Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores

Computação na Nuvem - Trabalho Final

47202 : Manuel Maria Penha Gonçalves Henriques (47202@alunos.isel.pt)

47230 : Miguel Assuda Neves (47230@alunos.isel.pt)

47198 : Ricardo Filipe Matos Rocha (47198@alunos.isel.pt)

Relatório para a Unidade Curricular de Computação na Nuvem
da Licenciatura em Engenharia Informática e de Computadores

Professor : Doutor José Simão

04 – 06 – 2023

Índice

Lista de Figuras

Listagens

1

Introdução

Este trabalho vem pôr em prática os conhecimentos adquiridos ao longo da Unidade Curricular de Computação na Nuvem, nomeadamente a criação e implementação de contratos Protobuf, os serviços da Google Cloud Platform, como o Cloud Storage, Firestore, Pub/Sub, Compute Engine e Cloud Functions. Adicionalmente, demos uso à Vision API para realização exclusiva do mesmo.

1.1 Objetivos

Neste trabalho pretende-se desenvolver um sistema para submissão e execução de tarefas de computação na nuvem, com capacidade de adaptação a variações de carga, ou seja, elasticidade, pondo em prática os conhecimentos acima mencionados.

1.1.1 Funcionamento

O objetivo do trabalho centra-se na deteção de pontos de interesse (monumentos, parques, locais de entretenimento, etc.) numa determinada imagem em formato .png, e consequentemente determinar a localização do mesmo, bem como criar um mapa estático do local. Associado a estes resultados, existe um valor de pontuação que define um grau de certeza na deteção (entre 0 e 1).

Deste modo, o sistema desenvolvido deverá conseguir:

- Deixar um utilizador submeter qualquer tipo de foto no formato correto, sendo atribuído um identificador único para cada submissão;
- Identificar os pontos de interesse presentes no mesmo;
- Para cada ponto de interesse identificado corresponde um grau de certeza, um mapa estático e um par de coordenadas.
- Deixar um utilizador adquirir as informações da identificação, bem como o mapa estático através do identificador;
- Deixar um utilizador fazer uma pesquisa pelos pontos de interesse submetidos através duma filtragem de resultados com base num grau de certeza determinado pelo mesmo.

1.2 Conceitos Importantes

1.2.1 Elasticidade

Este conceito no mundo da informática é a capacidade de aumentar ou diminuir os recursos informáticos de processamento, memória e armazenamento rapidamente, para responder às mudanças de necessidades sem se preocupar com o planeamento da capacidade e a engenharia para picos de utilização. A elasticidade é geralmente controlada por ferramentas de monitorização de sistemas, o que possibilita alocar o número de recursos alocados conforme a necessidade real, sem causar danos às operações. Com a elasticidade na nuvem, as companhias não têm que pagar por capacidades não utilizadas ou por recursos inativos e não precisam se preocupar em investir na aquisição nem na manutenção de recursos e equipamentos adicionais.

1.2.2 Vision API

O Cloud Vision API é um serviço de processamento de imagens na nuvem disponibilizado pela Google Cloud Platform, que oferece modelos de machine learning pré-treinados por meio de API's REST e RPC. Através do mesmo é possível associar rótulos a imagens e classificá-las em diversas categorias predefinidas. Desde detetar objetos, ler textos (manuscritos ou impressos), identificar pontos de interesse (caso implementado no trabalho) e criar metadados relacionados com as imagens submetidas, existem várias implementações possíveis para este serviço.

1.2.3 Instances Groups

Um instance group é uma coleção de instâncias de máquinas virtuais que podem ser regidos como uma única entidade. São oferecidos dois tipos de grupos:

- Os gerenciados, que permitem que 'apps' operem em várias máquinas virtuais idênticas. As suas cargas de trabalho podem ser escaláveis e pode haver implementação regional e atualização automática;
- Os não gerenciados, que permitem o balanceamento de cargas de trabalho de máquinas virtuais pelo utilizador.

2

Arquitetura

A seguir, apresenta-se o esquema em blocos fornecido no enunciado que representa o trabalho pretendido.

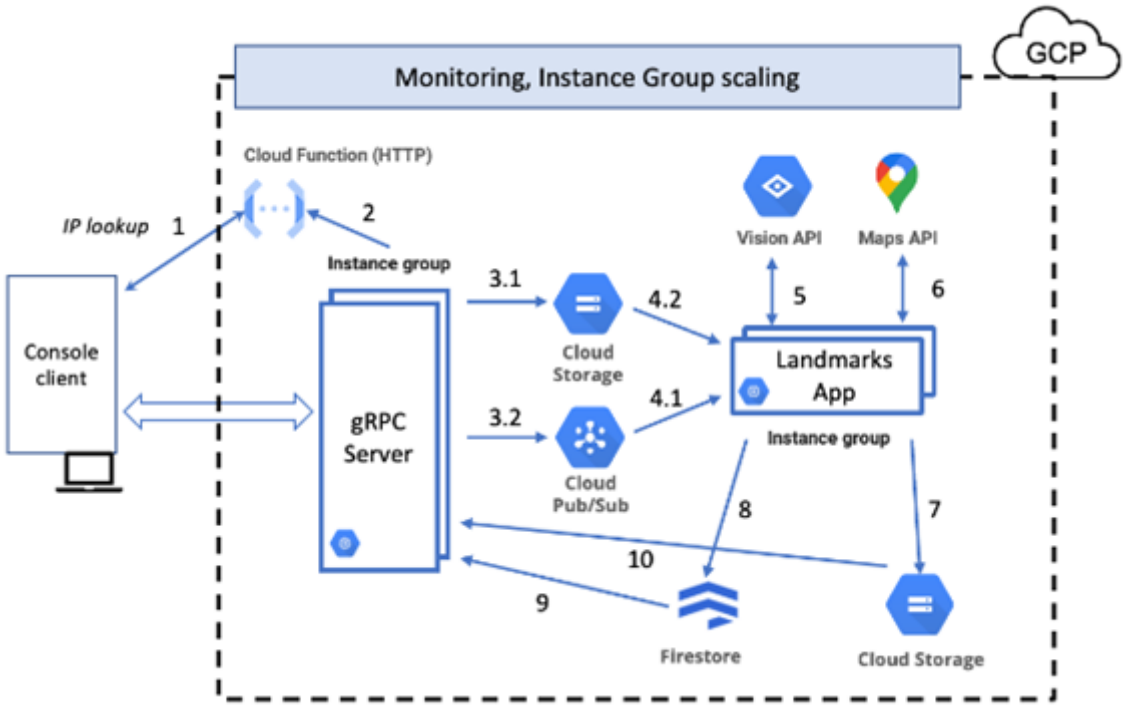


Figura 2.1: O esquema em blocos do enunciado.

2.1 Contrato

Em Grpc, o contrato refere-se à interface que define a estrutura do serviço usado para a comunicação entre o cliente e o servidor. O contrato usado no projeto foi escrito em *protocol buffer*, uma vez que assim não fica dependente da linguagem em que será implementado.

O serviço definido no contrato estabelece quatro métodos diferentes:

- **submitImage**: recebe como parâmetro uma imagem em stream de bytes e retorna o identificador da operação
- **requestMap**: recebe como parâmetro o identificador da operação e retorna uma imagem de um mapa em stream de bytes.
- **requestData**: recebe como parâmetro o identificador da operação e retorna um objeto data.
- **requestRelatedImages**: recebe como parâmetro o identificador da operação e retorna um stream de objetos relatedresult.

```
1 service Processor {
2   rpc submitImage(stream ImagePayload) returns (Identifier);
3   rpc requestMap(Identifier) returns (stream ImagePayload);
4   rpc requestData(Identifier) returns (Data);
5   rpc requestRelatedImages(Precision) returns (stream
     RelatedResults);
6 }
```

Listagem 2.1: Contrato.

```
1 message Coordinates {
2   float x = 1;
3   float y = 2;
4 }
5
6 message Landmark {
7   string name = 1;
8   Coordinates coordinates = 2;
9   float precision = 3;
10 }
```

```
11
12 message Data {
13   string id = 1;
14   repeated Landmark landmarks = 2;
15 }
```

Listagem 2.2: Objeto Data.

```
1 message RelatedResults{
2   string id = 1;
3   string name = 2;
4 }
```

Listagem 2.3: Objeto RelatedResults.

2.2 LandmarksApp

A aplicação LandmarksApp tem como função tratar das imagens enviadas pelo cliente, enviando-as para o VisionAPI e recebendo a resposta, pondo no Cloud Storage num Bucket, numa diretoria com o nome "maps". O LandmarksApp utiliza os serviços Cloud Pub/Sub, Cloud Storage e Firestore da Google Cloud.

Este é um subscritor de um tópico chamado "cn2223-g001-final-pubsub", com a subscrição de nome "cn2223-g001-final-pubsub", do tipo "pull". A aplicação aguarda por uma mensagem no tópico e, através desta mensagem, utiliza o URI de acesso à imagem com o nome do bucket e do blob presentes na mensagem para o envio da imagem para o VisionAPI. Este é apenas subscritor do tópico.

O VisionAPI é utilizado para obter o mapa estático do local de interesse, tendo um zoom de 20 com uma resolução de 1920x1080. A chave do Maps API é necessária, sendo utilizada uma variável de ambiente de nome "MAPS_API_KEY". Na receção do mapa estático vindo do uso das APIs, este é enviado para o bucket da mesma mensagem de onde se extrai o blob original, tornando o mapa estático público.

No fim, o LandmarksApp utiliza uma coleção de nome "cn2223-g001-final" com o Firestore, inserindo os dados, resultado do envio da imagem requesitada na mensagem pub/sub. Esta insere vários dados como o id da transação, o nome da imagem que foi utilizada no pedido e dados sobre as imagens enviadas para o Cloud Storage, como o nome e as coordenadas.

2.3 Instance Groups

Para a criação dos Instance Groups foram usados Instance Templates. Para esses templates, foram criadas imagens costumizadas que utilizam discos de máquinas virtuais criadas na consola Google na secção VM Instances. Existe um template por aplicação, apenas permanencendo o servidor e o LandmarkApp na nuvem. Estas máquinas têm uma pasta com o path /var/(server/landmark)/app, onde permanecem os ficheiros *jar* com o código do servidor Grpc e do LandmarkApp e também os scripts de iniciação das aplicações Java. Estes scripts são utilizados no startup script presente nos templates, com a linha **"bash /var/(server/landmark)/app/startup(server/landmark).sh"**. Assim, as máquinas ao ser criadas podem iniciar imediatamente as aplicações que contêm.

2.4 Cloud Function

Para o utilizador se conectar com o servidor Grpc, o mesmo dá uso a uma Cloud Function que devolve os IP's externos de cada instância, com uma máquina virtual responsável por correr a app do Servidor, pertencentes a um Instance Group de nome "grpc-instance-group".

Esta cloud function está presente no deliverable de código no formato do projeto de nome "GetIpsFunctionApp"com a classe Entrypoint onde está a função referida e um ficheiro .bat com os comandos necessários para dar deploy da mesma no projeto da Google Cloud.

```
1  public class Entrypoint implements HttpFunction {
2
3      @Override
4      public void service(HttpRequest request, HttpResponse
       response) throws Exception {
5          BufferedWriter writer = response.getWriter();
6          String projectId = "cn2223-t1-g01";
7          String zone = "europe-southwest1-a";
8          String instanceName = request.getFirstQueryParameter("
       name").orElse("grpc-instance-group");
9          try(InstancesClient client = InstancesClient.create()) {
10             for (Instance instance : client.list(projectId,zone)
       .iterateAll()){
```

```
11                  if (instance.getName().contains(instanceName) &&
     instance.getStatus().compareTo("RUNNING") == 0){
12                      String ip = instance.getNetworkInterfaces(0)
    .getAccessConfigs(0).getNatIP();
13                      writer.write(ip+",");
14                  }
15              }
16          }
17
18      }
19  }
```

Listagem 2.4: Class Entrypoint.

```
1 gcloud functions deploy getInstancesIps --allow-unauthenticated
    --project=cn2223-t1-g01 --region=europe-west1 --entry-point=
    Entrypoint --runtime=java11 --trigger-http --source=target/
    deployment --service-account=cn2223-t1-g01@appspot.
    gserviceaccount.com
```

Listagem 2.5: Comando de deploy.

3

Cloud Storage

O serviço Google Cloud Storage é um serviço da Google Cloud que permite o armazenamento de dados em nuvem com um alto nível de escalabilidade.

No projeto realizado, utilizamos o Cloud Storage para armazenar tanto as imagens submetidas pelos clientes, como as geradas pelas API's no decorrer da aplicação. As imagens guardadas são separadas em duas diretorias:

- /images
- /maps

Na diretoria /imagens são guardas todas as imagens submetidas pelos clientes. A cada imagem que é submetida, o servidor atribuí um id único, o qual vai ser usado para nomear a imagem da seguinte forma: "img-[id]". Isto permite que através do id, a imagem possa ser facilmente acessível pelos outros módulos.

Na diretoria /maps estão presentes todos os mapas gerados pelo Maps API na aplicação Landmark. Assim como nas imagens submetidas pelo cliente, os mapas também são identificados por um id no nome, que segue o seguinte formato: "static_map_[id]".

4

Cloud Firestore

A Cloud Firestore é uma base de dados NoSQL que permite guardar dados em formato de documentos, guardados no que é chamado de coleção, mantida no serviço cloud da Google. Uma das suas características mais notórias é a flexibilidade, permitindo uma organização complexa com objetos internos e subcoleções.

4.1 Coleção Firestore

A coleção realizada, com o nome de "cn2223-g001-final", para o sistema pretendido tem a seguinte estrutura:

- **LandmarkResponse:** Define o resultado de uma submissão de uma imagem pelo utilizador, que contêm um id gerado no servidor Grpc, no formato UUID, um blobId correspondente à imagem submetida e a listagem dos pontos de interesse encontrados.
- **Landmark:** Representa um ponto de interesse e as informações associadas, contêm um nome que o define, um par de coordenadas, um blobId correspondente ao mapa estático gerado e um grau de certeza relativo à sua identificação.
- **Coordinates:** Representação de uma coordenada, com um valor de latitude e longitude.

```
1 public class LandmarkResponse {
2     public String id;
3
4     public String origin;
5     public List<Landmark> images;
6 }
```

Listagem 4.1: Class LandmarkResponse.

```
1 public class Landmark {
2     public String blobId;
3     public Coordinates imageLocation;
4     public String landmarkName;
5     public float score;
6 }
```

Listagem 4.2: Class Landmark.

```
1 public class Coordinates {
2     public double latitude;
3     public double longitude;
4 }
```

Listagem 4.3: Class Coordinates.

4.2 Implementação

No lado do servidor, para extrair a informação necessária para devolver ao utilizador através do acesso aos documentos é necessário saber o nome da coleção e o identificador do mesmo.

```
1         DocumentReference docRef = db.collection(collectionName)
    .document(id);
2         ApiFuture<DocumentSnapshot> future = docRef.get();
3         DocumentSnapshot document = future.get();
```

Listagem 4.4: Snapshot de um documento.

A partir de cada snapshot é possível retirar os valores de origin, id e lista de pontos de interesse através de um .get(<nome do campo>). Ao tentar processar a lista, reparamos

que o retorno do Firestore estava no formato de um HashMap<String,Object>, e foi desse modo que decidimos retirar a informação de cada "Landmark". À semelhança do retorno da lista, também as coordenadas vêm no mesmo formato.

```
1         ArrayList<HashMap<String,Object>> images = (ArrayList<
    HashMap<String,Object>>) document.get("images");
2         ArrayList<Landmark> landmarks = new ArrayList<>();
3         if (images != null){
4             for (int i = 0; i < images.size(); i++) {
5                 String blobId = (String) images.get(i).get("
    blobId");
6                 String landmarkName = (String) images.get(i).get
    ("landmarkName");
7                 Float score = ((Double) images.get(i).get("score
    ")).floatValue();
8                 HashMap<String,Double> coord = (HashMap<String,
    Double>) images.get(i).get("imageLocation");
9                 double x = coord.get("latitude");
10                double y = coord.get("longitude");
11                Coordinates loc = new Coordinates(x,y);
12                Landmark lm = new Landmark(blobId,loc,
    landmarkName,score);
13                landmarks.add(lm);
14            }
15        }
```

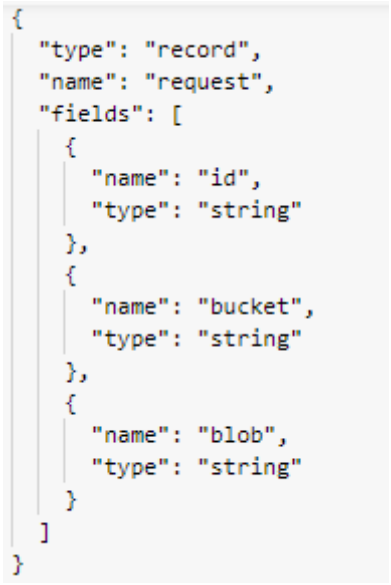
Listagem 4.5: Processamento de uma lista de Landmarks.

5

Pub/Sub

O Pub/Sub é um padrão de comunicação que permite a troca de mensagens entre vários sistemas distribuídos. Seguindo este padrão, os publicadores conseguem enviar mensagens para um determinado tópico, que por sua vez irá reencaminhar para as suas várias subscrições, subscrições essas que possuem vários consumidores, a quem entregam as mensagens.

No contexto do nosso projeto, utilizamos o serviço Pub/Sub da Google Cloud para realizar a comunicação entre o nosso ServidorGrpc e a aplicação Landmark, responsável pelos acessos às API's usadas para o processamento dos pedidos. Assim, o nosso ServidorGrpc serve como publicador e envia uma mensagem que contém as informações necessárias para se acessar à imagem-alvo, guardada no Google Cloud Storage, para uma única subscrição. A aplicação Landmark, por sua vez, serve como consumidor, estando subscrita ao tópico usado pelo servidor, vai consumir a mensagem e obter as informações necessárias para acessar a imagem. Vale notar que a subscrição é do tipo *pull*, o que significa que a mensagem é obtida por meio de um pedido pull enviado pelo Landmark à subscrição. A mensagem enviada no Pub/Sub segue o seguinte formato:



```
{
  "type": "record",
  "name": "request",
  "fields": [
    {
      "name": "id",
      "type": "string"
    },
    {
      "name": "bucket",
      "type": "string"
    },
    {
      "name": "blob",
      "type": "string"
    }
  ]
}
```

Figura 5.1: Schema da mensagem Pub/Sub.

É também importante mencionar que deve apenas existir uma única subscrição do tópico usado pelo servidor, independentemente do número de instâncias da aplicação Landmark, isto para garantir que cada mensagem enviada é consumida uma única vez.

6

Conclusão

O desenvolvimento deste projeto permitiu-nos não só aplicar como também aprofundar os nossos conhecimentos sobre os vários temas abordados no decorrer da cadeira. Fomos capazes de conjugar as várias tecnologias utilizadas em aula para produzir uma única aplicação sólida e escalável que demonstra os nossos conhecimentos nos tópicos de computação na nuvem.

6.1 Divisão de trabalho

Neste ponto do documento serão indicadas as participações por membro em cada parte do trabalho final, vale a pena referir que estas indicações serão feitas num modo geral uma vez que todas as partes do trabalho tiveram participação coletiva.

- **App Cliente:** Miguel Neves
- **App Servidor:**
  - **Grpc:** Manuel Henriques
  - **Firestore:** Manuel Henriques, Miguel Neves
  - **Pub/Sub:** Manuel Henriques
  - **Cloud Storage:** Manuel Henriques
- **Contrato:** Miguel Neves

- **App LandMarks:** Ricardo Rocha
- **Cloud Function:** Miguel Neves
- **Instance Groups:** Ricardo Rocha

6.2 Dificuldades encontradas

O maior ponto a notar seria os erros encontrados ao testar os instance groups, que talvez pelo facto dos executáveis referentes à app do Servidor e LandMarks não estarem com as dependências necessárias. No entanto, estes erros não são explícitos o suficiente para chegar a essas conclusões em tempo útil.

Tivemos também vários problemas com algumas das dependências Maven utilizadas no projeto, nomeadamente as dos serviços Grpc, Pub/Sub e Firestore. Acreditamos que o erro seja um resultado do facto de estas possuírem um módulo chamado "guava", como dependência interna, que por sua vez entrava em conflito com as suas várias instâncias. A solução encontrada foi remover manualmente o "guava"de todas as dependências e, posteriormente, adicionar diretamente o "guava"como uma dependência do projeto.

Outro ponto seria o facto da linguagem usada na descrição do contrato, presente no enunciado, estar aberto para interpretação, o que no início causou alguma dúvida no seu funcionamento geral.

Para finalizar, na app do Servidor, quando extraímos as informações dos documentos, esperávamos poder fazer a chamada de document.toObect(<nome da classe correspondente>.java) da mesma maneira que usamos o .set(<objeto correspondente>) no lado da app das Landmarks. O aspeto mais importante neste caso seria o facto de as listas de Landmarks não corresponderem exatamente a um List<Landmarks> como estaria definido, mas sim a um ArrayList< HashMap<String,Object> > o que nos obriga a ir buscar cada campo manualmente para cada um dos elementos da lista.