Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №4**

По дисциплине «ПИС»

за 5-й семестр

Выполнил:
студент 3 курса
группы ПО-3 (1)
Лущ М. Г.

Проверил:
Лаврущик А.И.

Брест, 2021

**Вариант 4**

**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).

**Тема:** Учёт военнообязанных.

## Ход работы

Для более точного тестирования были добавлены несколько «заглушек» в виде «mock» классов по взаимодействию с БД:

**unitTests:**

```php
<?php


use PHPUnit\Framework\TestCase;

class MilitaryUnitServiceTest extends TestCase
{
    private MilitaryUnitService $service;
    private MockMilitaryUnitRepository $militaryUnitRepository;

    protected function setUp(): void {
        $this->militaryUnitRepository = new MockMilitaryUnitRepository();
        $this->service = new MilitaryUnitService($this->militaryUnitRepository);
    }

    public function test_getAllReservists_Should_ReturnExceptedValue()
    {
        $first = new MilitaryUnit();
        $first->setId(1);
        $first->setName('text');
        $first->setRegion('text');
        $first->setNumber(1);

        $second = new MilitaryUnit();
        $second->setId(2);
        $second->setName('text2');
        $second->setRegion('text2');
```

```
        $second->setNumber(2);

        $excepted = [$first, $second];

        $result = $this->service->getAllMilitaryUnits();

        $this->assertEquals($excepted, $result);
    }

    public function test_recordReservist_Should_CreateValue()
    {
        $excepted = new MilitaryUnit();
        $excepted->setId(3);
        $excepted->setName('text3');
        $excepted->setRegion('text3');
        $excepted->setNumber(3);

        $this->service->recordMilitaryUnit($excepted);

        $this->assertEquals(true, in_array($excepted, $this->militaryUnitRepository-
>store));
    }
}

<?php


use PHPUnit\Framework\TestCase;

class ReservistRegistrationServiceTest extends TestCase
{
    private ReservistRegistrationService $service;
    private MockMilitaryReservistRepository $militaryReservistRepository;
    private MockReservistRepository $reservistRepository;

    protected function setUp(): void {
        $this->militaryReservistRepository = new MockMilitaryReservistRepository();
        $this->reservistRepository = new MockReservistRepository();
        $this->service = new ReservistRegistrationService($this-
>militaryReservistRepository, $this->reservistRepository);
    }

    public function test_getAllMilitaryUnitReservists_Should_ReturnExceptedValue()
    {
        $first = new MilitaryReservist();
        $first->setId(1);
        $first->setMilitaryUnitId(1);
        $first->setReservistId(1);

        $second = new MilitaryReservist();
        $second->setId(2);
```

```
        $second->setMilitaryUnitId(2);
        $second->setReservistId(2);

        $excepted = [$first, $second];

        $result = $this->service->getAllMilitaryUnitReservists();

        $this->assertEquals($excepted, $result);
    }

    public function test_registerReservist_Should_CreateValue()
    {
        $excepted = new MilitaryReservist();
        $excepted->setId(3);
        $excepted->setMilitaryUnitId(3);
        $excepted->setReservistId(3);

        $this->service->registerReservist($excepted);

        $this->assertEquals(true, in_array($excepted, $this->militaryReservistRepository-
>store));
    }

    public function test_getAllReservists_Should_ReturnExceptedValue()
    {
        $first = new Reservist();
        $first->setId(1);
        $first->setFirstName('text');
        $first->setLastName('text');
        $first->setBirthDate(new DateTime('2020-10-10'));

        $second = new Reservist();
        $second->setId(2);
        $second->setFirstName('text2');
        $second->setLastName('text2');
        $second->setBirthDate(new DateTime('2020-10-10'));

        $excepted = [$first, $second];

        $result = $this->service->getAllReservists();

        $this->assertEquals($excepted, $result);
    }

    public function test_recordReservist_Should_CreateValue()
    {
        $excepted = new Reservist();
        $excepted->setId(4);
        $excepted->setFirstName('text');
        $excepted->setLastName('text');
```

```
        $excepted->setBirthDate(new DateTime('2020-10-10'));

        $this->service->recordReservist($excepted);

        $this->assertEquals(true, in_array($excepted, $this->reservistRepository->store));
    }
}
```

Mock repositories:

```
<?php
include("Interfaces/IMilitaryReservistRepository.php");

class MockMilitaryReservistRepository implements IMilitaryReservistRepository
{
    public array $store;

    public function __construct()
    {
        $first = (object) [
            'Id' => 1,
            'ReservistId' => 1,
            'MilitaryUnitId' => 1];

        $second = (object)  [
            'Id' => 2,
            'ReservistId' => 2,
            'MilitaryUnitId' => 2];

        $this->store = [$first, $second];
    }

    public function getAll(): array
    {
        return $this->store;
    }

    public function create(MilitaryReservist $entity)
    {
        array_push($this->store, $entity);
    }

    public function getByIdOrNull($id): MilitaryReservist
    {
        // TODO: Implement getByIdOrNull() method.
    }
```

```
    public function update(MilitaryReservist $entity)
    {
        // TODO: Implement update() method.
    }

    public function delete($id)
    {
        // TODO: Implement delete() method.
    }
}
```

```
<?php
include("Interfaces/IMilitaryUnitRepository.php");

class MockMilitaryUnitRepository implements IMilitaryUnitRepository
{
    public array $store;

    public function __construct()
    {
        $first = (object) [
            'Id' => 1,
            'Name' => 'text',
            'Number' => 1,
            'Region' => 'text'];

        $second = (object) [
            'Id' => 2,
            'Name' => 'text2',
            'Number' => 2,
            'Region' => 'text2'];

        $this->store = [$first, $second];
    }

    public function getAll(): array
    {
        return $this->store;
    }

    public function create(MilitaryUnit $entity)
    {
        array_push($this->store, $entity);
    }

    public function getByIdOrNull($id): MilitaryUnit
    {
        // TODO: Implement getByIdOrNull() method.
```

```
    }

    public function update(MilitaryUnit $entity)
    {
        // TODO: Implement update() method.
    }

    public function delete($id)
    {
        // TODO: Implement delete() method.
    }
}
```

```
<?php
include("Interfaces/IReservistRepository.php");

class MockReservistRepository implements IReservistRepository
{
    public array $store;

    public function __construct()
    {
        $first = (object) [
            'Id' => 1,
            'BirthDate' => '2020-10-10',
            'FirstName' => 'text',
            'LastName' => 'text'];

        $second = (object) [
            'Id' => 2,
            'BirthDate' => '2020-10-10',
            'FirstName' => 'text2',
            'LastName' => 'text2'];

        $this->store = [$first, $second];
    }

    public function getAll(): array
    {
        return $this->store;
    }

    public function create(Reservist $entity)
    {
        array_push($this->store, $entity);
    }

    public function getByIdOrNull($id): Reservist
```

```
    {
        // TODO: Implement getByIdOrNull() method.
    }

    public function update(Reservist $entity)
    {
        // TODO: Implement update() method.
    }

    public function delete($id)
    {
        // TODO: Implement delete() method.
    }
}
```

Результат выполнения тестов: