

**Trabalho Final**

|  |  |
| --- | --- |
|  | Daniel Caseiro  Douglas Incáo  Kaiwei |
|  |  |

|  |  |
| --- | --- |
| Orientadores | José Simão |
|  |  |

Relatório realizado no âmbito de Computação na Nuvem,  
do curso de licenciatura em Engenharia Informática e de Computadores  
Semestre de Verão 2022/23

Junho de 2023

**Instituto Superior de Engenharia de Lisboa**  
Licenciatura em Engenharia Informática e de Computadores

**Trabalho Final**

|  |  |
| --- | --- |
| 46052 | Daniel Caseiro |
| 47269 | Douglas |
| 48294 | Kaiwei |

|  |  |
| --- | --- |
| Orientadores: | José Simão |
|  |  |

Relatório realizado no âmbito de Computação na Nuvem,  
do curso de licenciatura em Engenharia Informática e de Computadores  
Semestre de Verão 2022/23

Junho de 2023

# Resumo

Esta aplicação foi desenvolvida com o objetivo de processar imagens para identificar a presença de monumentos ou locais famosos. Esta utiliza serviços integrados da Google Cloud Platform (GCP) para armazenamento, comunicação e computação, oferecendo uma solução flexível e escalável na nuvem.

O sistema permite que os clientes enviem arquivos de imagem para deteção de monumentos, armazenando-os no Cloud Storage. Após a identificação dos locais, o sistema recupera uma imagem correspondente a um mapa estático da região do monumento identificado. Além disso, o sistema possui capacidade de adaptação a variações de carga, permitindo aumentar ou diminuir sua capacidade de processamento de imagens simultaneamente.

As funcionalidades do sistema são disponibilizadas por meio de uma interface gRPC, que permite às aplicações cliente interagirem com o sistema de forma eficiente. Essa interface oferece operações para submissão de imagens, obtenção de resultados de deteção de monumentos e recuperação de informações sobre as fotos processadas.

Para garantir a disponibilidade e o load balancing do sistema, são utilizadas várias réplicas do servidor gRPC, executando-se em instâncias de máquinas virtuais (VMs) de um instance group. Além disso, réplicas da aplicação Landmarks App, responsáveis pela identificação de monumentos nas fotos, também são executadas em instâncias de VMs. A arquitetura do sistema utiliza serviços da GCP, como o Cloud Storage, Firestore, Pub/Sub, Compute Engine, Cloud Functions, Vision API e Static Maps API, para armazenamento, comunicação, computação, deteção de monumentos e obtenção de mapas estáticos.

Neste relatório, serão apresentados os detalhes de implementação da aplicação, bem como os fluxos de operações e a sua capacidade de elasticidade. Serão discutidos os serviços da GCP utilizados, as integrações entre eles e as APIs empregadas para realizar as funcionalidades desejadas. Por fim, serão abordados aspetos relevantes da implementação e as considerações necessárias para garantir o desempenho, a escalabilidade e a disponibilidade do sistema na nuvem.

**Palavras-chave:** Aplicação, Processar Imagens, Google Cloud Platform (GCP), Armazenamento, Comunicação, Submissão de Imagens, Interface gRPC, Servidor, Máquina Virtual (VM), Aplicativo LandMarks, Desempenho, Escalabilidade e Disponibilidade.

# Abstract

This application has been developed with the aim of processing images to identify the presence of landmarks or famous places. It utilizes integrated services from the Google Cloud Platform (GCP) for storage, communication, and computation, providing a flexible and scalable solution in the cloud.

The system allows clients to submit image files for monument detection, storing them in Cloud Storage. After identifying the locations, the system retrieves a corresponding static map image of the identified monument's region. Additionally, the system has the ability to adapt to variations in workload, allowing for simultaneous scaling up or down of its image processing capacity.

The system's functionalities are exposed through a gRPC interface, enabling client applications to interact with the system efficiently. This interface offers operations for image submission, retrieval of monument detection results, and retrieval of information about processed photos.

To ensure system availability and load balancing, multiple replicas of the gRPC server are utilized, running on virtual machine (VM) instances within an instance group. Furthermore, replicas of the Landmarks App, responsible for monument identification in photos, are also deployed on VM instances. The system architecture leverages GCP services such as Cloud Storage, Firestore, Pub/Sub, Compute Engine, Cloud Functions, Vision API, and Static Maps API for storage, communication, computation, monument detection, and retrieval of static maps.

This report will present the implementation details of the application, including operation flows and its elasticity capabilities. The GCP services used, their integrations, and the APIs employed to achieve the desired functionalities will be discussed. Lastly, relevant implementation aspects and considerations necessary to ensure performance, scalability, and availability of the system in the cloud will be addressed.

**Keywords:** Application, Process Images, Google Cloud Platform (GCP), Storage, Communication, Image submission, gRCP interface, Server, Virtual Machine (VM), LandMarks App, Performance, Scability and Availability.

**Índice**

[Resumo iv](#_Toc136783300)

[Abstract vii](#_Toc136783301)

[Lista de Figuras x](#_Toc136783302)

[1. Introdução 1](#_Toc136783303)

[1.1 Organização do documento 1](#_Toc136783304)

[2. Formulação do Problema 3](#_Toc136783305)

[2.1 Problemas – Client 3](#_Toc136783306)

[2.1.1 IP Lookup 3](#_Toc136783307)

[2.1.2 Operações base 3](#_Toc136783308)

[2.1.3 Operações Server - Client 3](#_Toc136783309)

[2.2 Problemas – Server 3](#_Toc136783310)

[2.2.1 Google Storage 3](#_Toc136783311)

[2.2.2 Google Firebase 3](#_Toc136783312)

[2.2.3 Google Pub/Sub 3](#_Toc136783313)

[2.2.4 Landmarks 3](#_Toc136783314)

[2.3 Problemas – Landmarks 3](#_Toc136783315)

[2.3.1 Google Vision API 3](#_Toc136783316)

[2.3.2 Google Maps API 3](#_Toc136783317)

[2.3.2 Google Storage 3](#_Toc136783318)

[2.3.3 Google Firebase 3](#_Toc136783319)

[2.4 Problemas – Módulos adicionais 3](#_Toc136783320)

[3. Solução Proposta - Grandes Ideias 6](#_Toc136783321)

[3.1 Organização do Projeto 6](#_Toc136783322)

[3.2 Componentes do Projeto 8](#_Toc136783323)

[3.2.1 Cliente 8](#_Toc136783324)

[3.2.2 Contract 9](#_Toc136783325)

[3.2.3 Lookup 10](#_Toc136783326)

[3.2.4 Monitor 11](#_Toc136783327)

[3.2.5 Server 12](#_Toc136783328)

[4. Avaliação Experimental 13](#_Toc136783329)

[4.1 Análise Experimental 13](#_Toc136783330)

[5. Conclusões 15](#_Toc136783331)

[Referências 16](#_Toc136783332)

# Lista de Figuras

[Figura 1 - Componentes do projeto e respetivas interações 7](#_Toc136814492)

[Figura 2 - Estado inicial das VMs 14](file:///C:\Users\Utilizador\Downloads\Relatório_CN_G07(1).docx#_Toc136814493)

[Figura 3 - Estado atual do Monitor 14](file:///C:\Users\Utilizador\Downloads\Relatório_CN_G07(1).docx#_Toc136814494)

[Figura 4 - Pedido ao Servidor a partir do Cliente, este pedido submete uma imagem e retorna um identificador com o blob e o bucket 14](file:///C:\Users\Utilizador\Downloads\Relatório_CN_G07(1).docx#_Toc136814495)

[Figura 5 - Confirmação do upload da imagem 14](file:///C:\Users\Utilizador\Downloads\Relatório_CN_G07(1).docx#_Toc136814496)

[Figura 6 - Deteção/Inserção do landmarking da imagem anteriormente submetida 15](file:///C:\Users\Utilizador\Downloads\Relatório_CN_G07(1).docx#_Toc136814497)

[Figura 7 - Obtenção de informação sobre a imagem submetida 15](file:///C:\Users\Utilizador\Downloads\Relatório_CN_G07(1).docx#_Toc136814498)

[Figura 8 - Obtenção da foto satélite da localização da imagem submetida 16](file:///C:\Users\Utilizador\Downloads\Relatório_CN_G07(1).docx#_Toc136814499)

[Figura 9 - Query de monumentos em Lisboa com incerteza de 0.6 16](file:///C:\Users\Utilizador\Downloads\Relatório_CN_G07(1).docx#_Toc136814500)

# Introdução

Esta aplicação foi desenvolvida com o objetivo de processar imagens para identificar monumentos ou locais famosos. Utiliza serviços da Google Cloud Platform (GCP) para armazenamento, comunicação e computação, proporcionando uma solução escalável na nuvem. Os clientes podem enviar imagens para deteção de monumentos, armazenando-as no Cloud Storage. O sistema é capaz de se adaptar a variações de carga, aumentando ou diminuindo a capacidade de processamento de imagens conforme necessário. As funcionalidades são disponibilizadas através de uma interface gRPC, permitindo aos clientes interagir com o sistema de forma eficiente. Réplicas do servidor gRPC e do aplicativo Landmarks App são executadas em máquinas virtuais para garantir disponibilidade e load balancing. O sistema utiliza serviços como Cloud Storage, Firestore, Pub/Sub, Compute Engine, Cloud Functions, Vision API e Static Maps API da GCP.

## 1.1 Organização do documento

O documento está organizado de forma clara e estruturada, fornecendo uma visão abrangente sobre a aplicação. Inicia-se com um resumo conciso, que apresenta os objetivos do sistema e destaca a utilização dos serviços integrados da Google Cloud Platform. Além disso, é mencionada a arquitetura do sistema, com o uso de diferentes serviços da GCP e a distribuição de réplicas do servidor gRPC e do aplicativo Landmarks App em máquinas virtuais.

O relatório segue uma organização estruturada nas 3 bases: Formulação do Problema, Análise do Problema, Soluções Adotadas e por fim Avaliação Experimental.

Ao longo do relatório é explicado do fluxo de operações, demonstrando como as diferentes etapas são realizadas, desde a submissão das imagens até a obtenção dos resultados. Destaca-se também a elasticidade do sistema, com a possibilidade de ajustar a capacidade de processamento conforme a carga de trabalho. Em adição, são abordados os aspetos de implementação, descrevendo as APIs utilizadas para deteção de monumentos e obtenção de mapas estáticos. São fornecidos detalhes sobre o armazenamento das imagens no Cloud Storage e as informações relevantes no Firestore.

A organização do relatório facilita a compreensão da aplicação desenvolvida, permitindo uma análise completa das suas funcionalidades, arquitetura, elasticidade e aspetos de implementação.

# 2. Formulação do Problema

## 2.1 Problemas – Client

O componente "client" é responsável por interagir com os usuários ou aplicações cliente, fornecendo uma interface intuitiva para submissão de imagens, obtenção de resultados e informações sobre as fotos processadas. Ele utiliza a interface gRPC para comunicar com o servidor.

### 2.1.1 IP Lookup

O componente "lookup" desempenha um papel fundamental na disponibilidade e no load balancing do sistema. Ele é responsável por obter os endereços IP das réplicas do servidor gRPC, permitindo que o cliente conecte-se a uma instância disponível. Essa funcionalidade é implementada como uma Cloud Function, facilitando a atualização dos endereços IP em tempo real.

### 2.1.2 Operações base

No modulo de cliente, as operações base são getImageNames(1), submitImage(1), getInfo(1) e getLocations(1). Todas estas operações irão ser explicadas de forma mais detalhada ao longo do relatório.

## 2.2 Problemas – Server

O componente "server" é a parte central do sistema, que recebe as solicitações do cliente através da interface gRPC e coordena o processamento das imagens. Ele interage com os serviços da Google Cloud Platform, como Cloud Storage, Firestore, Pub/Sub, Compute Engine, Cloud Functions, Vision API e Static Maps API, para armazenar as imagens, identificar monumentos, obter mapas estáticos e armazenar informações relevantes. O servidor também lida com a distribuição de tarefas para as réplicas do aplicativo Landmarks App e retorna os resultados ao cliente.

A divisão do projeto em partes bem definidas facilita o desenvolvimento, a manutenção e a compreensão do projeto. Cada componente possui responsabilidades específicas possibilita uma solução robusta e escalável na nuvem.

### 2.2.1 Google Storage

As imagens enviadas pela cliente são guardadas no Google Storage, as imagens são Blob que são guardados num Bucket específico.

### 2.2.2 Google Firebase

As informações da imagem, como por exemplo a localização geográfica, nome do monumento e referencia do mapa estático, estas informações são armazenadas no Google Firebase numa determinada Collection em Documents, posteriormente utilizada pelo server.

### 2.2.3 Google Pub/Sub

O Google Pub/Sub é onde estabelece ligação de servidor com Landmarks, o servidor envia uma mensagem que contem informação da localização da imagem no Google Storage. O landmarks recebe essa mesma mensagem e processa-o.

## 2.3 Problemas – Landmarks

A aplicação Landmarks tem como objetivo detectar e processar informações sobre pontos de referência (landmarks) em imagens. Ela utiliza várias APIs do Google, incluindo a Google Vision API, Google Maps API, Google Storage e Google Firebase.

### 2.3.1 Google Vision API

A Google Vision API é usada para detetar pontos de referência em imagens. O problema principal relacionado a essa API é realizar a deteção precisa dos landmarks nas imagens fornecidas. Garantir a deteção correta e precisa dos landmarks é essencial para o funcionamento adequado da aplicação.

### 2.3.2 Google Maps API

A Google Maps API é utilizada para obter informações geográficas associadas aos landmarks detectados. A aplicação utiliza essa API para obter o endereço correspondente às coordenadas geográficas dos landmarks.

### 2.3.2 Google Storage

O Google Storage é utilizado para armazenar as imagens processadas e os mapas estáticos gerados. A manipulação correta das permissões de acesso aos objetos armazenados, a prevenção de perda de dados e a garantia de que as imagens e mapas estejam disponíveis para acesso rápido e eficiente pela aplicação são problemas que vão ser abordados ao longo do relatório.

### 2.3.3 Google Firebase

O Google Firebase é usado para armazenar as informações sobre os landmarks detetados, incluindo os dados sobre o nome, confiança, localização geográfica e endereço associado a cada landmark. O problema relacionado ao Firebase é garantir a correta persistência e recuperação dos dados, bem como a sincronização adequada entre os dispositivos e a aplicação. É fundamental que os dados sejam armazenados e recuperados de forma confiável e eficiente, garantindo a consistência das informações para os usuários da aplicação.

## 2.4 Problemas – Monitor

A parte "monitor" é responsável por monitorizar e gerir a elasticidade do sistema. Ela acompanha a carga de trabalho e realiza ajustes na quantidade de réplicas do servidor gRPC e do aplicativo Landmarks App, conforme necessário. Isso permite que o sistema aumente ou diminua sua capacidade de processamento de imagens em tempo real, garantindo eficiência e escalabilidade.

## 2.5 Problemas – Contract

A parte "contract" define o contrato da interface gRPC, estabelecendo as operações disponíveis e os parâmetros necessários para cada operação. Isso promove uma comunicação consistente entre o cliente e o servidor, garantindo que ambos estejam em conformidade com o contrato estabelecido.

# 3. Solução Proposta - Grandes Ideias

## 3.1 Organização do Projeto

A divisão do projeto em cinco partes distintas - client, contract, lookup, monitor e server - foi adotada para promover a modularidade, a clareza e a organização do projeto.

Uma imagem com texto, diagrama, file, captura de ecrã

Descrição gerada automaticamente

Figura 1 - Componentes do projeto e respetivas interações

## 3.2 Componentes do Projeto

### 3.2.1 Cliente

O package "*Client*" contém classes que implementam uma aplicação do lado do cliente para interagir com um serviço remoto usando a tecnologia gRPC (Google Remote Procedure Call).

A classe “*Client*” representa o ponto de entrada principal da aplicação do cliente. Esta estabelece uma conexão com o serviço remoto, configura o stub gRPC e fornece um menu de linha de comando para interagir com o serviço. As opções do menu incluem enviar uma imagem, obter informações da imagem, obter localizações da imagem, obter nomes de imagem e sair da aplicação. A entrada do utilizador é lida a partir da linha de comando e os métodos correspondentes são invocados com base na opção selecionada. Ainda nesta classe, é recebida uma lista de IPs provenientes do lookup que irão ser processados e utilizados na mesma.

A classe “*CustomStreamObserver*” é uma implementação personalizada da interface StreamObserver fornecida pelo gRPC. Esta classe foi implementada de forma genérica de modo a ser utilizada em todas as operações do cliente, lidando com as respostas recebidas do serviço remoto de forma assíncrona.

As opções disponíveis no menu incluem:

**Submit Image**: Permite ao utilizador enviar uma imagem para o serviço. O utilizador é solicitado a fornecer o caminho da imagem, e a imagem é lida e enviada para o servidor em blocos usando o StreamObserver do gRPC.

**Get Image Info**: Recupera informações da imagem do servidor com base no identificador fornecido. O utilizador é solicitado a fornecer o identificador, e as informações da imagem correspondente são obtidas usando o StreamObserver do gRPC.

**Get Locations**: Recupera a imagem de mapa associada ao identificador fornecido do servidor. O utilizador é solicitado a fornecer o identificador, e os dados da imagem são recebidos e salvos em blocos usando o StreamObserver do gRPC.

**Get Image Names**: Recupera uma lista de nomes de imagem que correspondem a critérios específicos do servidor. O utilizador é solicitado a fornecer uma cidade e um nível de certeza, e os nomes de imagem correspondentes são recebidos usando o StreamObserver do gRPC.

### 3.2.2 Contract

O objetivo desta definição de protocolo (proto) no package "Contract" é descrever os serviços e as mensagens usadas na comunicação entre o cliente e o servidor no contexto de uma aplicação de análise de imagens. Essa definição permite que o cliente e o servidor comuniquem de forma eficiente, trocando informações relevantes para realizar tarefas como enviar imagens, obter informações sobre imagens, recuperar localizações associadas a imagens e obter nomes de imagens com base em critérios específicos.

Aqui estão os principais componentes da definição do protocolo:

**Serviço VisionService**: Este serviço define quatro métodos RPC (Remote Procedure Call) que podem ser invocados pelo cliente para interagir com o servidor. Os métodos permitem enviar imagens, obter informações sobre imagens, recuperar localizações de imagens e obter nomes de imagens.

**Mensagens**: A definição do protocolo inclui várias mensagens usadas para trocar informações entre o cliente e o servidor. Algumas das mensagens importantes incluem:

**SearchRequest**: Uma mensagem usada para especificar critérios de pesquisa ao solicitar nomes de imagens.

**ImageNames**: Uma mensagem que contém uma lista de nomes de imagens como resposta à solicitação de nomes de imagens.

**Results**: Uma mensagem que contém uma lista de resultados, cada um contendo informações sobre uma imagem.

**Result**: Uma mensagem que representa as informações de uma imagem, incluindo seu nome, localização, certeza e endereço associado.

**Address**: Uma mensagem que representa o endereço associado a uma imagem, contendo informações de país e cidade.

**Localization**: Uma mensagem que representa a localização geográfica de uma imagem, especificada pelas coordenadas de latitude e longitude.

**Identifier**: Uma mensagem usada para identificar uma imagem com base em um identificador único.

**Blocks**: Uma mensagem que contém um bloco de dados de uma imagem, incluindo o nome da imagem e os bytes do bloco.

Essa definição do protocolo estabelece uma base sólida para a comunicação entre o cliente e o servidor, permitindo a troca de informações necessárias para realizar operações relacionadas à visão de imagens.

### 3.2.3 Lookup

O objetivo do package e class "Lookup" é implementar uma função HTTP que lista as máquinas virtuais em execução (VMs) num determinado projeto do Google Cloud Compute Engine. A função em específico é projetar, para posteriormente, ser implementada e executada num ambiente de Google Cloud function.

A classe Lookup implementa a interface HttpFunction e define o método service, que é chamado quando a função é acionada aquando um pedido HTTP. A função recebe parâmetros de consulta opcionais para especificar a zona e o grupo de instâncias para as quais deseja listar as VMs em execução.

Dentro do método service, a função realiza as seguintes etapas:

1. Obtém um objeto BufferedWriter para escrever a resposta HTTP.
2. Obtém os parâmetros de consulta da solicitação para determinar a zona e o grupo de instâncias desejados.
3. Regista as informações de log sobre a zona e o grupo de instâncias.
4. Escreve uma mensagem inicial na resposta HTTP.
5. Utiliza a biblioteca InstancesClient para conectar-se ao Compute Engine e listar todas as instâncias em execução no projeto e na zona especificados.
6. Para cada instância listada, verifica se está em execução e pertence ao grupo de instâncias especificado. Se sim, regista as informações de log e escreve detalhes da instância na resposta HTTP, incluindo o nome, o último horário de inicialização e o endereço IP público da instância.
7. Fecha a conexão com o cliente InstancesClient.

Portanto, a classe Lookup fornece uma função HTTP que pode ser usada para recuperar informações sobre as VMs em execução em um projeto específico do Google Cloud Compute Engine, permitindo a monitorização e gestão dessas VMs por meio de uma solicitação HTTP.

### 3.2.4 Monitor

O objetivo de "*Monitor*" é implementar uma função de plano de fundo (background *function*) para monitorar e ajustar automaticamente o número de instâncias de um grupo de instâncias no Google Cloud Compute Engine.

A classe *Monitor* implementa a interface *BackgroundFunction* e define o método *accept*, que é chamado quando a função é acionada para processar uma mensagem de entrada do *Pub/Sub*. A mensagem é recebida como um objeto *PSmessage*, contendo dados e atributos. O Monitor, para realizar uma alteração no número de instâncias concorrentes é necessário enviar uma mensagem Pub/Sub com os *key-atributes*:

instance-group -> instance-group-landmark

size -> 2

Dentro do método *accept*, a função realiza as seguintes etapas:

1. Obtém os atributos da mensagem para determinar a zona e o grupo de instâncias para os quais deseja monitorar e ajustar o número de instâncias. Caso nenhum atributo seja fornecido.
2. Obtém uma referência ao Firestore de forma a armazenar e recuperar informações de monitorização.
3. Recupera o objeto *Info* do documento específico no Firestore que contém informações sobre o número de instâncias atual, a contagem de chamadas e o último tempo registado.
4. Realiza cálculos com base nas informações recuperadas e nas configurações definidas (como os limiares e os limites máximo e mínimo de instâncias) para determinar se o número de instâncias deve ser ajustado.
5. Se o número de instâncias tiver de ser ajustado, utiliza a biblioteca *InstanceGroupManagersClient* para redimensionar o grupo de instâncias, chamando o método resizeAsync.
6. Atualiza o documento no Firestore com as novas informações de monitorização.

Além da classe Monitor, também são definidas as classes *Info* e *PSmessage*.

A classe *Info* representa os dados armazenados no Firestore e contém os campos *instances* (número de instâncias), time (último tempo registado) e *calls* (contagem de chamadas).

A classe *PSmessage* é usada para representar a mensagem de entrada do *Pub/Sub* e possui os campos data (dados da mensagem) e *atributes* (atributos da mensagem).

Em resumo, a classe *Monitor* implementa uma função de plano de fundo que monitoriza e ajusta automaticamente o número de instâncias de um grupo de instâncias no Google Cloud Compute Engine com base em critérios definidos, usando o Firestore para armazenar e recuperar informações de monitorização.

### 3.2.5 Server

O package “*Server”* contém classes que implementam uma aplicação do lado do Server para interagir com vários serviços remotos (clientes) usando a tecnologia gRPC (Google Remote Procedure Call).

A classe “*ServerApp”* é o “cérebro” desta operação. Este estabelece uma conexão com os serviços remotos, configura o stub gRPC seguindo as normas do Contrato que respeita as funções atribuídas no menu de linhas de comando disponibilizado aos Clientes. As opções do menu, tal como indicadas anteriormente, incluem: enviar uma imagem, obter informações da imagem, obter localizações da imagem, obter nomes de imagens e sair da aplicação. O servidor é responsável de receber pedidos de clientes e tratar de enviar uma resposta esperada baseada no contrato que assim lhes fora disponibilizado. Esta classe vai ser imediatamente executada no arranque da máquina virtual na Google Cloud.

A classe “*FirestoreOperation*” tem como responsabilidade o manuseamento de informações entre a Google Firestore e o Servidor corrente.

A classe “*PubSubOperation*” representa, evidentemente, o serviço PubSub do envio de mensagens em fila dos vários clientes para tópicos subscritos.

A classe “*StorageLocation*” é uma simples implementação que segue a estruturada enunciada com o necessário para envio de mensagens a publicar pelo PubSub. Isto é, cada objeto desta classe é um *holder* para um *blobname* e *bucketname* associado.

A classe “*StorageOperations”* trata de realizar a transmissão de informação entre o servidor e o Google Storage. Este trata principalmente de upload Blobs para os Buckets e de enviar Blobs dos Buckets para os Clientes.

A classe “*VisionResult*” é uma classe *holder* de informação para *Localization* e *Adress*. Ela tem como objetivo principal guardar a informação da biblioteca Google Cloud Vision cuja API tirámos partido.

A classe “*LandmarksApp*” é a classe responsável por detetar os monumentos nas imagens. Esta classe tira partido da classe *VisionResult* para organização de informação e de também de upload em forma de um blob para um bucket holder da localização no mapa de onde o Vision “pensa” que se encontra o monumento, juntamente com a informação das coordenadas e endereço.

# Avaliação Experimental

## Análise Experimental

Estado inicial do servidor:

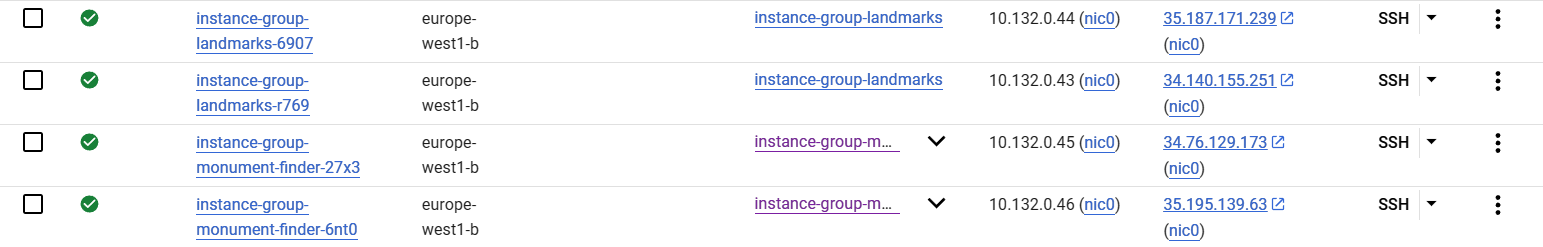


Figura - Estado inicial das VMs

Uma imagem com texto, Tipo de letra, file, número

Descrição gerada automaticamente

Figura - Estado atual do Monitor

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamente

Figura - Pedido ao Servidor a partir do Cliente, este pedido submete uma imagem e retorna um identificador com o blob e o bucket

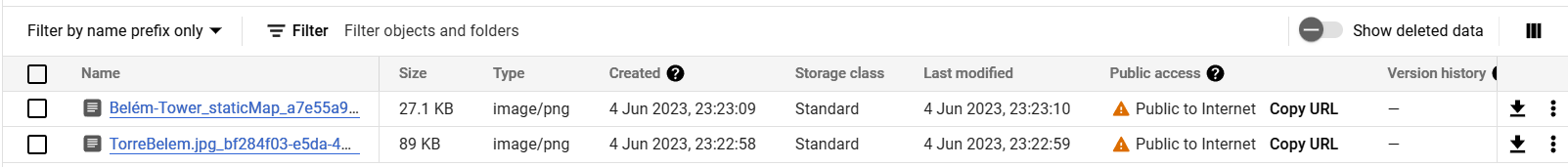


Figura - Confirmação do upload da imagem

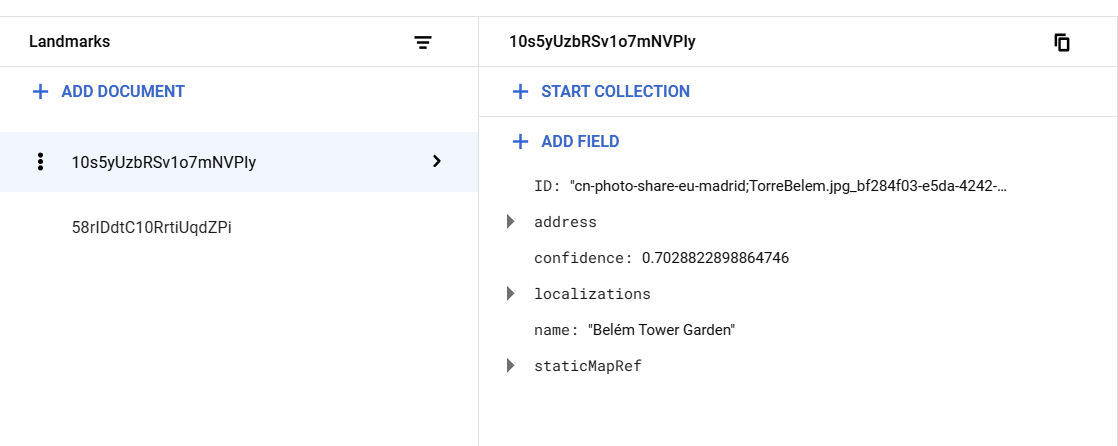


Figura - Deteção/Inserção do landmarking da imagem anteriormente submetida

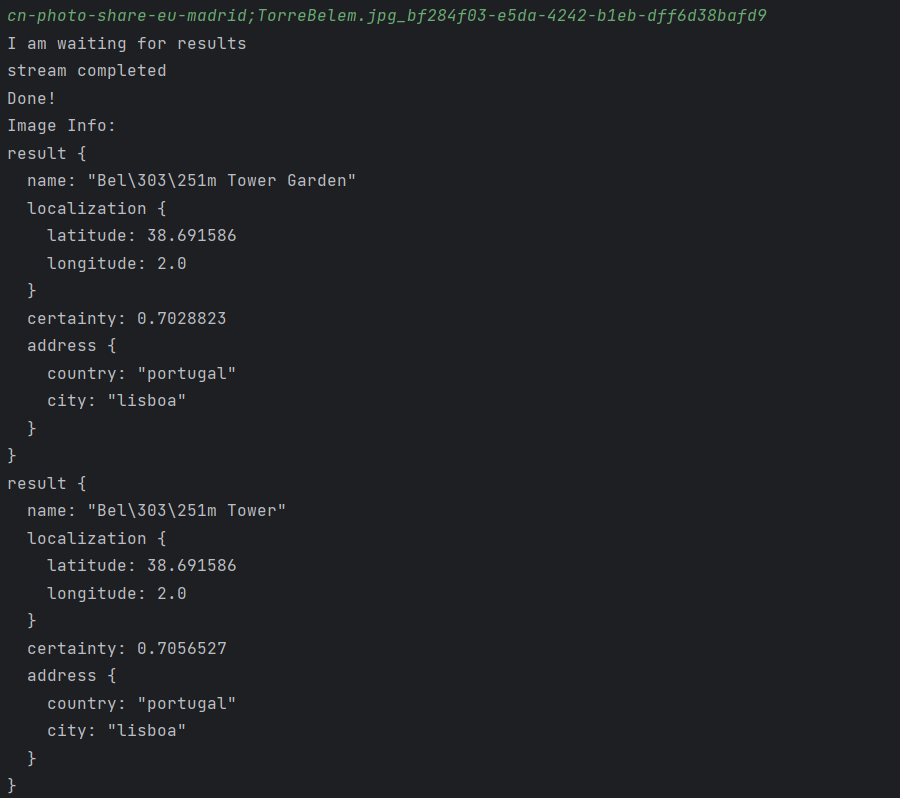


Figura - Obtenção de informação sobre a imagem submetida

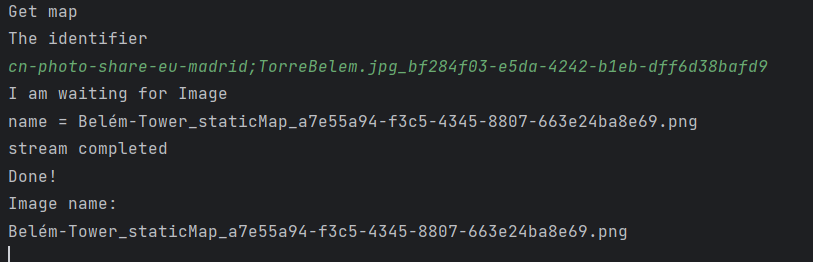


Figura - Obtenção da foto satélite da localização da imagem submetida

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - Query de monumentos em Lisboa com incerteza de 0.6

# Conclusões

Neste projeto conseguimos concluir que a implementação bem-sucedida. Durante o desenvolvimento do projeto, enfrentámos múltiplos desafios, como a configuração e integração de vários componentes, a resolução de problemas de conectividade e a otimização do desempenho do sistema.

Ao longo do projeto, deparamo-nos com alguns problemas de implementação e até mesmo, a própria compreensão do problema. Contudo, após algum tempo de análise e reflexão, com ajuda dos documentos e código fornecido pelo docente, conseguimos ultrapassar esses mesmo problemas e acabar com um projeto eficiente e eficaz, cumprindo as etapas deste trabalho.

Em resumo, o projeto CN2223-TF foi uma ótima experiência para reter conhecimento na utilização do Google Cloud Platform (GCP).

# Referências

|  |  |
| --- | --- |
| [1] | Documentação do docente e Moodle - Computação na Nuvem - LEIC - 2223SV |
| [2] | Documentação da Google Cloud - https://cloud.google.com/docs?hl=pt-br |