НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕЇНІКИ

ЛАБОРАТОРНА РОБОТА 4

з дисципліни:

«Сучасні методології і технології розробки програмного забезпечення»

на тему:

«Багатошарова архітектура програмних додатків. Піраміда тестування. Інтеграційні тести.»

Студента 3 курсу групи ІТ-81
Венделовського Івана Сергійовича

Кількість балів:_____ Оцінка_____

Викладач:_____к.т.н. Штифурак Юрій Михайлович
(оцінка)

Київ – 2021

ЗМІСТ

1 ПОСТАНОКА ЗАДАЧІ

Мета: Розуміти суть піраміди тестування. Навчитися реалізовувати інтеграційні тести.

Завдання:

1. За відсутності механізмів впровадження інтеграційних тестів (наприклад, зазорів) провести відповідний рефакторинг коду.

2. Реалізувати інтеграційні тести.

3. За умови наявності складної логіки, що не можливо покрити автономними тестами, провести рефакторинг коду, який забезпечить можливість написання unit-тестів для змістовної логіки розробленого ПЗ.

1 ПОКРИТТЯ ЗМІСТОВНОЇ ЛОГІКИ МОДУЛЬНИМИ ТЕСТАМИ

1.1 Реалізація тестів для класу «BillSevice»

```
public class BillServiceTest
{
    private BillService _billService;
    private MockDbContext _context;

    [SetUp]
    public void SetupBeforeEachTest()
    {
        _context = new MockDbContext();
        _billService = new BillService(new BillRepository(_context), new UserRepository(_context),
            new DeliveryRepository(_context), new WayRepository(_context));
    }
```

Рисунок 1.1 – Тестовий клас, необхідні фікстури та їх ініціалізація.

1.1.1 Реалізація тестів методу «InitializeBill»

```
[Test]
public void initializeBillCorrect()
{
    Way setupWayWithTarif = EntitySetuper.SetupWayWithTarif(_context);
    User setupAdresee = EntitySetuper.SetupAdresee(_context);
    User setupAdreser = EntitySetuper.SetupAdreser(_context);
    DeliveryOrderCreateModel deliveryOrderCreateModel = new DeliveryOrderCreateModel( deliveryWeight: 10,
        setupWayWithTarif.LocalitySandLocalityId, setupWayWithTarif.LocalityGetLocalityId, setupAdresee.Email);
    int expectedCost = 200;


    Bill billResult = _billService.InitializeBill(deliveryOrderCreateModel,
        setupAdreser.UserName);

    Assert.AreEqual( expected: setupAdreser.Id,  actual: billResult.User.Id);
    Assert.False(billResult.IsDeliveryPaid);
    Assert.False(billResult.Delivery.IsPackageReceived);
    Assert.AreEqual(expectedCost,  actual: billResult.CostInCents);
}
```

Рисунок 1.2 – Перевірка коректності створення рахунку при коректних вхідних даних.

```
[Test]
public void initializeBillIncorrectInWay()
{
    EntitySetuper.SetupWayWithTarif(_context);

    User setupAdresee = EntitySetuper.SetupAdresee(_context);
    User setupAdreser = EntitySetuper.SetupAdreser(_context);
    int incorrectLocalitySandId = 1000;
    int incorrectLocalityGetId = 100;
    DeliveryOrderCreateModel deliveryOrderCreateModel = new DeliveryOrderCreateModel( deliveryWeight: 10,
        incorrectLocalitySandId, incorrectLocalityGetId, setupAdresee.Email);

    var actualResult =
        Assert.Throws<NoSuchWayException>( code: () => _billService.InitializeBill(deliveryOrderCreateModel,
            setupAdreser.UserName));

    Assert.AreEqual( expected: typeof(NoSuchWayException), actualResult.GetType());
}
```

Рисунок 1.3 – Перевірка виникнення «NoSuchUserException» виключення у випадку якщо не існує користувача отриманого із вхідних даних.

```
[Test]
public void initializeBillIncorrectInWay()
{
    EntitySetuper.SetupWayWithTarif(_context);

    User setupAdresee = EntitySetuper.SetupAdresee(_context);
    User setupAdreser = EntitySetuper.SetupAdreser(_context);
    int incorrectLocalitySandId = 1000;
    int incorrectLocalityGetId = 100;
    DeliveryOrderCreateModel deliveryOrderCreateModel = new DeliveryOrderCreateModel( deliveryWeight: 10,
        incorrectLocalitySandId, incorrectLocalityGetId, setupAdresee.Email);

    var actualResult =
        Assert.Throws<NoSuchWayException>( code: () => _billService.InitializeBill(deliveryOrderCreateModel,
            setupAdreser.UserName));

    Assert.AreEqual( expected: typeof(NoSuchWayException), actualResult.GetType());
}
```

Рисунок 1.4 – Перевірка виникнення «NoSuchWayException» виключення у випадку якщо не існує маршруту отриманого із вхідних даних.

1.1.2 Реалізація тестів методу «GetBillsToPayByUserName»

```
[Test]
public void getBillsToPayByUserId()
{
    Delivery setupDeliveryAndBill = EntitySetuper.SetupDeliveryAndBill(_context, isDeliveryRecived: false, isDeliveryPayed: false);
    BillInfoToPayModel expectedBillInfoToPayModel = new BillInfoToPayModel(setupDeliveryAndBill.Bill.BillId,
        price: setupDeliveryAndBill.Bill.CostInCents, setupDeliveryAndBill.DeliveryId, setupDeliveryAndBill.Weight,
        setupDeliveryAndBill.Addressee.Email);
    expectedBillInfoToPayModel.LocalityGetName = setupDeliveryAndBill.Way.LocalityGet.NameEn;
    expectedBillInfoToPayModel.LocalitySandName = setupDeliveryAndBill.Way.LocalitySand.NameEn;

    List<BillInfoToPayModel> result =
        _billService.GetBillsToPayByUserName(setupDeliveryAndBill.Bill.User.UserName);

    Assert.AreEqual( expected: ServicesTestConstant.getBills().Count, actual: result.Count);
    Assert.AreEqual(expectedBillInfoToPayModel, actual: result[0]);
}
```

Рисунок 1.5 – Перевірка коректності отримання платежів за ім’ям користувача при коректних вхідних даних.

```
[Test]
public void getBillsToPayByUserIdUserIsNotExist()
{
    string notexcistusername = "NotExcistUserName";
    List<BillInfoToPayModel> billInfoToPayDtos = _billService.GetBillsToPayByUserName(notexcistusername);

    Assert.AreEqual( expected: 0, actual: billInfoToPayDtos.Count);
}
```

Рисунок 1.6 – Перевірка відсутності платежів у випадку якщо не існує зданого користувача.

```
[Test]
public void payForDeliveryWhenAllCorrect()
{
    Delivery setupDeliveryAndBill = EntitySetuper.SetupDeliveryAndBill(_context, isDeliveryRecived: false, isDeliveryPayed: false);

    bool payResult =
        _billService.PayForDelivery(setupDeliveryAndBill.Bill.User.Email, setupDeliveryAndBill.Bill.BillId);

    Assert.IsTrue(payResult);
    Assert.IsTrue(setupDeliveryAndBill.Bill.IsDeliveryPaid);
}
```

Рисунок 1.7 – Перевірка виникнення «NoSuchWayException» виключення у випадку якщо не існує маршруту отриманого із вхідних даних.

1.1.3 Реалізація тестів методу «PayForDelivery»

```
[Test]
public void payForDeliveryWhenAllCorrect()
{
    Delivery setupDeliveryAndBill = EntitySetuper.SetupDeliveryAndBill(_context, isDeliveryRecived: false, isDeliveryPayed: false);

    bool payResult =
        _billService.PayForDelivery(setupDeliveryAndBill.Bill.User.Email, setupDeliveryAndBill.Bill.BillId);

    Assert.IsTrue(payResult);
    Assert.IsTrue(setupDeliveryAndBill.Bill.IsDeliveryPaid);
}
```

Рисунок 1.8 – Перевірка коректності оплати при коректних вхідних даних.

```
[Test]
public void payForDeliveryNotEnoughMoney(){
    Delivery setupDeliveryAndBill = EntitySetuper.SetupDeliveryAndBill(_context, isDeliveryRecived: false, isDeliveryPayed: false);
    setupDeliveryAndBill.Bill.User.UserMoneyInCents = 0;
    _context.SaveChanges();

    var actualResult =
        Assert.Throws<NotEnoughMoneyException>
            ( code: ()=>_billService.PayForDelivery(ServicesTestConstant.getUserId(), ServicesTestConstant.getBillId()));

    Assert.AreEqual( expected: typeof(NotEnoughMoneyException), actualResult.GetType());
}
```

Рисунок 1.9 – Перевірка виникнення «NotEnoughMoneyException» виключення у випадку якщо у користувача не достатньо коштів для виконання оплати.

```
[Test]
public void payForDeliveryDeliveryAlreadyPaid() {
    Delivery setupDeliveryAndBill = EntitySetuper.SetupDeliveryAndBill(_context, isDeliveryRecived: false, isDeliveryPayed: true);

    var actualResult =
        Assert.Throws<DeliveryAlreadyPaidException>( code: ()=>_billService.PayForDelivery
            (ServicesTestConstant.getUserId(), ServicesTestConstant.getBillId()));

    Assert.AreEqual( expected: typeof(DeliveryAlreadyPaidException), actualResult.GetType());
}
```

Рисунок 1.10 – Перевірка виникнення «DeliveryAlreadyPaidException» виключення у випадку якщо рахунок вже оплачено.

1.2 Реалізація тестів для класу «DeliveryService»

```
public class DeliveryServiceTest
{
    private DeliveryService _deliveryService;
    private Mock<IWayRepository> _wayRepository;
    private Mock<IDeliveryRepository> _deliveryRepository;

    [SetUp]
    public void SetupBeforeEachTest()
    {
        _wayRepository = new Mock<IWayRepository>();
        _deliveryRepository = new Mock<IDeliveryRepository>();
        _deliveryService = new DeliveryService(_wayRepository.Object, _deliveryRepository.Object);
    }
```

Рисунок 1.11 – Тестовий клас, необхідні фікстури та їх ініціалізація.

1.2.1 Реалізація тестів методу «GetDeliveryInfoToGet»

```
namespace integration_test
{
    public class DeliveryServiceTests
    {
        private DeliveryService _deliveryService;
        private MockDbContext _context;

        [SetUp]
        public void SetupBeforeEachTest()
        {
            _context = new MockDbContext();
            _deliveryService = new DeliveryService(new WayRepository(_context), new DeliveryRepository(_context));
        }
```

Рисунок 1.12 – Перевірка коректності списку доставок на отримання при коректних вхідних даних.

1.2.2 Реалізація тестів методу «ConfirmGettingDelivery»

```
[Test]
public void confirmGettingDeliveryAllCorrect()
{
    Delivery delivery = EntitySetuper.SetupDeliveryAndBill(_context, isDeliveryRecived: false, isDeliveryPayed: true);

    bool result = _deliveryService.ConfirmGettingDelivery(delivery.Addressee.UserName, delivery.DeliveryId);

    Assert.IsTrue(result);
    Assert.IsTrue(delivery.IsPackageReceived);
}
```

Рисунок 1.13 – Перевірка коректності підтвердження отримання доставки при коректних вхідних даних.

```
[Test]
public void confirmGettingDeliveryIsNoExistDelivery()
{
    User adresee = EntitySetuper.SetupAdresee(_context);
    long randomNotExsistDeliveryId = 100L;

    var actualResult =
        Assert.Throws<AskedDataIsNotExist>( code: () => _deliveryService.ConfirmGettingDelivery(
            adresee.UserName,
            randomNotExsistDeliveryId));
    Assert.AreEqual( expected: typeof(AskedDataIsNotExist), actualResult.GetType());
}
```

Рисунок 1.14 – Перевірка виникнення виключення «AskedDataIsNotExist» у випадку якщо заданої доставки не існує.

1.2.3 Реалізація тестів методу «GetDeliveryCostAndTimeDto»

```
[Test]
public void getDeliveryCostAndTimeDtoAllCorrect()
{
    Way way = EntitySetuper.SetupWayWithTarif(_context);
    DeliveryInfoRequestModel deliveryInfoRequestDto =
        new DeliveryInfoRequestModel( deliveryWeight: 10, way.LocalitySandLocalityId, way.LocalityGetLocalityId);
    PriceAndTimeOnDeliveryModel priceAndTimeOnDeliveryModel = new PriceAndTimeOnDeliveryModel( costInCents: 200, timeOnWayInHours: 2);

    PriceAndTimeOnDeliveryModel result = _deliveryService.GetDeliveryCostAndTimeDto(deliveryInfoRequestDto);

    Assert.AreEqual( expected: priceAndTimeOnDeliveryModel, actual: result);
}
```

Рисунок 1.15 – Перевірка коректності отримання даних про час та вартість доставки при коректних вхідних даних.

```
[Test]
public void getDeliveryCostAndTimeIncorrectWay()
{
    Way way = EntitySetuper.SetupWayWithTarif(_context);
    int notExistLocalitySend = 300;
    DeliveryInfoRequestModel deliveryInfoRequestDto =
        new DeliveryInfoRequestModel( deliveryWeight: 10,  localitySandId: notExistLocalitySend, way.LocalityGetLocalityId);

    var actualResult =
        Assert.Throws<NoSuchWayException>( code: () =>
            _deliveryService.GetDeliveryCostAndTimeDto(deliveryInfoRequestDto));

    Assert.AreEqual( expected: typeof(NoSuchWayException), actualResult.GetType());
}
```

Рисунок 1.16 – Перевірка виникнення «NoSuchWayException» виключення у випадку якщо заданого маршруту не існує.

```
[Test]
public void getDeliveryCostAndTimeIncorrectWeightFactorBiggerOnOne()
{
    Way way = EntitySetuper.SetupWayWithTarif(_context);
    int weightBigerOnOneOfMaximumTarifWeightFactor =
        EntitySetuper.MAX_WRIGHT_ON_SETUPED_TARIF_WEIGHT_FACTOR + 1;
    DeliveryInfoRequestModel deliveryInfoRequestDto =
        new DeliveryInfoRequestModel(weightBigerOnOneOfMaximumTarifWeightFactor, way.LocalitySandLocalityId,
            way.LocalityGetLocalityId);

    var actualResult =
        Assert.Throws<UnsupportableWeightFactorException>( code: () =>
            _deliveryService.GetDeliveryCostAndTimeDto(deliveryInfoRequestDto));

    Assert.AreEqual( expected: typeof(UnsupportableWeightFactorException), actualResult.GetType());
}
```

Рисунок 1.17 – Перевірка виникнення «UnsupportableWeightFactorException» виключення у випадку вантаж занадто важки для заданого маршруту.

ВИСНОВКИ

В дані роботі було протестовано логіку додатку з допомогою інтеграційних тестів. Додаток було спроектовано коректно, з повним дотриманням принципів солід та модульності це дозволило уникнути необхідності в рефакторингу коду перед та у процесі тестування.

В звіті було наведено тести ключових методів що реалізують основний функціонал додатку. Всі тести можна знайти у репозиторії проекту за адресою «https://github.com/VINIPOOH/delivery_dot_net».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Ноубл, Дж., Андерсон, Т., Брэйтуэйт, Г., Казарио, М., Третола, Р. Flex 4. Рецепты программирования. — БХВ-Петербург, 2011. — С. 548. — 720 с

2) Самоучитель UML 2. – СПб.: БХВ-Петербург, 2007. – 567 с.: ил. ISBN 978-5-94157-878-8

3) Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. П75 Приемы объектно-ориентированного проектирования. Паттерны проектирования. — СПб: Питер, 2001. — 368 с.: ил. (Серия «Библиотека программиста») ISBN 5-272-00355-1

4) Мартин Фаулер., Чистий код: создание, анализ и рефакторинг. – СПб.: Питер, 2019. – 464 с.: ил.