

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

Отчет по лабораторной работе №7
Дисциплина “ПИС”

Выполнила:
студентка, гр. ПО-3
Гаврилюк Р. И.

Проверил:
Лаврущик А.

Лабораторная работа №7
Инфраструктура: хранение и доставка. Межсервисное взаимодействие
Вариант 21

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.

Предметная область: продажа игрушек.

Persistence-layer реализуем через реляционную SQL базу данных. Настроим MySQL (файл .env)

```
DATABASE_URL="mysql://root:root@127.0.0.1:3306/shop?serverVersion=5.7"
```

Сформируем миграции для полученных ранее моделей данных:

```
<?php

declare(strict_types=1);

namespace DoctrineMigrations;

use Doctrine\DBAL\Schema\Schema;
use Doctrine\Migrations\AbstractMigration;

/**
 * Auto-generated Migration: Please modify to your needs!
 */
final class Version20201017140635 extends AbstractMigration
{
    public function getDescription() : string
    {
        return '';
    }

    public function up(Schema $schema) : void
    {
        // this up() migration is auto-generated, please modify it to your needs
    }
}
```

```

        $this->
>addSql('CREATE TABLE shop_cart (id INT AUTO_INCREMENT NOT NULL, shop_item_id INT NOT NULL
, session_id VARCHAR(255) NOT NULL, count INT NOT NULL, INDEX IDX_CA516ECC115C1274 (shop_i
tem_id), PRIMARY KEY(id)) DEFAULT CHARACTER SET utf8mb4 COLLATE `utf8mb4_unicode_ci` ENGIN
E = InnoDB');
        $this->
>addSql('CREATE TABLE shop_items (id INT AUTO_INCREMENT NOT NULL, price VARCHAR(255) NOT N
ULL, title VARCHAR(255) NOT NULL, description LONGTEXT NOT NULL, PRIMARY KEY(id)) DEFAULT
CHARACTER SET utf8mb4 COLLATE `utf8mb4_unicode_ci` ENGINE = InnoDB');
        $this->
>addSql('CREATE TABLE shop_order (id INT AUTO_INCREMENT NOT NULL, session_id VARCHAR(255)
NOT NULL, status INT NOT NULL, user_name VARCHAR(255) NOT NULL, user_email VARCHAR(255) NO
T NULL, user_phone VARCHAR(255) NOT NULL, PRIMARY KEY(id)) DEFAULT CHARACTER SET utf8mb4 C
OLLATE `utf8mb4_unicode_ci` ENGINE = InnoDB');
        $this->
>addSql('ALTER TABLE shop_cart ADD CONSTRAINT FK_CA516ECC115C1274 FOREIGN KEY (shop_item_i
d) REFERENCES shop_items (id)');
    }

    public function down(Schema $schema) : void
    {
        // this down() migration is auto-generated, please modify it to your needs
        $this->addSql('ALTER TABLE shop_cart DROP FOREIGN KEY FK_CA516ECC115C1274');
        $this->addSql('DROP TABLE shop_cart');
        $this->addSql('DROP TABLE shop_items');
        $this->addSql('DROP TABLE shop_order');
    }
}

```

Создадим репозитории для взаимодействия с данными:

ShopCartRepository.php

```

<?php

namespace App\Repository;

use App\Entity\ShopCart;
use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
use Doctrine\Persistence\ManagerRegistry;

/**
 * @method ShopCart|null find($id, $lockMode = null, $lockVersion = null)
 * @method ShopCart|null findOneBy(array $criteria, array $orderBy = null)
 * @method ShopCart[]    findAll()
 * @method ShopCart[]    findBy(array $criteria, array $orderBy = null, $limit = null, $of
fset = null)
 */
class ShoppingCartRepository extends ServiceEntityRepository
{
    public function __construct(ManagerRegistry $registry)
    {

```

```

        parent::__construct($registry, ShopCart::class);
    }

    /**
     * @return ShopCart[] Returns an array of ShopCart objects
     */
    public function findByExampleField($value)
    {
        return $this->createQueryBuilder('s')
            ->andWhere('s.exampleField = :val')
            ->setParameter('val', $value)
            ->orderBy('s.id', 'ASC')
            ->setMaxResults(10)
            ->getQuery()
            ->getResult()
        ;
    }

    public function findOneBySomeField($value): ?ShopCart
    {
        return $this->createQueryBuilder('s')
            ->andWhere('s.exampleField = :val')
            ->setParameter('val', $value)
            ->getQuery()
            ->getOneOrNullResult()
        ;
    }
}

```

ShopItemsRepository.php

```

<?php

namespace App\Repository;

use App\Entity\ShopItems;
use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
use Doctrine\Persistence\ManagerRegistry;

/**
 * @method ShopItems|null find($id, $lockMode = null, $lockVersion = null)
 * @method ShopItems|null findOneBy(array $criteria, array $orderBy = null)
 * @method ShopItems[]    findAll()
 * @method ShopItems[]    findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)
 */
class ShopItemsRepository extends ServiceEntityRepository
{
    public function __construct(ManagerRegistry $registry)
    {
        parent::__construct($registry, ShopItems::class);
    }
}

```

```

// /**
//  * @return ShopItems[] Returns an array of ShopItems objects
//  */
public function findByExampleField($value)
{
    return $this->createQueryBuilder('s')
        ->andWhere('s.exampleField = :val')
        ->setParameter('val', $value)
        ->orderBy('s.id', 'ASC')
        ->setMaxResults(10)
        ->getQuery()
        ->getResult()
    ;
}

public function findOneBySomeField($value): ?ShopItems
{
    return $this->createQueryBuilder('s')
        ->andWhere('s.exampleField = :val')
        ->setParameter('val', $value)
        ->getQuery()
        ->getOneOrNullResult()
    ;
}
}

```

ShopOrderRepository.php

```

<?php

namespace App\Repository;

use App\Entity\ShopOrder;
use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
use Doctrine\Persistence\ManagerRegistry;

/**
 * @method ShopOrder|null find($id, $lockMode = null, $lockVersion = null)
 * @method ShopOrder|null findOneBy(array $criteria, array $orderBy = null)
 * @method ShopOrder[]    findAll()
 * @method ShopOrder[]    findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)
 */
class ShopOrderRepository extends ServiceEntityRepository
{
    public function __construct(ManagerRegistry $registry)
    {
        parent::__construct($registry, ShopOrder::class);
    }

    // /**
    //  * @return ShopOrder[] Returns an array of ShopOrder objects
    //  */

```

```

// */
public function findByExampleField($value)
{
    return $this->createQueryBuilder('s')
        ->andWhere('s.exampleField = :val')
        ->setParameter('val', $value)
        ->orderBy('s.id', 'ASC')
        ->setMaxResults(10)
        ->getQuery()
        ->getResult()
    ;
}

public function findOneBySomeField($value): ?ShopOrder
{
    return $this->createQueryBuilder('s')
        ->andWhere('s.exampleField = :val')
        ->setParameter('val', $value)
        ->getQuery()
        ->getOneOrNullResult()
    ;
}
}

```

Микросервис:

```

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

include("../DatabaseContext.php");
include("../Repositories/ShopOrderRepository.php");
include("../Repositories/ShopCartRepository.php");
include("../Application/ShopOrderService.php");
include("../Entities/Order.php");

$connection = DatabaseContext::getInstance()->getConnection();
$shopOrderRepository = new ShopOrderRepository($connection);
$shopCartRepository = new ShopCartRepository($connection);

$service = new ShopOrderService($shopOrderRepository, $shopCartRepository);

$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" => $value->getLastName()
    );
    array_push($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 ShopOrder(call, callback) {
    var fetch = require('node-fetch');
    fetch('http://localhost:8000/Api/ShopOrder.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.ShopOrderService.service, {ShopOrder: ShopOrder});
    server.bindAsync('0.0.0.0:65000', grpc.ServerCredentials.createInsecure(), ()=>{ server.start(); });
}
main();
```

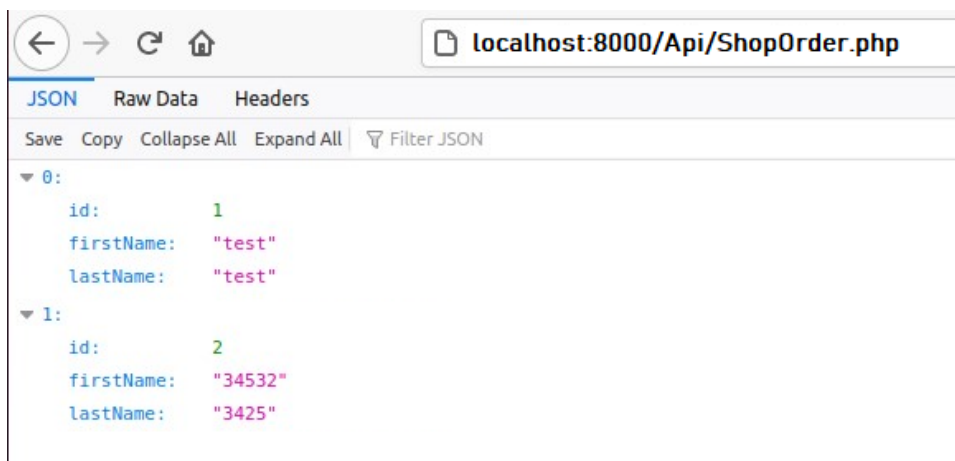
Клиентская часть:

```
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.ShopOrder('localhost:65000', grpc.credentials.createInsecure());
    client.ShopOrder({id: customerID}, function(err, response) { console.log('ShopOrder:', response); });
}
main();
```

Результат работы API:



Результат работы gRPC:

```
avocado@renata:~/Documents/PIS/7/grpc/server$ node server.js
[
  {
    id: 1,
    firstName: 'test',
    lastName: 'test'
  },
  {
    id: 2,
    firstName: '34532',
    lastName: '3425'
  }
]
avocado@renata:~/Documents/PIS/7/grpc/server$ node server.js --Id 1
{
  id: 1,
  firstName: 'test',
  lastName: 'test'
}
```

Вывод: познакомилась с практической реализацией принципа инверсии зависимостей, а также протоколами межсервисного взаимодействия.