|  |
| --- |
| 09.02.07 ПР-421  **ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ**  **ПМ.11 Разработка, администрирование и защита баз данных** |
| **ПП.11.01 Разработка приложения** |

|  |  |  |  |
| --- | --- | --- | --- |
| Практикант |  |  | Бородин Д.Ю. |
| Руководитель практики |  |  | **Бабикова Т.М.** |
|  |  |  |  |

Оглавление

[ПП.11.01 Разработка приложения 3](#_Toc185633546)

[**База практики** 3](#_Toc185633547)

[**Период прохождения практики** 3](#_Toc185633548)

[**Основная задача практики** 3](#_Toc185633549)

[**Виды работ** 3](#_Toc185633550)

[**Задание на разработку** 4](#_Toc185633551)

[Описание организации данных 7](#_Toc185633552)

[**Логическая модель данных** 7](#_Toc185633553)

[**Схема данных** 7](#_Toc185633554)

[**Описание таблиц** 7](#_Toc185633555)

[**Описание интерфейса** 10](#_Toc185633556)

[**Описание логики приложения** 14](#_Toc185633557)

[**Описание отчетов** 20](#_Toc185633558)

[**Заключение** 22](#_Toc185633559)

[Используемые источники информации 23](#_Toc185633560)

**ПП.11.01 Разработка приложения**

**База практики**

Практика проводилась на базе полигона вычислительной техники БПОУ ОО «Омский авиационный колледж имени Н.Е. Жуковского»

**Период прохождения практики**

Практика Разработка приложения по разработке программных продуктов проходила с **16.12.2024 г.** по **21.12.2024 г.** на полигоне вычислительной техники Омавиат.

**Основная задача практики**

Основная задача практики - формирование практических навыков по разработке кода программного продукта на основе готовой спецификации на уровне модуля, проведения тестирования программного модуля по определенному сценарию, использования основных методологий процессов разработки программного обеспечения, работы с объектами базы данных в конкретной СУБД.

**Виды работ**

* Системный анализ и проектирование;
* Разработка программного обеспечения;
* Интеграция программных модулей;
* Документирование и презентация решений.

**Задание на разработку**

**Вид работы 1. Системный анализ и проектирование**

На основе описания предметной области Вам необходимо:

* **спроектировать ER-диаграмму.** Обязательна 3 нормальная форма с обеспечением ссылочной целостности. При разработке диаграммы обратите внимание на согласованную осмысленную схему именования, создайте необходимые первичные и внешние ключи, определите ограничения внешних ключей, отражающие характер предметной области. Создайте все необходимые сущности, определите отношения, создайте ограничения на связи между сущностями (при наличии всех связей), приведите базу данных к 3НФ (при наличии всех сущностей и связей).

ER- диаграмма должна быть представлена в формате PDF и содержать таблицы, связи между ними, атрибуты и ключи (типами данных на данном этапе можно пренебречь).

* **спроектировать интерфейс программного модуля** по составленному алгоритму в среде разработки в соответствии технического задания. Все компоненты системы должны иметь единый согласованный внешний вид, **соответствующий руководству по стилю**, а также следующим требованиям:

− последовательный пользовательский интерфейс, позволяющий перемещаться между существующими окнами в приложении (в том числе обратно, например, с помощью кнопки «Назад»).

* **спроектировать алгоритм решения одной из функций приложения в виде блок-схемы**. На основе анализа технического задания составить краткую спецификацию разрабатываемого модуля выделить входные и выходные данные; сформировать основной алгоритм решения в виде блок-схемы в соответствии с техническим заданием. Детализировать в виде алгоритма одну из функций (по описанию предметной области).

Алгоритмы представить одним в виде блок-схемы выполнить по правилам, установленным ГОСТ 19.701.

**Вид работы 2. Разработка программного обеспечения**

* **Разработка базы данных.**

В СУБД SQL Server необходимо создать все таблицы в соответствии с ER-диаграммой, настроить связи между ними и разработать схему данных. Сформировать скрипт базы данных.

* **Импорт данных в базу данных.**

Воспользовавшись файлами для импорта заполнить базу данных, где необходимо отредактировать данные, исключить их дублирование.

* **Разработка модуля авторизации.**

Программное решение должно быть многопользовательским. Авторизация пользователя осуществляется по логину и паролю. Пароли должны обязательно храниться в зашифрованном виде.

* Шифрование паролей должно производиться на стороне ПРИЛОЖЕНИЯ.
* Шифрование паролей происходит с помощью хеширования, с использованием алгоритма SHA256.
* В файлах для импорта логин и пароль пользователя совпадают.

При первом неверном вводе пароля генерируется CAPTCHA из 10 символов (цифры и буквы английского алфавита). Разработан функционал для пересоздания CAPTCHA;

После 3х неудачных попыток ввода пароля, окно авторизации блокируется на 30 секунд. Данная блокировка действует в случае перезапуска приложения.

После каждой неудачной попытке ввода пароля отображается окно с информацией – сколько у пользователя осталось попыток до блокировки.

* **Разработка интерфейса меню и основных форм с данными.**

Меню должно содержать пункты для перехода на следующие окна:

* Категории продукции
* Продукция
* Платежи пользователя за продукцию
* Отчет

Заголовок окна должен соответствовать назначению. Следует установить иконку приложения, если это реализуемо в рамках платформы, и логотип компании на главной форме.

Реализовать обработку исключительных ситуаций в приложении. Необходимо уведомлять пользователя о совершаемых им ошибках или о запрещенных в рамках задания действиях, предупреждать о неотвратимых операциях. Окна сообщений соответствующих типов (например, ошибка, предупреждение, информация) должны отображаться с соответствующим заголовком и пиктограммой. Текст сообщения должен быть полезным и информативным, содержать полную информацию о совершенных ошибках пользователя и порядок действий для их исправления. Также можно использовать визуальные подсказки для пользователя при вводе данных.

* **Организовать подключение приложения к созданной базе данных.**

Каждое из окон должно содержать полный список выводимой информации из БД. Данные должны помещаться на один экран по ширине. В окне Платежи выводятся только данные текущего авторизованного пользователя.

* **Разработка форм добавления/редактирования данных.**

Переходы на эту форму должны быть реализованы из каждой основной формы списка данных: при нажатии соответствующих кнопок.

На формах добавления предусмотреть выпадающие списки для загрузки и выбора связанных данных. При открытии формы редактирования все поля выбранного объекта должны быть подгружены из базы данных. При сохранении вся информация сохраняется в базу данных, а списки/таблицы с данными основных форм должны обновляться актуальными значениями.

* **Разработка логики поиска и фильтрации данных.**

Данные по продукции можно фильтровать по категориям. При отсутствии выбранной категории (или установки фильтра – Все категории) отображается вся продукция.

Данные по продукции можно искать по названию (контекстный живой поиск).

Реализовать одновременное применение этих двух фильтров.

Данные по платежам можно отображать за выбранный период (от даты до даты, либо с указанием только одной из дат). Данные по платежам можно фильтровать по категориям.

Реализовать по платежам одновременное применение этих двух фильтров.

**Вид работы 3. Интеграция программных решений**

* **Реализовать алгоритм подсчета итоговой суммы платежа и скидки.**

Для каждого платежа необходимо вести подсчет итоговой суммы как цена продукции \* количество продукции. Стоимость пересчитывается в реальном времени при изменении цены или количества.

Реализовать алгоритм функции расчета скидки, предоставляемой по платежу пользователя.

Величина скидки для пользователя рассчитывается на основании объема оплаченной продукции за весь период работы. Скидка зависит от общего количества оплаченной пользователем продукции и составляет: до 10000 – 0%, от 10000 – до 50000 – 5%, от 50000 – до 300000 – 10%, более 300000 – 15%.

Необходимо использовать комментарии для пояснения неочевидных фрагментов кода. Комментарии должны присутствовать только в местах, которые требуют дополнительного пояснения.

* **Реализовать отображение истории платежей с подведением итогов в виде отчета.**

Отчет может выводиться встроенными средствами, либо в распространенных программных продуктах (Microsoft Office Word, Microsoft Office Excel, Internet-браузер);

У отчета обязательно должен быть заголовок на первой странице. В заголовке может быть указан период и категория платежей. Отчет может быть сформирован в виде списка (см. макет). На каждой странице указывается ФИО пользователя и номер страницы. Платежи группируются по категориям. Внутри каждой категории платежи сортируются по дате. В конце отчета выводится суммарная стоимость всех показанных платежей.

**Вид работы 4. Документирование и презентация решений**

* **Работа с системой контроля версий.**

Необходимо загрузить исходный код приложения в отдельный репозиторий с названием, совпадающим с названием проекта.

Все практические результаты должны быть переданы путем загрузки файлов на предоставленный репозиторий системы контроля версий.

Практические результаты:

– исходный код приложения (структура с файлами, не архив);

– исполняемые файлы;

– файл скрипта базы данных;

– прочие графические/текстовые файлы.

* **Оформление отчета по практике в соответствии с макетом.**

Отчет должен быть распечатан и сдан в бумажном виде.

Описание организации данных

**Логическая модель данных**

Описание основных объектов разрабатываемой системы:

Таблица 1 - Описание основных объектов системы

|  |  |
| --- | --- |
| Объект | Описание |
| Пользователь | Информация о пользователях системы |
| Категория | Информация о категориях продуктов |
| Платеж | Информация о платежах пользователей |
| Продукт | Информация о продуктах |

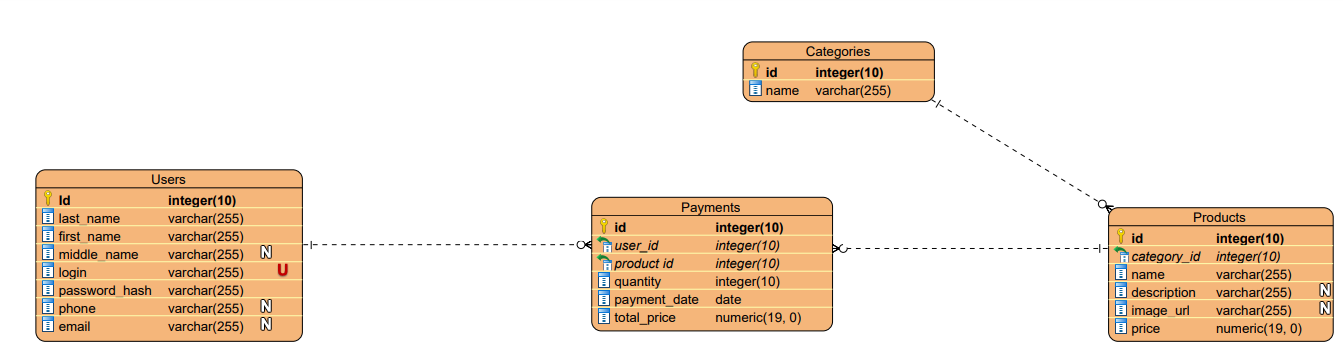


Рисунок 1 - Erd диаграмма базы данных

**Схема данных**

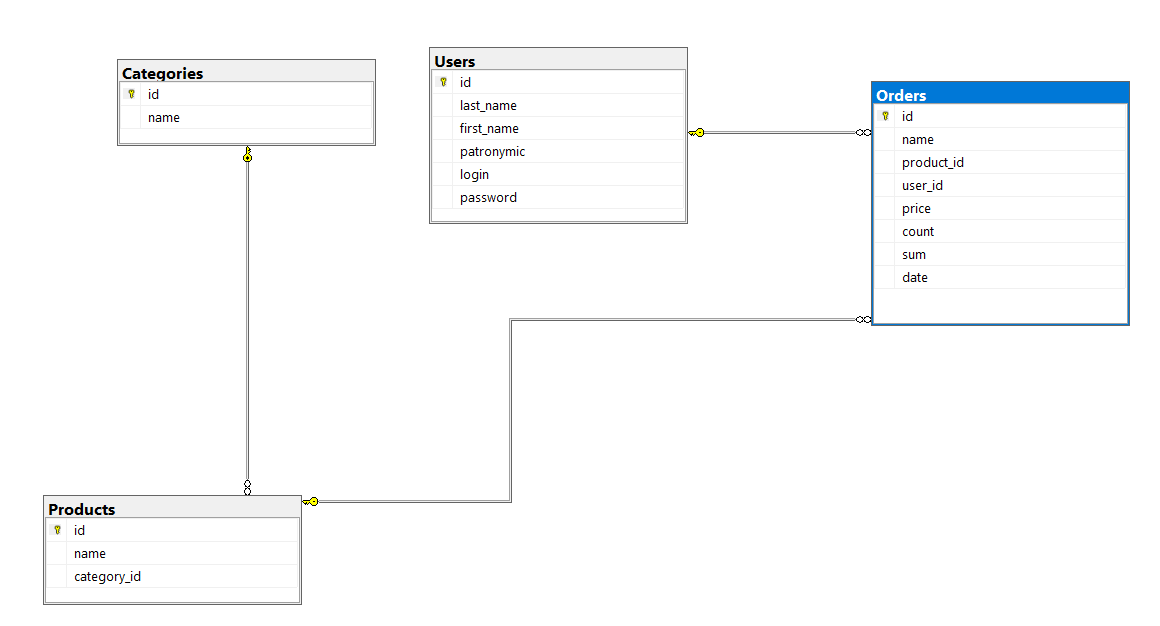


Рисунок 2 - Схема данных из СУБД

**Описание таблиц**

Таблица Users

Таблица 2 - Users

| **Название поля** | **Описание** | **Тип значения/поля** | **Кол-во знаков** |
| --- | --- | --- | --- |
| Id | Уникальный идентификатор пользователя | INT | - |
| login | Логин пользователя | VARCHAR | 50 |
| password | Пароль пользователя | VARCHAR | 50 |
| first\_name | Имя пользователя | VARCHAR | 50 |
| last\_name | Фамилия пользователя | VARCHAR | 50 |
| patronymic | Отчество пользователя | VARCHAR | 50 |

Таблица Orders

Таблица 3 - Orders

| **Название поля** | **Описание** | **Тип значения/поля** | **Кол-во знаков** |
| --- | --- | --- | --- |
| Id | Уникальный идентификатор заказа | INT | - |
| user\_id | Идентификатор пользователя | INT | - |
| product\_id | Идентификатор продукта | INT | - |
| count | Количество продукта в заказе | INT | - |
| sum | Общая сумма заказа | DECIMAL | 10,2 |
| date | Дата создания заказа | DATE | - |

Таблица Products

Таблица 4 - Таблица Products

| **Название поля** | **Описание** | **Тип значения/поля** | **Кол-во знаков** |
| --- | --- | --- | --- |
| Id | Уникальный идентификатор продукта | INT | - |
| name | Название продукта | VARCHAR | 100 |
| category\_id | Идентификатор категории продукта | INT | - |
| price | Цена продукта | DECIMAL | 10,2 |

Таблица Categories

Таблица 5 - Таблица Cetegories

| **Название поля** | **Описание** | **Тип значения/поля** | **Кол-во знаков** |
| --- | --- | --- | --- |
| Id | Уникальный идентификатор категории | INT | - |
| name | Название категории | VARCHAR | 100 |

**Описание интерфейса**

Форма авторизации:

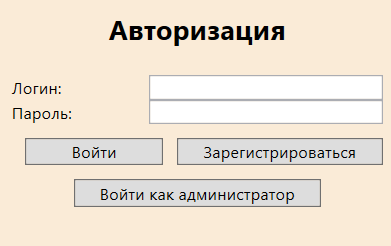


Рисунок 3 - Форма авторизации

Назначение формы: Форма авторизации предназначена для входа пользователей в систему. Пользователь вводит логин и пароль, после чего система проверяет их в базе данных. Если данные корректны, пользователь получает доступ к системе.

Фрагмент кода авторизации:

*private void LoginButton\_Click(object sender, RoutedEventArgs e)*

*{*

*string login = LoginTextBox.Text.Trim();*

*string password = PasswordBox.Password.Trim();*

*if (string.IsNullOrEmpty(login) || string.IsNullOrEmpty(password))*

*{*

*ErrorTextBlock.Text = "Заполните все поля.";*

*return;*

*}*

*// Хеширование пароля*

*string hashedPassword = HashPassword(password);*

*// Проверка в базе данных*

*var user = Practice8Entities1.GetContext().Users*

*.FirstOrDefault(u => u.login == login && u.password == hashedPassword);*

*if (user != null)*

*{*

*// Устанавливаем текущего пользователя*

*App.CurrentUser = user;*

*// Переход на главную страницу*

*NavigationService.Navigate(new CategoriesPage());*

*// Сделаем кнопки меню на главном окне доступными*

*if (Application.Current.MainWindow is MainWindow mainWindow)*

*{*

*mainWindow.IsUserAuthorized = true;*

*}*

*}*

*else*

*{*

*failedAttempts++;*

*ErrorTextBlock.Text = "Неверный логин или пароль.";*

*if (failedAttempts >= 3)*

*{*

*// Показываем капчу*

*CaptchaTextBlock.Visibility = Visibility.Visible;*

*CaptchaTextBox.Visibility = Visibility.Visible;*

*if (!CheckCaptcha())*

*{*

*ErrorTextBlock.Text = "Неверная капча. Попробуйте снова.";*

*GenerateCaptcha();*

*return;*

*}*

*}*

*}*

*}*

Форма редактирования категории:

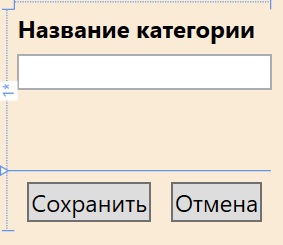


Рисунок 4 - Форма редактирования категории

Назначение формы: Форма редактирования категории позволяет администратору добавлять или изменять информацию о категориях продуктов. Пользователь может ввести название категории, после чего данные сохраняются в базе данных.

Фрагмент кода сохранения категории:

*private void SaveButton\_Click(object sender, RoutedEventArgs e)*

*{*

*if (string.IsNullOrWhiteSpace(CategoryNameTextBox.Text))*

*{*

*MessageBox.Show("Название категории не может быть пустым.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);*

*return;*

*}*

*try*

*{*

*// Обновляем категорию*

*\_category.name = CategoryNameTextBox.Text;*

*var context = Practice8Entities1.GetContext();*

*if (!context.Сategories.Any(c => c.id == \_category.id))*

*{*

*context.Сategories.Add(\_category); // Добавляем, если это новая категория*

*}*

*context.SaveChanges();*

*MessageBox.Show("Категория сохранена.", "Успешно", MessageBoxButton.OK, MessageBoxImage.Information);*

*// Обновляем список категорий*

*\_onCategoryUpdated?.Invoke();*

*NavigationService.GoBack();*

*}*

*catch (Exception ex)*

*{*

*MessageBox.Show($"Ошибка сохранения: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);*

*}*

*}*

Форма вывода продуктов:

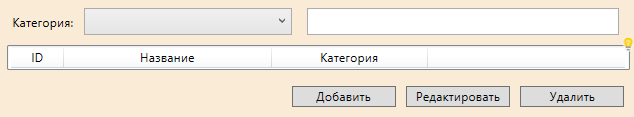


Рисунок 5 - Форма вывода продуктов

Назначение формы: Форма вывода продуктов позволяет администратору добавлять или изменять информацию о продуктах. Пользователь может ввести название продукта и выбрать категорию, к которой относится продукт для поиска.

Фрагмент кода вывода продуктов:

private void LoadProducts()

{

try

{

Products.Clear();

var products = Practice8Entities1.GetContext().Products.ToList();

foreach (var product in products)

{

Products.Add(product);

}

// Применяем сортировку по умолчанию

ApplySorting();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка загрузки продуктов: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

Форма отчета

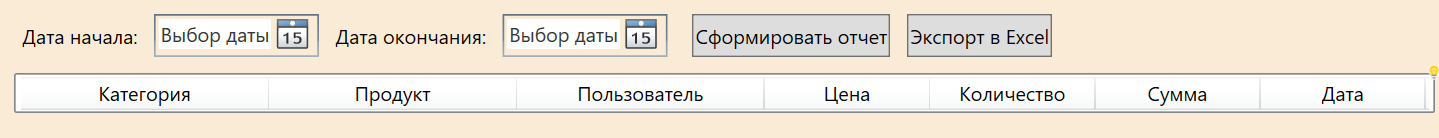


Рисунок 5 - Форма вывода отчетов

Назначение формы: Форма отчета позволяет администратору формировать отчет по заказам за определенный период. Пользователь выбирает даты начала и окончания периода, после чего система формирует отчет и предоставляет возможность экспорта данных в Excel.

Фрагмент кода генерации отчета:

private void GenerateReport\_Click(object sender, RoutedEventArgs e)

{

try

{

var startDate = StartDatePicker.SelectedDate;

var endDate = EndDatePicker.SelectedDate

if (startDate == null || endDate == null)

{

MessageBox.Show("Выберите даты для формирования отчета.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (startDate > endDate)

{

MessageBox.Show("Дата начала не может быть позже даты окончания.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

var context = Practice8Entities1.GetContext();

var reportData = context.Orders

.Where(o => o.date != null && o.date >= startDate && o.date <= endDate)

.Select(o => new

{

CategoryName = o.Products.Сategories.name,

ProductName = o.Products.name,

UserName = o.Users.last\_name + " " + o.Users.first\_name,

Price = o.price,

Count = o.count,

Sum = o.sum,

Date = o.date

})

.OrderBy(o => o.CategoryName)

.ThenBy(o => o.Date)

.ToList();

// Добавляем итоговую строку

var totalSum = reportData.Sum(o => o.Sum);

reportData.Add(new

{

CategoryName = "Итого",

ProductName = string.Empty,

UserName = string.Empty,

Price = (decimal?)null,

Count = (int?)null,

Sum = totalSum,

Date = (DateTime?)null

});

ReportListView.ItemsSource = reportData;

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка формирования отчета: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}}

**Описание логики приложения**

Приложение представляет собой систему управления заказами, продуктами, категориями и пользователями. Основные функции приложения включают авторизацию пользователей, управление данными (добавление, редактирование, удаление), формирование отчетов и экспорт данных. Ниже приведено описание логики приложения по основным модулям.

1. **Авторизация и регистрация пользователей**

#### ****Логика авторизации****

* Пользователь вводит логин и пароль на форме авторизации.
* Система проверяет введенные данные в базе данных.
* Если данные корректны, пользователь получает доступ к системе.
* Если пользователь вводит неверные данные три раза, система требует ввода капчи.

**Фрагмент кода авторизации:**

private void LoginButton\_Click(object sender, RoutedEventArgs e)

{

string login = LoginTextBox.Text.Trim();

string password = PasswordBox.Password.Trim();

if (string.IsNullOrEmpty(login) || string.IsNullOrEmpty(password))

{

ErrorTextBlock.Text = "Заполните все поля.";

return;

}

// Хеширование пароля

string hashedPassword = HashPassword(password);

// Проверка в базе данных

var user = Practice8Entities1.GetContext().Users

.FirstOrDefault(u => u.login == login && u.password == hashedPassword);

if (user != null)

{

// Устанавливаем текущего пользователя

App.CurrentUser = user;

// Переход на главную страницу

NavigationService.Navigate(new CategoriesPage());

}

else

{

failedAttempts++;

ErrorTextBlock.Text = "Неверный логин или пароль.";

if (failedAttempts >= 3)

{

// Показываем капчу

CaptchaTextBlock.Visibility = Visibility.Visible;

CaptchaTextBox.Visibility = Visibility.Visible;

if (!CheckCaptcha())

{

ErrorTextBlock.Text = "Неверная капча. Попробуйте снова.";

GenerateCaptcha();

return;

}

}

}

}

\

2. **Управление категориями**

#### ****Логика добавления/редактирования категории****

* Администратор может добавлять или редактировать категории продуктов.
* На форме редактирования категории пользователь вводит название категории.
* После сохранения данные обновляются в базе данных.

**Фрагмент кода сохранения категории:**

private void SaveButton\_Click(object sender, RoutedEventArgs e)

{

if (string.IsNullOrWhiteSpace(CategoryNameTextBox.Text))

{

MessageBox.Show("Название категории не может быть пустым.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

try

{

// Обновляем категорию

\_category.name = CategoryNameTextBox.Text;

var context = Practice8Entities1.GetContext();

if (!context.Сategories.Any(c => c.id == \_category.id))

{

context.Сategories.Add(\_category); // Добавляем, если это новая категория

}

context.SaveChanges();

MessageBox.Show("Категория сохранена.", "Успешно", MessageBoxButton.OK, MessageBoxImage.Information);

// Обновляем список категорий

\_onCategoryUpdated?.Invoke();

NavigationService.GoBack();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка сохранения: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

3. **Управление продуктами**

#### ****Логика добавления/редактирования продукта****

* Администратор может добавлять или редактировать продукты.
* На форме редактирования продукта пользователь вводит название продукта и выбирает категорию.
* После сохранения данные обновляются в базе данных.

**Фрагмент кода сохранения продукта:**

private void SaveButton\_Click(object sender, RoutedEventArgs e)

{

if (string.IsNullOrWhiteSpace(ProductNameTextBox.Text) || CategoryComboBox.SelectedValue == null)

{

MessageBox.Show("Заполните все поля.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

try

{

// Обновляем продукт

\_product.name = ProductNameTextBox.Text;

\_product.category\_id = (int)CategoryComboBox.SelectedValue;

var context = Practice8Entities1.GetContext();

if (!context.Products.Any(p => p.id == \_product.id))

{

context.Products.Add(\_product); // Добавляем, если это новый продукт

}

context.SaveChanges();

MessageBox.Show("Продукт сохранен.", "Успешно", MessageBoxButton.OK, MessageBoxImage.Information);

// Обновляем список продуктов

\_onProductUpdated?.Invoke();

NavigationService.GoBack();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка сохранения: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

4. **Формирование отчетов**

#### ****Логика формирования отчета****

* Администратор может формировать отчеты по заказам за определенный период.
* Пользователь выбирает даты начала и окончания периода.
* Система формирует отчет, включая итоговую сумму заказов.

**Фрагмент кода формирования отчета:**

private void GenerateReport\_Click(object sender, RoutedEventArgs e)

{

try

{

var startDate = StartDatePicker.SelectedDate;

var endDate = EndDatePicker.SelectedDate;

if (startDate == null || endDate == null)

{

MessageBox.Show("Выберите даты для формирования отчета.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (startDate > endDate)

{

MessageBox.Show("Дата начала не может быть позже даты окончания.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

var context = Practice8Entities1.GetContext();

var reportData = context.Orders

.Where(o => o.date != null && o.date >= startDate && o.date <= endDate)

.Select(o => new

{

CategoryName = o.Products.Сategories.name,

ProductName = o.Products.name,

UserName = o.Users.last\_name + " " + o.Users.first\_name,

Price = o.price,

Count = o.count,

Sum = o.sum,

Date = o.date

})

.OrderBy(o => o.CategoryName)

.ThenBy(o => o.Date)

.ToList();

// Добавляем итоговую строку

var totalSum = reportData.Sum(o => o.Sum);

reportData.Add(new

{

CategoryName = "Итого",

ProductName = string.Empty,

UserName = string.Empty,

Price = (decimal?)null,

Count = (int?)null,

Sum = totalSum,

Date = (DateTime?)null

});

ReportListView.ItemsSource = reportData;

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка формирования отчета: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

5. **Экспорт данных в Excel**

#### ****Логика экспорта отчета в Excel****

* Администратор может экспортировать сформированный отчет в Excel.
* Система создает Excel-файл с данными отчета и сохраняет его в выбранном месте.

**Фрагмент кода экспорта в Excel:**

private void ExportToExcel\_Click(object sender, RoutedEventArgs e)

{

try

{

var startDate = StartDatePicker.SelectedDate;

var endDate = EndDatePicker.SelectedDate;

if (startDate == null || endDate == null)

{

MessageBox.Show("Выберите даты для формирования отчета.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (startDate > endDate)

{

MessageBox.Show("Дата начала не может быть позже даты окончания.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

var context = Practice8Entities1.GetContext();

var reportData = context.Orders

.Where(o => o.date != null && o.date >= startDate && o.date <= endDate)

.Select(o => new

{

CategoryName = o.Products.Сategories.name,

ProductName = o.Products.name,

UserName = o.Users.last\_name + " " + o.Users.first\_name,

Price = o.price,

Count = o.count,

Sum = o.sum,

Date = o.date

})

.OrderBy(o => o.CategoryName)

.ThenBy(o => o.Date)

.ToList();

// Добавляем итоговую строку

var totalSum = reportData.Sum(o => o.Sum);

reportData.Add(new

{

CategoryName = "Итого",

ProductName = string.Empty,

UserName = string.Empty,

Price = (decimal?)null,

Count = (int?)null,

Sum = totalSum,

Date = (DateTime?)null

});

// Создаем Excel-файл

using (var workbook = new XLWorkbook())

{

var worksheet = workbook.Worksheets.Add("Отчет");

// Заголовки столбцов

worksheet.Cell(1, 1).Value = "Категория";

worksheet.Cell(1, 2).Value = "Продукт";

worksheet.Cell(1, 3).Value = "Пользователь";

worksheet.Cell(1, 4).Value = "Цена";

worksheet.Cell(1, 5).Value = "Количество";

worksheet.Cell(1, 6).Value = "Сумма";

worksheet.Cell(1, 7).Value = "Дата";

// Заполняем данные

for (int i = 0; i < reportData.Count; i++)

{

var row = reportData[i];

worksheet.Cell(i + 2, 1).Value = row.CategoryName;

worksheet.Cell(i + 2, 2).Value = row.ProductName;

worksheet.Cell(i + 2, 3).Value = row.UserName;

worksheet.Cell(i + 2, 4).Value = row.Price;

worksheet.Cell(i + 2, 5).Value = row.Count;

worksheet.Cell(i + 2, 6).Value = row.Sum;

worksheet.Cell(i + 2, 7).Value = row.Date?.ToString("dd.MM.yyyy");

}

// Сохраняем файл

var saveFileDialog = new SaveFileDialog

{

Filter = "Excel files (\*.xlsx)|\*.xlsx",

FileName = "Отчет.xlsx"

};

if (saveFileDialog.ShowDialog() == true)

{

workbook.SaveAs(saveFileDialog.FileName);

MessageBox.Show("Отчет успешно экспортирован в Excel.", "Успешно", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка экспорта отчета: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

6. **Управление пользователями**

#### ****Логика добавления/редактирования пользователя****

* Администратор может добавлять или редактировать пользователей.
* На форме редактирования пользователя пользователь вводит фамилию, имя, отчество, логин и пароль.
* После сохранения данные обновляются в базе данных.

**Фрагмент кода сохранения пользователя:**

private void SaveButton\_Click(object sender, RoutedEventArgs e)

{

if (string.IsNullOrWhiteSpace(LastNameTextBox.Text) ||

string.IsNullOrWhiteSpace(FirstNameTextBox.Text) ||

string.IsNullOrWhiteSpace(LoginTextBox.Text) ||

string.IsNullOrWhiteSpace(PasswordTextBox.Text))

{

InfoTextBlock.Text = "Заполните все обязательные поля!";

return;

}

\_user.last\_name = LastNameTextBox.Text.Trim();

\_user.first\_name = FirstNameTextBox.Text.Trim();

\_user.patronymic = PatronymicTextBox.Text.Trim();

\_user.login = LoginTextBox.Text.Trim();

\_user.password = PasswordTextBox.Text.Trim();

try

{

var context = Practice8Entities1.GetContext();

if (\_user.id == 0)

{

context.Users.Add(\_user);

}

context.SaveChanges();

MessageBox.Show("Пользователь успешно сохранен!", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

\_onUserUpdated?.Invoke();

NavigationService.GoBack();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при сохранении: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

**Описание отчетов**

В ходе практики была создана возможность формировать отчеты, которые позволяют анализировать данные о заказах, платежах и других операциях в системе. Отчеты формируются на основе данных из базы данных и предоставляют возможность экспорта в Excel. Ниже приведены описания созданных отчетов, скриншоты и фрагменты кода, демонстрирующие их формирование.

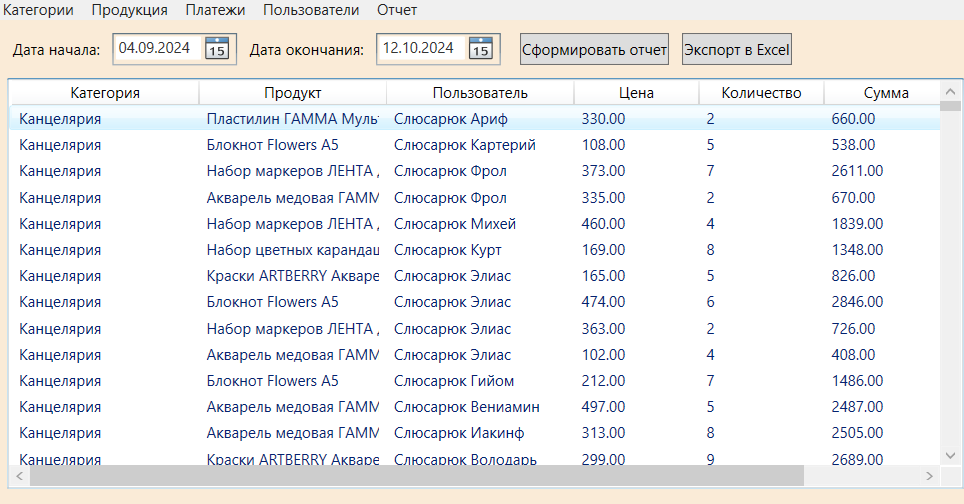
Оба отчета содержат информацию о категориях продуктов, названиях продуктов, пользователях, цене, количестве продуктов, общей сумме и дате платежа, но один формируется внутри приложения, а второй выводится в Excel, код формирования отчетов предоставлен выше. 

Рисунок 5 - Отчет, формируемый в приложении

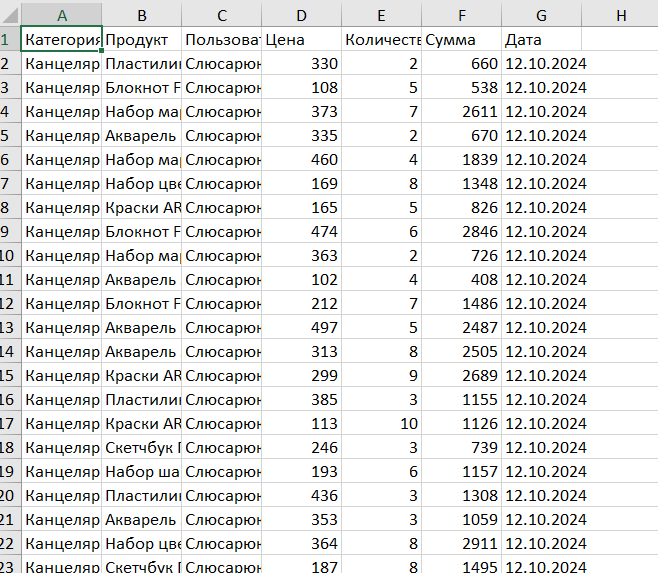


Рисунок 6 - Отчет, экспортируемый в Excel

**Заключение**

В ходе прохождения практики по разработке программного продукта мною были успешно выполнены все поставленные задачи и достигнуты основные цели практики. В процессе работы я приобрел практические навыки в области системного анализа, проектирования, разработки и интеграции программных решений, а также работы с базами данных.

На этапе системного анализа и проектирования была разработана ER-диаграмма базы данных, соответствующая третьей нормальной форме, с учетом ссылочной целостности. Также был спроектирован интерфейс программного модуля и разработана блок-схема алгоритма решения одной из функций приложения. Это позволило сформировать четкую структуру будущего приложения и обеспечить его логическую согласованность.

На этапе разработки программного обеспечения была создана база данных в СУБД SQL Server, настроены связи между таблицами и выполнен импорт данных. Разработан модуль авторизации с использованием хеширования паролей (SHA256), что обеспечило безопасность хранения пользовательских данных. Также были реализованы основные формы приложения, включая меню, формы добавления/редактирования данных, а также функционал поиска и фильтрации данных. Особое внимание было уделено обработке исключительных ситуаций, что повысило удобство и надежность работы приложения.

Интеграция программных решений включала реализацию алгоритма подсчета итоговой суммы платежа и расчета скидки, а также отображение истории платежей в виде отчета. Это позволило обеспечить функциональность приложения, соответствующую поставленным требованиям.

На заключительном этапе была выполнена работа с системой контроля версий, что обеспечило хранение и структурирование исходного кода, а также других результатов практики. Также был оформлен отчет по практике, который отражает все этапы работы и полученные результаты.

В целом, практика позволила мне закрепить теоретические знания и приобрести практические навыки в области разработки программного обеспечения, работы с базами данных и проектирования пользовательских интерфейсов. Полученные знания и опыт будут полезны в дальнейшей профессиональной деятельности.

# Используемые источники информации

1. Скотт Митчелл. Введение в ASP.NET Core. — Питер, 2020. — 448 с.
2. Александр Шевчук. Entity Framework Core 2 для начинающих. — БХВ-Петербург, 2018. — 320 с.
3. Андрей Кобец, Джим Хопкинс. WPF для профессионалов. — Вильямс, 2019. — 896 с.
4. Андрей Карпов. C# 7 и .NET Core. Кросс-платформенная разработка. — Питер, 2018. — 352 с.
5. Дмитрий Немчанинов. SQL Server 2019. Основы. — БХВ-Петербург, 2020. — 320 с.
6. Александр Иванов. Разработка приложений на C# с использованием Visual Studio 2019. — Питер, 2020. — 448 с.
7. Андрей Новиков. Основы программирования на C#. — БХВ-Петербург, 2017. — 320 с.
8. Дмитрий Тепляков. WPF. Практическое руководство. — БХВ-Петербург, 2015. — 416 с.
9. Сергей Тепляков. Entity Framework. Основы. — БХВ-Петербург, 2014. — 320 с.
10. Дмитрий Кирсанов. SQL Server 2016. Основы. — БХВ-Петербург, 2017. — 320 с.