**Государственное бюджетное профессиональное образовательное учреждение**

**«Нижегородский Губернский колледж»**

Методическая комиссия «Информатика и вычислительная техника»

Допущен к защите:

преподаватель

\_\_\_\_\_\_\_\_\_\_\_\_ Л.В. Мухина,

\_\_\_\_\_\_\_\_\_\_\_\_ Е.П. Голубева,

«21»\_\_марта\_\_2025 г.

**ОТЧЕТ** **ПО**

**УЧЕБНОЙ ПРАКТИКЕ**

ПМ.01 РАЗРАБОТКА МОДУЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ КОМПЬЮТЕРНЫХ СИСТЕМ

Руководитель: \_\_\_\_\_\_\_\_\_\_ Мухина Л.В., Голубева Е.П. 21.03.2025 г.

Студент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Гончарова Н. Е. 21.03.2025 г.

Специальность, группа: 09.02.07, 43П

Нижний Новгород

2025 г.

**Содержание**

[**ВВЕДЕНИЕ** 3](#_Toc192236257)

[**РАЗРАБОТКА НАСТОЛЬНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ** 5](#_Toc192236258)

[**1.1.** **Разработка, администрирование и защита баз данных** 5](#_Toc192236259)

[**1.2.** **Разработка модулей программного обеспечения для компьютерных систем** 9](#_Toc192236260)

[**1.3.** **Сопровождение и обслуживание программного обеспечения компьютерных** 14](#_Toc192236261)

[**1.4.** **Осуществление интеграции программных модулей** 17](#_Toc192236262)

[**1.5.** **Ручное тестирование** 19](#_Toc192236263)

[**1.6.** **Проектирование UML-диаграмм** 23](#_Toc192236264)

[**ЗАКЛЮЧЕНИЕ** 25](#_Toc192236265)

[**ПРИЛОЖЕНИЯ** 26](#_Toc192236266)

[**Приложение А** 26](#_Toc192236267)

# **ВВЕДЕНИЕ**

Учебная практика по ПМ.01 «Разработка модулей программного обеспечения для компьютерных систем» проходила в ГБПОУ «Нижегородский Губернский колледж» в период с 01.03.2025 по 21.03.2025 г.

В современном мире информационные технологии играют ключевую роль во всех сферах жизни: от бизнеса и промышленности до образования и медицины. Разработка программного обеспечения (ПО) для компьютерных систем является неотъемлемой частью цифровизации и автоматизации процессов. Учебная практика по данной дисциплине актуальна по следующим причинам:

1. Высокий спрос на разработчиков ПО. Современные компании нуждаются в квалифицированных специалистах, способных разрабатывать и поддерживать программные решения для различных компьютерных систем. Обучение практическим навыкам разработки модулей ПО позволяет студентам подготовиться к реальной работе в IT-сфере.
2. Развитие модульного подхода в программировании. В настоящее время разработка ПО строится на принципах модульности, что упрощает создание, тестирование и поддержку программных продуктов. Владение методами модульного программирования является важным навыком для будущих специалистов.
3. Актуальность использования современных технологий. Практика охватывает изучение языков программирования, инструментов разработки, методов тестирования и отладки программного кода. Это позволяет студентам освоить востребованные технологии, используемые в реальных проектах.

Таким образом, учебная практика «Разработка модулей программного обеспечения для компьютерных систем» является актуальной и востребованной, так как способствует подготовке квалифицированных специалистов, способных разрабатывать современные программные продукты и решать актуальные задачи IT-индустрии.

Целью данной учебной практики является овладение практических навыков разработки, тестирования и отладки модулей программного обеспечения, а также освоение современных методов и инструментов программирования для создания эффективной и надёжной компьютерной системы.

Для достижения данной цели поставлены следующие задачи:

1. Изучить принципы модульного программирования и ознакомиться с архитектурой программных систем и принципами их построения;
2. Ознакомиться с принципами объектно-ориентированного программирования.
3. Разработать программные модули на языке программирования C# на кроссплатформенном XAML фреймворке для платформы .NET – Avalonia;
4. Использовать современные инструменты и среды разработки. Реализовать взаимодействия между модулями ПО;
5. Применить модульное и интеграционное тестирование. Использовать системы контроля версий (Git).

Практическим результатом прохождения учебной практики является отчёт по выполнению заданий с представленными в нём скриншотами, программными кодами и диаграммами.

# **РАЗРАБОТКА НАСТОЛЬНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

* 1. **Разработка, администрирование и защита баз данных (день 1)**

В рамках учебной практики была предоставлена предметная область, которая описывала магазин ООО «Пиши-стирай» занимающийся продажей канцелярских товаров.

Для данной предметной области была разработана база данных с использованием PostgreSQL — одной из самых надежных и мощных реляционных систем управления базами данных (СУБД). При проектировании базы данных была соблюдена третья нормальная форма, что позволило:

* Исключить избыточность данных.
* Минимизировать вероятность аномалий при добавлении, изменении и удалении записей.
* Обеспечить логичную структуру данных и удобство работы с ними.

Для предотвращения несогласованности данных в базе реализованы механизмы ссылочной целостности. Это означает, что:

* Все связи между таблицами построены с использованием внешних ключей (FOREIGN KEY), что предотвращает удаление или изменение записей, которые связаны с другими таблицами.
* Используются ограничения целостности (CONSTRAINTS), обеспечивающие корректность вводимых данных, например:
* Уникальные значения для идентификаторов (PRIMARY KEY).
* Ограничения на допустимые диапазоны значений.
* Контроль каскадного удаления или обновления данных.

Скрип со всеми созданными таблицами представлен в Приложении А в таблице А1. База данных включает в себя сущности, отражающие бизнес-процессы магазина «Пиши-стирай»:

1. Продукты (product)
2. Производители продуктов (productmanufacturer)
3. Производители (manufacturers)
4. Поставщики продуктов (productsupplier)
5. Поставщики (suppliers)
6. Единицы измерения (unitofmeasurement)
7. Категории (categories)
8. Заказанные продукты (orderproduct)
9. Заказы (orders)
10. Статусы (status)
11. Пункты выдачи (pickuppoint)
12. Пользователи (users)
13. Роли (role).

Весь список сущностей отображён в таблице №1.

Таблица №1 – Список сущностей

|  |  |  |
| --- | --- | --- |
| **№** | **Название** | **Назначение** |
| 1 | product | Содержит информацию о продуктах, их характеристиках, стоимости, наличии и статусе. |
| 2 | productmanufacturer | Связывает продукты с их производителями. |
| 3 | manufacturers | Хранит информацию о производителях продукции. |
| 4 | productsupplier | Связывает продукты с их поставщиками. |
| 5 | suppliers | Содержит информацию о поставщиках. |
| 6 | unitofmeasurement | Хранит данные об единицах измерения продукции. |
| 7 | categories | Содержит информацию о категориях товаров. |
| 8 | orderproduct | Связывает заказы с продуктами, указывая количество каждого товара в заказе. |
| 9 | orders | Хранит информацию о заказах, их статусе, дате, пункте выдачи и коде получения. |
| 10 | status | Содержит статусы заказов. |
| 11 | pickuppoint | Хранит данные о пунктах выдачи заказов. |
| 12 | users | Содержит информацию о пользователях системы, включая их роли. |
| 13 | role | Описывает роли пользователей в системе. |

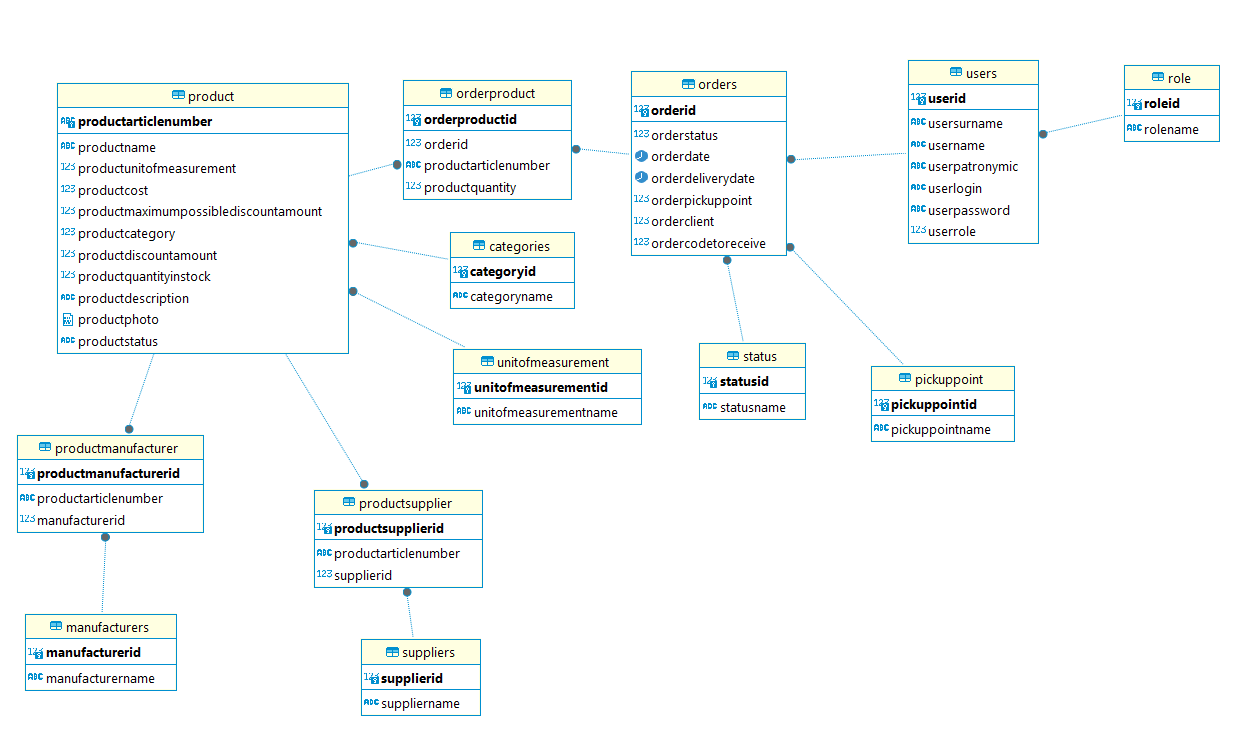
ER-диаграмма — это способ графического представления информационной модели, который позволяет показать все таблицы, их атрибуты и связи между ними. В нашем случае ER-диаграмма представлена на рисунке №1.

Рисунок №1 – ER-диаграмма БД «Пиши-стирай»

В таблице №2 описаны связи между таблицами в базе данных.

Таблица №2 – Список связей

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Название связи** | **Сущности, участвующие в связи** | **Назначение** |
| 1 | M:M | product – productsupplier – suppliers | Один продукт может иметь нескольких поставщиков, и один поставщик может поставлять несколько продуктов. |
| 2 | 1:M | manufacturers – productmanufacturer – product | Один производитель может выпускать несколько продуктов. |
| 3 | 1:M | categories – product | Одна категория может включать несколько продуктов. |
| 4 | 1:M | unitofmeasurement – product | Одна единица измерения может относиться к нескольким продуктам. |
| 5 | 1:M | orders – orderproduct – product | Один заказ может включать несколько продуктов. |
| 6 | 1:M | users – orders | Один пользователь может сделать несколько заказов. |
| 7 | 1:M | role – users | Одна роль может быть присвоена нескольким пользователям. |
| 8 | 1:M | pickuppoint – orders | Один пункт выдачи может обслуживать несколько заказов. |
| 9 | 1:M | status – orders | Один статус может быть назначен нескольким заказам. |

* 1. **Разработка модулей программного обеспечения для компьютерных систем (день 2)**

Разработка приложения для магазина «Пиши-стирай» реализовалась на принципах паттерна MVVM.

MVVM (Model–View–ViewModel) – это архитектурный паттерн, используемый в разработке приложений с разделением логики, представления и данных. Он помогает сделать код более структурированным, удобным для сопровождения и тестирования.

Основные компоненты паттерна MVVM:

1. Model (Модель)

* Отвечает за данные и бизнес-логику приложения.
* Может взаимодействовать с базой данных, веб-API или другими источниками данных.
* Не зависит от пользовательского интерфейса.

1. View (Представление)

* Отображает информацию пользователю.
* Представляет собой визуальный интерфейс приложения (окна, кнопки, списки и т. д.).
* Не содержит логики обработки данных, а только подписывается на изменения в ViewModel.

1. ViewModel (Модель представления)

* Посредник между Model и View.
* Содержит свойства и команды, к которым привязан интерфейс.
* Реализует механизм двустороннего связывания данных (Data Binding), позволяя View автоматически обновляться при изменении данных в ViewModel.

Работа MVVM заключается в следующем:

1. ViewModel получает данные из Model, обрабатывает их и предоставляет в удобном виде для View.
2. View подписывается на изменения в ViewModel и автоматически обновляется при изменении данных.
3. Пользователь взаимодействует с View, вызывая команды ViewModel, которые изменяют Model.
4. Изменения в Model передаются обратно в ViewModel, а затем обновляют View.

Преимущества MVVM:

* Разделение логики и интерфейса – код становится более чистым и структурированным.
* Упрощенное тестирование – можно тестировать Model и ViewModel отдельно от интерфейса.
* Гибкость и повторное использование – ViewModel можно применять с разными представлениями.
* Удобство поддержки и расширения – проще вносить изменения в код.

Паттерн MVVM особенно полезен при разработке UI-приложений, так как позволяет четко разделить ответственность между логикой обработки данных и отображением интерфейса.

Для разработки интерфейсов страниц приложения с авторизацией, выводом списка товаров, просмотром заказов, их редактированием и формированием, а также добавлением, редактированием и удалением товаров для магазина «Пиши-стирай» был выбран кроссплатформенный XAML фреймворк для платформы .NET – Avalonia. Avalonia позволяет создавать интерактивные и интуитивно понятные пользовательские интерфейсы, которые работают на различных операционных системах, включая Windows, macOS и Linux.

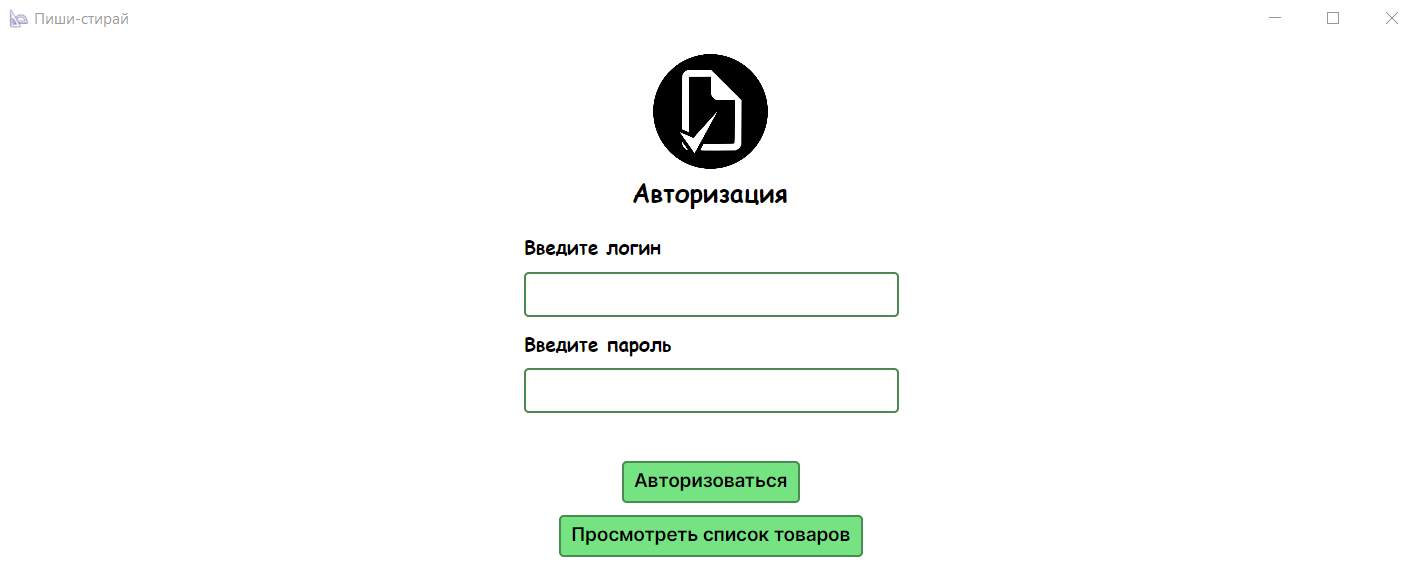
Использование Avalonia обеспечивает высокую производительность приложения и возможность удобного масштабирования интерфейсов для различных устройств и разрешений экранов.

* + 1. **Страница авторизации**

При разработке интерфейса страницы с авторизацией приложения были в основном использованы следующие элементы:

1. Логотип (Image) — отображает логотип приложения, находится в верхней части страницы.
2. Заголовок "Авторизация" (TextBlock) — текстовый блок с заголовком страницы, указывает пользователю на необходимость авторизации.
3. Текст "Введите логин" (TextBlock) — пояснительный текст для поля ввода логина.
4. Поле для ввода логина (TextBox) — текстовое поле, куда пользователь вводит логин, может быть заблокировано.
5. Текст "Введите пароль" (TextBlock) — пояснительный текст для поля ввода пароля.
6. Поле для ввода пароля (TextBox) — текстовое поле для ввода пароля, может быть заблокировано.
7. Капча (UserControl) — компонент для отображения капчи, помогает защититься от автоматической авторизации.
8. Поле для ввода кода с капчи (TextBox) — появляется при необходимости, используется для ввода кода с капчи.
9. Кнопка "Проверить" (Button) — кнопка для проверки введённого кода капчи, становится видимой при необходимости.
10. Кнопка "Авторизоваться" (Button) — основная кнопка для завершения процесса авторизации.
11. Кнопка "Просмотреть список товаров" (Button) — кнопка для перехода на страницу с перечнем товаров, доступна без авторизации.

В качестве результата разработки интерфейса страницы с авторизацией в системе для магазина «Пиши-стирай» ниже представлен рисунок №2.

Рисунок №2 – Интерфейс страницы с авторизацией

На данной странице реализуется процесс авторизации пользователей с дополнительной защитой с помощью капчи, а также поддерживает возможность просмотра страницы с продуктами в гостевом режиме.

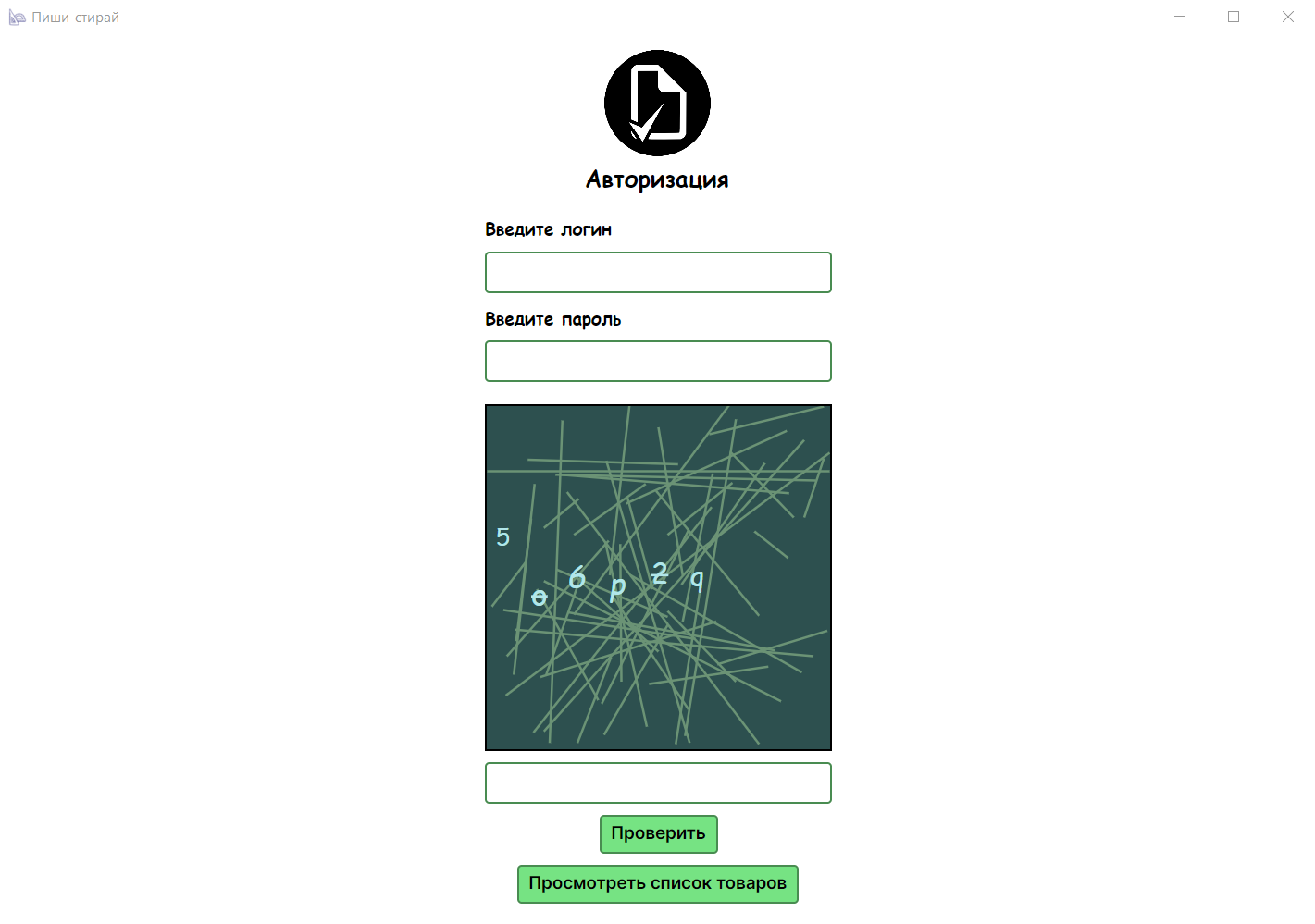
При вводе логина и пароля данные проверяются по списку пользователей. Если они корректны, происходит переход на страницу с продуктами. Если данные неверные, вызывается функция создания капчи — на экране появляется случайный набор символов и линий, усложняющих распознавание кода. Страница с отображением капчи при некорректной авторизации представлен на рисунке №3.

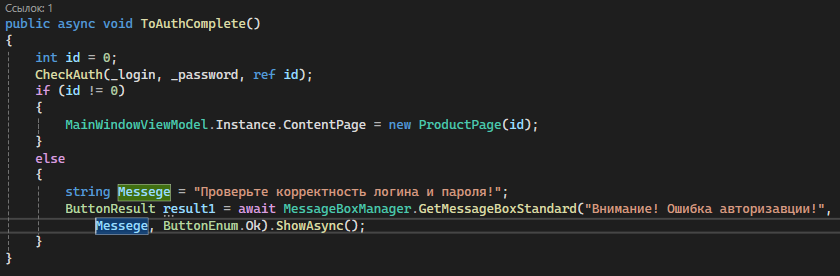
Рисунок №3 – Интерфейс страницы с авторизацией с капчей

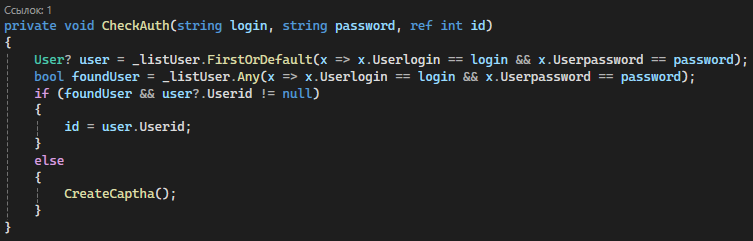
Пользователь должен ввести капчу вместе с логином и паролем. Если ввод верный, авторизация завершается успешно. При повторной ошибке включается таймер блокировки ввода на 10 секунд — поля для ввода отключаются, а после истечения времени появляется новая капча для следующей попытки.

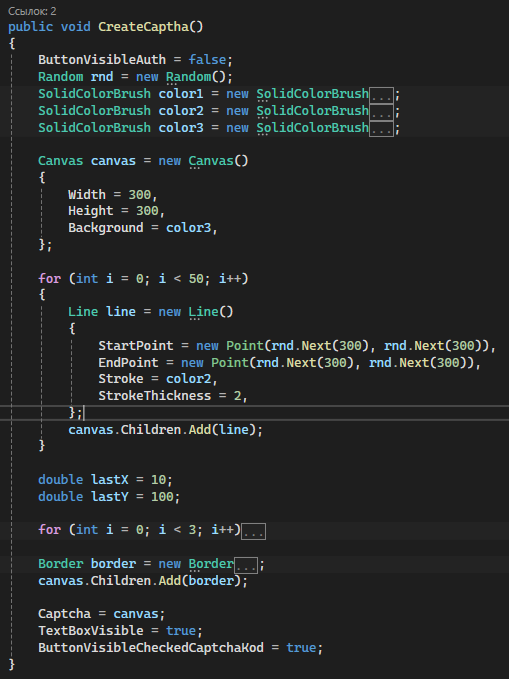
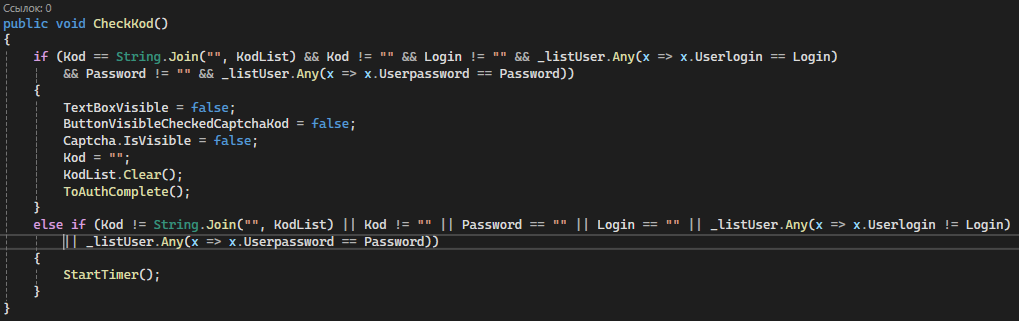
Помимо авторизации, предусмотрен гостевой режим: пользователь может открыть страницу с товарами без ввода данных, переходя на неё с фиксированным идентификатором (гость). Это позволяет просматривать товары, но с ограниченным функционалом по сравнению с авторизованными пользователями.

Таким образом, программа поддерживает два пути доступа к продуктам — через защищённую авторизацию и свободный гостевой просмотр.

**(День 3)** Ниже представлены рисунки с №4 по №8 с кодом реализации логики для данной страницы.

Рисунок №4 – Код реализации кнопки «Авторизоваться»

Рисунок №5 – Код реализации проведения авторизации

Рисунок №6 – Код реализации создания капчи

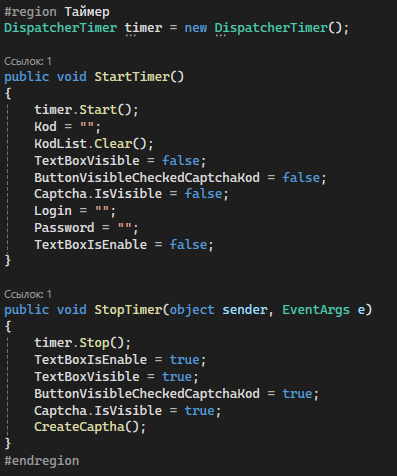
Рисунок №7 – Код реализации проверки кода капчи и повторной авторизации

Рисунок №8 – Код реализации таймера при неверном коде капчи

* 1. **Сопровождение и обслуживание программного обеспечения компьютерных**
  2. **Осуществление интеграции программных модулей**
  3. **Ручное тестирование**
  4. **Проектирование UML-диаграмм**

# **ЗАКЛЮЧЕНИЕ**

В ходе учебной практики была достигнута поставленная цель – овладение практическими навыками разработки, тестирования и отладки модулей программного обеспечения. В процессе выполнения работы были изучены принципы модульного программирования, архитектура программных систем, а также основные концепции объектно-ориентированного программирования.

В рамках практики разработаны программные модули на языке C# на кроссплатформенном XAML фреймворке для платформы .NET – Avalonia, что позволило закрепить полученные теоретические знания на практике. Также были использованы современные инструменты и среды разработки, что способствовало эффективному созданию и интеграции программных компонентов. Реализованы взаимодействия между модулями программного обеспечения, что позволило оценить важность гибкости и масштабируемости архитектурных решений.

Особое внимание было уделено тестированию программных модулей, включая модульное и интеграционное тестирование, что повысило надёжность разрабатываемого программного обеспечения. В процессе работы также применялись системы контроля версий (Git), что способствовало эффективному управлению разработкой и командной работе.

Таким образом, в результате прохождения учебной практики были приобретены ценные практические навыки, необходимые для дальнейшей профессиональной деятельности в области разработки программного обеспечения.

# **ПРИЛОЖЕНИЯ**

## **Приложение А**

Таблица №А1 – Скрип базы данных для компании «Пиши-стирай»

|  |
| --- |
| **create** **table** **Role**  (  RoleID serial **primary** **key**,  RoleName **varchar**(100) **not** **null**  );  **create** **table** Users  (  UserID serial **primary** **key**,  UserSurname **varchar**(100) **not** **null**,  UserName **varchar**(100) **not** **null**,  UserPatronymic **varchar**(100) **not** **null**,  UserLogin **varchar**(100) **not** **null**,  UserPassword **varchar** **not** **null**,  UserRole **int** **not** **null**,  **constraint** UserRole\_fk **foreign** **key** (UserRole) **references** **Role**(RoleID),  **constraint** unique\_user\_login **unique** (UserLogin)  );  **create** **table** Status  (  StatusID serial **primary** **key**,  StatusName **varchar**(100) **not** **null**  );  **create** **table** PickupPoint  (  PickupPointID serial **primary** **key**,  PickupPointName **varchar**(100) **not** **null**  );  **create** **table** Orders  (  OrderID serial **primary** **key**,  OrderStatus **int** **not** **null**,  OrderDate **date** **not** **null**,  OrderDeliveryDate **date**,  OrderPickupPoint **int**,  OrderClient **int** **not** **null** **DEFAULT** 1,  OrderCodeToReceive **int**,  **constraint** OrderStatus\_fk **foreign** **key** (OrderStatus) **references** Status(StatusID),  **constraint** OrderPickupPoint\_fk **foreign** **key** (OrderPickupPoint) **references** PickupPoint(PickupPointID),  **constraint** OrderClient\_fk **foreign** **key** (OrderClient) **references** Users(UserID)  );  **create** **table** Categories  (  CategoryID serial **primary** **key**,  CategoryName **varchar**(100) **not** **null**  );  **create** **table** UnitOfMeasurement  (  UnitOfMeasurementID serial **primary** **key**,  UnitOfMeasurementName **varchar**(100) **not** **null**  );  **create** **table** Product  (  ProductArticleNumber **varchar**(100) **primary** **key**,  ProductName **varchar** **not** **null**,  ProductUnitOfMeasurement **int** **not** **null**,  ProductCost **float4** **not** **null**,  ProductMaximumPossibleDiscountAmount **smallint** **null**,  ProductCategory **int** **not** **null**,  ProductDiscountAmount **float4** **null**,  ProductQuantityInStock **int** **not** **null**,  ProductDescription **varchar** **not** **null**,  ProductPhoto **varchar** **not** **null**,  ProductStatus **varchar** **null**,  **constraint** ProductUnitOfMeasurement\_fk **foreign** **key** (ProductUnitOfMeasurement) **references** UnitOfMeasurement(UnitOfMeasurementID),  **constraint** ProductCategory\_fk **foreign** **key** (ProductCategory) **references** Categories(CategoryID)  );  **create** **table** OrderProduct  (  OrderProductID serial **primary** **key**,  OrderID **int** **not** **null**,  ProductArticleNumber **varchar**(100) **not** **null**,  ProductQuantity **int** **not** **null**,  **constraint** Order\_OrderProduct\_fk **foreign** **key** (OrderID) **references** Orders(OrderID),  **constraint** Product\_OrderProduct\_fk **foreign** **key** (ProductArticleNumber) **references** Product(ProductArticleNumber)  );  **create** **table** Manufacturers  (  ManufacturerID serial **primary** **key**,  ManufacturerName **varchar**(100) **not** **null**  );  **create** **table** ProductManufacturer  (  ProductManufacturerID serial **primary** **key**,  ProductArticleNumber **varchar**(100) **not** **null**,  ManufacturerID **int** **not** **null**,  **constraint** Product\_ProductManufacturer\_fk **foreign** **key** (ProductArticleNumber) **references** Product(ProductArticleNumber),  **constraint** Manufacturer\_ProductManufacturer\_fk **foreign** **key** (ManufacturerID) **references** Manufacturers(ManufacturerID)  );  **create** **table** Suppliers  (  SupplierID serial **primary** **key**,  SupplierName **varchar**(100) **not** **null**  );  **create** **table** ProductSupplier  (  ProductSupplierID serial **primary** **key**,  ProductArticleNumber **varchar**(100) **not** **null**,  SupplierID **int** **not** **null**,  **constraint** Product\_ProductSupplier\_fk **foreign** **key** (ProductArticleNumber) **references** Product(ProductArticleNumber),  **constraint** Supplier\_ProductSupplier\_fk **foreign** **key** (SupplierID) **references** Suppliers(SupplierID)  ); |

Таблица №А2 – Код DLL-библиотеки для расчёта общего количества материала в продукте.

|  |
| --- |
|  |