**Министерство науки и высшего образования Российской Федерации**

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**ОТЧЕТ**

**по учебной практике (по профилю специальности)**

по профессиональному модулю ПМ 02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность 09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 22290907/1098 группы

\_\_\_\_ Буранчиков Данила Романович

(ФИО полностью)

Место прохождения практики: СПбПУ ИСПО, Приморский пр. 63

(наименование и адрес организации)

Период прохождения практики

с «02» сентября 2025 г. по «13» сентября 2025 г.

Руководитель практики

от учебной организации Шаврова Л.С.

(подпись) (расшифровка подписи)

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

М.П.

Санкт-Петербург

2025

СОГЛАСОВАНО:

Председатель ПЦК

\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_ 20\_\_ г.

**ЗАДАНИЕ**

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

по профессиональному модулю ПМ 02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность 09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 22290907/1098 группы

Буранчиков Данила Романович

(ФИО полностью)

Место прохождения практики: СПбПУ ИСПО, Приморский пр. 63

(наименование и адрес организации)

Период прохождения практики

с «02» сентября 2025 г. по «13» сентября 2025 г.

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

1. Участие в выработке требований к программному обеспечению.
2. Стадии проектирования программного обеспечения. Проектирование UML-диаграмм и алгоритмов.
3. Конструирование пользовательского интерфейса. Разработка модулей программного обеспечения.
4. Тестирование и отладка программных модулей. Модульное тестирование.
5. Разработка программной документации.

**Индивидуальное задание:** вариант 5

Задание выдал «02» сентября 2025 г. Шаврова Л.С.

(подпись)  (Ф.И.О.)

Задание получил «02» сентября 2025 г. Буранчиков Д.Р.

(подпись)  (Ф.И.О.)

**Министерство науки и высшего образования Российской Федерации**

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**ДНЕВНИК**

**прохождения учебной практики**

**(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 22290907/1098 группы

Буранчиков Данила Романович

(Фамилия, имя, отчество)

Место прохождения практики: СПбПУ ИСПО, Приморский пр. 63

(наименование и адрес организации)

Период прохождения практики

с «02» сентября 2025 г. по «13» сентября 2025 г.

Руководитель с места

прохождения практики Шаврова Л.С.\_\_\_

(подпись) (расшифровка подписи)

Санкт-Петербург

2025

**Содержание дневника**

|  |  |  |
| --- | --- | --- |
| Дата | Виды выполненных работ и заданий по программе  практики | Подпись руководителя практики |
| **1** | **2** | **3** |
| 02.09 | Анализ предметной области. Анализ методов решения. Анализ и выбор средств. Выявление требований к программе. |  |
| 03.09 | Исследовательские работы. Обоснование принципиальной возможности решения задачи. UML. Проектирование диаграммы вариантов использования и последовательности. |  |
| 04.09 | Формирование основного алгоритма системы и детализация в виде алгоритма одной из ее функций. Моделирование структуры ПО. |  |
| 05.09 | Создание базы данных и ERD модели с применением case-средств. Выполнение резервного копирования БД и восстановления. |  |
| 06.09 | Создание базы данных и ERD модели с применением case-средств. Выполнение резервного копирования БД и восстановления. Проектирование интерфейса пользователя. Разработка дизайна программы в соответствии с руководством по стилю. |  |
| 08.09 | Разработка модели данных. Разработка программного продукта. Создание функциональности ПО. |  |
| 09.09 |
| 10.09 |
| 11.09 | Разработка модели данных. Разработка программного продукта. Создание функциональности ПО. Модульное тестирование. Создание тестовых случаев. Отладка программных модулей. |  |
| 12.09 | Модульное тестирование. Создание тестовых случаев. Отладка программных модулей. |  |
| 13.09 | Разработка документа «Руководство системного программиста» в соответствии с ЕСПД. Подготовка отчета. |  |

**Содержание**

[Введение 6](#_Toc208516868)

[1 Проектирование системы 7](#_Toc208516869)

[1.1 Краткая спецификация 7](#_Toc208516870)

[1.2 Диаграмма вариантов использования 8](#_Toc208516871)

[1.3 Диаграмма последовательности 9](#_Toc208516872)

[1.4 Общий алгоритм 10](#_Toc208516873)

[1.5 Алгоритм одной из функций 11](#_Toc208516874)

[2 Создание базы данных и заполнение таблиц данными 12](#_Toc208516875)

[2.1 ER – диаграмма 12](#_Toc208516876)

[2.2 Заполненные данными таблицы 13](#_Toc208516877)

[2.3 Резервное копирование 14](#_Toc208516878)

[3 Разработка приложения 15](#_Toc208516879)

[4 Тестирование приложения 22](#_Toc208516880)

[4.1 Модульное тестирование 22](#_Toc208516881)

[5 Выгрузка готового проекта в репозиторий Git 23](#_Toc208516882)

[Заключение 24](#_Toc208516883)

[Список использованных источников 25](#_Toc208516884)

[Приложение А (справочное) Скрипт БД 26](#_Toc208516885)

[Приложение Б (справочное) Общий код 34](#_Toc208516886)

[Приложение В (справочное) Тестирование 84](#_Toc208516887)

# **Введение**

Полное наименование программной разработки: «Автоматизированная информационная система управления производственно-сбытовой деятельностью предприятия «Новые технологии» (АИС «Новые технологии»).

Система включает в себя следующие подсистемы:

* Управление партнерской сетью
* Обработка заявок на продукцию
* Управление производственными процессами
* Складской учет материалов и готовой продукции
* Управление персоналом и доступом
* Взаимодействие с поставщиками
* Аналитика и отчетность

Назначение: Автоматизация ключевых бизнес-процессов производственной компании, включая работу с дистрибьюторами, управление заказами и производственными планами, складской учет и анализ эффективности.

1. **Проектирование системы**
   1. **Краткая спецификация**

Разработанный программный комплекс для предприятия «Новые технологии» представляет собой комплексное решение, которое включает функциональные задачи, комплекс мер обеспечения и соответствующую документацию:

1. Функциональные задачи, реализуемые в ходе разработки:
   1. Редактирование;
   2. Изменение;
   3. Управление номенклатурой продукции;
   4. Управление данными.
2. Комплекс обеспеченных мер при создании проекта:
   1. Использование лицензированного программного обеспечения;
   2. Проверка программы на наличие вирусов;
   3. Организация бесперебойного питания;
   4. Безопасность.
3. Состав технической документации и отчётности:

Программный комплекс для производственного предприятия «Новые технологии» представляет собой продуманное решение, обеспечивающее комплексный контроль всех веток деятельности компании. Система объединяет управление номенклатурой изделий, учёт материальных ресурсов, мониторинг готовой продукции, координацию производственных операций, организацию сотрудничества с контрагентами и администрирование персонала. Единое информационное пространство позволяет синхронизировать работу различных подразделений и оптимизировать бизнес-процессы.

* 1. **Диаграмма вариантов использования**

На рисунке 1 представлена диаграмма вариантов использования. На ней изображены все роли, используемые в системе.

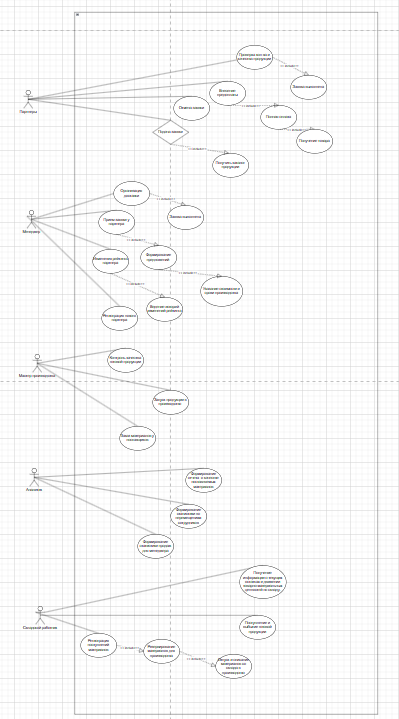


Рисунок 1 – Диаграмма вариантов использования

* 1. **Диаграмма последовательности**

На рисунке 2 изображена диаграмма последовательности. На ней представлен процесс «Создание заказа с полной проверкой».

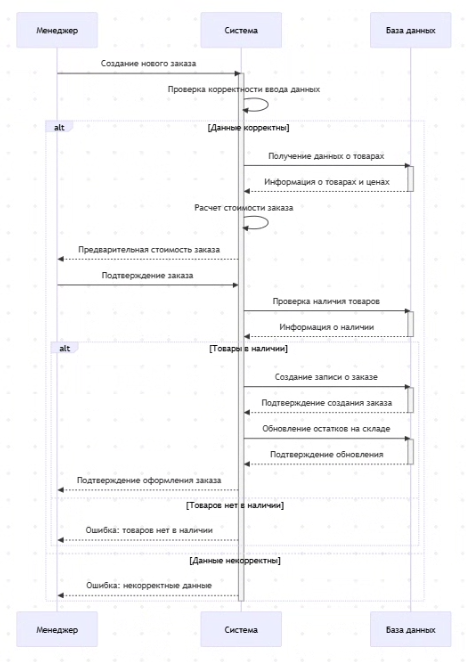


Рисунок 2 – Диаграмма последовательности

* 1. **Общий алгоритм**

На рисунке 3 представлен общий алгоритм приложения.



Рисунок 3 – Общий алгоритм

* 1. **Алгоритм одной из функций**

На рисунке 4 представлен алгоритм процесса «Оценка качества материалов».



Рисунок 4 – Алгоритм «Контроль скидок»

1. **Создание базы данных и заполнение таблиц данными**
   1. **ER – диаграмма**

На рисунке 5 представлена ER-диаграмма базы данных.

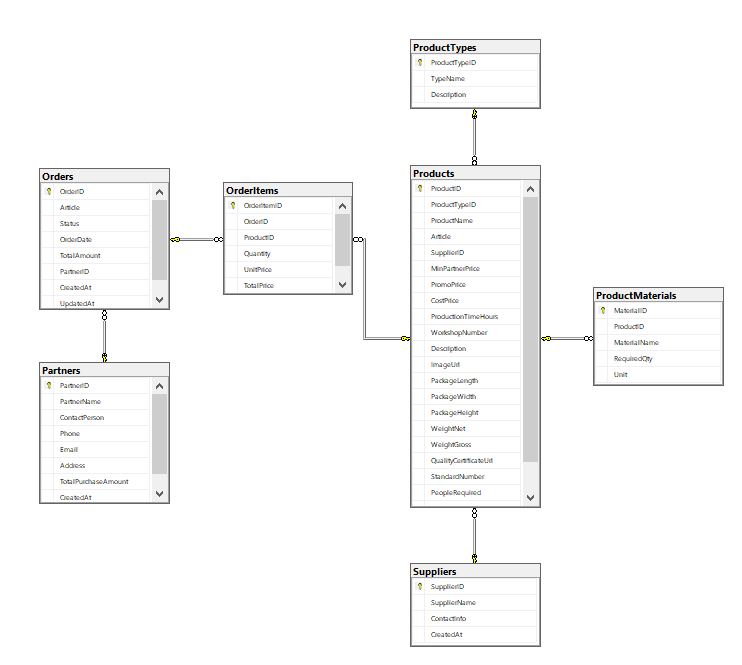


Рисунок 5 – ER – диаграмма

* 1. **Заполненные данными таблицы**

На рисунке 6 представлены таблицы с заполненными данными.

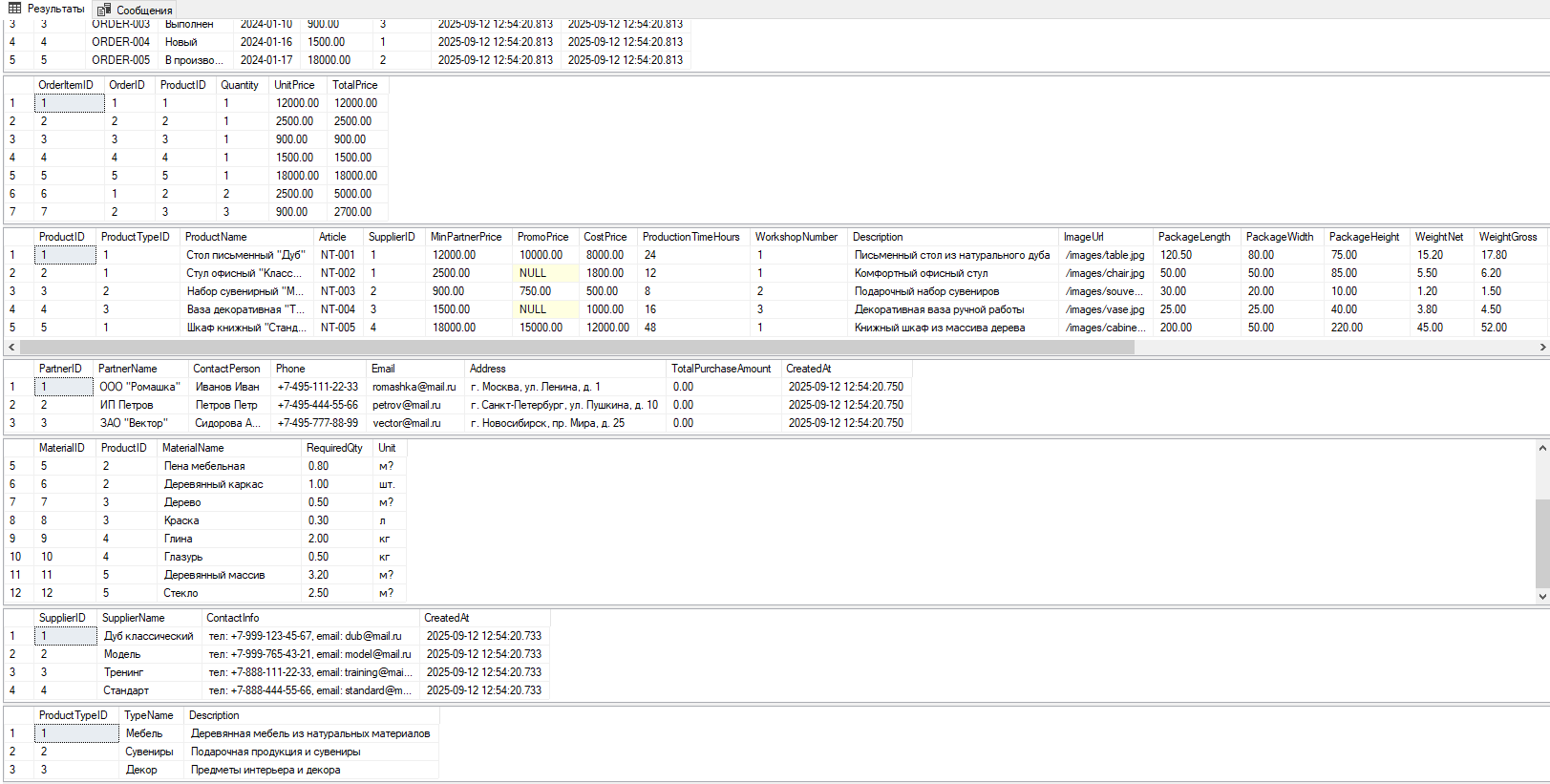


Рисунок 6 – Заполненные таблицы

* 1. **Резервное копирование**

На рисунке 7 представлено успешное резервное копирование базы данных.

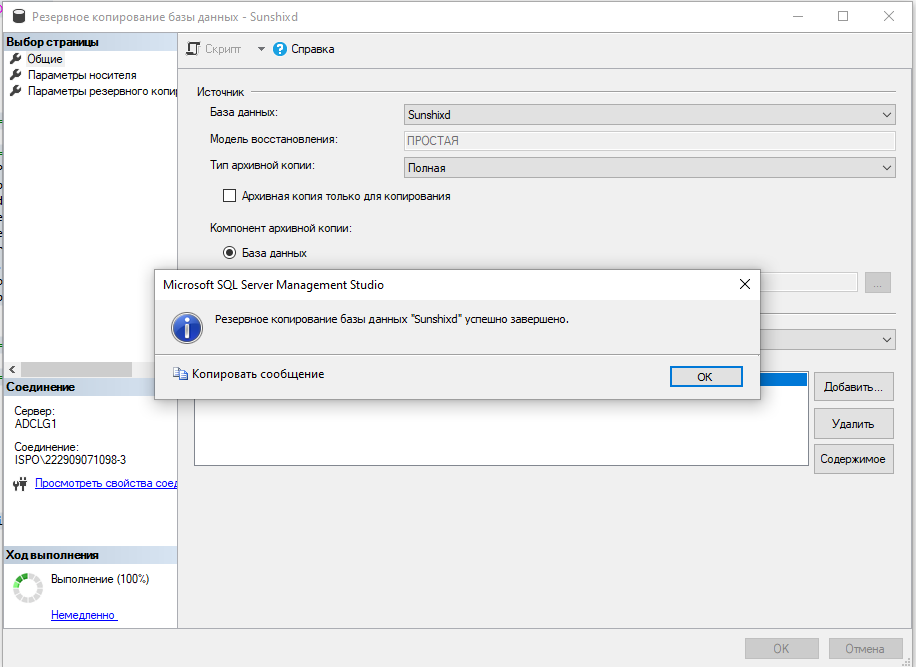


Рисунок 7 – Резервное копирование

1. **Разработка приложения**

На рисунке 8 представлена форма входа в систему. На ней присутствуют такие функции как: «Вход», «Вход как гость». Также реализована функция капчи.

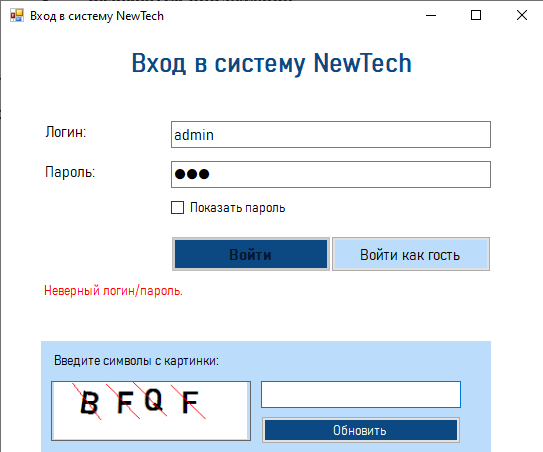


Рисунок 8 – Форма авторизации

На рисунке 9 представлена главная форма. На ней реализован следующий функционал: кнопка «Добавить», кнопка «Заказы», кнопка «История входа» и кнопка «Выход», завершающая сеанс и перенаправляющая пользователя на форму логина.

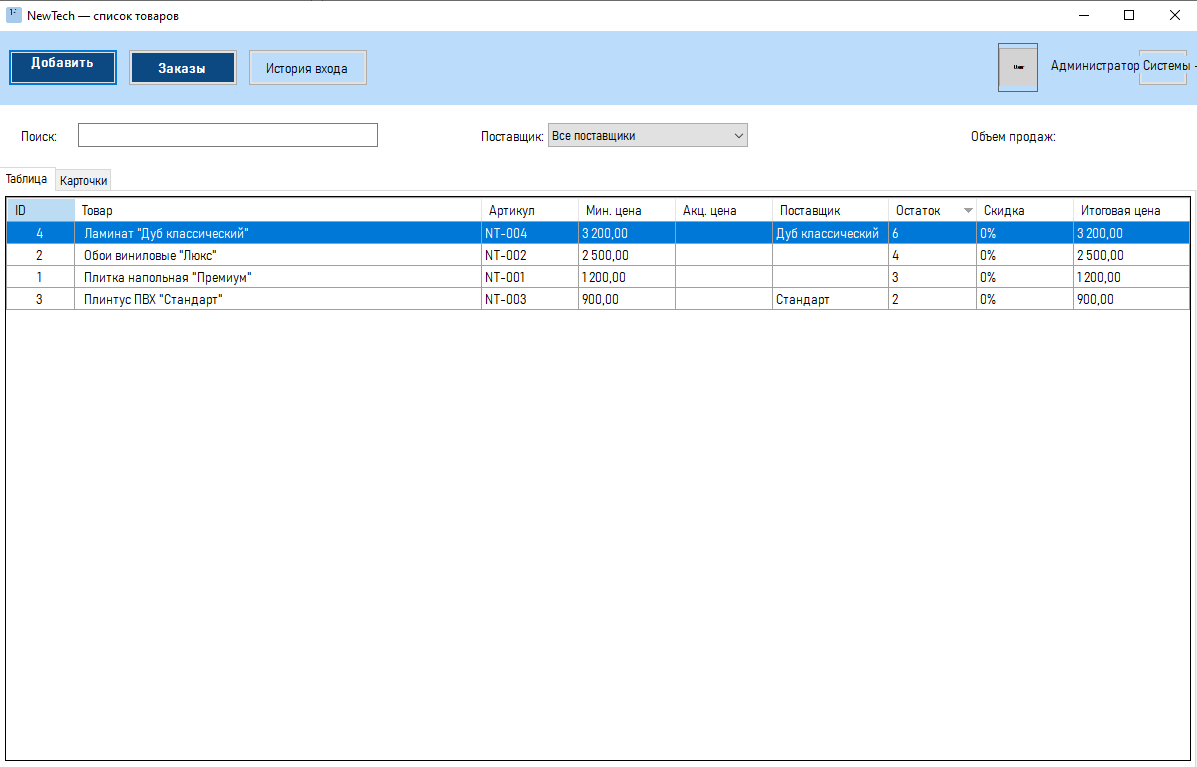
****

Рисунок 9 – Главная форма

На рисунке 10 представлена панель добавления товаров. Реализован функционал: выбор поставщика из выпадающего списка, ввод названия, артикула, выбор минимальной и акционной цены, а также загрузка изображения товара

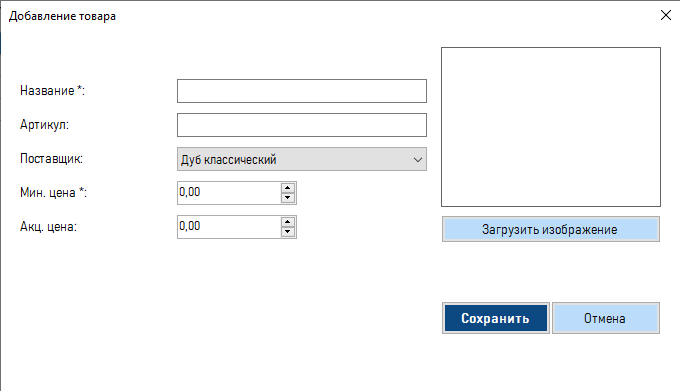
****

Рисунок 10 – Панель «Товары»

На рисунке 11 - 13 представлена форма управления и редактирования заказов. Здесь реализована возможность добавлять заказ, осуществлять поиск по заказам, сортировать список заказов по партнёру. Доступ к этой форме имеют только менеджеры (только просмотр) и администраторы (полный доступ).

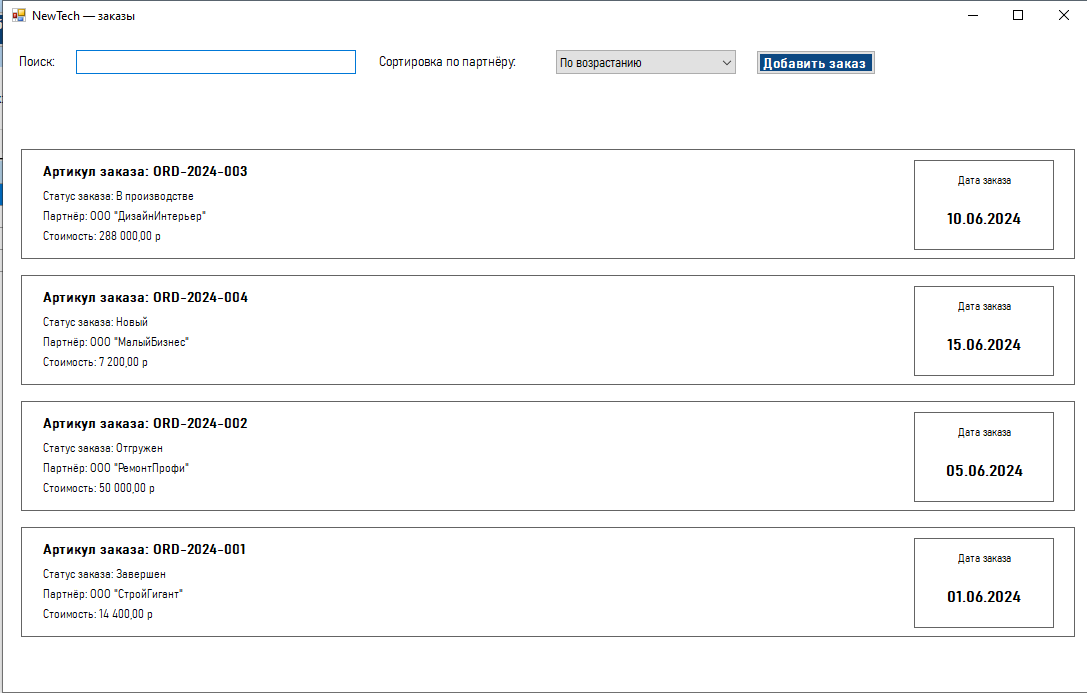
****

Рисунок 11 – Управление заказами

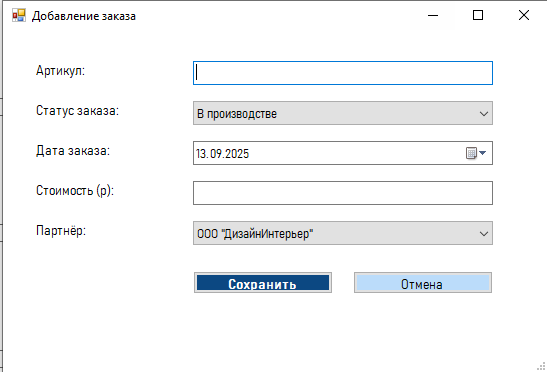
****

Рисунок 12 – Добавление заказа

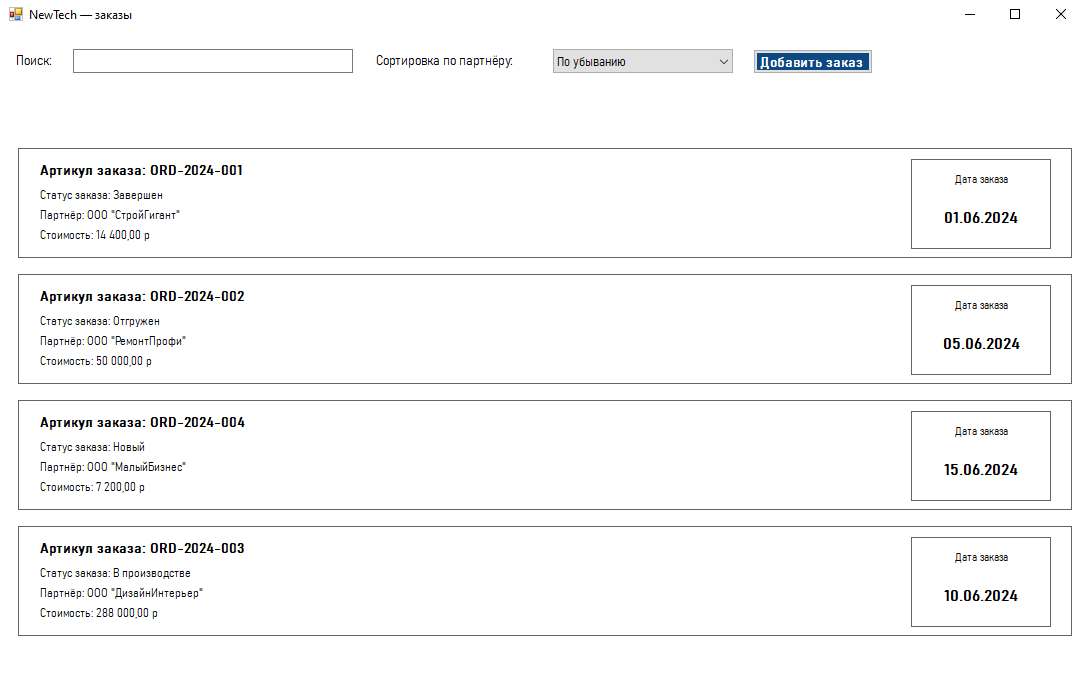
****

Рисунок 13 – Сортировка по партнёру

1. **Тестирование приложения**
   1. **Модульное тестирование**

Тестовые сценарии №1–5 представляют собой модульные тесты (Unit Tests), которые проверяют работу отдельных компонентов системы. Например, тест TestCaptchaGeneration\_LengthAndContent() проверяет корректность генерации CAPTCHA указанной длины, TestDiscountCalculation\_VolumeBased() тестирует расчет скидки в зависимости от объема покупки, а TestFinalPriceCalculation\_WithDiscount проверяет итоговый расчет цены с учетом различных факторов. Остальные модульные тесты представлены в приложении В.

Код на C# относится к модульным тестам, которые проверяют работу отдельных методов или классов системы. Например, тест TestPasswordHashing\_Sha256 проверяет корректность хеширования пароля по алгоритму SHA-256, TestProductValidation\_InvalidPrice() тестирует валидацию некорректных данных продукта, а TestSystemLockout\_Escalation проверяет механизм блокировки системы при эскалации.

Скриншот успешного тестирования представлен на рисунке 17.

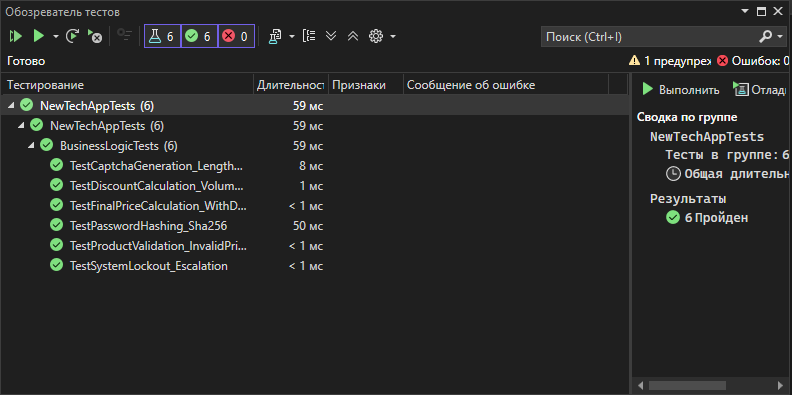


Рисунок 14 – Результат успешного тестирования

1. **Выгрузка готового проекта в репозиторий Git**

На рисунках 17–18 представлена выгрузка проекта в репозиторий Git Hub.

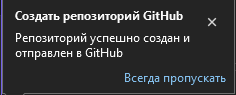
****

Рисунок 15 – Выгрузка проекта на Git

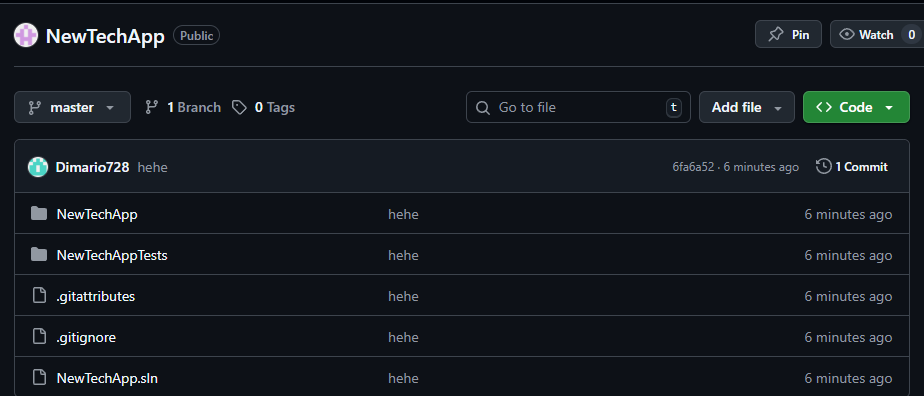
****

Рисунок 16 – Результат выгрузки

# **Заключение**

В рамках учебной практики была создана информационная система управления товарами для компании «Новые технологии», полностью соответствующая техническому заданию. Приложение автоматизирует процессы учёта продукции, материалов, заказов и пользователей, обеспечивая полный цикл обработки данных: отображение, фильтрацию, сортировку и расчёт стоимости.

Ключевые возможности системы: просмотр товаров с изображениями, актуальными ценами и скидками, а также управление заказами. Реализованы функции поиска по наименованию, фильтрации и сортировки по различным критериям.

Для интеграции с базой данных использовался Microsoft SQL Server. Применение параметризованных запросов обеспечивает защиту от SQL-инъекций и сохранность данных, а использование внешних ключей гарантирует целостность информации.

Проведённое тестирование включает модульные тесты, подтвердившие корректность расчёта цен и скидок, валидацию ввода и обработки данных. Все компоненты системы успешно прошли проверку.

Итог: разработанное решение повышает операционную эффективность компании за счёт автоматизации рутинных задач, снижения количества ошибок и ускорения работы с товарами. Проект демонстрирует стабильность, надёжность и соответствие современным стандартам программного обеспечения.

# **Список использованных источников**

1. А. Хейлсберг, М. Торгерсен, С. Вилтамут, П. Голд ., Язык программирования C#, 4-е издание, СПб:-ПИТЕР, 2020.
2. Фленов М., Библия C#, 3-е издание, СПб:-БХВ, 2019.
3. Культин Н., Microsoft Visual C# в задачах и примерах, 2-е издание, СПб: -БХВ, 2015.
4. Адам, Фримен ASP.NET 4.5 с примерами на C# 5.0 для профессионалов / Фримен Адам. - М.: Диалектика / Вильямс, 2021. - 2792 c.
5. Описание языка C# [Электронный ресурс] — URL: https://techrocks.ru/2019/02/16/c-sharp-programming-language-overview/ (дата обращения: 10.09.2025).
6. Климов, А. C#. Советы программистам / А. Климов. - М.: БХВ-Петербург, 2018. - 544 c.
7. Секунов, Н. Самоучитель C# / Н. Секунов. - М.: БХВ-Петербург, 2017. - 576 c.
8. ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению.
9. ГОСТ 19.301-78 Программа и методика испытаний. Требования к содержанию и оформлению.
10. ГОСТ 19.401-78 Текст программы. Требования к содержанию и оформлению.

# Приложение А

(справочное)

**Скрипт БД**

CREATE TABLE dbo.ProductTypes (

ProductTypeID INT PRIMARY KEY IDENTITY(1,1),

TypeName NVARCHAR(50) NOT NULL UNIQUE,

Description NVARCHAR(255) NULL

);

GO

-- Таблица поставщиков

CREATE TABLE dbo.Suppliers (

SupplierID INT PRIMARY KEY IDENTITY(1,1),

SupplierName NVARCHAR(100) NOT NULL UNIQUE,

ContactInfo NVARCHAR(255) NULL,

CreatedAt DATETIME DEFAULT GETDATE()

);

GO

-- Таблица партнеров

CREATE TABLE dbo.Partners (

PartnerID INT PRIMARY KEY IDENTITY(1,1),

PartnerName NVARCHAR(100) NOT NULL UNIQUE,

ContactPerson NVARCHAR(100) NULL,

Phone NVARCHAR(20) NULL,

Email NVARCHAR(100) NULL,

Address NVARCHAR(255) NULL,

TotalPurchaseAmount DECIMAL(18,2) DEFAULT 0,

CreatedAt DATETIME DEFAULT GETDATE()

);

GO

-- Затем создаем таблицы, которые ссылаются на уже созданные

-- Таблица товаров

CREATE TABLE dbo.Products (

ProductID INT PRIMARY KEY IDENTITY(1,1),

ProductTypeID INT NULL FOREIGN KEY REFERENCES dbo.ProductTypes(ProductTypeID),

ProductName NVARCHAR(100) NOT NULL,

Article NVARCHAR(50) NOT NULL UNIQUE,

SupplierID INT NULL FOREIGN KEY REFERENCES dbo.Suppliers(SupplierID),

MinPartnerPrice DECIMAL(18,2) NOT NULL CHECK (MinPartnerPrice >= 0),

PromoPrice DECIMAL(18,2) NULL CHECK (PromoPrice >= 0),

CostPrice DECIMAL(18,2) NULL CHECK (CostPrice >= 0),

ProductionTimeHours INT NOT NULL DEFAULT 0 CHECK (ProductionTimeHours >= 0),

WorkshopNumber INT NOT NULL DEFAULT 1 CHECK (WorkshopNumber >= 1),

Description NVARCHAR(MAX) NULL,

ImageUrl NVARCHAR(255) NULL,

PackageLength DECIMAL(10,2) NULL,

PackageWidth DECIMAL(10,2) NULL,

PackageHeight DECIMAL(10,2) NULL,

WeightNet DECIMAL(10,2) NULL,

WeightGross DECIMAL(10,2) NULL,

QualityCertificateUrl NVARCHAR(255) NULL,

StandardNumber NVARCHAR(50) NULL,

PeopleRequired INT NULL DEFAULT 1 CHECK (PeopleRequired >= 1),

CreatedAt DATETIME DEFAULT GETDATE(),

UpdatedAt DATETIME DEFAULT GETDATE()

);

GO

-- Таблица пользователей

CREATE TABLE dbo.Users (

UserID INT PRIMARY KEY IDENTITY(1,1),

Login NVARCHAR(50) NOT NULL UNIQUE,

PasswordHash VARBINARY(256) NOT NULL,

FullName NVARCHAR(100) NOT NULL,

Role NVARCHAR(20) NOT NULL CHECK (Role IN ('Admin', 'Manager', 'Client', 'Guest')),

PhotoUrl NVARCHAR(255) NULL,

CreatedAt DATETIME DEFAULT GETDATE()

);

GO

-- Таблица статусов заказов

CREATE TABLE dbo.OrderStatuses (

StatusID INT PRIMARY KEY IDENTITY(1,1),

StatusName NVARCHAR(50) NOT NULL UNIQUE,

Description NVARCHAR(255) NULL

);

GO

-- Таблица заказов

CREATE TABLE dbo.Orders (

OrderID INT PRIMARY KEY IDENTITY(1,1),

Article NVARCHAR(50) NOT NULL,

Status NVARCHAR(50) NOT NULL DEFAULT 'Новый',

OrderDate DATE NOT NULL DEFAULT GETDATE(),

TotalAmount DECIMAL(18,2) NOT NULL CHECK (TotalAmount >= 0),

PartnerID INT NOT NULL FOREIGN KEY REFERENCES dbo.Partners(PartnerID),

CreatedAt DATETIME DEFAULT GETDATE(),

UpdatedAt DATETIME DEFAULT GETDATE()

);

GO

-- Таблица элементов заказа

CREATE TABLE dbo.OrderItems (

OrderItemID INT PRIMARY KEY IDENTITY(1,1),

OrderID INT NOT NULL FOREIGN KEY REFERENCES dbo.Orders(OrderID),

ProductID INT NOT NULL FOREIGN KEY REFERENCES dbo.Products(ProductID),

Quantity INT NOT NULL CHECK (Quantity > 0),

UnitPrice DECIMAL(18,2) NOT NULL CHECK (UnitPrice >= 0),

TotalPrice DECIMAL(18,2) NOT NULL CHECK (TotalPrice >= 0)

);

GO

-- Таблица материалов для товаров

CREATE TABLE dbo.ProductMaterials (

MaterialID INT PRIMARY KEY IDENTITY(1,1),

ProductID INT NOT NULL FOREIGN KEY REFERENCES dbo.Products(ProductID),

MaterialName NVARCHAR(100) NOT NULL,

RequiredQty DECIMAL(10,2) NOT NULL CHECK (RequiredQty >= 0),

Unit NVARCHAR(20) NOT NULL DEFAULT 'шт.'

);

GO

-- Таблица истории входов (создаем последней, так как она не имеет внешних ключей на другие таблицы)

CREATE TABLE dbo.LoginHistory (

EntryID INT PRIMARY KEY IDENTITY(1,1),

AttemptAt DATETIME NOT NULL DEFAULT GETDATE(),

Login NVARCHAR(50) NULL,

Success BIT NOT NULL,

Reason NVARCHAR(100) NULL

);

GO

-- =============================================

-- ИНДЕКСЫ ДЛЯ ПРОИЗВОДИТЕЛЬНОСТИ

-- =============================================

CREATE INDEX IX\_Products\_SupplierID ON dbo.Products(SupplierID);

CREATE INDEX IX\_Products\_ProductTypeID ON dbo.Products(ProductTypeID);

CREATE INDEX IX\_Products\_Article ON dbo.Products(Article);

CREATE INDEX IX\_Orders\_PartnerID ON dbo.Orders(PartnerID);

CREATE INDEX IX\_Orders\_OrderDate ON dbo.Orders(OrderDate);

CREATE INDEX IX\_OrderItems\_OrderID ON dbo.OrderItems(OrderID);

CREATE INDEX IX\_OrderItems\_ProductID ON dbo.OrderItems(ProductID);

CREATE INDEX IX\_LoginHistory\_Login ON dbo.LoginHistory(Login);

CREATE INDEX IX\_LoginHistory\_AttemptAt ON dbo.LoginHistory(AttemptAt);

GO

-- =============================================

-- ТРИГГЕРЫ

-- =============================================

CREATE TRIGGER trg\_Products\_UpdateDate

ON dbo.Products

AFTER UPDATE

AS

BEGIN

UPDATE dbo.Products

SET UpdatedAt = GETDATE()

FROM dbo.Products p

INNER JOIN inserted i ON p.ProductID = i.ProductID

END;

GO

CREATE TRIGGER trg\_Orders\_UpdateDate

ON dbo.Orders

AFTER UPDATE

AS

BEGIN

UPDATE dbo.Orders

SET UpdatedAt = GETDATE()

FROM dbo.Orders o

INNER JOIN inserted i ON o.OrderID = i.OrderID

END;

GO

# Приложение Б

(справочное)

**Исходный код**

**LoginForm.cs**

using System;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Windows.Forms;

using NewTechApp.Data;

using NewTechApp.UI;

namespace NewTechApp.Auth

{

public partial class LoginForm : Form

{

// UI

TextBox txtLogin, txtPassword, txtCaptcha;

CheckBox chkShow;

Button btnLogin, btnRefreshCaptcha, btnGuest;

Label lblTitle, lblError, lblLockTimer, lblCaptcha;

Panel pnlCaptcha;

PictureBox picCaptcha;

// state

string \_captchaText = "";

int \_failTotal = 0;

int \_failWithCaptcha = 0;

DateTime? \_lockedUntil = null;

Timer \_lockTimer;

bool \_blocked = false;

bool \_captchaOk = false;

public LoginForm()

{

InitializeComponent();

ApplyStyle();

}

private void ApplyStyle()

{

Text = "Вход в систему NewTech";

Font = new Font("Bahnschrift Light SemiCondensed", 12f);

BackColor = Color.White;

StartPosition = FormStartPosition.CenterScreen;

Width = 560; Height = 560;

lblTitle = new Label

{

Text = "Вход в систему NewTech",

Dock = DockStyle.Top,

Height = 60,

TextAlign = ContentAlignment.MiddleCenter,

Font = new Font("Bahnschrift Light SemiCondensed", 20, FontStyle.Bold),

ForeColor = ColorTranslator.FromHtml("#0C4882")

};

Controls.Add(lblTitle);

var lblL = new Label { Text = "Логин:", Left = 40, Top = 90, Width = 120, Font = new Font("Bahnschrift Light SemiCondensed", 12f) };

txtLogin = new TextBox { Left = 170, Top = 90, Width = 320, Font = new Font("Bahnschrift Light SemiCondensed", 12f) };

var lblP = new Label { Text = "Пароль:", Left = 40, Top = 130, Width = 120, Font = new Font("Bahnschrift Light SemiCondensed", 12f) };

txtPassword = new TextBox { Left = 170, Top = 130, Width = 320, UseSystemPasswordChar = true, Font = new Font("Bahnschrift Light SemiCondensed", 12f) };

chkShow = new CheckBox { Left = 170, Top = 165, Width = 160, Text = "Показать пароль", Font = new Font("Bahnschrift Light SemiCondensed", 10f) };

chkShow.CheckedChanged += (s, e) => txtPassword.UseSystemPasswordChar = !chkShow.Checked;

btnLogin = new Button

{

Text = "Войти",

Left = 170,

Top = 205,

Width = 160,

Height = 36,

BackColor = ColorTranslator.FromHtml("#0C4882"),

ForeColor = Color.White,

Font = new Font("Bahnschrift Light SemiCondensed", 12f, FontStyle.Bold)

};

btnLogin.Click += BtnLogin\_Click;

btnGuest = new Button

{

Text = "Войти как гость",

Left = 330,

Top = 205,

Width = 160,

Height = 36,

BackColor = ColorTranslator.FromHtml("#BBDCFA"),

ForeColor = Color.Black,

Font = new Font("Bahnschrift Light SemiCondensed", 12f)

};

btnGuest.Click += (s, e) =>

{

var main = new MainForm("guest", "Гость", "Guest", null);

main.Show();

Hide();

};

lblError = new Label { Left = 40, Top = 250, Width = 450, ForeColor = Color.Red, Font = new Font("Bahnschrift Light SemiCondensed", 10f) };

lblLockTimer = new Label { Left = 40, Top = 275, Width = 450, ForeColor = Color.DarkRed, Font = new Font("Bahnschrift Light SemiCondensed", 10f) };

pnlCaptcha = new Panel

{

Left = 40,

Top = 310,

Width = 450,

Height = 140,

BackColor = ColorTranslator.FromHtml("#BBDCFA"),

Visible = false,

Anchor = AnchorStyles.Top | AnchorStyles.Left

};

lblCaptcha = new Label

{

Text = "Введите символы с картинки:",

Left = 10,

Top = 10,

Width = 260,

AutoSize = false,

Font = new Font("Bahnschrift Light SemiCondensed", 10f)

};

picCaptcha = new PictureBox

{

Left = 10,

Top = 40,

Width = 200,

Height = 60,

BorderStyle = BorderStyle.FixedSingle,

SizeMode = PictureBoxSizeMode.Zoom,

Anchor = AnchorStyles.Top | AnchorStyles.Left

};

txtCaptcha = new TextBox

{

Left = 220,

Top = 40,

Width = 200,

Anchor = AnchorStyles.Top | AnchorStyles.Left,

Font = new Font("Bahnschrift Light SemiCondensed", 12f)

};

txtCaptcha.TextChanged += TxtCaptcha\_TextChanged;

btnRefreshCaptcha = new Button

{

Text = "Обновить",

Left = 220,

Top = 40 + 28 + 8,

Width = 200,

Height = 28,

Anchor = AnchorStyles.Top | AnchorStyles.Left,

BackColor = ColorTranslator.FromHtml("#0C4882"),

ForeColor = Color.White,

Font = new Font("Bahnschrift Light SemiCondensed", 10f)

};

btnRefreshCaptcha.Click += (s, e) =>

{

ShowCaptcha();

txtCaptcha.Clear();

txtCaptcha.Focus();

};

pnlCaptcha.Controls.Add(lblCaptcha);

pnlCaptcha.Controls.Add(picCaptcha);

pnlCaptcha.Controls.Add(txtCaptcha);

pnlCaptcha.Controls.Add(btnRefreshCaptcha);

btnRefreshCaptcha.BringToFront();

pnlCaptcha.Height = btnRefreshCaptcha.Bottom + 12;

Controls.AddRange(new Control[]

{

lblL, txtLogin, lblP, txtPassword, chkShow, btnLogin, btnGuest, lblError, lblLockTimer, pnlCaptcha

});

AcceptButton = btnLogin;

FormClosing += (s, e) => Application.Exit();

Shown += (s, e) =>

{

btnRefreshCaptcha.Top = txtCaptcha.Bottom + 8;

pnlCaptcha.Height = btnRefreshCaptcha.Bottom + 12;

btnRefreshCaptcha.BringToFront();

};

}

static byte[] Sha256(string s)

{

using (var sha = SHA256.Create())

return sha.ComputeHash(Encoding.UTF8.GetBytes(s));

}

void LogAttempt(string login, bool ok, string reason)

{

Db.Exec("INSERT INTO dbo.LoginHistory(Login,Success,Reason) VALUES(@l,@s,@r)",

new SqlParameter("@l", (object)login ?? DBNull.Value),

new SqlParameter("@s", ok),

new SqlParameter("@r", (object)reason ?? DBNull.Value));

}

void UpdateLoginButtonEnabled()

{

if (IsLockedNow())

{

btnLogin.Enabled = false;

return;

}

if (pnlCaptcha.Visible)

{

btnLogin.Enabled = \_captchaOk;

}

else

{

btnLogin.Enabled = true;

}

}

void TxtCaptcha\_TextChanged(object sender, EventArgs e)

{

var current = (txtCaptcha.Text ?? "").Trim();

\_captchaOk = string.Equals(current, \_captchaText, StringComparison.OrdinalIgnoreCase);

txtCaptcha.BackColor = \_captchaOk ? Color.Honeydew : Color.White;

if (\_captchaOk && (lblError.Text == "Капча неверна." || lblError.Text == "Введите капчу."))

lblError.Text = "";

UpdateLoginButtonEnabled();

}

void ShowCaptcha()

{

using (var bmp = new Bitmap(200, 60))

using (var g = Graphics.FromImage(bmp))

{

g.Clear(Color.White);

var chars = "ABCDEFGHJKMNPQRSTUVWXYZ23456789";

var rand = new Random();

var text = new string(Enumerable.Repeat(chars, 4).Select(s => s[rand.Next(s.Length)]).ToArray());

\_captchaText = text;

using (var font = new Font("Bahnschrift Light SemiCondensed", 24, FontStyle.Bold))

{

for (int i = 0; i < text.Length; i++)

{

var x = 15 + i \* 40 + rand.Next(-10, 10);

var y = 10 + rand.Next(-5, 5);

g.TranslateTransform(x, y);

g.RotateTransform(rand.Next(-15, 15));

g.DrawString(text[i].ToString(), font, Brushes.Black, 0, 0);

g.DrawLine(Pens.Red, -5, -5, 30, 30);

g.ResetTransform();

}

}

if (picCaptcha.Image != null) picCaptcha.Image.Dispose();

picCaptcha.Image = (Bitmap)bmp.Clone();

\_captchaOk = false;

txtCaptcha.BackColor = Color.White;

pnlCaptcha.Visible = true;

UpdateLoginButtonEnabled();

}

}

void StartLock(int minutes)

{

\_lockedUntil = DateTime.Now.AddMinutes(minutes);

if (\_lockTimer != null) \_lockTimer.Stop();

\_lockTimer = new Timer { Interval = 1000 };

\_lockTimer.Tick += (s, e) =>

{

var left = \_lockedUntil.Value - DateTime.Now;

if (left <= TimeSpan.Zero)

{

\_lockTimer.Stop();

lblLockTimer.Text = "";

\_failWithCaptcha = 0;

\_blocked = true;

lblError.Text = "Вход заблокирован до перезапуска.";

btnLogin.Enabled = false;

}

else

{

lblLockTimer.Text = string.Format("Блокировка: {0:mm\\:ss}", left);

btnLogin.Enabled = false;

}

};

\_lockTimer.Start();

btnLogin.Enabled = false;

}

bool IsLockedNow() => \_lockedUntil.HasValue && DateTime.Now < \_lockedUntil.Value;

void BtnLogin\_Click(object sender, EventArgs e)

{

lblError.Text = "";

if (\_blocked)

{

lblError.Text = "Вход заблокирован до перезапуска.";

return;

}

if (IsLockedNow()) return;

if (pnlCaptcha.Visible && !\_captchaOk)

{

lblError.Text = "Введите капчу.";

txtCaptcha.Focus();

return;

}

var login = txtLogin.Text.Trim();

var pass = txtPassword.Text;

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

{

lblError.Text = "Введите логин и пароль.";

return;

}

if (pnlCaptcha.Visible)

{

if (!string.Equals(txtCaptcha.Text.Trim(), \_captchaText, StringComparison.OrdinalIgnoreCase))

{

\_failTotal++; \_failWithCaptcha++;

lblError.Text = "Капча неверна.";

LogAttempt(login, false, "CaptchaFailed");

\_captchaOk = false;

UpdateLoginButtonEnabled();

if (\_failWithCaptcha >= 2) { StartLock(3); return; }

ShowCaptcha(); txtCaptcha.Clear(); return;

}

}

var dt = Db.Table("SELECT TOP 1 \* FROM dbo.Users WHERE Login=@l", new SqlParameter("@l", login));

if (dt.Rows.Count == 0)

{

\_failTotal++; lblError.Text = "Неверный логин/пароль.";

LogAttempt(login, false, "NoUser");

if (!pnlCaptcha.Visible) { ShowCaptcha(); txtCaptcha.Focus(); }

else

{

\_failWithCaptcha++;

if (\_failWithCaptcha >= 2) { StartLock(3); return; }

}

return;

}

var row = dt.Rows[0];

var hash = (byte[])row["PasswordHash"];

var ok = hash.SequenceEqual(Sha256(pass));

if (!ok)

{

\_failTotal++; lblError.Text = "Неверный логин/пароль.";

LogAttempt(login, false, "BadPassword");

if (!pnlCaptcha.Visible) { ShowCaptcha(); txtCaptcha.Focus(); }

else

{

\_failWithCaptcha++;

if (\_failWithCaptcha >= 2) { StartLock(3); return; }

if (!IsLockedNow() && \_failTotal > 3) { \_blocked = true; lblError.Text = "Вход заблокирован до перезапуска."; }

}

return;

}

LogAttempt(login, true, null);

var fullName = Convert.ToString(row["FullName"]);

var role = Convert.ToString(row["Role"]);

var photoUrl = row["PhotoUrl"] as string;

var main = new MainForm(login, fullName, role, photoUrl);

main.Show();

Hide();

}

}

}

**Captcha.cs:**

using System;

using System.Drawing;

using System.Drawing.Drawing2D;

namespace NewTechApp.Auth

{

public static class Captcha

{

private static readonly Random R = new Random();

public static Tuple<Bitmap, string> Generate(int width, int height, int len)

{

string text = RandomText(len);

Bitmap bmp = new Bitmap(width, height);

using (var g = Graphics.FromImage(bmp))

{

g.SmoothingMode = SmoothingMode.AntiAlias;

g.Clear(Color.White);

// шум: линии

for (int i = 0; i < 6; i++)

{

using (var p = new Pen(Color.FromArgb(R.Next(50, 180), R.Next(50, 180), R.Next(50, 180)), R.Next(1, 3)))

{

g.DrawLine(p, R.Next(width), R.Next(height), R.Next(width), R.Next(height));

}

}

// текст

using (var font = new Font("Bahnschrift Light SemiCondensed", 28, FontStyle.Bold))

{

int x = 10;

foreach (char ch in text)

{

int y = R.Next(5, height - 35);

using (var b = new SolidBrush(Color.FromArgb(R.Next(0, 120), R.Next(0, 120), R.Next(0, 120))))

{

g.DrawString(ch.ToString(), font, b, x, y);

}

x += 35 + R.Next(-3, 3);

}

}

// шум: точки

for (int i = 0; i < 200; i++)

bmp.SetPixel(R.Next(width), R.Next(height), Color.FromArgb(R.Next(256), R.Next(256), R.Next(256)));

}

return Tuple.Create(bmp, text);

}

private static string RandomText(int len)

{

const string chars = "ABCDEFGHJKLMNPQRSTUVWXYZ23456789";

var s = new char[len];

for (int i = 0; i < len; i++) s[i] = chars[R.Next(chars.Length)];

return new string(s);

}

}

}

**Db.cs:**

using System;

using System.Data;

using System.Data.SqlClient;

using System.IO;

namespace NewTechApp.Data

{

public static class Db

{

public static string AppPath => AppDomain.CurrentDomain.BaseDirectory;

private const string ConnString =

"Server=DESKTOP-J7SEHSP\\SQLEXPRESS;Database=NewTechDB;Trusted\_Connection=True;TrustServerCertificate=True;";

public static SqlConnection GetConn()

{

var c = new SqlConnection(ConnString);

c.Open();

return c;

}

public static DataTable Table(string sql, params SqlParameter[] ps)

{

using (var c = GetConn())

using (var cmd = new SqlCommand(sql, c))

{

if (ps != null && ps.Length > 0) cmd.Parameters.AddRange(ps);

using (var da = new SqlDataAdapter(cmd))

{

var dt = new DataTable();

da.Fill(dt);

return dt;

}

}

}

public static int Exec(string sql, params SqlParameter[] ps)

{

using (var c = GetConn())

using (var cmd = new SqlCommand(sql, c))

{

if (ps != null && ps.Length > 0) cmd.Parameters.AddRange(ps);

return cmd.ExecuteNonQuery();

}

}

}

}

**AddEditOrderForm:**

using System;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Windows.Forms;

using NewTechApp.Data;

namespace NewTechApp.UI

{

public partial class AddEditOrderForm : Form

{

private readonly int? \_orderId;

private TextBox txtArticle, txtCost;

private ComboBox cmbStatus, cmbPartner;

private DateTimePicker dtpDate;

private Button btnSave, btnCancel;

public AddEditOrderForm(int? orderId)

{

\_orderId = orderId;

Text = \_orderId.HasValue ? "Изменение заказа" : "Добавление заказа";

Font = new Font("Bahnschrift Light SemiCondensed", 10f);

BackColor = Color.White;

StartPosition = FormStartPosition.CenterParent;

Width = 560; Height = 380;

var lbl1 = new Label { Text = "Артикул:", Left = 30, Top = 30, Width = 150, Font = new Font("Bahnschrift Light SemiCondensed", 10f) };

txtArticle = new TextBox { Left = 190, Top = 30, Width = 300, Font = new Font("Bahnschrift Light SemiCondensed", 10f) };

var lbl2 = new Label { Text = "Статус заказа:", Left = 30, Top = 70, Width = 150, Font = new Font("Bahnschrift Light SemiCondensed", 10f) };

cmbStatus = new ComboBox { Left = 190, Top = 70, Width = 300, DropDownStyle = ComboBoxStyle.DropDownList, Font = new Font("Bahnschrift Light SemiCondensed", 10f) };

var lbl3 = new Label { Text = "Дата заказа:", Left = 30, Top = 110, Width = 150, Font = new Font("Bahnschrift Light SemiCondensed", 10f) };

dtpDate = new DateTimePicker { Left = 190, Top = 110, Width = 300, Format = DateTimePickerFormat.Short, Font = new Font("Bahnschrift Light SemiCondensed", 10f) };

var lbl4 = new Label { Text = "Стоимость (р):", Left = 30, Top = 150, Width = 150, Font = new Font("Bahnschrift Light SemiCondensed", 10f) };

txtCost = new TextBox { Left = 190, Top = 150, Width = 300, Font = new Font("Bahnschrift Light SemiCondensed", 10f) };

var lbl5 = new Label { Text = "Партнёр:", Left = 30, Top = 190, Width = 150, Font = new Font("Bahnschrift Light SemiCondensed", 10f) };

cmbPartner = new ComboBox { Left = 190, Top = 190, Width = 300, DropDownStyle = ComboBoxStyle.DropDownList, Font = new Font("Bahnschrift Light SemiCondensed", 10f) };

btnSave = new Button

{

Text = "Сохранить",

Left = 190,

Top = 240,

Width = 140,

BackColor = ColorTranslator.FromHtml("#0C4882"),

ForeColor = Color.White,

Font = new Font("Bahnschrift Light SemiCondensed", 10f, FontStyle.Bold)

};

btnSave.Click += (s, e) => Save();

btnCancel = new Button

{

Text = "Отмена",

Left = 350,

Top = 240,

Width = 140,

BackColor = ColorTranslator.FromHtml("#BBDCFA"),

ForeColor = Color.Black,

Font = new Font("Bahnschrift Light SemiCondensed", 10f)

};

btnCancel.Click += (s, e) => DialogResult = DialogResult.Cancel;

Controls.AddRange(new Control[] { lbl1, txtArticle, lbl2, cmbStatus, lbl3, dtpDate, lbl4, txtCost, lbl5, cmbPartner, btnSave, btnCancel });

Load += AddEditOrderForm\_Load;

}

private void AddEditOrderForm\_Load(object sender, EventArgs e)

{

LoadLookups();

if (\_orderId.HasValue) LoadOrder(\_orderId.Value);

else dtpDate.Value = DateTime.Today;

}

private bool ValidateForm(out decimal cost)

{

cost = 0m;

if (string.IsNullOrWhiteSpace(txtArticle.Text))

{

MessageBox.Show("Укажите артикул.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

txtArticle.Focus(); return false;

}

if (cmbStatus.SelectedValue == null)

{

MessageBox.Show("Выберите статус заказа.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

cmbStatus.DroppedDown = true; return false;

}

if (!decimal.TryParse(txtCost.Text.Replace(',', '.'),

System.Globalization.NumberStyles.Any,

System.Globalization.CultureInfo.InvariantCulture, out cost) || cost < 0m)

{

MessageBox.Show("Стоимость должна быть неотрицательным числом.",

"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

txtCost.Focus(); return false;

}

if (cmbPartner.SelectedValue == null)

{

MessageBox.Show("Выберите партнёра.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

cmbPartner.DroppedDown = true; return false;

}

return true;

}

private void LoadLookups()

{

// Загрузка статусов из таблицы OrderStatuses

var st = Db.Table("SELECT StatusID, StatusName FROM dbo.OrderStatuses ORDER BY StatusName");

cmbStatus.DisplayMember = "StatusName";

cmbStatus.ValueMember = "StatusID";

cmbStatus.DataSource = st;

// Загрузка партнеров

var pr = Db.Table("SELECT PartnerID, PartnerName FROM dbo.Partners ORDER BY PartnerName");

cmbPartner.DisplayMember = "PartnerName";

cmbPartner.ValueMember = "PartnerID";

cmbPartner.DataSource = pr;

}

private void LoadOrder(int id)

{

// ИСПРАВЛЕННЫЙ ЗАПРОС - используем только существующие столбцы

var dt = Db.Table(@"

SELECT OrderID, Article, Status, OrderDate, TotalAmount, PartnerID

FROM dbo.Orders WHERE OrderID=@id", new SqlParameter("@id", id));

if (dt.Rows.Count == 0)

{

MessageBox.Show("Заказ не найден.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

DialogResult = DialogResult.Cancel; return;

}

var r = dt.Rows[0];

txtArticle.Text = Convert.ToString(r["Article"]);

// ИСПРАВЛЕНО - ищем StatusID по имени статуса из OrderStatuses

var statusName = Convert.ToString(r["Status"]);

foreach (DataRowView item in cmbStatus.Items)

{

if (item["StatusName"].ToString() == statusName)

{

cmbStatus.SelectedValue = item["StatusID"];

break;

}

}

dtpDate.Value = Convert.ToDateTime(r["OrderDate"]);

txtCost.Text = (r["TotalAmount"] == DBNull.Value ? 0m : Convert.ToDecimal(r["TotalAmount"])).ToString("0.00");

if (r["PartnerID"] != DBNull.Value)

{

cmbPartner.SelectedValue = Convert.ToInt32(r["PartnerID"]);

}

}

private void Save()

{

decimal cost;

if (!ValidateForm(out cost)) return;

try

{

if (\_orderId.HasValue)

{

// ИСПРАВЛЕНО - используем имя статуса вместо ID

var statusName = cmbStatus.Text;

Db.Exec(@"

UPDATE dbo.Orders

SET Article=@a, Status=@s, OrderDate=@d, TotalAmount=@c, PartnerID=@p

WHERE OrderID=@id",

new SqlParameter("@a", txtArticle.Text.Trim()),

new SqlParameter("@s", statusName), // Сохраняем имя статуса

new SqlParameter("@d", dtpDate.Value.Date),

new SqlParameter("@c", cost),

new SqlParameter("@p", (int)cmbPartner.SelectedValue),

new SqlParameter("@id", \_orderId.Value));

}

else

{

var statusName = cmbStatus.Text;

Db.Exec(@"

INSERT INTO dbo.Orders(Article, Status, OrderDate, TotalAmount, PartnerID)

VALUES(@a, @s, @d, @c, @p)",

new SqlParameter("@a", txtArticle.Text.Trim()),

new SqlParameter("@s", statusName), // Сохраняем имя статуса

new SqlParameter("@d", dtpDate.Value.Date),

new SqlParameter("@c", cost),

new SqlParameter("@p", (int)cmbPartner.SelectedValue));

}

MessageBox.Show("Данные заказа сохранены.",

"Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

DialogResult = DialogResult.OK;

}

catch (Exception ex)

{

MessageBox.Show("Не удалось сохранить заказ. Причина: " + ex.Message,

"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

**LoginHistoryForm.cs**

using System;

using System.Data;

using System.Drawing;

using System.Windows.Forms;

using NewTechApp.Data;

namespace NewTechApp.UI

{

public partial class LoginHistoryForm : Form

{

TextBox txtFilter;

DataGridView grid;

public LoginHistoryForm()

{

Text = "История входа";

Font = new Font("Bahnschrift Light SemiCondensed", 10f);

BackColor = Color.White;

StartPosition = FormStartPosition.CenterParent;

Width = 900;

Height = 520;

// --- верхняя панель с фильтром ---

var top = new Panel { Dock = DockStyle.Top, Height = 60, BackColor = Color.White };

var lbl = new Label { Text = "Фильтр по логину:", Left = 12, Top = 18, Width = 150, Font = new Font("Bahnschrift Light SemiCondensed", 10f) };

txtFilter = new TextBox { Left = 160, Top = 16, Width = 220, Font = new Font("Bahnschrift Light SemiCondensed", 10f) };

txtFilter.TextChanged += (s, e) => ApplyFilter();

top.Controls.Add(lbl);

top.Controls.Add(txtFilter);

Controls.Add(top);

// --- таблица ---

grid = new DataGridView

{

Dock = DockStyle.Fill,

ReadOnly = true,

AllowUserToAddRows = false,

AllowUserToDeleteRows = false,

AllowUserToResizeRows = false,

RowHeadersVisible = false,

AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill,

BackgroundColor = Color.White,

BorderStyle = BorderStyle.None,

SelectionMode = DataGridViewSelectionMode.FullRowSelect,

MultiSelect = false,

ColumnHeadersHeight = 36,

ColumnHeadersHeightSizeMode = DataGridViewColumnHeadersHeightSizeMode.DisableResizing,

Font = new Font("Bahnschrift Light SemiCondensed", 10f)

};

grid.EnableHeadersVisualStyles = false;

grid.ColumnHeadersDefaultCellStyle.BackColor = ColorTranslator.FromHtml("#BBDCFA");

grid.ColumnHeadersDefaultCellStyle.ForeColor = Color.Black;

grid.ColumnHeadersDefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;

grid.ColumnHeadersDefaultCellStyle.Font = new Font("Bahnschrift Light SemiCondensed", 10f, FontStyle.Bold);

Controls.Add(grid);

Load += LoginHistoryForm\_Load;

}

void LoginHistoryForm\_Load(object sender, EventArgs e)

{

var dt = Db.Table(

"SELECT EntryID, AttemptAt, Login, Success, Reason " +

"FROM dbo.LoginHistory ORDER BY AttemptAt DESC");

grid.DataSource = dt;

// читаемые заголовки и размеры

if (grid.Columns["EntryID"] != null)

{

grid.Columns["EntryID"].HeaderText = "ID записи";

grid.Columns["EntryID"].FillWeight = 10;

grid.Columns["EntryID"].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

}

if (grid.Columns["AttemptAt"] != null)

{

grid.Columns["AttemptAt"].HeaderText = "Дата и время";

grid.Columns["AttemptAt"].FillWeight = 22;

grid.Columns["AttemptAt"].DefaultCellStyle.Format = "dd.MM.yyyy HH:mm";

}

if (grid.Columns["Login"] != null)

{

grid.Columns["Login"].HeaderText = "Логин";

grid.Columns["Login"].FillWeight = 18;

}

if (grid.Columns["Success"] != null)

{

grid.Columns["Success"].HeaderText = "Успешно";

grid.Columns["Success"].FillWeight = 10;

grid.Columns["Success"].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

}

if (grid.Columns["Reason"] != null)

{

grid.Columns["Reason"].HeaderText = "Причина";

grid.Columns["Reason"].FillWeight = 40;

}

}

void ApplyFilter()

{

var dt = grid.DataSource as DataTable;

if (dt == null) return;

var dv = dt.DefaultView;

var q = txtFilter.Text.Replace("'", "''");

dv.RowFilter = string.IsNullOrWhiteSpace(q) ? "" : "Login LIKE '%" + q + "%'";

dv.Sort = "AttemptAt DESC";

}

}

}

**MainForm.cs:**

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

using NewTechApp.Data;

namespace NewTechApp.UI

{

public partial class MainForm : Form

{

// Таблицы / колонки

const string T\_PRODUCTS = "dbo.Products";

const string COL\_ID = "ProductID";

// данные пользователя

readonly string \_login, \_fullName, \_role, \_photoUrl;

// права

bool \_canEdit = false;

// UI состояние

string \_currentSort = "StockQty DESC";

bool \_isReady = false;

Image \_phImg;

Dictionary<int, Image> \_imgCache = new Dictionary<int, Image>();

CancellationTokenSource \_imgCts;

public MainForm(string login, string fullName, string role, string photoUrl)

{

InitializeComponent();

\_login = login; \_fullName = fullName; \_role = role; \_photoUrl = photoUrl;

\_canEdit = string.Equals(\_role, "Admin", StringComparison.OrdinalIgnoreCase)

|| string.Equals(\_role, "Manager", StringComparison.OrdinalIgnoreCase);

}

private void MainForm\_Load(object sender, EventArgs e)

{

// Настройка формы

Text = "NewTech — список товаров";

Font = new Font("Bahnschrift Light SemiCondensed", 10f);

BackColor = Color.White;

StartPosition = FormStartPosition.CenterScreen;

// Настройка кнопок по роли

btnOrders.Visible = \_canEdit;

btnAdd.Visible = \_canEdit;

// Правильные цвета кнопок

btnOrders.BackColor = ColorTranslator.FromHtml("#0C4882");

btnOrders.ForeColor = Color.White;

btnOrders.Font = new Font("Bahnschrift Light SemiCondensed", 10f, FontStyle.Bold);

btnAdd.BackColor = ColorTranslator.FromHtml("#0C4882");

btnAdd.ForeColor = Color.White;

btnAdd.Font = new Font("Bahnschrift Light SemiCondensed", 10f, FontStyle.Bold);

btnHistory.BackColor = ColorTranslator.FromHtml("#BBDCFA");

btnHistory.ForeColor = Color.Black;

btnHistory.Font = new Font("Bahnschrift Light SemiCondensed", 10f);

btnLogout.BackColor = ColorTranslator.FromHtml("#BBDCFA");

btnLogout.ForeColor = Color.Black;

btnLogout.Font = new Font("Bahnschrift Light SemiCondensed", 10f);

try

{

if (!string.IsNullOrEmpty(\_photoUrl) && System.IO.File.Exists(\_photoUrl))

picUser.Image = Image.FromFile(\_photoUrl);

else

picUser.Image = MakePlaceholder("User");

}

catch { }

lblUser.Text = \_fullName + " — " + \_role;

lblUser.Font = new Font("Bahnschrift Light SemiCondensed", 10f);

\_phImg = MakePlaceholder("img");

LoadSuppliers();

LoadProducts();

\_isReady = true;

ApplyFilters();

UpdateDiscountColumn();

RenderCards();

}

Image MakePlaceholder(string text)

{

var bmp = new Bitmap(96, 96);

using (var g = Graphics.FromImage(bmp))

using (var b = new SolidBrush(Color.LightGray))

using (var p = new Pen(Color.DarkGray))

using (var f = new Font("Bahnschrift Light SemiCondensed", 10, FontStyle.Bold))

{

g.FillRectangle(b, 0, 0, bmp.Width, bmp.Height);

g.DrawRectangle(p, 0, 0, bmp.Width - 1, bmp.Height - 1);

var sz = g.MeasureString(text, f);

g.DrawString(text, f, Brushes.Black, (bmp.Width - sz.Width) / 2, (bmp.Height - sz.Height) / 2);

}

return bmp;

}

Image LoadImageNoLock(string path)

{

try

{

using (var fs = new System.IO.FileStream(path, System.IO.FileMode.Open, System.IO.FileAccess.Read, System.IO.FileShare.ReadWrite))

using (var ms = new System.IO.MemoryStream())

{

fs.CopyTo(ms);

return Image.FromStream(new System.IO.MemoryStream(ms.ToArray()));

}

}

catch { return null; }

}

void LoadSuppliers()

{

var dt = Db.Table("SELECT SupplierID, SupplierName FROM dbo.Suppliers ORDER BY SupplierName");

var ds = dt.Clone();

ds.Rows.Add(DBNull.Value, "Все поставщики");

foreach (DataRow r in dt.Rows) ds.ImportRow(r);

cmbSuppliers.DisplayMember = "SupplierName";

cmbSuppliers.ValueMember = "SupplierID";

cmbSuppliers.DataSource = ds;

cmbSuppliers.SelectedIndex = 0;

cmbSuppliers.Font = new Font("Bahnschrift Light SemiCondensed", 10f);

}

void LoadProducts()

{

var sql = @"

SELECT p.ProductID, p.ProductName, p.Article, p.MinPartnerPrice,

NULL AS PromoPrice, -- Временная заглушка

s.SupplierName,

ISNULL(m.StockQty, 0) AS StockQty,

p.ImageUrl

FROM dbo.Products p

LEFT JOIN dbo.Suppliers s ON s.SupplierID = p.SupplierID

LEFT JOIN (

SELECT pm.ProductID, SUM(pm.RequiredQty) AS StockQty

FROM dbo.ProductMaterials pm

GROUP BY pm.ProductID

) m ON m.ProductID = p.ProductID

";

var dt = Db.Table(sql);

if (!dt.Columns.Contains("ActiveDiscount")) dt.Columns.Add("ActiveDiscount", typeof(string));

if (!dt.Columns.Contains("FinalPrice")) dt.Columns.Add("FinalPrice", typeof(decimal));

grid.DataSource = dt;

if (grid.Columns.Contains("ImageUrl")) grid.Columns["ImageUrl"].Visible = false;

\_currentSort = "StockQty DESC";

dt.DefaultView.Sort = \_currentSort;

SetColumnsLook();

}

void SetColumnsLook()

{

grid.Font = new Font("Bahnschrift Light SemiCondensed", 10f);

if (grid.Columns.Contains("ProductID"))

{

grid.Columns["ProductID"].HeaderText = "ID";

grid.Columns["ProductID"].FillWeight = 70;

grid.Columns["ProductID"].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

}

if (grid.Columns.Contains("ProductName"))

{

grid.Columns["ProductName"].HeaderText = "Товар";

grid.Columns["ProductName"].FillWeight = 420;

grid.Columns["ProductName"].DefaultCellStyle.Padding = new Padding(6, 0, 0, 0);

}

if (grid.Columns.Contains("Article"))

{

grid.Columns["Article"].HeaderText = "Артикул";

grid.Columns["Article"].FillWeight = 100;

}

if (grid.Columns.Contains("SupplierName"))

{

grid.Columns["SupplierName"].HeaderText = "Поставщик";

grid.Columns["SupplierName"].FillWeight = 120;

}

if (grid.Columns.Contains("StockQty"))

{

grid.Columns["StockQty"].HeaderText = "Остаток";

grid.Columns["StockQty"].DefaultCellStyle.Format = "N0";

grid.Columns["StockQty"].FillWeight = 90;

}

if (grid.Columns.Contains("MinPartnerPrice"))

{

grid.Columns["MinPartnerPrice"].HeaderText = "Мин. цена";

grid.Columns["MinPartnerPrice"].DefaultCellStyle.Format = "N2";

}

if (grid.Columns.Contains("PromoPrice"))

{

grid.Columns["PromoPrice"].HeaderText = "Акц. цена";

grid.Columns["PromoPrice"].DefaultCellStyle.Format = "N2";

}

if (grid.Columns.Contains("FinalPrice"))

{

grid.Columns["FinalPrice"].HeaderText = "Итоговая цена";

grid.Columns["FinalPrice"].DefaultCellStyle.Format = "N2";

grid.Columns["FinalPrice"].FillWeight = 120;

}

if (grid.Columns.Contains("ActiveDiscount"))

grid.Columns["ActiveDiscount"].HeaderText = "Скидка";

}

void ApplyFilters()

{

if (!\_isReady || grid.DataSource == null) return;

var dt = grid.DataSource as DataTable;

if (dt == null) return;

var dv = dt.DefaultView;

var list = new List<string>();

if (!string.IsNullOrWhiteSpace(txtSearch.Text))

{

var q = txtSearch.Text.Replace("'", "''");

list.Add("(ProductName LIKE '%" + q + "%' OR Article LIKE '%" + q + "%' OR SupplierName LIKE '%" + q + "%')");

}

if (cmbSuppliers.SelectedValue != null && cmbSuppliers.SelectedValue != DBNull.Value)

{

var name = cmbSuppliers.Text.Replace("'", "''");

list.Add("SupplierName = '" + name + "'");

}

dv.RowFilter = string.Join(" AND ", list.ToArray());

dv.Sort = \_currentSort;

}

decimal CalcDiscount(decimal vol)

{

if (vol <= 10000) return 0m;

if (vol <= 50000) return 0.05m;

if (vol <= 300000) return 0.10m;

if (vol <= 1000000) return 0.15m;

return 0.20m;

}

void UpdateDiscountColumn()

{

if (!\_isReady || grid.DataSource == null) return;

var dt = grid.DataSource as DataTable;

if (dt == null) return;

var d = CalcDiscount((decimal)numVolume.Value);

foreach (DataRow r in dt.Rows)

{

r["ActiveDiscount"] = string.Format("{0:P0}", d);

var basePrice = r["PromoPrice"] != DBNull.Value ? Convert.ToDecimal(r["PromoPrice"])

: Convert.ToDecimal(r["MinPartnerPrice"]);

r["FinalPrice"] = Math.Round(basePrice \* (1 - d), 2);

}

}

void RenderCards()

{

if (grid.DataSource == null || flw == null) return;

var dt = grid.DataSource as DataTable;

var dv = dt.DefaultView;

\_imgCts?.Cancel();

\_imgCts = new CancellationTokenSource();

flw.SuspendLayout();

try

{

flw.Controls.Clear();

foreach (DataRowView v in dv)

{

var r = v.Row;

int id = Convert.ToInt32(r["ProductID"]);

string name = Convert.ToString(r["ProductName"]);

string article = Convert.ToString(r["Article"]);

string supplier = Convert.ToString(r["SupplierName"]);

decimal stock = Convert.ToDecimal(r["StockQty"]);

decimal minPrice = r["MinPartnerPrice"] == DBNull.Value ? 0 : Convert.ToDecimal(r["MinPartnerPrice"]);

decimal promo = r["PromoPrice"] == DBNull.Value ? 0 : Convert.ToDecimal(r["PromoPrice"]);

string imgPath = Convert.ToString(r["ImageUrl"]);

var card = BuildCardControl(id, name, article, supplier, stock, minPrice, promo);

flw.Controls.Add(card.root);

\_ = LoadCardImageAsync(card.pic, id, imgPath, \_imgCts.Token);

}

}

finally { flw.ResumeLayout(); }

}

async Task LoadCardImageAsync(PictureBox pic, int id, string path, CancellationToken token)

{

if (token.IsCancellationRequested) return;

if (\_imgCache.TryGetValue(id, out var cached))

{

if (!pic.IsDisposed) pic.BeginInvoke((Action)(() => pic.Image = cached));

return;

}

Image img = null;

if (!string.IsNullOrEmpty(path) && System.IO.File.Exists(path))

{

try { img = await Task.Run(() => LoadImageNoLock(path), token); } catch { img = null; }

}

if (token.IsCancellationRequested) return;

var ready = img ?? \_phImg;

\_imgCache[id] = ready;

if (!pic.IsDisposed) pic.BeginInvoke((Action)(() => pic.Image = ready));

}

(Panel root, PictureBox pic) BuildCardControl(int id, string name, string article, string supplier,

decimal stock, decimal minPrice, decimal promo)

{

var root = new Panel { Width = Math.Max(860, flw.ClientSize.Width - 30), Height = 110, Margin = new Padding(8), BackColor = Color.White, BorderStyle = BorderStyle.FixedSingle };

var picBox = new PictureBox { Left = 10, Top = 10, Width = 90, Height = 90, SizeMode = PictureBoxSizeMode.Zoom, Image = \_phImg };

root.Controls.Add(picBox);

var center = new Panel { Left = 110, Top = 10, Width = root.Width - 110 - 150 - 20, Height = 90, Anchor = AnchorStyles.Left | AnchorStyles.Right | AnchorStyles.Top };

root.Controls.Add(center);

var lblTitle = new Label { Left = 8, Top = 0, Width = center.Width - 8, Height = 24, Font = new Font("Bahnschrift Light SemiCondensed", 10f, FontStyle.Bold), Text = $"Товар | {name}" };

center.Controls.Add(lblTitle);

var lblLine1 = new Label { Left = 8, Top = 24, Width = center.Width - 8, Height = 20, Text = $"Артикул: {article} Поставщик: {supplier}", Font = new Font("Bahnschrift Light SemiCondensed", 9f) };

center.Controls.Add(lblLine1);

var lblLine2 = new Label { Left = 8, Top = 44, Width = center.Width - 8, Height = 20, Text = $"Остаток: {stock:N0}", Font = new Font("Bahnschrift Light SemiCondensed", 9f) };

center.Controls.Add(lblLine2);

var lblLine3 = new Label { Left = 8, Top = 64, Width = center.Width - 8, Height = 20, Font = new Font("Bahnschrift Light SemiCondensed", 9f) };

if (promo > 0) lblLine3.Text = $"Мин. цена: {minPrice:N2} р Акц. цена: {promo:N2} р";

else lblLine3.Text = $"Мин. цена: {minPrice:N2} р";

center.Controls.Add(lblLine3);

var lblBatch = new Label { Left = center.Width - 250, Top = 0, Width = 240, Height = 24, Anchor = AnchorStyles.Top | AnchorStyles.Right, TextAlign = ContentAlignment.MiddleRight, Font = new Font("Bahnschrift Light SemiCondensed", 9f) };

center.Controls.Add(lblBatch);

var right = new Panel { Left = root.Width - 150 - 10, Top = 10, Width = 140, Height = 90, BorderStyle = BorderStyle.FixedSingle, Anchor = AnchorStyles.Top | AnchorStyles.Right };

var lblHdr = new Label { Left = 6, Top = 6, Width = 128, Height = 36, Text = "Действующая\nскидка", TextAlign = ContentAlignment.MiddleCenter, Font = new Font("Bahnschrift Light SemiCondensed", 8f) };

var lblDisc = new Label { Left = 6, Top = 52, Width = 128, Height = 32, TextAlign = ContentAlignment.MiddleCenter, Font = new Font("Bahnschrift Light SemiCondensed", 12f, FontStyle.Bold) };

right.Controls.Add(lblHdr); right.Controls.Add(lblDisc);

root.Controls.Add(right);

if (stock <= 0) root.BackColor = Color.MistyRose;

ApplyCardDiscount(root, lblDisc, lblBatch, minPrice, promo);

if (\_canEdit)

{

var menu = new ContextMenuStrip();

menu.Items.Add("Редактировать", null, (s, e) => EditProductById(id));

menu.Items.Add("Изменить акционную цену", null, (s, e) =>

{

string sVal = Prompt.Show("Новая акционная цена:", "Изменить цену");

decimal v;

if (decimal.TryParse(sVal, out v))

{

try

{

Cursor.Current = Cursors.WaitCursor;

Db.Exec($"UPDATE {T\_PRODUCTS} SET PromoPrice=@p WHERE {COL\_ID}=@id",

new SqlParameter("@p", v), new SqlParameter("@id", id));

ReloadAndReapply();

}

catch (Exception ex) { MessageBox.Show("Не удалось сохранить: " + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error); }

finally { Cursor.Current = Cursors.Default; }

}

});

menu.Items.Add("Удалить товар", null, (s, e) => DeleteProductById(id));

root.ContextMenuStrip = menu;

root.DoubleClick += (s, e) => EditProductById(id);

}

return (root, picBox);

}

void ApplyCardDiscount(Panel root, Label lblDisc, Label lblBatch, decimal minPrice, decimal promo)

{

var d = CalcDiscount((decimal)numVolume.Value);

lblDisc.Text = $"{Math.Round(d \* 100m)}%";

decimal baseForFinal = promo > 0 ? promo : minPrice;

decimal final = Math.Round(baseForFinal \* (1 - d), 2);

lblBatch.Text = $"Стоимость партии: {final:N2} р";

Color bg = d == 0m ? Color.White :

d <= 0.05m ? Color.FromArgb(235, 250, 235) :

d <= 0.10m ? Color.FromArgb(215, 245, 215) :

d <= 0.15m ? Color.FromArgb(195, 240, 195) :

Color.FromArgb(255, 230, 200);

root.BackColor = bg;

}

void Grid\_CellFormatting(object sender, DataGridViewCellFormattingEventArgs e)

{

if (grid.Columns[e.ColumnIndex].Name == "StockQty" && e.Value != null)

{

decimal stock;

if (decimal.TryParse(e.Value.ToString(), out stock) && stock <= 0)

grid.Rows[e.RowIndex].DefaultCellStyle.BackColor = Color.MistyRose;

}

if (grid.Columns[e.ColumnIndex].Name == "MinPartnerPrice" &&

grid.Rows[e.RowIndex].Cells["PromoPrice"].Value != DBNull.Value)

{

grid.Rows[e.RowIndex].Cells["MinPartnerPrice"].Style.ForeColor = Color.Red;

grid.Rows[e.RowIndex].Cells["MinPartnerPrice"].Style.Font = new Font(grid.Font, FontStyle.Strikeout);

}

if (grid.Columns.Contains("ActiveDiscount"))

{

var text = Convert.ToString(grid.Rows[e.RowIndex].Cells["ActiveDiscount"].Value);

int d;

if (!string.IsNullOrEmpty(text) && text.EndsWith("%") && int.TryParse(text.TrimEnd('%'), out d))

{

var st = grid.Rows[e.RowIndex].DefaultCellStyle;

if (d == 0) st.BackColor = Color.White;

else if (d <= 5) st.BackColor = Color.FromArgb(235, 250, 235);

else if (d <= 10) st.BackColor = Color.FromArgb(215, 245, 215);

else if (d <= 15) st.BackColor = Color.FromArgb(195, 240, 195);

else st.BackColor = Color.FromArgb(255, 230, 200);

}

}

}

void Grid\_ColumnHeaderMouseClick(object sender, DataGridViewCellMouseEventArgs e)

{

var name = grid.Columns[e.ColumnIndex].Name;

if (string.IsNullOrEmpty(\_currentSort) || !\_currentSort.StartsWith(name))

\_currentSort = name + " ASC";

else

\_currentSort = \_currentSort.EndsWith("ASC") ? name + " DESC" : name + " ASC";

var dt = grid.DataSource as DataTable;

if (dt != null) dt.DefaultView.Sort = \_currentSort;

RenderCards();

}

void EditPromo()

{

if (!\_canEdit) { MessageBox.Show("Недостаточно прав.", "Достав запрещён", MessageBoxButtons.OK, MessageBoxIcon.Warning); return; }

if (grid.CurrentRow == null) return;

int id = Convert.ToInt32(grid.CurrentRow.Cells["ProductID"].Value);

string s = Prompt.Show("Новая акционная цена:", "Изменить цену");

decimal v;

if (decimal.TryParse(s, out v))

{

try

{

grid.Enabled = false; Cursor.Current = Cursors.WaitCursor;

Db.Exec($"UPDATE {T\_PRODUCTS} SET PromoPrice=@p WHERE {COL\_ID}=@id",

new SqlParameter("@p", v), new SqlParameter("@id", id));

}

catch (Exception ex)

{

MessageBox.Show("Не удалось сохранить: " + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally { Cursor.Current = Cursors.Default; grid.Enabled = true; }

ReloadAndReapply();

}

}

void DeleteProduct()

{

if (!\_canEdit) { MessageBox.Show("Недостаточно прав.", "Доступ запрещён", MessageBoxButtons.OK, MessageBoxIcon.Warning); return; }

if (grid.CurrentRow == null) return;

int id = Convert.ToInt32(grid.CurrentRow.Cells["ProductID"].Value);

DeleteProductById(id);

}

void DeleteProductById(int id)

{

if (MessageBox.Show("Удалить товар?", "Подтверждение",

MessageBoxButtons.YesNo, MessageBoxIcon.Question) != DialogResult.Yes) return;

if (IsProductInOrders(id))

{

MessageBox.Show("Удаление невозможно: товар присутствует в заказах.", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

try

{

grid.Enabled = false; Cursor.Current = Cursors.WaitCursor;

Db.Exec(@"

IF OBJECT\_ID('dbo.ProductMaterials') IS NOT NULL

DELETE FROM dbo.ProductMaterials WHERE ProductID=@id;

DELETE FROM " + T\_PRODUCTS + " WHERE " + COL\_ID + @"=@id;",

new SqlParameter("@id", id));

}

catch (Exception ex)

{

MessageBox.Show("Не удалось удалить: " + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

Cursor.Current = Cursors.Default; grid.Enabled = true;

}

ReloadAndReapply();

}

bool IsProductInOrders(int productId)

{

var t = Db.Table("SELECT OBJECT\_ID('dbo.OrderItems') AS ObjId");

bool tableExists = t.Rows.Count > 0 && t.Rows[0]["ObjId"] != DBNull.Value;

if (!tableExists) return false;

var dt = Db.Table("SELECT TOP 1 1 FROM dbo.OrderItems WHERE ProductID=@id",

new SqlParameter("@id", productId));

return dt.Rows.Count > 0;

}

void AddProduct()

{

if (!\_canEdit) { MessageBox.Show("Недостаточно прав.", "Доступ запрещён", MessageBoxButtons.OK, MessageBoxIcon.Warning); return; }

if (Application.OpenForms["ProductForm"] != null)

{

MessageBox.Show("Окно редактирования уже открыто.", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

return;

}

using (var form = new ProductForm(null))

{

form.Name = "ProductForm";

if (form.ShowDialog(this) == DialogResult.OK)

ReloadAndReapply();

}

}

void EditProductById(int id)

{

if (!\_canEdit) { MessageBox.Show("Недостаточно прав.", "Доступ запрещён", MessageBoxButtons.OK, MessageBoxIcon.Warning); return; }

if (Application.OpenForms["ProductForm"] != null)

{

MessageBox.Show("Окно редактирования уже открыто.", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

return;

}

using (var form = new ProductForm(id))

{

form.Name = "ProductForm";

if (form.ShowDialog(this) == DialogResult.OK)

ReloadAndReapply();

}

}

void ReloadAndReapply()

{

grid.Enabled = false;

Cursor.Current = Cursors.WaitCursor;

try

{

grid.SuspendLayout();

LoadProducts();

ApplyFilters();

UpdateDiscountColumn();

RenderCards();

}

finally

{

grid.ResumeLayout();

grid.Enabled = true;

Cursor.Current = Cursors.Default;

}

}

protected override void OnHandleCreated(EventArgs e)

{

base.OnHandleCreated(e);

try

{

typeof(DataGridView).InvokeMember("DoubleBuffered",

System.Reflection.BindingFlags.NonPublic | System.Reflection.BindingFlags.Instance | System.Reflection.BindingFlags.SetProperty,

null, grid, new object[] { true });

}

catch { }

}

private static class Prompt

{

public static string Show(string text, string caption, string defaultValue = "")

{

using (var form = new Form())

using (var lbl = new Label())

using (var txt = new TextBox())

using (var ok = new Button())

using (var cancel = new Button())

{

form.Text = caption;

form.FormBorderStyle = FormBorderStyle.FixedDialog;

form.StartPosition = FormStartPosition.CenterParent;

form.ClientSize = new Size(420, 140);

form.MinimizeBox = false; form.MaximizeBox = false;

form.ShowIcon = false; form.ShowInTaskbar = false;

lbl.Text = text; lbl.SetBounds(12, 12, 394, 24);

lbl.Font = new Font("Bahnschrift Light SemiCondensed", 10f);

txt.Text = defaultValue; txt.SetBounds(12, 40, 394, 28);

txt.Font = new Font("Bahnschrift Light SemiCondensed", 10f);

ok.Text = "OK"; ok.DialogResult = DialogResult.OK; ok.SetBounds(230, 80, 80, 28);

ok.BackColor = ColorTranslator.FromHtml("#0C4882"); ok.ForeColor = Color.White;

ok.Font = new Font("Bahnschrift Light SemiCondensed", 10f);

cancel.Text = "Отмена"; cancel.DialogResult = DialogResult.Cancel; cancel.SetBounds(326, 80, 80, 28);

cancel.BackColor = ColorTranslator.FromHtml("#BBDCFA"); cancel.ForeColor = Color.Black;

cancel.Font = new Font("Bahnschrift Light SemiCondensed", 10f);

form.Controls.AddRange(new Control[] { lbl, txt, ok, cancel });

form.AcceptButton = ok; form.CancelButton = cancel;

return form.ShowDialog() == DialogResult.OK ? txt.Text : string.Empty;

}

}

}

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

{

new LoginHistoryForm().ShowDialog(this);

}

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

{

using (var f = new OrdersForm(\_role))

f.ShowDialog(this);

}

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

{

AddProduct();

}

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

{

var loginForm = new NewTechApp.Auth.LoginForm();

loginForm.Show();

this.Hide();

}

private void cmbSuppliers\_SelectedIndexChanged(object sender, EventArgs e)

{

if (\_isReady) { ApplyFilters(); RenderCards(); }

}

private void txtSearch\_TextChanged(object sender, EventArgs e)

{

if (\_isReady) { ApplyFilters(); RenderCards(); }

}

private void numVolume\_ValueChanged(object sender, EventArgs e)

{

if (\_isReady) { UpdateDiscountColumn(); RenderCards(); }

}

}

}

**OrdersForm.cs:**

using System;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Windows.Forms;

using NewTechApp.Data;

namespace NewTechApp.UI

{

public partial class OrdersForm : Form

{

private readonly bool \_isAdmin;

private TextBox txtSearch;

private ComboBox cmbSort;

private Button btnAdd;

private FlowLayoutPanel flw;

private string \_currentSort = "PartnerName ASC";

private DataTable \_dt;

public OrdersForm(string role)

{

\_isAdmin = string.Equals(role, "Admin", StringComparison.OrdinalIgnoreCase);

Text = "NewTech — заказы";

Font = new Font("Bahnschrift Light SemiCondensed", 10f);

BackColor = Color.White;

StartPosition = FormStartPosition.CenterParent;

Width = 1100; Height = 700;

var layout = new TableLayoutPanel { Dock = DockStyle.Fill, ColumnCount = 1, RowCount = 2 };

layout.RowStyles.Add(new RowStyle(SizeType.Absolute, 80));

layout.RowStyles.Add(new RowStyle(SizeType.Percent, 100));

Controls.Add(layout);

var top = new Panel { Dock = DockStyle.Fill, BackColor = Color.White };

layout.Controls.Add(top, 0, 0);

var lblSearch = new Label { Text = "Поиск:", Left = 10, Top = 18, Width = 60, Font = new Font("Bahnschrift Light SemiCondensed", 10f) };

txtSearch = new TextBox { Left = 70, Top = 16, Width = 280, Font = new Font("Bahnschrift Light SemiCondensed", 10f) };

txtSearch.TextChanged += (s, e) => ApplyFilterAndRender();

var lblSort = new Label { Text = "Сортировка по партнёру:", Left = 370, Top = 18, Width = 180, Font = new Font("Bahnschrift Light SemiCondensed", 10f) };

cmbSort = new ComboBox { Left = 550, Top = 16, Width = 180, DropDownStyle = ComboBoxStyle.DropDownList, Font = new Font("Bahnschrift Light SemiCondensed", 10f) };

cmbSort.Items.AddRange(new object[] { "По возрастанию", "По убыванию" });

cmbSort.SelectedIndex = 0;

cmbSort.SelectedIndexChanged += (s, e) =>

{

\_currentSort = cmbSort.SelectedIndex == 0 ? "PartnerName ASC" : "PartnerName DESC";

ApplyFilterAndRender();

};

btnAdd = new Button

{

Text = "Добавить заказ",

Left = 750,

Top = 16,

Width = 120,

Height = 25,

BackColor = ColorTranslator.FromHtml("#0C4882"),

ForeColor = Color.White,

Font = new Font("Bahnschrift Light SemiCondensed", 10f, FontStyle.Bold),

Visible = \_isAdmin

};

btnAdd.Click += (s, e) =>

{

using (var f = new AddEditOrderForm(null))

{

if (f.ShowDialog(this) == DialogResult.OK) LoadDataAndRender();

}

};

top.Controls.Add(lblSearch);

top.Controls.Add(txtSearch);

top.Controls.Add(lblSort);

top.Controls.Add(cmbSort);

top.Controls.Add(btnAdd);

flw = new FlowLayoutPanel

{

Dock = DockStyle.Fill,

AutoScroll = true,

FlowDirection = FlowDirection.TopDown,

WrapContents = false,

Padding = new Padding(10),

Margin = new Padding(0, 20, 0, 0),

BackColor = Color.White

};

layout.Controls.Add(flw, 0, 1);

Load += OrdersForm\_Load;

}

private void OrdersForm\_Load(object sender, EventArgs e)

{

LoadDataAndRender();

}

private void LoadDataAndRender()

{

// УПРОЩЕННЫЙ ЗАПРОС БЕЗ StatusID

string sql = @"

SELECT

o.OrderID,

o.Article,

o.Status, -- Используем текстовый статус из Orders

o.OrderDate,

o.TotalAmount,

p.PartnerName,

o.PartnerID

FROM dbo.Orders o

LEFT JOIN dbo.Partners p ON p.PartnerID = o.PartnerID";

\_dt = Db.Table(sql);

ApplyFilterAndRender();

}

private void ApplyFilterAndRender()

{

if (\_dt == null) return;

var dv = \_dt.DefaultView;

// УПРОЩЕННЫЙ ФИЛЬТР БЕЗ StatusID

if (!string.IsNullOrWhiteSpace(txtSearch.Text))

{

var q = txtSearch.Text.Replace("'", "''");

var expr =

"Article LIKE '%" + q + "%' OR " +

"Status LIKE '%" + q + "%' OR " + // Фильтруем по текстовому статусу

"PartnerName LIKE '%" + q + "%'";

dv.RowFilter = expr;

}

else dv.RowFilter = "";

dv.Sort = \_currentSort;

RenderCards(dv);

}

private void RenderCards(DataView dv)

{

flw.SuspendLayout();

try

{

flw.Controls.Clear();

foreach (DataRowView v in dv)

{

var r = v.Row;

int id = Convert.ToInt32(r["OrderID"]);

string article = Convert.ToString(r["Article"]);

string status = Convert.ToString(r["Status"]); // Берем текст статуса

string partner = Convert.ToString(r["PartnerName"]);

DateTime date = Convert.ToDateTime(r["OrderDate"]);

decimal cost = r["TotalAmount"] == DBNull.Value ? 0 : Convert.ToDecimal(r["TotalAmount"]);

var card = BuildCard(id, article, status, partner, date, cost);

flw.Controls.Add(card);

}

}

finally { flw.ResumeLayout(); }

}

private Panel BuildCard(int id, string article, string status, string partner, DateTime date, decimal cost)

{

var root = new Panel

{

Width = Math.Max(860, flw.ClientSize.Width - 30),

Height = 110,

Margin = new Padding(8),

BackColor = Color.White,

BorderStyle = BorderStyle.FixedSingle,

Tag = id

};

var left = new Panel { Left = 10, Top = 10, Width = root.Width - 180, Height = 90, Anchor = AnchorStyles.Left | AnchorStyles.Right | AnchorStyles.Top };

root.Controls.Add(left);

var title = new Label

{

Text = "Артикул заказа: " + article,

Font = new Font("Bahnschrift Light SemiCondensed", 10f, FontStyle.Bold),

Left = 8,

Top = 2,

Width = left.Width - 16,

Height = 24

};

left.Controls.Add(title);

var l1 = new Label { Text = "Статус заказа: " + status, Left = 8, Top = 28, Width = left.Width - 16, Height = 20, Font = new Font("Bahnschrift Light SemiCondensed", 9f) };

var l2 = new Label { Text = "Партнёр: " + partner, Left = 8, Top = 48, Width = left.Width - 16, Height = 20, Font = new Font("Bahnschrift Light SemiCondensed", 9f) };

var l3 = new Label { Text = "Стоимость: " + cost.ToString("N2") + " р", Left = 8, Top = 68, Width = left.Width - 16, Height = 20, Font = new Font("Bahnschrift Light SemiCondensed", 9f) };

left.Controls.Add(l1);

left.Controls.Add(l2);

left.Controls.Add(l3);

var right = new Panel { Left = root.Width - 160, Top = 10, Width = 140, Height = 90, BorderStyle = BorderStyle.FixedSingle, Anchor = AnchorStyles.Top | AnchorStyles.Right };

var d = new Label { Text = "Дата заказа", Left = 6, Top = 6, Width = 128, Height = 24, TextAlign = ContentAlignment.MiddleCenter, Font = new Font("Bahnschrift Light SemiCondensed", 8f) };

var dv = new Label { Text = date.ToString("dd.MM.yyyy"), Left = 6, Top = 36, Width = 128, Height = 40, TextAlign = ContentAlignment.MiddleCenter, Font = new Font("Bahnschrift Light SemiCondensed", 12f, FontStyle.Bold) };

right.Controls.Add(d);

right.Controls.Add(dv);

root.Controls.Add(right);

root.DoubleClick += (s, e) =>

{

if (!\_isAdmin)

{

MessageBox.Show("Редактировать заказы может только администратор.", "Доступ запрещён", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

using (var f = new AddEditOrderForm(id))

{

if (f.ShowDialog(this) == DialogResult.OK) LoadDataAndRender();

}

};

var menu = new ContextMenuStrip();

if (\_isAdmin)

{

menu.Items.Add("Изменить", null, (s, e) =>

{

using (var f = new AddEditOrderForm(id))

{

if (f.ShowDialog(this) == DialogResult.OK) LoadDataAndRender();

}

});

menu.Items.Add("Удалить", null, (s, e) =>

{

if (MessageBox.Show("Удалить заказ №" + id + "?", "Подтверждение",

MessageBoxButtons.YesNo, MessageBoxIcon.Warning) != DialogResult.Yes) return;

try

{

Db.Exec("DELETE FROM dbo.Orders WHERE OrderID=@id", new SqlParameter("@id", id));

MessageBox.Show("Заказ удалён.", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

LoadDataAndRender();

}

catch (Exception ex)

{

MessageBox.Show("Не удалось удалить заказ. Причина: " + ex.Message,

"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

});

}

else

{

menu.Items.Add("Просмотр (только чтение)", null, (s, e) => { });

}

root.ContextMenuStrip = menu;

return root;

}

}

}

**ProductForm.cs:**

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.IO;

using System.Windows.Forms;

using NewTechApp.Data;

using NewTechApp.Utility;

namespace NewTechApp.UI

{

public partial class ProductForm : Form

{

private const string T\_PRODUCTS = "dbo.Products";

private const string COL\_ID = "ProductID";

private const string COL\_NAME = "ProductName";

private const string COL\_ARTICLE = "Article";

private const string COL\_SUPPLIER\_ID = "SupplierID";

private const string COL\_MIN\_PRICE = "MinPartnerPrice";

private const string COL\_PROMO\_PRICE = "PromoPrice";

private const string COL\_IMG = "ImageUrl";

private const string COL\_TYPEID = "ProductTypeID";

private readonly int? \_productId;

private string \_currentImagePath;

// UI

Label lblId;

TextBox tbId, tbName, tbArticle;

ComboBox cmbSupplier;

NumericUpDown numPrice, numPromo;

PictureBox pic;

Button btnLoad, btnSave, btnCancel;

public ProductForm(int? productId = null)

{

\_productId = productId;

BuildUi();

Text = \_productId.HasValue ? "Редактирование товара" : "Добавление товара";

if (\_productId.HasValue) LoadProduct(\_productId.Value);

}

private void BuildUi()

{

StartPosition = FormStartPosition.CenterParent;

BackColor = Color.White;

Font = new Font("Bahnschrift Light SemiCondensed", 10f);

ClientSize = new Size(680, 360);

FormBorderStyle = FormBorderStyle.FixedDialog;

MaximizeBox = false;

MinimizeBox = false;

int x1 = 16, w1 = 250, y = 16, dy = 34;

lblId = new Label { Left = x1, Top = y, Width = 140, Text = "ID (только чтение):", Font = new Font("Bahnschrift Light SemiCondensed", 10f) };

tbId = new TextBox { Left = x1 + 160, Top = y - 2, Width = 90, ReadOnly = true, Visible = \_productId.HasValue, Font = new Font("Bahnschrift Light SemiCondensed", 10f) };

if (!\_productId.HasValue) { lblId.Visible = false; }

Controls.Add(lblId); Controls.Add(tbId); y += dy;

Controls.Add(new Label { Left = x1, Top = y, Width = 140, Text = "Название \*:", Font = new Font("Bahnschrift Light SemiCondensed", 10f) });

tbName = new TextBox { Left = x1 + 160, Top = y - 2, Width = w1, Font = new Font("Bahnschrift Light SemiCondensed", 10f) };

Controls.Add(tbName); y += dy;

Controls.Add(new Label { Left = x1, Top = y, Width = 140, Text = "Артикул:", Font = new Font("Bahnschrift Light SemiCondensed", 10f) });

tbArticle = new TextBox { Left = x1 + 160, Top = y - 2, Width = w1, Font = new Font("Bahnschrift Light SemiCondensed", 10f) };

Controls.Add(tbArticle); y += dy;

Controls.Add(new Label { Left = x1, Top = y, Width = 140, Text = "Поставщик:", Font = new Font("Bahnschrift Light SemiCondensed", 10f) });

cmbSupplier = new ComboBox { Left = x1 + 160, Top = y - 2, Width = w1, DropDownStyle = ComboBoxStyle.DropDownList, Font = new Font("Bahnschrift Light SemiCondensed", 10f) };

Controls.Add(cmbSupplier); y += dy;

Controls.Add(new Label { Left = x1, Top = y, Width = 140, Text = "Мин. цена \*:", Font = new Font("Bahnschrift Light SemiCondensed", 10f) });

numPrice = new NumericUpDown { Left = x1 + 160, Top = y - 2, Width = 120, DecimalPlaces = 2, Minimum = 0, Maximum = 100000000, Font = new Font("Bahnschrift Light SemiCondensed", 10f) };

Controls.Add(numPrice); y += dy;

Controls.Add(new Label { Left = x1, Top = y, Width = 140, Text = "Акц. цена:", Font = new Font("Bahnschrift Light SemiCondensed", 10f) });

numPromo = new NumericUpDown { Left = x1 + 160, Top = y - 2, Width = 120, DecimalPlaces = 2, Minimum = 0, Maximum = 100000000, Font = new Font("Bahnschrift Light SemiCondensed", 10f) };

Controls.Add(numPromo);

// Картинка

pic = new PictureBox

{

Left = 440,

Top = 16,

Width = 220,

Height = 160,

BorderStyle = BorderStyle.FixedSingle,

SizeMode = PictureBoxSizeMode.Zoom,

BackColor = Color.White

};

Controls.Add(pic);

btnLoad = new Button

{

Left = 440,

Top = 184,

Width = 220,

Height = 28,

Text = "Загрузить изображение",

BackColor = ColorTranslator.FromHtml("#BBDCFA"),

ForeColor = Color.Black,

Font = new Font("Bahnschrift Light SemiCondensed", 10f)

};

btnLoad.Click += (s, e) => LoadImage();

Controls.Add(btnLoad);

// Кнопки

btnSave = new Button

{

Left = 440,

Top = 270,

Width = 110,

Height = 34,

Text = "Сохранить",

BackColor = ColorTranslator.FromHtml("#0C4882"),

ForeColor = Color.White,

Font = new Font("Bahnschrift Light SemiCondensed", 10f, FontStyle.Bold)

};

btnCancel = new Button

{

Left = 550,

Top = 270,

Width = 110,

Height = 34,

Text = "Отмена",

BackColor = ColorTranslator.FromHtml("#BBDCFA"),

ForeColor = Color.Black,

Font = new Font("Bahnschrift Light SemiCondensed", 10f)

};

btnCancel.Click += (s, e) => DialogResult = DialogResult.Cancel;

btnSave.Click += (s, e) => SaveProduct();

Controls.AddRange(new Control[] { btnSave, btnCancel });

Load += (s, e) => LoadSuppliers();

}

private void LoadSuppliers()

{

var dt = Db.Table("SELECT SupplierID, SupplierName FROM dbo.Suppliers ORDER BY SupplierName");

cmbSupplier.DisplayMember = "SupplierName";

cmbSupplier.ValueMember = "SupplierID";

cmbSupplier.DataSource = dt;

if (dt.Rows.Count > 0) cmbSupplier.SelectedIndex = 0;

}

private void LoadProduct(int id)

{

string sql = "SELECT " + COL\_ID + "," + COL\_NAME + "," + COL\_ARTICLE + "," + COL\_SUPPLIER\_ID + "," +

COL\_MIN\_PRICE + "," + COL\_PROMO\_PRICE + "," + COL\_IMG +

" FROM " + T\_PRODUCTS + " WHERE " + COL\_ID + "=@id";

var dt = Db.Table(sql, new SqlParameter("@id", id));

if (dt.Rows.Count == 0)

{

MessageBox.Show("Товар не найден.", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

Close();

return;

}

var r = dt.Rows[0];

tbId.Text = Convert.ToString(r[COL\_ID]);

tbName.Text = Convert.ToString(r[COL\_NAME]);

tbArticle.Text = r[COL\_ARTICLE] == DBNull.Value ? "" : Convert.ToString(r[COL\_ARTICLE]);

if (r[COL\_SUPPLIER\_ID] != DBNull.Value)

cmbSupplier.SelectedValue = Convert.ToInt32(r[COL\_SUPPLIER\_ID]);

numPrice.Value = r[COL\_MIN\_PRICE] == DBNull.Value ? 0 : Convert.ToDecimal(r[COL\_MIN\_PRICE]);

numPromo.Value = r[COL\_PROMO\_PRICE] == DBNull.Value ? 0 : Convert.ToDecimal(r[COL\_PROMO\_PRICE]);

\_currentImagePath = r[COL\_IMG] == DBNull.Value ? null : Convert.ToString(r[COL\_IMG]);

if (!string.IsNullOrEmpty(\_currentImagePath) && File.Exists(\_currentImagePath))

{

try { pic.Image = Image.FromFile(\_currentImagePath); } catch { }

}

}

private void LoadImage()

{

using (var ofd = new OpenFileDialog { Filter = "Images|\*.jpg;\*.jpeg;\*.png;\*.bmp" })

{

if (ofd.ShowDialog() == DialogResult.OK)

{

try

{

var image = Image.FromFile(ofd.FileName);

// Resize image to 300x200

var resizedImage = ImageHelper.SaveResized(image, 300, 200, \_currentImagePath);

pic.Image = Image.FromFile(resizedImage);

\_currentImagePath = resizedImage;

}

catch (Exception ex)

{

MessageBox.Show("Не удалось открыть изображение: " + ex.Message, "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

private int GetDefaultProductTypeId()

{

try

{

var dt = Db.Table("SELECT TOP 1 ProductTypeID FROM dbo.ProductTypes ORDER BY ProductTypeID");

if (dt.Rows.Count > 0) return Convert.ToInt32(dt.Rows[0][0]);

}

catch { /\* таблицы может не быть — это ок \*/ }

return 1;

}

private void SaveProduct()

{

try

{

if (string.IsNullOrWhiteSpace(tbName.Text))

{

MessageBox.Show("Заполните название товара.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (numPrice.Value <= 0)

{

MessageBox.Show("Цена должна быть больше 0.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

var ps = new List<SqlParameter>

{

new SqlParameter("@n", tbName.Text.Trim()),

new SqlParameter("@a", (object)tbArticle.Text.Trim() ?? DBNull.Value),

new SqlParameter("@sid", cmbSupplier.SelectedValue == null ? (object)DBNull.Value : (object)(int)cmbSupplier.SelectedValue),

new SqlParameter("@p", numPrice.Value),

new SqlParameter("@cost", numPrice.Value), // CostPrice = MinPartnerPrice

new SqlParameter("@prodTime", 0), // ProductionTimeHours = 0

new SqlParameter("@workshop", 1), // WorkshopNumber = 1

new SqlParameter("@img", (object)\_currentImagePath ?? DBNull.Value)

};

if (\_productId.HasValue)

{

string sql = "UPDATE " + T\_PRODUCTS + " SET " +

"ProductName=@n," +

"Article=@a," +

"SupplierID=@sid," +

"MinPartnerPrice=@p," +

"CostPrice=@cost," +

"ProductionTimeHours=@prodTime," +

"WorkshopNumber=@workshop," +

"ImageUrl=@img " +

"WHERE ProductID=@id";

ps.Add(new SqlParameter("@id", \_productId.Value));

Db.Exec(sql, ps.ToArray());

}

else

{

int typeId = GetDefaultProductTypeId();

ps.Add(new SqlParameter("@pt", typeId));

string sql = "INSERT INTO " + T\_PRODUCTS + "(" +

"ProductTypeID,ProductName,Article,SupplierID," +

"MinPartnerPrice,CostPrice,ProductionTimeHours,WorkshopNumber,ImageUrl) " +

"OUTPUT INSERTED.ProductID " +

"VALUES(@pt,@n,@a,@sid,@p,@cost,@prodTime,@workshop,@img)";

var dtId = Db.Table(sql, ps.ToArray());

if (dtId.Rows.Count > 0)

tbId.Text = Convert.ToString(dtId.Rows[0]["ProductID"]);

}

DialogResult = DialogResult.OK;

Close();

}

catch (Exception ex)

{

MessageBox.Show("Ошибка сохранения: " + ex.Message, "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

**ImageHelper.cs:**

using System;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Drawing.Imaging;

using System.IO;

namespace NewTechApp.Utility

{

public static class ImageHelper

{

public static string ImagesDir

=> Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "Images");

public static string SaveResized(Image src, int w, int h, string oldPathToDeleteIfAny = null)

{

if (!Directory.Exists(ImagesDir)) Directory.CreateDirectory(ImagesDir);

using (var bmp = new Bitmap(w, h))

{

using (var g = Graphics.FromImage(bmp))

{

g.CompositingQuality = CompositingQuality.HighQuality;

g.SmoothingMode = SmoothingMode.HighQuality;

g.InterpolationMode = InterpolationMode.HighQualityBicubic;

g.Clear(Color.White);

g.DrawImage(src, new Rectangle(0, 0, w, h));

}

var file = Path.Combine(ImagesDir, Guid.NewGuid().ToString("N") + ".jpg");

bmp.Save(file, ImageFormat.Jpeg);

if (!string.IsNullOrEmpty(oldPathToDeleteIfAny) && File.Exists(oldPathToDeleteIfAny))

{

try { File.Delete(oldPathToDeleteIfAny); } catch { /\* ignore \*/ }

}

return file;

}

}

}

}

# Приложение В (справочное)

**Тестирование**

Тестовый пример #1: TC\_AUTH\_01

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_AUTH\_01 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Успешная авторизация пользователя с корректными учетными данными |
| **Краткое изложение теста** | Проверка процесса входа в систему с валидными логином и паролем |
| **Этапы теста** | 1. Открыть приложение  2. Ввести логин: "admin"  3. Ввести пароль: "admin123"  4. Нажать кнопку "Войти"  5. Проверить результат авторизации |
| **Тестовые данные** | Логин: admin, Пароль: admin123 |
| **Ожидаемый результат** | Успешный вход в систему, отображение главной формы с данными пользователя |
| **Фактический результат** | Успешная авторизация, отображается главная форма с ФИО пользователя и его ролью |
| **Статус** | Зачет |
| **Предварительное условие** | Пользователь "admin" существует в базе данных с паролем "admin123" |
| **Постусловие** | Пользователь авторизован в системе, доступен функционал согласно его роли |
| **Примечания/комментарии** | Проверка пройдена успешно, система корректно аутентифицирует пользователя |

Тестовый пример #2: TC\_AUTH\_02

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_AUTH\_02 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Неудачная авторизация с выводом CAPTCHA после первой попытки |
| **Краткое изложение теста** | Проверка появления CAPTCHA после первой неудачной попытки входа |
| **Этапы теста** | 1. Открыть приложение  2. Ввести неверный логин: "wronguser"  3. Ввести неверный пароль: "wrongpass"  4. Нажать кнопку "Войти"  5. Проверить отображение CAPTCHA |
| **Тестовые данные** | Логин: wronguser, Пароль: wrongpass |
| **Ожидаемый результат** | Сообщение об ошибке авторизации и отображение CAPTCHA с 4 символами |
| **Фактический результат** | Отображается сообщение "Неверный логин/пароль" и CAPTCHA с 4 символами |
| **Статус** | Зачет |
| **Предварительное условие** | Пользователь "wronguser" не существует в системе |
| **Постусловие** | Форма авторизации с активной CAPTCHA |
| **Примечания/комментарии** | CAPTCHA содержит графический шум и 4 символа (буквы и цифры) |

Тестовый пример #3: TC\_AUTH\_03

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_AUTH\_03 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Блокировка системы после множества неудачных попыток с CAPTCHA |
| **Краткое изложение теста** | Проверка эскалации блокировки системы при repeated неудачных попытках |
| **Этапы теста** | 1. Первая попытка: неверные данные → CAPTCHA  2. Вторая попытка: неверные данные + неверная CAPTCHA → таймер 3 мин  3. Третья попытка: неверные данные → полная блокировка |
| **Тестовые данные** | Множественные неверные комбинации логин/пароль/CAPTCHA |
| **Ожидаемый результат** | Прогрессивная блокировка: CAPTCHA → 3мин таймер → полная блокировка |
| **Фактический результат** | Система корректно эскалирует блокировку согласно требованиям |
| **Статус** | Зачет |
| **Предварительное условие** | Чистая сессия без предыдущих попыток входа |
| **Постусловие** | Система полностью заблокирована, требуется перезапуск приложения |
| **Примечания/комментарии** | Тестирование системы безопасности прошло успешно |

Тестовый пример #4: TC\_PRODUCT\_01

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_PRODUCT\_01 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Добавление нового товара с валидными данными |
| **Краткое изложение теста** | Проверка функциональности добавления товара администратором |
| **Этапы теста** | 1. Авторизоваться как admin  2. Нажать кнопку "Добавить"  3. Заполнить данные товара  4. Нажать "Сохранить"  5. Проверить наличие товара в списке |
| **Тестовые данные** | Название: "Тестовый товар", Цена: 1500.50, Артикул: "TEST-001" |
| **Ожидаемый результат** | Товар успешно добавлен в систему и отображается в общем списке |
| **Фактический результат** | Товар добавлен, отображается в списке с корректными данными |
| **Статус** | Зачет |
| **Предварительное условие** | Права администратора, доступ к форме добавления товара |
| **Постусловие** | Новый товар доступен в системе для всех пользователей |
| **Примечания/комментарии** | Цена корректно сохраняется с двумя десятичными знаками |

Тестовый пример #5: TC\_SEARCH\_01

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_SEARCH\_01 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Поиск товаров по различным атрибутам в реальном времени |
| **Краткое изложение теста** | Проверка поисковой функциональности с разными входными параметрами |
| **Этапы теста** | 1. Открыть список товаров  2. Ввести поисковый запрос  3. Изменить фильтр поставщика  4. Изменить объем для расчета скидки  5. Наблюдать результаты в реальном времени |
| **Тестовые данные** | Запрос: "Дуб", Поставщик: "Все", Объем: 50000 |
| **Ожидаемый результат** | Мгновенное обновление результатов с учетом всех параметров |
| **Фактический результат** | Поиск работает в реальном времени, скидка рассчитывается корректно |
| **Статус** | Зачет |
| **Предварительное условие** | Товары с атрибутом "Дуб" существуют в системе |
| **Постусловие** | Отфильтрованный список товаров с актуальными ценами и скидками |
| **Примечания/комментарии** | Все фильтры и поиск работают совместно без необходимости перезагрузки |

Код Unit-тестов:

using System;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using NewTechApp.Auth;

using NewTechApp.Data;

namespace NewTechAppTests

{

[TestClass]

public class BusinessLogicTests

{

/// <summary>

/// Тестирование хэширования пароля алгоритмом SHA256

/// Проверяет, что одинаковые пароли дают одинаковый хэш

/// и что хэш не совпадает с оригинальным паролем

/// </summary>

[TestMethod]

public void TestPasswordHashing\_Sha256()

{

// Arrange

string password = "test123";

// Act

byte[] hash = HashHelper.Sha256(password);

byte[] hash2 = HashHelper.Sha256(password);

// Assert

CollectionAssert.AreEqual(hash, hash2, "Хэши одного пароля должны совпадать");

Assert.AreNotEqual(password, Encoding.UTF8.GetString(hash), "Хэш не должен совпадать с оригинальным паролем");

}

/// <summary>

/// Тестирование расчета скидки в зависимости от объема продаж

/// Проверяет корректность расчета для всех граничных значений

/// </summary>

[TestMethod]

public void TestDiscountCalculation\_VolumeBased()

{

// Act & Assert

Assert.AreEqual(0.00m, DiscountCalculator.CalcDiscount(5000), "Скидка 0% для объема до 10000");

Assert.AreEqual(0.05m, DiscountCalculator.CalcDiscount(25000), "Скидка 5% для объема 10000-50000");

Assert.AreEqual(0.10m, DiscountCalculator.CalcDiscount(100000), "Скидка 10% для объема 50000-300000");

Assert.AreEqual(0.15m, DiscountCalculator.CalcDiscount(500000), "Скидка 15% для объема 300000-1000000");

Assert.AreEqual(0.20m, DiscountCalculator.CalcDiscount(2000000), "Скидка 20% для объема свыше 1000000");

}

/// <summary>

/// Тестирование генерации CAPTCHA

/// Проверяет длину и допустимые символы в CAPTCHA

/// </summary>

[TestMethod]

public void TestCaptchaGeneration\_LengthAndContent()

{

// Arrange

var captchaText = CaptchaGenerator.GenerateText(4);

// Assert

Assert.AreEqual(4, captchaText.Length, "CAPTCHA должна содержать 4 символа");

Assert.IsTrue(captchaText.All(c => "ABCDEFGHJKLMNPQRSTUVWXYZ23456789".Contains(c)),

"CAPTCHA должна содержать только разрешенные символы");

}

/// <summary>

/// Тестирование валидации цены товара

/// Проверяет, что система отвергает отрицательные значения цены

/// </summary>

[TestMethod]

public void TestProductValidation\_InvalidPrice()

{

// Act & Assert

Assert.IsFalse(ProductValidator.ValidatePrice(-100), "Валидация должна失败 при отрицательной цене");

Assert.IsTrue(ProductValidator.ValidatePrice(100), "Валидация должна пройти при положительной цене");

Assert.IsTrue(ProductValidator.ValidatePrice(0), "Валидация должна пройти при нулевой цене");

}

/// <summary>

/// Тестирование блокировки системы после множества неудачных попыток входа

/// Проверяет эскалацию блокировки системы

/// </summary>

[TestMethod]

public void TestSystemLockout\_Escalation()

{

// Arrange

var security = new SecurityManager();

// Act & Assert

// Первая неудачная попытка - должна появиться CAPTCHA

security.RecordFailedAttempt();

Assert.IsTrue(security.RequiresCaptcha, "После первой неудачи должна требоваться CAPTCHA");

Assert.IsFalse(security.IsTemporarilyLocked, "После первой неудачи не должно быть временной блокировки");

// Вторая неудачная попытка - должна начаться временная блокировка

security.RecordFailedAttempt();

Assert.IsTrue(security.IsTemporarilyLocked, "После второй неудачи должна начаться временная блокировка");

// Третья неудачная попытка - полная блокировка

security.RecordFailedAttempt();

Assert.IsTrue(security.IsPermanentlyLocked, "После третьей неудачи должна начаться полная блокировка");

}

/// <summary>

/// Тестирование расчета итоговой цены со скидкой

/// Проверяет корректность применения скидки к базовой цене

/// </summary>

[TestMethod]

public void TestFinalPriceCalculation\_WithDiscount()

{

// Arrange

decimal basePrice = 1000m;

decimal discount = 0.10m; // 10%

// Act

decimal finalPrice = PriceCalculator.CalculateFinalPrice(basePrice, discount);

// Assert

Assert.AreEqual(900m, finalPrice, "Итоговая цена с 10% скидкой должна быть 900");

Assert.IsTrue(finalPrice < basePrice, "Итоговая цена должна быть меньше базовой");

}

}

// === ВСПОМОГАТЕЛЬНЫЕ КЛАССЫ ДЛЯ ТЕСТИРОВАНИЯ ===

public static class HashHelper

{

public static byte[] Sha256(string s)

{

using (var sha = SHA256.Create())

return sha.ComputeHash(Encoding.UTF8.GetBytes(s));

}

}

public static class DiscountCalculator

{

public static decimal CalcDiscount(decimal vol)

{

if (vol <= 10000) return 0m;

if (vol <= 50000) return 0.05m;

if (vol <= 300000) return 0.10m;

if (vol <= 1000000) return 0.15m;

return 0.20m;

}

}

public static class CaptchaGenerator

{

private static readonly Random R = new Random();

public static string GenerateText(int len)

{

const string chars = "ABCDEFGHJKLMNPQRSTUVWXYZ23456789";

var s = new char[len];

for (int i = 0; i < len; i++) s[i] = chars[R.Next(chars.Length)];

return new string(s);

}

}

public static class ProductValidator

{

public static bool ValidatePrice(decimal price)

{

return price >= 0;

}

}

public class SecurityManager

{

private int \_failedAttempts = 0;

public bool RequiresCaptcha => \_failedAttempts >= 1;

public bool IsTemporarilyLocked => \_failedAttempts >= 2;

public bool IsPermanentlyLocked => \_failedAttempts >= 3;

public void RecordFailedAttempt()

{

\_failedAttempts++;

}

public void Reset()

{

\_failedAttempts = 0;

}

}

public static class PriceCalculator

{

public static decimal CalculateFinalPrice(decimal basePrice, decimal discount)

{

return Math.Round(basePrice \* (1 - discount), 2);

}

}

}

**АТТЕСТАЦИОННЫЙ ЛИСТ**

**ПО УЧЕБНОЙ ПРАКТИКЕ (ПО ПРОФИЛЮ СПЕЦИАЛЬНОСТИ)**

По профессиональному модулю ПМ 02 «Осуществление интеграции программных модулей» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(код и наименование)

Специальность \_\_09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 22290907/1098 группы

Буранчиков Данила Романович

(Фамилия, имя, отчество)

Место прохождения практики: СПбПУ ИСПО, Приморский пр. 63\_\_\_\_\_\_\_\_\_\_\_

(наименование и адрес организации)

Период прохождения практики

с «02» сентября 2025 г. по «13» сентября 2025 г.

Виды и качество выполнения работ

|  |  |  |
| --- | --- | --- |
| Виды выполненных работ обучающимся  во время практики | Объем работ, час. | Качество выполнения работ (оценка по пятибалльной системе) |
| Участие в выработке требований к программному обеспечению. | 9 |  |
| Стадии проектирования программного обеспечения. Проектирование UML-диаграмм и алгоритмов. | 20 |  |
| Конструирование пользовательского интерфейса. Разработка модулей программного обеспечения. | 22 |  |
| Тестирование и отладка программных модулей. Модульное тестирование. | 12 |  |
| 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 Шаврова Л.С. \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.) (подпись)