Article
Guillaume Rongier · Sep 30, 2022 8m read

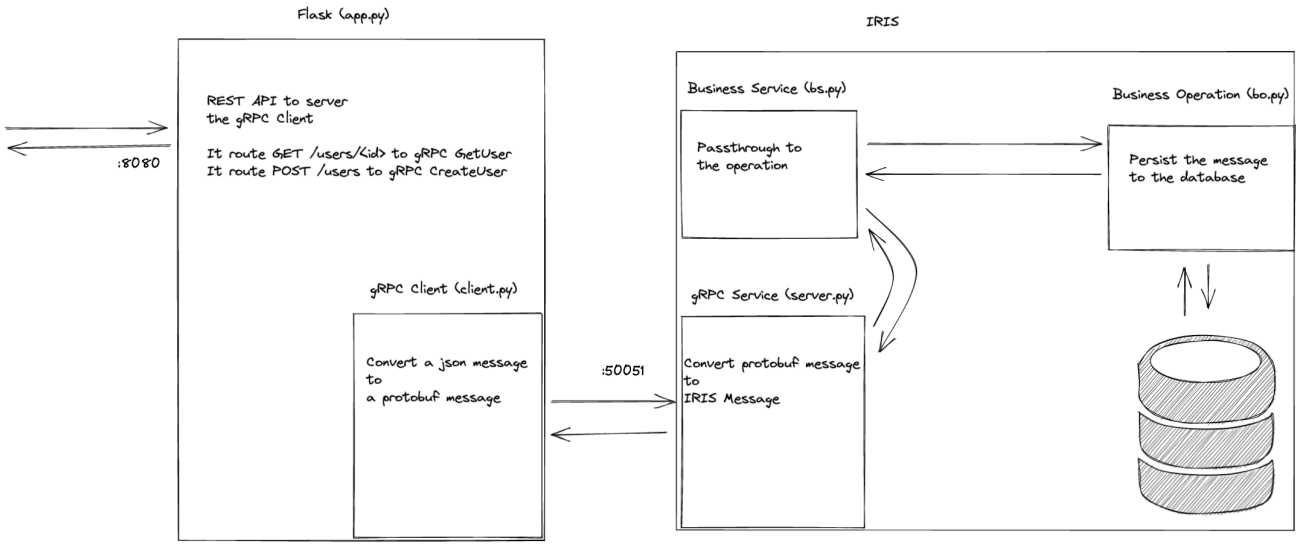
Open Exchange

gRPC and IRIS Interoperability

grpc-iris-interop

The aim of this proof of concept is to show how the gRPC protocl can be implemented with the IRIS ineroperabilty module.

architecture



On this schema, we can see that the gRPC Service is hosted by IRIS.

This service must invoke the IRIS interoperability module. For that it transforms the protobuf messages to IRIS messages.

The gRPC client is host by a Flask server for demo purpose, the gRPC client can also be invoke by the python script.

definition of each file

users.proto

```
syntax = "proto3";
package users;

service Users {
  rpc CreateUser (users.CreateUserRequest) returns (users.CreateUserResponse);
  rpc GetUser (users.GetUserRequest) returns (users.GetUserResponse);
}

message User {
```

```
  uint32 id = 1;
  string name = 2;
  string dob = 3;
  string company = 4;
  string phone = 5;
  string title = 6;

}

message CreateUserRequest {
  User user = 1;
}

message CreateUserResponse {
  User user = 1;
}

message GetUserRequest {
  uint32 id = 1;
}

message GetUserResponse {
  User user = 1;
}
```

This file is in ./misc/proto/users

This file holds the definition of the gRPC implementation, it ' s kinda the swagger spec.

To create the implementation classes you have to invoke this command :

```
python3 -m grpc_tools.protoc \
        --proto_path=./misc/proto/users/ \
        --python_out=src/python/grpc/ \
        --grpc_python_out=src/python/grpc/ \
        ./misc/proto/users/users.proto
```

This command generate two files :

- userspb2.py that hold the definition of the protobuf objects User and messages GetUser and CreateUser
- userspb2grpc.py that hold a Client and Service

obj.py

This module is in ./src/python/grpc/

```
import users_pb2 as users

from dataclasses import dataclass
from datetime import date

@dataclass
class User:
    """
```

```
    this class represents a user

    this class have the following attributes:
        id: int
        name: str
        dob: date
        company: str
        phone: str
        title: str
    """

    id: int = 0
    name: str = ''
    dob: date = date.fromisoformat('0001-01-01')
    company: str = ''
    phone: str = ''
    title: str = ''

    def to_protobuf(self) -> users.User:
        """
        this method converts a User object to a protobuf User object

        :return: a protobuf User object
        """

        return users.User(
            id=self.id,
            name=self.name,
            company=self.company,
            dob=self.dob.isoformat(),
            phone=self.phone,
            title=self.title
        )

    @staticmethod
    def from_protobuf(user: users.User) -> 'User':
        """
        this method converts a protobuf User object to a User object

        :param user: a protobuf User object
        :return: a User object
        """

        return User(
            id=user.id,
            name=user.name,
            dob=date.fromisoformat(user.dob),
            company=user.company,
            phone=user.phone,
            title=user.title
        )
```

This module has on class User.

This class is a dataclass that represents a user.

This class is used to store user information.

This class is used to serialize and deserialize user information to protobuf.

This class is used in the IRIS messages.

server.py

This module is in ./src/python/grpc/

```
from concurrent import futures

from grongier.pex import Director

import users_pb2_grpc as service
import users_pb2 as message

import grpc

from obj import User

from msg import (CreateUserRequest, GetUserRequest)

class UsersService(service.UsersServicer):
    """
    this class is used to create a server for the gRPC service

    it inherits from users_pb2_grpc.UsersServicer
    """

    def CreateUser(self, request, context):
        """
        this method is used to add a user to the database

        :param request: a protobuf CreateUserRequest object
        :param context: a grpc.ServicerContext object
        :return: a protobuf CreateUserResponse object
        """

        user = User.from_protobuf(request.user)
        msg = CreateUserRequest(user=user)

        service = Director.create_python_business_service("Python.gRPCService")
        iris_response = service.on_process_input(msg)

        response = message.CreateUserResponse(user=iris_response.user.to_protobuf())

        return response

    def GetUser(self, request, context):
        """
        this method is used to get a user from the database

        :param request: a protobuf GetUserRequest object
        :param context: a grpc.ServicerContext object
        :return: a protobuf GetUserResponse object
        """
```

```
        msg = GetUserRequest(id=request.id)

        service = Director.create_python_business_service("Python.gRPCService")
        iris_response = service.on_process_input(msg)

        response = message.GetUserResponse(user=iris_response.user.to_protobuf())

        return response

def main():
    """
    this function is used to start the server
    """

    server = grpc.server(futures.ThreadPoolExecutor(max_workers=10))
    service.add_UsersServicer_to_server(UsersService(), server)
    server.add_insecure_port('[::]:50051')
    server.start()
    server.wait_for_termination()

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

This module is used to create a server for the gRPC service.

This module has the following classes:

- UsersService: this class is used to create a server for the gRPC service it inherits from userspb2grpc.UsersServicer

This module has the following functions:

- main: this function is used to start the server

UsersService has the following functions:

- CreateUser:
  - This function is used to retrive an Business Service instance with the IRIS Director module.
  - It makes use of User classe to deserialize the protobuf message and convert it in User dataclass.
- GetUser:
  - This function is used to retrive an Business Service instance with the IRIS Director module.
  - It makes use of User classe to deserialize the protobuf message and convert it in User dataclass.

client.py

This module is in ./src/python/grpc/

```
from concurrent import futures

from datetime import date
```

```
import users_pb2_grpc as service
import users_pb2 as message

import grpc

from obj import User

class UsersClient:
    """
    this class is an helper class for the gRPC client
    it overrides the methods of the UsersStub class
    to make use of the User class
    this class has the following methods:
        create_user: this method is used to create a user
                     this method converts the request to a User protobuf object
                     this method calls the original method
        get_user: this method is used to get a user
                  this method calls the original method
        __init__: create a new stub from the channel
    """
    channel = None
    stub = None

    def __init__(self, channel):
        """
        create a new stub from the channel
        """
        self.channel = channel
        self.stub = service.UsersStub(channel)

    def create_user(self, user: User):
        """
        this method is used to create a user
        :param user: a User object
        :return: a User object
        """
        request = message.CreateUserRequest(user=user.to_protobuf())
        response = self.stub.CreateUser(request)

        return User.from_protobuf(response.user)

    def get_user(self, id: int):
        """
        this method is used to get a user
        :param id: the id of the user
        :return: a User object
        """
        request = message.GetUserRequest(id=id)
        response = self.stub.GetUser(request)
        return User.from_protobuf(response.user)

def main():
    """
    this function is used to start the client
    """

    with grpc.insecure_channel('localhost:50051') as channel:
        client = UsersClient(channel)
```

```
        user = User(
            name='John Doe',
            dob=date.fromisoformat('2000-01-01'),
            company='Acme',
            phone='555-555-5555',
            title='CEO'
        )

        user = client.create_user(user)
        print(user)

        user = client.get_user(user.id)
        print(user)

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

This module is used to create a client for the gRPC service.

This module has the following classes:

- UsersClient: this class is used to create a client for the gRPC service on init it create a userspb2grpc.UsersStub with a channel parameter

This module has the following functions:

- main: this function is used to start the client outside of the Flask application

The UsersClient class has the following methods:

- createuser:
    - this method is used to create a user
    - this method converts the request to a User protobuf object
    - this method calls the original method
- getuser:
    - this method is used to get a user
    - this method calls the original method
- init: create a new stub from the channel

app.py

This module is in ./src/python/grpc/

```
import grpc

from flask import Flask, request, jsonify
```

```
from datetime import date

from client import UsersClient

from obj import User

app = Flask(__name__)

# create a new client
client = UsersClient(grpc.insecure_channel('localhost:50051'))

@app.route('/users', methods=['POST'])
def create_user():
    """
    this method is used to create a user
    :return: a json object
    """

    # get the data from the request
    data = request.get_json()

    # create a new user from the data
    user = User(dob=date.fromisoformat(data['dob']) ,name=data['name'], company=data[
'company'], phone=data['phone'], title=data['title'])

    # create the user
    user = client.create_user(user)

    # return the user
    return jsonify(user)

@app.route('/users/<int:id>', methods=['GET'])
def get_user(id):
    """
    this method is used to get a user
    :return: a json object
    """

    # get the user
    user = client.get_user(id)

    # return the user
    return jsonify(user)

# start the app if main
if __name__ == '__main__':
    app.run('0.0.0.0', port = "8080")
```

This module is the flask app to server the gRCP client

This module has the following classes:

- app:

- this class is used to create the flask app
  - it has the following methods:
    - createuser:
      - this method is used to create a user
      - it returns a json object
      - it converts the request payload to a User object
      - it calls the gRPC client to create the user
    - getuser:
      - this method is used to get a user
      - it returns a json object
      - it calls the gRPC client to get the user by id

This module has the following functions:

- main: this function is used to start the flask app

Run this project

```
docker-compose up
```

Play with it :

```
POST http://localhost:8080/users HTTP/1.1
Content-Type: application/json

{
    "company": "tesdfst",
    "dob": "0001-01-01",
    "name": "fsd",
    "phone": "fdsffdsf",
    "title": "sfd"
}


GET http://localhost:8080/users/1 HTTP/1.1
```

#Embedded Python #Python #InterSystems IRIS
Check the related application on InterSystems Open Exchange