**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 42919/7 группы

Баширов Матвей Андреевич

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «28» октября 2024 г. по «09» ноября 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 42919/7 группы

Баширов Матвей Андреевич

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «28» октября 2024 г. по «09» ноября 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 1**

Задание выдал «28» октября 2024 г. \_\_\_\_\_\_\_\_\_\_\_\_\_ Хисамутдинова А.С.

(подпись) (Ф.И.О.)

Задание получил «28» октября 2024 г. \_\_\_\_\_\_\_\_\_\_\_\_ Баширов М.А. (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 42919/7 группы

Баширов Матвей Андреевич

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «28» октября 2024 г. по «09» ноября 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** |
| 28.10.2024 | Анализ предметной области. Анализ методов решения. Анализ и выбор средств. |  |
| 29.10.2024 | Анализ и выбор средств. Исследовательские работы. Разработка технического задания. |  |
| 30.10.2024 | UML. Проектирование диаграммы вариантов использования и последовательности. UML. Проектирование диаграммы вариантов активности и состояния. |  |
| 31.10.2024 | Моделирование структуры ПО. Проектирование инфологической и даталогической модели данных. |  |
| 01.11.2024 | Проектирование интерфейса пользователя. |  |
| 02.11.2024 | Разработка схемы алгоритма программного продукта. |  |
| 05.11.2024 | Разработка модели данных. Разработка программного продукта. Создание функциональности ПО. |  |
| 06.11.2024 | Модульное тестирование. Создание тестовых случаев. |  |
| 07.11.2024 | Создание тестовых случаев. Отладка программных модулей. |  |
| 08.11.2024 | Разработка документа «Руководство пользователя» в соответствии с ЕСПД. |  |
| 09.11.2024 | Разработка документа «Программа и методика испытаний» в соответствии с ЕСПД. Подготовка отчета. |  |

СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

В ходе практики была поставлена задача изучить ключевые аспекты разработки программного обеспечения, включая анализ предметной области, выбор методов решения и средств разработки.

Требуется провести анализ предметной области, выявить основные требования и ограничения, связанные с будущей системой. Необходимо разработать инфологическую модель данных, чтобы обеспечить ясное представление о структуре и взаимосвязях данных в системе. Проектирование интерфейса пользователя должно быть направлено на создание удобного и интуитивно понятного взаимодействия. Создание функциональности программного обеспечения включает модульное тестирование и разработку тестовых случаев, что обеспечит качество и надежность продукта.

# Описание предметной области

Главной задачей разрабатываемого модуля является оптимизация процесса приобретения музыкальных инструментов в магазинах сети, управление скидками для постоянных клиентов, а также сбор и анализ данных о продажах в различных точках.

Задача разработки создаваемого программного модуля состоит в создании удобного и эффективного инструмента для обеспечения более эффективного учета товарооборота, а также ведения отчетности и анализа продаж, что позволит принимать более обоснованные решения по ассортименту и маркетинговым активностям.

Основные функции и возможности модуля включают:

1. Оформление покупки: предусматривает создание сотрудником магазина записи с информацией о приобретаемых товарах и покупателе. Запись может содержать данные о товарах, такие как название, количество и данные о клиенте – ФИО, номер телефона, персональная скидка постоянного клиента (от суммы всех покупок > 10 000 рублей), а также информация о дате покупки и сумме всех приобретаемых товаров.

2. Система должна предоставлять возможность формирования отчетов, включающих в себя общую сумму продаж за выбранный период, количество проданных единиц по каждому товару.

3. Модуль должен обеспечивать гибкую систему разграничения доступа к функциям в зависимости от роли пользователя. Основные роли и их права доступа могут быть следующими: администратор (с полным доступом), продавец (с доступ к функциям оформления покупки, включая создание и редактирование записей о продажах) и менеджер по продажам (с доступом к аналитическим отчетам о продажах по всем товарам).

# Техническое задание

1. Общие сведения
   1. Наименование проекта: Разработка программного модуля по оптимизации процесса приобретения музыкальных инструментов
   2. Заказчик: Цапиев Роман
   3. Исполнитель: Баширов Матвей
2. Функциональные требования:
   1. Создание записей о покупке, содержащей следующие поля:

- Номер записи (ID)

- Дата покупки

- Название товара

- Количество

- ФИО клиента

- Номер телефона клиента

- Сумма заказа

* 1. Формирование отчета содержащий информацию о:

- Сумме продаж за выбранный период

- Количество единиц проданных товаров

Информация должна быть представлена в виде таблиц и графиков.

* 1. Возможность просмотра списка товаров, представленных в магазине в виде таблицы, поиск определенных товаров по ключевому полю

1. Нефункциональные требования:
   1. Кроссплатформенность:

- Поддержка работы на ОС семейства Windows.

* 1. Безопасность:

- Логин и пароль для доступа к приложению

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

* 1. Удобство использования:

- Простой и интуитивный интерфейс;

- Информативные уведомления и подсказки.

* 1. Производительность:

- Приложение должно иметь быстрый доступ к данным;

- Минимальное время отклика на запросы пользователя.

1. Требования к реализации:
   1. Язык программирования: C#
   2. СУБД: Microsoft SQL Server
2. Требования к документации:
   1. Техническое задание на разработку программного модуля
3. Руководство по стилю:
   1. Шрифт: Использовать шрифт Cassadia Mono, размер 14px. Заголовки должны быть выделены с помощью увеличенного размера (Первый уровень - 24px, Второй уровень - 20px).
   2. Цветовая схема:
      1. Цвет фона – #FFFFFF
      2. Цвет меню интерфейса – #7CBBFF
      3. Акцентные цвета для кнопок – #007BFF
      4. Основной цвет текста – #333333
      5. Цвет текста для кнопок – #FFFFFF
      6. Дополнительный цвет текста – #007BFF

# Разработка диаграмм базы данных

Создана база данных в СУБД Microsoft SQL Server для магазина музыкальных инструментов (Рисунок 1):



Рисунок 1 – ER-диаграмма базы данных

Ниже приведен скрипт для создания и заполнения таблиц БД:

CREATE TABLE CLIENTS (

ID INT IDENTITY PRIMARY KEY NOT NULL,

[NAME] VARCHAR(255) NOT NULL,

PHONE VARCHAR(12),

IS\_REGULAR BIT NOT NULL,

CONSTRAINT chk\_PhoneFormat CHECK (PHONE LIKE '+[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')

);

INSERT INTO CLIENTS ([NAME], PHONE, IS\_REGULAR) VALUES

('Иванов Иван Иванович', '+78982333131', 1),

('Петров Петр Петрович', '+79872236587', 0),

('Петров Петр Петрович', '+79872236587', 1),

('Григорьев Никанор Федорович', '+79123854561', 1),

('Лисовец Эдуард Макарович', '+79123854561', 0);

CREATE TABLE CATEGORIES (

ID INT IDENTITY PRIMARY KEY,

[NAME] VARCHAR(255) NOT NULL

);

INSERT INTO CATEGORIES ([NAME]) VALUES

('Инструменты'),

('Электроника'),

('Аксессуары');

CREATE TABLE PRODUCTS (

ID INT IDENTITY PRIMARY KEY NOT NULL,

[NAME] VARCHAR(255) NOT NULL,

PRICE MONEY NOT NULL,

QUANTITY INT NOT NULL,

CATEGORY INT,

FOREIGN KEY (CATEGORY) REFERENCES CATEGORIES(ID)

);

INSERT INTO PRODUCTS ([NAME], PRICE, QUANTITY, CATEGORY) VALUES

('Гитара «Highway»', 15000.00, 5, 1),

('Скрипка «Lonely»', 20000.00, 3, 1),

('Наушники «GreyDay»', 3000.00, 15, 2),

('Ударная установка «KickMachine»', 50000.00, 2, 1),

('Синтезатор «PianoRoll»', 30000.00, 4, 2),

('Микрофон «Loud&Louder»', 5000.00, 10, 3),

('Акустическая система «WooWhoo»', 25000.00, 6, 3);

CREATE TABLE SELLS (

ID INT IDENTITY PRIMARY KEY NOT NULL,

CLIENT INT NOT NULL,

SELL\_DATE DATE NOT NULL,

FOREIGN KEY (CLIENT) REFERENCES CLIENTS(ID)

);

INSERT INTO SELLS (CLIENT, SELL\_DATE) VALUES

(1, '2024-10-29'),

(2, '2024-10-29'),

(3, '2024-10-29');

CREATE TABLE SELL\_DETAILS (

ID INT IDENTITY PRIMARY KEY NOT NULL,

SELL INT NOT NULL,

PRODUCT INT NOT NULL,

QUANTITY INT NOT NULL,

FOREIGN KEY (SELL) REFERENCES SELLS(ID),

FOREIGN KEY (PRODUCT) REFERENCES PRODUCTS(ID)

);

INSERT INTO SELL\_DETAILS (SELL, PRODUCT, QUANTITY) VALUES

(1, 1, 2),

(1, 2, 1),

(2, 1, 3),

(2, 3, 1),

(3, 2, 5);

CREATE TABLE ROLES (

ID INT IDENTITY PRIMARY KEY NOT NULL,

[NAME] VARCHAR(50) NOT NULL,

[LOGIN] VARCHAR(50) NOT NULL,

[PASSWORD] VARCHAR(50) NOT NULL

);

INSERT INTO ROLES ([NAME], [LOGIN], [PASSWORD]) VALUES

('Admin', 'admin\_user', 'admin\_pass123'),

('Manager', 'manager\_user', 'manager\_pass456'),

('Employee', 'employee\_user', 'employee\_pass789');

После создания и заполнения таблицы в базе данных выглядят следующим образом (Рисунок 2-3):

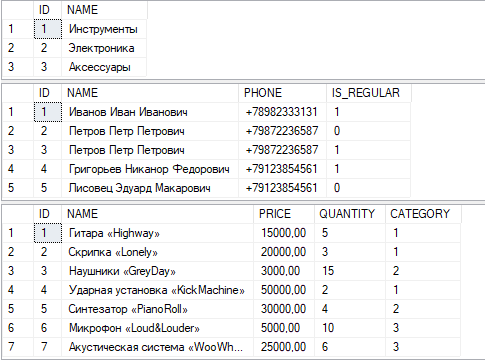


Рисунок 2 – Таблицы базы данных

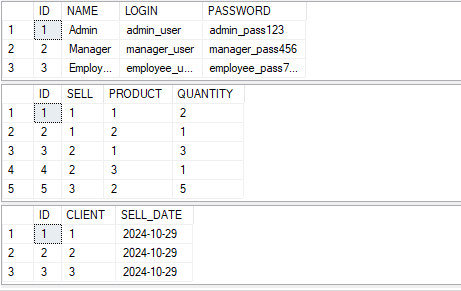


Рисунок 3 – Таблицы базы данных

Ниже приведен словарь данных для каждой добавленной таблицы:

Таблица 1 – «ROLES»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле таблицы | Тип данных | Ограничения | Описание |
| ID | INT | PRIMARY KEY, IDENTITY, NOT NULL | Уникальный идентификатор роли |
| NAME | VARCHAR(50) | NOT NULL | Название роли |
| LOGIN | VARCHAR(50) | NOT NULL | Логин для авторизации |
| PASSWORD | VARCHAR(50) | NOT NULL | Пароль для авторизации |

Таблица 2 – «SELLS»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле таблицы | Тип данных | Ограничения | Описание |
| ID | INT | PRIMARY KEY, IDENTITY, NOT NULL | Уникальный идентификатор продажи |
| CLIENT | INT | NOT NULL, FOREIGN KEY REFERENCES CLIENTS(ID) | Идентификатор покупателя |
| SELL\_DATE | DATE | NOT NULL | Дата продажи |

Таблица 3 – «SELL\_DETAILS»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле таблицы | Тип данных | Ограничения | Описание |
| ID | INT | PRIMARY KEY, IDENTITY, NOT NULL | Уникальный идентификатор детали продажи |
| SELL | INT | NOT NULL, FOREIGN KEY REFERENCES SELLS(ID) | Идентификатор продажи |
| PRODUCT | INT | NOT NULL, FOREIGN KEY REFERENCES PRODUCTS(ID) | Идентификатор товара |
| QUANTITY | INT | NOT NULL | Количество проданных единиц |

Таблица 4 – «CLIENTS»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле таблицы | Тип данных | Ограничения | Описание |
| ID | INT | PRIMARY KEY, IDENTITY, NOT NULL | Уникальный идентификатор покупателя |
| NAME | VARCHAR(255) | NOT NULL | Имя покупателя |
| PHONE | VARCHAR(12) |  | Телефон покупателя |
| IS\_REGULAR | BIT | NOT NULL | Признак постоянного клиента (1 - да, 0 - нет) |

Таблица 5 – «PRODUCTS»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле таблицы | Тип данных | Ограничения | Описание |
| ID | INT | PRIMARY KEY, IDENTITY, NOT NULL | Уникальный идентификатор товара |
| NAME | VARCHAR(255) | NOT NULL | Название товара |
| PRICE | MONEY | NOT NULL | Цена товара |
| QUANTITY | INT | NOT NULL | Количество товара |
| CATEGORY | INT | FOREIGN KEY REFERENCES CATEGORIES(ID) | Идентификатор категории товара |

Таблица 6 – «CATEGORIES»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле таблицы | Тип данных | Ограничения | Описание |
| ID | INT | PRIMARY KEY, IDENTITY | Уникальный идентификатор категории |
| NAME | VARCHAR(255) | NOT NULL | Название категории |

Создана диаграмма вариантов использования, в которой предусмотрены 3 роли со своими правами доступа – «Администратор», «Продавец», «Менеджер по продажам» (Рисунок 4):



Рисунок 4 – Диаграмма вариантов использования

Разработаны диаграммы последовательностей для создание администратором нового клиента и оформлением продавца новой покупки (Рисунок 5-6):



Рисунок 5 – Добавление клиента администратором



Рисунок 6 – Создание продавцом записи о новой продаже

Для процесса авторизации была разработана диаграмма активности (Рисунок 7):



Рисунок 7 – Процесс авторизации пользователей

# Разработка макетов приложения

Для создаваемого проекта разработал макеты интерфейса в нескольких вариациях. Ниже представлен макет Wireframe (Рисунок 8-11):

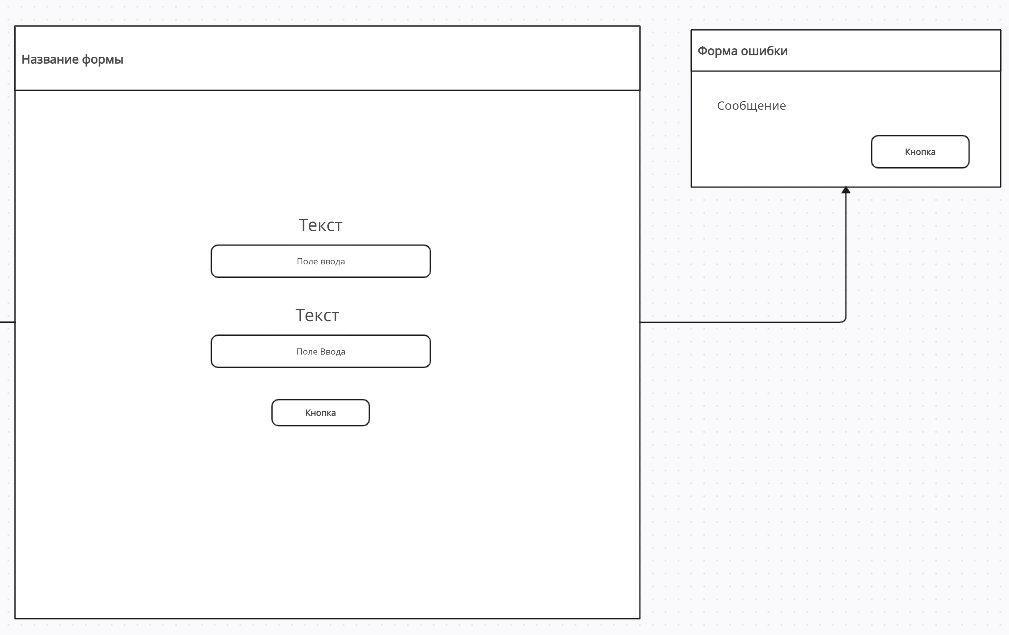


Рисунок 8 – Форма авторизации



Рисунок 9 – Форма пользовательского интерфейса администратора



Рисунок 10 – Форма отчетов



Рисунок 11 – Процесс авторизации пользователей

По разрабатываемому программному обеспечению составлен макет Mockup (Рисунок 12-14):

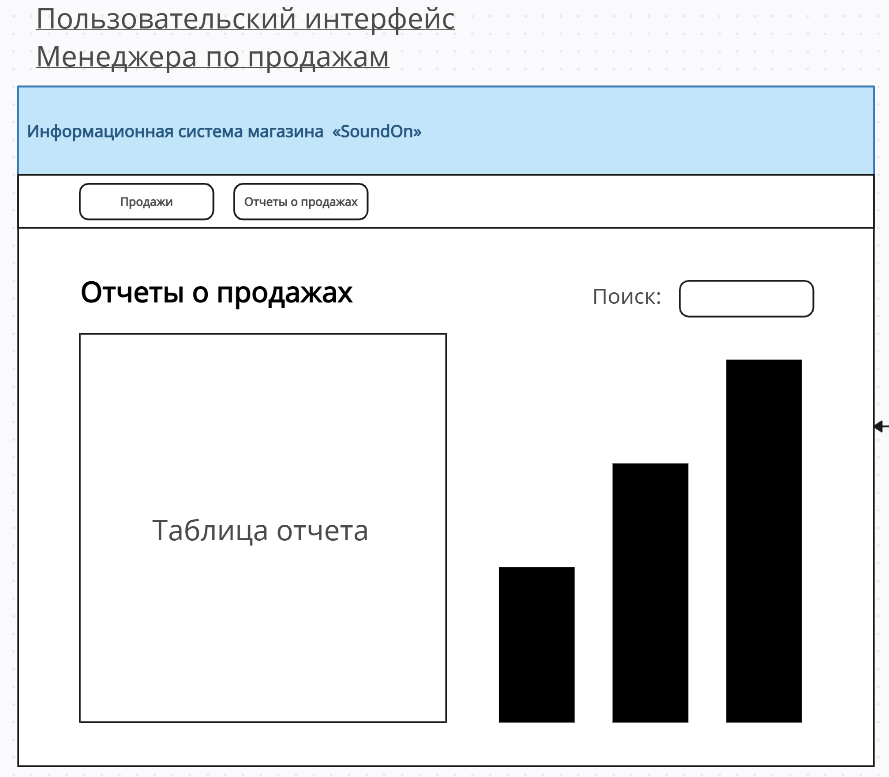


Рисунок 12 – Форма отчета Mockup

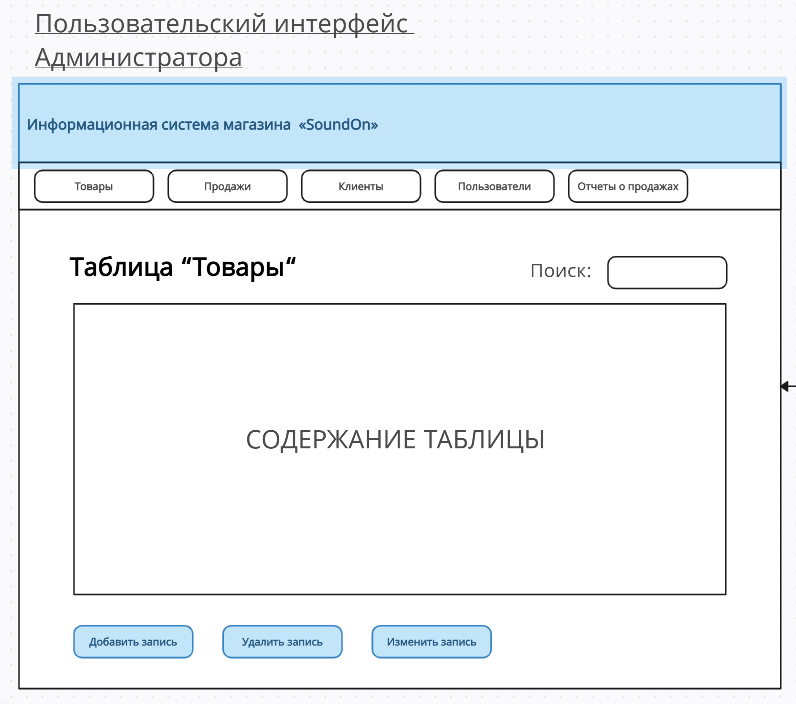


Рисунок 13 – Пользовательский интерфейс администратора Mockup

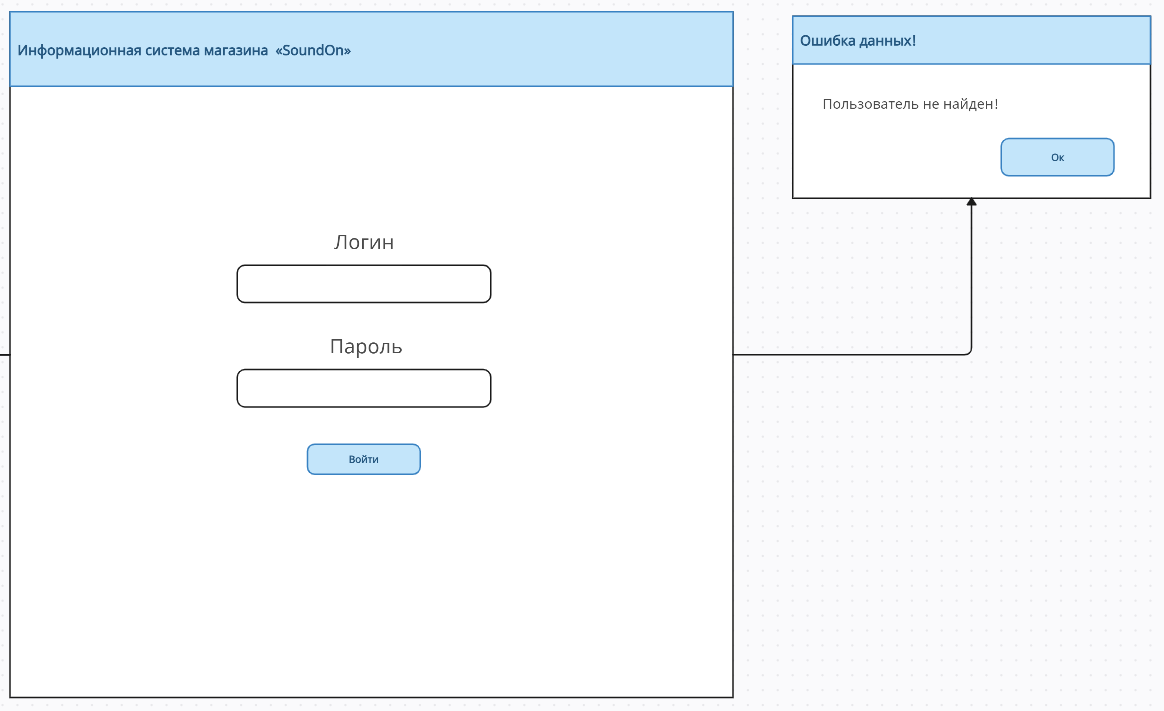


Рисунок 14 – Форма авторизации Mockup

Для лучшего понимания процесса работы с приложением пользователем была разработана карта навигации (Рисунок 15):



Рисунок 15 – Карта навигации приложения

# Разработка информационной системы

В ходе разработки программы по предоставленному техническому заданию были разработаны формы для авторизации, работами с таблицами и формы добавления и изменения записей (Рисунок 16-18).

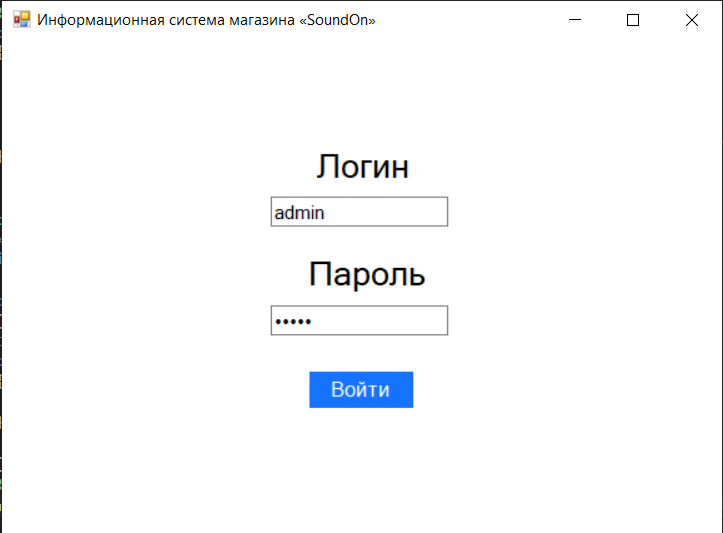


Рисунок 16 – Форма авторизации

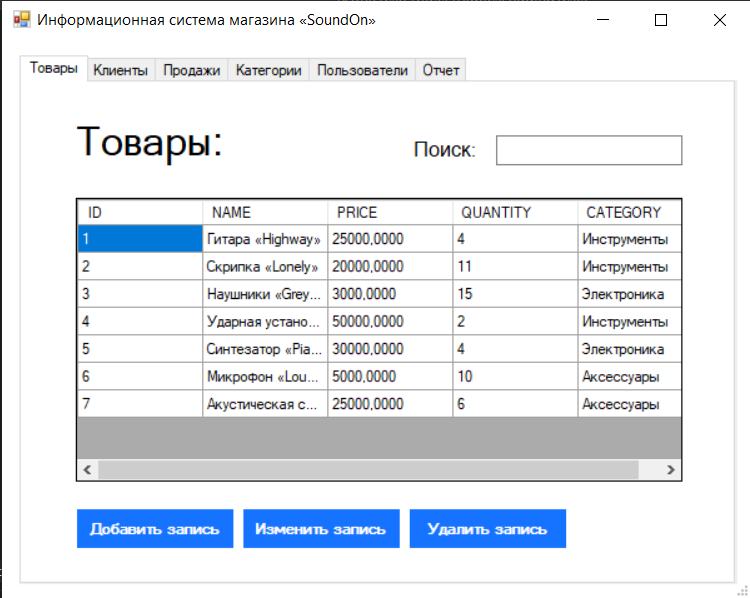


Рисунок 17 – Форма с таблицей товаров

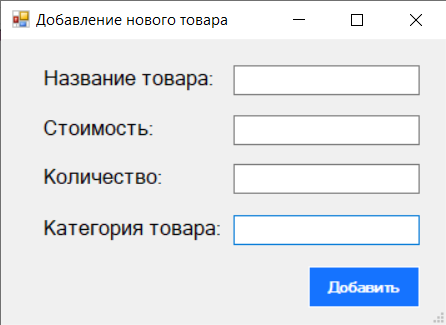


Рисунок 18 – Форма с таблицей товаров

Для форм добавления и обновления данных в таблицах были разработаны классы, отвечающие за сбор данных пользователя и добавления или обновления данных в соответствующих таблицах.

public partial class UpdateRoleForm : Form

{

private Database database = new Database();

private int roleId;

public string RoleName { get; private set; }

public string Login { get; private set; }

public string Password { get; private set; }

public UpdateRoleForm(int roleId, string roleName, string login, string password)

{

InitializeComponent();

this.roleId = roleId;

roleNameTextBox.Text = roleName;

loginTextBox.Text = login;

passwordTextBox.Text = password;

}

private void okButton\_Click(object sender, EventArgs e)

{

RoleName = roleNameTextBox.Text;

Login = loginTextBox.Text;

Password = passwordTextBox.Text;

if (string.IsNullOrWhiteSpace(RoleName) || string.IsNullOrWhiteSpace(Login) || string.IsNullOrWhiteSpace(Password))

{

MessageBox.Show("Пожалуйста, заполните все поля.");

return;

}

try

{

string updateQuery = "UPDATE ROLES SET NAME = @Name, LOGIN = @Login, PASSWORD = @Password WHERE ID = @Id";

database.openConnection();

SqlCommand command = new SqlCommand(updateQuery, database.getConnection());

command.Parameters.AddWithValue("@Name", RoleName);

command.Parameters.AddWithValue("@Login", Login);

command.Parameters.AddWithValue("@Password", Password);

command.Parameters.AddWithValue("@Id", roleId);

command.ExecuteNonQuery();

database.closeConnection();

DialogResult = DialogResult.OK;

this.Close();

}

catch (SqlException ex)

{

MessageBox.Show($"Ошибка при обновлении роли: {ex.Message}");

database.closeConnection();

}

catch (Exception ex)

{

MessageBox.Show($"Произошла ошибка: {ex.Message}");

database.closeConnection();

}

}

}

Для формы с отчетом был разработан код, который получает данные из таблиц базы данных, обрабатывает их и вносит соответствующие данные в таблицу отчета (Рисунок 19) и создает диаграмму.

private void GenerateSalesReport(DateTime startDate, DateTime endDate)

{

database.openConnection();

string query = @"

SELECT

P.NAME AS ProductName,

SUM(SD.QUANTITY) AS TotalQuantity,

SUM(SD.QUANTITY \* P.PRICE) AS TotalSales

FROM

SELL\_DETAILS SD

JOIN

SELLS S ON SD.SELL = S.ID

JOIN

PRODUCTS P ON SD.PRODUCT = P.ID

WHERE

S.SELL\_DATE BETWEEN @StartDate AND @EndDate

GROUP BY

P.NAME";

using (SqlCommand command = new SqlCommand(query, database.getConnection()))

{

command.Parameters.AddWithValue("@StartDate", startDate);

command.Parameters.AddWithValue("@EndDate", endDate);

using (SqlDataReader reader = command.ExecuteReader())

{

// Создаем DataTable для хранения результатов

DataTable salesData = new DataTable();

salesData.Load(reader);

// Отображаем данные в DataGridView

dataGridView1.DataSource = salesData;

// Подготовка данных для диаграммы

chart1.Series.Clear();

chart1.Series.Add("Sales");

chart1.Series["Sales"].ChartType = SeriesChartType.Pie;

foreach (DataRow row in salesData.Rows)

{

string productName = row["ProductName"].ToString();

int totalQuantity = Convert.ToInt32(row["TotalQuantity"]);

chart1.Series["Sales"].Points.AddXY(productName, totalQuantity);

}

}

}

database.closeConnection();

}

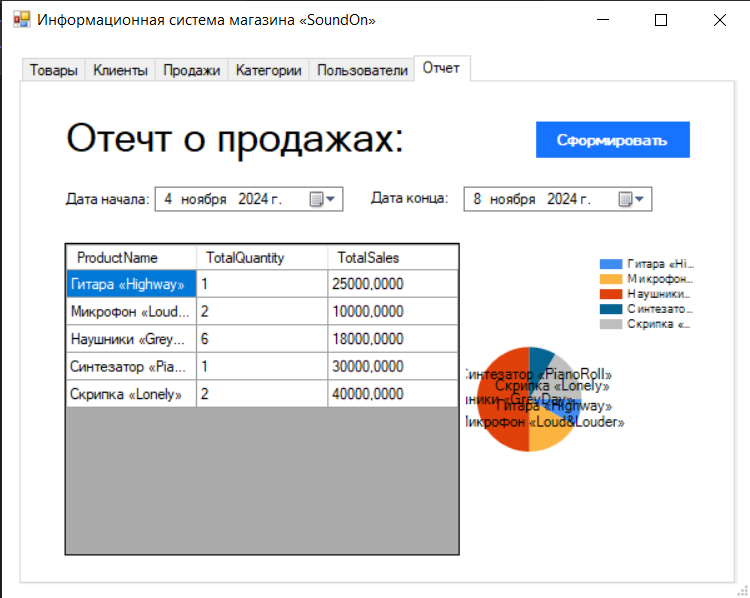


Рисунок 19 – Форма отчета по продажам

# Тестирование приложения

Для созданного приложения были разработаны соответствующие Тест-кейсы и Unit-тесты, чтобы проверить правильность работы отдельных модулей и оценить корректность обработки тех или иных данных в тестовых случаях. Ниже представлены результаты тестовых испытаний (Таблица 7-10).\

Таблица 7 – Тест кейс авторизации №1

|  |  |
| --- | --- |
| **Название проекта** | Информационная система магазина музыкальных инструментов «SoundOn» |
| **Рабочая версия** | V 1.0 |
| **Имя тестирующего** | Матвей Баширов |
| **Дата(ы) теста** | 07.112024 |

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_Authorization\_1 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Вход в информационную систему с действительным логином и паролем. |
| **Краткое изложение теста** | Пользователь на форме авторизации заполняет поля логин и пароль действительными данными и совершает вход в систему |
| **Этапы теста** | 1. Заполнить поля ввода «Логин» и «Пароль» корректными данными для авторизации 2. Нажать кнопку входа 3. Проверить успешность авторизации 4. Вернуться к форме авторизации |
| **Тестовые данные** | 1. Логин пользователя «Manager» - manager 2. Пароль пользователя «Manager» - manager |
| **Ожидаемый результат** | Успешная авторизация, переключение на основную форму приложения. |
| **Фактический результат** | По данному логину и паролю происходит успешная авторизация, пользователь видит рабочую область в соответствии с правами доступа. |
| **Предварительное условие** | Отсутствует |
| **Постусловие** | Пользователь находится в основном окне приложения, ограниченное по правам доступа |
| **Статус** | Зачет |
| **Примечания/комментарии** | Отсутствует |

Таблица 8 – Тест кейс авторизации №2

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_Authorization\_2 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Вход в информационную систему с недействительным логином и паролем. |
| **Краткое изложение теста** | Пользователь на форме авторизации заполняет поля логин и пароль недействительными данными и совершает вход в систему |
| **Этапы теста** | 1. Заполнить поля ввода «Логин» и «Пароль» некорректными данными для авторизации 2. Нажать кнопку входа 3. Проверить отображение сообщения об ошибке |
| **Тестовые данные** | 1. Недействительный логин – login 2. Недействительный пароль – password |
| **Ожидаемый результат** | Отображение сообщения об ошибке: «Введены неверный логин или пароль!» |
| **Фактический результат** | При вводе некорректного логина и пароля, пользователь видит сообщение об ошибке при входе, авторизация не происходит |
| **Предварительное условие** | Отсутствует |
| **Постусловие** | Отображено сообщение об ошибке |
| **Статус** | Зачет |
| **Примечания/комментарии** | Отсутствует |

Таблица 9 – Тест кейс добавления данных №1

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_DataAdd\_1 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Добавление данных в таблицу «Товары» с корректными данными |
| **Краткое изложение теста** | Пользователь на форме добавления записи заполняет необходимые поля недействительными данными и добавляет новую запись в таблицу |
| **Этапы теста** | 1. Открыть форму добавления записи в таблицу "Товары" 2. Заполнить обязательные поля корректными данными:   - Название товара (string)  - Стоимость (decimal)  - Количество (int)  - Категория (int)   1. Нажать кнопку «Добавить» 2. Проверить отображение сообщения об успешном добавлении записи 3. Проверить наличие добавленной записи в таблице "Товары" |
| **Тестовые данные** | 1. Название товара – Ксилофон 2. Стоимость – 50 000 3. Количество – 3 4. Категория – 1 |
| **Ожидаемый результат** | Запись успешно добавлена в таблицу, все введенные данные корректно отображаются в таблице |
| **Фактический результат** | При вводе данных в поля для добавления записи и нажатии кнопки «Добавить», пользователь видит сообщение об успешном добавлении, данные в таблице «Товары» соответствует добавленным |
| **Предварительное условие** | Пользователь авторизован с правами администратора, открыта форма «Товары» |
| **Постусловие** | При необходимости удалить тестовые данные после завершения теста |
| **Статус** | Зачет |
| **Примечания/комментарии** | Отсутствует |

Таблица 10 – Тест кейс добавления данных №2

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_DataAdd\_2 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Добавление данных в таблицу «Товары» с некорректными данными |
| **Краткое изложение теста** | Пользователь на форме добавления записи заполняет необходимые поля недействительными данными и пытается добавить новую запись в таблицу. |
| **Этапы теста** | 1. Открыть форму добавления записи в таблицу "Товары" 2. Заполнить обязательные поля некорректными данными 3. Нажать кнопку «Добавить» 4. Проверить отображение сообщения об ошибке, указывающего на некорректные данные 5. Проверить, что запись не добавлена в таблицу "Товары" |
| **Тестовые данные** | 1. 1 Название товара – (пустое значение) 2. Стоимость – 100 3. Количество – "три" 4. Категория – 9999 |
| **Ожидаемый результат** | Отображается сообщение об ошибке, указывающее на некорректные данные. Запись не добавляется в таблицу "Товары". |
| **Фактический результат** | При вводе некорректных данных в поля для добавления записи и нажатии кнопки «Добавить», пользователь видит сообщение об ошибке добавления, введенные данные в таблице «Товары» отсутствуют |
| **Предварительное условие** | Пользователь авторизован с правами администратора, открыта форма «Товары» |
| **Постусловие** | Отсутствует |
| **Статус** | Зачет |
| **Примечания/комментарии** | Отсутствует |

Таблица 11 – Тест кейс удаления данных №1

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_Delete\_1 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Удаление записи из таблицы «Товары», не использующейся в других отношениях |
| **Краткое изложение теста** | Пользователь выбирает запись в таблице «Товары» и пытается удалить её. |
| **Этапы теста** | 1. Найти и выбрать запись, которую необходимо удалить 2. Нажать кнопку «Удалить» 3. Проверить отображение сообщения об успешном удалении записи 4. Проверить, что запись больше не отображается в таблице «Товары» |
| **Тестовые данные** | 1. Товар для удаления с названием «Ксилофон» |
| **Ожидаемый результат** | Отображается сообщение об успешном удалении записи. Запись с названием «Ксилофон» больше не отображается в таблице «Товары». |
| **Фактический результат** | При выборе строки товара «Ксилофон» и нажатии на кнопку «Удалить» пользователь видит сообщение об успешном удалении записи. |
| **Предварительное условие** | Пользователь авторизован с правами администратора, открыта форма «Товары» |
| **Постусловие** | В таблице «Товары» больше нет записи о товаре «Ксилофон» |
| **Статус** | Зачет |
| **Примечания/комментарии** | Отсутствует |

**Разработанные Unit-тесты:**

Класс тестов для проверки строки соединения с БД:

[TestClass]

public class SQlConnectionTest

{

[TestMethod]

public void GetConnection\_ReturnsSqlConnection()

{

var database = new Database();

SqlConnection \_connection = database.getConnection();

Assert.IsNotNull(\_connection, "Соединение не должно иметь пустого значения");

Assert.IsInstanceOfType(\_connection, typeof(SqlConnection), "Соедиенение должно иметь тип SqlConnection");

}

}

Класс тестов для проверки корректности отображения форм приложения:

[TestClass]

public class FormTests

{

private ShopBase \_form;

[TestInitialize]

public void Form\_Init()

{

\_form = new ShopBase("Admin");

}

[TestMethod]

public void SetTabPageAccess\_Admin\_AddsAllTabs()

{

\_form.SetTabPageAccess("Admin");

Assert.AreEqual(6, \_form.GetTabCount(), "Администратор должен иметь доступ ко всем вкладкам");

}

[TestMethod]

public void SetTabPageAccess\_Manager\_AddsManagerTabs()

{

\_form.SetTabPageAccess("Manager");

Assert.IsNotNull(\_form.GetTabPageByName("tabPage3"), "tabPage3 должна отобразиться для Менеджера");

Assert.IsNotNull(\_form.GetTabPageByName("tabPage6"), "tabPage6 должна отобразиться для Менеджера");

}

[TestMethod]

public void SetTabPageAccess\_Employee\_AddsEmployeeTabs()

{

\_form.SetTabPageAccess("Employee");

Assert.IsNotNull(\_form.GetTabPageByName("tabPage1"), "tabPage1 должна отобразиться для Продавца");

Assert.IsNotNull(\_form.GetTabPageByName("tabPage3"), "tabPage3 должна отобразиться для Продавца");

Класс тестов для проверки правильности получения таблиц из БД:

[TestClass]

public class TablesLoadingTests

{

private ShopBase database;

[TestInitialize]

public void Setup()

{

database = new ShopBase("Admin");

}

[TestMethod]

public void Products\_Loading() {

database.Products\_Load();

var bindingSourceData = (DataTable)database.productsBindingSource.DataSource;

Assert.AreEqual(7, bindingSourceData.Rows.Count, "Количество записей не соответствует количеству в исходной таблице БД");

Assert.AreEqual("Гитара «Highway»", bindingSourceData.Rows[0]["NAME"], "Несоответствие названия первого продукта");

Assert.AreEqual("Скрипка «Lonely»", bindingSourceData.Rows[1]["NAME"], "Несоответствие названия второго продукта");

}

[TestMethod]

public void Clients\_Loading()

{

database.Clients\_Load();

var bindingSourceData = (DataTable)database.clientsBindingSource.DataSource;

Assert.AreEqual(5, bindingSourceData.Rows.Count, "Количество записей не соответствует количеству в исходной таблице БД");

Assert.AreEqual("Иванов Иван Иванович", bindingSourceData.Rows[0]["NAME"], "Несоответствие названия первого продукта");

Assert.AreEqual("Петров Петр Петрович", bindingSourceData.Rows[1]["NAME"], "Несоответствие названия второго продукта");

}

[TestMethod]

public void SalesData\_Loading()

{

database.LoadSalesData();

var bindingSourceData = (DataTable)database.salesBindingSource.DataSource;

Assert.AreEqual(7, bindingSourceData.Rows.Count, "Количество записей не соответствует количеству в исходной таблице БД");

Assert.AreEqual("Иванов Иван Иванович", bindingSourceData.Rows[0]["ClientName"], "Несоответствие данных первого клиента");

Assert.AreEqual("Петров Петр Петрович", bindingSourceData.Rows[1]["ClientName"], "Несоответствие данных идентификатора второго клиента");

}

}

Класс тестов для проверки авторизации пользователей:

[TestClass]

public class AuthorizationTests

{

Database database = new Database();

[TestInitialize]

public void Setup() {

database.openConnection();

var command = new SqlCommand("INSERT INTO ROLES ([NAME],LOGIN, PASSWORD) VALUES ('TestUser', 'testUser', 'testPassword')", database.getConnection());

command.ExecuteNonQuery();

}

[TestCleanup]

public void Cleanup()

{

var command = new SqlCommand("DELETE FROM ROLES WHERE LOGIN = 'testUser'", database.getConnection());

command.ExecuteNonQuery();

database.closeConnection();

}

[TestMethod]

public void IsUser\_ActualUser\_Test()

{

bool result = Login.IsUser("testUser", "testPassword", database.getConnection());

Assert.IsTrue(result, "Пользователь не принадлежит базе данных");

}

[TestMethod]

public void IsUser\_WrongPass\_Test()

{

bool result = Login.IsUser("testUser", "wrongPassword", database.getConnection());

Assert.IsFalse(result, "Пароль для пользователя указан верно");

}

[TestMethod]

public void IsUser\_NoUser\_Test()

{

bool result = Login.IsUser("noUser", "noPassword", database.getConnection());

Assert.IsFalse(result, "Пользователь принадлежит базе данных");

}

}

Результаты тестирования подтвердили правильность работы всех тестируемых методов (Рисунок 20):

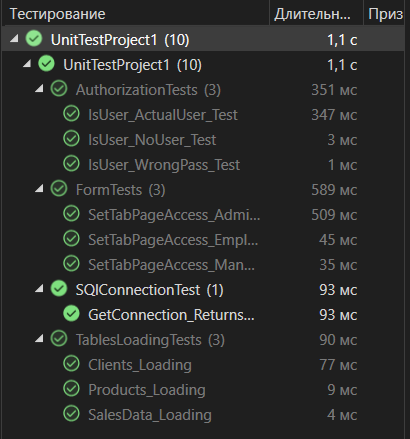


Рисунок 20 – Прохождение Юнит-тестов

# Загрузка проекта на удаленный репозиторий

Созданная программа и программная документация была загружена на удаленный репозиторий GitHub (Рисунок 21).