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

Отчет по лабораторной работе 4
Дисциплина “ПрИС”

Выполнил:

Студент группы ПО-3

Кабачук Д. С.

Проверил:

Лаврущик А. И.

Брест 2021

Модульное тестирование

Вариант 11

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: продажа игрушек.

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 финала 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. PassengersTest — тесты модели данных пассажиров.
2. ShipsTest — тесты модели данных кораблей.
3. TravelsTest — тесты модели данных путешествий.

PassengersTest.php

<?php namespace App\Tests;

use PHPUnit\Framework\TestCase;

use Symfony\Bundle\FrameworkBundle\Test\KernelTestCase;

```
class PassengersTest 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\Passenger::class);

        $original_count = $repository->getTrainerCount();
        $repository->addPassenger("Danik", 20);
        $new_count = $repository->getPassengerCount();
        $this->assertEquals($original_count + 1, $new_count);
    }
```

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

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

ShipsTest.php

```
<?php namespace App\Tests;

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

class ShipsTest 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\Ship::class);

    $original_count = $repository->getShipsCount();
    $repository->addShip("Danik", 60, 180);
    $new_count = $repository->getShipCount();
    $this->assertEquals($original_count + 1, $new_count);
}

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

    $our_count = $repository->getShipCount();
```

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

TravelsTest.php

```
<?php namespace App\Tests;

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

class TravelsSessionTest 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\TravelSession::class);

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

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

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

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