Programación de Objetos Distribuidos
Trabajo Práctico
gRPC - Communication Patterns

Ejercicio 1

Indicar los errores y/o mejoras que aplican en los siguientes fragmentos de definiciones de servicios .proto:

1.1

```
message SearchRequest {
 string query = 0;
 uint32 pageNumber = 1;
 uint32 resultPerPage = 2;
}
```

Salvo el ENUM que requiere el numero = 0, en los mensajes NO se permite = 0

1.2

```
message DoubleMessage {
 double value = 1;
}

service CalculatorService {
 rpc Add(DoubleMessage, DoubleMessage) returns (DoubleResponse);
 rpc Substract(DoubleMessage, DoubleMessage) returns (DoubleResponse);
 rpc Random() returns (DoubleMessage);
}
```

Podrian usar el wrapper de Google

MAL -> gRPC solo puede recibir UN mensaje y devolver UN mensaje

Creo que no se puede dejar vario -> usar Empty de Google

1.3

```
service FlightService {
 rpc FlightStatus(FlightStatusRequest) returns (FlightStatusResponse);
}

message FlightStatusRequest {
 string flightCode = 1;
}

enum FlightStatus {
 ON_TIME = 0;
 DELAYED = 1;
 CANCELLED = 2;
}

message FlightStatusResponse {
 FlightStatus flightStatus = 1;
```

MAL, El 0 es para el UNKNOWN y no respeta convencion de nombres (CREO)

```
}
```

Ejercicio 2

Se desea ofrecer un servicio gRPC que cuente los siguientes métodos:

- **`ping`**: operación sin parámetros que contesta inmediatamente con la cadena: “pong”
- **`time`**: operación sin parámetros que retorna la hora del servidor.
- **`echo`**: operación que recibe una cadena e inmediatamente la retorna.
- **`hello`**: operación que recibe una cadena (pensando que es un nombre) y retorna el texto “hello <nombre>”.
- **`fortune`**: operación sin parámetros que retorna una cadena de texto, seleccionando la misma aleatoriamente de una lista de textos que recibe durante la construcción el servant (los textos que son frases tipo fortune cookies) pueden estar “cableados” en el código de la clase servidor).

Para ello se pide implementar el contrato del servicio en un archivo .proto, un Servant que responda a los métodos del servicio, un Server que publique el Servant anterior y un Client que consuma el servicio publicado utilizando un *stub* bloqueante.