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

Отчет
По лабораторной работе №7
по дисциплине «ПрИС»

Выполнила
студентка 3 курса
группы ПО-3
Гаврилкович Е.В.
Проверил
Лаврущик А.И.

Брест 2021

Лабораторная работа №7

Инфраструктура: хранение и доставка
Межсервисное взаимодействие

ВАРИАНТ 5

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. Промышленная робототехника.

Ход работы

Для реализации Persistence-layer’a используем реляционную SQL базу данных. Для использования данной SQLite раскомментируем строку в файле .env.

```
DATABASE_URL="sqlite:///%kernel.project_dir%/var/data.db"
```

Сформируем миграции:

```
public function up(Schema $schema): void
    {
        // this up() migration is auto-generated, please modify it to your needs
        $this->addSql('CREATE TABLE new_programmer (id INTEGER PRIMARY KEY
AUTOINCREMENT NOT NULL, name VARCHAR(255) NOT NULL)');
        $this->addSql('DROP INDEX IDX_6117D13B1FB354CD');
        $this->addSql('CREATE TEMPORARY TABLE_temp_robot AS SELECT id,membership_id, name,
cost, version FROM robot');
        $this->addSql('DROP TABLE robot');
        $this->addSql('CREATE TABLE robot (id INTEGER PRIMARY KEY AUTOINCREMENT NOT
NULL, membership_id INTEGER DEFAULT NULL, name VARCHAR(255)NOT NULL COLLATE BINARY,
cost INTEGER NOT NULL, version INTEGER NOT NULL,
CONSTRAINT FK_6117D13B1FB354CD FOREIGN KEY (membership_id) REFERENCES membership(id) NOT
DEFERRABLE INITIALLY IMMEDIATE)');
        $this->addSql('INSERT INTO robot (id, membership_id, name, cost, version) SELECT id, membership_id,
name, cost, version FROM__temp__robot');
```

```
            $this->addSql('DROP TABLE___temp__robot');
            $this->addSql('CREATE INDEX IDX_6117D13B1FB354CD ON robot
(membership_id)');
            $this->addSql('DROP INDEX IDX_CECE98A619EB6921');
            $this->addSql('CREATE TEMPORARY TABLE_temp__robot_session AS SELECTid, programmer_id, date
FROM robot_session');
            $this->addSql('DROP TABLE robot_session');
            $this->addSql('CREATE TABLE robot_session (id INTEGER PRIMARY KEY AUTOINCREMENT
NOT NULL, programmer_id INTEGER NOT NULL, date DATE NOT NULL, CONSTRAINT
FK_CECE98A619EB6921 FOREIGN KEY (programmer_id) REFERENCES programmer (id)NOT
DEFERRABLE INITIALLY IMMEDIATE)');
            $this->addSql('INSERT INTO robot_session (id, programmer_id, date) SELECTid, programmer_id, date
FROM__temp__robot_session');
            $this->addSql('DROP TABLE___temp__robot_session');
            $this->addSql('CREATE INDEX IDX_CECE98A619EB6921 ON robot_session(programmer_id)');
            $this->addSql('DROP INDEX IDX_93CC5B6B558FBEB9');
            $this->addSql('DROP INDEX IDX_93CC5B6B7EF00B89');
            $this->addSql('CREATE TEMPORARY TABLE__temp__robot_session_robotAS SELECT
robot_session_id, robot_id FROM robot_session_robot');
            $this->addSql('DROP TABLE robot_session_robot');
            $this->addSql('CREATE TABLE robot_session_robot (robot_session_id INTEGER NOT NULL, robot_id
INTEGER NOT NULL, PRIMARY KEY(robot_session_id, robot_id), CONSTRAINT FK_93CC5B6B7EF00B89
FOREIGN KEY (robot_session_id) REFERENCES robot_session (id) ON DELETE CASCADE NOTDEFERRABLE
INITIALLY IMMEDIATE, CONSTRAINT FK_93CC5B6B558FBEB9 FOREIGN KEY
(robot_id) REFERENCES robot (id) ON DELETE CASCADE NOT DEFERRABLEINITIALLY
IMMEDIATE)');
            $this->addSql('INSERT INTO robot_session_robot (robot_session_id, robot_id) SELECT robot_session_id,
robot_id FROM
_temp__robot_session_robot');
            $this->addSql('DROP TABLE___temp__robot_session_robot');
            $this->addSql('CREATE INDEX IDX_93CC5B6B558FBEB9 ON
robot_session_robot (robot_id)');
            $this->addSql('CREATE INDEX IDX_93CC5B6B7EF00B89 ON
robot_session_robot (robot_session_id)');
        }
```

Создадим репозиторий для взаимодействия с данными ProgrammerRepository

```
<?php
namespace App\Repository;use

App\Entity\Programmer;

use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;use
Doctrine\Persistence\ManagerRegistry;

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

        public function add(Programmer $programmer)
```

```
    {
        $this->addProgrammer($programmer->getName(), $programmer->getNumber());
    }

    public function addProgrammer(string $name, int $number): Programmer
    {
        $entityManager = $this->getEntityManager();
        $programmer = new Programmer();
        $programmer->setName($name);
        $programmer->setNumber($number);
        $entityManager->persist($programmer);
        $entityManager->flush();

        return $programmer;
    }

    public function getProgrammersCount(): int
    {
        $entityManager = $this->getEntityManager();
        $records = $entityManager->getRepository(Programmer::class)->findAll();return count($records);
    }
}
```

RobotRepository

```
<?php
namespace App\Repository;use

App\Entity\Robot;

use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;use
Doctrine\Persistence\ManagerRegistry;

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

    public function addRobot(string $name, int $cost, int $version): Robot
    {
        $entityManager = $this->getEntityManager();
        $robot = new Robot();
        $robot->setName($name);
        $robot->setCost($cost);
        $robot->setVersion($version);

        $entityManager->persist($robot);
        $entityManager->flush();

        return $robot;
    }

    public function getRobotsCount(): int
    {
```

```
            $entityManager = $this->getEntityManager();
            $records = $entityManager->getRepository(Robot::class)->findAll();return count($records);
        }
}
```

RobotSessionRepository

```
<?php

namespace App\Repository;

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

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

        public function addRobotSession(\DateTimeInterface $date,
\App\Entity\Robot $robot, \App\Entity\Programmer $programmer)
        {
            $entityManager = $this->getEntityManager();
            $shoppingSession = new RobotSession();
            $shoppingSession->setDate($date);
            $shoppingSession->setProgrammer($programmer);
            $shoppingSession->addRobot($robot);

            $entityManager->persist($shoppingSession);
            $entityManager->flush();
        }

        public function getRobotSessionCount(): int
        {
            $entityManager = $this->getEntityManager();
            $records = $entityManager->getRepository(RobotSession::class)-
>findAll();
            return count($records);
        }
}
```

Создадим Rest Api Controller.

MainApiController

```
<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;use
Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\Serializer\Encoder\JsonEncoder;use
Symfony\Component\Serializer\Serializer;
use Symfony\Component\Serializer\Normalizer\ObjectNormalizer;use
Symfony\Component\HttpFoundation\Response;
use App\Entity\Robot;


class MainApiController extends AbstractController
{
	#[Route('/main/api', name: 'main_api')]public function
	index(): Response
	{
		return $this->render('main_api/index.html.twig', ['controller_name' =>
			'MainApiController',
		]);
	}

	public function getAllRobots()
	{
		$entityManager = $this->getDoctrine()->getManager();
		$robots = $entityManager->getRepository(Robot::class)->findAll();
		$encoders = [new JsonEncoder()];
		$normalizers = [new ObjectNormalizer()];
		$serializer = new Serializer($normalizers, $encoders);
		$result_json = [];
		foreach ($robots as $robot)
		{
			$vrobot_json = [];
			$vrobot_json['id'] = $robot->getId();
			$vrobot_json['name'] = $robot->getName();
			$vrobot_json['cost'] = $robot->getCost();
			$vrobot_json['version'] = $robot->getVersion();array_push($result_json,
			$vrobot_json);
}
		$response = new Response($serializer->serialize($result_json,
'json'));
		$response->headers->set('Content-Type', 'application/json');return $response;
	}
}
```

Результат работы

[{"id" :1, "name" : "ad", "cost" : 33, "version" : 1},{"id" :2, "name" : "ad", "cost" : 33, "version" : 1},{"id" :3, "name" : "Robot-php7", "cost" : 1000, "version" : 4},{"id" :4, "name" : "RobotZX3000", "cost" : 5000, "version" : 3}]

Доступ из консоли реализуем посредством скрипта:

```
<?php shell_exec('php bin/console doctrine:query:sql "select * from robots;"');
```

```
string(1) "1"
["membership_id"]=>
string(1) "1"
["name"]=>
string(2) "ad"
["cost"]=>
string(2) "33"
["version"]=>
string(1) "1"

}

[1]=>

array(5) {
["id"]=>
string(1) "2"
["membership_id"]=>
string(1) "2"
["name"]=>
string(11) "RobotZX3000"
["cost"]=>
string(4) "5000"
["version"]=>
string(1) "3"

  }
}
```

Далее реализуем микросервис. Описание данных:

```
syntax = "proto3"
message test {
        int32 associated_value = 0;
}
```

Реализуем микросервис для отправки (используя Node.js):

```
const express = require('express')const app =
express()
const port = 3000
var PROTO_PATH = _dirname + './test.proto'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 test = grpc.loadPackageDefinition(packageDefinition).test
var programmer = new test.Test('localhost:50051',grpc.credentials.createInsecure())app.get('/', (req, res) => {
        programmer.sendTest({associated_data: 40})
})
app.listen(port, () => {6
console.log(`Microservice started`)
})
```

Реализуем дополнительный слой аналогично верхнему на Node.js, который будет вызывать наш php код.

```
var PROTO_PATH =_dirname + './test.proto'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
})
const { exec } = require("child_process")
var test = grpc.loadPackageDefinition(packageDefinition).testfunction sendTest(call,
callback) {
        console.log(call.request.associated_date)exec("php
        read_robots.php")
}
var server = new grpc.Server() server.addService(test.Test.service, { sendTest: sendTest })
server.bindAsync('0.0.0.0:50051', grpc.ServerCredentials.createInsecure(), ()
=> {
        server.start()
})
```

Вывод аналогичен предыдущему, но с дополнительно передаваемым вполе associated_data числом.

```
40
array(2) {

{0]=>

array(5) {
["id"]=>
string(1) "1"
["membership_id"]=>
string(1) "1"
["name"]=>
string(2) "ad"
["cost"]=>
string(2) "33"
["version"]=>
string(1) "1"

}

[1]=>

array(5) {
["id"]=>
string(1) "2"
["membership_id"]=>
string(1) "2"
["name"]=>
string(11) "RobotZX3000"
["cost"]=>
string(4) "5000"
["version"]=>
string(1) "3"
  }
}
```

Вывод

В данной лабораторной работе я познакомилась с практическойреализацией принципа инверсии зависимостей, а также протоколами межсервисного взаимодействия.