[**1. Введение 6**](#_gjdgxs)

[**2. Основная часть 8**](#_3u2o9y9k9lzi)

[**2.1. Анализ современного состояния проблемы 8**](#_dq5umtnv2yg0)

[**2.2. Цели и задачи проекта 8**](#_6mf0ezvhe7kp)

[**2.3. Изучение текущих методов и систем управления складом 9**](#_48zwr324r4ge)

[**2.4. Определение ключевых требований к системе 10**](#_diuqm975pbp7)

[**2.5. Разработка телеграмм-бота 11**](#_8gnewfr9z6af)

[**2.5.1. Функциональные возможности телеграмм-бота 11**](#_x5n280n8f9du)

[**2.5.2. Техническое описание логики работы бота 14**](#_kdmvo3sbov5c)

[**2.6. Разработка десктоп-приложения для пользователей на C# с использованием WindowsForms 17**](#_667pl6jspmaa)

[**2.6.1. Функциональные возможности десктоп-приложения 17**](#_z6cvhiz768gb)

[**2.6.2. Техническое описание логики работы десктоп-приложения 25**](#_d7xho37hgph9)

[**2.7. Разработка десктоп-приложения для складских компьютеров на C# с использованием WindowsForms 30**](#_985nv8avc5vd)

[**2.7.1. Функциональные возможности десктоп-приложения 30**](#_xaf95ubm1mbs)

[**2.7.2. Техническое описание логики работы десктоп-приложения 32**](#_8e35lja8tlqb)

[**3. Заключение 35**](#_j04axex8d19l)

[**4. Список литературы 37**](#_j59x3ryf82pv)

# **Введение**

Современные предприятия сталкиваются с постоянными вызовами, связанными с эффективным управлением складской логистики. В условиях роста объемов товарооборота, ускоренного ритма торговли и высоких требований к скорости и точности обработки данных, традиционные методы управления складскими запасами становятся недостаточно эффективными. Основные проблемы включают в себя трудоемкость ручной регистрации товаров, высокую вероятность ошибок при учете и недостаточную прозрачность процессов. Существующие системы автоматизации складских операций зачастую имеют ограничения, связанные с их сложностью и высокими затратами на внедрение. В связи с этим, растет потребность в доступных и эффективных решениях, которые могли бы интегрироваться в существующие бизнес-процессы и обеспечивать высокую точность и оперативность обработки данных.

Разработка проекта основывается на необходимости повышения эффективности складских операций и улучшения логистических процессов. В качестве исходных данных были использованы современные методы автоматизации и технологии, такие как телеграмм-боты и десктоп-приложения, которые обеспечивают удобство использования и высокую степень интеграции.

Планируемый научно-технический уровень разработки включает создание комплексной системы управления складом и логистикой, интегрирующей телеграмм-бот для генерации и обработки QR-кодов и десктоп-приложения для регистрации и отслеживания товаров. Ожидается, что внедрение данной системы позволит существенно повысить точность учета товаров, сократить время на их обработку и улучшить контроль за движением товаров на складе.

Аналогичные системы существуют, однако большинство из них либо не обладают необходимой функциональностью, либо требуют значительных затрат на внедрение и обслуживание. Предлагаемая система отличается высокой степенью интеграции, простотой использования и доступностью, что обеспечивает ее конкурентное преимущество.

Актуальность проекта обусловлена необходимостью повышения эффективности управления складскими операциями и логистикой в условиях современного рынка. Новизна заключается в интеграции телеграмм-бота и десктоп-приложений, что обеспечивает удобство и точность учета товаров, а также возможность гибкого управления их размещением.

# **Основная часть**

# **Анализ современного состояния проблемы**

Актуальность проекта обусловлена необходимостью повышения эффективности управления складскими операциями и логистикой в условиях современного рынка. Новизна заключается в интеграции телеграмм-бота и десктоп-приложений, что обеспечивает удобство и точность учета товаров, а также возможность гибкого управления их размещением.

# **Цели и задачи проекта**

Основной целью проекта является создание эффективной и доступной автоматизированной системы, которая позволит улучшить процессы регистрации и учета товаров на складе.

Для достижения цели необходимо решить следующие задачи:

1. *Анализ потребностей и требований пользователей:*
   1. Изучение текущих методов и систем управления складом.
   2. Определение ключевых требований к новой системе на основе анализа потребностей пользователей и складских операций.
2. *Разработка телеграмм-бота на C#:*
   1. Создание телеграмм-бота, который позволяет пользователям вводить информацию о товарах и генерировать QR-коды.
   2. Обеспечение функциональности для ввода названия и описания товара.
   3. Генерация уникальных QR-кодов для каждого товара.
3. *Разработка десктоп-приложения для пользователей на C# с использованием WindowsForms:*
   1. Создание интерфейса для регистрации пользователей и их магазинов.
   2. Реализация функциональности для сканирования и регистрации QR-кодов товаров.
   3. Предоставление возможностей для выбора места хранения товаров (склад или домашнее хранение).
   4. Обеспечение удобного интерфейса для управления зарегистрированными товарами.
4. *Разработка десктоп-приложения для складских компьютеров на C# с использованием WindowsForms:*
   1. Создание интерфейса для сканирования QR-кодов на складе.
   2. Обеспечение функциональности для отметки нахождения товара на складе и его готовности к размещению на торговой площадке.
   3. Интеграция с сервером для обновления состояния товаров в системе.

# **Изучение текущих методов и систем управления складом**

Современные системы управления складом (WMS, Warehouse Management System) и логистикой активно используются крупными и мелкими предприятиями для оптимизации складских операций.

1. *Автоматизированные системы управления складом (WMS):*

Wildberries и Ozon: Эти компании используют сложные WMS, которые автоматизируют все складские операции от приемки товаров до их отгрузки. Такие системы позволяют эффективно управлять большими объемами данных, обеспечивать точный учет и быстрое перемещение товаров по складу.

Amazon: Компания использует собственные WMS, интегрированные с роботизированными системами, которые оптимизируют процесс хранения и доставки товаров. Автоматизация достигается за счет использования роботов, которые перемещают стеллажи с товарами к работникам, что значительно ускоряет процессы сборки и упаковки заказов.

1. *Интеграция с ERP-системами:*

Многие предприятия интегрируют свои WMS с ERP (Enterprise Resource Planning) системами, такими как SAP и Oracle, что позволяет улучшить управление ресурсами и повысить общую эффективность бизнеса. Эти системы обеспечивают централизованное управление всеми аспектами бизнеса, включая складские операции, финансы, производство и управление персоналом.

1. *Использование мобильных приложений и сканеров штрих-кодов:*

Компании используют мобильные приложения и портативные сканеры штрих-кодов для оперативного учета и отслеживания товаров на складе. Это позволяет сократить время на обработку данных и уменьшить вероятность ошибок при ручном вводе данных.

# **Определение ключевых требований к системе**

1. *Автоматизация процессов:*

Необходимость автоматизации процессов регистрации и учета товаров для снижения трудозатрат и уменьшения вероятности ошибок. Система должна поддерживать автоматическое сканирование и регистрацию товаров с использованием QR-кодов.

1. *Интеграция с телеграмм-ботом:*

Интеграция с телеграмм-ботом для упрощения ввода данных о товарах и генерации QR-кодов. Бот должен обеспечивать удобный интерфейс для пользователей, позволяя быстро добавлять новые товары в систему.

1. *Простота использования:*

Интерфейс системы должен быть интуитивно понятным и удобным для пользователей, независимо от их технического уровня. Важно обеспечить минимальное время обучения для новых пользователей и складских сотрудников.

1. *Реальное время и точность данных:*

Система должна обеспечивать точность и актуальность данных в реальном времени. Это особенно важно для управления запасами и принятия решений на основе текущих данных о состоянии товаров на складе.

1. *Поддержка различных типов хранения:*

Система должна поддерживать различные типы хранения товаров, включая хранение на складе и домашнее хранение. Пользователи должны иметь возможность выбирать и управлять местами хранения своих товаров.

1. *Поддержка многопользовательского режима:*

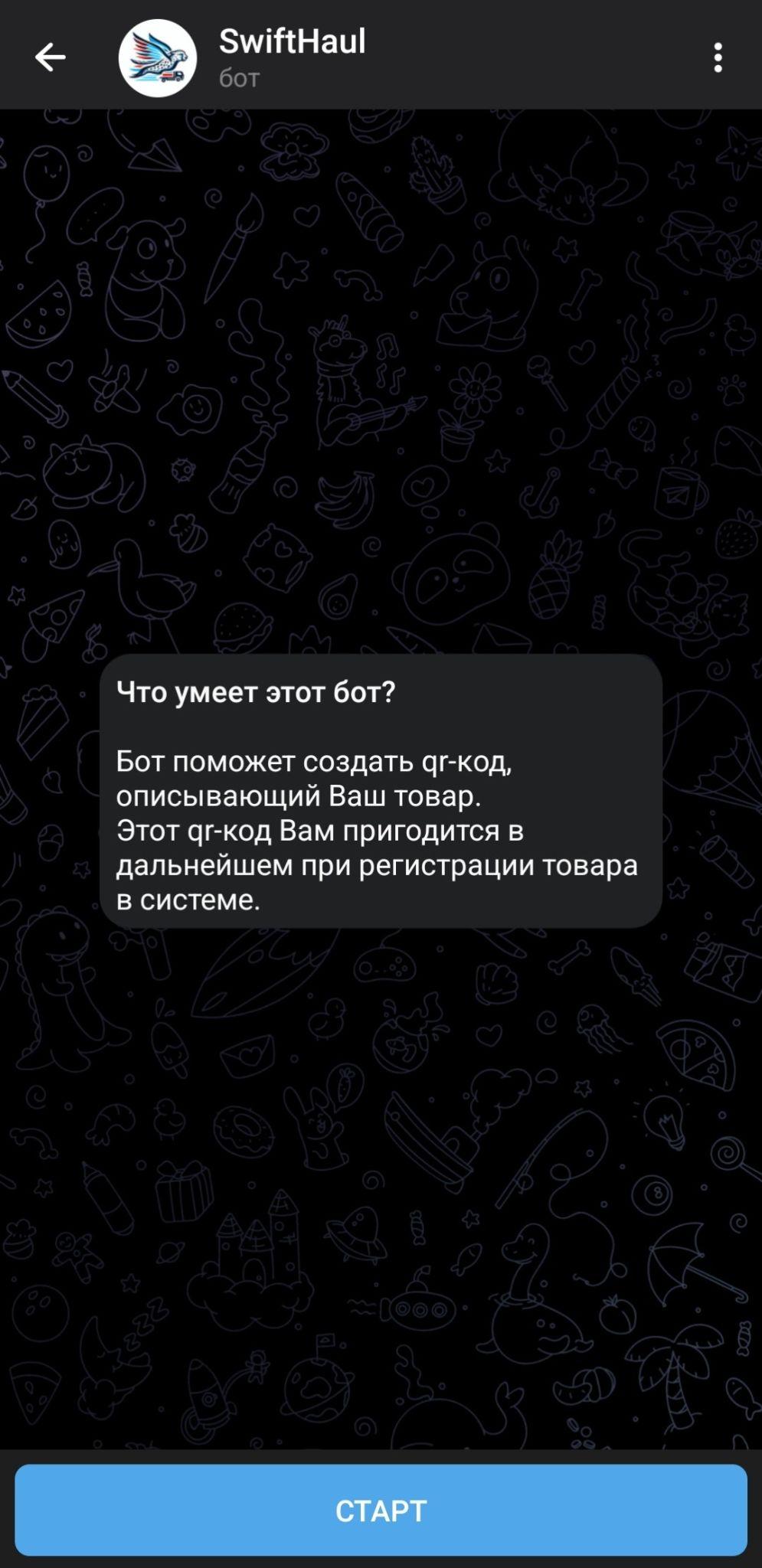
Система должна поддерживать работу нескольких пользователей одновременно, обеспечивая доступ к необходимым функциям и данным в зависимости от ролей и прав доступа.

# **Разработка телеграмм-бота**

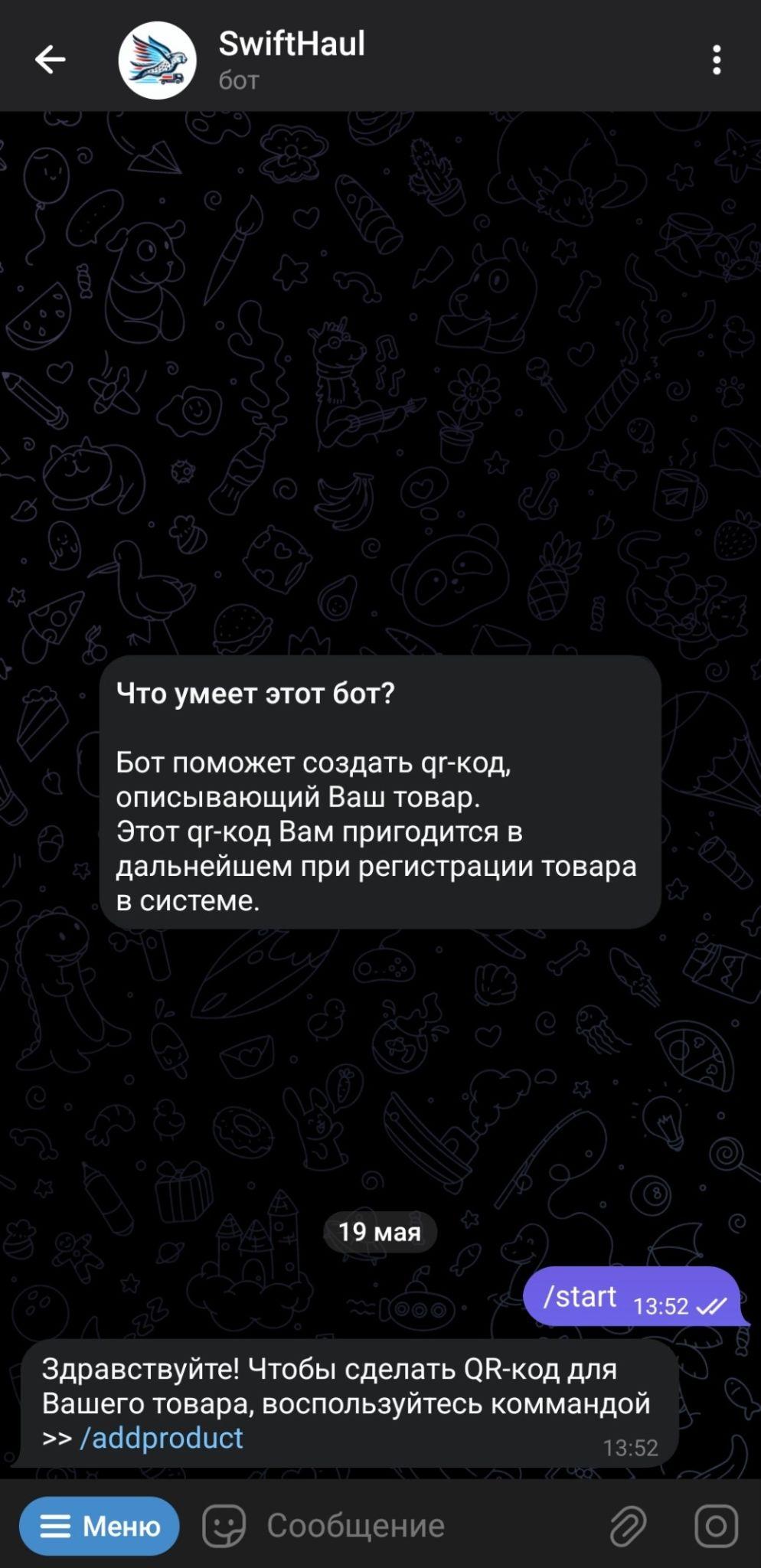
# **Функциональные возможности телеграмм-бота**

Телеграмм-бот, разработанный на языке программирования C#, выполняет ключевую роль в системе управления складом и логистикой. Основная функция бота заключается в обеспечении пользователей возможностью быстро и эффективно создавать и регистрировать товары посредством генерации QR-кодов. Описание функциональных возможностей бота можно разделить на несколько этапов, соответствующих последовательности действий пользователя:

1. Инициализация команды “/addproduct”. Бот предоставляет пользователю единственную команду “/addproduct”, которая инициирует процесс создания qr-кода, описывающего товар.

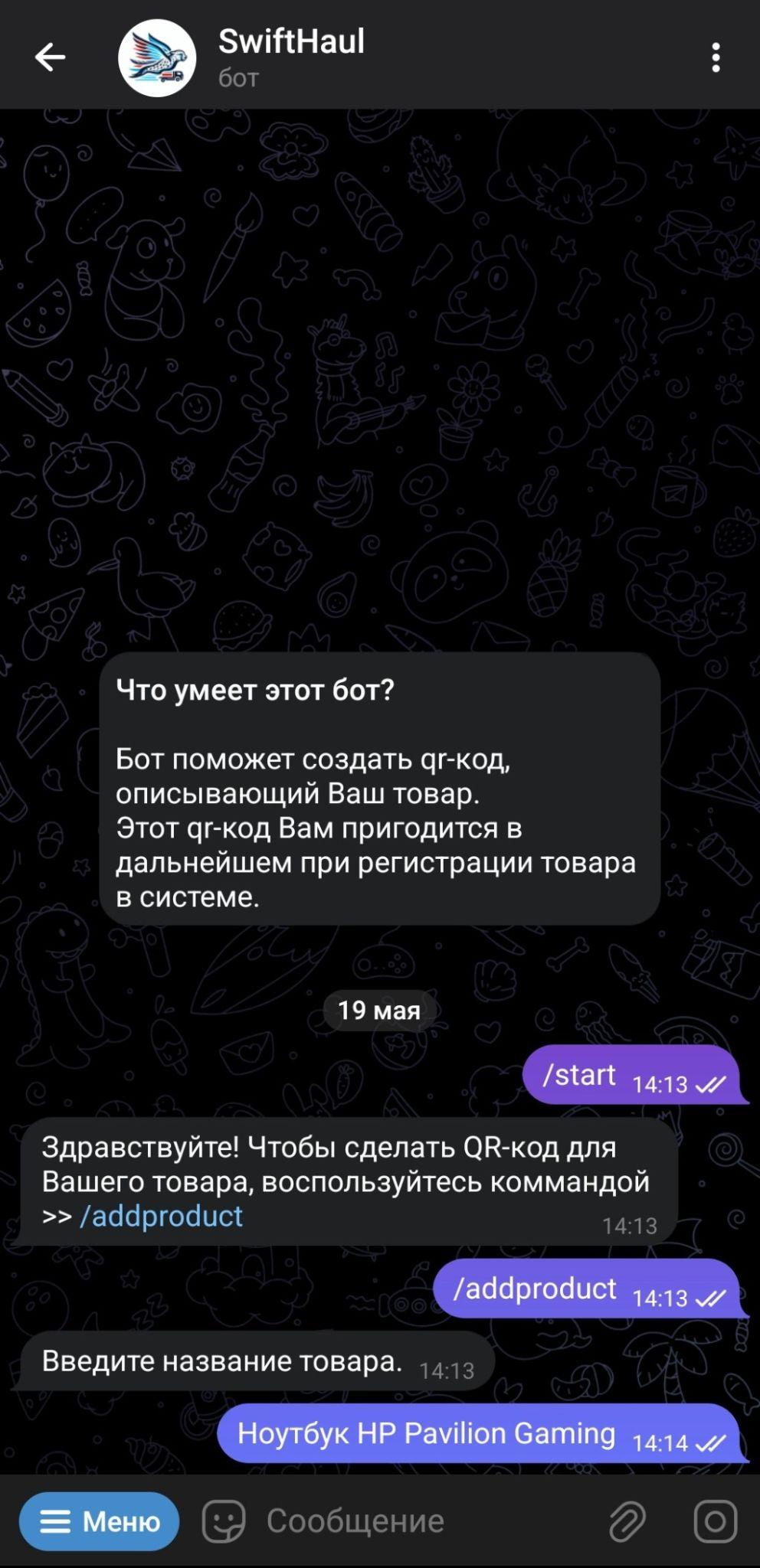


*Рисунок 1.* Демонстрация работы телеграмм бота (Этап 1)

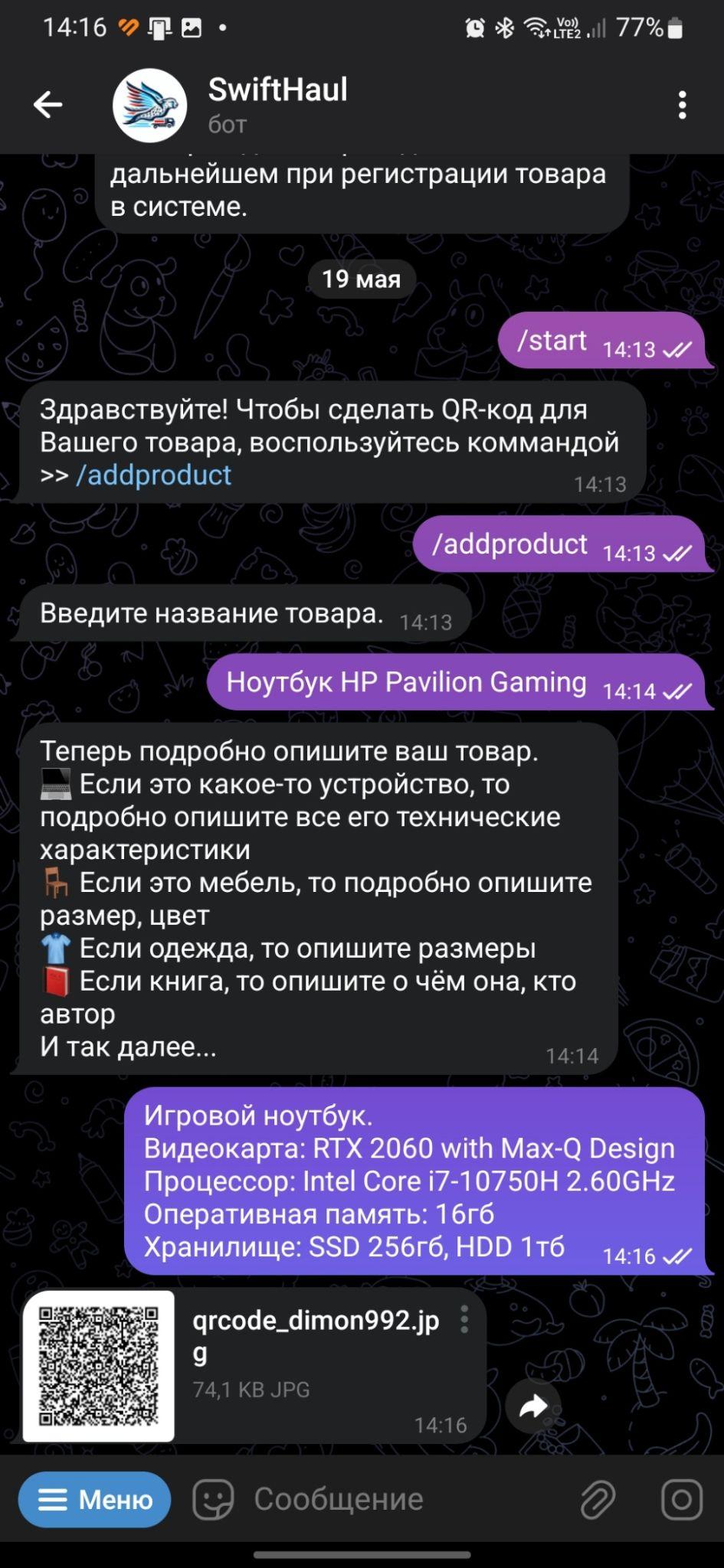


*Рисунок 2.* Демонстрация работы телеграмм бота (Этап 2)

1. Ввод названия товара. Бот отправляет пользователю запрос на ввод названия товара. Пользователь вводит название товара, после чего бот переходит к следующему этапу.
2. Ввод описания товара. После получения названия товара бот отправляет пользователю запрос на ввод его описания. Пользователь вводит описание товара, которое включает ключевые характеристики и особенности товара.
3. Генерация QR-кода. На основе введенных пользователем данных (название и описание товара) бот генерирует уникальный QR-код. QR-код включает всю необходимую информацию о товаре, закодированную в удобном для последующего сканирования формате.



*Рисунок 3.* Демонстрация работы телеграмм бота (Этап 3)



*Рисунок 4.* Демонстрация работы телеграмм бота (Этап 4)

# **Техническое описание логики работы бота**

1. *Общая логика работы бота:*

В основе логики работы бота лежит словарь, который следит за тем, на каком этапе регистрации товара находится пользователь, и функция HandleCommand, которая отслеживает вводимые команды.

Словарь:

**private** **static** Dictionary<**long**, (Activity activity, Product product)> workingUsers;

где **long -** это id пользователя,

Activity activity**-** это emun текущего состояния пользователя:

**enum** Activity

{

waitCommand,

enterProductName,

enterProductDescription,

endProductEntery

}

Product product**-** это объект класса Product, в котором есть поля productName и productDescription, которые заполняет пользователь.

Функция HandleCommand:

**public** **static** **async** Task **HandleCommand**(ITelegramBotClient botClient, Message message)

{

**string** command = message.Text.Split(' ')[**0**]; // Получаем первое слово из текста сообщения, это будет команда

**switch** (command)

{

**case** "/start":

**await** **HandleStartCommand**(botClient, message);

**break**;

**case** "/addproduct":

**await** **HandleAddProductCommand**(botClient, message);

**break**;

**default**:

**await** **HandleUnknownCommand**(botClient, message);

**break**;

}

}

Реализации функций обработчиков каждой из команд:

**private** **static** **async** Task **HandleStartCommand**(ITelegramBotClient botClient, Message message)

{

**await** botClient.SendTextMessageAsync(message.From.Id, "Здравствуйте! Чтобы сделать QR-код для Вашего товара, воспользуйтесь коммандой >> /addproduct");

}

**private** **static** **async** Task **HandleAddProductCommand**(ITelegramBotClient botClient, Message message)

{

**await** botClient.SendTextMessageAsync(message.Chat.Id, "Введите название товара.");

workingUsers.Add(message.From.Id, (Activity.enterProductName, **new** Product()));

}

**private** **static** **async** Task **HandleHelpCommand**(ITelegramBotClient botClient, Message message)

{

**await** botClient.SendTextMessageAsync(message.Chat.Id, "Помощь по работе с командами и ботом");

}

**private** **static** **async** Task **HandleUnknownCommand**(ITelegramBotClient botClient, Message message)

{

**await** botClient.SendTextMessageAsync(message.Chat.Id, "Неизвестная команда");

}

В главной функции Update, как только пользователь вводит какой-либо текст, мы просто проверяем на каком этапе он находится.

1. *Логика генерации QR-кода:*

Идея заключается в том, что как только пользователь ввёл название товара и его описание, то класс Product сериализуется в json формат, а потом json строка кодируется в QR-код.

Заполнение полей productName и productDescription:

**if** (workingUsers[message.From.Id].Item1 == Activity.enterProductName)

{

**var** tuple = workingUsers[message.From.Id];

Product tmpProduct = tuple.product;

tmpProduct.ProductName = message.Text;

workingUsers[message.From.Id] = (Activity.enterProductDescription, tmpProduct);

**await** botClient.SendTextMessageAsync(message.From.Id, "Теперь подробно опишите ваш товар.\n" +

"\U0001f4bb Если это какое-то устройство, то подробно опишите все его технические характеристики\n" +

"\U0001fa91 Если это мебель, то подробно опишите размер, цвет\n" +

"\U0001f455 Если одежда, то опишите размеры\n" +

"\U0001f4d5 Если книга, то опишите о чём она, кто автор\n" +

"И так далее...");

}

**else** **if** (workingUsers[message.From.Id].Item1 == Activity.enterProductDescription)

{

**var** tuple = workingUsers[message.From.Id];

Product tmpProduct = tuple.product;

tmpProduct.DateTime = DateTime.Now;

tmpProduct.ProductDescription = message.Text;

…

Сериализация класса tmpProduct с помощью пакета **using** **System.Text.Json**; :

…

JsonSerializerOptions options = **new** JsonSerializerOptions

{

Encoder = System.Text.Encodings.Web.JavaScriptEncoder.Create(UnicodeRanges.All)

};

**string** json = JsonSerializer.Serialize(tmpProduct, options);

…

Создание уникального QR-кода с помощью пакетов

**using** **ZXing**; **using** **ZXing.QrCode**; **using** **ZXing.Windows.Compatibility**; :

…

BarcodeWriter<Bitmap> qrCode = **new** BarcodeWriter<Bitmap>

{

Format = BarcodeFormat.QR\_CODE,

Options = **new** QrCodeEncodingOptions { Height = **500**, Width = **500**, CharacterSet = "UTF-8" },

Renderer = **new** BitmapRenderer()

};

qrCode.Write(json).Save(filePath, ImageFormat.Jpeg);

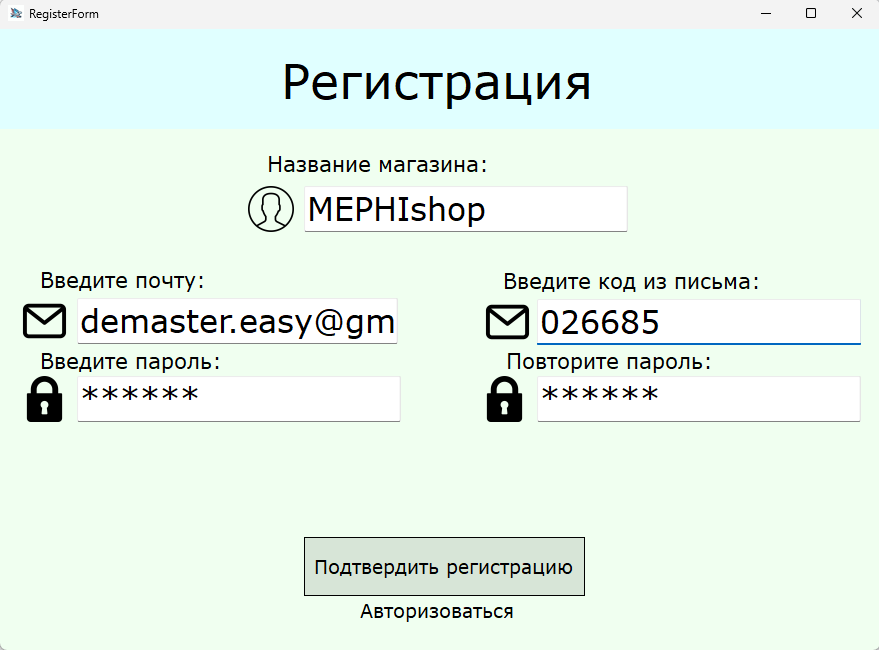
…

# **Разработка десктоп-приложения для пользователей на C# с использованием WindowsForms**

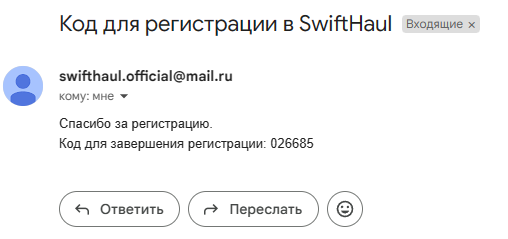
# **Функциональные возможности десктоп-приложения**

1. *Регистрация:*

В окне регистрации пользователю необходимо ввести название его магазина (логин), почту и пароль. Почту нужно вводить действующую, потому что регистрация завершается проверкой кода, полученного из письма, высланного на почту.



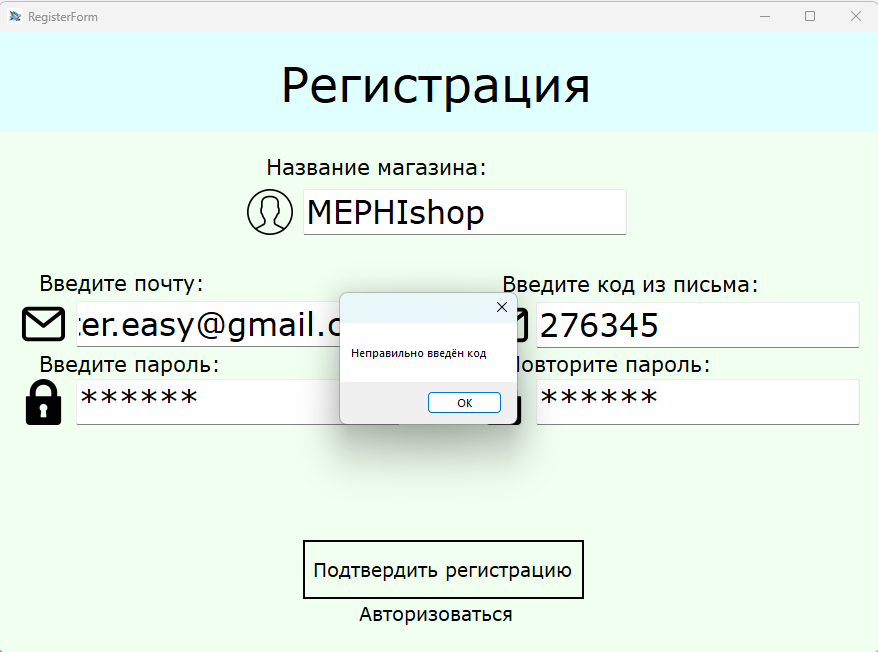
*Рисунок 5.* Окно регистрации



*Рисунок 6.* Проверочный код, высланный на почту

После успешной регистрации пользователя встречает главная форма.

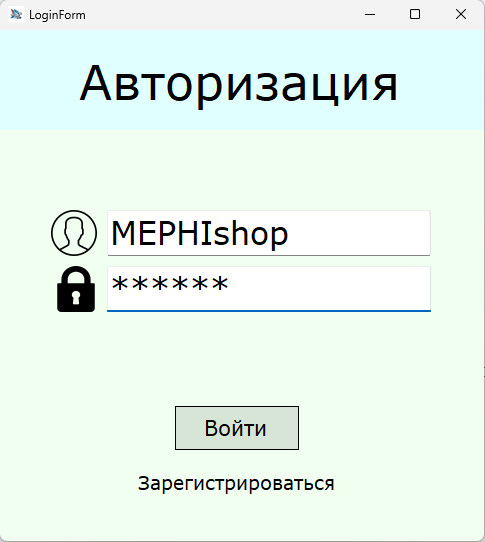
В случае некорректного ввода проверочного кода, выводится соответствующее сообщение:



*Рисунок 7*. Окно регистрации в случае некорректного ввода проверочного кода

1. *Вход:*

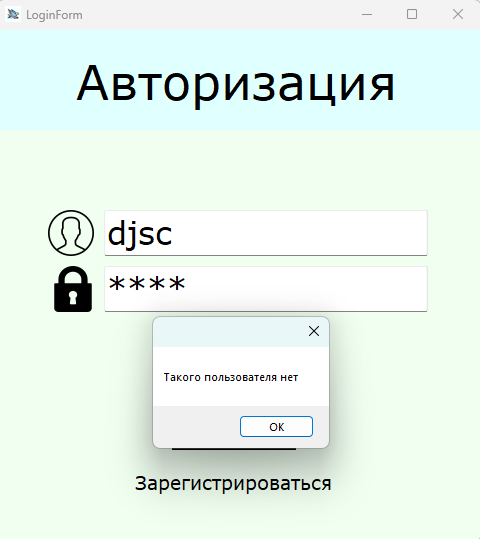
В окне входа пользователю необходимо ввести название его магазина (логин) и пароль, указанный при регистрации.



*Рисунок 8.* Окно авторизации

После успешного входа пользователя встречает главная форма.

В случае некорректного ввода логина или пароля выводится сообщение о том, что такого пользователя нет:

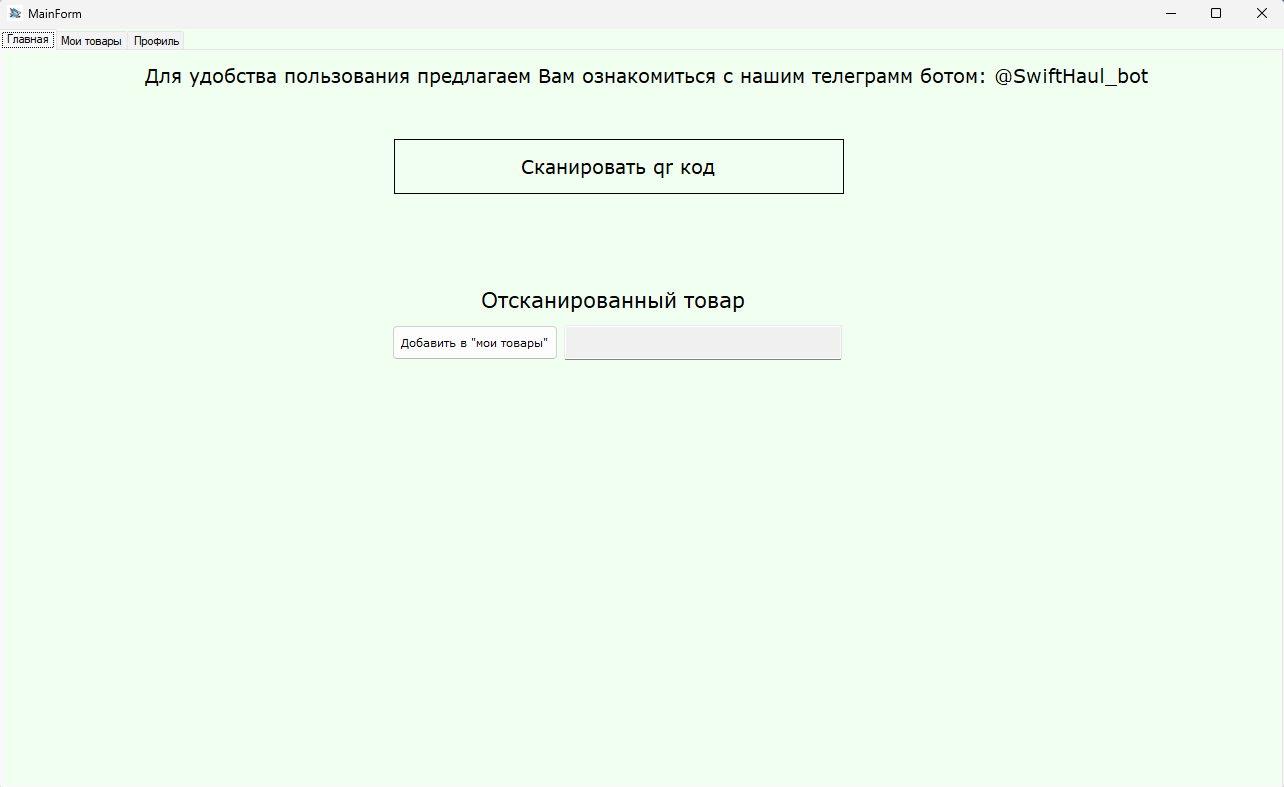


*Рисунок 9.* Окно авторизации в случае некорректного ввода логина или пароля

1. *Главная форма:*

На главной форме есть 3 вкладки: “Главная”, “Мои товары” и “Профиль”.

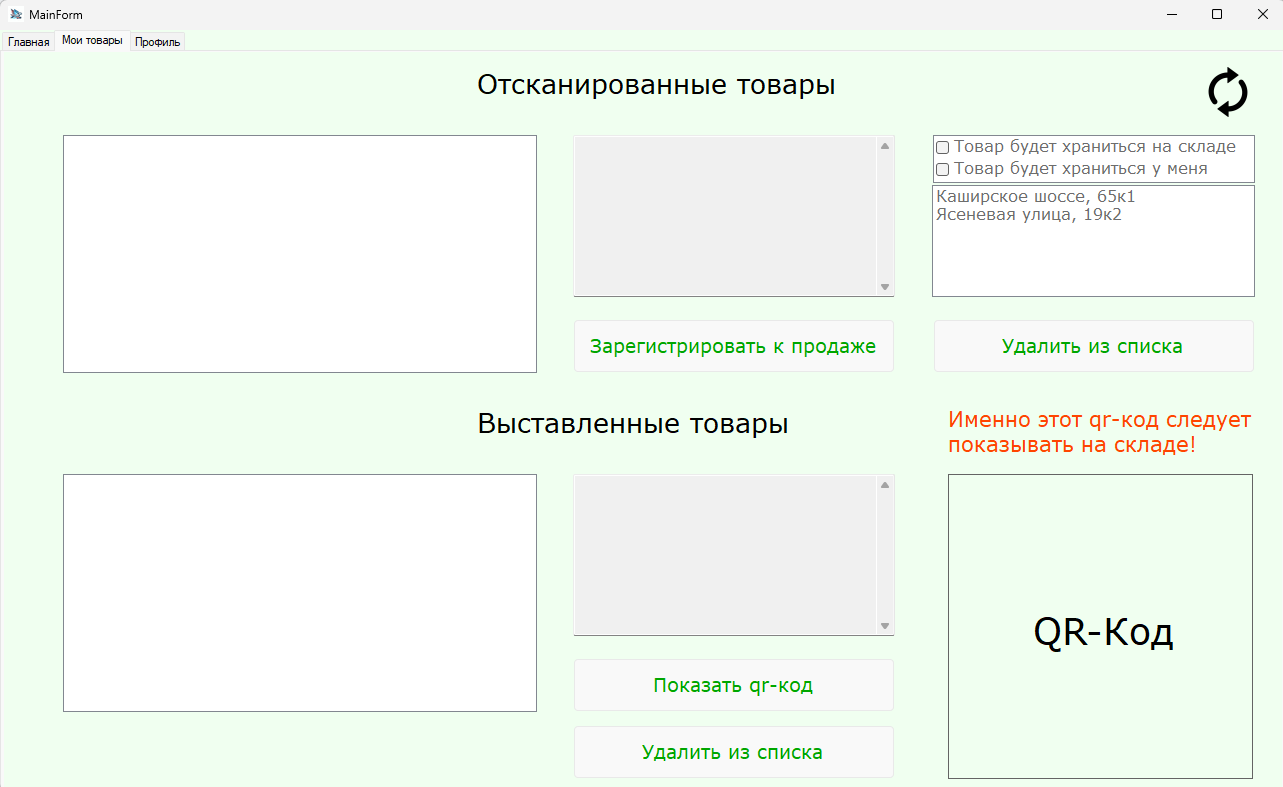
* 1. *Главная:*



*Рисунок 10.* Вкладка “Главная”

На вкладке “Главная” располагается информация о телеграмм боте, кнопка, которая перенаправит пользователя на окно сканирования QR-кода, и информация об отсканированном товаре.

* 1. *Мои товары:*

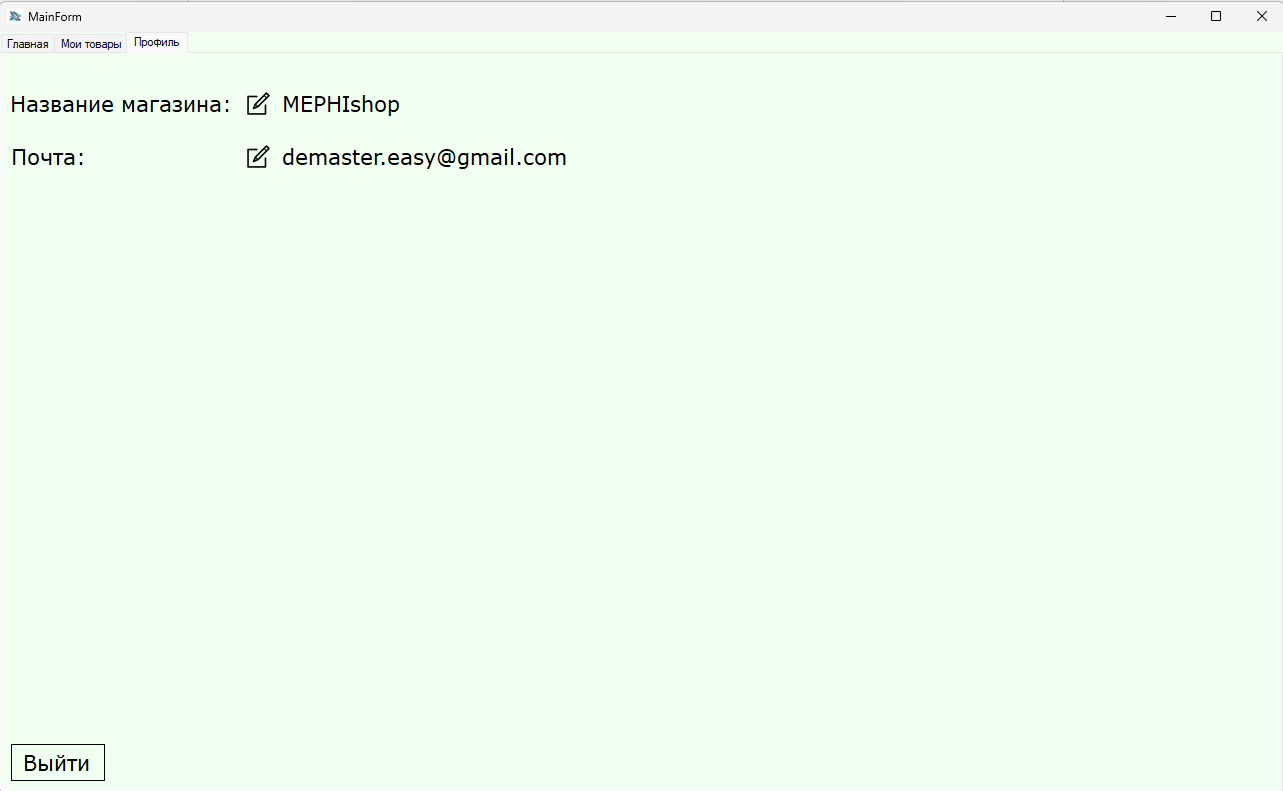


*Рисунок 11.* Вкладка “Мои товары”

На вкладке “Мои товары” располагается информация об отсканированных товарах (товары, QR-код которых пользователь создал в телеграмм боте и отсканировал в приложении) и о выставленных товарах (после сканирования товара, выбора места его хранения и нажатия кнопки “Зарегистрировать к продаже”, товар попадает в список “Выставленные товары”).

Информация об отсканированных товарах хранится на компьютере, пока открыто приложение, а информация о выставленных товарах хранится уже в базе данных на сервере и ожидает поступления на склад, если местом хранения был выбран склад. Если же товар будет храниться у продавца, то такой товар автоматически является авторизованным в системе и уже готов к размещению на маркетплейсе.

* 1. *Профиль:*



*Рисунок 12.* Вкладка “Профиль”

Во вкладке “Профиль” находится информация о пользователе. Его почта и название магазина (логин). Так же из этой вкладки можно осуществить выход из своего аккаунта.

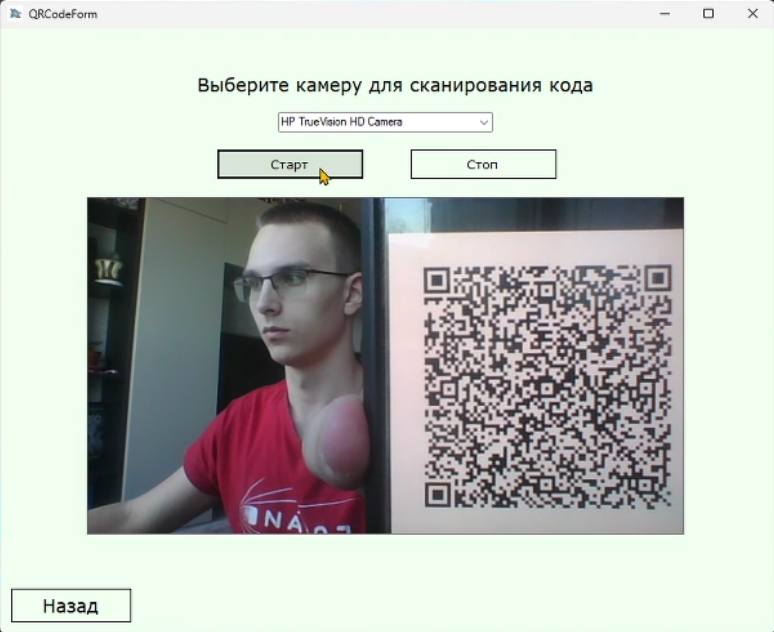
1. *Сканирование QR-кода:*

Чтобы приступить к сканированию QR-кода, необходимо на вкладке “Главная” нажать “Сканировать qr код”, после чего откроется окно для сканирования.



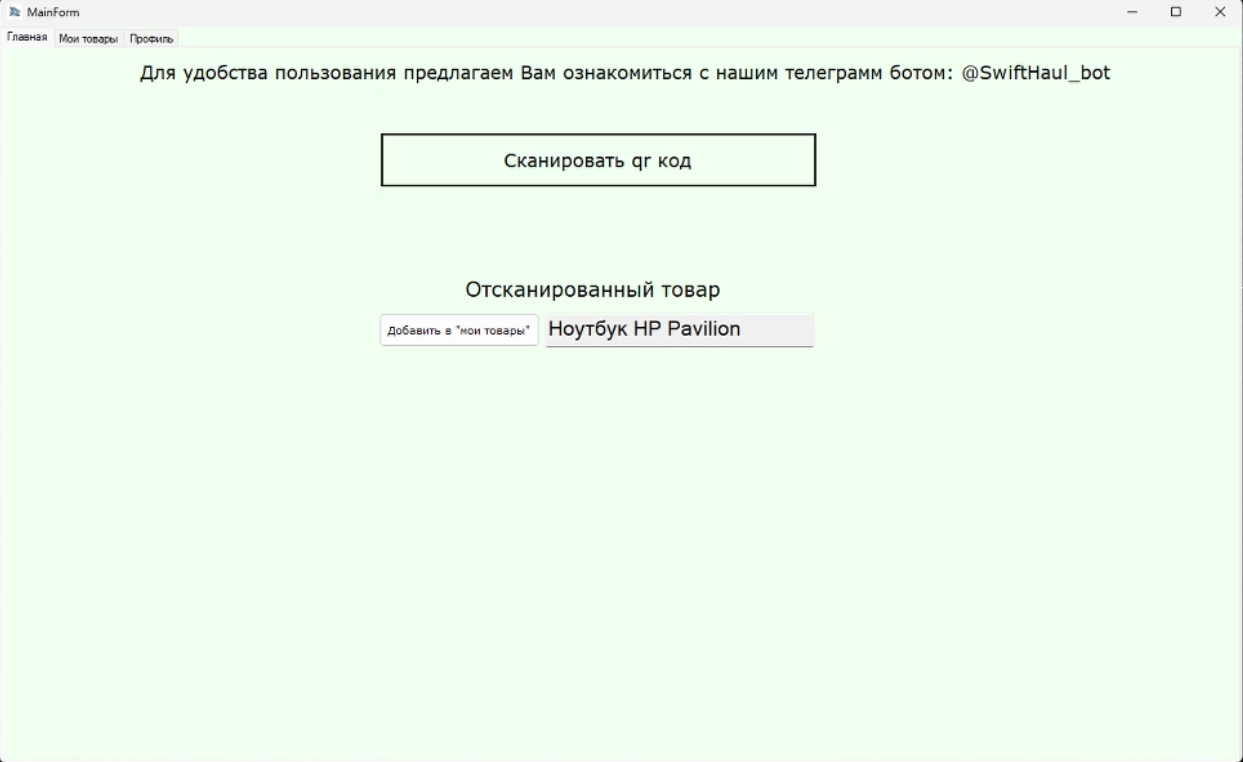
*Рисунок 13.* Окно сканирования QR-кода

В этом окне следует выбрать камеру для сканирования и нажать кнопку “Старт”, после чего включится выбранная камера, которая будет стараться обнаружить на на изображении QR-код.



*Рисунок 14.* Демонстрация попытки обнаружения QR-кода

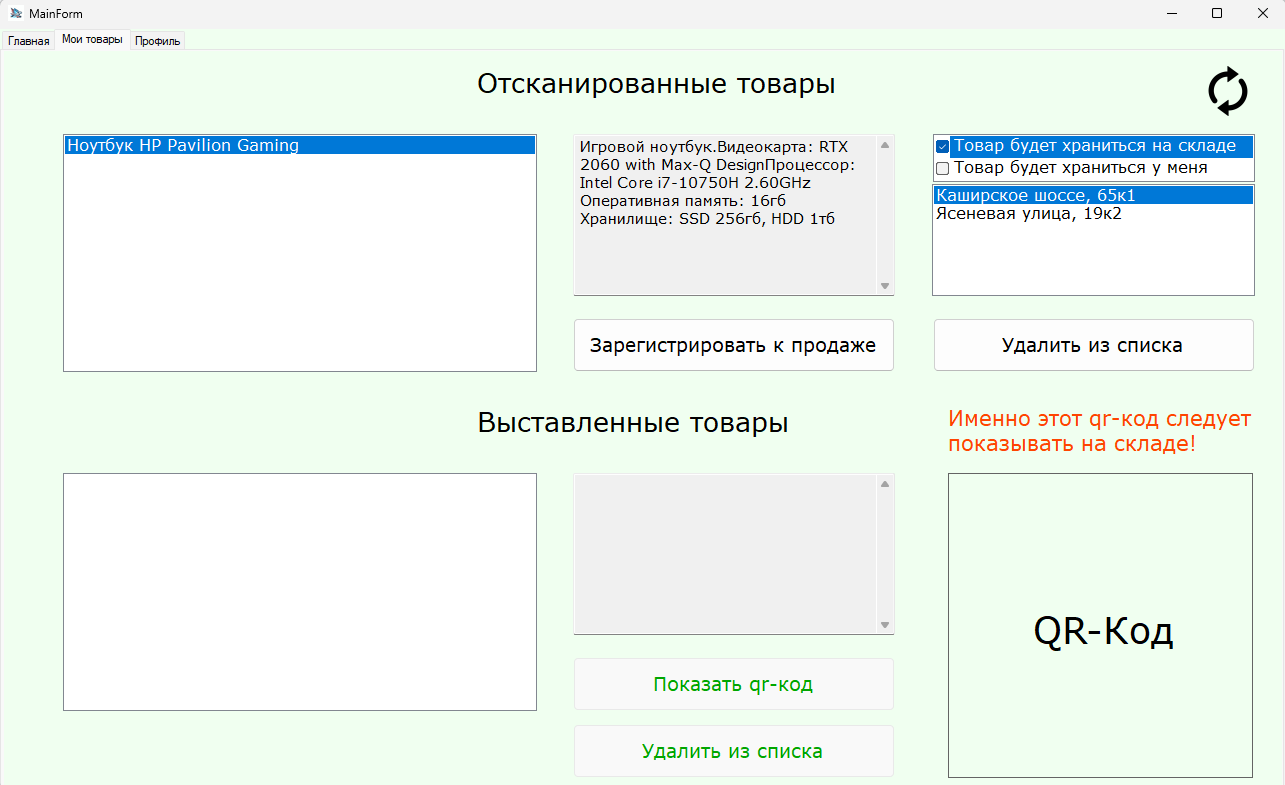
После успешного обнаружения QR-кода программой, текущее окно закрывается и пользователь обратно переходит на главную форму.



*Рисунок 15.* Главная форма после успешного сканирования QR-кода

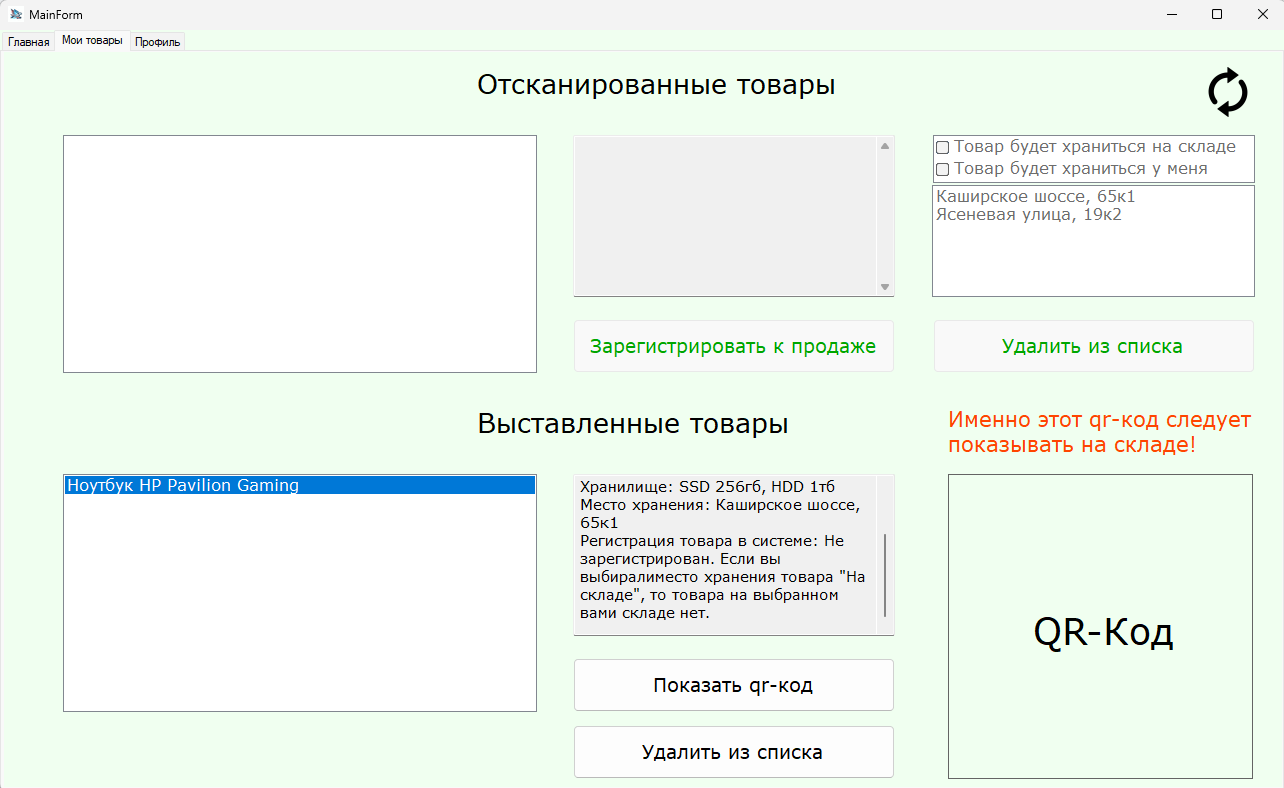
Теперь в текстовом поле “Отсканированного товара” можно увидеть название товара. После того, как пользователь нажмёт кнопку “Добавить в “мои товары”, товар добавиться в список отсканированных товаров.

Выберем в этом списке наш товар, после выберем, что товар будет храниться на складе и выберем адрес склада.



*Рисунок 16.* Подготовка товара к регистрации

Пользователь нажимает кнопку “Зарегистрировать к продаже” и товар появляется в списке выставленных товаров. В сером текстовом поле можно увидеть информацию о товаре, а также информацию о том, что товар не зарегистрирован в системе.



*Рисунок 17.* Демонстрация выставленного товара

Для регистрации товара в системе необходимо отобразить его QR-код, сохранить этот QR-код для того, чтобы показать на выбранном складе и только тогда товар будет считать зарегистрированным.

# **Техническое описание логики работы десктоп-приложения**

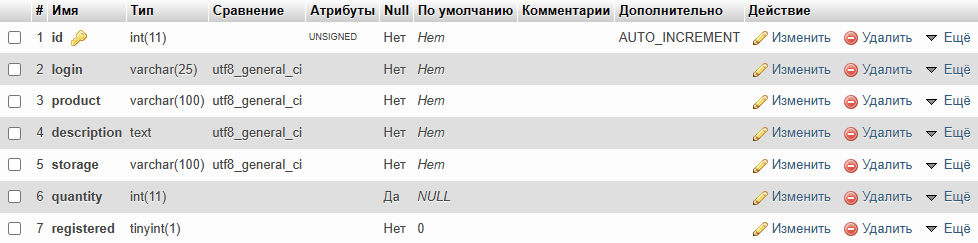
Для хранения информации о пользователях, их товарах и об адресах складских помещений использовались три базы данных MySql:

1. *users* - для хранения информации о пользователях



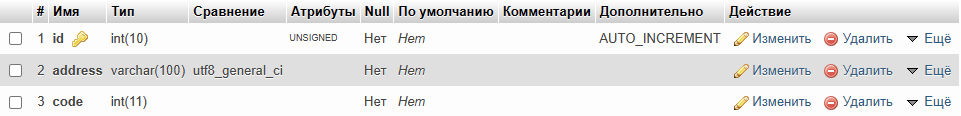
*Рисунок 18.* Структура базы данных *users*

1. *products* - для хранения информации о продуктах конкретного пользователя



*Рисунок 19.* Структура базы данных *products*

1. *warehouses* - для хранения информации об адресах складских помещений



*Рисунок 20.* Структура базы данных *warehouses*

1. *Механизм входа и регистрации пользователя*

Операции по подключению и выполнению команд, взаимодействующие с MySQL, выполнялись при помощи пакета **using** **MySql.Data.MySqlClient**; Регистрация пользователя происходит только после того, что введённый им код, присланный на почту, совпадает с отправленным. Сам механизм регистрации выглядит следующим образом:

DataBase db;

**try**

{

db = **new** DataBase();

}

**catch** (Exception ex)

{

MessageBox.Show($"Не удалось выполнить подключение к базе данных.\n{ex.Message}");

**return**;

}

MySqlCommand command = **new** MySqlCommand("INSERT INTO `users` (`login`, `password`, `email`, `isActive`) VALUES (@login, @password, @email, @isActive);", db.getConnection());

command.Parameters.Add("@login", MySqlDbType.VarString).Value = loginField.Text;

command.Parameters.Add("@password", MySqlDbType.VarString).Value = passwordField.Text;

command.Parameters.Add("@email", MySqlDbType.VarString).Value = loginFiel.Text;

command.Parameters.Add("@isActive", MySqlDbType.Byte).Value = **1**;

db.openConnection();

**if** (command.ExecuteNonQuery() == **1**)

{

Hide();

MainForm mainForm = **new** MainForm(db);

mainForm.Show();

}

**else**

MessageBox.Show("Аккаунт не был создан. Какие-то проблемы");

db.closeConnection();

Как только команда успешно выполняется, то пользователя перенаправляет на главную форму.

Механизм входа аналогичен регистрации, за исключением того, что строка в базу данных не добавляется, а сравнивается с уже имеющимися и если такая есть, и логин с паролем совпадают, то происходит переход на главную форму:

DataBase db;

**try**

{

db = **new** DataBase();

}

**catch** (Exception ex)

{

MessageBox.Show($"Не удалось выполнить подключение к базе данных.\n{ex.Message}");

**return**;

}

DataTable table = **new** DataTable();

MySqlCommand command = **new** MySqlCommand("SELECT \* FROM `users` WHERE `login` = @log AND `password` = @pass", db.getConnection());

command.Parameters.Add("@log", MySqlDbType.VarChar).Value = loginField.Text;

command.Parameters.Add("@pass", MySqlDbType.VarChar).Value = password;

MySqlDataAdapter adapter = **new** MySqlDataAdapter();

adapter.SelectCommand = command;

adapter.Fill(table);

**if** (table.Rows.Count > **0**)

{

MainForm.userEmail = table.Rows[**0**].Field<**string**>(**4**);

Hide();

MainForm mainForm = **new** MainForm(db);

mainForm.Show();

}

**else**

MessageBox.Show("Такого пользователя нет");

1. *Сканирование QR-кода*

Изображение с камеры программа получает с помощью пакетов **using** **AForge.Video**; **using** **AForge.Video.DirectShow**; Декодирование изображения и попытки распознавания в нём QR-кода выполняется с помощью пакета **using** **ZXing**;

Функция распознавания выполняется ежесекундно. Как только QR-код удалось распознать, информация об этом передаётся на главную форму, где начинается десериализация класса Product.

**if** (cameraBox.Image != **null**)

{

BarcodeReader barcodeReader = **new** BarcodeReader();

Result result = barcodeReader.Decode((Bitmap)cameraBox.Image);

**if** (result != **null**)

{

StopVideo();

cameraBox.Image = **null**;

cameraImageText.Visible = **true**;

timer1.Stop();

// Product product = JsonSerializer.Deserialize<Product>(result.ToString());

// MessageBox.Show(result.ToString());

mainForm.JsonObject = result.ToString();

Hide();

mainForm.startDeserialize();

mainForm.Show();

}

}

Функция десериализации класса Product выглядит следующим образом:

**public** **void** **startDeserialize**()

{

scannedProduct = **null**;

productNameTextBox.Text = **string**.Empty;

JsonSerializerOptions options = **new** JsonSerializerOptions

{

PropertyNameCaseInsensitive = **true**

};

scannedProduct = JsonSerializer.Deserialize<Product>(jsonObject, options);

productNameTextBox.Text = scannedProduct.ProductName;

}

Где scannedProduct это объект класса Product.

1. *Регистрация товара в системе*

Регистрация товара в системе происходит по тому же принципу, что и регистрация пользователя. То есть в таблицу *products* добавляется строка с информацией о выставленном товаре.

Если в пункте хранения товара был выбран склад, то регистрации товара в системе ставится значение 0, если товар будет храниться у продавца, то ставится 1. В колонку description записывается сериализованный в json объект класса Product.

MySqlCommand command = **new** MySqlCommand("INSERT INTO `products` (`login`, `product`, `description`, `storage`, `registered`) VALUES (@login, @product, @description, @storage, @reg);", db.getConnection());

command.Parameters.Add("@login", MySqlDbType.VarString).Value = shopName;

command.Parameters.Add("@product", MySqlDbType.VarString).Value = scannedProducts.SelectedItem;

scannedProductsList[scannedProducts.SelectedIndex].Seller = shopName;

JsonSerializerOptions options = **new** JsonSerializerOptions

{

Encoder = System.Text.Encodings.Web.JavaScriptEncoder.Create(UnicodeRanges.All)

};

**string** json = JsonSerializer.Serialize(scannedProductsList[scannedProducts.SelectedIndex], options);

command.Parameters.Add("@description", MySqlDbType.VarString).Value = json;

**if** (storageSelection.SelectedIndex == **0**)

{

command.Parameters.Add("@reg", MySqlDbType.Binary).Value = **0**;

command.Parameters.Add("@storage", MySqlDbType.VarString).Value = warehouseAddress.Items[warehouseAddress.SelectedIndex];

}

**else**

{

command.Parameters.Add("@reg", MySqlDbType.Binary).Value = **1**;

command.Parameters.Add("@storage", MySqlDbType.VarString).Value = "Дома";

}

db.openConnection();

**if** (command.ExecuteNonQuery() == **1**)

{

updateInformation();

}

**else**

{

MessageBox.Show("Товар не удалось зарегистрировать. Какие-то проблемы");

db.closeConnection();

**return**;

}

db.closeConnection();

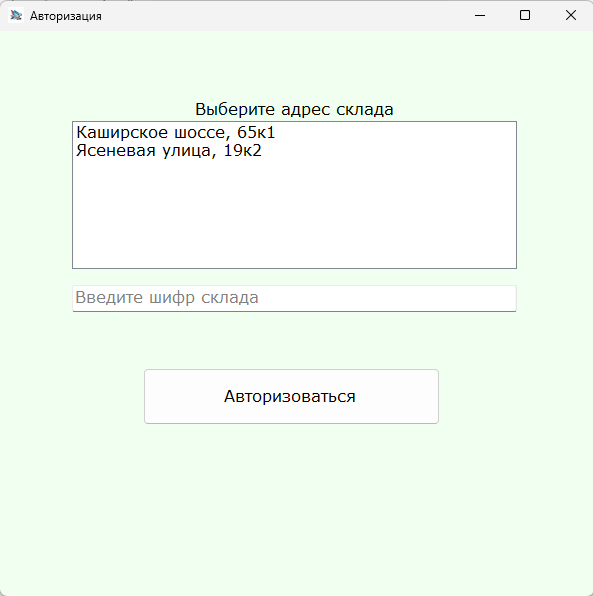
При выполнении функции updateInformation(); происходит полное обновление всей информации, подгружаемой с базы данных.

# **Разработка десктоп-приложения для складских компьютеров на C# с использованием WindowsForms**

# **Функциональные возможности десктоп-приложения**

1. *Авторизация:*

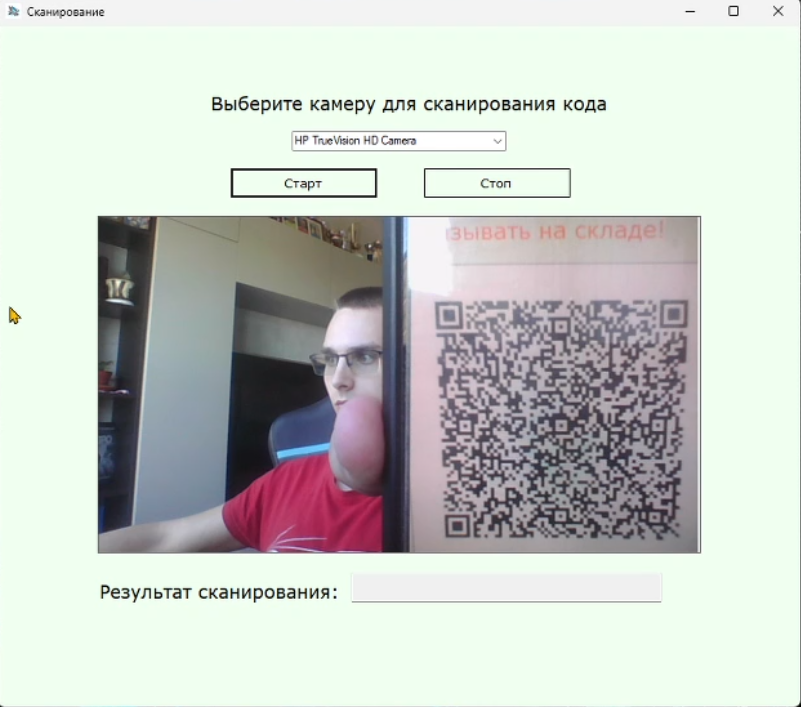
В окне авторизации сотруднику необходимо войти в систему склада по указанному адресу и ввести шифр выбранного склада. Такое решение предотвратит случайное подключение не к тому складскому помещению.



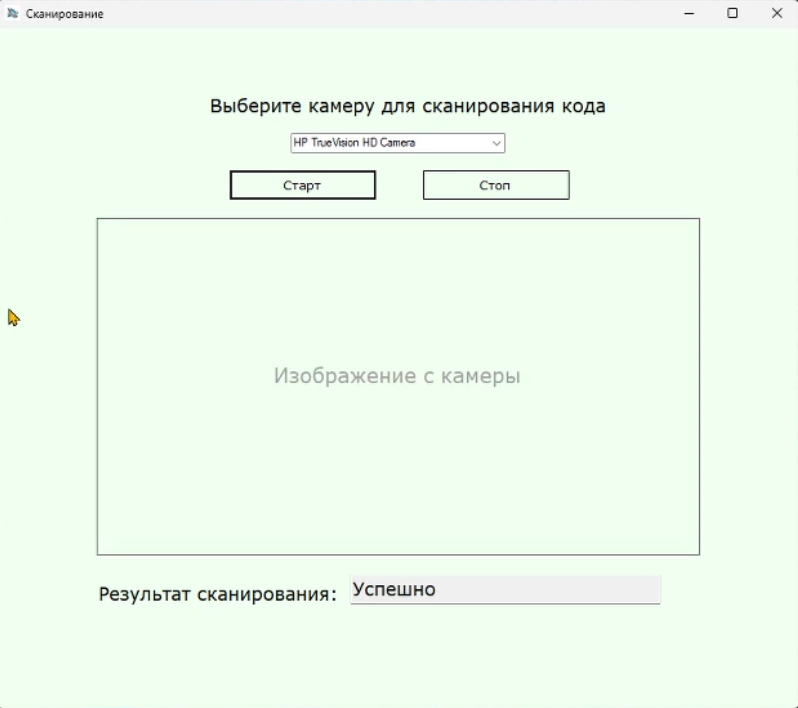
*Рисунок 21*. Окно авторизации.

1. *Форма сканирования:*

После успешного входа сотрудником в систему, открывается форма, которая позволяет сканировать QR-код. После успешного сканирования товар отмечается зарегистрированным в системе.



*Рисунок 22*. Демонстрация попытки обнаружения QR-кода



*Рисунок 23*. Демонстрация успешного сканирования и успешной регистрации товара в системе

В базе данных это сканирование выглядит следующим образом:



*Рисунок 24*. Информация о товаре до успешного сканирования



*Рисунок 25*. Информация о товаре после успешного сканирования

# **Техническое описание логики работы десктоп-приложения**

1. *Авторизация сотрудника в системе.*

Авторизация происходит по принципу авторизации пользователя в системе регистрации его товара. Сотрудник выбирает склад, вводит шифр для этого склада и эти данные сравниваются с теми, что лежат в базе данных.

**try**

{

db = **new** DataBase();

}

**catch** (Exception ex)

{

MessageBox.Show($"Не удалось выполнить подключение к базе данных.\n{ex.Message}");

**return**;

}

table = **new** DataTable();

MySqlCommand command = **new** MySqlCommand("SELECT address, code FROM warehouses;", db.getConnection());

MySqlDataAdapter adapter = **new** MySqlDataAdapter();

adapter.SelectCommand = command;

adapter.Fill(table);

**if** (table.Rows[warehouseAddress.SelectedIndex].Field<**int**>(**1**) == **int**.Parse(codeWarehouseTextBox.Text))

{

Hide();

ScannerForm sf = **new** ScannerForm(db, table.Rows[warehouseAddress.SelectedIndex].Field<**string**>(**0**));

sf.Show();

}

**else**

{

MessageBox.Show("Неправильно введён шифр"); }

1. *Сканирование QR-кода.*

Сканирование QR-кода происходит ровно по тому же принципу, что и сканирование QR-кода в десктоп-приложении пользователя. Проверка успешного сканирования происходит ежесекундно. В случае успешного сканирования проверяется совпадения складов (склад, указанный в товаре и склад, как место сканирования) и наличие такого товара. Если проверка успешно пройдена, то значение столбца registered меняется на 1.

**string** json = result.ToString();

Product product = **new** Product();

JsonSerializerOptions options = **new** JsonSerializerOptions

{

PropertyNameCaseInsensitive = **true**

};

product = JsonSerializer.Deserialize<Product>(json, options);

DataTable table = **new** DataTable();

MySqlCommand command = **new** MySqlCommand("SELECT \* FROM `products` WHERE `login` = @log AND `product` = @prodName AND `description` = @descr;", db.getConnection());

command.Parameters.Add("@log", MySqlDbType.VarChar).Value = product.Seller;

command.Parameters.Add("@prodName", MySqlDbType.VarChar).Value = product.ProductName;

command.Parameters.Add("@descr", MySqlDbType.VarChar).Value = json;

MySqlDataAdapter adapter = **new** MySqlDataAdapter();

adapter.SelectCommand = command;

adapter.Fill(table);

**if**(table.Rows.Count > **0**)

{

**if** (table.Rows[**0**].Field<**string**>(**4**) != warehouse)

{

scannResultText.Text = "Не удалось";

scannResultText.ForeColor = Color.Red;

command.Dispose();

**return**;

}

}

command.Dispose();

command = **new** MySqlCommand("UPDATE `products` SET `registered` = @reg WHERE `login` = @log AND `product` = @prodName AND `description` = @descr;", db.getConnection());

command.Parameters.Add("@reg", MySqlDbType.Int64).Value = **1**;

command.Parameters.Add("@log", MySqlDbType.VarChar).Value = product.Seller;

command.Parameters.Add("@prodName", MySqlDbType.VarChar).Value = product.ProductName;

command.Parameters.Add("@descr", MySqlDbType.VarChar).Value = json;

db.openConnection();

**int** rowsAffected = command.ExecuteNonQuery();

**if** (rowsAffected > **0**)

{

MySqlCommand checkCommand = **new** MySqlCommand("SELECT \* FROM `products` WHERE `login` = @log AND `product` = @prodName AND `description` = @descr;", db.getConnection());

checkCommand.Parameters.Add("@log", MySqlDbType.VarChar).Value = product.Seller;

checkCommand.Parameters.Add("@prodName", MySqlDbType.VarChar).Value = product.ProductName;

checkCommand.Parameters.Add("@descr", MySqlDbType.VarChar).Value = json;

**using** (MySqlDataReader reader = checkCommand.ExecuteReader())

{

**if** (reader.Read())

{

**if** (reader.GetInt64("registered") == **1**)

{

scannResultText.Text = "Успешно";

scannResultText.ForeColor = Color.Green;

}

**else**

{

scannResultText.Text = "Не удалось";

scannResultText.ForeColor = Color.Red;

}

}

}

}

db.closeConnection();

# 

# **Заключение**

Проект "Система управления складом (складские операции) и логистикой" был разработан с целью создания эффективного и удобного инструмента для автоматизации процессов регистрации и учета товаров. В ходе работы были решены поставленные задачи и достигнуты основные цели проекта, что позволило существенно повысить точность и скорость обработки данных, а также улучшить контроль за состоянием и перемещением товаров на складе. Основные достижения проекта включают:

1. *Разработка телеграмм-бота на C#.*

Создан функциональный телеграмм-бот, который позволяет пользователям вводить данные о товарах и генерировать уникальные QR-коды. Бот обеспечивает интуитивно понятный интерфейс и высокую скорость обработки запросов.

1. *Создание десктоп-приложения для пользователей на C# с использованием WindowsForms.*

Реализовано приложение, которое позволяет пользователям регистрировать товары, сканировать QR-коды, выбирать место хранения и управлять информацией о товарах. Приложение интегрируется с телеграмм-ботом, обеспечивая удобство и простоту использования.

1. *Разработка десктоп-приложения для складских компьютеров на C# с использованием WindowsForms.*

Создано приложение для складских сотрудников, которое позволяет сканировать QR-коды товаров и отмечать их нахождение на складе. Это приложение интегрируется с сервером для обновления информации о товарах в реальном времени.

1. *Тестирование системы.*

Проведено всестороннее тестирование всех компонентов системы, выявлены и устранены ошибки, проведена оптимизация производительности.

Рекомендации для дальнейшего развития.

1. *Расширение функциональности.*

Добавление новых функций, таких как автоматическое уведомление пользователей о статусе товаров, интеграция с другими системами управления и отчетности.

1. *Мобильное приложение.*

Разработка мобильного приложения для удобства пользователей, что позволит сканировать QR-коды и управлять товарами непосредственно со смартфонов и планшетов.

1. *Оптимизация производительности.*

Проведение дополнительной оптимизации производительности системы для обеспечения стабильной работы при увеличении объема данных и числа пользователей.

1. *Аналитика и отчеты.*

Внедрение модулей аналитики и отчетности, которые позволят пользователям получать детализированную информацию о движении и состоянии товаров на складе.

Заключение подчеркивает значимость проекта для улучшения управления складскими операциями и логистикой, отмечает достигнутые результаты и предоставляет рекомендации для дальнейшего развития и совершенствования системы. Проект создает предпосылки для дальнейших исследований и внедрения новых технологий в области автоматизации складских процессов, что способствует повышению конкурентоспособности и эффективности предприятий.

# **Список литературы**

1. .NET Client for Telegram Bot API. - Текст: электронный // GitHub: [сайт]. - 2024. - URL:<https://github.com/TelegramBots/Telegram.Bot>
2. WindowsForms C#. - Текст: электронный // metanit.com: [сайт]. - 2024. - URL: <https://metanit.com/sharp/windowsforms/>
3. Администрирование MySQL через Интернет. - Текст: электронный // phpmyadmin.net: [сайт]. - 2024. - URL: https://www.phpmyadmin.net/
4. Получить изображение с веб-камеры в форме на C#. - Текст: электронный // code-enjoy.ru: [сайт]. - 2024. - URL: https://code-enjoy.ru/poluchit\_izobrazheniye\_v\_formu\_na\_si\_sharp/
5. .NET и MySQL. - Текст: электронный // Oracle: [сайт]. - 2024. - URL: https://dev.mysql.com/doc/connector-net/en/connector-net-tutorials.html
6. .NET Windows Forms + Базы данных SQL. - Текст: электронный // youtube.com: [сайт]. - 2024. - URL: https://www.youtube.com/playlist?list=PL0lO\_mIqDDFWOMqSKFaLypANf1W7-o87q
7. Сканирование QR-кода. - Текст: электронный // stackoverflow.com: [сайт]. - 2024. - URL: https://stackoverflow.com/questions/75630846/read-qr-code-from-byte-using-zxing-net-on-net7
8. Генерация QR-кода. - Текст: электронный // stackoverflow.com: [сайт]. - 2024. - URL: <https://ru.stackoverflow.com/questions/1422834/%d0%93%d0%b5%d0%bd%d0%b5%d1%80%d0%b0%d1%86%d0%b8%d1%8f-qr-%d0%ba%d0%be%d0%b4%d0%b0-zxing>
9. Как работает валберис: все, что нужно знать о самом популярном маркетплейсе. - Текст: электронный // FB.ru: [сайт]. - 2024. - URL: <https://fb.ru/article/517745/2023-kak-rabotaet-valberis-vse-chto-nujno-znat-o-samom-populyarnom-marketpleyse>
10. Логистика на Озоне. - Текст: электронный // mpstats.io: [сайт]. - 2024. - URL: <https://mpstats.io/media/ozon/logistika>
11. Курьеры, серверы и налоги: как устроен бизнес Amazon. - Текст: электронный // journal.tinkoff.ru: [сайт]. - 2024. - URL: https://journal.tinkoff.ru/news/review-amazon/