

Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №7

По дисциплине «ПИС»
за 5-й семестр

Выполнил:
студент 2 курса
группы ПО-3 (1)
Афанасьев В.В.

Проверил:
Лаврущик А.И.

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

Вариант: 2

Задание:

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.

Предметная область: продажа билетов на мероприятия.

Ход работы:

Api/events.php

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

include_once '../Repositories/EventRepository.php';
include("../DatabaseContext.php");

$connection = DatabaseContext::getInstance() -> getConnection();
$eventRepository = new EventRepository($connection);

$stmt = $eventRepository->getAll();

if (Count($stmt) > 0) {
    http_response_code(200);
    echo json_encode($stmt);
}
else {
    http_response_code(404);
    echo json_encode(array("message" => "Категории не найдены."), JSON_UNESCAPED_UNICODE);
}
```



CLI:

```
docker exec -it 0c346e6714669d8337606ff24200436df1441332242e4cccc236930d0c3ed9ba /bin/sh
# ls
Api                               Controllers      Views        css       vendor
Application                      DatabaseContext.php  bootstrap.php  index.php
ApplicationTests                  Entities        composer.json js
'C:\Users\Nord\Desktop\pis\.phpunit.result.cache' Repositories  composer.lock  phpunit.xml
# cd Api
# php -f events.php
[{"Id":1,"EventName":"RadioTapok","EventTime":"15-10-2023"}, {"Id":2,"EventName":"RadioTapok","EventTime":"15-10-2023"}, {"Id":3,"EventName":"RadioTapok","EventTime":"15-10-2023"}, {"Id":4,"EventName":"12","EventTime":"12"}, {"Id":5,"EventName":"Test1221","EventTime":"Test1221"}, {"Id":6,"EventName":"4","EventTime":"4"}]#
```

gRPC:

server.js

```
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 Event(call, callback) {
  var fetch = require('node-fetch');

  fetch('http://localhost:8001/Api/events.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.EventService.service, {Event: Event});
}
```

```
server.bindAsync('0.0.0.0:50051', grpc.ServerCredentials.createInsecure(), () => {
  server.start();
}) ;

}

main();
```

client.js

```
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(2));
  const eventID = argv['eventID'];

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

  client.Event({Id: eventID}, function(err, response) {
    console.log('Event:', response);
  });
}
```

```
main();
```

definitions.proto

```
syntax = "proto3";

service EventService {
  rpc Event(EventId) returns (Event) {}
```

```
}
```

```
message Event {  
    int32 Id = 1;  
    string EventName = 2;  
    string EventTime = 3;  
}
```

```
message EventId {  
    int32 Id = 1;  
}
```

The screenshot shows a terminal window with the title "1: node, powershell". The code being run is:

```
[  
3  
[  
{ Id: 1, EventName: 'RadioTapok', EventTime: '15-10-2023' },  
{ Id: 2, EventName: 'RadioTapok', EventTime: '15-10-2023' },  
{ Id: 3, EventName: 'RadioTapok', EventTime: '15-10-2023' },  
{ Id: 4, EventName: '12', EventTime: '12' },  
{ Id: 5, EventName: 'Test1221', EventTime: 'Test1221' },  
{ Id: 6, EventName: '4', EventTime: '4' }  
]  
5
```

The output of the command `node client --eventID 1` is:

```
Event: { Id: 1, FirstName: '', LastName: '' }  
PS C:\Users\Nord\Desktop\node_grpc\client> node client --eventID 1  
Event: { Id: 1, EventName: 'RadioTapok', EventTime: '15-10-2023' }  
PS C:\Users\Nord\Desktop\node_grpc\client>
```

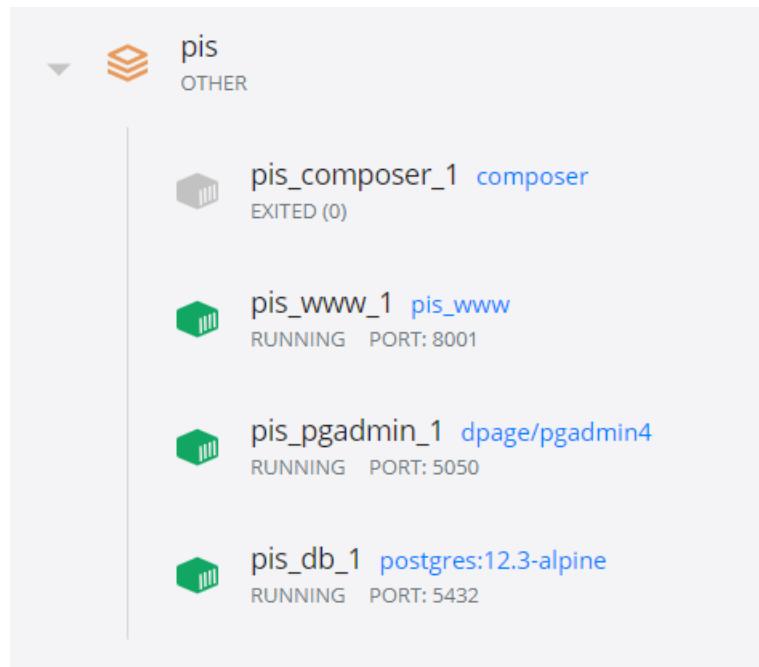
The output of the command `node client --eventID 3` is:

```
Event: { Id: 3, FirstName: '', LastName: '' }  
PS C:\Users\Nord\Desktop\node_grpc\client> node client --eventID 3  
Event: { Id: 3, EventName: 'RadioTapok', EventTime: '15-10-2023' }  
PS C:\Users\Nord\Desktop\node_grpc\client>
```

The output of the command `node client --eventID 5` is:

```
Event: { Id: 5, FirstName: '', LastName: '' }  
PS C:\Users\Nord\Desktop\node_grpc\client> node client --eventID 5  
Event: { Id: 5, EventName: 'Test1221', EventTime: 'Test1221' }  
PS C:\Users\Nord\Desktop\node_grpc\client>
```

База данных развернута в контейнере:



Выводы: в ходе выполнения лабораторной работы были ознакомлены с практической реализацией принципа инверсии зависимостей, а также протоколами межсервисного взаимодействия.