

Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores

Trabalho Final

47283 : Ricardo Duarte Cardoso Bernardino (a47283@alunos.isel.pt)

47249 : Miguel Henriques Couto de Almeida (a47249@alunos.isel.pt)

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

Professor : Doutor José Manuel de Campos Lages Garcia Simão

Contents

List of Figures	v
Listagens	vii
1 Introdução	1
1.1 O que é Elasticidade	1
1.2 O que é o Serviço Cloud Storage?	2
1.3 O que é o Serviço Firestore?	2
1.4 O que é o Serviço Pub/Sub?	3
1.5 O que é o Serviço Compute Engine?	3
1.6 O que é o Serviço Vision API?	3
1.7 O que é o Serviço Maps API?	3
1.8 O que é o Serviço Cloud Functions?	4
2 Desenvolvimento	5
2.1 Contrato Protobuf	6
2.2 Servidor gRPC	7
2.3 Aplicação Landmarks App	8
2.4 Configuração do Serviço Pub/Sub	8
2.5 Configuração do Serviço Storage	9
2.6 Configuração do Serviço Firestore	10

2.7	Configuração do Serviço Cloud Function	10
2.8	Aplicação Cliente	11
2.9	Deployment	12
3	Conclusão	15

List of Figures

2.1	Arquitetura do Projeto	5
2.2	Pub/Sub	9
2.3	Storage	9
2.4	Firestore	10
2.5	Interface Cliente	12
2.6	Deployment na Cloud	13

Listagens

2.1	Contrato Protobuff	6
2.2	Código Cloud Function	10



Introdução

1.1 O que é Elasticidade

A elasticidade é a capacidade de um certo sistema adaptar-se dinamicamente às necessidades de processamento e armazenamento de dados. Permite que os recursos computacionais, como servidores, capacidade de armazenamento e largura de banda, sejam automaticamente dimensionados de acordo com a procura do momento. A elasticidade é especialmente benéfica na área da Computação na Nuvem pois permite que as empresas dimensionem os seus recursos de forma rápida e eficiente, sem a necessidade de investir em infraestrutura física adicional. Proporcionando uma maior flexibilidade operacional, agilidade e eficiência nos custos, já que os recursos podem ser alojados e desalojados conforme necessário, de acordo com a procura.

No âmbito deste trabalho final, a elasticidade do projeto é realizada através de configurações na plataforma *Google Cloud Platform (GCP)*, a plataforma usada neste projeto, tanto para configurações de armazenamento em nuvem, como o uso do sistema *Pub-Sub*, *Cloud Storage*, etc. A plataforma *GCP* fornece a capacidade de configuração de Instance Groups, estes têm como base imagens das máquinas virtuais desejadas, para que possa replica-las ao criar instâncias das mesmas. Estas réplicas auxiliam o atendimento de pedidos, quando estes são demasiados para ser atendidos por uma só instância, sendo assim os pedidos distribuídos pelas várias instâncias da mesma aplicação, correndo em imagens diferentes da mesma máquinas virtual.

Como pedido para este trabalho final, foi um criado um *Instance Group* para a máquina

virtual que contém a aplicação Servidor, e outro *Instance Group* para a máquina virtual que contém a aplicação Landmarks. A dimensão de cada *Instance Group* pode ser configurada para ser aumentada ou diminuída para um certo número. Neste caso foi configurado, como pedido, que o *Instance Group* da aplicação Servidor escala-se para 3 instâncias no máximo, ou diminuída para 1 no mínimo. Enquanto a aplicação Landmarks escala-se para 2 no máximo, ou diminuída para 0, no mínimo.

1.2 O que é o Serviço Cloud Storage?

O Cloud Storage é um serviço para armazenamento de objetos no *Google Cloud*. Um objeto é um dado imutável composto de um arquivo em qualquer formato. Os objetos, ou *blobs*, são armazenados em containers chamados de *buckets*. Todos os *buckets* estão associados a um projeto. Cada projeto, *bucket* e objeto no *Google Cloud* é um recurso do *Google Cloud*, assim como as instâncias do *Compute Engine* (Máquinas Virtuais).

Para este trabalho final foram criados 2 buckets, um para o armazenamento de imagens, e outro para o armazenamento de mapas.

1.3 O que é o Serviço Firestore?

O *Cloud Firestore* é uma base de dados documental *NoSQL* que permite armazenar, sincronizar e consultar dados facilmente para aplicações Web e de dispositivos móveis, em escala global. Seguindo o modelo de dados *NoSQL*, é possível armazenar dados em documentos que contém mapeamentos de campos para valores. Estes documentos são armazenados em coleções. Os tipos de dados dos documentos podem variar entre *Strings*, *Arrays*, ou até objetos. O *Cloud Firestore* também oferece a possibilidade de integração com outros serviços do *GCP*, como o *Cloud Functions*.

No âmbito deste trabalho final, o *Cloud Firestore* é usado para armazenar os documentos respetivos a cada pedido de processamento de imagem, contendo informação importante sobre este, como os identificadores dos mapas gerados após este processamento, sendo compostos pelo *bucket* e *blob* respetivo de cada mapa, para fácil acesso a este no *Cloud Storage*. Estes documentos são também compostos pelo nomes, e coordenadas das localizações de cada mapa, e também a confiança do seu processamento.

1.4 O que é o Serviço Pub/Sub?

O *Pub/Sub* é um serviço de mensagens assíncrono e escalável que separa os serviços que produzem mensagens dos serviços que consomem e processam essas mensagens. Permite criar sistemas de produtores e consumidores de eventos, *producers* e *consumers*, respectivamente. Os *producers* enviam mensagens para o serviço *Pub/Sub*, sem se preocupar com como ou quando essas respectivas mensagens serão consumidas. O *Pub/Sub* entrega essas mensagens para todos os consumidores inscritos aos chamados tópicos.

O *Pub/Sub* permite, para este trabalho, que a aplicação Servidora publique neste uma mensagem com o identificador do pedido respectivo, para que a aplicação Landmarks possa retirar a imagem respectiva do *Cloud Storage*, para a processar, e criar o documento respectivo para esse pedido, no *Cloud Firestore*.

1.5 O que é o Serviço Compute Engine?

O *Compute Engine* oferece uma plataforma para criar e gerir VM's (Máquinas Virtuais) com flexibilidade e controle total sobre os recursos computacionais. Estas VM's podem ser configuradas para serem alojadas numa certa região, ou por exemplo terem um certo tamanho de espaço de armazenamento no seu disco.

Estas máquinas virtuais são chave para executar as aplicações desejadas, como no caso deste trabalho final, a aplicação Servidor numa máquina, e a Landmarks App noutra.

1.6 O que é o Serviço Vision API?

O Serviço Vision API é um serviço oferecido na *Google Cloud Platform* que permite a análise de imagens por meio de tecnologias de visão computacional. Utiliza algoritmos de *Machine Learning* para identificar e extrair informações relevantes de imagens, como objetos, rostos, textos, etc.

No caso deste trabalho, a informação relevante extraída de cada imagem foram monumentos ou locais famosos (*landmarks*), a sua localização e confiança na resposta.

1.7 O que é o Serviço Maps API?

O Serviço Maps API é um serviço da *Google Cloud Platform* que permite a integração de mapas interativos e informações geográficas em aplicações. Fornece acesso a dados e

ferramentas do *Google Maps*, permitindo a criação de mapas estáticos.

O uso do serviço *Vision API* permite, neste trabalho final, a obtenção de coordenadas de monumentos e locais famosos detectados nas imagens analisadas. Através destas coordenadas, o serviço *Maps API* gera mapas estáticos dessas localizações que serão guardados no *bucket* respectivo dos mapas no *Cloud Storage*.

1.8 O que é o Serviço Cloud Functions?

O *Google Cloud Functions* é um serviço fornecido pelo *GCP* que permite desenvolver e executar código de forma serverless na nuvem. É projetado para responder a eventos e executar pequenas funções de maneira rápida e escalável. O *Google Cloud Functions* é usado para criar e implantar funções individuais que são acionadas por eventos específicos, como acionadores de armazenamento, acionadores do *Pub/Sub*, chamadas *HTTP*, eventos de alteração do *Firestore*, entre outros. Quando um evento é acionado, a função correspondente é executada em um ambiente isolado, sem necessidade de preocupação com a infraestrutura subjacente.

No âmbito do trabalho final, o *Google Cloud Functions* foi configurado para alocar uma função *Service 2.2*, que consulta todas as endereços IP's das instâncias de máquinas virtuais que estão a executar a aplicação servidora. O cliente depois pode prosseguir em escolher um desses endereços para realizar os seus pedidos, obedecendo à política de elasticidade implementada neste trabalho e evitando possíveis *bottlenecks* numa só instância.

Desenvolvimento

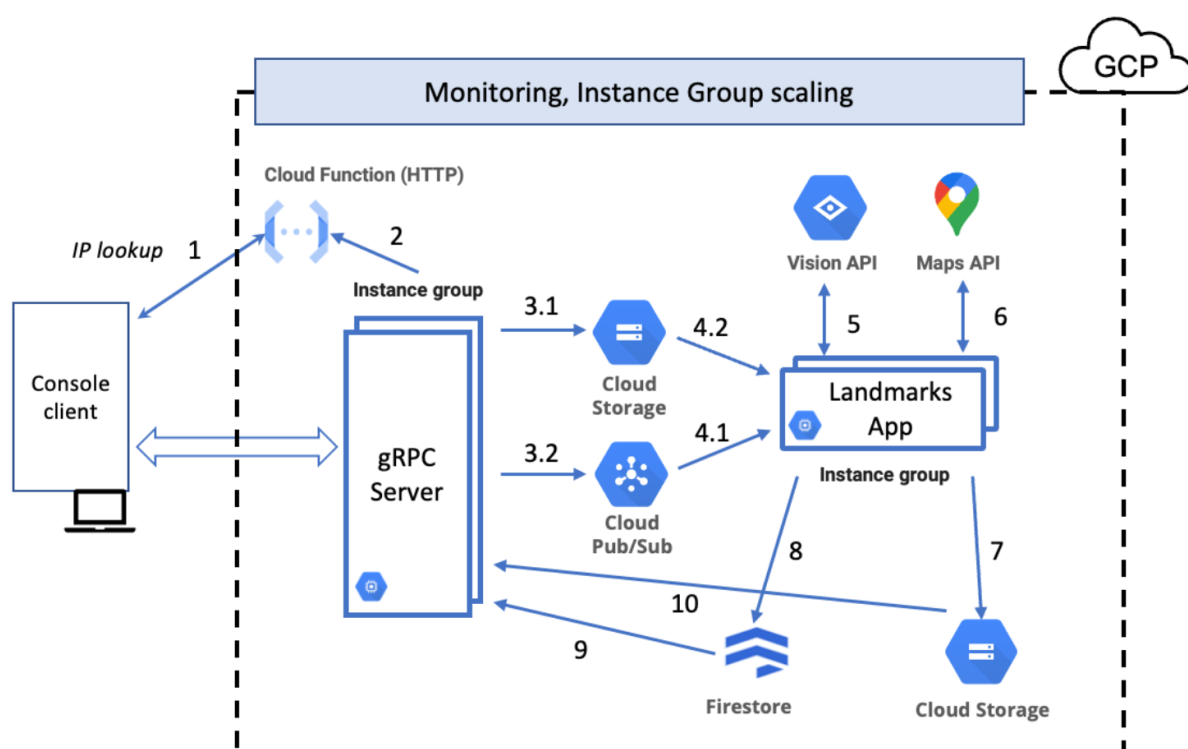


Figura 2.1: Arquitetura do Projeto

2.1 Contrato Protobuff

O primeiro passo no desenvolvimento deste trabalho final foi a conceção do contrato gRPC, composto pelos pedidos necessários à realização deste trabalho, e dos tipos necessários como parâmetros de entrada e saída destes pedidos. Como desejado, o primeiro pedido *uploadImage* recebe uma imagem, por uma *stream* de blocos, contendo a sua metadata, e conteúdo, e retorna uma resposta composta por um identificador único, e o estado da resposta do servidor a este pedido.

O segundo método do contrato, *getLandmarks* obtém através do identificador único gerado e retornado pelo servidor a lista de landmarks que foram identificadas com o seu nome, posição e confiança, enquanto o terceiro método *getImage* obtém através do mesmo identificador a imagem produzida pelo Maps API com as posições retornadas pela Vision API guardada pelo Landmarks App no bucket criado para mapas, por último o quarto método faz uma query ao serviço Firestore pelos documentos que contêm campos *confidence* a cima do passado como parâmetro, ou seja faz uma pesquisa pelas imagens identificadas com uma confiança a cima da passada e retorna os seus identificadores.

```
1 service CN2223TFService {
2   rpc uploadImage(stream Block) returns (ImageUploadResponse);
3   rpc getLandmarks(Identifier) returns (SubmissionResponse);
4   rpc getImage(Identifier) returns (stream ImageResponse);
5   rpc getAllImages(Parameters) returns (ImagesResponse);
6 }
7 message Block {
8   Image image=1;
9 }
10 message Image {
11   Metadata metadata=1;
12   bytes content = 2;
13 }
14 message Metadata {
15   string name=1;
16   string type=2;
17 }
18 message ImageUploadResponse {
19   Identifier identifier = 1;
20   Status status = 2;
21 }
```

```
22 message Identifier{
23     string uuid=1;
24 }
25 message SubmissionResponse {
26     repeated DetectedLandmark landmarks = 1;
27 }
28 message DetectedLandmark{
29     string name=1;
30     double latitude =2;
31     double longitude =3;
32     double confidence =4;
33 }
34 message ImageResponse{
35     Image image = 1;
36 }
37 message Parameters {
38     double certainty=1;
39 }
40 message ImagesResponse {
41     repeated IdentifiedImage identified_image = 1;
42 }
43 message IdentifiedImage{
44     string name = 1;
45     string location=2;
46 }
47 enum Status {
48     SUCCESS = 0;
49     FAILURE = 1;
50 }
```

Listagem 2.1: Contrato Protobuff

2.2 Servidor gRPC

O servidor gRPC neste trabalho final foi o responsável por orquestrar a comunicação entre o cliente e a Landmarks App submetendo os pedidos para a mesma através do canal Pub/Sub, entretanto forneceu também a comunicação entre o cliente e o resto dos serviços da Google Cloud Platform, incluindo o Firestore e Storage, onde estavam

guardadas as informações resultantes do processamento feito pela landmarksApp.

2.3 Aplicação Landmarks App

A aplicação Landmarks App no âmbito do trabalho final foi responsável por realizar todo o trabalho vindo do Pub/Sub, na forma de tickets, esta quando ativada retirava do sistema Pub/Sub a primeira mensagem e começaria o trabalho com base naquele ticket, ticket este que vem acompanhado de um identificador único, representando uma imagem guardada no bucket em Storage dedicado às imagens que foram submetidas pelo cliente.

A aplicação descarregava a imagem logo de seguida do bucket e enviaria a mesma para a Vision API aguardando uma resposta desta com os landmarks identificados, seguindo esta resposta, pegando nesta vez nas localizações dos landmarks, pediria à Maps API que lhe desse uma imagem para cada posição fornecida e estas imagens após serem retornadas seriam enviadas para o bucket das imagens sendo nomeadas através de um identificador único, identificador este que também seria posteriormente usado para identificar o sítio onde foram alojadas no bucket dedicado aos Mapas. Concluído todos estes passos e com a informação toda o último passo foi criar o documento identificado com o mesmo identificador que recebeu na mensagem enviada pelo Pub/Sub, a informação da Vision API e os identificadores únicos dos mapas.

Este documento ficaria então disponível para consulta posteriormente pelo servidor a pedido do cliente.

2.4 Configuração do Serviço Pub/Sub

Para o serviço Pub/Sub usámos o modo Push e foi configurado sem recorrer a um schema para as mensagens. Foi criado um tópico e uma subscrição, usada por todas as instâncias de landmarks App a correr.

LIST

METRICS

Filter

Filter topics

<input type="checkbox"/>	Topic ID ↑	Encryption key	Topic name
<input type="checkbox"/>	detectionworkers	Google-managed	projects/cn2223-t1-g08/topics/detectionworkers

(a) Tópico Pub/Sub

Subscriptions

CREATE SUBSCRIPTION

DELETE

SHOW

LIST

METRICS

Filter

Filter subscriptions

<input type="checkbox"/>	State	Subscription ID ↑	Delivery type	Topic name	Ack deadline	Retention	Message ordering	Exactly once delivery	Expiration
<input type="checkbox"/>		detectionworkers-sub	Pull	projects/cn2223-t1-g08/topics/detectionworkers	10 seconds	7 days	Disabled	Disabled	31 days

(b) Subscrição Pub/Sub

Figura 2.2: Pub/Sub

2.5 Configuração do Serviço Storage

Para o serviço de Storage criámos dois buckets, um para alojar as imagens vindas do servidor e outro para os mapas vindas da Landmarks App.

landmark_detection_app_images

Location	Storage class	Public access	Protection
europe-southwest1 (Madrid)	Standard	Not public	None

OBJECTS

CONFIGURATION

PERMISSIONS

PROTECTION

LIFECYCLE

OBSERVABILITY

NEW

INVENTORY REPORTS

Buckets > landmark_detection_app_images

UPLOAD FILES

UPLOAD FOLDER

CREATE FOLDER

TRANSFER DATA

MANAGE HOLDS

DOWNLOAD

DELETE

Filter by name prefix only

Filter

Filter objects and folders

<input type="checkbox"/>	Name	Size	Type	Created	Storage class	Last modified	Public access	Version history	Encryption
<input type="checkbox"/>	d505ff37-1e85-4d68-8628-5cf39d...	4.4 KB	application/octet-stream	Jun 3, 2023, 4:55:58 PM	Standard	Jun 3, 2023, 4:55:58 PM	Not public	—	Google-managed key

(a) Bucket para as Imagens

landmark_detection_app_maps

Location

Storage class

Public access

Protection

europe-southwest1 (Madrid)

Standard

Not public

None

OBJECTS

CONFIGURATION

PERMISSIONS

PROTECTION

LIFECYCLE

OBSERVABILITY

NEW

INVENTORY REPORTS

Buckets

>

landmark_detection_app_maps

UPLOAD FILES

UPLOAD FOLDER

CREATE FOLDER

TRANSFER DATA

MANAGE HOLDS

DOWNLOAD

DELETE

Filter by name prefix only

Filter

Filter objects and folders

<input type="checkbox"/>	Name	Size	Type	Created	Storage class	Last modified	Public access
<input type="checkbox"/>	<div><div></div><div>87814cee-6af5-4afb-ab35-a504ed...</div></div>	13.3 KB	application/octet-stream	Jun 3, 2023, 4:57:58 PM	Standard	Jun 3, 2023, 4:57:58 PM	Not public
<input type="checkbox"/>	<div><div></div><div>92d46ac7-6ba7-45b4-b5b3-56f4d...</div></div>	13.1 KB	application/octet-stream	Jun 3, 2023, 4:57:58 PM	Standard	Jun 3, 2023, 4:57:58 PM	Not public

(b) Bucket para os Mapas

Figura 2.3: Storage

2.6 Configuração do Serviço Firestore

Para o serviço Firestore foram criados documentos com o formato 2.4a. Guardámos os dados no formato de array pela facilidade em indexar.

cn2223-t1-g08	ImagesAndLandmarks	d505ff37-1e85-4d68-8628-5cf39d632e40
+ START COLLECTION ImagesAndLandmarks OcupacaoTemporaria	+ ADD DOCUMENT d505ff37-1e85-4d68-8628-5cf39d632e40	+ START COLLECTION + ADD FIELD ImageId: "landmark_detection_app_images/d505ff37-1e85-4d68-8628-5cf39d632e40" LocationName 0: "Statue Of Liberty" 1: "Liberty Island" LocationPosition 0: "40.6892494;-74.04450039999999" 1: "40.689730999999995;-74.0453577" MapId 0: "landmark_detection_app_maps/87814cee-faf5-4afb-ab35-a504ed3abcd6" 1: "landmark_detection_app_maps/92d46ac7-6ba7-45b4-b5b3-56f4d9b43fda" confidence 0: 0.786922037601471 1: 0.674034059047699

(a) Documentos em Firestore

Figura 2.4: Firestore

2.7 Configuração do Serviço Cloud Function

A Cloud Function foi deployed através da máquina do aluno e ficou encarregue de retornar os endereços IP's dos servidores gRPC para que o cliente se pudesse ligar a um deles sendo o trigger de tal ação um pedido HTTP.

```

1 public class Entrypoint implements HttpFunction {
2     @Override
3     public void service(HttpRequest request, HttpResponse response) throws Exception
4     {
5         BufferedWriter writer = response.getWriter();
6         String project_id = request.getFirstQueryParameter("projectId").orElse("
cn2223-t1-g08");
7         String zone = request.getFirstQueryParameter("zone").orElse("europe-west1-
b");
8         String grpName = request.getFirstQueryParameter("grpName").orElse("
instance-group-servers");
9         listManagedInstanceGroupVMs(project_id, zone, grpName, writer);
10    }
11    static void listManagedInstanceGroupVMs(String project, String zone, String
grpName, BufferedWriter writer) throws IOException {

```

```
12     try (InstancesClient client = InstancesClient.create()) {
13         for (Instance instance : client.list(project, zone).iterateAll()) {
14             if(instance.getName().contains(grpName) && instance.getStatus().
equalsIgnoreCase("running")) {
15                 String ip = instance.getNetworkInterfaces(0).getAccessConfigs(0).getNatIP
();
16                 writer.write(ip + ", ");
17             }
18         }
19     }
20 }
21 }
```

Listagem 2.2: Código Cloud Function

Esta está a retornar os endereço IP's separados por uma vírgula das máquinas a correr presentes no "instance-group-servers" alojados na região "europe-west1-b" pertencentes ao projeto "cn2223-t1-g08".

2.8 Aplicação Cliente

Na aplicação cliente, tirámos partido dos métodos partilhados na interface 2.1 e demos ao cliente uma interface texto para interagir com a mesma, sendo possível através desta chamar todos os métodos e receber notificações tanto dos pedidos como das respostas em tempo real.

Para todos os pedidos recorreremos aos stubs não bloqueantes e antes sequer do cliente começar a utilizar a aplicação, esta recorre à Cloud Function 2.7 e aos endereços IP's retornados por ela para abrir um Managed Channel com o servidor.

Enter an Option: 1

Insert image path

/Users/miguelalmeida/Desktop/download.jpg

MENU

Options for Google Storage Operations:

- 1: uploadImage
- 2: getLandmarks
- 3: getImage
- 4: getAllImages

99: Exit

Enter an Option:

Image and Request ID: 193e2c98-cbcc-44f8-b6aa-59da8965d93a

Done

1

(a) Upload da Imagem

Enter an Option:
Image and Request ID: 193e2c98-cbcc-44f8-b6aa-59da8965d93a
Done
2

Insert requestID
193e2c98-cbcc-44f8-b6aa-59da8965d93a

MENU

Options for Google Storage Operations:

- 1: uploadImage
- 2: getLandmarks
- 3: getImage
- 4: getAllImages

99: Exit

Enter an Option:

Objects in image given by requestId:

Name: Belén Tower Garden | Latitude: 38.6927205 | Longitude: -9.215710399999999 | Confidence: 0.7065483331680298
Name: Belén Tower | Latitude: 38.691583699999995 | Longitude: -9.215977299999999 | Confidence: 0.6941734552383423

Done

(b) Landmarks Identificadas na Imagem

Enter an Option:
Objects in image given by requestId:

Name: Belén Tower Garden | Latitude: 38.6927205 | Longitude: -9.215710399999999 | Confidence: 0.7065483331680298
Name: Belén Tower | Latitude: 38.691583699999995 | Longitude: -9.215977299999999 | Confidence: 0.6941734552383423

Done
3

Insert requestID
193e2c98-cbcc-44f8-b6aa-59da8965d93a

MENU #####
Options for Google Storage Operations:
1: uploadImage
2: getLandmarks
3: getImage
4: getAllImages
99: Exit
Enter an Option: File created successfully.

Image <a3ab95da-d587-491f-a7ec-6ae289642004.png> downloaded to <src/main/java/isel/cn/images>
Done

(c) Pedir o Mapa da Landmark



(d) Mapa Guardado

Insert the minimum confidence of the object presence in image [0-1]:
0.74

MENU #####
Options for Google Storage Operations:
1: uploadImage
2: getLandmarks
3: getImage
4: getAllImages
99: Exit
Enter an Option:

Landmarks with certainty greater than 0.74

Image: landmark_detection_app_images/4b248b09-3744-4c07-8d82-d811cc18c73d Location: The Regent's Park
Image: landmark_detection_app_images/d505ff37-1e85-4d68-8628-5cf39d632e40 Location: Statue Of Liberty

Done

(e) Obter as Landmarks com Confiança

Figura 2.5: Interface Cliente

2.9 Deployment

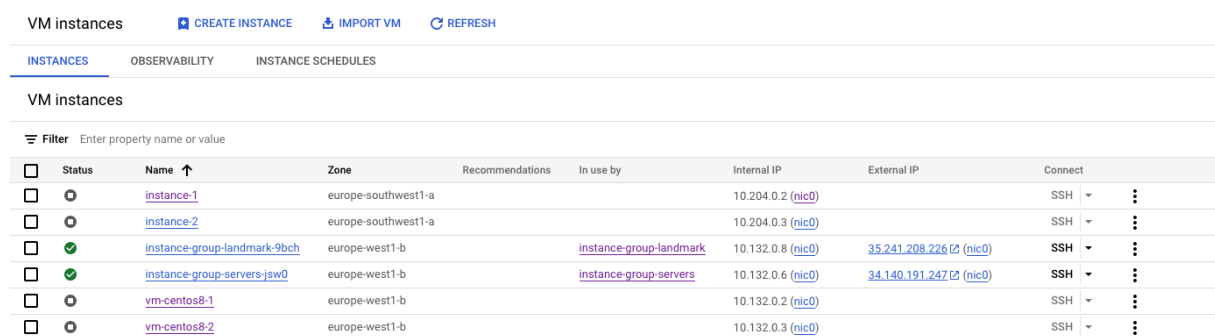
Tal como foi mencionado previamente neste relatório, à medida que a aplicação foi sendo desenvolvida, esta fora testada em máquinas locais, mas um dos requisitos do enunciado era que fosse feito o deployment na Cloud.

Foram portanto feitas as adaptações necessárias aos servidores que tínhamos implementado para que tal feito fosse possível. Inicialmente, foram criadas duas VM's onde dentro dos diretórios criados com base "/var/<server/app>" foram colocados tanto os

jar's do servidor numa VM e do LandmarkApp noutra, ao lado destes jar's está o ficheiro JSON responsável por dar permissões de acesso aos serviços da Google Cloud.

Nos running scripts destas VM's é colocada como variável de ambiente a localização deste ficheiro JSON e colocado o servidor/app a correr recorrendo ao comando "java -jar <server/app>"

Tendo estes passos feitos, foi realizado um teste com um servidor e uma landmark, tendo o cliente ligado-se diretamente ao servidor gRPC. Averiguado que tudo estava a correr como esperado avançámos para a criação dos instance groups, foram então criadas duas imagens tendo por base o disco das duas VM's criadas anteriormente, depois criámos os instance group templates recorrendo a estas imagens colocando o número de instâncias dito no enunciado do trabalho. Com os instance group templates criados, colocámos ambos em execução e adaptámos o código do Cliente para agora tirar partido da Cloud Function que tínhamos implementado.



VM instances

CREATE INSTANCE IMPORT VM REFRESH

INSTANCES OBSERVABILITY INSTANCE SCHEDULES

VM instances

Filter Enter property name or value

<input type="checkbox"/>	Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect
<input type="checkbox"/>		instance-1	europa-southwest1-a			10.204.0.2 (nic0)		SSH ▾ ⋮
<input type="checkbox"/>		instance-2	europa-southwest1-a			10.204.0.3 (nic0)		SSH ▾ ⋮
<input type="checkbox"/>		instance-group-landmark-9bch	europa-west1-b		instance-group-landmark	10.132.0.8 (nic0)	35.241.208.226 (nic0)	SSH ▾ ⋮
<input type="checkbox"/>		instance-group-servers-jsw0	europa-west1-b		instance-group-servers	10.132.0.6 (nic0)	34.140.191.247 (nic0)	SSH ▾ ⋮
<input type="checkbox"/>		vm-centos8-1	europa-west1-b			10.132.0.2 (nic0)		SSH ▾ ⋮
<input type="checkbox"/>		vm-centos8-2	europa-west1-b			10.132.0.3 (nic0)		SSH ▾ ⋮

(a) Instance Groups a correr

Figura 2.6: Deployment na Cloud

Com tudo isto concretizado a aplicação como um todo ficou a funcionar com a opção de autoscaling desabilitada, sendo portanto necessário aumentar e diminuir manualmente o número de instâncias tanto do Servidor como da Landmarks App se for esse o efeito desejado.



Conclusão

Este trabalho serviu para consolidar tudo o que foi sendo lecionado ao longo do semestre na UC tendo chegado o grupo a um resultado satisfatório na execução do mesmo.

Houve uma distribuição desde o início do trabalho que teria de ser realizado tendo sido a primeira divisão feita entre quem estaria responsável pelo cliente e pelo servidor, com estes dois implementados o Miguel ficou responsável pela implementação do serviço Pub/Sub e deployment da Cloud Function, enquanto o Ricardo trabalhou na Landmark App e Firestore, já no final quando já toda a aplicação estava a funcionar, foram ambos os elementos para a interface Web GCP configurar as VM's e todos os passos necessários para colocar a aplicação hospedada nos instance groups a correr.

O grupo gostaria de ter acrescentado outras funções no contrato gRPC que possibilitassem ao cliente aumentar e diminuir o número de instâncias a correr nos instance groups, também foi averiguado se tal funcionalidade, em vez de ser disponibilizada ao cliente, fosse implementada de forma automática numa Cloud Function a correr esporadicamente deixando esta como responsável por resolver o problema da escalabilidade.

