УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Рефакторинг баз данных и приложений»

Лабораторная работа №3

Выполнили:

Бордун Анастасия

Иванов Евгений

P34111

Преподаватель

Логинов И. П.

Санкт-Петербург, 2023 г.

Задание:

Провести рефакторинг приложения(курсовая работа по ИСБД). Описать планируемые изменения, описать для чего они необходимы, провести их.

Выполнение:

В данной итерации мы добавляем unit-тесты на все сервисы(тестовое покрытие). Для этого мы используем JUnit 5 и Mockito. Тесты в системе необходимы для быстрого и дешевого обнаружения ошибок, их исправления, показа корректности работы системы. Изменения в коде:

1. Добавление необходимых зависимостей gradle для последующей разработки тестов.

```
testImplementation 'org.springframework.boot:spring-boot-starter-test'
testImplementation 'org.junit.jupiter:junit-jupiter-api:5.8.1'
testRuntimeOnly 'org.junit.jupiter:junit-jupiter-engine:5.8.1'
```

2. Создание тестов на каждый метод в сервисном слое. Файлы тестовых классов названы по стандартному принципу(такое же имя и расположение, но в директории test и с суффиксом Test). Для методов с исключениями или различной логикой реализованы несколько тестов.

Пример для BookingServiceTest:

```
@ExtendWith(MockitoExtension.class)
public class BookingServiceTest {

    @Mock
    private BookingRepository bookingRepository;
    @Mock
    private ClassesService classesService;
    @Mock
    private ClientService clientService;
    @Mock
    private BalanceRepository balanceRepository;
    @InjectMocks
    private BookingService bookingService;

    private final Long ID = 1L;
    private final UtilsObjects utilsObjects = new UtilsObjects();

    @Test
    void getBookingByIdOk() {
        Booking booking = new Booking();
        booking.setId(ID);

when(bookingRepository.findById(ID)).thenReturn(Optional.of(booking));
        Booking result = bookingService.getBookingById(ID);
```

```
        assertEquals(booking, result);
        verify(bookingRepository, times(1)).findById(eq(ID));
    }

    @Test
    void getBookingByIdThrowBookingNotFound() {
        assertThrows(BookingNotFoundException.class, () ->
bookingService.getBookingById(ID));
    }

    @Test
    void addBookingOk() {
        Client client = utilsObjects.CLIENT;
        Classes classes = utilsObjects.CLASSES;
        BookingDTO bookingDTO = new BookingDTO(
                CLIENT_ID,
                CLASSES_ID
        );

        when(clientService.getClient(CLIENT_ID)).thenReturn(client);
        when(classesService.getClassById(CLASSES_ID)).thenReturn(classes);
        doNothing().when(balanceRepository).setNewBalanceById(anyLong(),
anyFloat());
        Booking booking = new Booking(null, client, classes);
        when(bookingRepository.save(any())).thenReturn(booking);

        assertEquals(booking, bookingService.addBooking(bookingDTO));
        verify(bookingRepository, times(1)).save(any());
        verify(balanceRepository, times(2)).setNewBalanceById(anyLong(),
anyFloat());
    }

    @Test
    void addBookingThrowNotEnoughMoney() {
        Client client = utilsObjects.CLIENT;
        Classes classes = utilsObjects.CLASSES;
        client.getBalance().setValue(50F);

        BookingDTO bookingDTO = new BookingDTO(
                CLIENT_ID,
                CLASSES_ID
        );

        when(clientService.getClient(CLIENT_ID)).thenReturn(client);
        when(classesService.getClassById(CLASSES_ID)).thenReturn(classes);

        assertThrows(NotEnoughMoneyToBookException.class, () ->
bookingService.addBooking(bookingDTO));
    }
}
```

И так далее для каждого сервиса.

3. Общая логика и/или объекты, которые используются многократно от теста к тесту вынесены в отдельный каталог *utils* для удобной работы с ними(можно один раз их там объявить и использовать в конкретных тестах)

Класс для объектов ролей:

```
public class RoleObjects {
    public static Role UNAUTHORIZED_ROLE = new Role(
            1L, "UNAUTHORIZED"
    );
    public static Role USER_ROLE = new Role(
            2L, "USER"
    );
    public static Role ADMIN_ROLE = new Role(
            3L, "ADMIN"
    );
    public static Role MANAGER_ROLE = new Role(
            4L, "MANAGER"
    );
    public static Role INSTRUCTOR_ROLE = new Role(
            5L, "INSTRUCTOR"
    );
}
```

Класс для различных объектов системы:

```
public class UtilsObjects {

    public static Long STUDIO_ID = 1L;
    public static Long POSITION_ID = 1L;
    public static Long CLASSES_ID = 1L;
    public static Long CLIENT_ID = 1L;
    public static Long LEGAL_INFO_ID = 1L;

    public Balance CLIENT_BALANCE = new Balance(
           CLIENT_ID,
            200F,
            null
    );

    public Balance STUDIO_BALANCE = new Balance(
            STUDIO_ID,
            15000F,
            MANAGER_ROLE
    );

    public Client CLIENT = new Client(
            CLIENT_ID,
            "CLIENT_NAME",
            "client@mail.ru",
            "+79527977524",
```

```
            Gender.MALE.name(),
            null,
            CLIENT_BALANCE
    );

    public LegalInfo LEGAL_INFO = new LegalInfo(
            LEGAL_INFO_ID,
            "FULL DESCRIPTION",
            "+79527977524",
            "legal-info@mail.ru",
            "7394-134"
    );

    public Studio STUDIO = new Studio(
            STUDIO_ID,
            "STUDIO_NAME",
            "DESCRIPTION",
            null,
            LEGAL_INFO,
            STUDIO_BALANCE
    );

    public Position POSITION = new Position(
            POSITION_ID,
            "ADDRESS",
            "06:00-22:00",
            STUDIO,
            null
    );
    public Classes CLASSES = new Classes(
            CLASSES_ID,
            "CLASSES_NAME",
            new Date(new java.util.Date().getTime() + 86400000),
            Time.valueOf("10:00:00"),
            Time.valueOf("12:00:00"),
            100,
            POSITION,
            List.of()
    );

}
```

Код изменений:

Всю проделанную работу(все изменения в коде) с кратким описанием изменений можно найти в PR на гитхабе проекта: https://github.com/3ilib0ba/ITMO-RDBaA/pull/3