**Unidade Curricular: Integração de Sistemas**

**Professor: Prof. Doutor Ricardo Baptista**

**Trabalho realizado por: Ivo Nunes nº 20232582**

**RELATÓRIO EUROMILREGISTER**

**Contexto**

O projeto EuromilRegister é uma implementação em Python de um sistema baseado em gRPC que permite aos utilizadores registar apostas para o jogo Euromilhões. O projeto demonstra como definir, implementar e testar um serviço gRPC usando Protocol Buffers para comunicação entre um cliente e um servidor.

**Estrutura do Projeto**

O projecto está organizado da seguinte forma:

A screenshot of a computer

Description automatically generated

**Definição dos buffers de protocolo**

O ficheiro euromil.proto define o contrato de comunicação entre o cliente e o servidor. Inclui a interface do serviço e as estruturas das mensagens.

A screenshot of a computer program

Description automatically generated

Serviço: A Euromil define um único método RPC RegistarEuroMil.

Mensagens:

RegisterRequest: Contém a chave e o checkid da aposta.

RegisterReply: Contém uma mensagem de sucesso ou insucesso.

**Geração do código**

Utilizando a biblioteca grpcio-tools, o ficheiro euromil.proto é compilado em dois ficheiros Python:

* euromil\_pb2.py: Define as classes de mensagem para o pedido e para a resposta.
* euromil\_pb2\_grpc.py: Define o stub do serviço e as classes base.

O comando para gerar estes ficheiros é:

* python -m grpc\_tools.protoc -I. --python\_out=. --grpc\_python\_out=. euromil.proto

**Implementação do servidor gRPC**

O servidor é implementado em server.py usando a API Python gRPC. A classe EuromilService estende a classe EuromilServicer gerada a partir do ficheiro Protocol Buffers.

Principais caraterísticas:

- Implementa o método RegisterEuroMil.

- Verifica se a entrada é válida (tanto key como checkid devem ser fornecidos).

- Devolve uma mensagem de êxito ou de erro.

Exemplo:

A screen shot of a computer program

Description automatically generated

O servidor está à escuta na porta 50051 e trata dos pedidos recebidos.

**Implementação do cliente gRPC**

O cliente é implementado em client.py e interage com o servidor criando um RegisterRequest e invocando o método RegisterEuroMil.

Exemplo:

A screen shot of a computer code

Description automatically generated

O cliente liga-se ao servidor em localhost:50051, envia um pedido e imprime a resposta.

**Dependências**

O projeto utiliza as seguintes dependências, listadas em requirements.txt:

grpcio==1.57.0

grpcio-tools==1.57.0

Para instalar as dependências:

Pip/Pip3 install -r requirements.txt

**Iniciar o Projeto**

* Iniciar servidor:
  + Python/Python3 server.py
* Num novo terminal, iniciar o cliente:

Output Esperado:

Do cliente:

Response from server: Bet registered with key: 12345 and check ID: abcde.

Do servidor:

Server is running on port 50051

**Conclusão**

O projeto EuromilRegister demonstra efetivamente a implementação de um sistema baseado em gRPC para registar apostas para o jogo Euromilhões usando Python.

O projeto mostra a utilização de Protocol Buffers (Protobuf) para definir o serviço e as estruturas de mensagens, o que proporciona um método claro e eficiente de comunicação entre o cliente e o servidor. Ao utilizar o gRPC e as suas ferramentas associadas, como a biblioteca grpcio-tools, o projeto destaca caraterísticas fundamentais como a definição de serviços, a geração de código e as chamadas de procedimento remoto (RPC).

A implementação do servidor e do cliente gRPC é simples, com o servidor a tratar os pedidos e a devolver as respostas adequadas com base na validação dos dados introduzidos.

A utilização do gRPC assegura uma comunicação eficiente e de baixa latência, enquanto a integração do Protobuf garante a segurança dos tipos e assegura que os dados são serializados num formato binário compacto, que é mais eficiente do que os formatos típicos baseados em texto, como o JSON.

O projeto EuromilRegister também fornece uma base para futuros desenvolvimentos, tais como a adição de medidas de segurança como a autenticação, o tratamento de erros e a potencial escalabilidade para acomodar um maior número de utilizadores. Com a estrutura modular e a separação clara entre a definição do serviço, o cliente e o código do servidor, este projeto serve como um bom ponto de partida para o desenvolvimento de aplicações gRPC mais complexas e prontas para a produção.

No geral, este projeto enfatiza as vantagens práticas de usar gRPC e Protocol Buffers, particularmente em cenários que requerem alto desempenho e troca de dados eficiente, e ilustra como estas ferramentas podem ser usadas eficazmente em aplicações baseadas em Python.