DEZSYS_GK72_DATAWAREHOUSE_GRPC

Mario Milkov 4DHIT

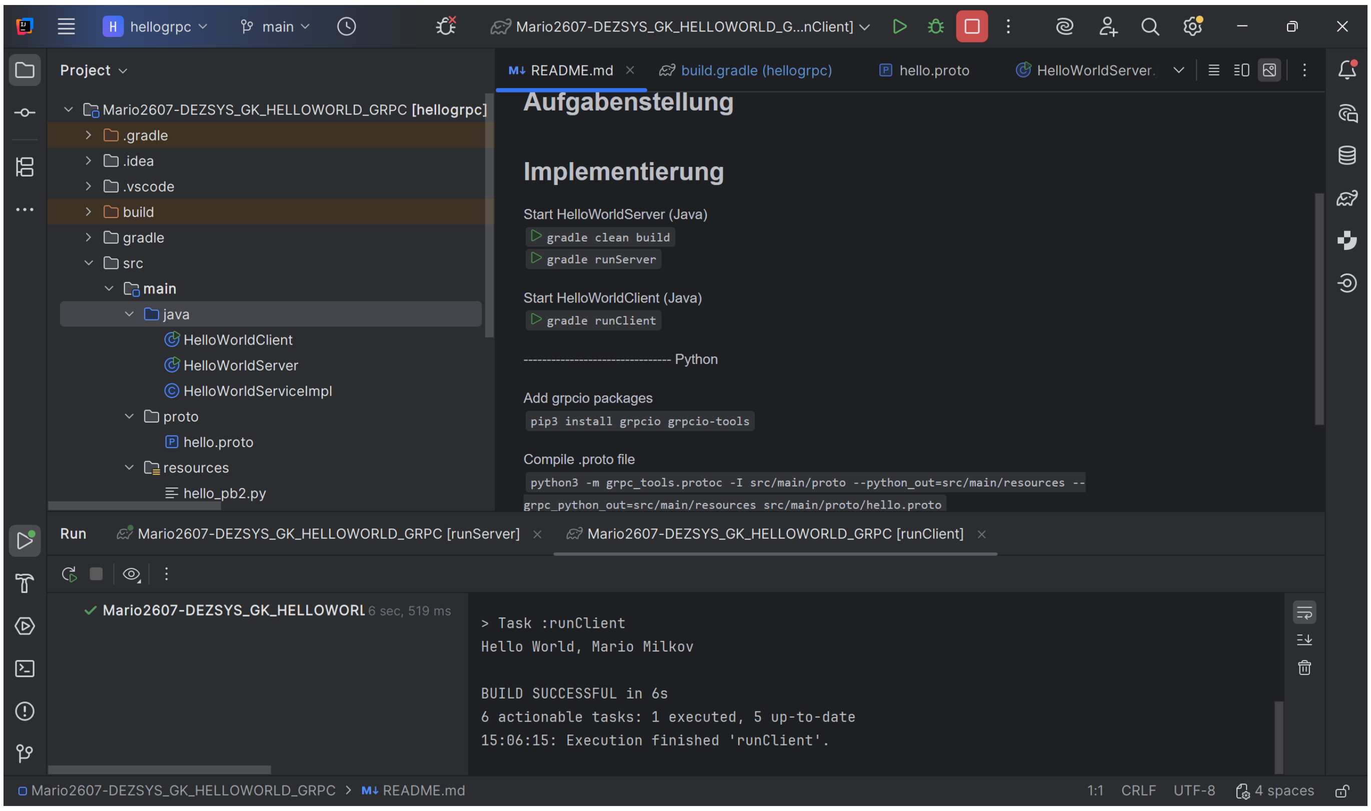
Grundlagen

Answer the questions below about the gRPC framework

- What is gRPC and why does it work accross languages and platforms?
  - gRPC is a high-performance open-source RPC framework by Google that lets programs communicate directly across systems. It works across languages and platforms because it uses HTTP/2, Protocol Buffers, and provides official libraries for many languages.
- Describe the RPC life cycle starting with the RPC client?
  - The client calls a local stub method, which sends a request to the server over HTTP/2; the server receives, processes it, and sends back a response; the client stub then returns the result to the caller.
- Describe the workflow of Protocol Buffers?
  - Protocol Buffers work by writing a `.proto` file, generating code from it, and then using the generated classes to serialize (encode) and deserialize (decode) data. Both client and server use the same `.proto` file so they exchange data in a compact, structured format.
- What are the benefits of using protocol buffers?
  - Protocol Buffers are very fast, use little network bandwidth, and create small, efficient messages. They are also strongly typed, easy to version, and work in many programming languages.
- When is the use of protocol not recommended?
  - They are not recommended when you need human-readable data (like JSON or XML). Also not ideal if your system does not need a strict schema or if you work mainly in environments that don't use binary formats.
- List 3 different data types that can be used with protocol buffers?
  - `string`
  - `int32`
  - `bool`

Demo Project

First i forked the project from our teacher and then proceeded with the following steps from the task. For the demo project I had to do all the steps from the `README.md` file and after that I just had to change my name in the `build.gradle` and in the `HelloWorldClient.java`.


hellogrpc
main
Mario2607-DEZSYS_GK_HELLOWORLD_G...nClient]
Project
README.md
build.gradle (hellogrpc)
hello.proto
HelloWorldServer.
Mario2607-DEZSYS_GK_HELLOWORLD_GRPC [hellogrpc]
.gradle
.idea
.vscode
build
gradle
src
main
java
HelloWorldClient
HelloWorldServer
HelloWorldServiceImpl
proto
hello.proto
resources
hello_pb2.py
Run
Mario2607-DEZSYS_GK_HELLOWORLD_GRPC [runServer]
Mario2607-DEZSYS_GK_HELLOWORLD_GRPC [runClient]
Mario2607-DEZSYS_GK_HELLOWORL 6 sec, 519 ms
Mario2607-DEZSYS_GK_HELLOWORLD_GRPC > README.md
1:1 CRLF UTF-8 4 spaces

Aufgabenstellung

Implementierung

Start HelloWorldServer (Java)

```
gradle clean build
gradle runServer
```

Start HelloWorldClient (Java)

```
gradle runClient
```

-------------------------------- Python

Add grpcio packages

```
pip3 install grpcio grpcio-tools
```

Compile .proto file

```
python3 -m grpc_tools.protoc -I src/main/proto --python_out=src/main/resources --grpc_python_out=src/main/resources src/main/proto/hello.proto
```

```
> Task :runClient
Hello World, Mario Milkov

BUILD SUCCESSFUL in 6s
6 actionable tasks: 1 executed, 5 up-to-date
15:06:15: Execution finished 'runClient'.
```

We can see very clear that in the console that the demo project worked!

Erweiterte Grundlagen

Customize the service so that a simple DataWarehouse record (see exercise MidEng 7.1) can be transferred

To complete this task, first i had to change the `hello.proto` file:

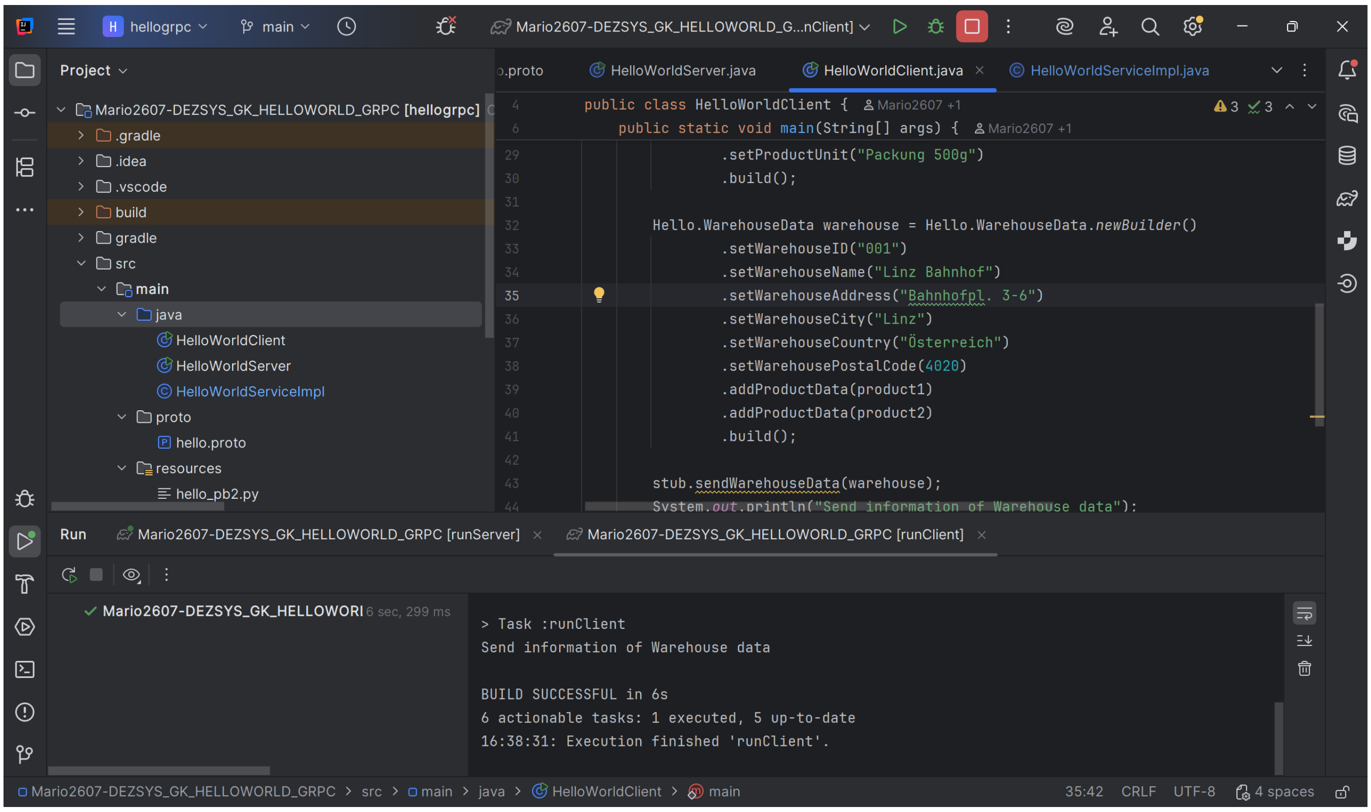
```
syntax = "proto3";

service HelloWorldService {
  rpc hello(HelloRequest) returns (HelloResponse) {}
  rpc sendWarehouseData (WarehouseData) returns (WarehouseResponse);
}

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

message HelloResponse {
  string text = 1;
}

message ProductData {
  string productId = 1;
  string productName = 2;
  string productCategory = 3;
  int32 productQuantity = 4;
  string productUnit = 5;
}

message WarehouseData {
  string warehouseID = 1;
  string warehouseName = 2;
  string warehouseCountry = 4;
  string warehouseCity = 5;
  string warehouseAddress = 6;
  repeated ProductData productData = 8;
}

message WarehouseResponse {
  WarehouseData data = 1;
}
```

The proto file defines what data the client and server can send to each other. It includes two functions and simple data structures like WarehouseData and ProductData that describe what information is transferred.

After that I changed the `HelloWorldClient.java` file:

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

public class HelloWorldClient {

    public static void main(String[] args) {

        String firstname = args.length > 0 ? args[0] : "Max";
        String lastname  = args.length > 1 ? args[1] : "Mustermann";

        ManagedChannel channel = ManagedChannelBuilder.forAddress("localhost", 50051)
                .usePlaintext()
                .build();

        HelloWorldServiceGrpc.HelloWorldServiceBlockingStub stub =
HelloWorldServiceGrpc.newBlockingStub(channel);
        Hello.ProductData product1 = Hello.ProductData.newBuilder()
                .setProductId("00-443175")
                .setProductName("Bio Orangensaft Sonne")
                .setProductCategory("Getraenk")
                .setProductQuantity(1500)
                .setProductUnit("Packung 1L")
                .build();

        Hello.ProductData product2 = Hello.ProductData.newBuilder()
                .setProductId("02-341867")
                .setProductName("Milka Tafel")
                .setProductCategory("Sueßigkeit")
                .setProductQuantity(1000)
                .setProductUnit("Packung 500g")
                .build();

        Hello.WarehouseData warehouse = Hello.WarehouseData.newBuilder()
                .setWarehouseID("001")
                .setWarehouseName("Linz Bahnhof")
                .setWarehouseAddress("Bahnhofpl. 3-6")
                .setWarehouseCity("Linz")
                .setWarehouseCountry("Österreich")
                .setWarehousePostalCode(4020)
                .addProductData(product1)
                .addProductData(product2)
                .build();

        stub.sendWarehouseData(warehouse);
        System.out.println("Send information of Warehouse data");
        channel.shutdown();

    }

}
```

Here the client creates all warehouse data and product data and sends it to the server using the sendWarehouseData function.



```java
public class HelloWorldClient {
    public static void main(String[] args) {
                .setProductUnit("Packung 500g")
                .build();

        Hello.WarehouseData warehouse = Hello.WarehouseData.newBuilder()
                .setWarehouseID("001")
                .setWarehouseName("Linz Bahnhof")
                .setWarehouseAddress("Bahnhofpl. 3-6")
                .setWarehouseCity("Linz")
                .setWarehouseCountry("Österreich")
                .setWarehousePostalCode(4020)
                .addProductData(product1)
                .addProductData(product2)
                .build();

        stub.sendWarehouseData(warehouse);
        System.out.println("Send information of Warehouse data");
```

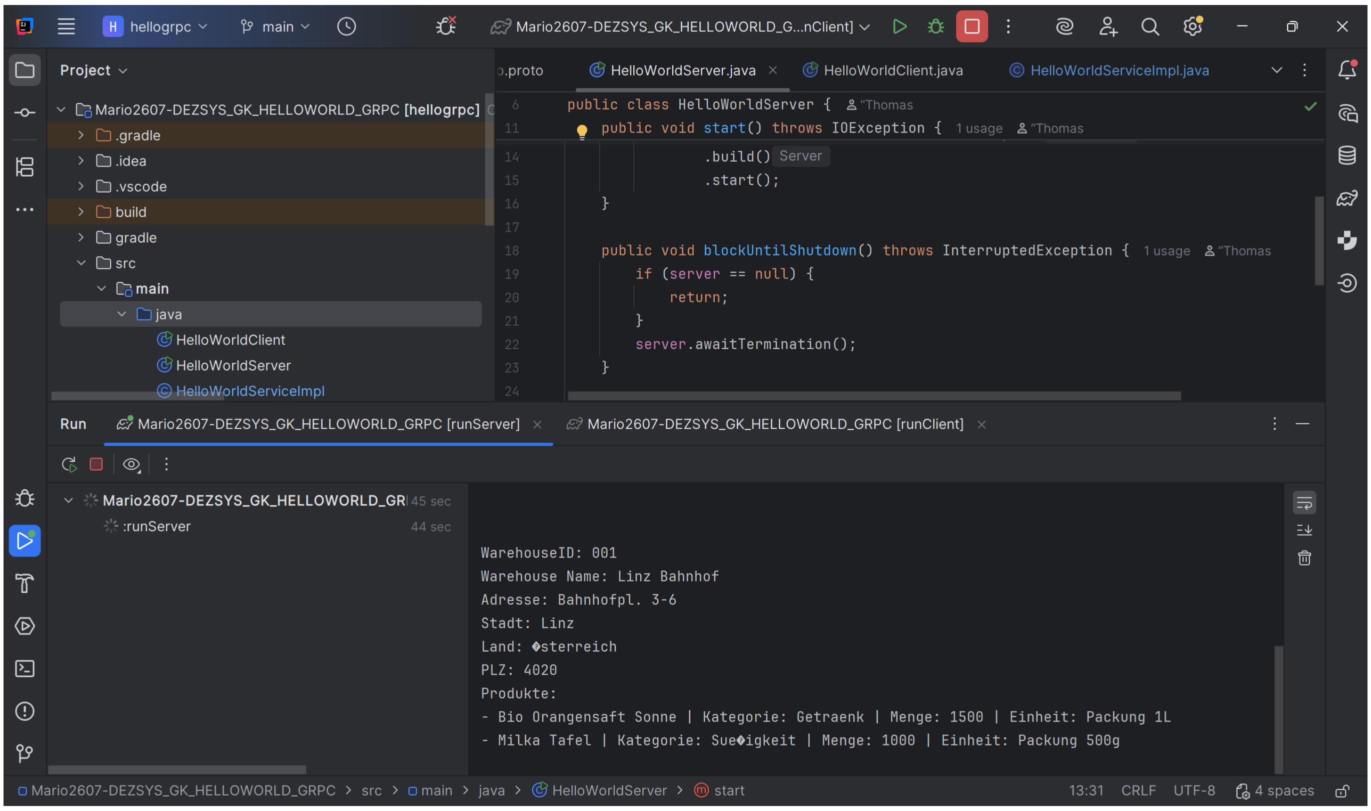
```
> Task :runClient
Send information of Warehouse data

BUILD SUCCESSFUL in 6s
6 actionable tasks: 1 executed, 5 up-to-date
16:38:31: Execution finished 'runClient'.
```

hellogrpc
main
Mario2607-DEZSYS_GK_HELLOWORLD_G...nClient]
Project
.proto
HelloWorldServer.java
HelloWorldClient.java
HelloWorldServiceImpl.java
Mario2607-DEZSYS_GK_HELLOWORLD_GRPC [hellogrpc]
.gradle
.idea
.vscode
build
gradle
src
main
java
HelloWorldClient
HelloWorldServer
HelloWorldServiceImpl
proto
hello.proto
resources
hello_pb2.py
Run
Mario2607-DEZSYS_GK_HELLOWORLD_GRPC [runServer]
Mario2607-DEZSYS_GK_HELLOWORLD_GRPC [runClient]
Mario2607-DEZSYS_GK_HELLOWORI 6 sec, 299 ms
Mario2607-DEZSYS_GK_HELLOWORLD_GRPC > src > main > java > HelloWorldClient > main
35:42 CRLF UTF-8 4 spaces

And lastly I had to change the `HelloWorldServiceImpl.java` file:

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

public class HelloWorldServiceImpl extends
HelloWorldServiceGrpc.HelloWorldServiceImplBase {

    @Override
    public void hello( Hello.HelloRequest request, StreamObserver<Hello.HelloResponse>
responseObserver) {

        System.out.println("Handling hello endpoint: " + request.toString());

        String text = "Hello World, " + request.getFirstname() + " " +
request.getLastname();
        Hello.HelloResponse response =
Hello.HelloResponse.newBuilder().setText(text).build();

        responseObserver.onNext(response);
        responseObserver.onCompleted();

    }

    @Override
    public void sendWarehouseData(Hello.WarehouseData request,
StreamObserver<Hello.WarehouseResponse> responseObserver) {

        System.out.println("WarehouseID: " + request.getWarehouseID());
        System.out.println("Warehouse Name: " + request.getWarehouseName());
        System.out.println("Adresse: " + request.getWarehouseAddress());
        System.out.println("Stadt: " + request.getWarehouseCity());
        System.out.println("Land: " + request.getWarehouseCountry());
        System.out.println("PLZ: " + request.getWarehousePostalCode());
        System.out.println("Produkte:");

        for (Hello.ProductData p : request.getProductDataList()) {
            System.out.println("- " + p.getProductName() + " | Kategorie: " +
p.getProductCategory() + " | Menge: " + p.getProductQuantity() + " | Einheit: " +
p.getProductUnit());
        }
        Hello.WarehouseResponse response =
Hello.WarehouseResponse.newBuilder().build();

        responseObserver.onNext(response);
        responseObserver.onCompleted();
    }
}
```

I had to do it so that the server receives the complete warehouse data from the client and prints all information to the console.


```
public class HelloWorldServer {
    public void start() throws IOException {
                .build()
                .start();
    }

    public void blockUntilShutdown() throws InterruptedException {
        if (server == null) {
            return;
        }
        server.awaitTermination();
    }
```

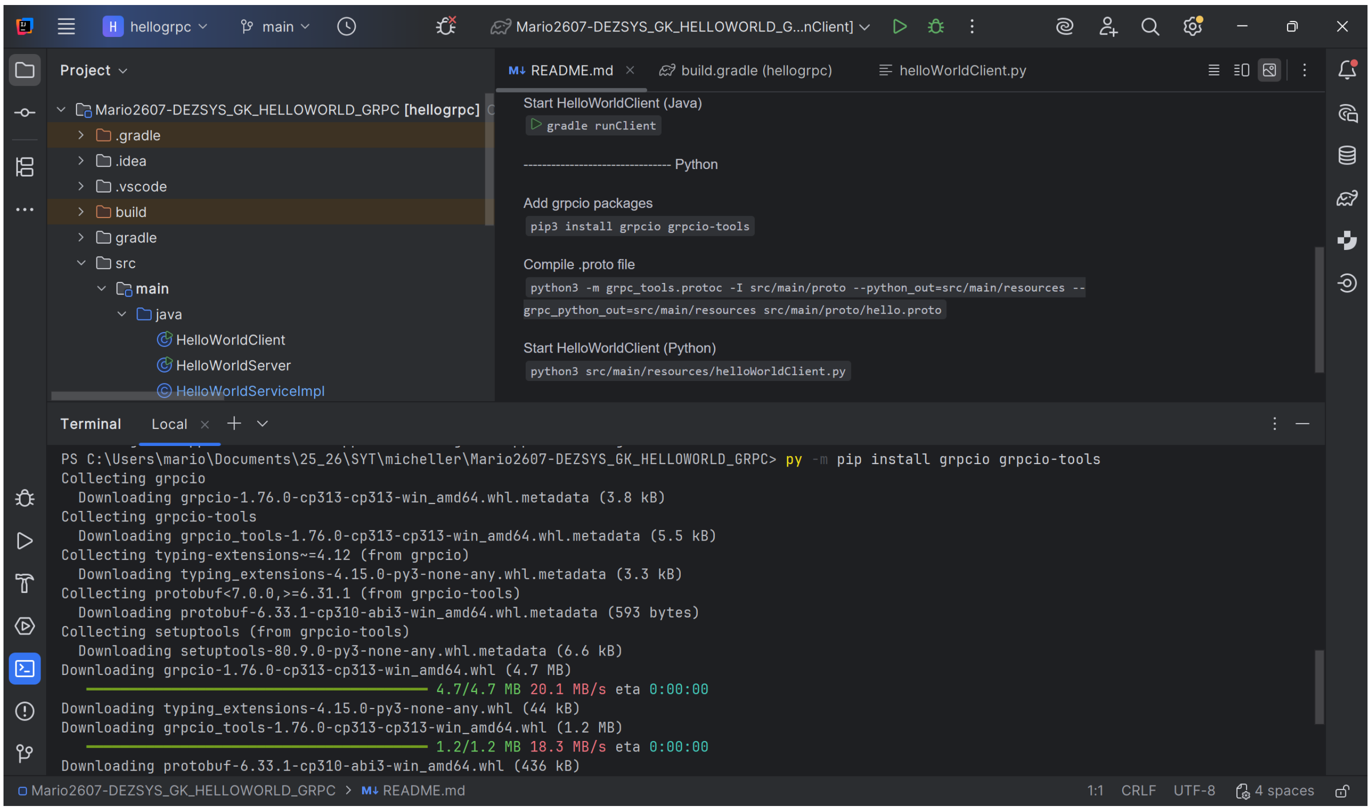
```
WarehouseID: 001
Warehouse Name: Linz Bahnhof
Adresse: Bahnhofpl. 3-6
Stadt: Linz
Land: �sterreich
PLZ: 4020
Produkte:
- Bio Orangensaft Sonne | Kategorie: Getraenk | Menge: 1500 | Einheit: Packung 1L
- Milka Tafel | Kategorie: Sue�igkeit | Menge: 1000 | Einheit: Packung 500g
```

hellogrpc
main
Mario2607-DEZSYS_GK_HELLOWORLD_G...nClient]
Project
HelloWorldServer.java
HelloWorldClient.java
HelloWorldServiceImpl.java
Mario2607-DEZSYS_GK_HELLOWORLD_GRPC [hellogrpc]
.gradle
.idea
.vscode
build
gradle
src
main
java
HelloWorldClient
HelloWorldServer
HelloWorldServiceImpl
Run
Mario2607-DEZSYS_GK_HELLOWORLD_GRPC [runServer]
Mario2607-DEZSYS_GK_HELLOWORLD_GRPC [runClient]
:runServer 44 sec
Mario2607-DEZSYS_GK_HELLOWORLD_GRPC > src > main > java > HelloWorldServer > start
13:31 CRLF UTF-8 4 spaces

And we can see that i works!

Vertiefung

I chose python for my second language and for that i had to run these commands: `py -m pip install grpcio grpcio-tools`, `py -m grpc_tools.protoc -I src/main/proto --python_out=src/main/resources --grpc_python_out=src/main/resources src/main/proto/hello.proto`



Start HelloWorldClient (Java)

```
gradle runClient
```

-------------------------------- Python

Add grpcio packages

```
pip3 install grpcio grpcio-tools
```

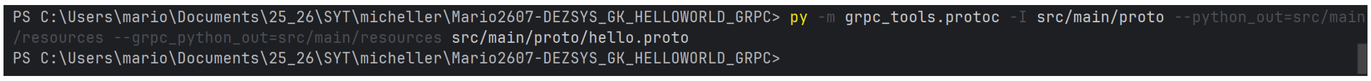
Compile .proto file

```
python3 -m grpc_tools.protoc -I src/main/proto --python_out=src/main/resources --
grpc_python_out=src/main/resources src/main/proto/hello.proto
```

Start HelloWorldClient (Python)

```
python3 src/main/resources/helloWorldClient.py
```

```
PS C:\Users\mario\Documents\25_26\SYT\micheller\Mario2607-DEZSYS_GK_HELLOWORLD_GRPC> py -m pip install grpcio grpcio-tools
Collecting grpcio
  Downloading grpcio-1.76.0-cp313-cp313-win_amd64.whl.metadata (3.8 kB)
Collecting grpcio-tools
  Downloading grpcio_tools-1.76.0-cp313-cp313-win_amd64.whl.metadata (5.5 kB)
Collecting typing-extensions~=4.12 (from grpcio)
  Downloading typing_extensions-4.15.0-py3-none-any.whl.metadata (3.3 kB)
Collecting protobuf<7.0.0,>=6.31.1 (from grpcio-tools)
  Downloading protobuf-6.33.1-cp310-abi3-win_amd64.whl.metadata (593 bytes)
Collecting setuptools (from grpcio-tools)
  Downloading setuptools-80.9.0-py3-none-any.whl.metadata (6.6 kB)
Downloading grpcio-1.76.0-cp313-cp313-win_amd64.whl (4.7 MB)
   ---------------------------------------- 4.7/4.7 MB 20.1 MB/s eta 0:00:00
Downloading typing_extensions-4.15.0-py3-none-any.whl (44 kB)
Downloading grpcio_tools-1.76.0-cp313-cp313-win_amd64.whl (1.2 MB)
   ---------------------------------------- 1.2/1.2 MB 18.3 MB/s eta 0:00:00
Downloading protobuf-6.33.1-cp310-abi3-win_amd64.whl (436 kB)
```



```
PS C:\Users\mario\Documents\25_26\SYT\micheller\Mario2607-DEZSYS_GK_HELLOWORLD_GRPC> py -m grpc_tools.protoc -I src/main/proto --python_out=src/main
/resources --grpc_python_out=src/main/resources src/main/proto/hello.proto
PS C:\Users\mario\Documents\25_26\SYT\micheller\Mario2607-DEZSYS_GK_HELLOWORLD_GRPC>
```

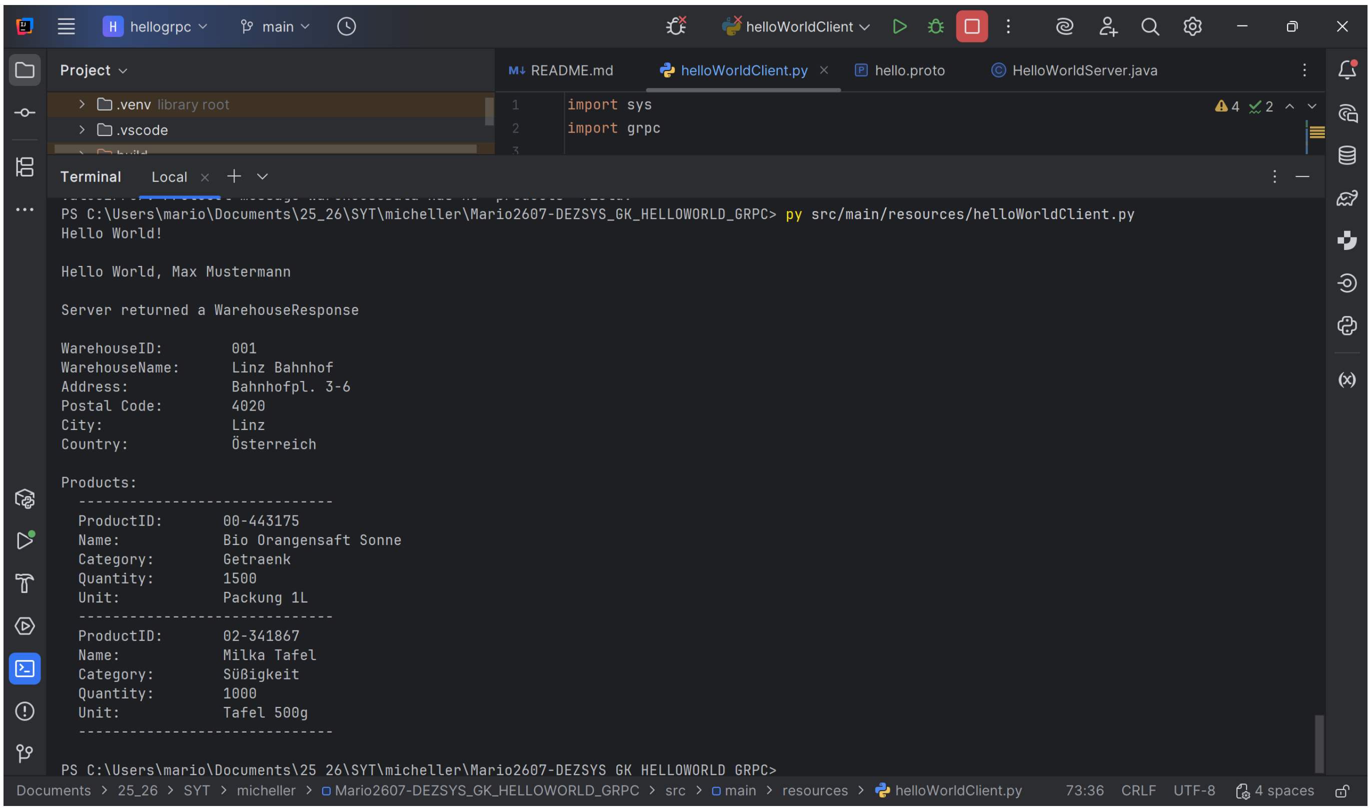
Here is the `helloWorldClient.py` file:

```python
import sys
import grpc

import hello_pb2
import hello_pb2_grpc


def main():
    print("Hello World!")
    firstname = sys.argv[1] if len(sys.argv) > 1 else "Max"
    lastname = sys.argv[2] if len(sys.argv) > 2 else "Mustermann"

    # Connect to server
    with grpc.insecure_channel("localhost:50051") as channel:
        stub = hello_pb2_grpc.HelloWorldServiceStub(channel)

        # ---- ORIGINAL HELLO ----
        request = hello_pb2.HelloRequest(firstname=firstname, lastname=lastname)
        response = stub.hello(request)
        print()
        print(response.text)
        print()

        # ---- NEW: PRODUCTS ----
        p1 = hello_pb2.ProductData(
            productId="00-443175",
            productName="Bio Orangensaft Sonne",
            productCategory="Getraenk",
            productQuantity=1500,
            productUnit="Packung 1L"
        )

        p2 = hello_pb2.ProductData(
            productId="02-341867",
            productName="Milka Tafel",
            productCategory="Süßigkeit",
            productQuantity=1000,
            productUnit="Tafel 500g"
        )

        # ---- NEW: WAREHOUSE ----
        warehouse_data = hello_pb2.WarehouseData(
            warehouseID="001",
            warehouseName="Linz Bahnhof",
            warehouseAddress="Bahnhofpl. 3-6",
            warehousePostalCode=4020,
            warehouseCity="Linz",
            warehouseCountry="Österreich",
            productData=[p1, p2]
        )
        # ---- SEND RPC ----
        stub.sendWarehouseData(warehouse_data)
        print("Server returned a WarehouseResponse")
        print()

        # ---- FULL ATTRIBUTE OUTPUT ----
        print("WarehouseID:       ", warehouse_data.warehouseID)
        print("WarehouseName:     ", warehouse_data.warehouseName)
        print("Address:           ", warehouse_data.warehouseAddress)
        print("Postal Code:       ", warehouse_data.warehousePostalCode)
        print("City:              ", warehouse_data.warehouseCity)
        print("Country:           ", warehouse_data.warehouseCountry)
        print()

        print("Products:")
        for product in warehouse_data.productData:
            print("  ------------------------------")
            print("  ProductID:      ", product.productId)
            print("  Name:           ", product.productName)
            print("  Category:       ", product.productCategory)
            print("  Quantity:       ", product.productQuantity)
            print("  Unit:           ", product.productUnit)
        print("  ------------------------------\n")


if __name__ == "__main__":
    main()()t("  ------------------------------\n")


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

This Python client first sends a hello request, then creates full warehouse data and product data and sends it to the gRPC server. After sending, it prints all data on the client side and shows the server's confirmation message.

After implementing the python file I just had to run this command:

```
py src/main/resources/helloWorldClient.py
```


hellogrpc
main
helloWorldClient
Project
.venv library root
.vscode
README.md
helloWorldClient.py
hello.proto
HelloWorldServer.java
Terminal
Local
Documents > 25_26 > SYT > micheller > Mario2607-DEZSYS_GK_HELLOWORLD_GRPC > src > main > resources > helloWorldClient.py
73:36 CRLF UTF-8 4 spaces

```
import sys
import grpc
```

```
PS C:\Users\mario\Documents\25_26\SYT\micheller\Mario2607-DEZSYS_GK_HELLOWORLD_GRPC> py src/main/resources/helloWorldClient.py
Hello World!

Hello World, Max Mustermann

Server returned a WarehouseResponse

WarehouseID:        001
WarehouseName:      Linz Bahnhof
Address:            Bahnhofpl. 3-6
Postal Code:        4020
City:               Linz
Country:            Österreich

Products:
  ----------------------------
  ProductID:        00-443175
  Name:             Bio Orangensaft Sonne
  Category:         Getraenk
  Quantity:         1500
  Unit:             Packung 1L
  ----------------------------
  ProductID:        02-341867
  Name:             Milka Tafel
  Category:         Süßigkeit
  Quantity:         1000
  Unit:             Tafel 500g
  ----------------------------

PS C:\Users\mario\Documents\25_26\SYT\micheller\Mario2607-DEZSYS_GK_HELLOWORLD_GRPC>
```

And after many tries and problems with python i finally managed to solve it and now we can see the correct and successfull output!!!