**МИНОБРНАУКИ РОССИИ**

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**ОТЧЕТ**

**по учебной практике УП.02.01 (по профилю специальности)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/8 группы

Алексейкин Николай Дмитриевич

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «07» октября 2024 г. по «19» октября 2024 г.

Руководитель практики Хисамутдинова А.С.

(подпись) (расшифровка подписи)

Итоговая оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2024

**ЗАДАНИЕ**

**на учебную практику (по профилю специальности)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/8 группы

Алексейкин Николай Дмитриевич

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «07» сентября 2024 г. по «19» октября 2024 г.

**Виды работ, обязательные для выполнения** *(переносится из программы, соответствующего ПМ):*

* Участие в выработке требований к программному обеспечению;
* Стадии проектирования программного обеспечения. Проектирование UML-диаграмм;
* Конструирование пользовательского интерфейса. Разработка модулей программного обеспечения;
* Разработка рабочего проекта и технологической документации.

**Индивидуальное задание: ВАРИАНТ 12**

Задание выдал «07» октября 2024 г. \_\_\_\_\_\_\_\_\_\_\_\_\_ Хисамутдинова А.С.

(подпись) (Ф.И.О.)

Задание получил «19» октября 2024 г. \_\_\_\_\_\_\_\_\_\_ Алексейкин Н. Д. (подпись) (Ф.И.О.)

**МИНОБРНАУКИ РОССИИ**

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**ДНЕВНИК**

**прохождения учебной практики УП.02.01**

**(по профилю специальности)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/8 группы

Алексейкин Николай Дмитриевич

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «07» октября 2024 г. по «19» октября 2024 г.

Руководитель практики Хисамутдинова А.С.

(подпись) (расшифровка подписи)

Итоговая оценка по практике

Санкт-Петербург

2024

**Содержание дневника**

|  |  |  |
| --- | --- | --- |
| **Дата** | **Виды выполненных работ и заданий по программе практики** | **Подпись руководителя практики** |
| **1** | **2** | **3** |
| 07.10.2024 | Анализ предметной области. Разработка технического задания |  |
| 08.10.2024 |
| 09.10.2024 | Проектирование диаграммы вариантов использования и последовательности. Проектирование диаграммы активности |  |
| 10.10.2024 |
| 11.10.2024 | Проектирование интерфейса пользователя |  |
| 12.10.2024 |
| 14.10.2024 | Разработка программного продукта. Создание функциональности ПО |  |
| 15.10.2024 |
| 16.10.2024 | Модульное тестирование. Создание тестовых случаев. Отладка программных модулей |  |
| 17.10.2024 |
| 18.10.2024 | Работа с GitHub |  |
| 19.10.2024 |

СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 5](#_Toc180061609)

[ВВЕДЕНИЕ 6](#_Toc180061610)

[Задание 1. Техническое задание 7](#_Toc180061611)

[Задание 2. Проектирование диаграмм 10](#_Toc180061612)

[Задание 3. Проектирование интерфейса пользователя 16](#_Toc180061613)

[Задание 4. Разработка ПО 19](#_Toc180061614)

[Задание 5. Модульное тестирование 21](#_Toc180061616)

[Задание 6. Работа с GitHub 25](#_Toc180061617)

[ЗАКЛЮЧЕНИЕ 30](#_Toc180061618)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 31](#_Toc180061619)

[ПРИЛОЖЕНИЕ А 32](#_Toc180061620)

ВВЕДЕНИЕ

Учебная практика посвящена анализу и разработке программного обеспечения для магазина «Shahstore». Основными целями практики являются изучение особенностей проектирования и реализации баз данных, а также приобретение навыков работы с Visual Studio и SQL Server.

В ходе практики будет составлено техническое задание, созданы UML диаграммы, разработана база данных и приложение к ней, проведено модульное тестирование.

Результатом выполнения данной работы будет надежная и эффективная система управления данными, которая позволит магазину эффективно продавать любую технику и обрабатывать заказы.

Задание 1. Техническое задание

Описание предметной области

Основной целью данного модуля является автоматизация процесса продажи техники Infinix. Модуль позволяет сотрудникам эффективно продавать технику.

Разработка программного модуля для продажи техники — это процесс создания программного продукта, который позволяет магазинам эффективно продавать любую технику и обрабатывать заказы.

Задача разработки такого программного модуля состоит в создании удобного и эффективного инструмента для учета и управления продажами техники, что помогает минимизировать время покупки, оптимизирует использование ресурсов и повышает удовлетворённость клиентов.

Основные функции и возможности модуля включают:

1. Управление товарными запасами: Возможность добавления, редактирования и удаления товаров с указанием названия, описания, цены, количества на складе и категории.
2. Обработка заказов: Возможность создания новых заказов, редактирования существующих и отслеживания статуса заказа (новый, в процессе, завершен).
3. Учет клиентов: Сохранение данных о клиентах, включая ФИО, номер телефона и адрес электронной почты, а также историю покупок.
4. Управление пользователями: Создание и управление ролями пользователей (администратор, продавец, менеджер) с соответствующими правами доступа.

Техническое задание

1. Общие сведения

* 1. Наименование проекта: Разработка программного модуля для продажи техники Infinix.
  2. Заказчик: Магазин «Shahstore».
  3. Исполнитель: IT-компания «B1tGuzok».

2. Функциональные требования

2.1. Возможность добавления товаров в базу данных с указанием следующих параметров:

* Номер товара;
* Название товара;
* Описание товара;
* Цена;
* Количество на складе;
* Категория товара.

2.2. Учёт клиентов:

* ФИО клиента;
* Номер телефона;
* Адрес электронной почты;
* История покупок.

2.3. Обработка заказов:

* Номер заказа;
* Дата заказа;
* Список товаров в заказе;
* Статус заказа (новый, в процессе, завершен);
* Возможность изменения статуса заказа.

2.4. Управление пользователями:

* Создание новых пользователей;
* Назначение ролей (администратор, продавец, менеджер);
* Управление правами доступа к функциям системы.

3. Нефункциональные требования

3.1. Кроссплатформенность:

- Поддержка работы на ОС семейства Windows.

3.2. Безопасность:

- Логин и пароль для доступа к приложению;

-Доступ к данным должен быть ограничен в зависимости от роли пользователя.

3.3. Удобство использования:

- Простой и интуитивный интерфейс;

- Информативные уведомления и подсказки.

3.4. Производительность:

- Приложение должно иметь быстрый доступ к данным;

- Минимальное время отклика на запросы пользователя.

4. Требования к реализации

4.1. Язык программирования: C#

4.2. СУБД: Microsoft SQL Server

5. Требования к документации

5.1. Техническое задание на разработку программного модуля

6. Руководство по стилю

6.1. Цветовая схема: зелёные и белые цвета с чёрным текстом. Оранжевый текст для заголовков в таблице.

Задание 2. Проектирование диаграмм

При разработке приложения были построены UML-диаграммы (Рисунок 1-5) и ER-диаграмма (Рисунок 6), разработан словарь данных (Таблица 1).



Рисунок 1 - Диаграмма вариантов использования



Рисунок 2 - Диаграмма последовательностей для кладовщика



Рисунок 3 - Диаграмма последовательностей для администратора



Рисунок 4 - Диаграмма последовательностей продавца



Рисунок 5 - Диаграмма активности



Рисунок 6 - ER-Диаграмма

Таблица 1 – Словарь данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Сущность** | **Атрибут** | **Тип данных** | **Описание** | **Ограничения** |
| **Пользователи**  **(Users)** | UserId | INT | Уникальный идентификатор пользователя | Первичный ключ, автоинкремент, Not null |
| Login | NVARCHAR(100) | Логин пользователя | Not null |
| Password | INT | Пароль пользователя | Not null |
| Role | NVARCHAR(100) | Роль пользователя в БД | Not null |
| **Заказы**  **(Orders)** | OrderId | INT | Уникальный идентификатор заказа | Первичный ключ, автоинкремент |
| ClientId | INT | Уникальный идентификатор клиента | Внешний ключ, Not null |
| OrderDate | DATE | Дата заказа | Not null |
| Status | BOOL | Статус заказа (доставлен или нет) | - |
| **Клиенты**  **(Clients)** | ClientId | INT | Уникальный идентификатор клиента | Первичный ключ, автоинкремент |
| FIO | INT | ФИО покупателя | Not null |
| Number | INT | Номер телефона покупателя | Not null |
| Email | NVARCHAR(100) | Почта покупателя | Not null |
| **Товары**  **(Products)** | ProductId | INT | Уникальный идентификатор товара | Первичный ключ, автоинкремент, Not null |
| Name | NVARCHAR(100) | Название товара | Not null |
| Description | NVARCHAR(100) | Характеристики товара | Not null |
| Price | DECIMAL(10, 2) | Цена товара | Not null |
| Quantity | INT | Кол-во на складе | Not null |
| **Список покупок**  **(PurchaseList)** | OrderId | INT | Уникальный идентификатор заказа | Внешний ключ, Not null |
| ProductId | INT | Уникальный идентификатор товара | Внешний ключ, Not null |

Код создания таблиц

Create DataBase Shahstore;

CREATE TABLE Users (

UserId INT PRIMARY KEY IDENTITY NOT NULL,

Login NVARCHAR(100) NOT NULL,

Password NVARCHAR(100) NOT NULL,

Role NVARCHAR(100) NOT NULL

);

CREATE TABLE Clients (

ClientId INT PRIMARY KEY IDENTITY NOT NULL,

FIO NVARCHAR(100) NOT NULL,

Number INT NOT NULL,

Email NVARCHAR(100) NOT NULL

);

CREATE TABLE Products (

ProductId INT PRIMARY KEY IDENTITY NOT NULL,

Name NVARCHAR(100) NOT NULL,

Description NVARCHAR(100) NOT NULL,

Price DECIMAL(10, 2) NOT NULL,

Quantity INT NOT NULL

);

CREATE TABLE Orders (

OrderId INT PRIMARY KEY IDENTITY NOT NULL,

ClientId INT NOT NULL,

OrderDate DATE NOT NULL,

Status BIT

FOREIGN KEY (ClientId) REFERENCES Clients(ClientId)

);

CREATE TABLE PurchaseList (

PurchaseId INT PRIMARY KEY IDENTITY NOT NULL,

OrderId INT NOT NULL,

ProductId INT NOT NULL,

FOREIGN KEY (OrderId) REFERENCES Orders(OrderId),

FOREIGN KEY (ProductId) REFERENCES Products(ProductId)

);

Код заполнения таблиц

INSERT INTO Products (Name, Description, Price, Quantity)

VALUES

('Infinix 15 2025', 'Смартфон. 6.1-дюймовый Super Retina XDR дисплей, A14 Bionic чип, 5G, двойная камера 12МП, Face ID', 799.99, 2),

('Infinix Galaxy S21', 'Смартфон. 6.2-дюймовый Dynamic AMOLED 2X дисплей, Exynos 2100, 8GB RAM, 4000mAh аккумулятор, 64МП тройная камера', 699.99, 75),

('Infinix IdeaPad 5', 'Ноутбук. 15.6-дюймовый FHD IPS дисплей, Intel Core i5-1135G7, 8GB RAM, 512GB SSD, Intel Iris Xe Graphics', 799.99, 30),

('Infinix XPS 13', 'Ноутбук. 13.3-дюймовый FHD+ InfinityEdge дисплей, Intel Core i7-1165G7, 16GB RAM, 512GB SSD, Intel Iris Xe Graphics', 1199.99, 20),

('Infinix WF-1000XM3', 'Беспроводные Наушники. Технология шумоподавления, Bluetooth 5.0, 24-битный аудиопроцессор DSEE HX, до 32 часов автономной работы', 228.99, 100);

INSERT INTO Clients (FIO, Number, Email)

VALUES

('John Doe', 12345, 'john.doe@example.com'),

('Jane Smith', 67890, 'jane.smith@example.com'),

('Michael Johnson', 13579, 'michael.johnson@example.com'),

('Emily Brown', 24680, 'emily.brown@example.com'),

('David Williams', 97531, 'david.williams@example.com');

INSERT INTO Users (Login, Password, Role)

VALUES

('King', 'admin', 'admin'),

('Spider', 'chest', 'storekeeper'),

('Nobody1', 'gnida1', 'customer'),

('Nobody2', 'gnida2', 'customer');

INSERT INTO Orders (ClientId, OrderDate, Status)

VALUES

(1, '2023-04-01', 1),

(2, '2023-04-02', 1),

(3, '2023-04-03', 0),

(4, '2023-04-04', 1),

(5, '2023-04-05', 0);

INSERT INTO PurchaseList (OrderId, ProductId)

VALUES

(1, 3),

(1, 4),

(2, 5),

(3, 6),

(4, 7),

(5, 4);

Задание 3. Проектирование интерфейса пользователя

При разработке приложения были разработаны wireframe (Рисунок 7-8) и mockup (Рисунок 9 - 10) макеты, а также карта навигации (Рисунок 11).

Изображение выглядит как текст, снимок экрана, диаграмма, дизайн

Автоматически созданное описание

Рисунок 7 - Wireframe макет окна авторизации

Изображение выглядит как текст, снимок экрана, линия, диаграмма

Автоматически созданное описание

Рисунок 8 - Wireframe макет для окна с таблицами

**Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание**

Рисунок 9 -Mockup макет для окна авторизации

Изображение выглядит как текст, снимок экрана, прямоугольный, Прямоугольник

Автоматически созданное описание

Рисунок 10 - Mockup макет для окна с таблицами



Рисунок 11 - Карта навигации

Задание 4. Разработка ПО

В системе предусмотрены 3 роли: storekeeper, customer, admin. В зависимости от роли пользователя в системе у него существуют ограничения. Кладовщик (storekeeper) имеет доступ к одной таблице Products. Продавец (customer) имеет доступ к трём таблицам Orders, Clients и PurchaseList. Администратор (admin) имеет доступ ко всем таблицам. Вид форм был разработан в соответствии с техническим заданием (Рисунок 12-15).

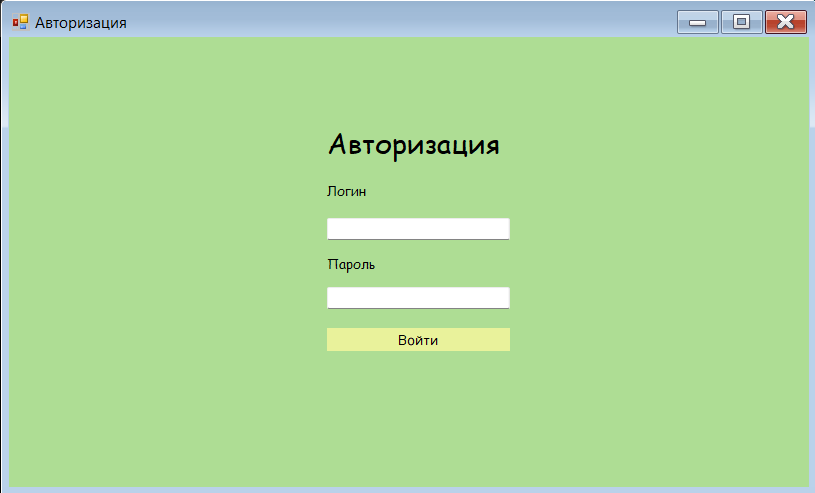


Рисунок 12 - Вид формы с авторизацией

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 13 - Вид формы с таблицами

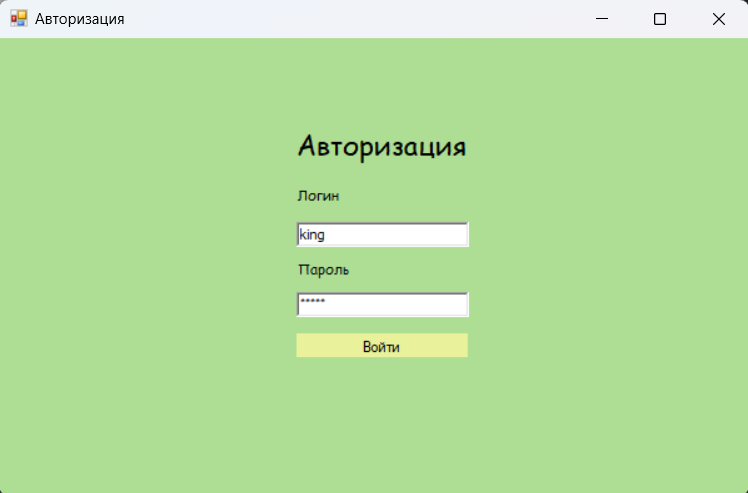


Рисунок 14 - Вид формы авторизации с заполненными полями

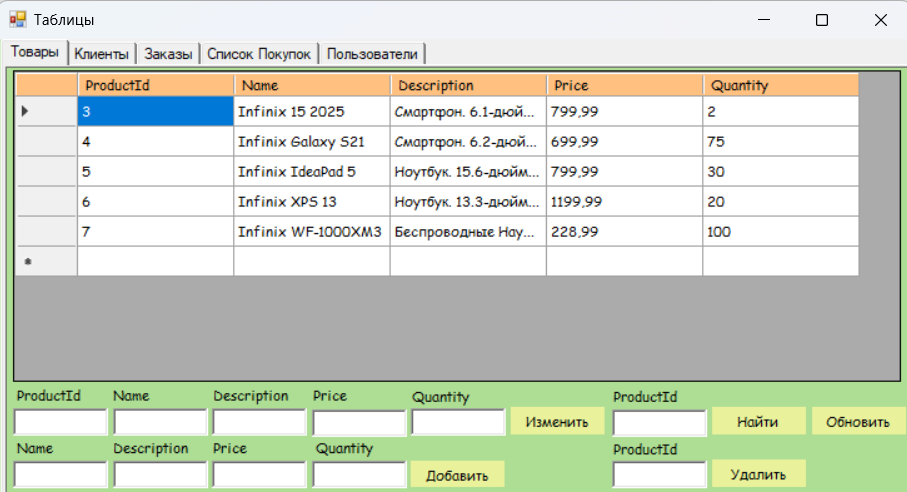


Рисунок 15 - Вид формы с таблицами с заполненными полями

Задание 5. Модульное тестирование

Разработка тестовых примеров (Таблица 2-7). Результат unit-тестов (Рисунок 16).

Таблица 2 – Аннотация теста

|  |  |
| --- | --- |
| **Название проекта** | Shahstore |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Николай |
| **Дата(ы) теста** | 17.10.2024 |

Таблица 3 - Проверка входа в систему

|  |  |
| --- | --- |
| **Тестовый пример #** | Test1 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Вход в систему пользователя с действительным логином и паролем |
| **Краткое изложение теста** | В результате выполнения теста должна открыться форма для пользователя и закрыться форма авторизации |
| **Этапы теста** | 1. Запустить приложение 2. Ввести логин и пароль 3. Нажать «Войти» |
| **Тестовые данные** | 1. логин: king 2. пароль: admin |
| **Ожидаемый результат** | Открытие новой формы и скрытие формы авторизации |
| **Фактический результат** | Открытие новой формы и скрытие формы авторизации |
| **Статус** | Зачет |
| **Предварительное условие** | Логин и пароль должны находиться в БД |
| **Постусловие** |  |
| **Примечания/комментарии** |  |

Таблица 4 - Проверка загрузки данных у администратора

|  |  |
| --- | --- |
| **Тестовый пример #** | Test2 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка загрузки данных во все таблицы при входе администратора |
| **Краткое изложение теста** | В результате выполнения теста должна открыться новая форма, в которой будут заполнены все элементы dataGridView |
| **Этапы теста** | 1. Запустить приложение 2. Ввести логин и пароль 3. Нажать «Войти» |
| **Тестовые данные** | 1. логин: king 2. пароль: admin |
| **Ожидаемый результат** | Открытие новой формы и наличие данных в таблицах |
| **Фактический результат** | Открытие новой формы и наличие данных в таблицах |
| **Статус** | Зачет |
| **Предварительное условие** | Логин и пароль должны находиться в БД, все таблицы должны быть заполнены |
| **Постусловие** |  |
| **Примечания/комментарии** |  |

Таблица 5 - Проверка работы окна подтверждения перед удалением записи

|  |  |
| --- | --- |
| **Тестовый пример #** | Test3 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Проверка появления окна подтверждения при попытке удаления записи |
| **Краткое изложение теста** | В результате выполнения теста должно появиться окно с вопросом об удалении записи |
| **Этапы теста** | 1. Запустить приложение 2. Ввести логин и пароль 3. Нажать «Войти» 4. Открыть таблицу Products 5. Ввести id и нажать кнопку «Удалить» |
| **Тестовые данные** | 1. логин: king 2. пароль: admin 3. id: 4 |
| **Ожидаемый результат** | Появление окна с надписью: “Вы уверены?” |
| **Фактический результат** | Появление окна с надписью: “Вы уверены?” |
| **Статус** | Зачет |
| **Предварительное условие** | Логин и пароль должны находиться в БД, в таблице Products должен быть товар с id = 4 |
| **Постусловие** |  |
| **Примечания/комментарии** |  |

Таблица 6 - Проверка появления окна с правильным форматом даты

|  |  |
| --- | --- |
| **Тестовый пример #** | Test4 |
| **Приоритет тестирования** | Низкий |
| **Заголовок/название теста** | Проверка появления окна с правильным форматом даты при вводе некорректной даты |
| **Краткое изложение теста** | В результате выполнения теста должно появиться окно с правильным форматом даты |
| **Этапы теста** | 1. Запустить приложение 2. Ввести логин и пароль 3. Нажать «Войти» 4. В таблице «Заказы» при добавлении нового заказа ввести неверную дату |
| **Тестовые данные** | 1. логин: king 2. пароль: admin 3. поле даты: 12/20/2005 |
| **Ожидаемый результат** | Появление окна с надписью: “Правильный формат даты: yyyy-mm-dd” |
| **Фактический результат** | Появление окна с надписью: “Правильный формат даты: yyyy-mm-dd” |
| **Статус** | Зачет |
| **Предварительное условие** | Логин и пароль должны находиться в БД |
| **Постусловие** |  |
| **Примечания/комментарии** |  |

Таблица 7 - Проверка появления окна с правильным форматом цены

|  |  |
| --- | --- |
| **Тестовый пример #** | Test5 |
| **Приоритет тестирования** | Низкий |
| **Заголовок/название теста** | Проверка появления окна с правильным форматом цены при вводе некорректной цены |
| **Краткое изложение теста** | В результате выполнения теста должно появиться окно с правильным форматом цены |
| **Этапы теста** | 1. Запустить приложение 2. Ввести логин и пароль 3. Нажать «Войти» 4. В таблице «Продукты» при добавлении нового заказа ввести неверную цену |
| **Тестовые данные** | 1. логин: king 2. пароль: admin 3. поле цены: 50..322 |
| **Ожидаемый результат** | Появление окна с надписью: “Правильный формат даты:999.99 ” |
| **Фактический результат** | Появление окна с надписью: “Правильный формат даты:999.99 ” |
| **Статус** | Зачет |
| **Предварительное условие** | Логин и пароль должны находиться в БД |
| **Постусловие** |  |
| **Примечания/комментарии** |  |

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 16 - Результат unit-тестов

CheckDate\_Second прошёл неудачно, потому что в функцию была специально передана неправильная дата. CheckPrice\_Second прошёл неудачно, потому в функцию была специально передана неправильная цена. SeatchProduct\_First прошёл неудачно, потому что при поиске специально ввели несуществующий id.

При тестировании было написано 10 Unit-тестов (Приложение A).

Задание 6. Работа с GitHub

Создал новый репозиторий на GitHub для проекта (Рисунок 17).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 17 - Созданный репозиторий

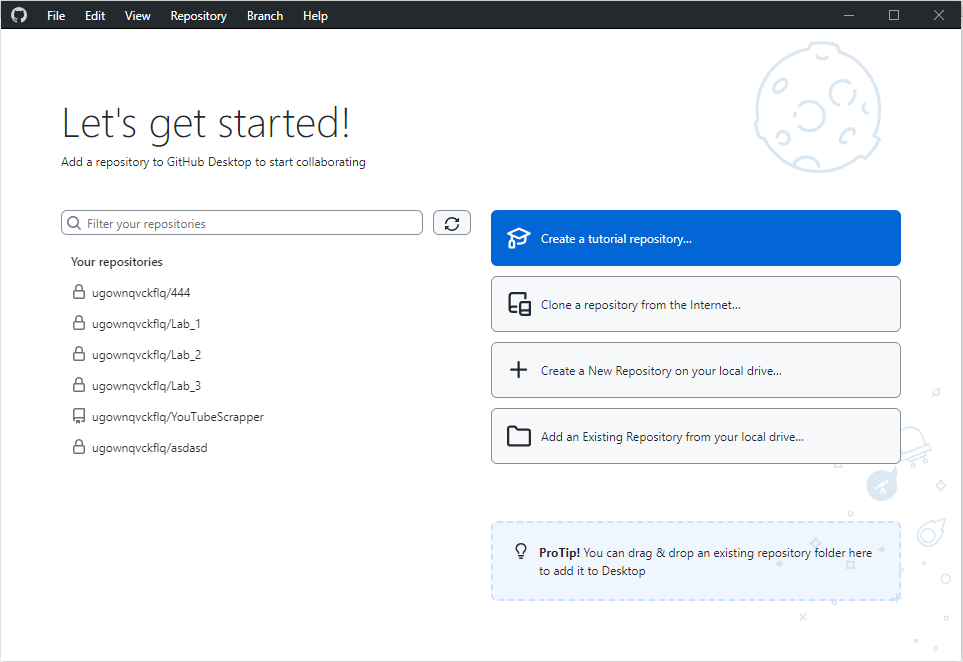


Рисунок 38 – Создание репозитория

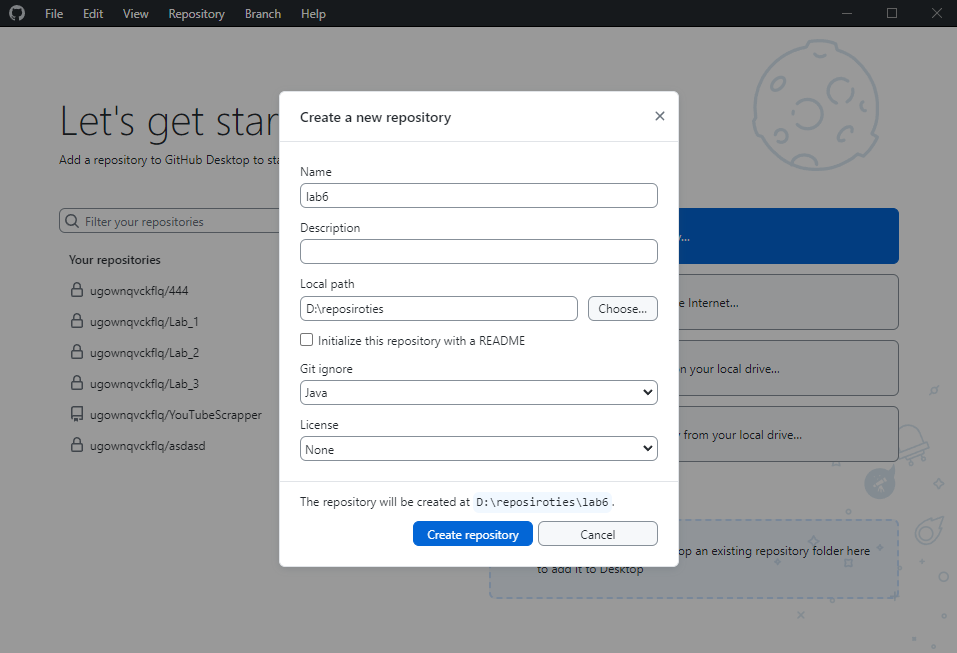


Рисунок 39 – Создание репозитория

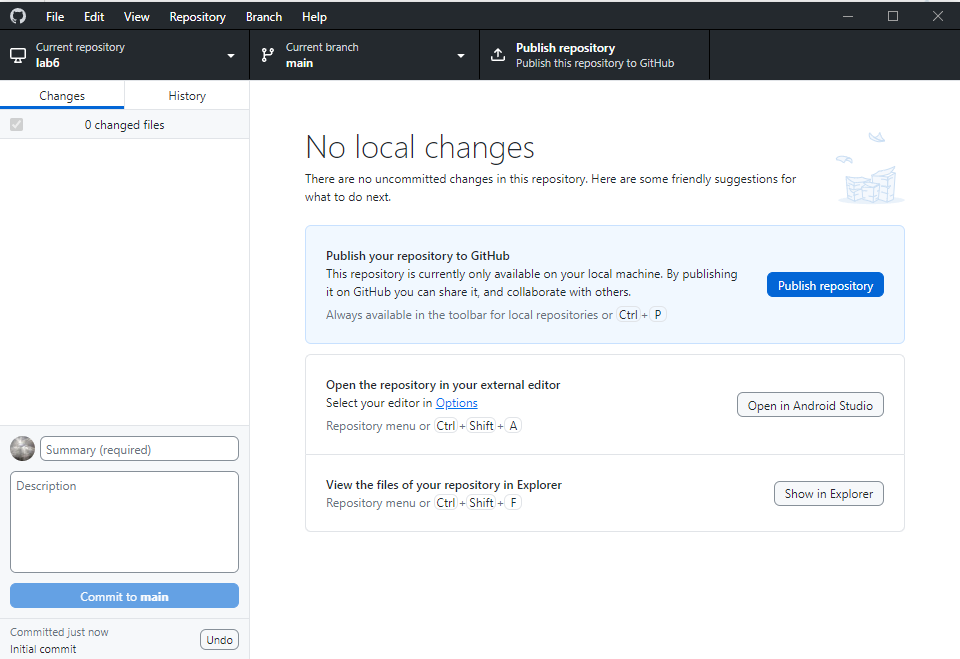


Рисунок 40 – Создание репозитория

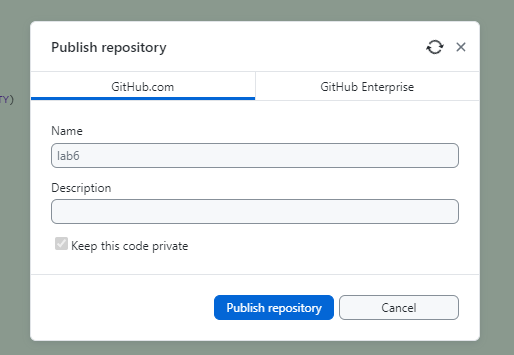


Рисунок 41 – Публикация репозитория

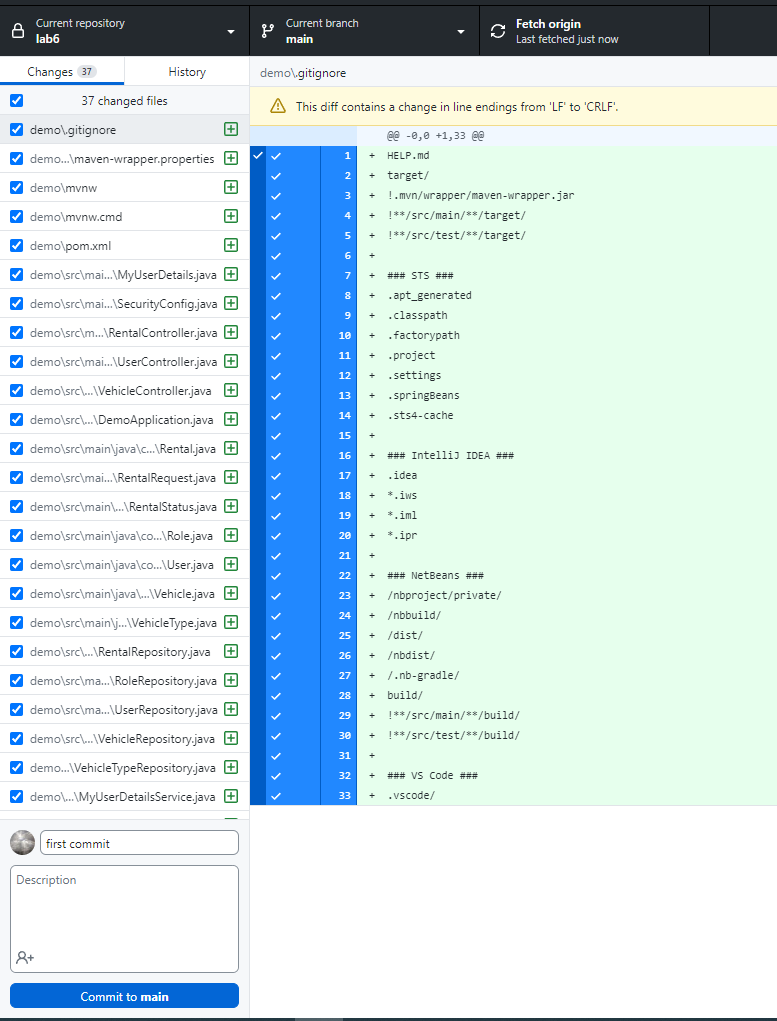


Рисунок 42 – Коммит изменений в репозиторий

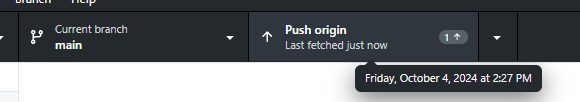


Рисунок 43 – Пуш коммита в удаленный репозиторий

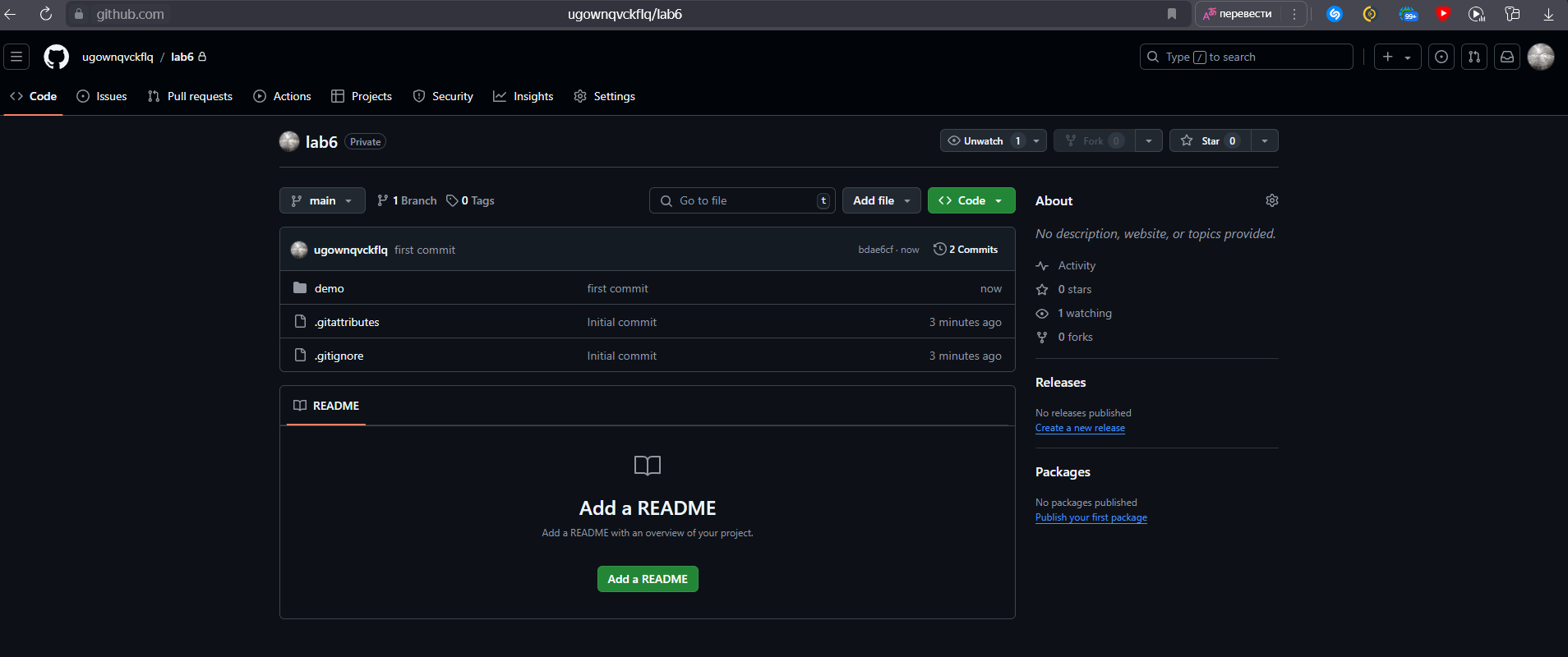


Рисунок 44 – Заполненный репозиторий GitHub

ЗАКЛЮЧЕНИЕ

Результатом выполнения данной учебной практики стала реализованная база данных по предметной области аренды автомобилей. Были разработаны формы для работы с этой базой данных, создано техническое задание и диаграммы, описывающие функционал работы приложения.

В СУБД были реализованы запросы для работы с базой данных, включая возможности добавления, редактирования, удаления и поиска записей. С помощью программы IntelliJ IDEA был создан удобный пользовательский интерфейс на основе Spring MVC, который позволяет пользователям выполнять различные действия с базой данных, а именно: бронирование автомобилей, продление аренды, управление данными пользователей и автомобилей, а также отслеживание сроков аренды.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Пирогов, В. Информационные системы и базы данных: организация и проектирование: Учебное пособие / В. Пирогов. - СПб. : BHV, 2009. - 528 c.
2. Перлова, О. Н. Проектирование и разработка информационных систем: Учебник / О. Н. Перлова. - М. : Академия, 2018. - 272 c.
3. Липаев, В.В. Сертификация программных средств / В.В. Липаев. - М.: Синтег, 2010. – 348 с.
4. Техническая документация [Офиц. сайт]. URL: <http://tdocs.su>
5. Соловьев, И. В. Проектирование информационных систем. Фундаментальный курс: Учебное пособие для высшей школы / И. В. Соловьев, А.А. Майоров; Под ред. В.П. Савиных. - М. : Академический проспект, 2009. - 398 c.
6. Гвоздева, Т. В. Проектирование информационных систем: технология автоматизированного проектирования. Лабораторный практикум. Учебно-справочное пособие / Т. В. Гвоздева, Б. А. Баллод. - СПб. : Лань, 2018. - 156 c.
7. Федоров, Н.В. Проектирование информационных систем на основе современных CASE-технологий / Н. В. Федоров. - М. : МГИУ, 2008. - 280 c
8. Мюллер, Р.Д. Проектирование баз данных и UML / Р.Д. Мюллер; Пер. с англ. Е.Н. Молодцова. - М.: Лори, 2013. - 420 c.
9. Карпов, В.Э. Об оформлении программной документации / В.Э. Карпов. [Электронный ресурс]. Режим доступа: <http://www.raai.org/about/persons/karpov/pages/ofdoc/ofdoc.html>
10. Баловсяк, Н. Видеосамоучитель создания реферата, курсовой, диплома на компьютере / Н. Баловсяк. – СПб: Питер, 2010. – 288 с.
11. MySQL [Электронный ресурс] — URL: <https://timeweb.cloud/blog/mysql-preimushchestva-i-nedostatki>

ПРИЛОЖЕНИЕ А

* package com.example.demo;  
    
  import com.example.demo.entity.User;  
  import com.example.demo.entity.Vehicle;  
  import com.example.demo.service.UserService;  
  import com.example.demo.service.VehicleService;  
  import lombok.RequiredArgsConstructor;  
  import org.junit.jupiter.api.Test;  
  import org.springframework.boot.test.context.SpringBootTest;  
    
  import static org.mockito.Mockito.*when*;  
    
  @SpringBootTest  
  @RequiredArgsConstructor  
  class DemoApplicationTests {  
    
   private final VehicleService vehicleService;  
   private final UserService userService;  
   private  
    
   @Test  
   void shouldStartRentalSuccessfully\_whenAllConditionsMet() {  
   // Arrange  
   User user = new User(1L, "user1", true); // зарегистрированный пользователь с активной картой  
   Vehicle vehicle = new Vehicle(1L, "Tesla Model S", true); // доступный автомобиль  
   *when*(vehicleService.isAvailable(vehicle.getId())).thenReturn(true);  
    
    
   // Act  
   boolean isRented = rentalService.startRental(user.getId(), vehicle.getId(), 120); // аренда на 120 минут  
    
   // Assert  
   assertTrue(isRented);  
   verify(vehicleService).markAsRented(vehicle.getId());  
   }  
    
   @Test  
   void shouldNotStartRental\_whenNoPaymentMethodIsAttached() {  
   // Arrange  
   User user = new User(2L, "user2", false); // пользователь без привязанной карты  
   Vehicle vehicle = new Vehicle(2L, "BMW X5", true);  
   *when*(paymentService.hasValidPaymentMethod(user.getId())).thenReturn(false);  
    
   // Act  
   boolean isRented = rentalService.startRental(user.getId(), vehicle.getId(), 60);  
    
   // Assert  
   assertFalse(isRented);  
   verify(vehicleService, never()).markAsRented(anyLong());  
   }  
    
   @Test  
   void shouldNotStartRental\_whenVehicleIsAlreadyRented() {  
   // Arrange  
   User user = new User(3L, "user3", true);  
   Vehicle vehicle = new Vehicle(3L, "Audi A4", false); // автомобиль уже арендован  
   *when*(vehicleService.isAvailable(vehicle.getId())).thenReturn(false);  
    
   // Act  
   boolean isRented = rentalService.startRental(user.getId(), vehicle.getId(), 90);  
    
   // Assert  
   assertFalse(isRented);  
   verify(paymentService, never()).processPayment(anyLong(), anyDouble());  
   }  
    
   @Test  
   void shouldEndRentalSuccessfully\_whenRentalIsActive() {  
   // Arrange  
   Rental rental = new Rental(1L, 1L, 1L, RentalStatus.ACTIVE);  
   *when*(rentalRepository.findById(rental.getId())).thenReturn(Optional.of(rental));  
    
   // Act  
   boolean isEnded = rentalService.endRental(rental.getId());  
    
   // Assert  
   assertTrue(isEnded);  
   verify(paymentService).chargeForRental(anyLong(), anyDouble());  
   verify(vehicleService).markAsAvailable(rental.getVehicleId());  
   }  
    
   @Test  
   void shouldFailToEndRental\_whenRentalDoesNotExist() {  
   // Arrange  
   *when*(rentalRepository.findById(1L)).thenReturn(Optional.empty());  
    
   // Act  
   boolean isEnded = rentalService.endRental(1L);  
    
   // Assert  
   assertFalse(isEnded);  
   verify(paymentService, never()).chargeForRental(anyLong(), anyDouble());  
   }  
    
    
   @Test  
   void shouldCalculateRentalCostCorrectly() {  
   // Arrange  
   Vehicle vehicle = new Vehicle(1L, "Tesla Model S", 10.0); // цена аренды за минуту  
   int durationMinutes = 120; // аренда на 120 минут  
    
   // Act  
   double cost = rentalService.calculateCost(vehicle.getId(), durationMinutes);  
    
   // Assert  
   assertEquals(1200.0, cost); // 10.0 \* 120  
   }  
    
    
   @Test  
   void shouldNotStartRental\_whenPaymentMethodIsExpired() {  
   // Arrange  
   User user = new User(4L, "user4", false); // у пользователя истекшая карта  
   Vehicle vehicle = new Vehicle(4L, "Mercedes-Benz", true);  
   *when*(paymentService.hasValidPaymentMethod(user.getId())).thenReturn(false);  
    
   // Act  
   boolean isRented = rentalService.startRental(user.getId(), vehicle.getId(), 30);  
    
   // Assert  
   assertFalse(isRented);  
   verify(vehicleService, never()).markAsRented(anyLong());  
   }  
    
    
   @Test  
   void shouldNotEndRental\_whenRentalIsAlreadyEnded() {  
   // Arrange  
   Rental rental = new Rental(1L, 1L, 1L, RentalStatus.ENDED);  
   *when*(rentalRepository.findById(rental.getId())).thenReturn(Optional.of(rental));  
    
   // Act  
   boolean isEnded = rentalService.endRental(rental.getId());  
    
   // Assert  
   assertFalse(isEnded);  
   verify(paymentService, never()).chargeForRental(anyLong(), anyDouble());  
   }  
    
   @Test  
   void shouldReturnAvailableVehicles() {  
   // Arrange  
   List<Vehicle> availableVehicles = List.of(  
   new Vehicle(1L, "Tesla Model S", true),  
   new Vehicle(2L, "BMW X5", true)  
   );  
   *when*(vehicleService.getAvailableVehicles()).thenReturn(availableVehicles);  
    
   // Act  
   List<Vehicle> result = vehicleService.getAvailableVehicles();  
    
   // Assert  
   assertEquals(2, result.size());  
   assertTrue(result.stream().allMatch(Vehicle::isAvailable));  
   }  
    
   @Test  
   void shouldNotStartRental\_whenUserIsNotAuthenticated() {  
   // Arrange  
   Vehicle vehicle = new Vehicle(5L, "Porsche 911", true);  
   *when*(authenticationService.isAuthenticated()).thenReturn(false); // пользователь не авторизован  
    
   // Act  
   boolean isRented = rentalService.startRental(5L, vehicle.getId(), 60);  
    
   // Assert  
   assertFalse(isRented);  
   verify(vehicleService, never()).markAsRented(anyLong());  
   }  
    
    
    
  }

**АТТЕСТАЦИОННЫЙ ЛИСТ**

**ПО УЧЕБНОЙ ПРАКТИКЕ УП.02.01 (ПО ПРОФИЛЮ СПЕЦИАЛЬНОСТИ)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/2 группы

Назаров Никита Дмитриевич

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «23» сентября 2024 г. по «05» октября 2024 г.

**Виды и качество выполнения работ**

|  |  |  |
| --- | --- | --- |
| **Виды выполненных работ обучающимся**  **во время практики** | **Объем работ, час.** | **Качество выполнения работ (оценка по пятибалльной системе)** |
| **Тема 1.** Участие в выработке требований к программному обеспечению. | 12 |  |
| **Тема 2.** Стадии проектирования программного обеспечения. Проектирование UML-диаграмм. | 15 |  |
| **Тема 3.** Конструирование пользовательского интерфейса. Разработка модулей программного обеспечения. | 25 |  |
| **Тема 4.** Тестирование и отладка программных модулей. Модульное тестирование. | 12 |  |
| **Тема 5.** Разработка программной документации. | 9 |  |

**Характеристика учебной/профессиональной деятельности обучающегося во время учебной практики (по профилю специальности):**

Общие и профессиональные компетенции, предусмотренные программой практики, освоены **/** не освоены.

(нужное подчеркнуть)

Итоговая оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель практики Хисамутдинова А.С. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.) (подпись)

Дата «05» октября 2024 г.