**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

**федеральное государственное бюджетное образовательное**

**учреждение высшего образования**

**«Казанский национальный исследовательский технический**

**университет им. А.Н. Туполева-КАИ»**

**(КНИТУ-КАИ)**

Институт компьютерных технологий и защиты информации

(наименование института (факультета), филиала)

\_\_Кафедра Прикладной Математики и Информатики

(наименование кафедры)

(09.03.04) Программная инженерия

(шифр и наименование направления подготовки (специальности))

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

по дисциплине:

«Базы данных»

на тему: «Онлайн магазин»

Обучающийся 4310 Нигамадянов Ф.М.

(номер группы) (подпись, дата) (Ф.И.О.)

Руководитель доцент каф. ПМИ Валитова Н.Л.

(должность) (Ф.И.О.)

Курсовая работа зачтена с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (подпись, дата)

Казань 2022

# Оглавление

[**Оглавление**](#_qr9zl8ru1wao) **2**

[**I. Системный анализ предметной области**](#_nuom0u3cf9ko) **3**

[**II. Проектирование инфологической модели предметной области**](#_pkeuwt6p46za) **3**

[1. Составление перечня атрибутов](#_qkrwx4q4mv2u) 3

[2. Определение сущностей](#_lspk7n21hsvx) 5

[3. Определение связей между сущностями](#_f9ulc5tyo2uq) 5

[4. Результат инфологической модели](#_8v7ten1zxjn4) 6

[**III. Проектирование концептуальной базы данных**](#_6v2546dwey00) **6**

[1. Преобразование сущностей в отношения](#_rjvhmj3ynatm) 6

[2. Определение свойств атрибутов](#_cuelh4jkwms) 7

[**VIII. Руководство программиста**](#_2zrwf9muri9j) **25**

[1. Класс Program.cs](#_8z3esxeyk5wa) 26

[2. Класс ShopDataBase.cs](#_qxupnjoev9y9) 26

[3. Класс Sign\_in.cs](#_zbbhu28dve8y) 27

[4. Класс Sign\_up.cs](#_ejpmm1tchpro) 27

[5. Класс OnlineShop.cs](#_ghuuvyus610g) 28

[6. Класс AlterFormForConsumer.cs](#_stygl3za2ydu) 29

[**Заключение**](#_lmlfqb54kzw3) **30**

[**Литература**](#_jcwtx9ehd477) **31**

# Системный анализ предметной области

В базе данных будет храниться информация о покупателях, товарах, приобретаемых покупателями, покупках, складах.

Нам будет необходимо знать электронную почту покупателя, который будет его идентифицировать. Не обойтись и без ФИО пользователя, чтобы понимать, кому будет вручаться товар. Для того, чтобы знать, куда доставлять товар, необходимо знать адрес местожительства покупателя.

У товара имеется код товара, который будет идентифицировать наш товар. Так же нужно иметь наименование товара, т.к. покупателю код товара не будет ни о чем говорить. Очевидно, у товара должна быть цена. Чтобы связать сущность «товар» и сущность «склад», необходимо, чтобы у товара было поле «код склада». Чтобы проверить товар на наличие, будем хранить количество определенного товара.

Помимо прочего у нас имеется информация о какой-либо покупке. Эта информация содержит: код товара, электронную почту покупателя, дата покупки и дата получения товара.

У склада будет иметься код (для его идентификации). Еще нам нужно хранить адрес расположения склада.

Состав пользователей: системные администраторы, контролирующие оборот товаров.

Типовые запросы:

* запрос на удостоверение того, что покупатель получил товар с определенным кодом
* сколько товаров было продано за определенный период

# Проектирование инфологической модели предметной области

## Составление перечня атрибутов

| Обозначение атрибута | Описание атрибута | Примечание |
| --- | --- | --- |
| X1 | Электронная почта |  |
| X2 | ФИО |  |
| X3 | Адрес местожительства |  |
| X4 | Код товара |  |
| X5 | Наим. товара |  |
| X6 | Цена |  |
| X7 | Код склада |  |
| X8 | Ко-во товара |  |
| X9 | Код товара |  |
| X10 | Электронная почта |  |
| X11 | Дата покупки товара |  |
| X12 | Дата получения товара |  |
| X13 | Код склада |  |
| X14 | Адрес склада |  |

## Определение сущностей

| Обозначение сущности | Описание сущности | Составляющие атрибуты | Первичный ключ |
| --- | --- | --- | --- |
| Y1 | Покупатель | X1-X3 | Электронная почта |
| Y2 | Товар | X4-X8 | Код товара |
| Y3 | Покупка | X9-X12 | Код товара & Электронная почта |
| Y4 | Склад | X13, X14 | Код склада |

## Определение связей между сущностями

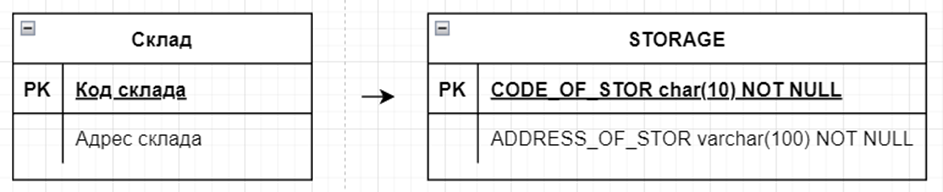
| Сущность №1 | Сущность №2 | Тип связи | Обязательность |
| --- | --- | --- | --- |
| Y1 | Y3 | 1:М | 0:1 |
| Y2 | Y3 | 1:М | 0:1 |
| Y4 | Y2 | 1:М | 0:1 |

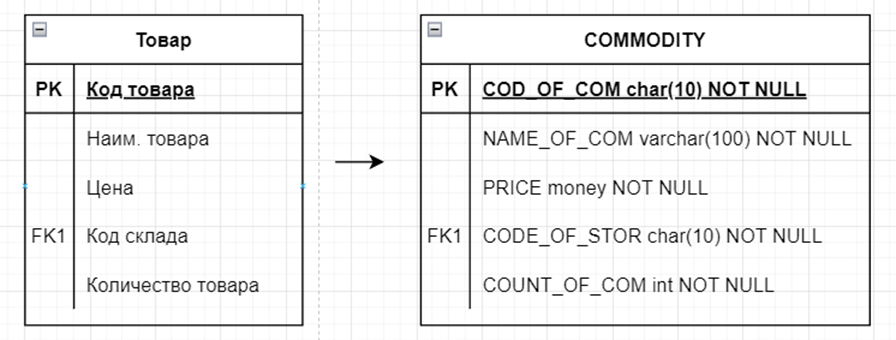
## Результат инфологической модели

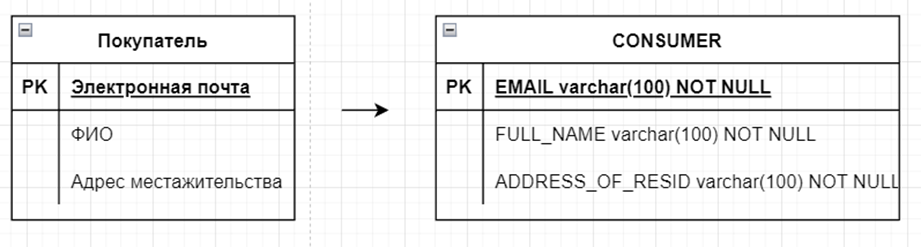


# Проектирование концептуальной базы данных

## Преобразование сущностей в отношения









## Определение свойств атрибутов

Сущность склад:

| Атрибут | Тип данных | Первичный ключ | Уникальность | Обязательность | Знач. по умолчанию | Лог. условие |
| --- | --- | --- | --- | --- | --- | --- |
| ADDRESS\_OF\_STOR | VARCHAR(200) | - | - | + |  |  |
| CODE\_OF\_STOR | CHAR(10) | + | + | + |  | LENGTH=10 |

Сущность товар:

| Атрибут | Тип данных | Первичный ключ | Уникальность | Обязательность | Знач. по умолчанию | Лог. условие |
| --- | --- | --- | --- | --- | --- | --- |
| CODE\_OF\_COM | CHAR(10) | + | + | + |  | LENGTH=10 |
| NAME\_OF\_COM | VARCHAR(100) | - | - | + |  |  |
| CODE\_OF\_STOR | CHAR(10) | - | - | + |  |  |
| PRICE | MONEY | - | - | + |  |  |
| COUNT | INT | - | - | + |  | >=0 |

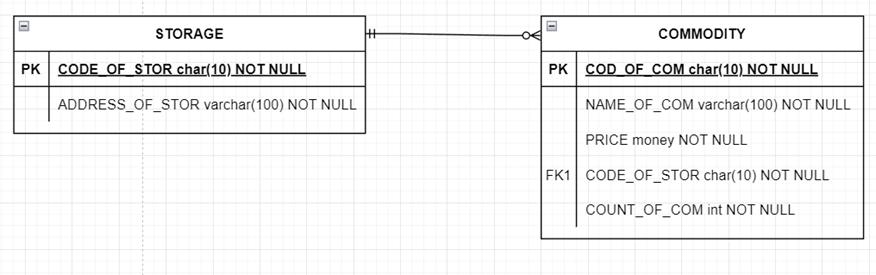
Сущность покупка:

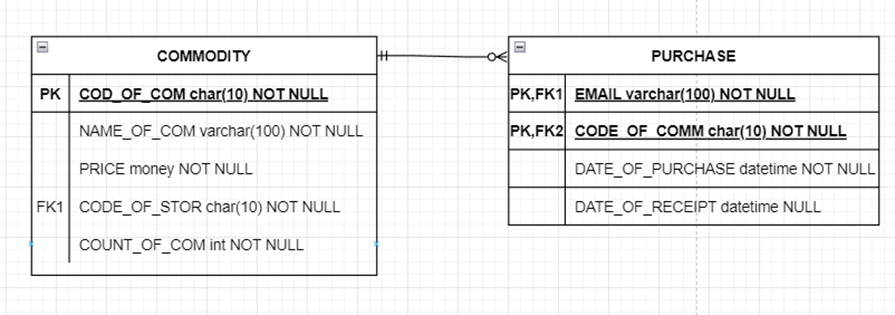
| Атрибут | Тип данных | Первичный ключ | Уникальность | Обязательность | Знач. по умолчанию | Лог. условие |
| --- | --- | --- | --- | --- | --- | --- |
| EMAIL | VARCHAR(100) | + | - | + |  |  |
| CODE\_OF\_COMM | VARCHAR(100) | + | - | + |  |  |
| DATE\_OF\_PURCHASE | DATETIME | + | - | + |  |  |
| DATE\_OF\_RECEIPT | DATETIME | - | - | + |  | >DATE\_OF\_PURCHASE |

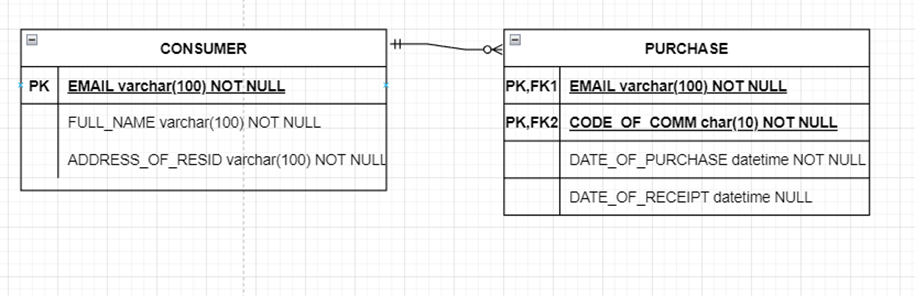
Сущность покупатель:

| Атрибут | Тип данных | Первичный ключ | Уникальность | Обязательность | Знач. по умолчанию | Лог. условие |
| --- | --- | --- | --- | --- | --- | --- |
| EMAIL | VARCHAR(100) | + | + | + |  |  |
| FULL\_NAME | VARCHAR(100) | - | - | + |  |  |
| ADDRESS\_OF\_RESID | VARCHAR(100) | - | - | + |  |  |

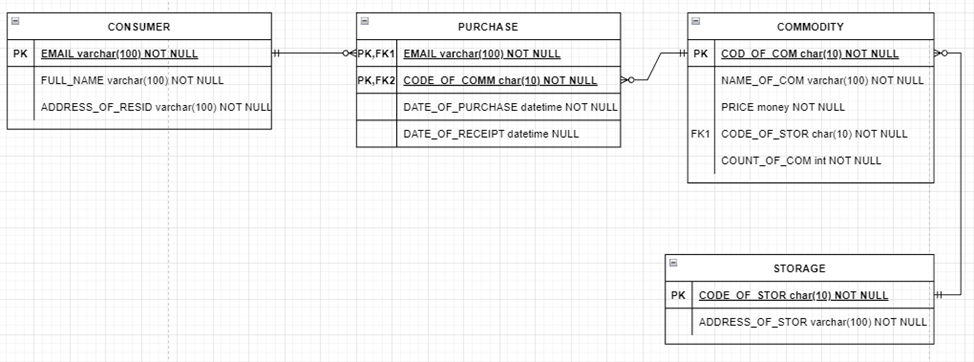
Определение внешних ключей(1:М)







* 1. Концептуальная схема



1. Нормализация отношений

Следующим шагом в проектировании является нормализация отношений.

Рассмотрим отношения «Покупатель», «Склад», «Товар», «Покупка». Каждый из атрибутов данного отношения простой, поэтому они все находятся в 1НФ.

Все отношения «Покупатель», «Склад», «Товар», «Покупка» находятся в 1НФ, а также каждый неключевой атрибут функционально полно зависит от составного ключа.

Для преобразования сущности к 3НФ необходимо проанализировать наличие в ней транзитивной зависимости неключевых атрибутов от первичного ключа и осуществить их декомпозицию, разрушив транзитивную зависимость.

В наших отношения транзитивных зависимостей нет, поэтому они все находятся в 3НФ.

1. Реализация базы данных в Microsoft SQL Server Management Studio

В процессе реализации базы данных были созданы четыре таблицы.

Таблица товаров:

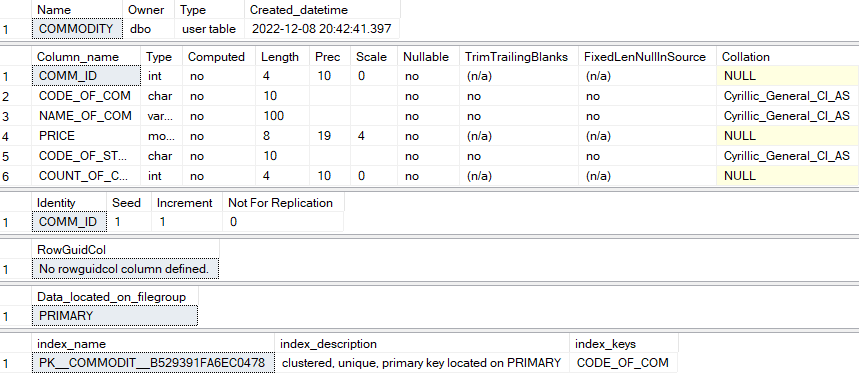


Таблица склад:

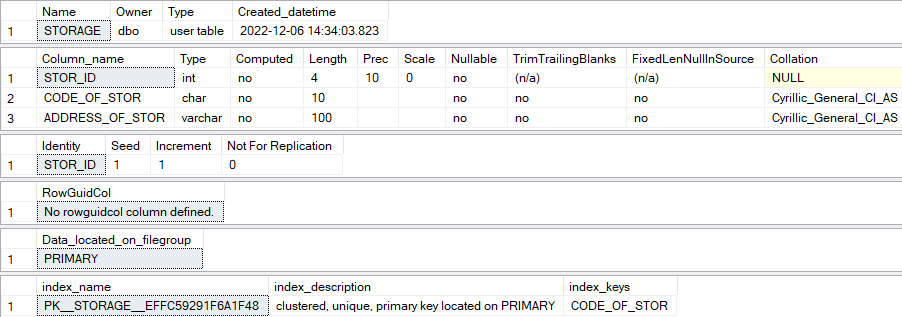


Таблица покупатель:

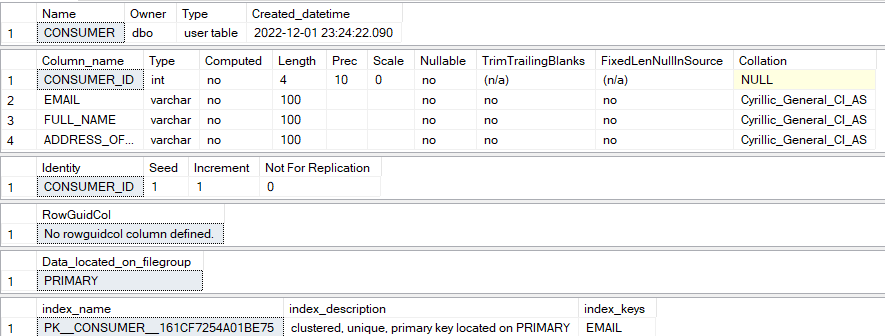


Таблица покупок:

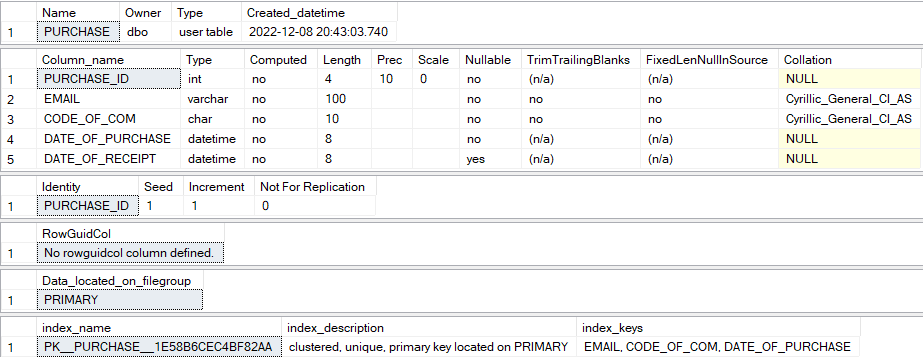
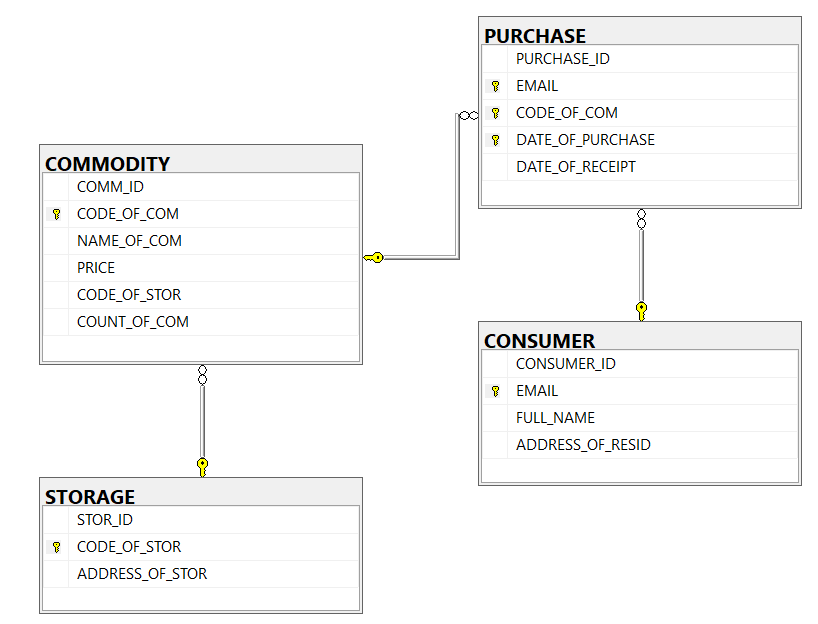


Схема данных и связей для данной реляционной базы данных выглядит следующим образом:



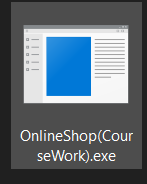
1. Реализация запросов

В рамках данной курсовой работы для создания рабочего приложения были реализованы следующие запросы:

| Удаление строки пользователя |
| --- |
| “DELETE FROM CONSUMER where EMAIL = @email” |
| Удаление строки товара |
| "DELETE FROM COMMODITY where CODE\_OF\_COM = @code\_of\_com" |
| Удаление строки склада |
| "DELETE FROM STORAGE where CODE\_OF\_STOR = @code\_of\_stor" |
| Удаление строки покупок |
| “DELETE FROM PURCHASE where EMAIL = @email and CODE\_OF\_COM = @code\_of\_com and DATE\_OF\_PURCHASE = @date\_of\_purchase" |
| Добавление в строку пользователя |
| "INSERT INTO CONSUMER VALUES (@email , @full\_name, @address\_of\_residence)" |
| Добавление в строку товара |
| "INSERT INTO COMMODITY VALUES (@code\_of\_com, @name\_of\_com, @price, @code\_of\_stor, @count\_of\_com)" |
| Добавление в строку склада |
| "INSERT INTO STORAGE VALUES (@code\_of\_stor, @address\_of\_stor)" |
| Добавление в строку покупок |
| "INSERT INTO PURCHASE VALUES (@email, @code\_of\_com, @date\_of\_purchase, @date\_of\_receipt)" |
| Поиск по пользователям |
| "SELECT \* FROM PURCHASE WHERE @selected\_field LIKE '%@what\_wrote\_user %'" |
| Поиск по товарам |
| "SELECT \* FROM COMMODITY WHERE @selected\_field LIKE '%@what\_wrote\_user %'" |
| Поиск по складам |
| “SELECT FROM STORAGE WHERE @selected\_field LIKE '%@what\_wrote\_user %'" |
| Поиск по покупкам |
| “SELECT FROM PURCHASE WHERE @selected\_field LIKE '%@what\_wrote\_user %'" |
| Изменение строки пользователя |
| "UPDATE CONSUMER SET EMAIL = @email, FULL\_NAME = @full\_name, ADDRESS\_OF\_RESID = @addres\_of\_resid WHERE CONSUMER\_ID = @consumer\_id" |
| Изменение строки товаров |
| "UPDATE COMMODITY SET CODE\_OF\_COM = @code\_of\_com, NAME\_OF\_COM = @name\_of\_com, PRICE = @price, CODE\_OF\_STOR = @code\_of\_stor, COUNT\_OF\_COM = @count\_of\_com WHERE COMM\_ID = @comm\_id |
| Изменение строки складов |
| “UPDATE STORAGE SET CODE\_OF\_STOR = @code\_of\_stor, ADDRESS\_OF\_STOR = @address\_of\_store WHERE STOR\_ID = @stor\_id” |
| Изменение строки покупок |
| “UPDATE PURCHASE SET EMAIL = @email, CODE\_OF\_COM = @code\_of\_com, DATE\_OF\_PURCHASE = @date\_of\_purchase, DATE\_OF\_RECEIPT = @date\_of\_receipt WHERE PURCHASE\_ID = @purchase\_id |
| Справочный вывод о покупках |
| SELECT CONSUMER.FULL\_NAME, COMMODITY.NAME\_OF\_COM, PURCHASE.DATE\_OF\_PURCHASE,PURCHASE.DATE\_OF\_RECEIPT  FROM (PURCHASE INNER JOIN CONSUMER ON CONSUMER.EMAIL=PURCHASE.EMAIL) INNER JOIN COMMODITY ON PURCHASE.CODE\_OF\_COM=COMMODITY.CODE\_OF\_COM |

1. Руководство пользователя

Приложение использует оконное приложение для взаимодействия с пользователем. Запуск осуществляется через файл OnlineShop(CourseWork).exe



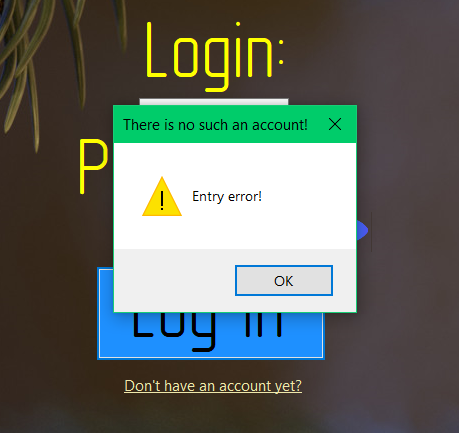
Для большего удобства и эстетической удовлетворенности был создан ярлык для этого .exe файла в рабочем столе с соответствующей иконкой.



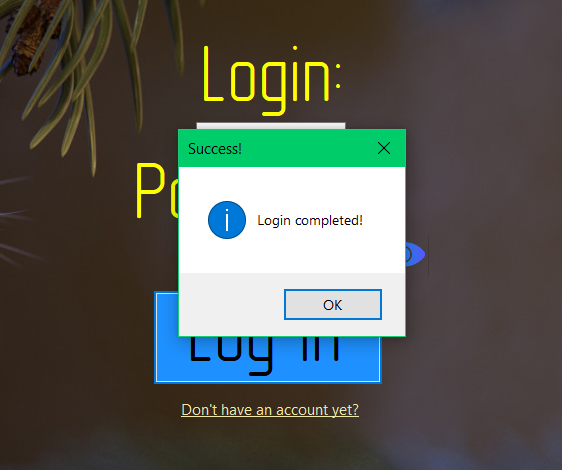
При запуске программы перед пользователем предстанет окно авторизации. Авторизация необходима для избежания несанкционированного доступа к данным пользователей нашего онлайн-магазина.



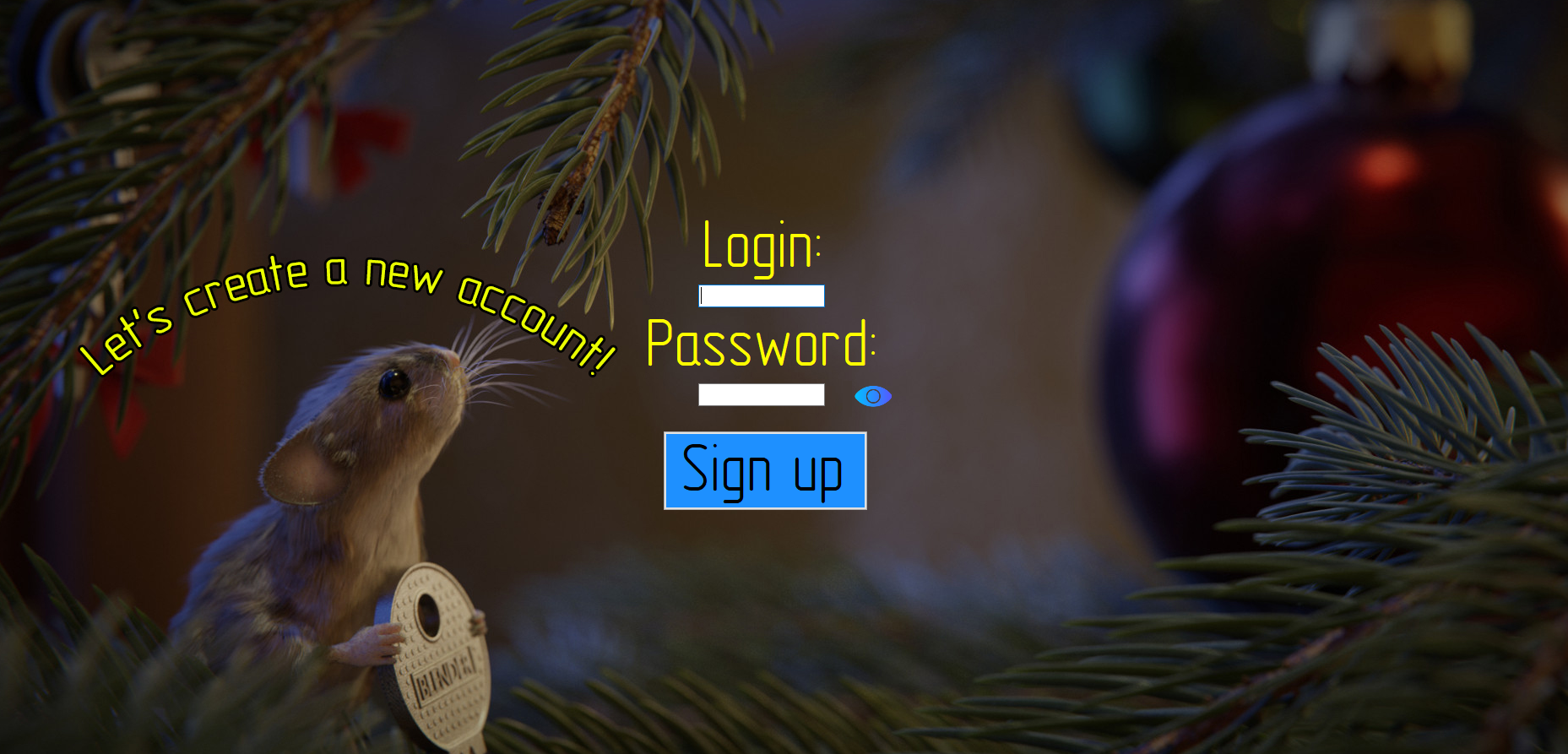
Если данные введены неверно:



Если данные введены верно:



В этом же окне имеется гиперссылка на окно для создания нового пользователя АИС.



После успешного входа нас встречает основное окно для работы с данными:

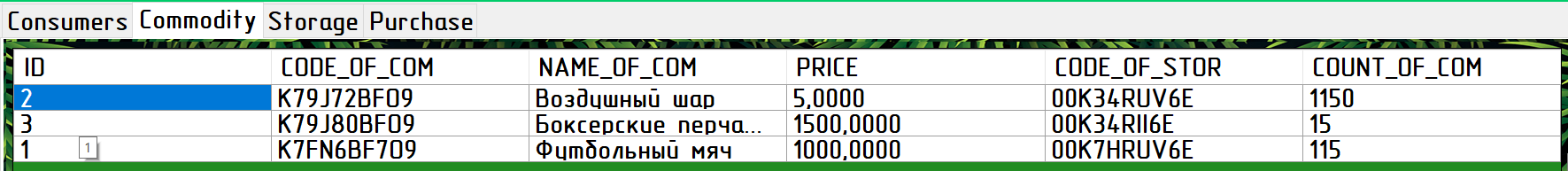


С помощью вкладок можно осуществлять переход между таблицами

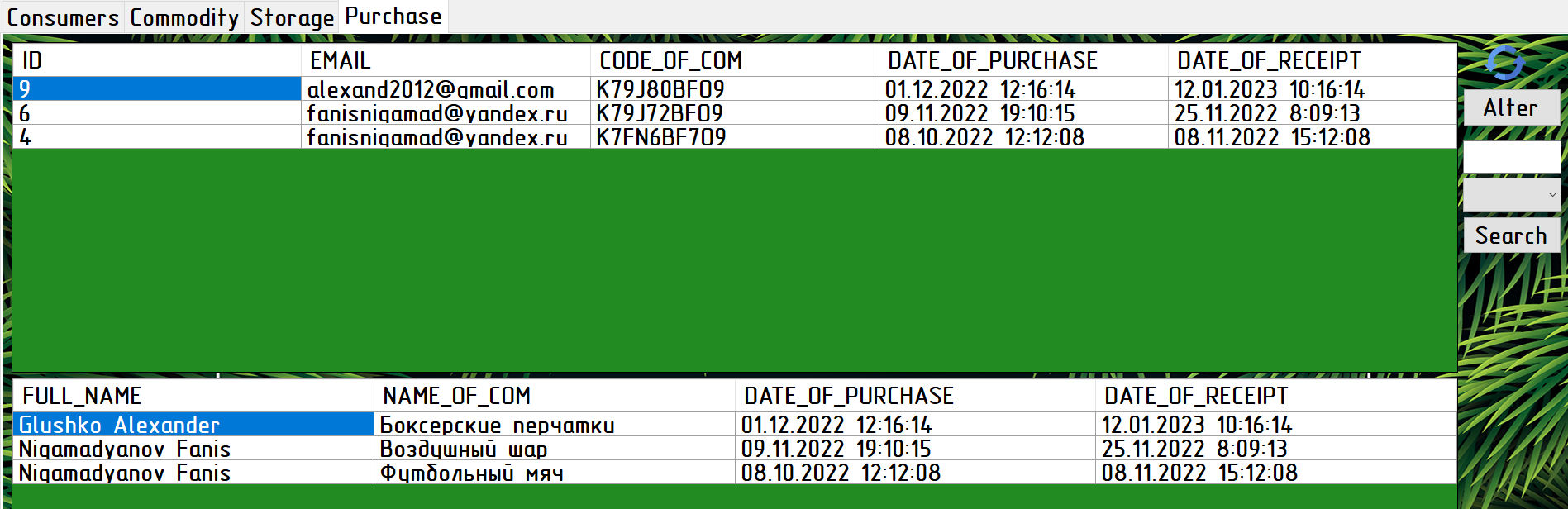


В каждой вкладке будет таблица с данными и возможные действия с этой таблицей (добавление, удаление, изменение обновление и поиск)



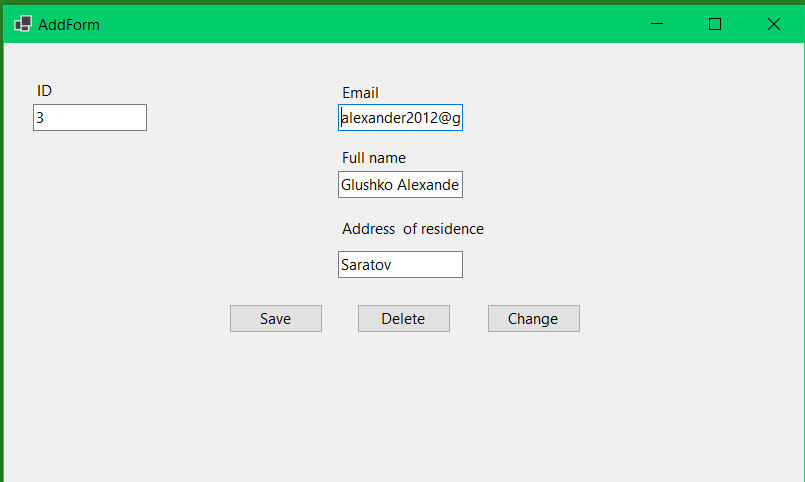




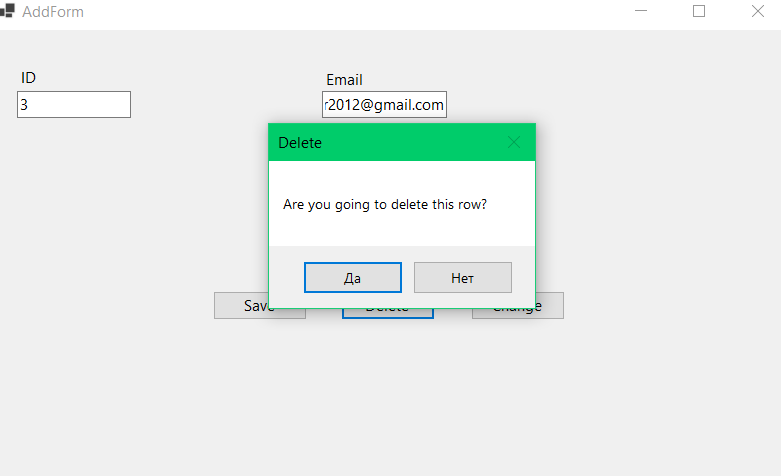


Рассмотрим всевозможные действия с таблицей пользователей онлайн-магазина.

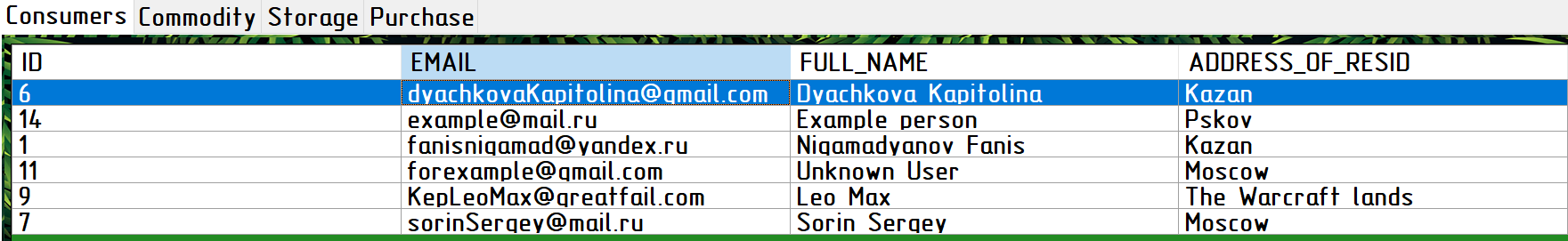
Для удаления необходимой строки нужно нажать на нее:



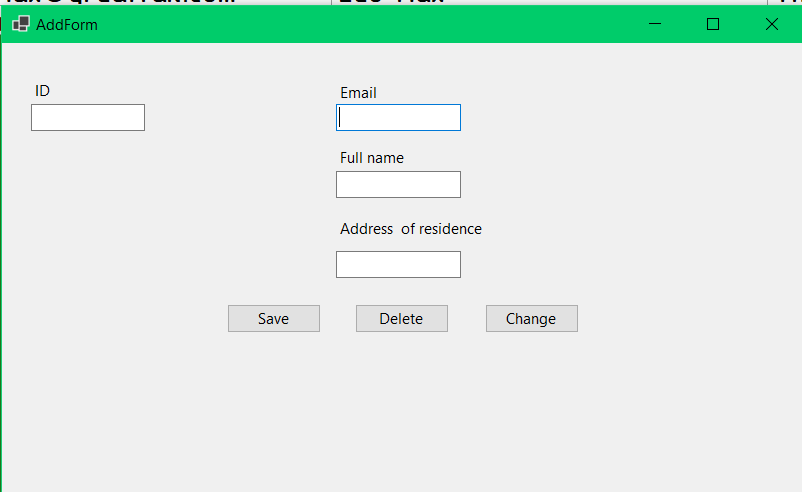
Таким образом наши текстовые поля нового окна заполнятся автоматически. Далее нажимаем кнопку “Delete”.



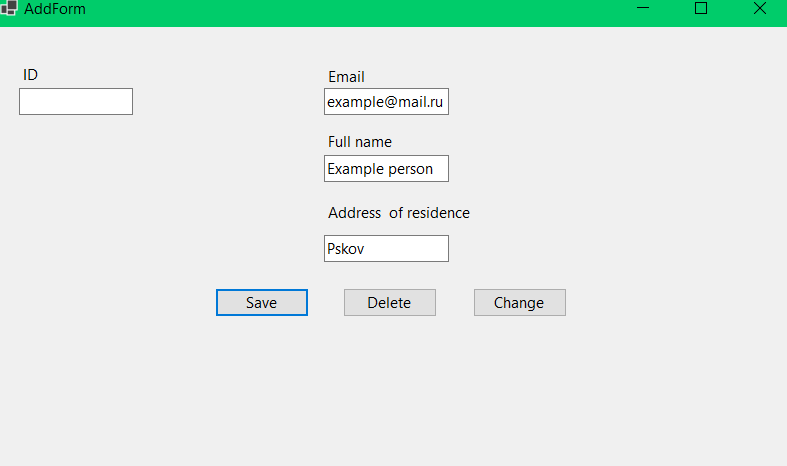
Результат:

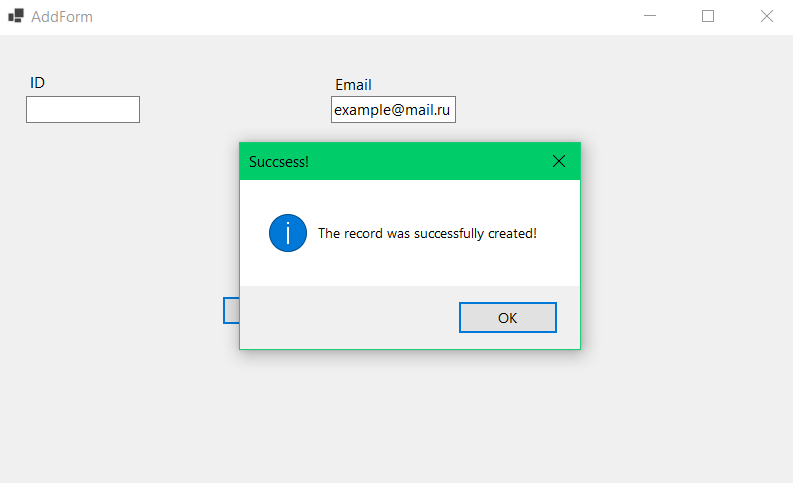


Для добавления новой строки нам необходимо нажать кнопку “Alter”.

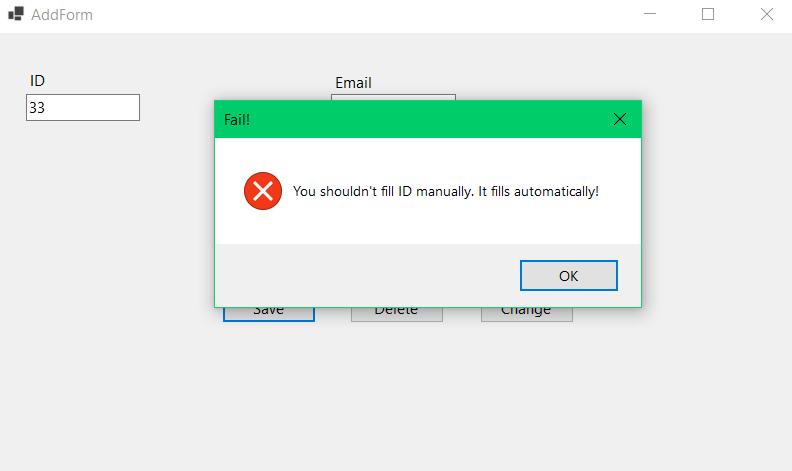


Как вы можете заметить, это окно идентично окну для удаления. Нажатием кнопки “Alter” можно, вручную заполнив, удалить интересующего пользователя с нашей базы данных. Однако это неудобно, поэтому рекомендуется удалять и изменять (изменение данных см. ниже) данные кликом по соответствующей строки.

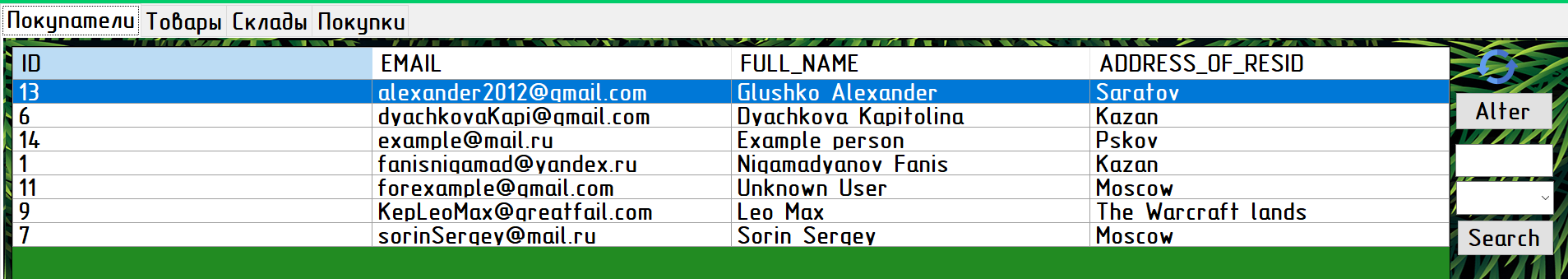


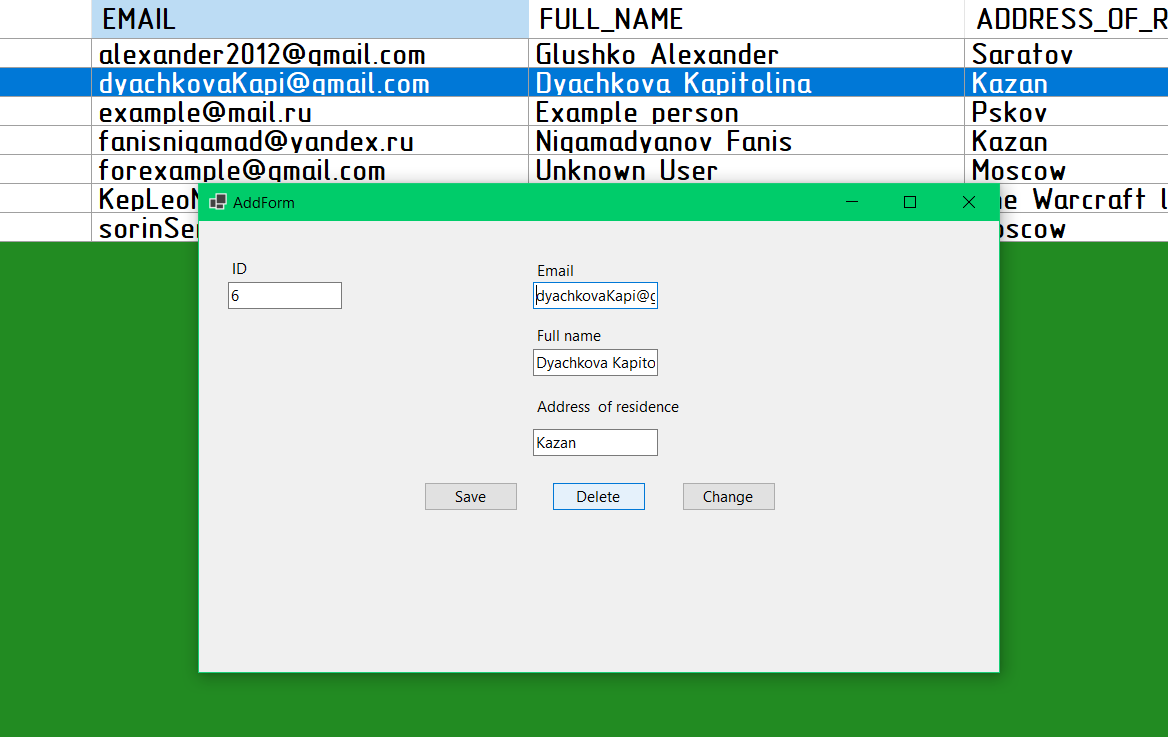


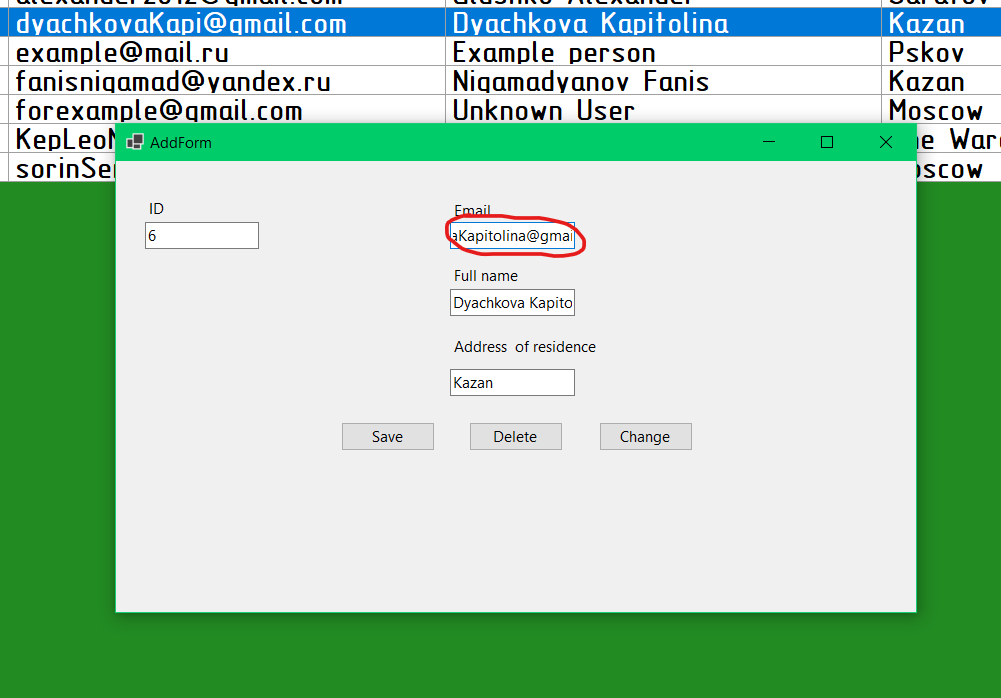
Обратите внимание, что при создании пользователя текстовое поле ID не заполняется, т.к. это автоматический счетчик для избежания повторов этого поля. Если вы проигнорируете это требование, появится следующая ошибка:

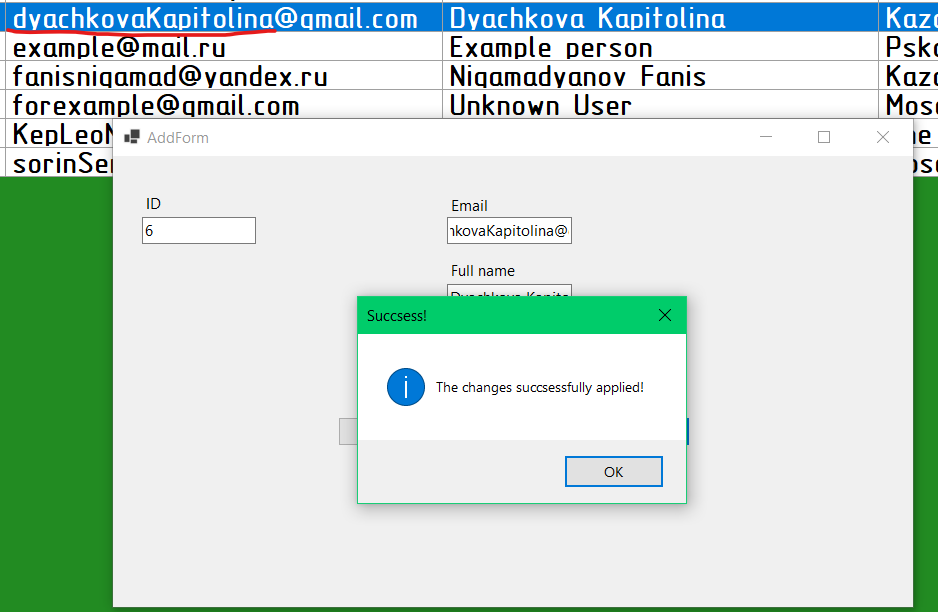


Для изменения данных пользователя нужно, как и при удалении, нажать на интересующую вас строку, а затем изменить все необходимые поля КРОМЕ поля ID, т.к. сверка командой WHERE в запросе к базе данных идет именно по ней. Это условие касается всех таблиц.

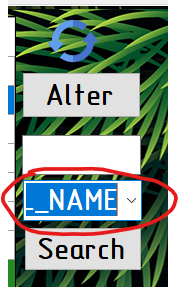




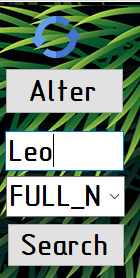




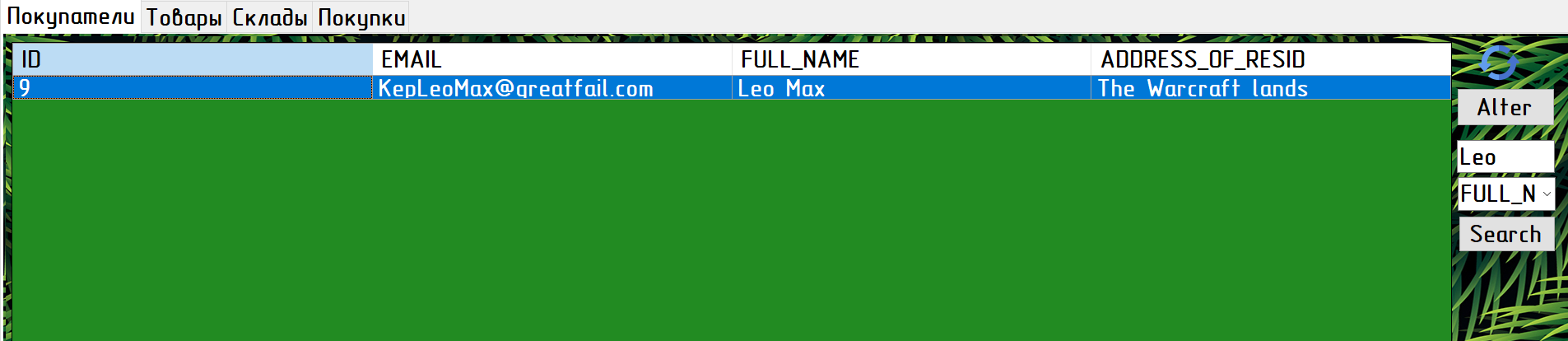
Рассмотрим поиск по таблице. Сначала нам нужно выбрать поле, по которому мы хотим фильтровать информацию. Для этого выбираем нужную строку в comboBox:



Затем заполняем textBox информацией для поиска и нажимаем на кнопку “Search”:

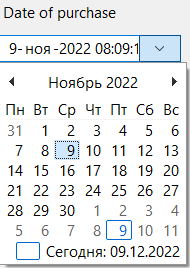


Результат:

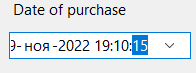


Рассмотрим некоторые нюансы для таблицы Purchase, а именно работу с объектами DateTimePicker:

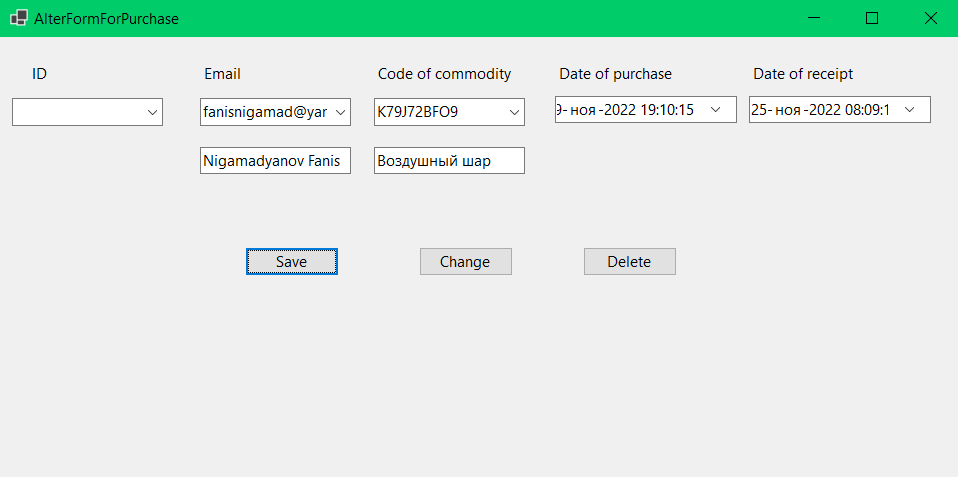
В DateTimePicker сначала выберем дату:



А затем прописываем время покупки товара пользователем:

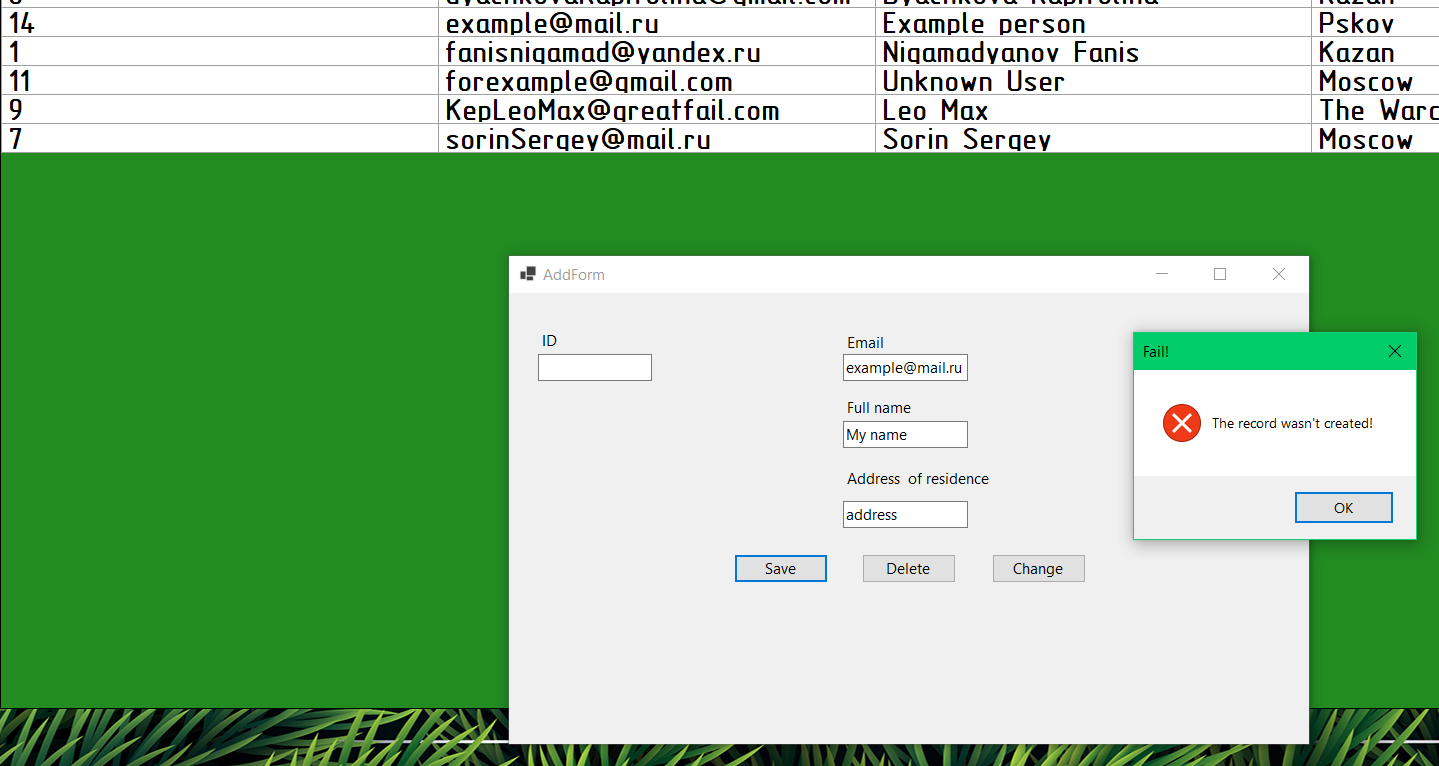


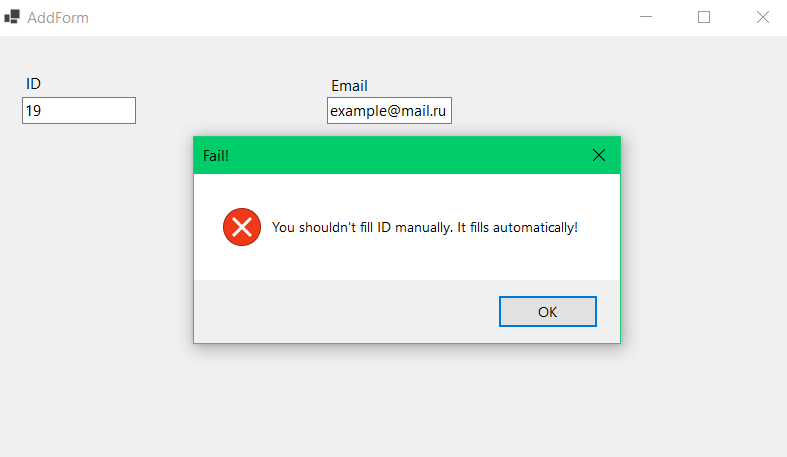
Вкупе получаем следующее:



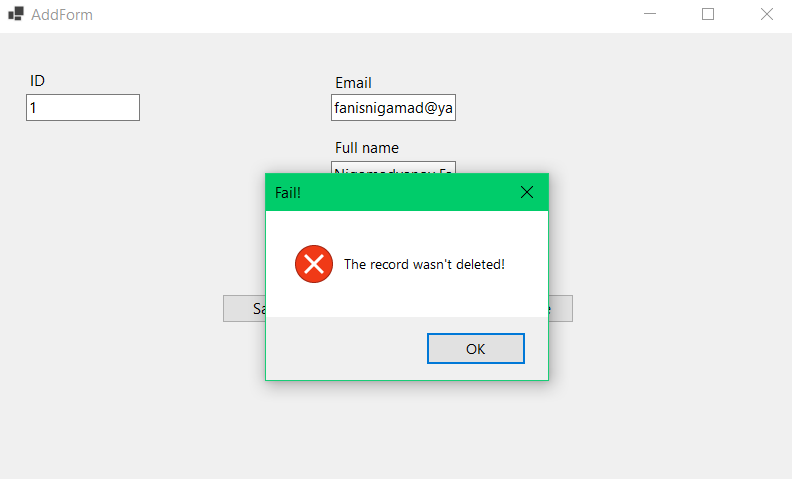
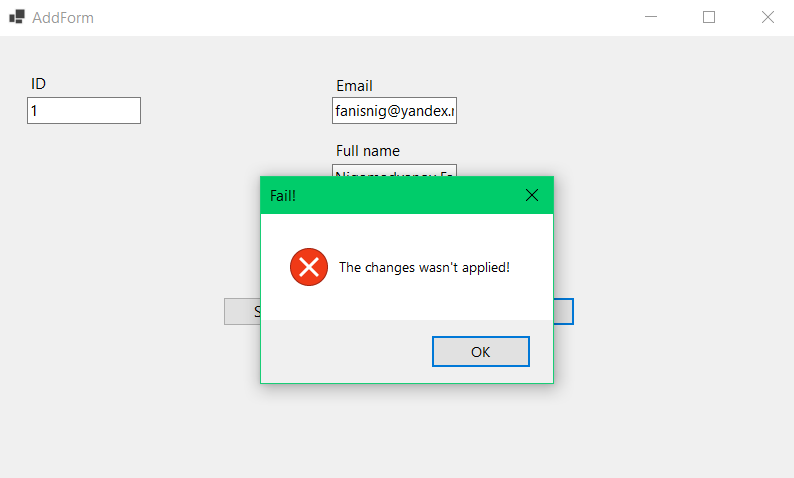
Обработка исключений.

Для таблицы покупателей:

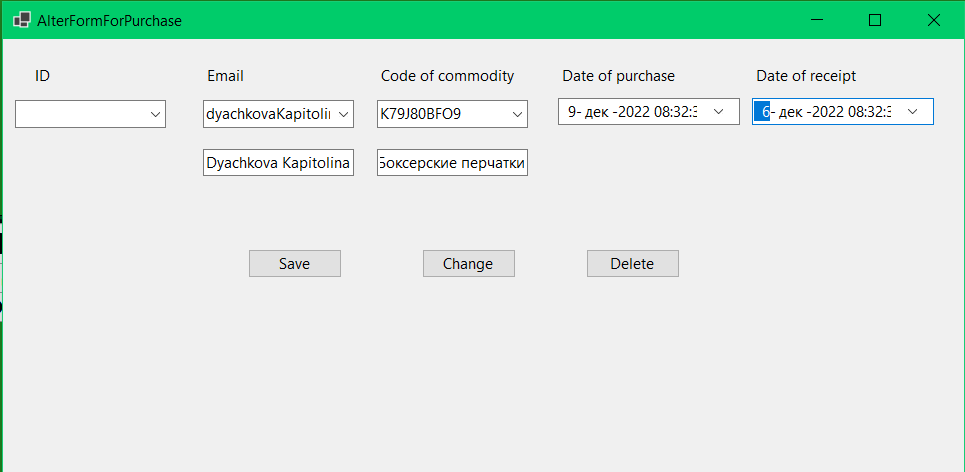
* 1. Добавление пользователя с уже существующей почтой  
     
  2. Заполнено поле ID

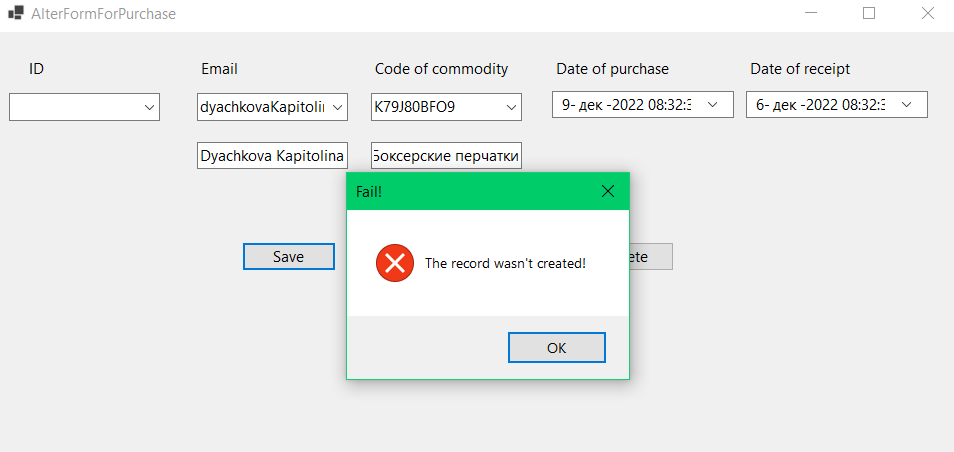


* 1. Попытка удалить/изменить ключевое поле пользователя, участвующего в других таблицах в качестве родителя

Для остальных таблиц все выглядит подобным образом, но в таблице Purchase есть дополнительное условие того, что дата получения должна быть больше даты покупки:





# Руководство программиста

Для разработки проекта была использована среда разработки программного обеспечения Microsoft Visual Studio 2019.

Весь программный код, включая основную программу, хранится в файлах .cs. Проект связан с сохраненной базой данных, разработанной на СУБД «Microsoft Sql Server», имеющей расширение .sql.

Было разработано 7 классов-форм, 2 обычных класса:

* 1. Program.cs - точка входа в программу. Осуществляет запуск окна регистрации
  2. ShopDataBase.cs - класс, осуществляющий подключение к базе данных
  3. Sign\_in.cs - класс-форма для авторизации пользователя
  4. Sign\_up.cs - класс-форма для регистрации пользователя
  5. OnlineShop.cs - основная класс-форма, отображающая данные
  6. AlterFormForConsumer.cs - окно для изменения данных пользователей онлайн-магазина
  7. AlterFormForCommodity.cs - окно для изменения данных о товарах
  8. AlterFormForStorage.cs - окно для изменения данных о складах
  9. AlterFormForPurchase.cs - окно для изменения данных о покупках

## Класс Program.cs

| Название метода | Описание метода |
| --- | --- |
| static void Main() | Точка входа в программу. Осуществляется запуск окна авторизации |

## Класс ShopDataBase.cs

| Название метода | Описание метода |
| --- | --- |
| public void openConnection() | Метод открывает соединение с базой данных |
| public void closeConnection() | Метод закрывает соединение с базой данных |
| public SqlConnection getConnection() | Метод возвращает соединение |

## Класс Sign\_in.cs

| Название метода | Описание метода |
| --- | --- |
| public Sign\_In() | Конструктор данного класса. Инициализация таблицы и настройка ее свойств. |
| private void Sign\_In\_Load(object sender, EventArgs e) | При загрузке окна PictureBox closedEye становится невидимым, и textBox для пароля будет заполнятся скрывающими пароль символами. |
| private void log\_in\_button\_Click(object sender, EventArgs e) | Событие при нажатии кнопки “Log in”. Проверяет наличие пользователя с таким логином и паролем в базе данных. |
| private void openedEye\_Click(object sender, EventArgs e) | Событие при нажатии PictureBox openedEye. При нажатии этот PictureBox скрывается и пароль раскрывается. Отсюда следует, что PicteureBox-ы openedEye и closedEye не отражают состояние, а отображают действие, которое произойдет при нажатии на них |
| private void closedEye\_Click(object sender, EventArgs e) | Событие, обратное событию выше. |
| private void linkLabel1\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e) | Событие при нажатии на гиперссылку. Открывает новое окно для регистрации |

## Класс Sign\_up.cs

| Название метода | Описание метода |
| --- | --- |
| public Sign\_up() | Конструктор данного класса. Инициализация таблицы и настройка ее свойств. |
| private void Sign\_up\_Load(object sender, EventArgs e) | См. класс выше |
| private void openedEye\_Click(object sender, EventArgs e) | См. класс выше |
| private void closedEye\_Click(object sender, EventArgs e) | См. класс выше |
| private void sign\_up\_button\_Click(object sender, EventArgs e) | Вставка в базу данных новых данных об админах. Данные считываются |
| private bool checkUser() | Проверка, имеется ли уже такой пользователь. Если так, то пользователь не будет добавлен |

## Класс OnlineShop.cs

| Название метода | Описание метода |
| --- | --- |
| public OnlineShop() | Инициализация формы. Автоматическая настройка размеров ячеек под окно DataGridView |
| private void CreateColumnsOfConsumer() | Добавление колонок в DataGridView таблицы покупателей |
| private void ReadSingleRowOfConsumer(DataGridView dgv, IDataRecord record) | Считывание одной строки. |
| public void RefreshDataGridOfConsumer(DataGridView dgv) | Обновление DataGridView, заполнение с базы данных. |
| private void addRecord1\_Click(object sender, EventArgs e) | Нажата кнопка “Alter” для страницы покупатель. Появляется окно AlterFormForConsumer |
| private void UpdateButton1\_Click(object sender, EventArgs e) | Если была нажата кнопка обновления, то в DataGridView отобразится актуальная информация с базы данных |
| private void dataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e) | Если была нажата какая-либо строка объекта DataGridView, то появится окно AlterFormForConsumer, заполненная данными этой строки. |
| private void SearchButton1\_Click(object sender, EventArgs e) | Если нажата кнопка поиска, то в DataGridView отобразится информация, фильтрованная по полю, выбранному в comboBox, и по строке, введенной в textBox. |
| … | … |

Методы, предназначенные для оставшихся таблиц, идентичны представленным методам.

## Класс AlterFormForConsumer.cs

| Название метода | Описание метода |
| --- | --- |
| public AlterFormForConsumer(OnlineShop onlineShop) | Инициализация формы. |
| private void SaveButton\_Click(object sender, EventArgs e) | Сохраняет данные, заполненные в форме, в базе данных. |
| private void deleteRow() | Удаляет нужную строку с DataGridView, но не с базы данных. |
| private void IsRowForDelete() | Удаляет строку с базы данных |
| private void DelButton1\_Click(object sender, EventArgs e) | Нажата кнопка “Delete”. Появляется окно для подтверждения удаления. |
| private void IsRowForChange() | Изменяет данные в базе данных |
| private void Change() | Изменяет данные в DataGridView |
| private void ChangeButton1\_Click(object sender, EventArgs e) | Выполняет два метода сверху, если была нажата кнопка “Change”. |

Для оставшихся классов с префиксом “AlterFormFor” описания методов те же, поэтому не целесообразно их описывать снова.

# Заключение

В ходе данной курсовой работы было создано приложение для автоматизации работы администратора онлайн-магазина.

Разработанное приложение отвечает всем требованиям предметной области, таблицы созданной базы данных отвечают требованиям нормализации, что позволяет обеспечить целостность и непротиворечивость информации.

# Литература

1. Альманах программиста. Том 1: Microsoft ADO.NET, Microsoft SQL Server, доступ к данным из приложений, 2003, Ю.Е.Купцевич.

2. Практическое использование ADO.NET. Доступ к данным в Internet, 2003, Вилдермьюс Ш.

3. Руководство по ADO.NET и работе с базами данных в .NET 6: [Электронный ресурс]. URL:<https://metanit.com/sharp/adonetcore/> (Дата обращения: 01.10.2022).