INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Licenciatura em Engenharia Informática e de Computadores

Semestre de Verão 2019/2020

CN-Text

Trabalho Final – G06

Tiago Pereira — Nº43592

Wilson Rúben Costa — Nº43593

Relatório do Trabalho Final

de

Computação na Nuvem

Professor Luís Assunção

20 junho 2020

ÍNDICE

1. Introdução

Este trabalho tem como foco principal realçar todas as capacidades de Computação na Nuvem adquiridas ao longo do semestre.

Com a pesquisa realizada no âmbito da fase inicial do projeto, ficou claro que a Plataforma de Cloud Pública da Google, oferece inúmeras ferramentas para os desenvolvedores, ficando definido, nesta fase, que só iriamos usar um pequeno conjunto das mesmas:

- Compute Engine
- Storage
- Firestore
- Pub/Sub

Com a realização de uma aplicação modular e baseada em trabalhos laboratoriais passados, os usuários do serviço criado pelo grupo, podem usufruir de uma experiência fluída e com possibilidade de criação de dois tipos de conta de utilizador (Free ou Premium) para melhor satisfazer as suas necessidades na tradução de imagens para texto.

1.1. Enquadramento

No panorama atual, em que o mundo da tecnologia caminha cada vez mais rápido, para a computação móvel e sem necessidade do peso financeiro e espacial que o alojamento próprio acarreta, a unidade curricular de Computação na Nuvem, oferece conhecimento teórico e prático sobre uma temática cada vez mais presente nas equipas de desenvolvimento de software.

Assim, foi possível evidenciar uma evolução nas ferramentas oferecidas aos programadores, ferramentas essas que começaram a ser desenvolvidas no século passado, tendo sofrido alterações significativas nos últimos anos.

Em suma, todos os alunos aumentaram o seu glossário de valências sendo estas, importantíssimas no contexto do mercado de trabalho atual.

1.2. Objetivos e Funcionalidades

O principal objetivo, e como o nome do projeto sugere, é a obtenção do texto presente em imagens de qualquer extensão e ainda com a opção de tradução do texto extraído.

Depois da autenticação, com sucesso, de um utilizador, este consegue fazer novos pedidos de obtenção de texto com a opção de tradução, ou não, e ainda aceder ao histórico de pedidos efetuados por ele e também os seus resultados finais.

O foco desta unidade curricular incidiu sobre os elementos de processamento de texto, deixando a responsabilidade do ensino de autenticação segura e viável para a unidade curricular de Segurança Informática.

1.3. Divisão de Tarefas e Responsabilidades

Para a criação de aplicação estável e com o intuito de providenciar a todos os elementos o grupo o contacto com todas as ferramentas usadas, ficou decidido que o aluno **Tiago Pereira** (43592) ficou encarregue da criação de uma aplicação cliente-servidor (local) de testes que despoletava o funcionamento de todos os sistemas e do desenvolvimento da aplicação de ORC.

O aluno **Wilson Costa** (43593) ficaria encarregue da criação de um contrato que refletisse as funcionalidades requeridas, da versão final do servidor, bem como do desenvolvimento da aplicação de Tradução.

Com o decorrer do projeto, o aluno **Wilson Costa** acumulou a responsabilidade da criação de um cliente que, respeitando o contrato, conseguiria comunicar com sucesso com o servidor remoto, enquanto que o aluno **Tiago Pereira** ficou responsável pela alocação das aplicações nas respetivas máquinas virtuais de Compute Engine e pela manipulação e afinação das configurações de todos os componentes remotos.

2. Solução Proposta

Para o desenvolvimento da solução do Projeto de tradução de texto é necessário utilizar um conjunto de técnicas e conhecimentos adquiridos ao longo da unidade curricular.

O Projeto trata um modelo Cliente-Servidor sendo possível comunicar entre ambos através da *framework Google Remote Procedure Call* (gRPC) e para que sejam apresentadas as funcionalidades pretendidas, será necessário implementar um conjunto de componentes. Esses componentes serão o Contrato, Cliente, Servidor, Aplicação OCR e Aplicação *Translator*.

Neste capítulo apresentam-se as soluções propostas para cada componente constituinte do Projeto, indicando uma análise inicial do mesmo e quais os passos a tomar em cada um deles.

2.1. Contrato

Para que a comunicação entre o cliente e o servidor seja bem-sucedida, é necessário que ambos implementem um contrato que descreva os pedidos que podem ser processados. Este contrato é definido com Protobuf, um método de serialização de dados, usando a linguagem *protocol buffer* e deverá conter funções que permitam a um utilizador iniciar ou terminar uma nova sessão bem como enviar ficheiros ou obter resultados desses mesmos ficheiros, que podem ser ou não traduzidos.

2.2. Cliente

Depois de definir um contrato, será necessário implementar uma aplicação cliente capaz de produzir uma *Graphical User Interface* (GUI) que demonstre ao utilizador as informações que pretenda. Esta GUI deve indicar ao utilizador que este insira os seus dados de acessos à aplicação, para que de seguida este possa enviar uma nova imagem que pretenda converter para que consiga consultar os seus dados textuais.

Deve também existir a hipótese de o utilizador poder indicar que pretende visualizar a resposta traduza para uma outra língua que não a original. Existe um conjunto de línguas possíveis de serem indicadas para tradução, como por exemplo: português, inglês, espanhol ou italiano.

2.3. Servidor

O servidor deve ser capaz de comunicar com diversos componentes para além do cliente. Primeiramente deve poder comunicar com uma base de dados que possa guardar informações de utilizadores, tarefas de processamento de imagem ou tarefas de tradução de texto. Essa base de dados irá ser não relacional pois não se pretende que haja associações com entidades, um aspeto que para este Projeto não é o adequado, visto que os dados que se pretendem guardar não estão relacionados entre si.

Deve também ser possível comunicar com um serviço que possa criar máquinas virtuais para os dois tipos de utilizadores existentes (free e premium), desta forma irá transmitir ou receber dados de aplicações que estejam alojadas de forma virtual num serviço.

É também necessário que este implemente serviços de transmissão de mensagens utilizando subscrições, para que haja uma maior escalabilidade do Projeto através de operações paralelas.

Finalmente será também essencial uma comunicação com um serviço que possa armazenar ficheiros de imagens, para mais tarde serem consumidos por outra aplicação.

Depois de existir uma relação entre todos estes componentes com o servidor, este irá criar um conjunto de funcionalidades que façam com que a aplicação Cliente envie ou obtenha os dados que pretenda.

2.4. Aplicação OCR

A aplicação OCR funciona como intermediária entre o Servidor e um serviço de conversão de imagens, em que delega as imagens indicadas pelo Servidor para o conversor. Para isso, deve ser capaz de comunicar com um serviço de armazenamento de ficheiros, para consumir o ficheiro de imagem que seja o pretendido. Depois disso, terá de indicar ao serviço de conversão a imagem textual para obter os seus dados, que por sua vez serão armazenados na base de dados não relacional.

Além destes procedimentos, terá de implementar técnicas de envio e receção de mensagens de forma paralela para poder processar diferentes pedidos de utilizadores de forma paralela.

2.5. Aplicação Translator

A aplicação de Translator irá funcionar de forma semelhante à aplicação OCR, porém trata pedidos de tradução de texto previamente convertido a partir de um ficheiro de imagem. Para isso deverá comunicar com um serviço externo de tradução de texto, para que depois armazene o resultado na base de dados não relacional.

3. Implementação

Neste capítulo é apresentada uma implementação da solução proposta, sendo indicado como cada componente do Projeto foi criado assim como a descrição da sua comunicação e as funções que desempenham.

A figura seguinte apresenta uma visão geral de toda a comunicação entre as partes que constituem o Projeto, indicando também o seu percurso para uma melhor perceção do fluxo de dados.

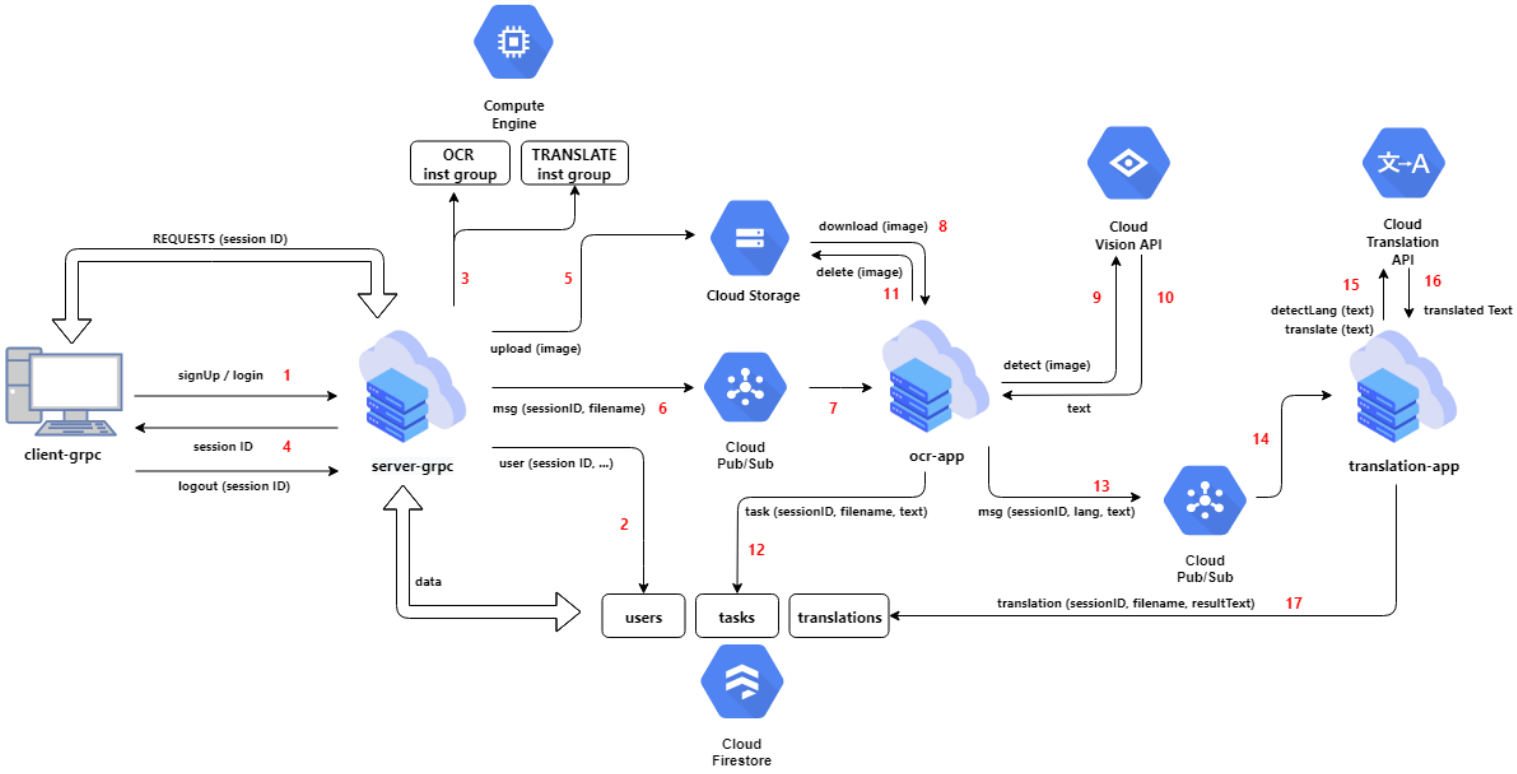

Figura 1 - Visão geral do fluxo de dados do Projeto

3.1. Descrição do Contrato

O contrato é a base de um serviço gRPC e é responsável por fornecer os *stubs* aos componentes que o irão utilizar, dando a possibilidade de estes comunicarem entre si.

Começa por descrever um serviço com os procedimentos que pretende que o Servidor e o Cliente implementem, são eles:

- signUp (User) returns (Session) → criar conta associada ao User indicado, retornando a Session que contém um ID.
- login (User) returns (Session) → iniciar sessão de um User já existente, retornando a Session que contém um ID.
- logout (Session) returns (Void) → terminar sessão de um utilizador que tenha a Session associada.
- uploadFile (File) returns (HashFile) → enviar um novo File que contém um ficheiro de imagem, retornando o HashFile com o seu nome gerado.
- getHashFiles (Session) returns (stream HashFile) → obter todos os HashFiles contendo ficheiros de imagem associados ao utilizador que contenha a Session.
- getTranslatedFiles (Session) returns (stream HashFile) → obter todos os HashFiles contendo ficheiros de tradução associados ao utilizador que contenha a Session.
- showResult (HashFile) returns (Result) → mostrar o Result de um determinado HashFile contendo o ficheiro de imagem.
- showTranslatedResult (HashFile) returns (Result) → mostrar o Result de um determinado HashFile contendo o ficheiro de tradução.

Verifica-se que foram usados dois tipos de chamadas: unária, onde o cliente envia um pedido e recebe uma resposta proveniente do servidor, utilizando um *ListenableFuture* que permite lidar com chamadas assíncronas de apenas um

elemento; *streaming* do servidor, onde o cliente envia um pedido e obtém uma sequência de respostas a partir de um *StreamObserver* dado pelo servidor.

O serviço implementado possui diferentes tipos de parâmetros (*messages*):

- User → utilizador com um nome, palavra-passe e tipo de conta;
- Session → sessão com um id, e tipo de conta;
- Result → resultado de uma conversão com o texto e o texto traduzido;
- File → ficheiro pretendido para conversão com o *array* de bytes dos seus dados, indicação de tradução, linguagem alvo, id de sessão e tipo de conta;
- HashFile → *hash* do ficheiro, com o seu nome e o id da sessão;
- Void → representação de tipo sem retorno.

Depois de serem descritos os procedimentos pretendidos, foi gerado o *package* do contrato do tipo *.jar* que contém os *stubs* e classes de serialização necessários para ser implementado pelo Cliente e Servidor.

3.2. Descrição do Servidor

O servidor é responsável por delegar funções aos diferentes serviços com que comunica e por enviar e obter dados provenientes do cliente.

Depois de implementar as funções do seu contrato, foi necessário instanciar cada serviço que será utilizado. Um deles será o Firestore [1], um serviço Google que fornece uma base de dados não relacional para guardar dados de um *user, task* ou *translation*.

Comunica também com o serviço Compute Engine [2], que providencia uma forma de gerir os grupos de instâncias associados a cada utilizador *premium*.

Para armazenar os ficheiros de imagem indicados pelo cliente utiliza o serviço Cloud Storage [3], para depois serem consumidos por outra aplicação.

O último serviço externo que utiliza é o serviço Cloud Pub/Sub [4], que facilita o envio e receção de mensagens de forma paralela usando tópicos e subscrições a esses tópicos. No essencial, o servidor irá utilizar este serviço para indicar qual o tipo de utilizador que está a aceder, para fazer com que este use as aplicações correspondentes (free ou premium).

A classe ServerGRPC é a classe principal do servidor e tem a função de criar as funcionalidades indicadas no contrato. Para criar um utilizador (**signUp**) é necessário utilizar uma instância do Firestore para que com os dados provenientes do Cliente (*request*) seja então criado. De seguida é necessário adicionar o utilizador à lista de utilizadores e, no caso de o utilizador ser *premium*, é necessário disponibilizar-lhe um novo grupo de instância. Para o caso em que o utilizador é *free*, irá utilizar a instância que é disponibilizada para todos os utilizadores *free*. Depois é retornado um *sessionID*, gerado com o hashCode da instância.

Para iniciar sessão de um utilizador (**login**), consulta-se a base de dados para verificar se o utilizador existe e se a palavra-passe que inseriu corresponde à

original. Como a funcionalidade anterior, terá de adicionar o utilizador à lista de utilizadores e retornar o *sessionID*.

O método **logout** apenas remove o utilizador da lista de utilizadores correntes.

Para enviar o ficheiro da imagem (**uploadFile**) começa-se por se obter os bytes do ficheiro da imagem. De seguida constrói-se o seu nome com o *sessionID* do utilizador concatenado com o *hashCode* dos bytes da imagem para que seja indicado na criação do ficheiro numa pasta local onde o servidor estará alojado.

Com o ficheiro criado inicia-se o envio do mesmo para o Storage do serviço Google, com o nome do *blob* a ser igual ao nome do ficheiro, para que de seguida se construa o mapa de atributos que irão ser enviados numa mensagem Pub/Sub para a aplicação OCR, sendo necessário verificar o tipo de conta para enviar para a aplicação correspondente (*free* ou *premium*). O conteúdo da mensagem será apenas o nome do ficheiro que a aplicação terá de descarregar para converter.

Assim que o envio da mensagem é terminado é ainda necessário notificar o cliente com o nome do ficheiro que foi enviado, para que este mais tarde possa verificar.

Para visualizar todos os nomes de ficheiros (**getHashFiles**), é necessário aceder à base de dados da coleção *task*, para que sejam retornados todos os ficheiros correspondentes a um determinado *sessionID*. Quando esse pedido for terminado, é necessário construir uma sequência (*stream*) de resultados para notificar ao Cliente todos os ficheiros que este inseriu. Para visualizar os ficheiros que contêm traduções (**getTranslatedFiles**) efetua-se o mesmo procedimento, desta vez acedendo à coleção *translation*.

A mostragem de resultados (**showResult**) é efetuada acedendo à base de dados da coleção *task*, obtendo o texto contido no documento especificado,

depois de verificado se o utilizador pode aceder ao seu conteúdo através do *sessionID*. Da mesma forma, para obter o texto traduzido (**showTranslationResult**), acede-se à base de dados da coleção *translation*, sabendo que o utilizador tem o mesmo *sessionID* do documento.

3.3. Descrição do Cliente

O cliente tem a função de intermediar a comunicação entre o usuário da aplicação, e o servidor alojado remotamente no serviço da Google, *Compute Engine*.

Para validar o seu correto funcionamento, bem como para avaliar de que forma era possível interagir com os restantes componentes da aplicação, foi criada uma aplicação cliente, do tipo consola, que realiza chamadas a funções presentes no contrato *proto*.

Com a utilização desta arquitetura, é facilmente criada outra aplicação cliente, com uma maior preocupação para o UI/UX, do que a apresentada.

Apenas com leituras e escritas na consola, foi possível coordenar quais as evocações de métodos a fazer, bem como apresentar de uma forma simples os resultados a pedidos efetuados.

O aspeto relevante da implementação assenta no facto de o usuário poder inserir uma imagem de qualquer extensão para efetuar o processamento, visto que, a transferência de dados com o servidor é feita ao nível dos bytes de não, ao nível de caminhos para ficheiros ou diretorias.

3.4. Descrição da Aplicação OCR

A aplicação OCR será responsável pelo envio dos ficheiros associados a cada utilizador para um serviço externo que converta os ficheiros de imagem em texto. Para isso deverá receber mensagens que indiquem os nomes dos ficheiros a converter.

O nome que receba será o nome a consultar num serviço que contenha os ficheiros de imagens, para que assim obtenha o ficheiro correspondente ao nome e o envie ao serviço de conversão.

Para a troca de mensagens entre o servidor e o presente módulo, utilizou-se o *Pub/Sub*.

O conteúdo das mensagens consistia no nome do *blob* presente no *storage* que representa a fonte de informação, mas também na inclusão do sessionId do utilizador em causa, para uma melhor identificação do remetente do pedido de processamento.

Com a informação dos bytes da imagem, a aplicação OCR, encaminha a tarefa para a *Vision API*, para o reconhecimento textual na imagem.

Tendo os resultados, é feito o upload para uma *collection* no *Firebase* chamada *pendingTasks*.

Se houver necessidade de tradução, é utlizado novamente o serviço de Pub/Sub para a notificação de outro serviço, a Aplicação *Translation*, que estão pedidos pendentes. Na mensagem, é inserido o texto resultante da *Vision API*, a linguagem destino e fonte, e também o *sessionId* do utilizador.

3.5. Descrição da Aplicação Translation

A aplicação *Translation* é muito similar à aplicação *OCR*, está desenhada da mesma forma, comunica e interage com os restantes componentes da rede de processamento da mesma forma, mas em vez de ser baseada na *Vision API*, tem como subsistema principal, a *Translation API*.

Esta aplicação recebe toda a informação necessária no corpo das mensagens de notificação não sendo necessário recorrer a outro serviço para obtenção de dados para processar.

Chegando a mensagem de notificação, são encaminhadas as linhas de texto, para a Translation API, sendo obrigatório o indicação da lingua destino e fonte, para uma tradução mais certeira e eficaz.

Com o receção dos resultados, estes são encaminhados para o Firestore, via upload de um documento para uma colecção previamente configurada, denominada *pendingTranslation*.

4. Conclusão

Neste trabalho abordámos o assunto da computação remota, atualmente diversamente utilizada na maioria dos dispositivos ativos.

Com a aplicação dos conhecimentos adquiridos ao longo do leccionamento das aulas teórico-práticas no projeto final, e a utilização de um contrato para permitir a simulação de um padrão de desenho Facade, ficou em aberto, a criação de mais clientes que facilmente respeitam as exigências no contrato acima referido, aumentando assim, as possibilidades de interação com a aplicação desenvolvida.

Cumprimos todos os objetivos (obrigatórios e opcionais) a que nos propusemos, uma vez que, conseguimos fazer uma eficaz divisão de tarefas e uma pesquisa inicial sólida.

Por fim, sublinhar a importância do aprofundamento deste tema, que, apontam as previsões e sondagens, ganhará uma importância abismal no contexto da engenharia informática.

Referências

DOCUMENTAÇÃO DOS SERVIÇOS GOOGLE

FIRESTORE	https://firebase.google.com/docs/firestore
COMPUTE ENGINE	https://cloud.google.com/compute/docs
STORAGE	https://cloud.google.com/storage/docs
PUB/SUB	https://cloud.google.com/pubsub/docs
VISION	https://cloud.google.com/vision/docs
TRANSLATION	https://cloud.google.com/translate/docs