ADDETC – Área Departamental de Engenharia Eletrónica e Telecomunicações e de Computadores

LEIM -Licenciatura Engenharia informática e multimédia

Computação na Nuvem

Trabalho final

Turma:

LEIM-62D

Trabalho realizado por:

Diogo Cadavez Nº43408

Pedro Ferreira Nº43747

Miguel Távora Nº45102

Docente:

Luís Assunção

Data: 18/06/2021

Índice

Índice ilustrações

1. Introdução

O projeto final de Computação na Nuvem tem como objetivo principal desenvolver um sistema a funcionar na Cloud capaz de detetar *labels* em ficheiros de imagem e obter a tradução das *labels* de inglês para português.

Este sistema funciona todo ele dentro do serviço de Cloud da Google nomeadamente o GCP. Dentro do serviço de Cloud serão utilizados os serviços: Cloud Storage, Firestore, Pub/Sub, Compute Engine e Cloud Functions.

- Cloud Storage – armazenamento das imagens a processar.
- Firestore – guarda informações relevantes sobre o processamento de um ficheiro.
- Pub/Sub – troca mensagens desacopladas entre componentes do sistema.
- Compute Engine – alojamento de máquinas virtuais onde serão guardados os servidores gRPC e servidores de obtenção das *labels*.
- Cloud Functions – obtenção dos endereços IP dos servidores gRPC (*trigger* HTTP) e para tradução de texto de inglês para português (*trigger* Pub/Sub)

Este sistema possui requesitos de elasticidade nomeadamente na recepção e processamento das imagens, visto que ambos os servidores se encontram dentro de um *instance-group* diferente. Um *instance-group* é nomeadamente um grupo de servidores todos iguais entre si onde é possível aumentar e diminuir o número de servidores, neste caso servidores gRPC e de obtenção de labels. Este aumento e diminuição tem de ser feito por um administrador da aplicação, visto que o próprio servidor não consegue aumentar a quantidade de servidores existentes. O aumento automático de servidores é possível dentro de um *intance-group* se este possuir *auto-scaling*. *Auto-scaling* é uma propiedade atribuida ao *instance-group* para aumentar e diminuir automaticamente o número de instâncias dos servidores conforme a quantidade de utilização. Contudo os *instance-group* não possuem essa propriedade neste sistema.

2. Desenvolvimento

2.1 Cliente

O cliente é a parte do sistema que corre na máquina do cliente. O utilizador ao interagir com o sistema são feitos pedidos aos diversos componentes do sistema, nomeadamente o servidor gRPC e também uma Cloud Function (HTTP).

Inicialmente o cliente interage com uma Cloud Function HTTP. A interação com a Cloud Function tem o propósito da obtenção de endereços IP dos servidores disponíveis para realizar as diversas operações do sistema. Para a obtenção dos endereços o cliente escreve qual é o nome do *instance-group* e a partir disso obtem os endereços existentes. Existem também um nome de um grupo por omissão que pode ser selecionada caso o utilizador não souber qual é o nome a que se deve ligar. Após introduzido o nome do grupo a aplicação obtêm todos os endereços IP e escolhe um aletóriamente.

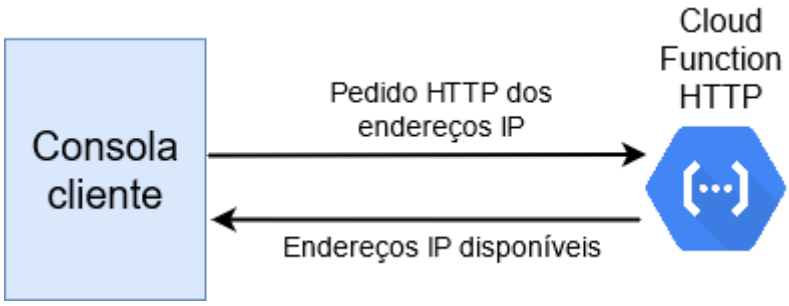

Figura 1 - Interação entre o cliente e o Cloud Function

Após a obtenção de um endereço IP é estabelecida a ligação com o servidor gRPC. A interação com os servidores gRPC serve para realizar as diversas operações de pedido-resposta disponíveis pelo contrato entre o cliente e o servidor. O cliente faz uma pergunta ao qual o servidor devolve uma resposta.

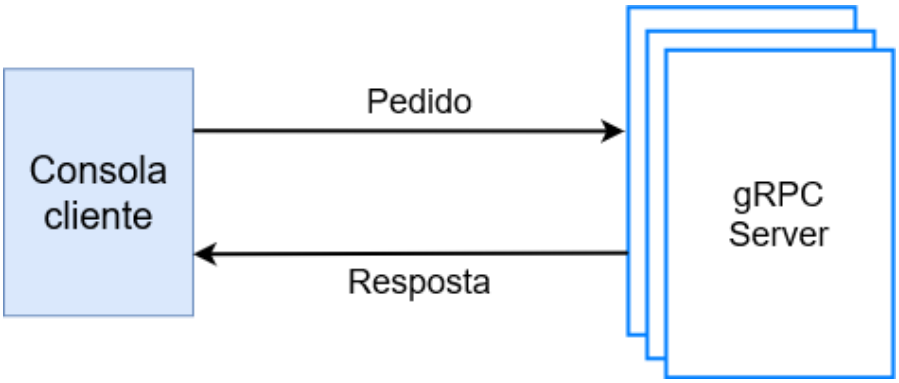

Figura 2 - Intereção entre o cliente e o servidor gRPC

Em termos de tolerância a falhas o cliente o cliente sempre que se conecta a um servidor com um determinado endereço IP envia uma mensagem a verificar se existe conexão com o mesmo, caso não receba resposta repete o processo de obtenção do endereço IP e testa a conexão.

Se o envio de uma mensagem der erro, por exemplo por falha por parte do servidor, o cliente volta a obter os endereços IP disponíveis e escolhe um aleatóriamente. Após a obtenção do novo endereço este realiza a operação que falhou para o novo servidor e passa a comunicar sempre com o servidor novo.

O cliente assume que a *Cloud Function* HTTP está a correr e que existe pelo menos um servidor gRPC a funcionar.

2.2 Cloud Function HTTP

A *cloud function* HTTP é uma parte do sistema que permite ao cliente obter os endereços IP do sistema a funcionar no momento em que o cliente se liga ao sistema. Para realizar esta funcionalidade as *cloud functions* possuem um URL ao qual o cliente utiliza para fazer pedidos e obter respostas. Para realizar esta funcionalidade a *cloud function* possui acesso ao recurso Compute Engine do qual é possível saber quais são as máquinas a correr e quais são os seus endereços IP.

Após a obtenção dos endereços através do Compute Engine, estes são enviados para o cliente no corpo da mensagem e caso tenha sido enviado com sucesso é enviado a mensagem com o código 200 OK. O corpo da mensagem é constituído por os endereços IP separados por ;, por exemplo: 200.10.10.2;200.15.10.4;.

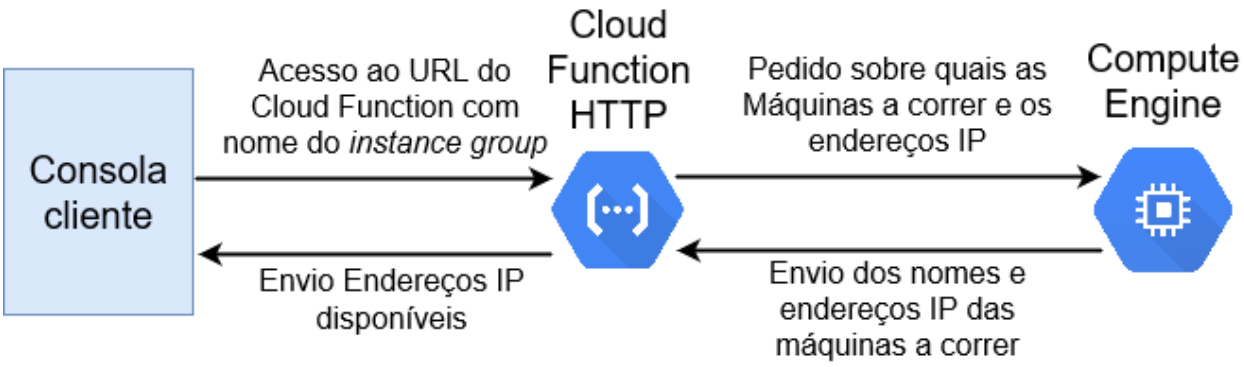

Figura 3 - Interação entre a Cloud Function HTTP o cliente e o Compute Engine

Em termos de tolerância a falhas caso a listagem das VMs dê erro este devolve uma mensagem de erro com o número 503, Service Unavailable. Caso não seja possível escrever o conteúdo no corpo da mensagem é enviado para o cliente o código 500, Interval Server Error.

Este serviço para ser disponibilizado é necessário ter instalado o Google Cloud SDK e ter uma conta serviço associada ao mesmo. De seguida definir na linha comandos os parâmetros necessários e desta forma a *cloud function* fica a funcionar na plataforma de *cloud functions* da Google.

2.3 Servidor gRPC

O servidor gRPC é a componente responsável por receber pedidos e devolver respostas ao cliente, guardar os ficheiros enviados pelo cliente no Storage, publicar mensagens num tópico do Pub/Sub e aceder a informações do Firestore. Este servidor corre num *instance group*, onde o serviço é multiplicado por diversos servidores de forma a obter maior disponibilidade.

Para a interação entre cliente e servidor existe um contrato implementado pelo servidor e que o cliente chama para realizar pedidos.

```
service ContractService {
  rpc sendFileBlocks(stream Content) returns(Identifier);
  rpc getCharacteristics(Identifier) returns(PortugueseLabels);
  rpc getFilesWithCharacteristics(Characteristics) returns(stream Identifier);
  rpc getContentStored(Name) returns (stream Content);
  rpc getAllFilesFromStorage(Void) returns (stream Name);
  rpc getAllLabelNames(Void) returns (PortugueseLabels);
  rpc checkServerConnection(Void) returns(Void);
}
```

Figura 4 - Contrato implementado entre o cliente e o servidor

O contrato possui no total 8 operações sendo elas:

- Operação: sendFileBlocks - envio de uma imagem dividido em blocos do cliente para o servidor, onde o servidor escreve no Storage o conteúdo da imagem recebida e devolve o identificador da imagem. Para evitar sobreposições é escolhido um bucket aleatóriamente, caso já exista o nome definido pelo utilizador é adicionado um número no final do nome evitando sobreposição tanto dos dados no Storage como no Firestore.
- Operação: getCharacteristics – o cliente envia um identificador da imagem e o servidor devolve uma mensagem com as características e a tradução das características do respetivo identificador.

- Operação: getFilesWithCharacteristics – é enviado do lado do cliente uma data inicial uma data final e uma característica ao qual o servidor devolve os respetivos identificadores das imagens que estejam entre as datas e que possua a determinada característica.
- Operação: getContentStored - o cliente envia o nome do bucket e do blob e recebe o conteúdo da imagem, fazendo assim o *download* da imagem numa diretoria á sua escolha.
- Operação: getAllFilesFromBucket - o cliente envia uma mensagem vazia e o servidor devolve todos os nomes dos blobs existentes em todos os buckets. Desta forma o cliente pode selecionar qual é a imagem que pretende realizar operações sobre.
- Operação: getAllLabelNames – é feito o envio por parte do cliente de uma mensagem vazia onde recebe todas as *labels* existentes no Firestore, serve para poder selecionar uma característica para receber os identificadores entre as duas datas e com uma característica.
- Operação: checkServerConnection – o cliente envia uma mensagem do cliente para o servidor somente para testar a conexão entre ambos.

O servidor gRPC interage com o Cloud Storage para armazenar o conteúdo das imagens enviadas pelo cliente em formato de *blobs*. Este armazenamento é feito através de um WriteChannel onde conforme vai recebendo o conteúdo da imagem vai escrevendo no Storage e no final fecha a conexão, ficando assim guardado e também é posta visibilidade pública na imagem. O servidor também consulta os *blobs*, nomeadamente para obter meta-data como o tamanho em bytes da imagem, data de criação, entre outros. Um outro acesso é quando o cliente quer fazer *download* da imagem, o servidor envia em blocos a imagem e o cliente quando recebe tudo escreve o seu conteúdo no disco.

O servidor interage também com o serviço Pub/Sub onde publica uma mensagem no tópico *topicworkers*. Esta mensagem é enviada sempre que é pretendido que seja feito a classificação e tradução das classificações de uma imagem, ou seja quando é chamado o método sendFileBlocks do servidor. Esta mensagem possui o formato:

HashMap: (Keys : Values) -> String : String

- bucket : Nome do bucket
- file : Nome do ficheiro
- type : extensão do ficheiro
- size : tamanho do ficheiro em bytes
- time : tempo que foi criada em formato Date do Java

Esta mensagem será posteriormente lida e processada pelo servidor Labels App.

Por fim o servidor interage com o Firestore para ir buscar as informações relevantes do processamento do sistema nomeadamente: *labels* detetadas, tradução, data processamento, data criação, tipo conteúdo armazenado, tamanho do ficheiro em bytes e Id da mensagem do tópico no Pub/Sub. Apesar de ter todos estes acessos o servidor somente envia para o cliente as *labels* obtidas e as respetivas traduções.

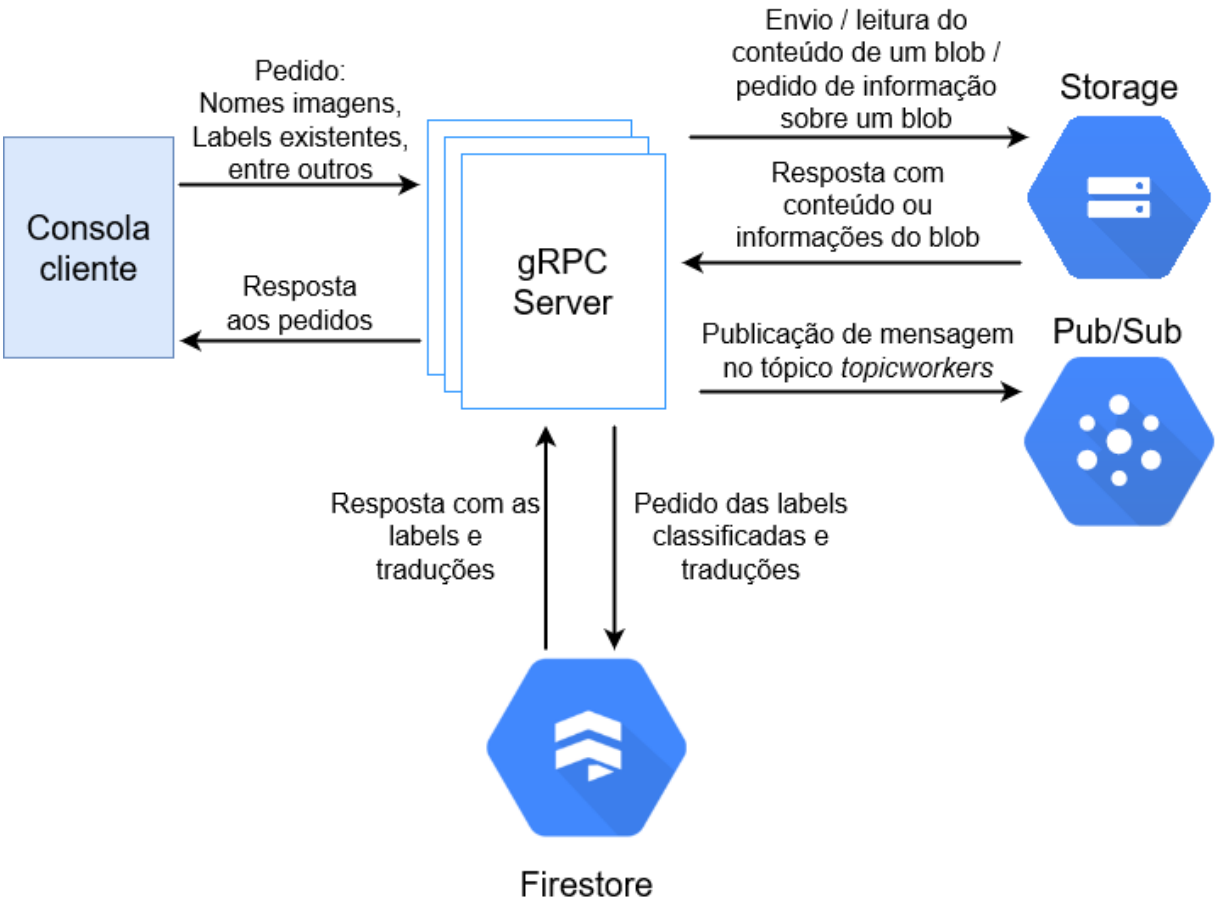

Figura 5 - Interação entre o Servidor gRPC o cliente, o Storage, o Pub/Sub e o Firestore

Em termos de tolerância a falhas na publicação da mensagem no tópico *topicworkers* do Pub/Sub caso dê erro repete até 3 vezes o envio da mensagem, caso passe as 3 vezes a mensagem não é publicada.

O servidor assume que já existe o tópico *topicworkers* e uma subscrição no tópico a quando da publicação da mensagem no tópico, visto que é criado o tópico e a subscrição quando o servidor Labels App inicia.

2.4 Servidor Labels App

O servidor Labels App é um servidor que quando existe uma mensagem no tópico *topicworkers* do Pub/Sub este faz *pull* da mensagem. Sempre que é recebida uma mensagem é criado um Subscriber, que funciona como tarefa. Esta tarefa a partir dos campos do nome do *bucket* e do *blob* é possível ir buscar através de um URL o *blob* da imagem, quando o *blob* é público. A partir da imagem e de uma dependência no pom.xml para a Vision API é assim utilizada a Vision API da Google para detetar as *labels* na imagem. Após processadas as *labels* é enviada uma mensagem para o tópico *translation* do Pub/Sub, a mensagem possui o seguinte formato:

- String data : bucket;blob
- HashMap: (Keys : Values) -> String : String
  - messageId : Id da mensagem publicada no topico *topicworkers*
  - type : tipo de conteudo
  - size : tamanho em bytes fo ficheiro
  - time : tempo de criação do blob
  - processedTime : tempo em que foram processadas as labels
  - (diversos valores de i) labeli – label obtida pela classificação da Vision API

Este servidor é um servidor desacuplado do servidor gRPC para permitir uma maior disponibilidade do sistema. Libertando mais o servidor gRPC para atender outros pedidos. Caso só existisse um servidor (servidor gRPC) este seria mais subcarregado, visto que teria de fazer um maior processamento nomeadamente das *labels*. O maior subcarregamento resultaria numa maior espera por parte do cliente e com muitos clientes simultaneamente poderia resultar numa quebra do sistema por falta de memória.

Caso não exista o tópico *topicworkers* é criado o tópico e uma subscrição assim que este servidor inicia. Na eventualidade de não conseguir criar a subscrição este tenta até 3 vezes, se ao fim das 3 tentativas não obtiver sucesso é gerada uma exceção e o programa termina a sua execução.

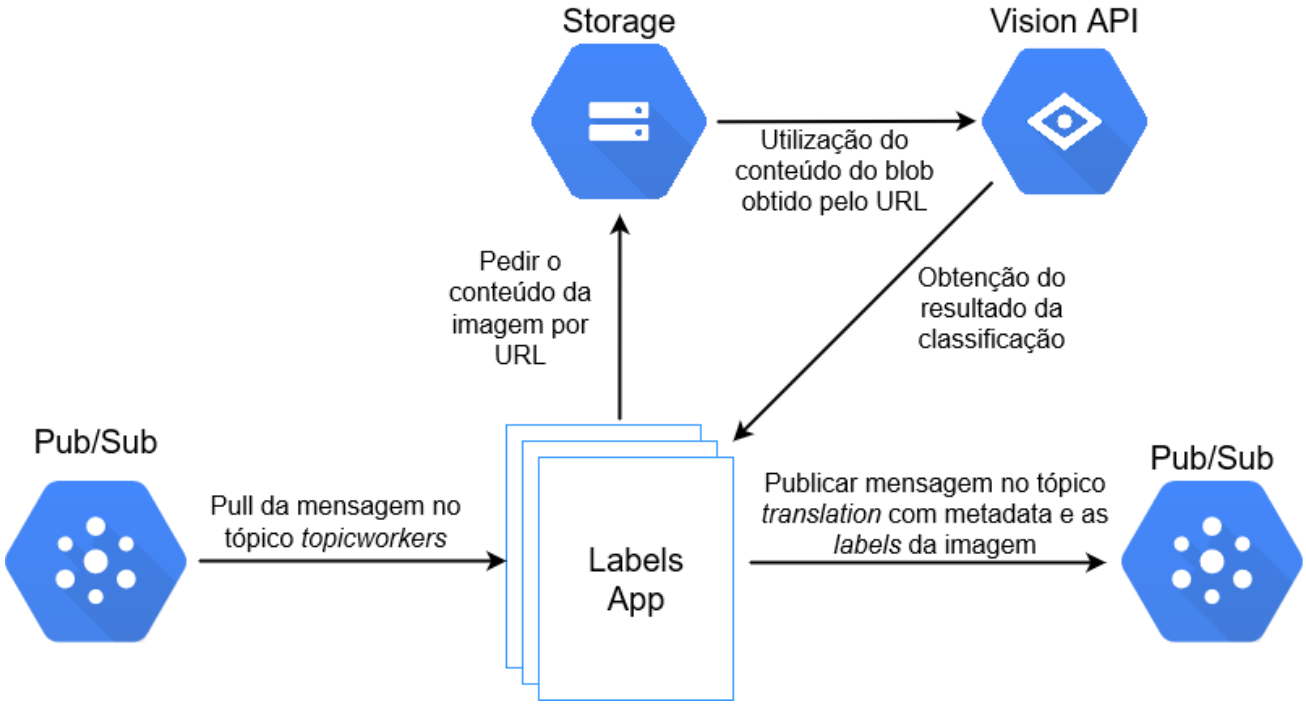

Figura 6 - Interação entre o servidor Labels App, Pub/Sub, Storage e Vision API

Na receção de uma mensagem no Pub/Sub do tópico *topicworkers* só é dado *acknowledge* na mensagem caso tenha dado sucesso na publicação da mensagem no tópico *translation*, interpretado pela Cloud Function do Pub/Sub. Desta forma mesmo que um servidor Labels App falhe a meio do processamento de uma imagem como não é feito o *acknowledge* da mensagem outro servidor Labels App a correr pode realizar o trabalho que não feito pelo servidor que falhou.

Este servidor assume o prossoposto que quando for publicar a mensagem já existe o tópico *translation* e uma subscrição para o mesmo criada pela *Cloud Function* do Pub/Sub.

2.5 Cloud Function (Pub/Sub)

A *Cloud Function* do Pub/Sub é chamado sempre que é submetida uma mensagem no serviço Pub/Sub associado ao tópico *translation* da *Cloud Function*. Quando a mensagem é recebida a *Cloud Function* a correr na Google interpreta a mensagem e a partir dos campos da mensagem das *labels* interage com a Translation API para traduzir as *labels* de inglês para português. Após traduzidas todas as mensagens é criado uma estrutura de dados para enviar os dados para o Firestore. Estes dados possuem a seguinte estrutura:

- HashMap: (Keys : Values) -> String : Object
  - messageId : Id da mensagem publicada no tópico *topicworkers* (String)
  - type : tipo do conteúdo (String)
  - size : tamanho em bytes do ficheiro (String)
  - time : tempo da data criação do blob (String)
  - processedTime : tempo em formato date do Java em que a imagem foi processada (String)
  - labels : labels obtida no processamento da imagem (ArrayList <String>)
  - translations : traduções das labels obtidas (ArrayList <String>)

Para executar esta Cloud Function é igual á Cloud Function HTTP mas é necessário definir qual é o nome do tópico na altura da criação. A Cloud Function do Pub/Sub no momento de criação caso não exista o tópico definido é criado o tópico e uma subscrição para o tópico automaticamente.

Em termos de falhas este sistema caso não consiga criar corretamente a instância do Firestore, o programa gera uma exceção e termina a sua execução. Caso dê exceção na obtenção das traduções a execução do programa termina e não é publicada a informação.

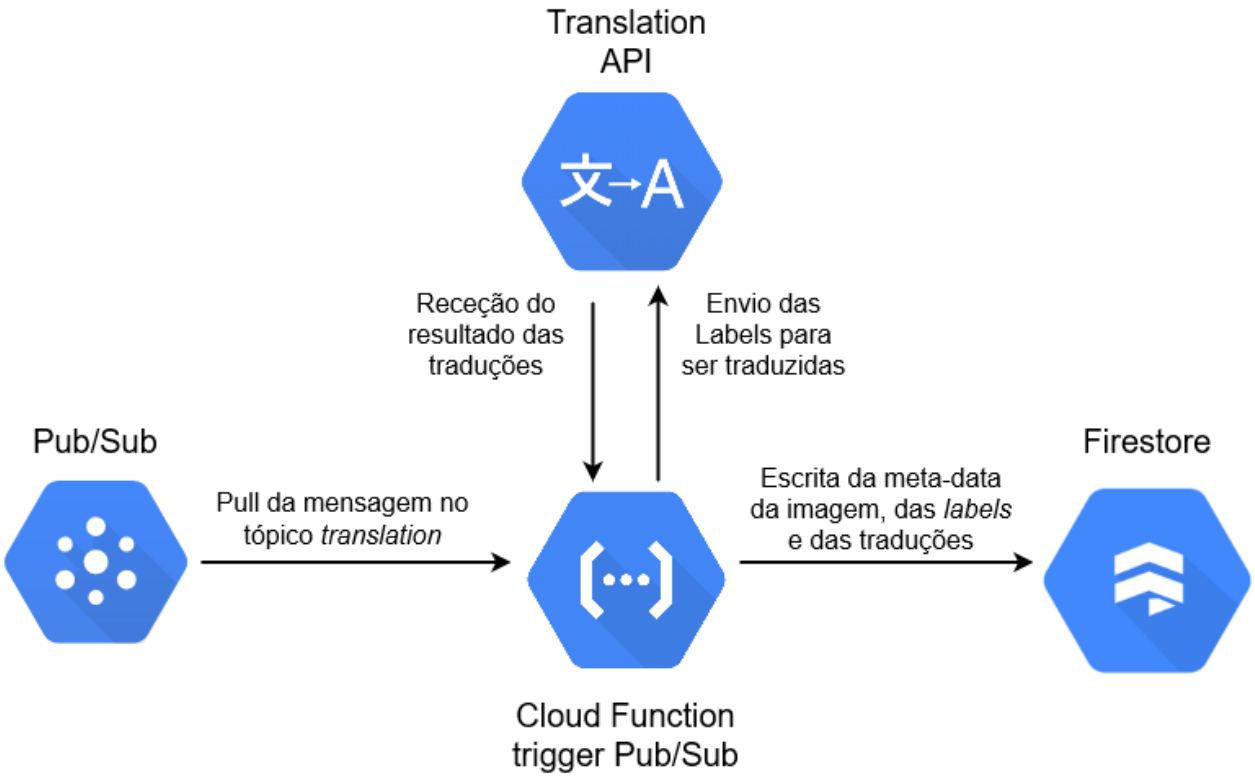

Figura 7 – Interação entre a Cloud Function triger Pub/Sub, Translation API e o Firestore

3. Conclusões

Em suma com a realização do presente trabalho prático o grupo adquiriu os seguintes conhecimentos:

- Utilizar o Google RPC para fazer troca de mensagens entre um cliente e um servidor, onde através da implementação de um contrato feito por um ficheiro proto é possível criar uma abstração ao nível da aplicação facilitando toda a implementação do sistema de troca de mensagens.
- Utilização do Storage para guardar ficheiros de grande dimensão de forma simples.
- Utilização de uma base dados No-SQL designada Firestore que permite ter vários campos de diferentes tipos, não estando restritos a uma estrutura fixa de uma base dados relacional.
- Utilização do Pub/Sub para envio e receção desacoplada de diversas partes do sistema.
- Utilizar o Compute Engine para criar máquinas virtuais para publicar os servidores ou outros serviços através de um endereço IP público. Quais as vantagens da existência de mais servidores e do aumento da disponibilidade e também as desvantagens da inconsistência da informação.
- Utilização de Cloud Function que permitem realizar operações como servidores sem a necessidade de uma *Virtual Machine* e o facto de serem escaláveis conforme a utilização.
- Utilização de APIs da Google como Vision API e Translation para realizar operações de interesse para o utilizador.

Em termos de implementação do sistema todo ele funciona e é possível utilizar este sistema sem possuir qualquer informação na base dados. Em termos de pontos de falha este sistema está robusto para quando falham algumas partes este consegue recuperar da falha, por exemplo falha de um servidor o cliente conecta-se a outro servidor, a repetição do envio de mensagens no caso de falhar o envio ou só dar *acknowledge* no fim de todo o processamento estar correto.

4. Bibliografia

Slides gRPC, google Storage, No-SQL, Pub/Sub e Compute Engine, Serverless Computing.

https://cloud.google.com/storage/docs/viewing-editing-metadata#code-samples_1