Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №7

По дисциплине «ПИС»

за 5-й семестр

Выполнил:
студент 3 курса
группы ПО-3 (1)
Лущ М. Г.

Проверил:
Лаврущик А.И.

Брест, 2021

Вариант 4

Цель работы: Познакомиться с практической реализацией принципа инверсии зависимостей, а также протоколами межсервисного взаимодействия.

Задание для выполнения:

Реализуйте механизмы хранения (persistence) для ранее разработанного приложения – технологию, фреймворк, ORM выберите сами. Это могут быть реляционная БД, NoSQL, InMemory, файловая система, Redis и т.д. Реализуйте минимум два механизма доставки – способов вызова функций вашего приложения. Например, REST и командная строка. Отдельно реализуйте микросервис, который не будет выполнять никакой полезной работы, кроме вызова какой-либо функции вашего приложения по протоколу gRPC (третий способ доставки для вашего приложения), используйте Google Protobuf. Большим плюсом станет, если данный микросервис будет реализован не на PHP:

Тема: Учёт военнообязанных.

Ход работы

Код Api:

```php
<?php

header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");

include("../DatabaseContext.php");
include("../Repositories/MilitaryReservistRepository.php");
include("../Repositories/ReservistRepository.php");
include("../Application/ReservistRegistrationService.php");
include("../Entities/Reservist.php");

$connection = DatabaseContext::getInstance()->getConnection();
$militaryReservistRepository = new MilitaryReservistRepository($connection);
$reservistRepository = new ReservistRepository($connection);

$service = new ReservistRegistrationService($militaryReservistRepository, $reservistReposi
tory);

$result = $service->getAllReservists();

if (count($result) == 0)
{
    http_response_code(404);
    echo json_encode(array('message' => 'reservists not found'));
}
```

```
else
{
    http_response_code(200);

    $response = array();
    foreach ($result as $value)
    {
        $responseItem = array(
            "id" => $value->getId(),
            "firstName" => $value->getFirstName(),
            "lastName" => $value->getLastName(),
            "birthDate" => $value->getBirthDate()->format('Y-m-d H:i:s'),
        );

        array_push($response, $responseItem);
    }

    echo json_encode($response);
}
```

Отдельный сервис взаимодействующий по протоколу gRPC:

Серверная часть:

```
const PROTO_PATH = 'definitions.proto';

const grpc = require('@grpc/grpc-js');
const protoLoader = require('@grpc/proto-loader');

const packageDefinition = protoLoader.loadSync(
    PROTO_PATH,
    {
        keepCase: true,
        longs: String,
        enums: String,
        defaults: true,
        oneofs: true
    });

var hello_proto = grpc.loadPackageDefinition(packageDefinition);

function Reservist(call, callback) {
    var fetch = require('node-fetch');

    fetch('http://localhost:8001/Api/Reservists.php')
            .then(res => {
                if(res.ok) return res
                else throw new Error(`The HTTP status of the reponse: ${res.status} (${res
.statusText})`)
            })
```

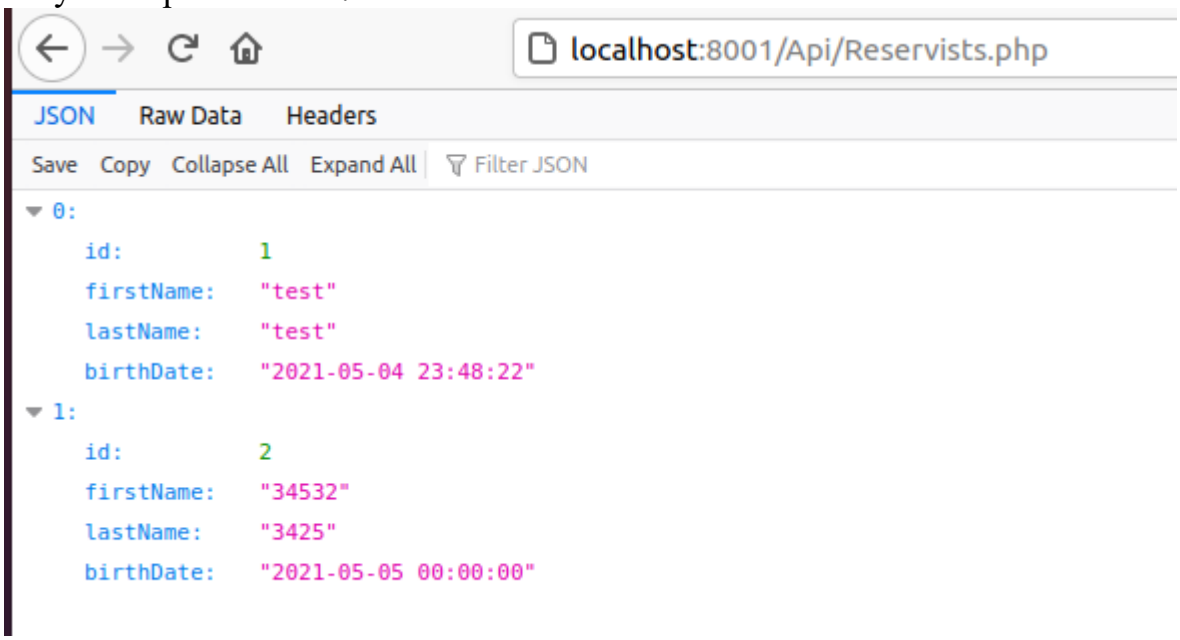
```
            .then(res => res.json())
            .then(json => {
                console.log(json);
                console.log(call.request.id);

                const data = json;
                const query = data.filter(el => el.id === call.request.id)
                callback(null, query[0]);
            })
}

function main() {
    const server = new grpc.Server();
    server.addService(hello_proto.ReservistService.service, {Reservist: Reservist});
    server.bindAsync('0.0.0.0:50051', grpc.ServerCredentials.createInsecure(), () => {
      server.start();
    });
}

main();
```

Клиентская часть:

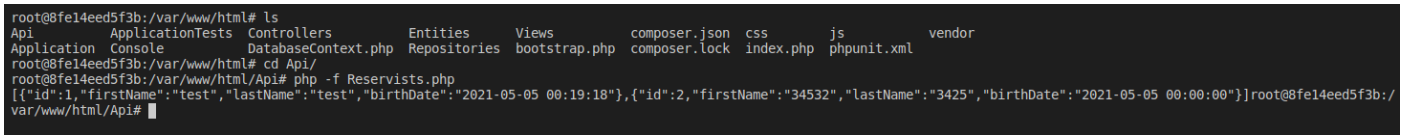
```
const PROTO_PATH = 'definitions.proto';

var parseArgs = require('minimist');
var grpc = require('@grpc/grpc-js');
var protoLoader = require('@grpc/proto-loader');

var packageDefinition = protoLoader.loadSync(
    PROTO_PATH,
    {keepCase: true,
     longs: String,
     enums: String,
     defaults: true,
     oneofs: true
    });

var hello_proto = grpc.loadPackageDefinition(packageDefinition);

function main() {
    var argv = parseArgs(process.argv.slice(1));
    const customerID = argv['Id'];

    var client = new hello_proto.ReservistService('localhost:50051',
        grpc.credentials.createInsecure());

    client.Reservist({id: customerID}, function(err, response) {
        console.log('Reservist:', response);
```

```
    });
}

main();
```

Результат работы API:



Результат работы в консоли:



Результат работы gRPC: