Министерство образования Республики Беларусь
Учреждение образования
“Брестский государственный технический университет”
Кафедра ИИТ

Отчёт
По лабораторной работе №4
По дисциплине ПрИС

Выполнил
Студент группы ПО-3
3-го курса
Кулинкович И. Т.

Проверил
Лаврущик А. И.

Брест 2021

Лабораторная работа №4

Модульное тестирование

ВАРИАНТ 17

Цель работы. Познакомиться с механизмами модульного тестирования веб-приложений, построенных на гексагональной архитектуре.

Задание для выполнения. Установите и настройте PHPUnit. Напишите модульные тесты для сценария транзакции из ЛР №1. Постарайтесь добиться 100% покрытия кода тестами. При написании постарайтесь учитывать, что в дальнейшем части этого кода вам могут пригодиться при тестировании доменной модели (ЛР №5) и сервисов приложения (ЛР №6).

Предметная область. Управление фитнес-центром.

Ход работы

Установим систему PHPUnit и интегрируем в имеющийся проект на Symfony, используя пакетный менеджер composer:

```
composer require --dev symfony/phpunit-bridge
```

Для дополнительной настройки (без каких-либо написанных тестов) напишем команду `./bin/phpunit` для финализации настройки тестового окружения. Команда успешно выполнилась и теперь мы можем перейти к выполнению написания тестов. Тесты будут хранится в папке tests и представляют из себя классы с функциями которые выполняют различные элементы логики приложения и проверяют её на целостность и корректность выполнения.

Для наших задач подойдет три файла тестов на php:
1. TrainerTest — тесты модели данных тренера.
2. VisitorTest — тесты модели данных посетителя.
3. TrainingSessionTest — тесты модели данных занятия фитнес-центра.

Ввиду того, что начальное приложение ЛР №1 было написано без Symfony, переписали логику на Symfony, получили три репозитория TrainerRepository, VisitorRepository, TrainingSessionRepository.

Перейдем к написанию тестов:

TrainerTest.php

```php
<?php namespace App\Tests;

use PHPUnit\Framework\TestCase;
use Symfony\Bundle\FrameworkBundle\Test\KernelTestCase;

class TrainerTest extends KernelTestCase
{
    private $entityManager;

    protected function setUp(): void
    {
        $kernel = self::bootKernel();

        $this->entityManager = $kernel->getContainer()
            ->get('doctrine')
            ->getManager();
    }

    public function testInsert()
```

```
    {
        $repository = $this->entityManager
            ->getRepository(\App\Entity\Trainer::class);

        $original_count = $repository->getTrainerCount();
        $repository->addTrainer("Ilya", 5);
        $new_count = $repository->getTrainerCount();
        $this->assertEquals($original_count + 1, $new_count);
    }

    public function testFindAll()
    {
        $repository = $this->entityManager
            ->getRepository(\App\Entity\Trainer::class);

        $our_count = $repository->getTrainerCount();
        $get_all_count = count($repository->findAll());
        $this->assertEquals($our_count, $get_all_count);
    }
}
```

VisitorTest.php

```
<?php namespace App\Tests;

use PHPUnit\Framework\TestCase;
use Symfony\Bundle\FrameworkBundle\Test\KernelTestCase;

class VisitorTest extends KernelTestCase
{
    private $entityManager;

    protected function setUp(): void
    {
        $kernel = self::bootKernel();

        $this->entityManager = $kernel->getContainer()
            ->get('doctrine')
            ->getManager();
    }

    public function testInsert()
    {
        $repository = $this->entityManager
            ->getRepository(\App\Entity\Visitor::class);

        $original_count = $repository->getVisitorCount();
        $repository->addVisitor("Ilya", 60, 180);
        $new_count = $repository->getVisitorCount();
        $this->assertEquals($original_count + 1, $new_count);
    }

    public function testFindAll()
    {
        $repository = $this->entityManager
            ->getRepository(\App\Entity\Visitor::class);

        $our_count = $repository->getVisitorCount();
        $get_all_count = count($repository->findAll());
        $this->assertEquals($our_count, $get_all_count);
```

```
    }
}
```

TrainingSessionTest.php

```php
<?php namespace App\Tests;

use PHPUnit\Framework\TestCase;
use Symfony\Bundle\FrameworkBundle\Test\KernelTestCase;

class TrainingSessionTest extends KernelTestCase
{
    private $entityManager;

    protected function setUp(): void
    {
        $kernel = self::bootKernel();

        $this->entityManager = $kernel->getContainer()
            ->get('doctrine')
            ->getManager();
    }

    public function testInsert()
    {
        $repository = $this->entityManager
            ->getRepository(\App\Entity\TrainingSession::class);

        $original_count = $repository->getTrainingSessionCount();
        $repository->addTrainingSession(new \DateTime(), 1, 1);
        $new_count = $repository->getTrainingSessionCount();
        $this->assertEquals($original_count + 1, $new_count);
    }

    public function testFindAll()
    {
        $repository = $this->entityManager
            ->getRepository(\App\Entity\TrainingSession::class);

        $our_count = $repository->getTrainingSessionCount();
        $get_all_count = count($repository->findAll());
        $this->assertEquals($our_count, $get_all_count);
    }
}
```

Используя команду `./bin/phpunit` проверим корректность наших тестов и, следовательно, корректность логики нашего приложения:

```
ilyakulinkovich@Ilyas-MacBook-Pro fitness_centre % php bin/phpunit
PHPUnit 8.5.14 by Sebastian Bergmann and contributors.

Testing Project Test Suite
......                                                              6 / 6 (100%)

Time: 1.92 seconds, Memory: 20.00 MB

OK (6 tests, 6 assertions)
```

Вывод

В данной лабораторной работе я познакомился с механизмами модульного тестирования веб-приложений, построенных на гексагональной архитектуре.