-se a existência de conectividade entre três componentes principais, sendo eles o cliente, o servidor e o subscritor ao serviço Pub/Sub que em conjunto realizam o pedido e o processamento de ficheiros de imagem, detetando e anotando os objetos presentes nos mesmos. Além de duas funções hospedadas na cloud para realizar o IpLookup dos servidores e a monitorização do ritmo de chamadas.

O principal objetivo deste trabalho foi cumprido, uma vez que é possível a comunicação entre os três componentes que realizam com sucesso a submissão e execução das tarefas pretendidas. Contudo, não foi possível alcançar o objetivo de redimensionar o número de instâncias a partir da função de monitorização, nem colocar os componentes correspondentes ao cliente e ao subscritor a correr hospedados numa instância da cloud.

Em anexo, encontra-se o trabalho em JAVA, que condiz com toda a explicação relatada ao longo do relatório.

7. Bibliografia

Simão, J., Materiais de Apoio da UC – CN, Ano Letivo 2021-2022, 2ºSemestre