Lab 3

The goal of this lab is to implement *remote invocation* in *Ballerina* using gRPC. You are strongly encouraged to use a git repository in your practical.

Problem 1

Create a gRPC implementation of a client and a server using a remove invocation to execute the following functionalities:

1. Create a new user;
2. View an existing user;
3. Update a field in an existing user;
4. Delete an existing user.

For the user creation operation, the sever will expect an object represented as follows:

```
{
    username: "billy",
    lastname: "Merlot",
    firstname: "William",
    email: "William.Merlot@adventurers.biz"
}
```

It returns a json object `{userid: "billy"}` when the creation operation is successful.
As for the view and delete operations, the sever expects a json object with the *userid* and returns the complete user object in the first case and a response of successful deletion in the second or an error when one occurs. Finally, for the update operation, the server expects a json object containing the (unchanged userid) and any of the other fields of the user object expect the username. When the operation completes a success or error message is returned.

Steps

To implement this protocol, you will take the following steps:

1. create a folder and write the *protocol buffers* file corresponding to the remote invocation
2. generate the client and the service corresponding to the protocol buffers
3. finalise the implementation of both
4. execute them

An example of the protocol buffers is given below:

```
syntax = "proto3";

service users {
    rpc create_user(CreateRequest) returns (CreateResponse);
}

message CreateRequest {
    string username = 1;
    string lastname = 2;
    string firstname = 3;
    string email = 4;
}

message CreateResponse {
    string userid = 1;
}
```

The command to generate the client or the service is as follows: `bal grpc --input /path/to/proto --mode client|service --output .`. Depending of the component being generated, you will choose *client* or *service* for the mode. It is recommended that you create separate projects for each component. The output will then point to the path of each project. When the implementation is completed, you will execute first the service and the client.

The samples below show a partial implementation of both the client and the service.

```
import ballerina/io;

usersClient ep = check new ("http://localhost:9090");


public function main() returns error? {
    io:println("about to send an RPC request to server");

       json res = check ep -> create_user({
        username: "tommy",
        firstname: "Thomas",
        lastname: "Picketty",
        email: "tpicketty@lanouvellegauche.org"
        });

        io:println(res);
}
```

```
import ballerina/io;
import ballerina/grpc;

listener grpc:Listener ep = new (9090);

@grpc:ServiceDescriptor {descriptor: ROOT_DESCRIPTOR, descMap: getDescriptorMap()}
service "users" on ep {

    remote function create_user(CreateRequest value) returns CreateResponse|error {
        io:println("received an RPC request from a client...");
        return {userid: value.username};
    }
}
```

Test the protocol for the *create_user* operation and complete the rest of the remote operations.