Formulário - GrpcServer

```
rpc Op1 (Request) returns (Response);

rpc Op2 (Request) returns (stream Response);

rpc Op3 (stream Request) returns (Response);

rpc Op4 (stream Request) returns (stream Response);
```

```
// imports

public class GrpcServer {

    private static int SERVER_PORT = 8000;

    public static void main(String[] args) {
        try {
            if (args.length > 0) SERVER_PORT = Integer.parseInt(args[0]);

            io.grpc.Server svc = ServerBuilder.forPort(SERVER_PORT)
                    .addService(new NameProcessor())
                    .build();

            svc.start();
            System.out.println("Server started on port " + SERVER_PORT);

            svc.awaitTermination();
        } catch (Exception ex) {
            System.err.println("An error occurred: " + ex.getMessage());
        }
    }
}
```

```
// imports

public class NameProcessor extends NameServiceGrpc.NameServiceImplBase {
    /* Variável de ambiente com chave
    * GOOGLE_APPLICATION_CREDENTIALS=<pathname do ficheiro json com chave>
    */

    GoogleCredentials credentials = GoogleCredentials.getApplicationDefault();

    FirestoreOptions options = FirestoreOptions
        .newBuilder()
        .setDatabaseId("db-name")
        .setCredentials(credentials)
        .build();
    Firestore db = options.getService();

    @Override
    public void op1(Request request, StreamObserver<Response> responseObserver) {
        // process request
```

```
        responseObserver.onNext(Response.newBuilder().build());
        responseObserver.onCompleted();
    }

    @Override
    public void op2(Request request, StreamObserver<Response> responseObserver) {
        // process request
        responseObserver.onNext(Response.newBuilder().build());
        responseObserver.onCompleted();
    }

    @Override
    public StreamObserver<Request> op3(StreamObserver<Response> responseObserver) {
        return new StreamObserver<Request>() {
            @Override
            public void onNext(Request request) {
                // process request
            }

            @Override
            public void onError(Throwable t) {
                // handle error
            }

            @Override
            public void onCompleted() {
                responseObserver.onNext(Response.newBuilder().build());
                responseObserver.onCompleted();
            }
        };
    }

    @Override
    public StreamObserver<Request> op4(StreamObserver<Response> responseObserver) {
        return new StreamObserver<Request>() {
            @Override
            public void onNext(Request request) {
                // process request
                responseObserver.onNext(Response.newBuilder().build());
            }

            @Override
            public void onError(Throwable t) {
                // handle error
            }

            @Override
            public void onCompleted() {
                responseObserver.onCompleted();
            }
        };
    }
}
```