Министерство образования Республики Беларусь Учреждение образования

«Брестский государственный технический университет» Кафедра ИИТ

## Отчет по лабораторной работе 5 Дисциплина “ПИС”

**Выполнил:**

Студент группы ПО-4 Яковчик И.А.

## Проверил:

Михняев А.Л.

Брест 2022

**Вариант 12**

# 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: блог о фермерском хозяйстве.

# Ход работы:

composer require --dev symfony/phpunit-bridge

./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 два файла тестов на php:

1. UserTest — тесты модели данных пользователя.
2. NoteTest — тесты модели данных заметки.

UserTest.php

<?php namespace App\Tests;

use Symfony\Bundle\FrameworkBundle\Test\KernelTestCase;

class UserTest extends KernelTestCase

{

private $manager;

function configure(): void

{

$kernel = self::bootKernel();

$this->manager = $kernel->getContainer()

->get('doctrine')

->getManager();

}

public function testInsert()

{

$repository = $this->manager

->getRepository(\App\Entity\User::class);

$original\_count = $repository->getUserCount();

$repository->addUser("Nastya", 19);

$new\_count = $repository->getUsersCount();

$this->assertEquals($original\_count + 1,

$new\_count);

}

public function testFindAll()

{

$repository = $this->manager

->getRepository(\App\Entity\User::class);

$our\_count = $repository->getUsersCount();

$get\_all\_count = count($repository->findAll());

$this->assertEquals($our\_count, $get\_all\_count);

}

}

NoteTest.php

<?php namespace App\Tests;

use Symfony\Bundle\FrameworkBundle\Test\KernelTestCase; class NotesTest extends KernelTestCase

{

private $manager;

function configure(): void

{

$kernel = self::bootKernel();

$this->manager = $kernel->getContainer()

->get('doctrine')

->getManager();

}

public function testInsert()

{

$repository = $this->manager

->getRepository(\App\Entity\Note::class);

$original\_count = $repository->getShipsCount();

$repository->addNote("University", 8);

$new\_count = $repository->getNotesCount();

$this->assertEquals($original\_count + 1,

$new\_count);

}

public function testFindAll()

{

$repository = $this->manager

->getRepository(\App\Entity\Note::class);

$our\_count = $repository->getNotesCount();

$get\_all\_count = count($repository->findAll());

$this->assertEquals($our\_count, $get\_all\_count);

}

}

# Выводы:

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