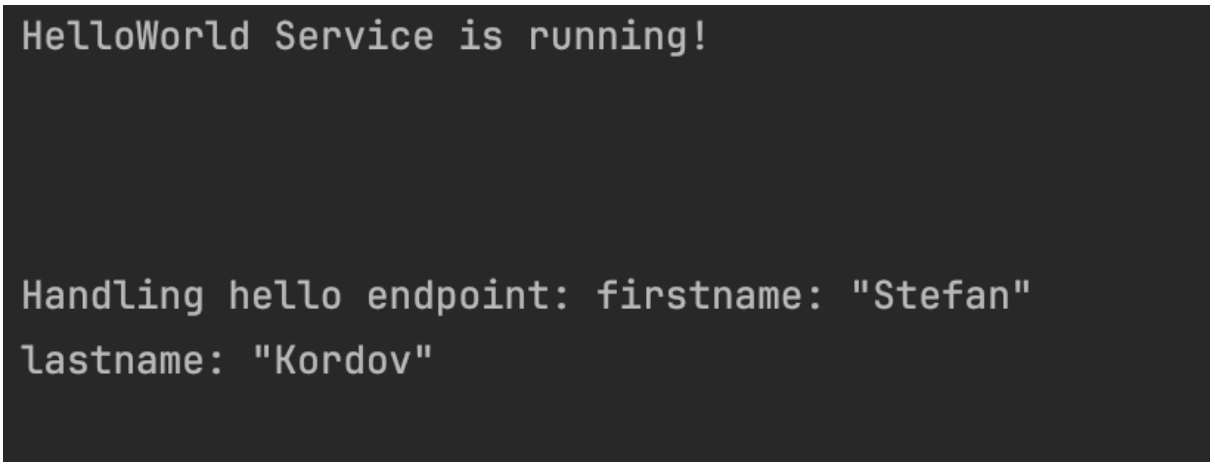
MidEng 7.2 gRPC Framework (Grundlagen plus Vertiefung)

First i forked the Repository of our Teacher from the following Link:

https://github.com/ThomasMicheler/DEZSYS_GK_HELLOWORLD_GRPC.git

Then all i had to do was change the Name in the build.gradle:

Now when i run the Gradle Task i can see the following output:



GKV

First i changed the `.proto` File:

```
syntax = "proto3";
```

```
service HelloWorldService {
  rpc hello(HelloRequest) returns (HelloResponse) {}
}

message HelloRequest {
  string firstname = 1;
  string lastname = 2;
}

message HelloResponse {
  string text = 1;
}
message ProductData {
  string productId = 1;
  string productName = 2;
  string productCategory = 3;
  string productAmount = 4;
  string productUnit = 5;
}

message WarehouseData {
  string warehouseID = 1;
  string warehouseName = 2;
  string timestamp = 3;
  string warehouseCountry = 4;
  string warehouseCity = 5;
  string address = 6;
  repeated ProductData productData = 7;
}

service WarehouseService {
  rpc GetWarehouseData (WarehouseRequest) returns (WarehouseData) {}
}

message WarehouseRequest {
  string warehouseID = 1;
  string warehouseName = 2;
  string timestamp = 3;
```

```
  string warehouseCountry = 4;
  string warehouseCity = 5;
  string address = 6;
  repeated ProductData productData = 7;
}
```

The Server stayed almost the same, i even made it do less:

```
import io.grpc.Server;
import io.grpc.ServerBuilder;

public class WarehouseServer {
    public static void main(String[] args) throws Exception {
        Server server = ServerBuilder.forPort(50051)
                .addService(new WarehouseServiceImpl())
                .build();
        server.start();
        System.out.println("Warehouse Service running");
        server.awaitTermination();
    }
}
```

Then the Client which i changed like this:

```
import io.grpc.ManagedChannel;
import io.grpc.ManagedChannelBuilder;

public class WarehouseClient {
    public static void main(String[] args) {
        ManagedChannel channel = ManagedChannelBuilder.forAddress("loca
lhost", 50051)
                .usePlaintext()
                .build();

        WarehouseServiceGrpc.WarehouseServiceBlockingStub stub =
                WarehouseServiceGrpc.newBlockingStub(channel);

        Hello.WarehouseRequest req = Hello.WarehouseRequest.newBuilder()
```

```
            .setWarehouseID("1001")
            .setWarehouseName("stefan kordov")
            .setTimestamp("2025-12-02")
            .setWarehouseCountry("AT")
            .setWarehouseCity("Vienna")
            .setAddress("wexstrasse 19")
            .addProductData(
                  Hello.ProductData.newBuilder()
                        .setProductId("1")
                        .setProductName("mario")
                        .setProductCategory("human")
                        .setProductAmount("65")
                        .setProductUnit("kg")
            )
            .build();


      Hello.WarehouseData data = stub.getWarehouseData(req);

      System.out.println(data);

      channel.shutdown();
   }
}
```

Then i made a WarehouseServiceImpl Class like this:

```
import io.grpc.stub.StreamObserver;

public class WarehouseServiceImpl extends WarehouseServiceGrpc.Wareh
ouseServiceImplBase {

   @Override
   public void getWarehouseData(Hello.WarehouseRequest request,
                        StreamObserver<Hello.WarehouseData> responseObse
rver) {

      System.out.println("warehouseID = " + request.getWarehouseID());
```

```
        System.out.println("warehouseName: " + request.getWarehouseName
());
        System.out.println("timestamp: " + request.getTimestamp());
        System.out.println("warehouseCountry: " + request.getWarehouseCou
ntry());
        System.out.println("warehouseCity: " + request.getWarehouseCity());
        System.out.println("address: " + request.getAddress());

        for (Hello.ProductData p : request.getProductDataList()) {
            System.out.println("productId: " + p.getProductId());
            System.out.println("productName: " + p.getProductName());
            System.out.println("productCategory: " + p.getProductCategory());
            System.out.println("productAmount: " + p.getProductAmount());
            System.out.println("productUnit: " + p.getProductUnit());
        }

        Hello.WarehouseData data = Hello.WarehouseData.newBuilder()
                .setWarehouseID(request.getWarehouseID())
                .setWarehouseName(request.getWarehouseName())
                .setTimestamp(request.getTimestamp())
                .setWarehouseCountry(request.getWarehouseCountry())
                .setWarehouseCity(request.getWarehouseCity())
                .setAddress(request.getAddress())
                .addAllProductData(request.getProductDataList())
                .build();

        responseObserver.onNext(data);
        responseObserver.onCompleted();
    }

}
```

Python

First i ran the 3 commands in the `README.md`

Then i made this python File, the same as my Java Class:

```
import grpc
import hello_pb2
import hello_pb2_grpc


def main():
    channel = grpc.insecure_channel("localhost:50051")
    stub = hello_pb2_grpc.WarehouseServiceStub(channel)

    req = hello_pb2.WarehouseRequest(
        warehouseID="1001",
        warehouseName="stefan kordov",
        timestamp="2025-12-02",
        warehouseCountry="AT",
        warehouseCity="Vienna",
        address="wexstrasse 19",
        productData=[
            hello_pb2.ProductData(
                productId="1",
                productName="mario",
                productCategory="human",
                productAmount="65",
                productUnit="kg"
            )
        ]
    )
    try:
        data = stub.GetWarehouseData(req)
        print(data)

    except grpc.RpcError as e:
        print("error")

    channel.close()
```

```
if __name__ == "__main__":
    main()
```

Fragen

1. What is gRPC and why does it work across languages and platforms?

gRPC is a system that lets programs talk to each other, and it works everywhere because it uses shared rules (proto files) that every language can understand.

2. Describe the RPC life cycle starting with the RPC client.

The client sends a request to the server, the server does the work, and then the server sends a reply back to the client.

3. Describe the workflow of Protocol Buffers.

You write your data format in a .proto file, the compiler generates code, and your program uses that code to send and receive data.

4. What are the benefits of using protocol buffers?

They are fast, small, and easy to use across different programming languages.

5. When is the use of protocol buffers not recommended?

They are not good when humans need to read or edit the data directly.

6. List 3 different data types that can be used with protocol buffers.

String, int32, bool.