|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ | | | | | | | | | | |
|  | | | |  | | |  | | | |
| ФГБОУ ВПО «Пермский государственный национальный исследовательский университет» | | | | | | | | | | |
|  | |  | | | | | | |  | |
|  | | ОТЧЕТ  по лабораторной работе «Автоматизированное рабочее место» по дисциплине «Учебная практика по базам данных» | | | | | | |  | |
|  | | |  | | | | |  | | |
|  | Работу выполнил  студент гр. ПМИ-1,2  Васильевых М.Г. \_\_\_\_\_\_\_\_\_\_  (подпись)  «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2021 | | | |  | Проверил  ассистент кафедры МОВС  Турова И.А. \_\_\_\_\_\_\_\_\_\_  (подпись)  «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2021 | | | |  |
|  | |  | | | | | | |  | |
|  | | Пермь 2021 | | | | | | |  | |

Содержание

[1 Постановка задачи 3](#_Toc83312391)

[2 Описание предметной области и базы данных 4](#_Toc83312392)

[3 Описание функционала разработанной программы 5](#_Toc83312393)

[3.1. Внешний вид и функционал приложения 5](#_Toc83312394)

[3.2 Общий алгоритм решения задачи 7](#_Toc83312395)

[4 Реализация программы 8](#_Toc83312396)

[5 Тестирование программы 13](#_Toc83312397)

# 1 Постановка задачи

Написать программу, которая на графическом интерфейсе пользователя позволяет просматривать, добавлять, удалять, изменять записи и осуществлять поиск минимум двух стандартных запросов в 4 связанных таблицах из спроектированной ранее базы данных предметной области без необходимости задания и отображения пользователю искусственных ключей. Между указанными таблицами должно присутствовать как отношение типа 1:М, так и отношение типа М:М.

При запуске программы пользователю должна отображаться форма аутентификации по логину и паролю, допускающая гостевой вход. Из формы аутентификации должна быть возможность перейти на форму регистрации нового логина и пароля для оператора АРМ.

Пользователь, зашедший в приложение под учётной записью администратора АРМ, должен иметь возможность просматривать, добавлять, удалять, изменять записи во всех используемых таблицах из спроектированной ранее базы данных предметной области и в таблице учётных записей пользователей. Пользователь, зашедший в приложение под учётной записью оператора АРМ, должен иметь возможность просматривать, добавлять, удалять, изменять записи только во всех используемых таблицах из спроектированной ранее базы данных предметной области. Пользователь, зашедший в приложение под гостевой учётной записью АРМ, должен иметь возможность только просматривать записи во всех используемых таблицах из спроектированной ранее базы данных предметной области.

# 2 Описание предметной области и базы данных

Для работы приложения нам таблицы «Поставщики», «Банки», «Поставки», «Продукты», «Пользователи» из базы данных «Поставки» - рисунок 1.1

Таблица «Поставки» реализует связь М:М, таблица «Поставщики» реализует связь 1:М

Таблица «Пользователи» предназначена для аутентификации пользователей АРМ.

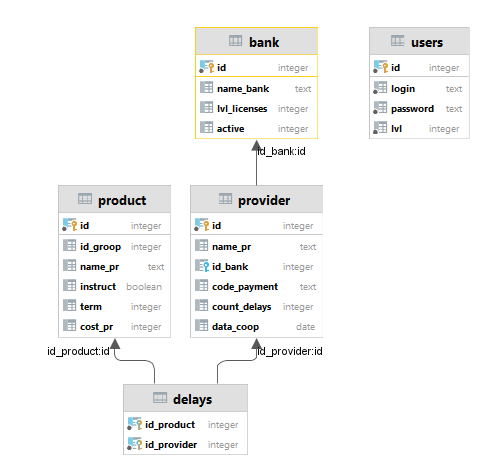
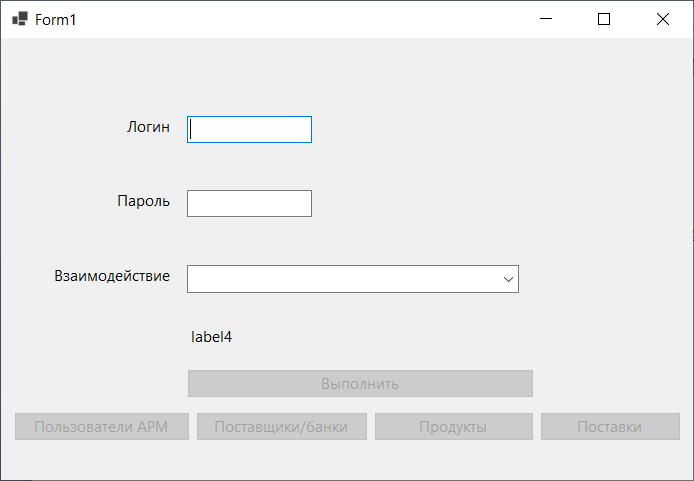


Рисунок 1.1 – Используемая БД

# 3 Описание функционала разработанной программы

3.1. Внешний вид и функционал приложения

При запуске приложение, появляется главная форма, форма аутентификации и регистрации. 

На ней есть 2 текстбокса для ввода значения пароля и логина

Лэйбл для вывода текущего состояния операции.

Комбобокс для выбора какие действия выполнить:

1)Вход как аутентифицированный пользователь

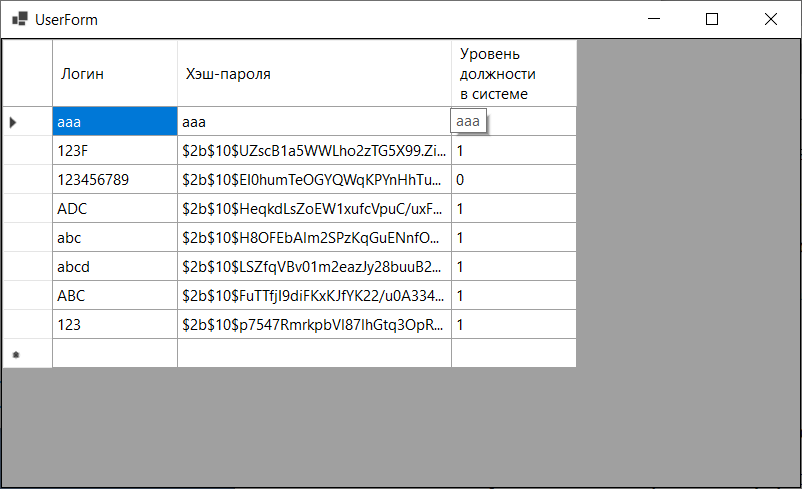
2)Вход как гость

3)Регистрация

Дальше имеются кнопка выполнения одного из перечисленных действий

И так же идут кнопки для перехода на формы управления данными.

1)Кнопка «Пользователи АРМ» переводит нас на форму для работы с записями таблицы пользователи.

****

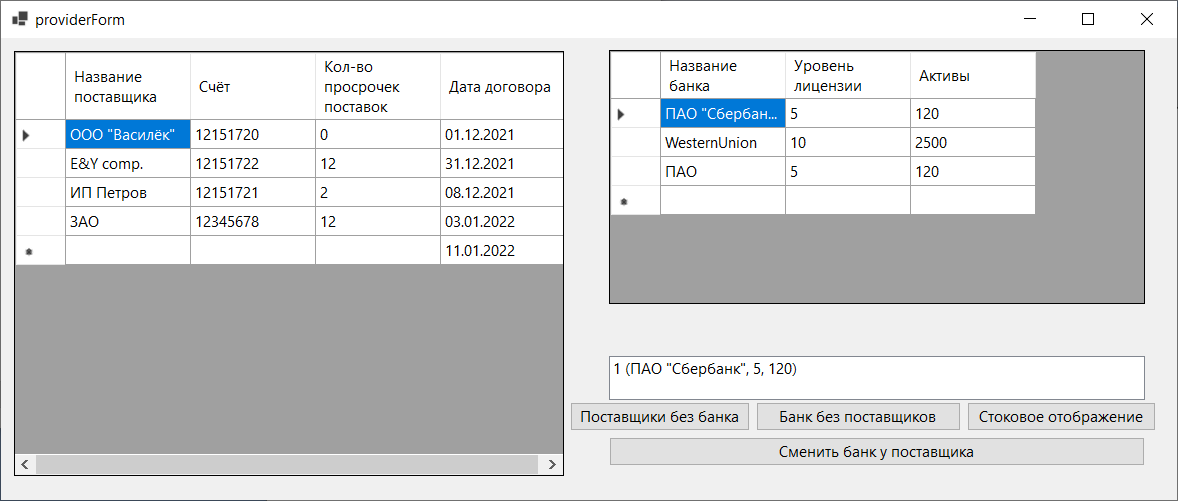
На ней расположен DataGridView для отображения записей в таблице

Было принято ряд условий для отображений данных:

1)Ключи не демонстрируются

2)Пароли демонстрируются в виде хэшей, что бы даже администратор АРМ не мог иметь доступ к чужим записям.

2) Кнопка «Поставщики банки»

****

На ней имеются два объекта, DataGridView. Один отображает все записи поставщиков, второй все записи банков.

Для реализации связи один ко многим был реализован листбокс, который отображает банк обслуживающий выбранного поставщика.

И 3 кнопки для реализации стандартных запросов.

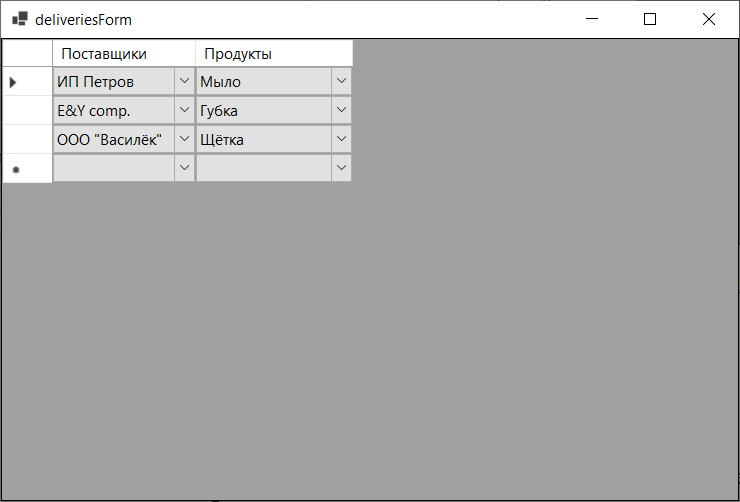
И одна кнопка для смены банка у поставщика.

3)Кнопка «Продукты»

****

Здесь представлен один обьект DataGridView, в который выводятся записи из таблицы «Продукты».

4)Кнопка «Поставки»



Здесь представлен один объект DataGridView с ячейками в виде комбобоксов, в который выводятся записи из таблицы «Поставки».

Здесь было принята условность, что название объекта строго определяет весь остальной кортеж записи.

## 3.2 Общий алгоритм решения задачи

Данная задача подразумевает общие решения для определенных задач, поэтому часть алгоритма я обобщу.

Для начала нам стоит определить операции для работы с DataGridView:

1. Отображение: Для начала нам нужно создать запрос в БД на получение данных из 1 таблицы, где мы отобразим все поля кроме ключей. Так же нам необходимо понять какие именно записи банков мы должны выводить, я остановился на варианте, что мы выводим банка выделенной строки в 1 элементе. По условию нам сказано не выводить пользователю искусственные ключи, мы будем их хранить в элементе только в сокрытом виде.
2. Добавление, изменение: Так как подразумевается, что элемент DataGridView обрабатывает только изменение ячейки, то будем вводить маркер для определения того, что именно мы собираемся делать. А именно если в обрабатываемой строке есть id, то будем считать, что мы изменяем поле, иначе мы его добавляем. И в соответствии с этими правилами будем запускать тот или иной запрос, в соответствии с правилами, установленными в БД. Если же мы передумали и отказались от изменений в ячейке, то нам необходимо ввести обработку события для отмены изменений на форме и создания запроса в БД.
3. Удаление: Так как элемент имеет событие для удаления строки из таблицы, мы можем, опустит рассмотрения удаления из самой формы, упомянув, что нам необходимо просто отправить запрос в БД на удаление записи.

Работа со связями 1:М

Так как один банк может обслуживать много поставщиков, будет удобнее отображать связь через поставщик и его банк.

И так же работать с связями тк мы будем раскладывать всю связь 1:м на 1:1, что на много более просто реализуется.

# 4 Реализация программы

Сначала необходимо создать с помощью Entity Framework модель данных из БД(ORM)

ORM — технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

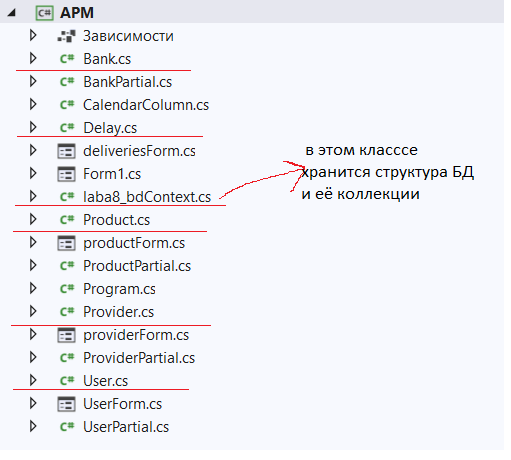
Создадим модель ОРМ в нашем проекте, вызвав команды в консоли

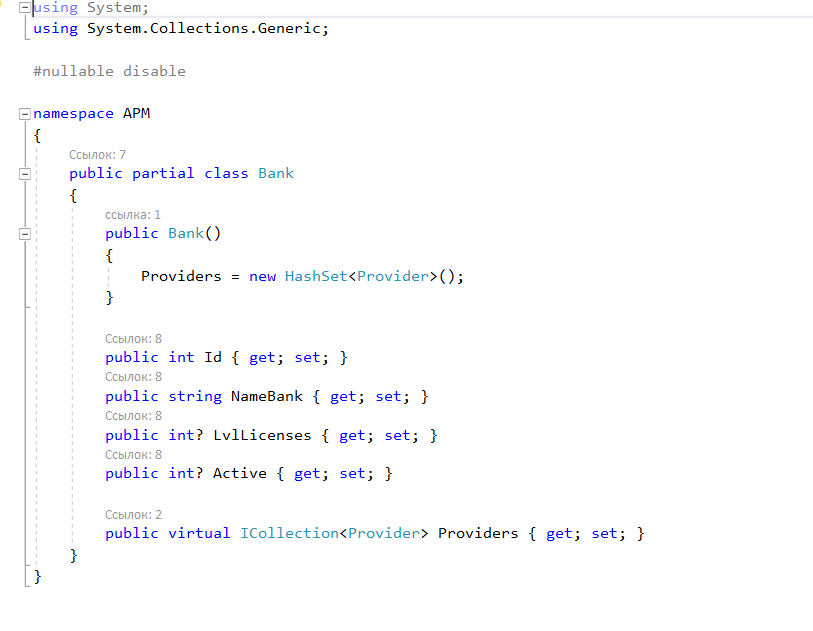
**Scaffold-DbContext "Host=localhost;Port=5432;Database=usersdb;Username=postgres;Password=password"**

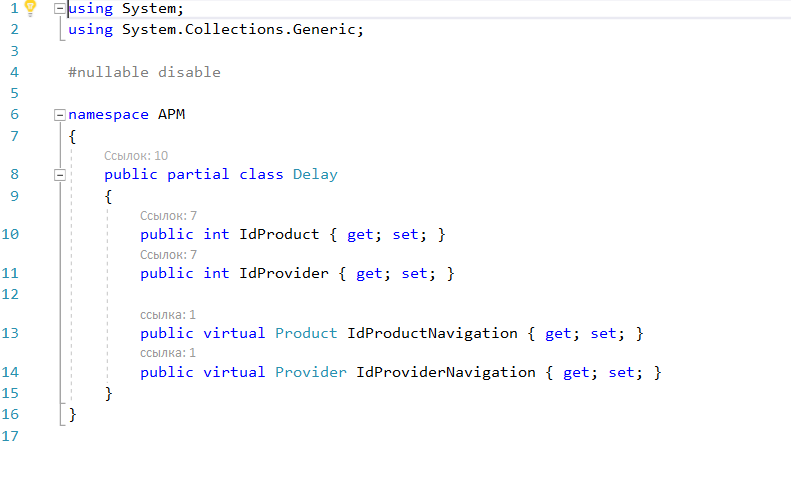
И инициализация провайдера БД

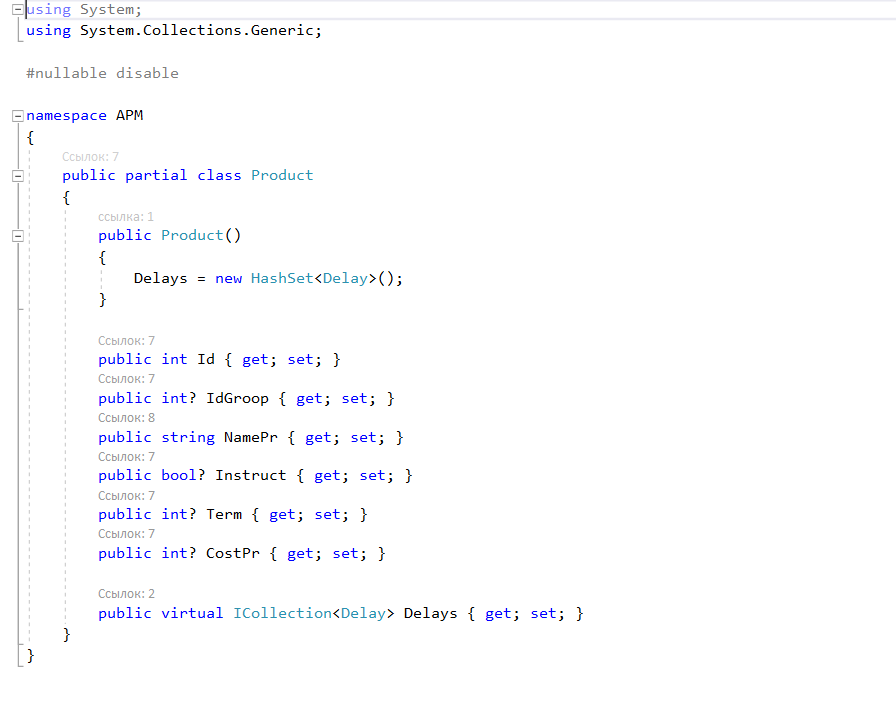
**Npgsql.EntityFrameworkCore.PostgreSQL**

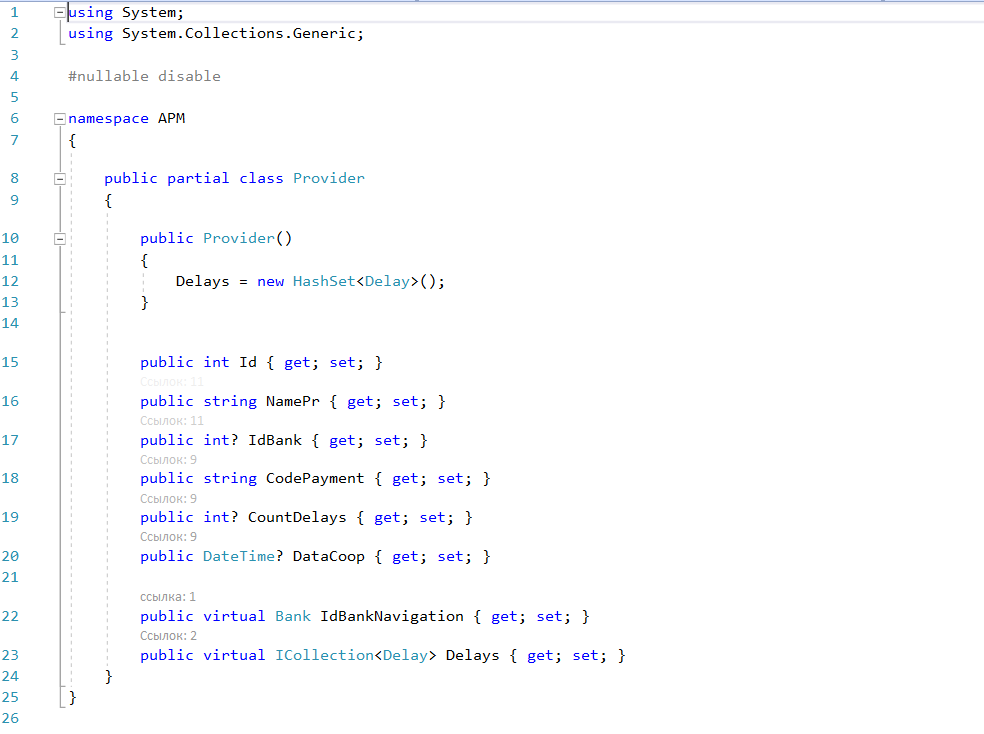
После выполнения этих команд мы получаем набор классов соответствующих таблицам из БД, где столбец таблицы это атрибут класса, а строка это элемент коллекции соответствующей таблицы.

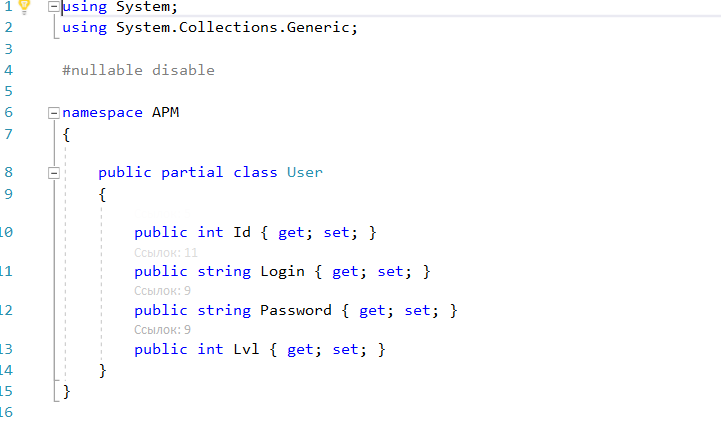


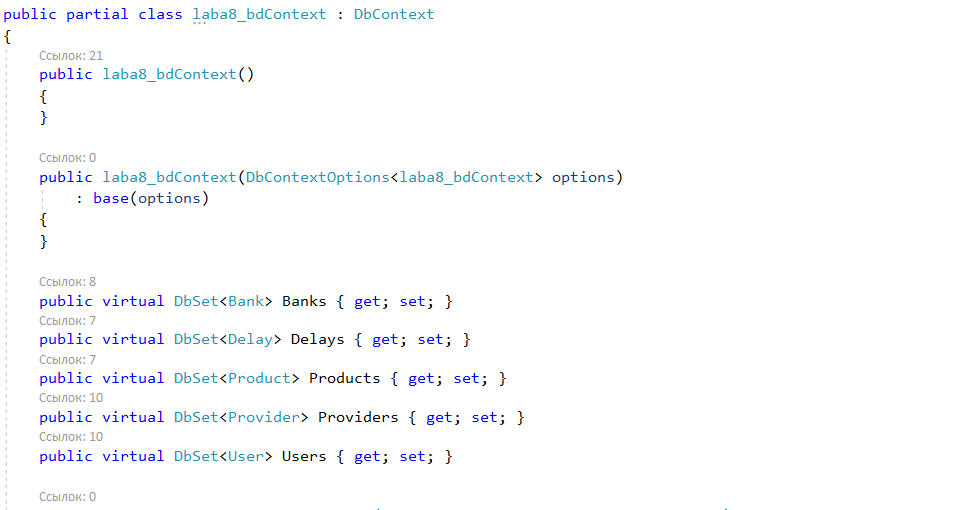












Для работы с коллекциями используется язык Linq.

Он в некоторой степени похож на язык sql но с некоторыми особенностями.

db.Products.Find(newid) // поиск значения по ключу

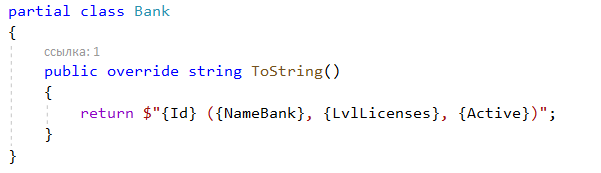
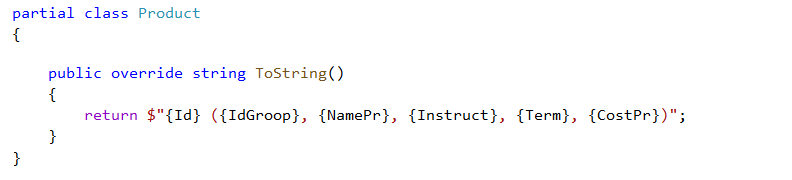
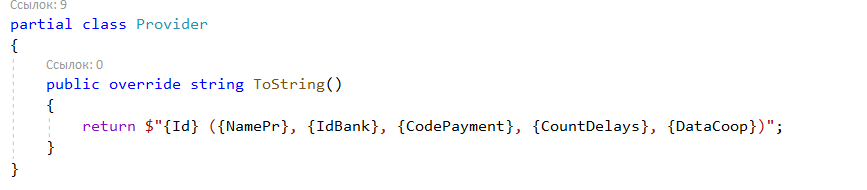
 поиск по определенному значению поля

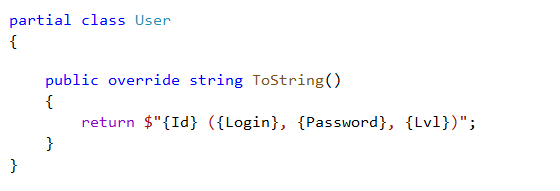
 сохранение изменений в БД

 удаление определенной записи из БД.

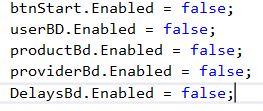
 Добавление определенной записи в БД.

И перегрузить метод toString для каждого класса



Аутентификация и регистрация



Кнопки доступа к таблицам заблокированы пока пользователь не войдёт.

Для того что бы понимать, а какая сейчас роль активна я добавил поле user



Изначально будем иметь 3 роли:

* Администратор АРМ, имеет доступ ко всем таблицам (чтение, редактирование)
* Пользователь, имеет доступ ко всем таблицам (чтение, редактирование) кроме, «пользователи»
* Гость, может смотреть данные во всех таблицах, кроме «пользователи», так как и остальные роли имеет доступ к стандартным запросам. В таблице «пользователи» никак не отображен.

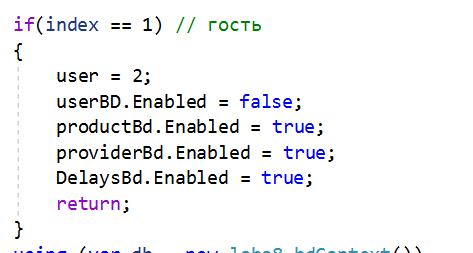
Будем считать, что мы можем создать только рядового пользователя.

Сначала нам нужно получить индекс выбранного метода работы



Затем исходя из того что было выбрано действовать.

Если был выбран гостевой вход



В остальных случаях нам необходимо обращаться к БД.

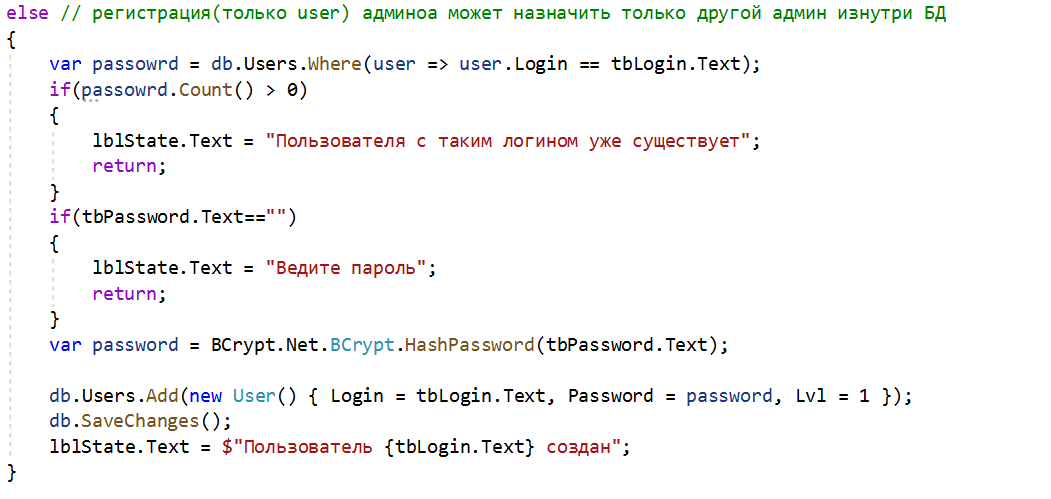
Создаем подключение к данным





Так как мы используем шифрование паролей то нам необходимо проверять не соответствие строк а соответствие хэш-функций.

Затем мы проверяем наличие такого логина в БД. Если все условия выполнены мы заполняем поле user в соответствии с правами пользователя и продолжаем работу.



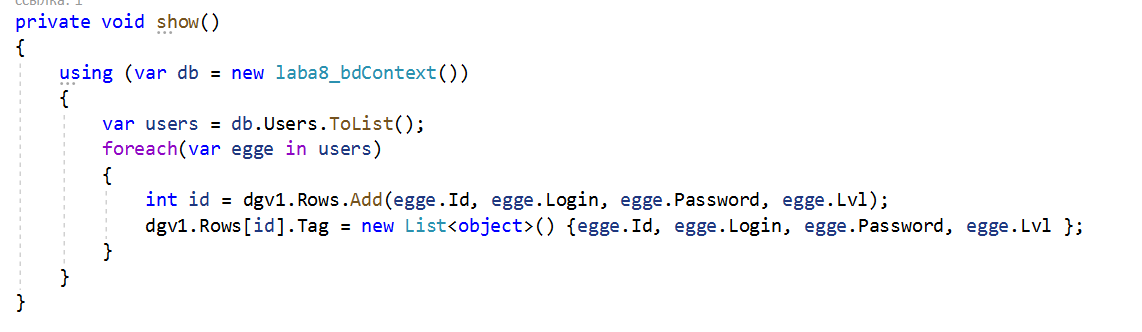
Но если мы входим то нам необходимо сначала проверить наличие такого логина в БД, а затем хэшировать пароль и вносить в модель и отправлять изменения в БД.

Работа с кнопками подразумевает запуск соответствующей формы.

Работа с таблицей «пользователи»

Так как параметр уровня пользователя наследуется всеми формами то проблем с работой с данными нет. 

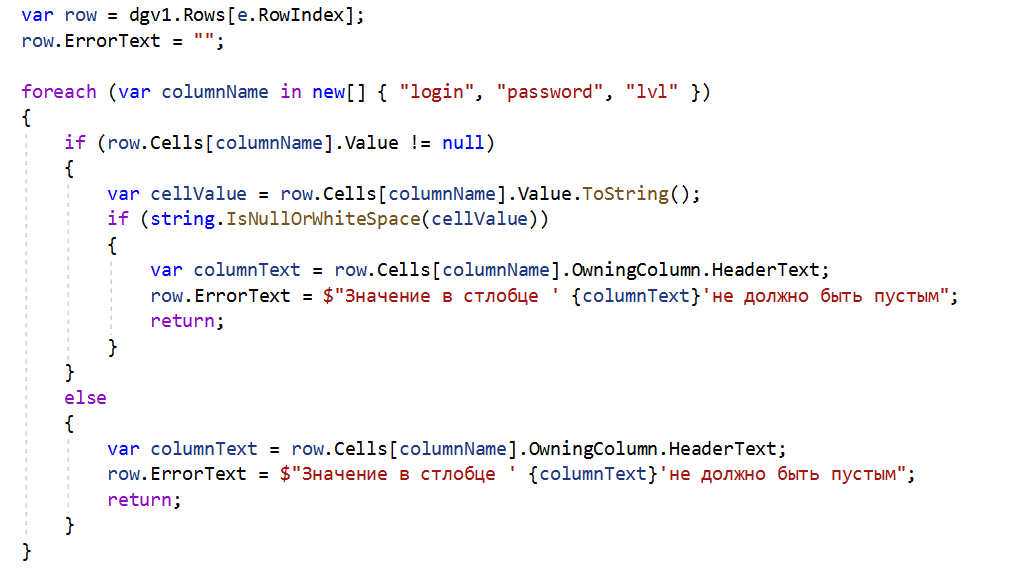
Создаем структуру датагрида.



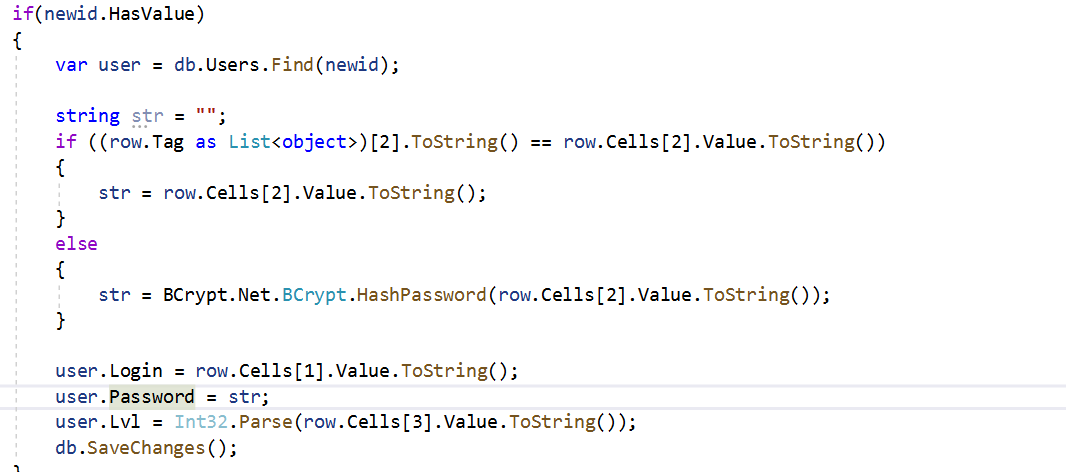
Заполняем его поля( в теге записи используется list<obbjets>)

Изменение данных

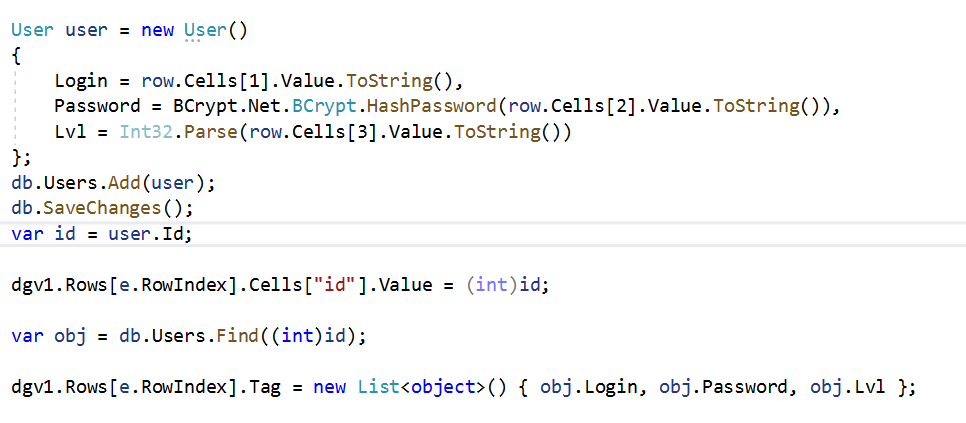
Проверка на пустоту полей

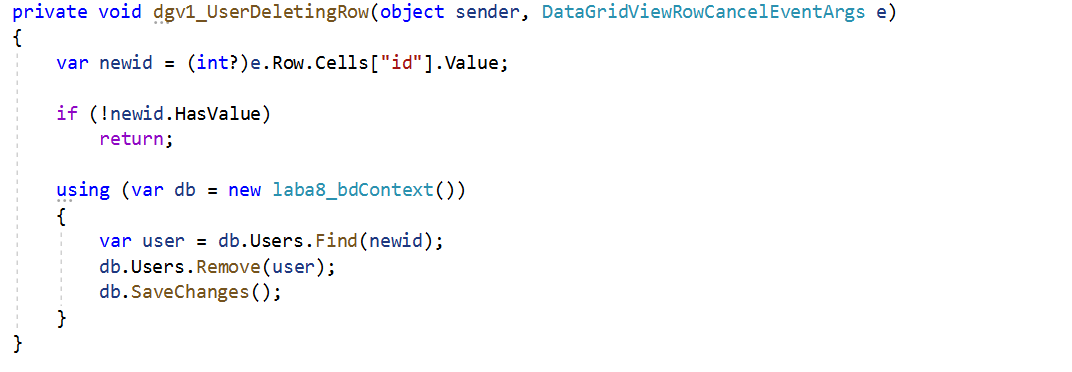


Получение id текущей записи

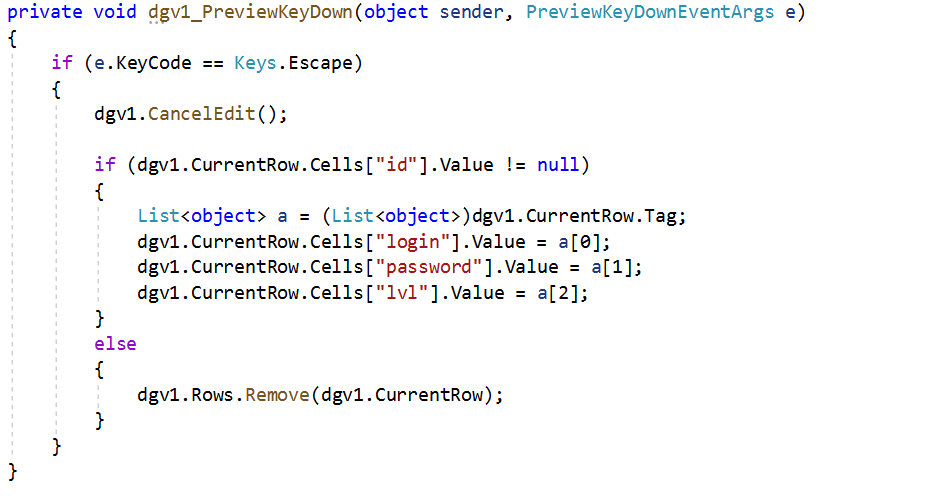


Изменение данных

Добавление новой записи



Удаление записи



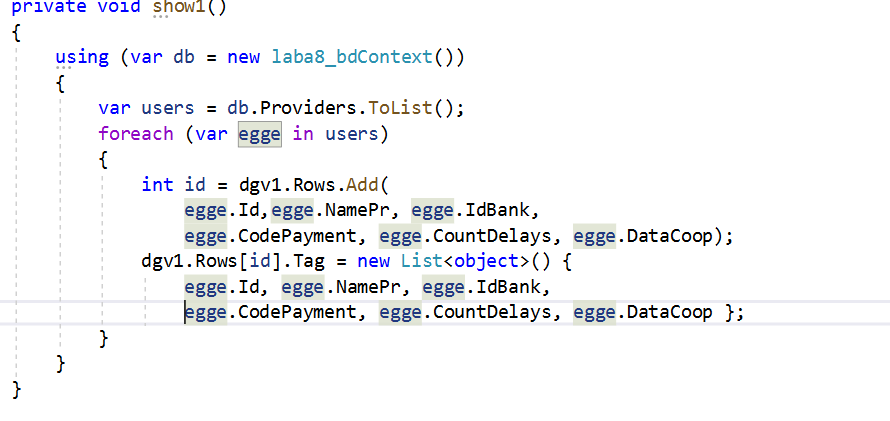
Откат изменений

Работа с таблицей «Поставщики»

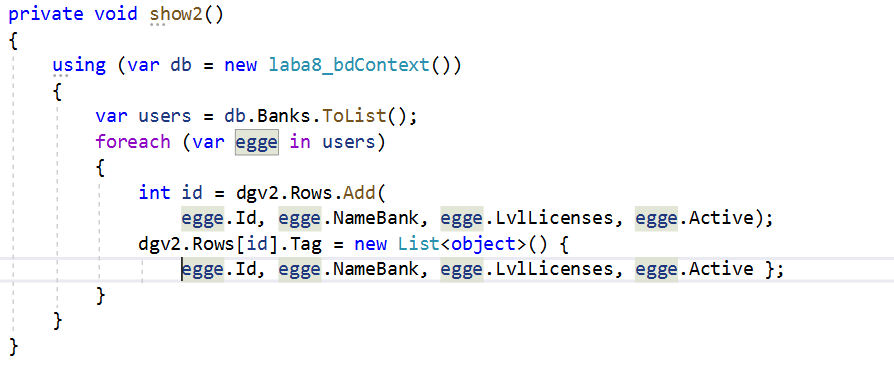


Формирование структуры датагридов

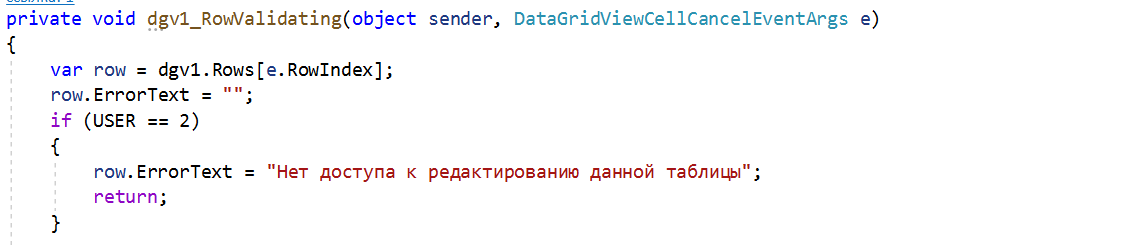
Отображение записей поставщика

****

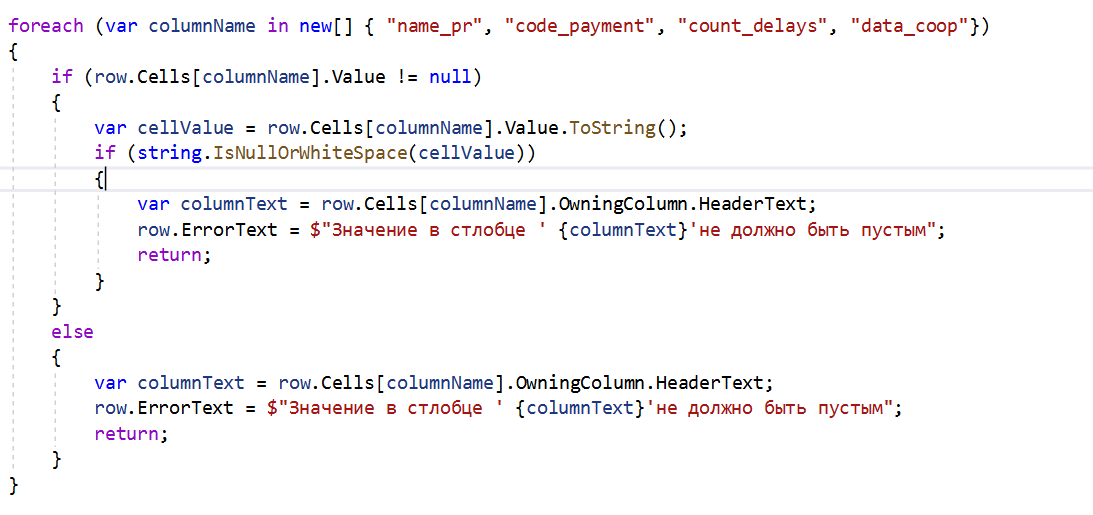
Отображение записей банков

****

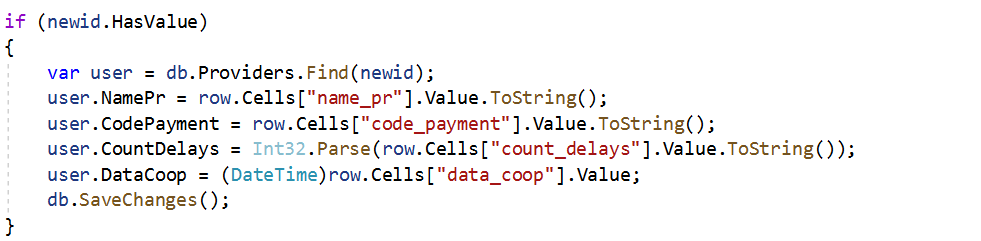
Изменение таблицы поставщики(все кроме связи с банком)



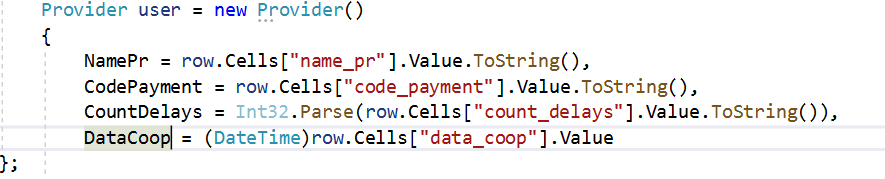
Проверка уровня доступа



Проверка на пустые столбцы

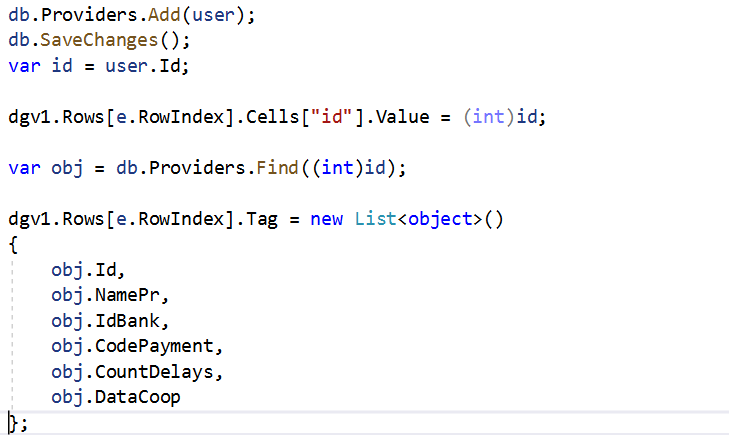


Изменение записи

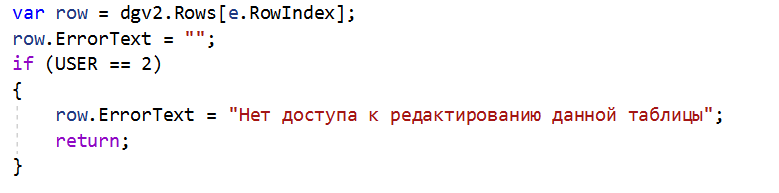


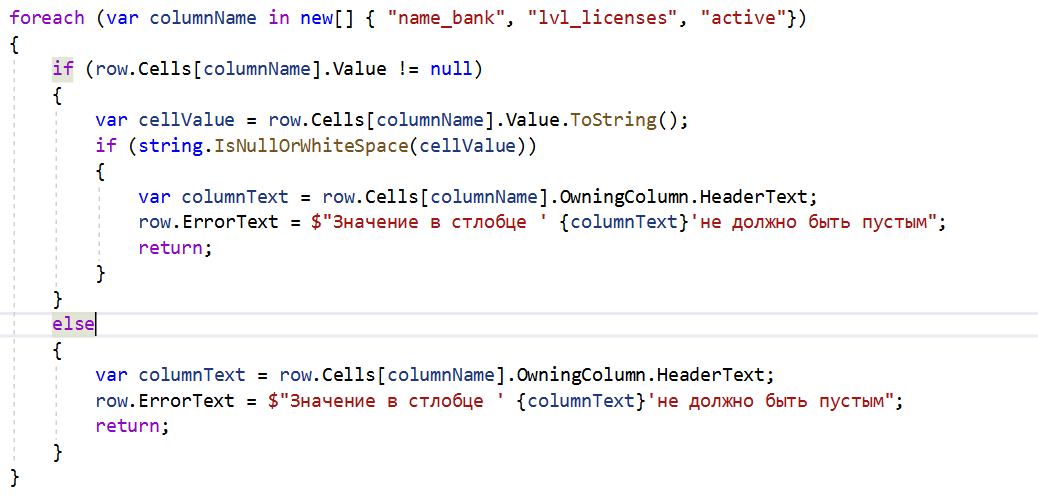
Добавление записи

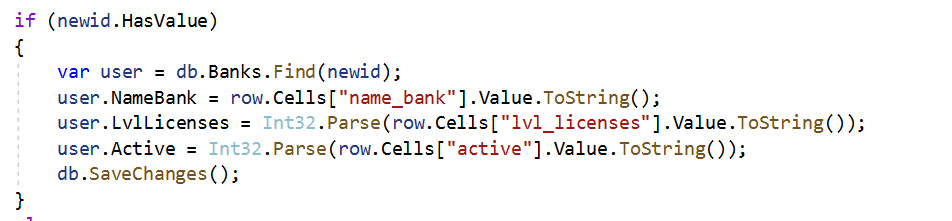
Сохранение изменений и создание тэга

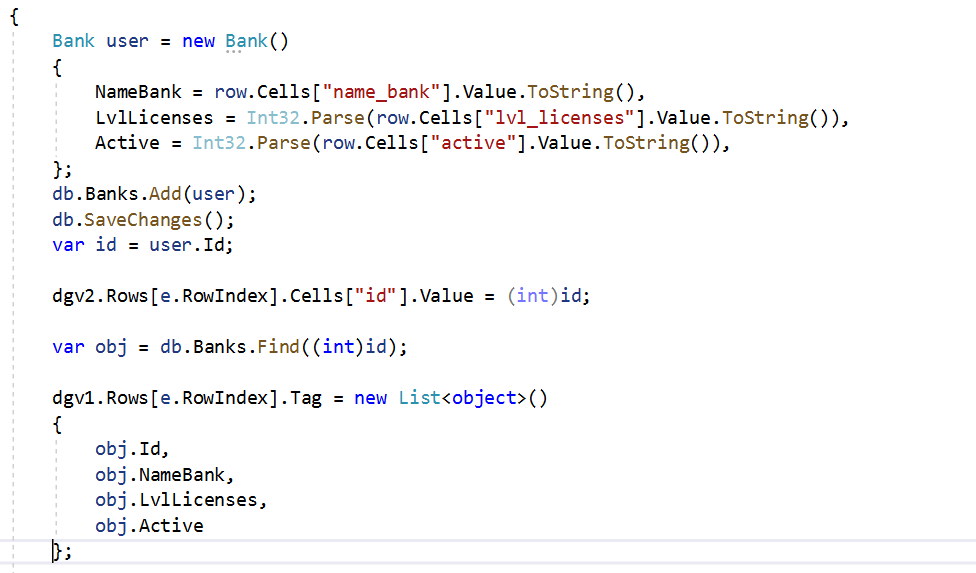


Изменение таблицы банки(процесс аналогичен предыдущему)

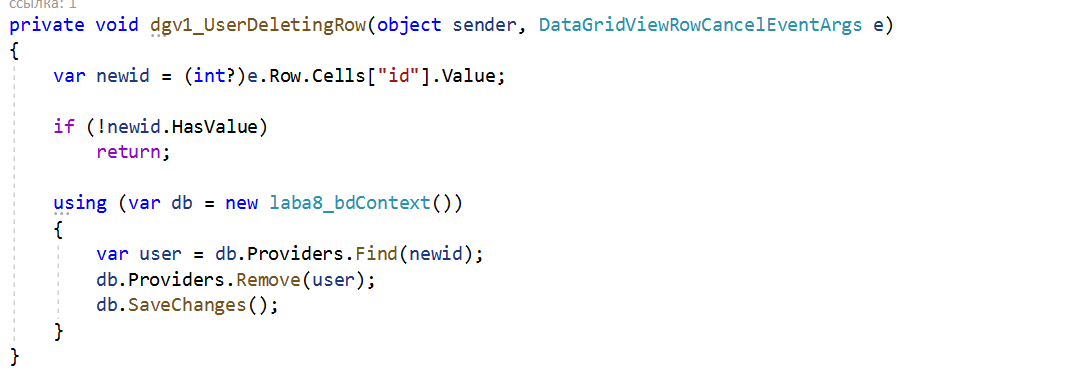




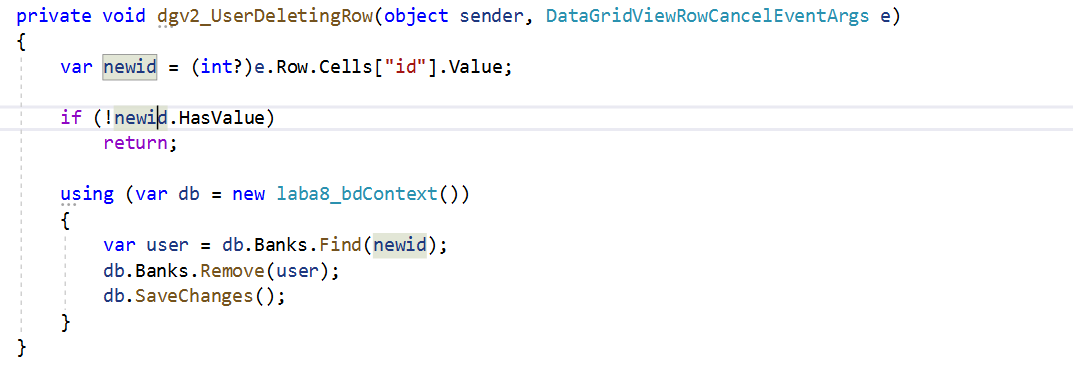




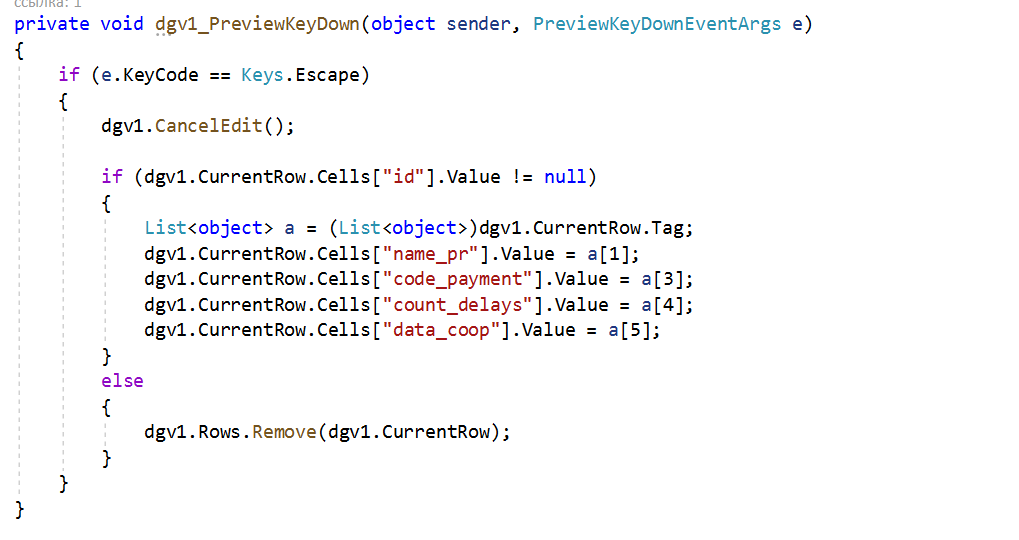
Удаление записи из таблицы «Поставщики»



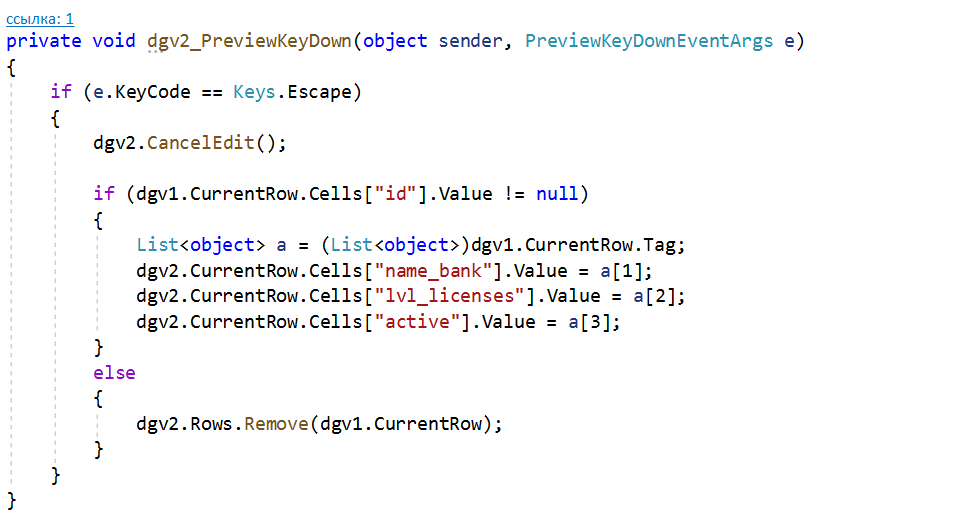
Удаление записи из таблицы «Банки»



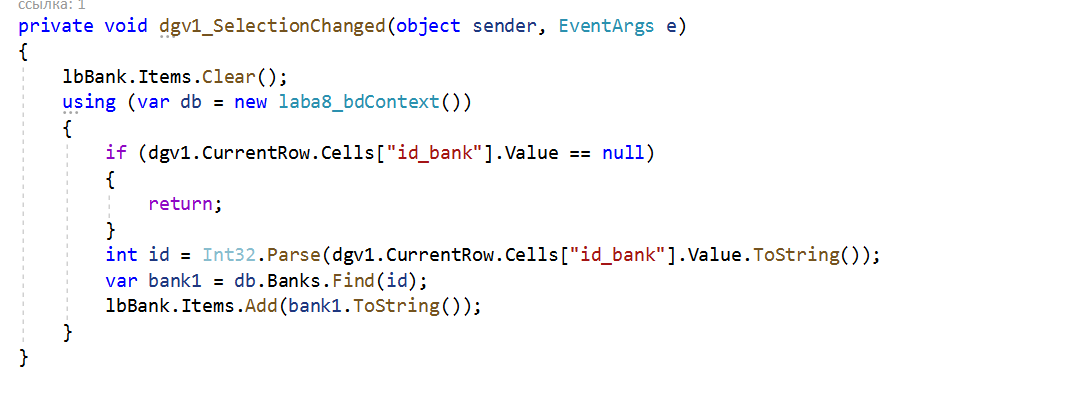
Откат изменений

Для 1 таблицы 

Для 2

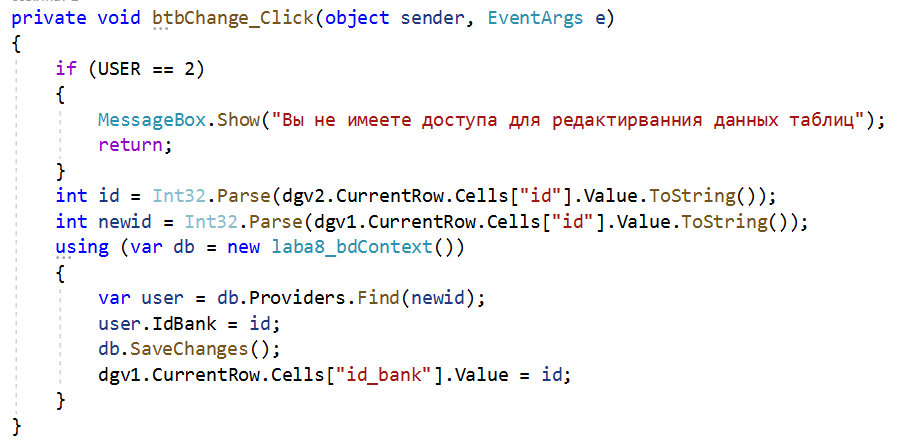


Отображение, какой банк обслуживает данного поставщика

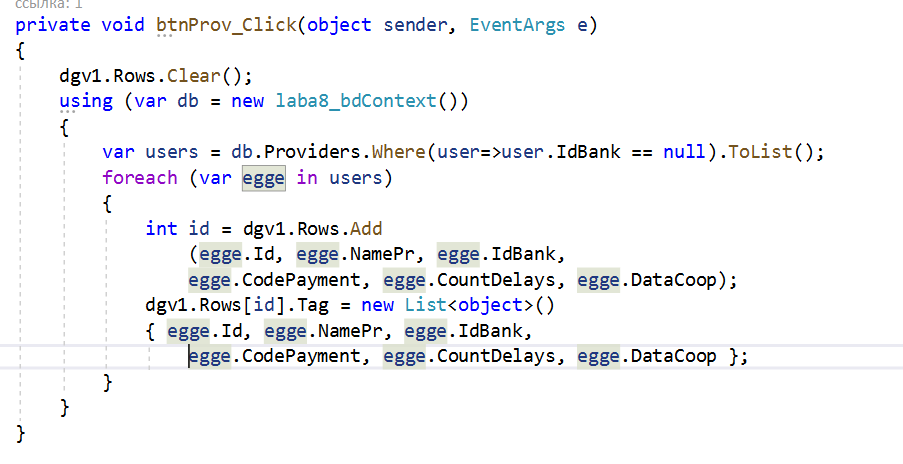


Здесь мы фиксируем процесс переключения между строками первого датагрида и значению id банка у записи ищем в таблице «банки» соответствующую запись и выводим её в листбокс.

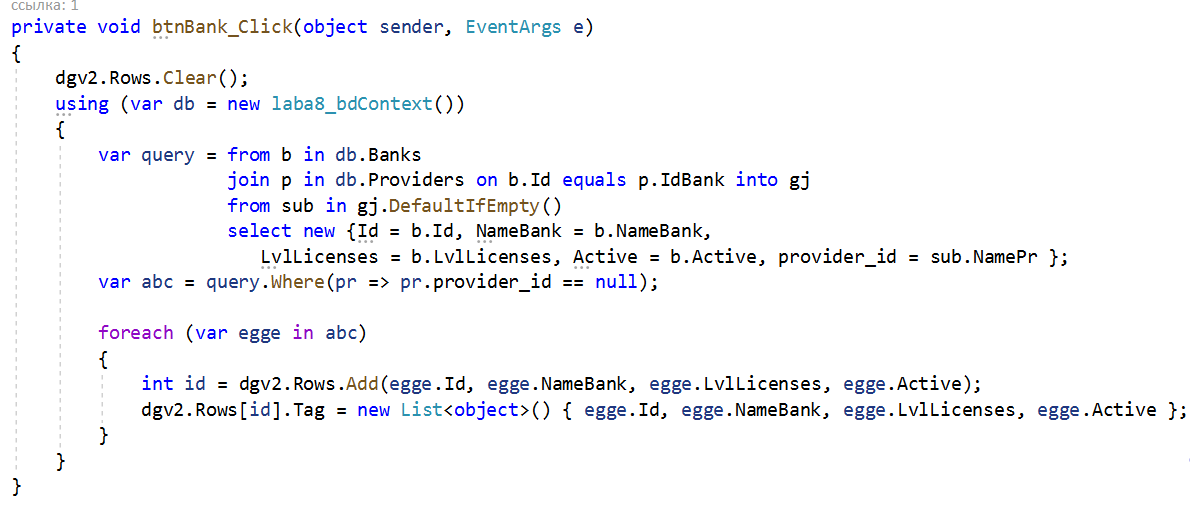
Меняем банк у определенного поставщика



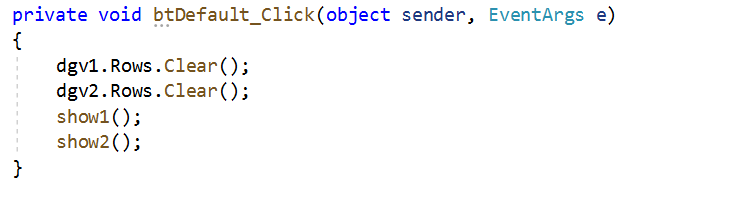
Для того что бы сменить банк у поставщика нам необходимо выделить в таблице банки нужный нам банк и нажать кнопку сменить и тогда связь в БД измениться.



Выводит провайдеров без банка



Выводит банк не обслуживающий никого



Вывод значений по умолчанию

Работа с таблице поставки идёт аналогичным образом как и работа с таблицей пользователи

**public** **partial** **class** **productForm** : Form

{

**int** curRow = -**1**;

**public** **productForm**(**int** user)

{

InitializeComponent();

dgv1.Columns.Add("id", "id");

dgv1.Columns["id"].Visible = **false**;

//dgv1.Columns["id"].ReadOnly = true;

dgv1.Columns.Add("id\_groop", "Группа товара");

dgv1.Columns.Add("name\_pr", "Название товара");

dgv1.Columns.Add("instruct", "Наличие инструкции к товару");

dgv1.Columns.Add("term", "Срок хранения(в днях)");

dgv1.Columns.Add("cost\_pr", "Цена");

**this**.user = user;

show();

}

**public** **int** user;

**private** **void** **show**()

{

**using** (**var** db = **new** laba8\_bdContext())

{

**var** users = db.Products.ToList();

**foreach** (**var** egge **in** users)

{

**int** id = dgv1.Rows.Add(egge.Id, egge.IdGroop, egge.NamePr, egge.Instruct, egge.Term, egge.CostPr);

dgv1.Rows[id].Tag = **new** List<**object**>() { egge.Id, egge.IdGroop, egge.NamePr, egge.Instruct, egge.Term, egge.CostPr };

}

}

}

**private** **void** **productForm\_Load**(**object** sender, EventArgs e)

{

}

**private** **void** **dgv1\_RowValidating**(**object** sender, DataGridViewCellCancelEventArgs e)

{

**var** row = dgv1.Rows[e.RowIndex];

row.ErrorText = "";

**if** (user == **2**)

{

row.ErrorText = "Нет доступа к редактированию данной таблицы";

**return**;

}

**foreach** (**var** columnName **in** **new**[] { "id\_groop", "name\_pr", "instruct", "term", "cost\_pr" })

{

**if** (row.Cells[columnName].Value != **null**)

{

**var** cellValue = row.Cells[columnName].Value.ToString();

**if** (**string**.IsNullOrWhiteSpace(cellValue))

{

**var** columnText = row.Cells[columnName].OwningColumn.HeaderText;

row.ErrorText = $"Значение в стлобце ' {columnText}'не должно быть пустым";

**return**;

}

}

**else**

{

**var** columnText = row.Cells[columnName].OwningColumn.HeaderText;

row.ErrorText = $"Значение в стлобце ' {columnText}'не должно быть пустым";

**return**;

}

}

**var** newid = (**int?**)dgv1.Rows[e.RowIndex].Cells["id"].Value;

**using** (**var** db = **new** laba8\_bdContext())

{

**if** (newid.HasValue)

{

**var** user = db.Products.Find(newid);

user.IdGroop = Int32.Parse(row.Cells[**1**].Value.ToString());

user.NamePr = row.Cells[**2**].Value.ToString();

user.Instruct = **bool**.Parse(row.Cells[**3**].Value.ToString());

user.Term = Int32.Parse(row.Cells[**4**].Value.ToString());

user.CostPr = Int32.Parse(row.Cells[**5**].Value.ToString());

db.SaveChanges();

}

**else**

{

Product user = **new** Product()

{

IdGroop = Int32.Parse(row.Cells[**1**].Value.ToString()),

NamePr = row.Cells[**2**].Value.ToString(),

Instruct = **bool**.Parse(row.Cells[**3**].Value.ToString()),

Term = Int32.Parse(row.Cells[**4**].Value.ToString()),

CostPr = Int32.Parse(row.Cells[**5**].Value.ToString())

};

db.Products.Add(user);

db.SaveChanges();

**var** id = user.Id;

dgv1.Rows[e.RowIndex].Cells["id"].Value = (**int**)id;

**var** obj = db.Products.Find((**int**)id);

dgv1.Rows[e.RowIndex].Tag = **new** List<**object**>() { obj.Id, obj.IdGroop, obj.NamePr, obj.Instruct, obj.Term, obj.CostPr };

}

}

}

**private** **void** **dgv1\_UserDeletingRow**(**object** sender, DataGridViewRowCancelEventArgs e)

{

**var** row = e.Row;

row.ErrorText = "";

**if** (user == **2**)

{

row.ErrorText = "Нет доступа к редактированию данной таблицы";

**return**;

}

**var** newid = (**int?**)e.Row.Cells["id"].Value;

**if** (!newid.HasValue)

**return**;

**using** (**var** db = **new** laba8\_bdContext())

{

**var** user = db.Products.Find(newid);

db.Products.Remove(user);

db.SaveChanges();

}

}

**private** **void** **dgv1\_PreviewKeyDown**(**object** sender, PreviewKeyDownEventArgs e)

{

**if** (e.KeyCode == Keys.Escape)

{

dgv1.CancelEdit();

**if** (dgv1.CurrentRow.Cells["id"].Value != **null**)

{

List<**object**> a = (List<**object**>)dgv1.CurrentRow.Tag;

dgv1.CurrentRow.Cells["name\_pr"].Value = a[**2**];

dgv1.CurrentRow.Cells["instruct"].Value = a[**3**];

dgv1.CurrentRow.Cells["term"].Value = a[**4**];

dgv1.CurrentRow.Cells["cost\_pr"].Value = a[**5**];

}

**else**

{

dgv1.Rows.Remove(dgv1.CurrentRow);

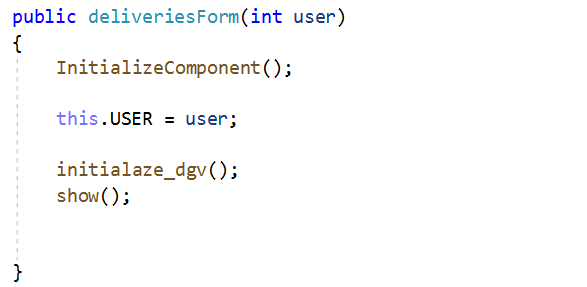
}

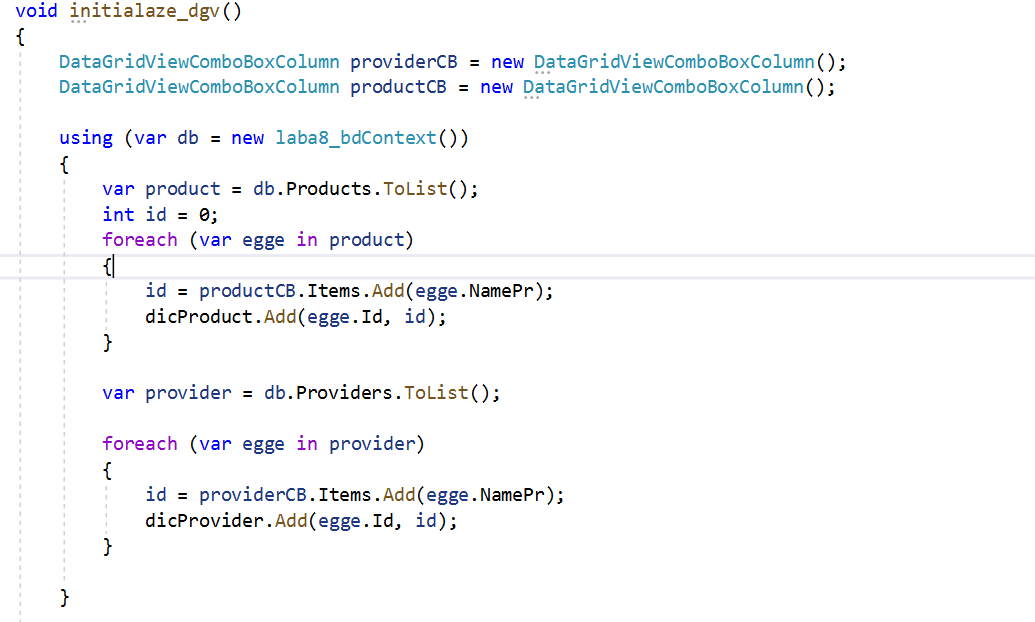
}

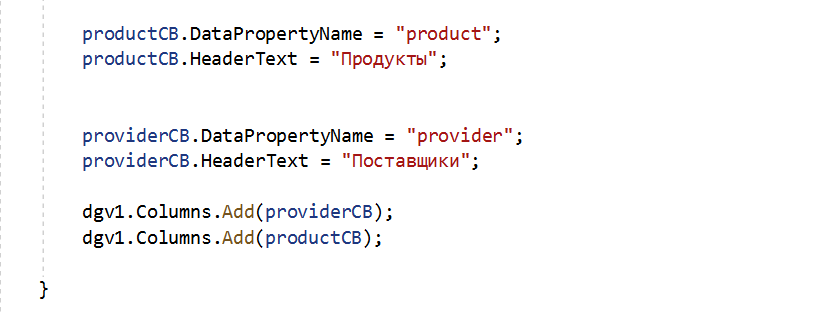
}

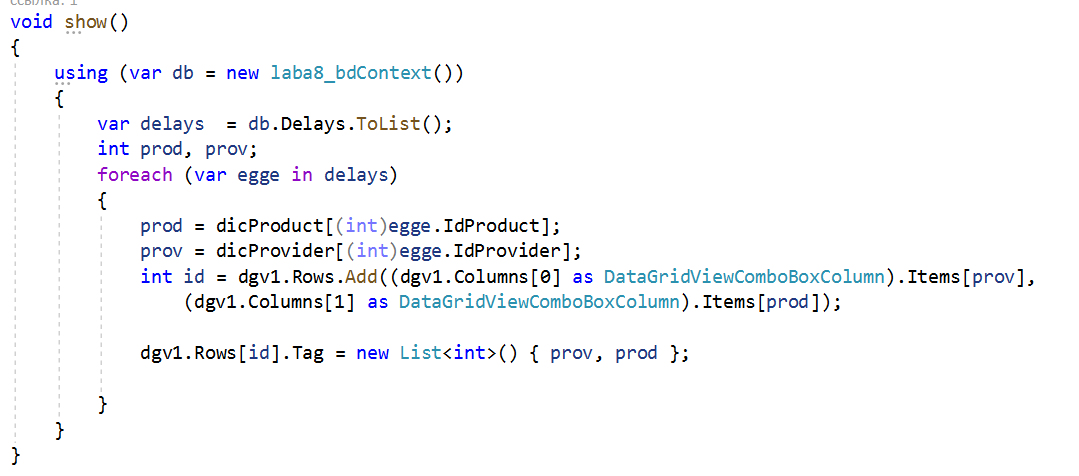
«Поставки»

Единственное отличие от прошлой формы в том что используется вместо ячеек комбобоксы заполненные вариантами всех полей таблиц поставщиков для 1 столбца и товаров для 2.

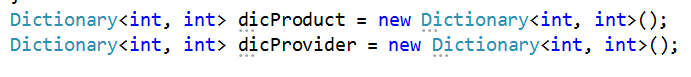




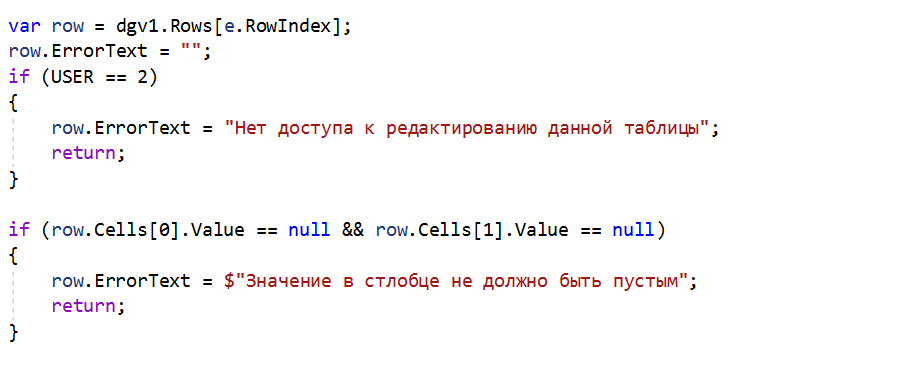


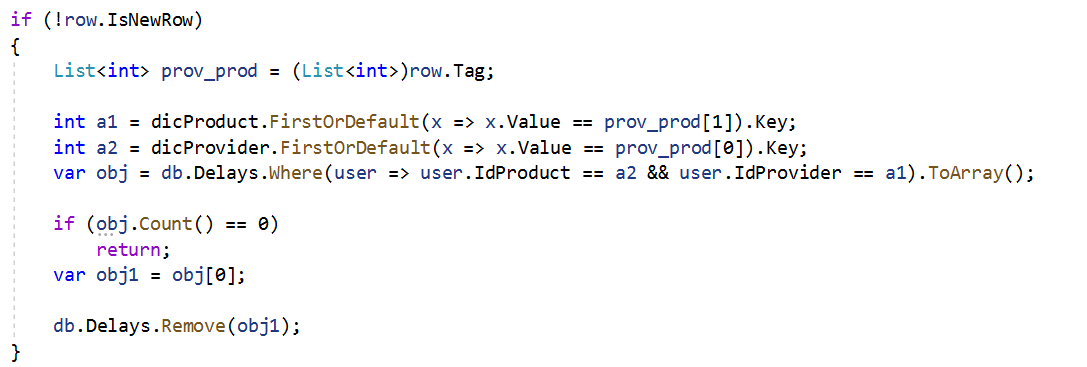


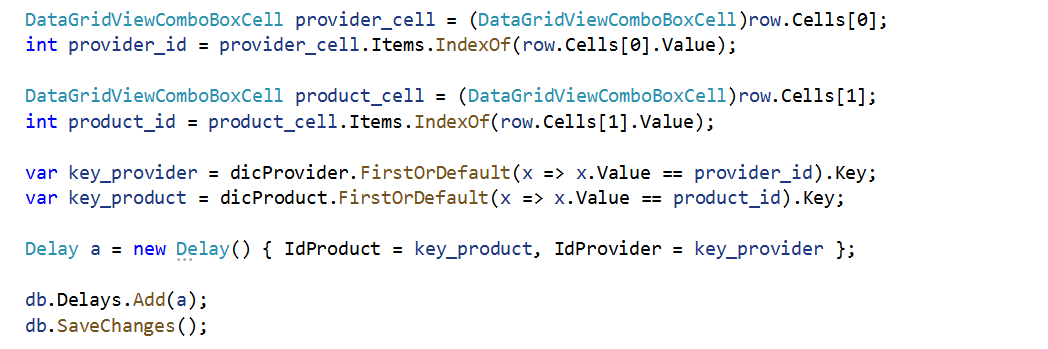
Так же тк id не упорядочены то приходится использовать словари типа<id в БД, номер в комбобоксе>



А в целом алгоритм работы тот же самый









# 5 Тестирование программы

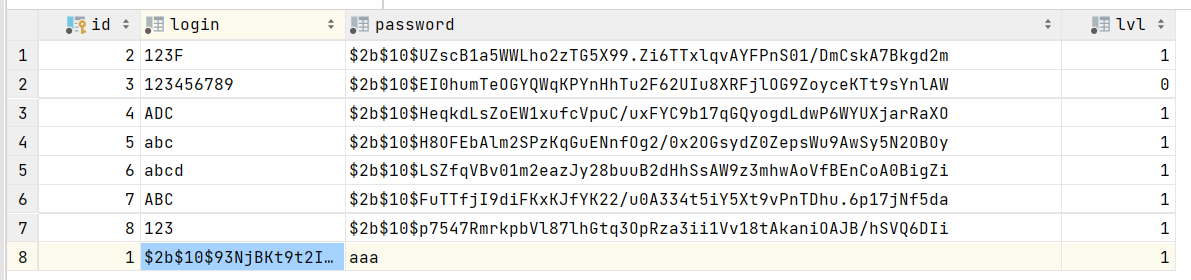
Тестирование программы можно разделить на 12 случаев:

1. Регистрация нового пользователя
2. Удаление пользователя
3. Изменение данных пользователя
4. Демонстрация возможностей гостя (просмотр таблиц)
5. Исполнение пользовательских запросов
6. Добавление данных в таблицы поставщик, банк, продукты
7. Изменение данных в таблице поставщик, банк, продукты
8. Удаление данных в таблице поставщик, банк, продукты
9. Добавление записей в таблице поставки
10. Изменение записей в таблице поставки
11. Удаление записей в таблице поставки
12. Изменение связи для поставщика
13. Регистрация нового пользователя

Входные данные:

Логин 12345

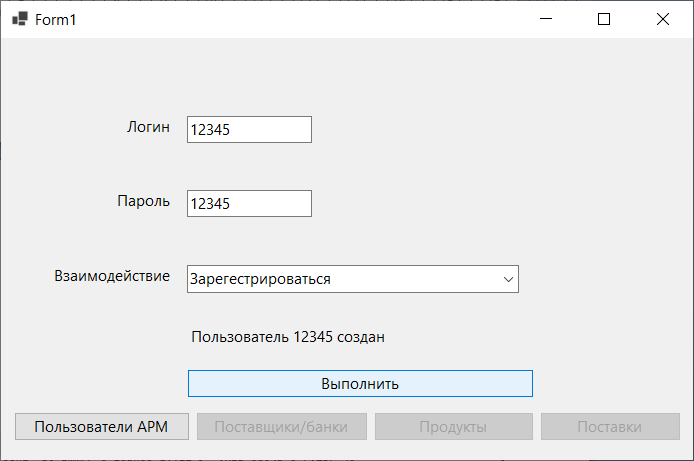
Пароль 12345

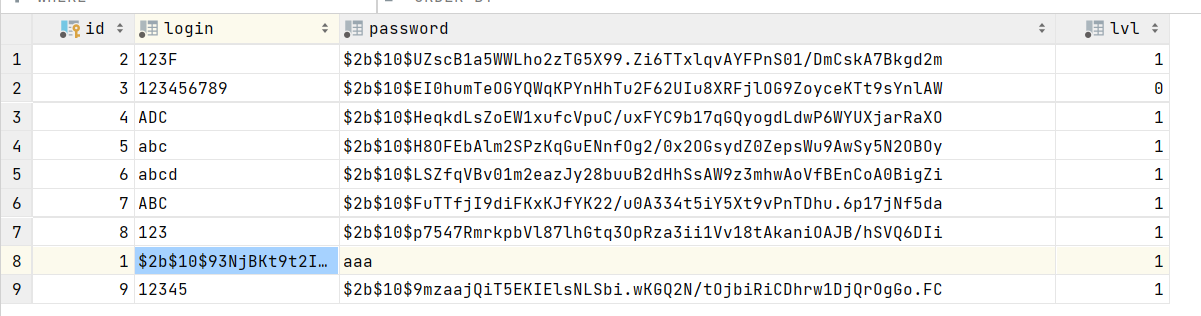


Ожидаемый результат

Регистрация нового пользователя

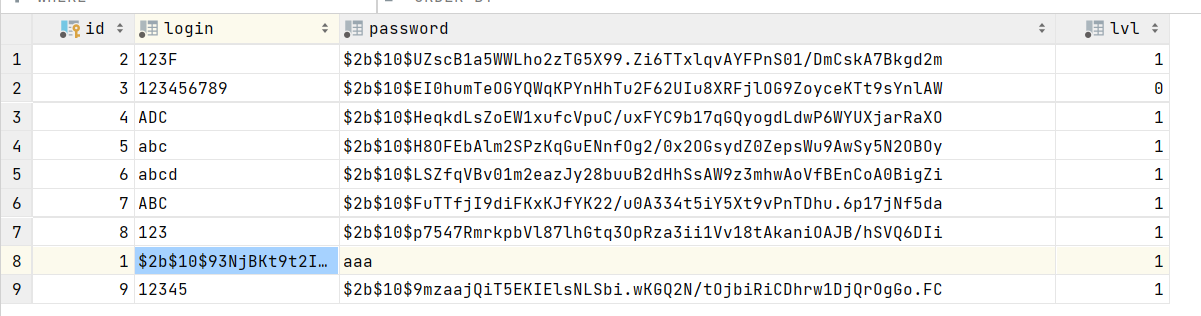
Выходные данные





1. Удаление пользователя

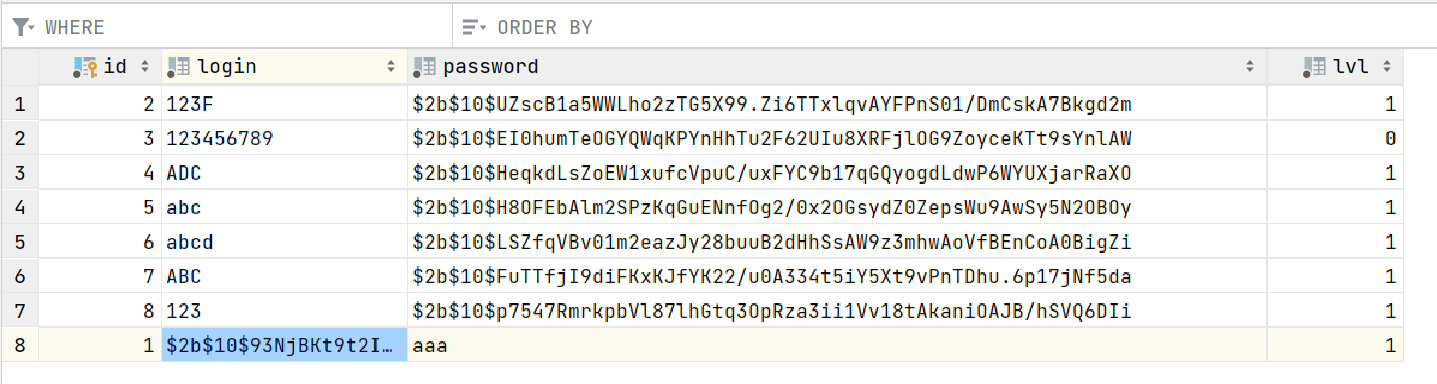
Входные данные



Ожидаемый результат

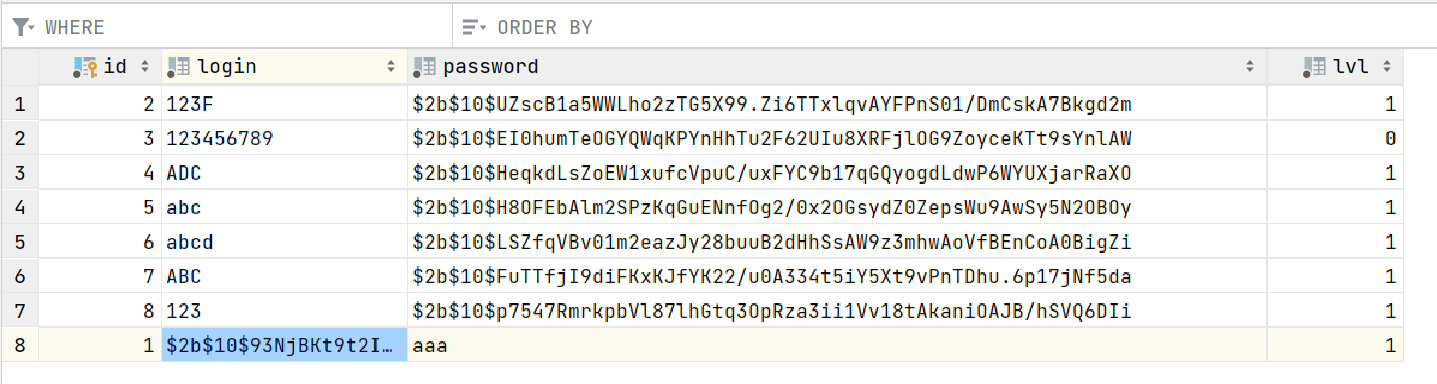
Удаление пользователя

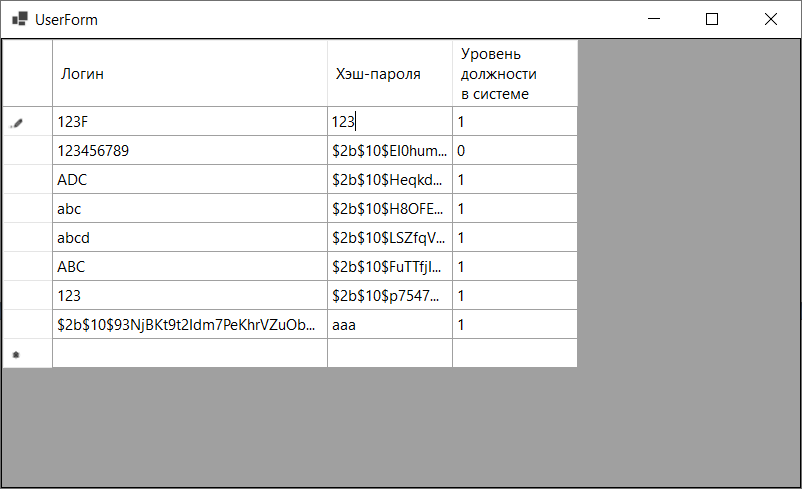
Выходные данные



1. Изменение данных пользователя

Входные данные

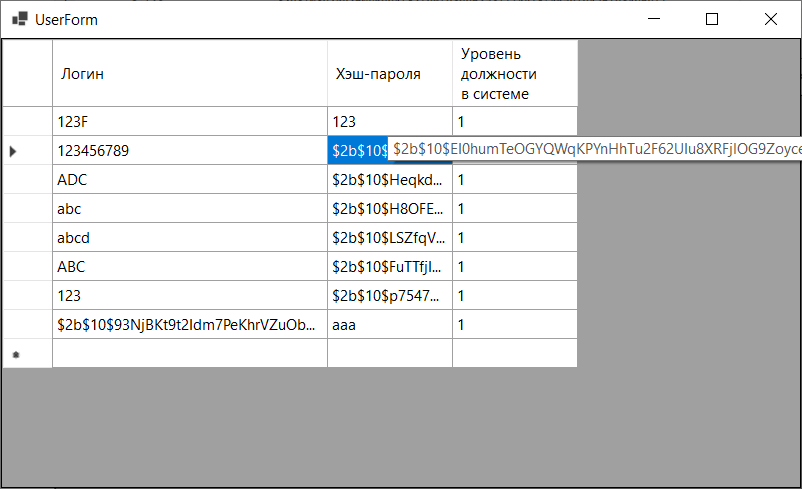




Ожидаемый результат

Изменение пароля у пользователя с логином 123F

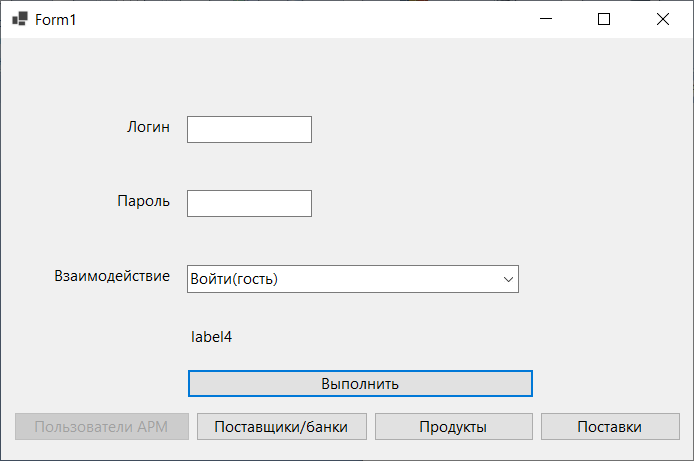
Выходные данные



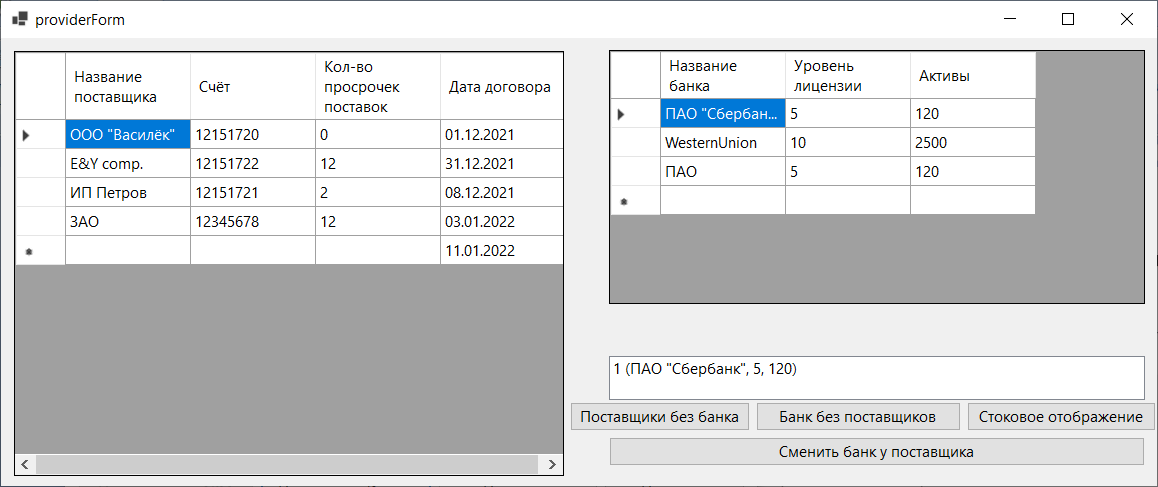


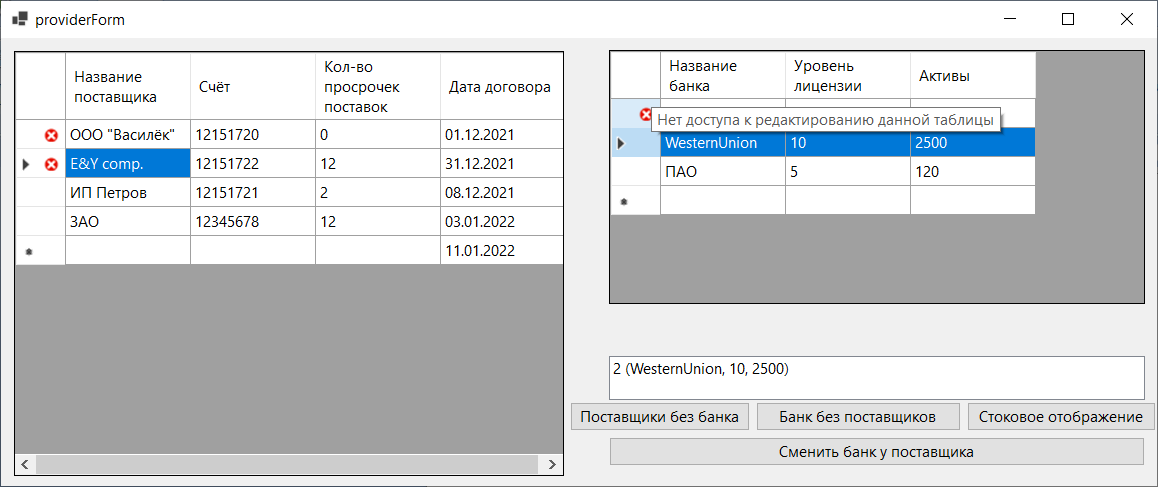
1. Демонстрация возможностей гостя (просмотр таблиц)

Входные данные

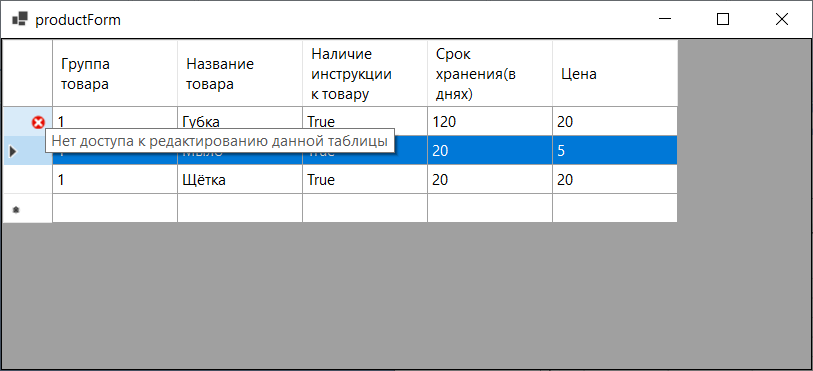


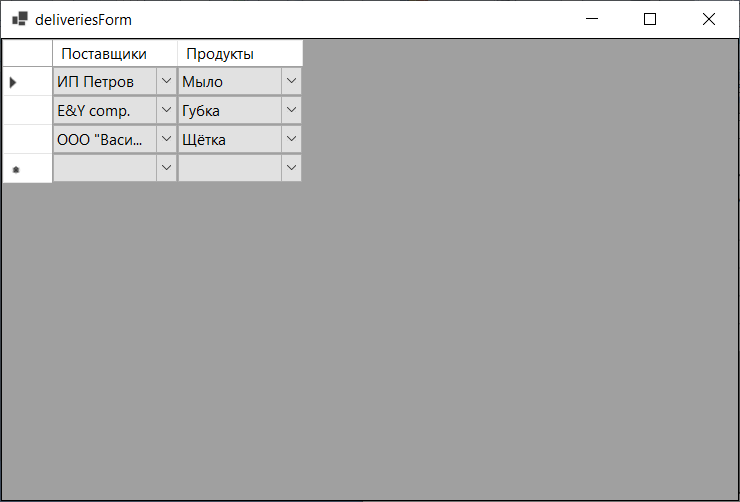
Выходные данные

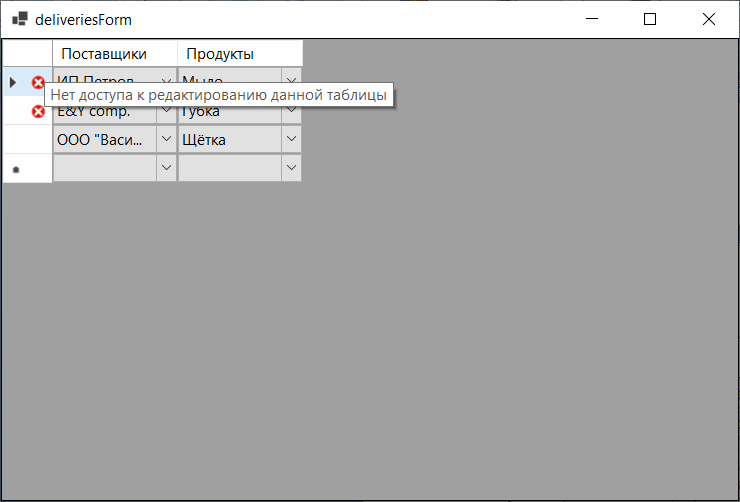




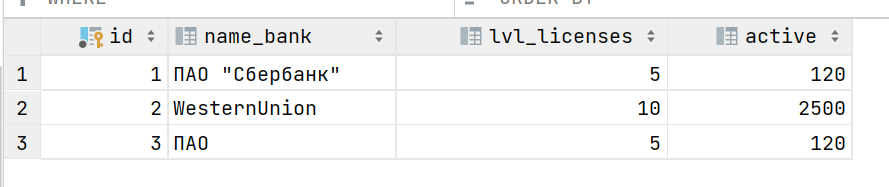


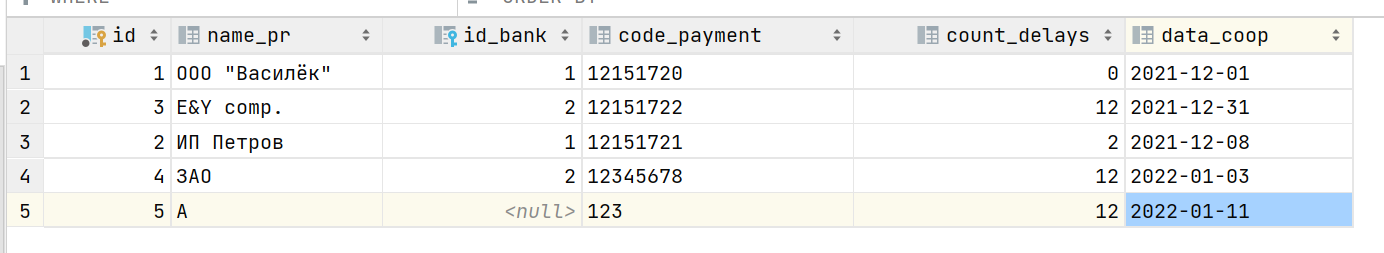


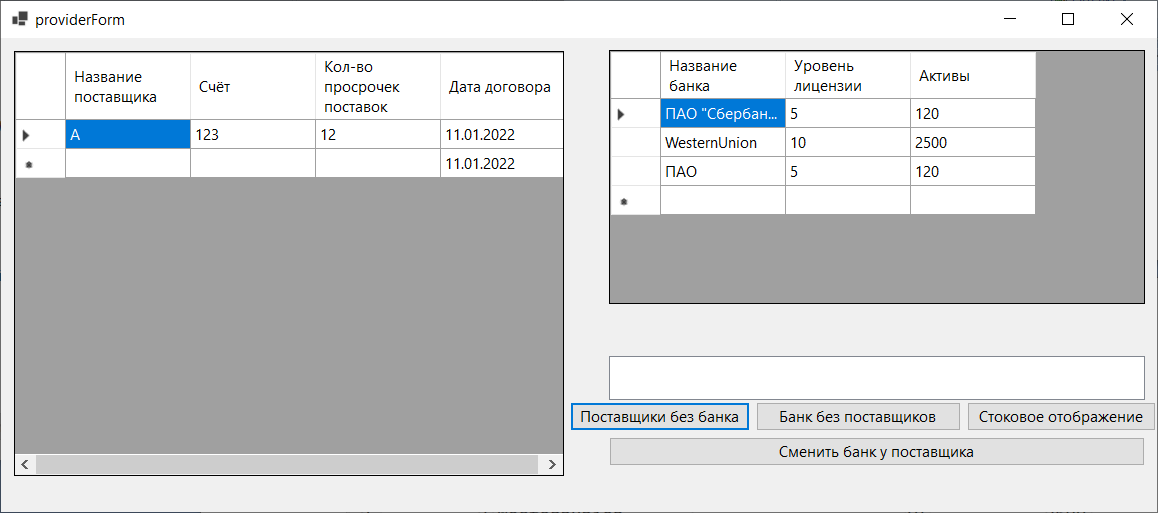


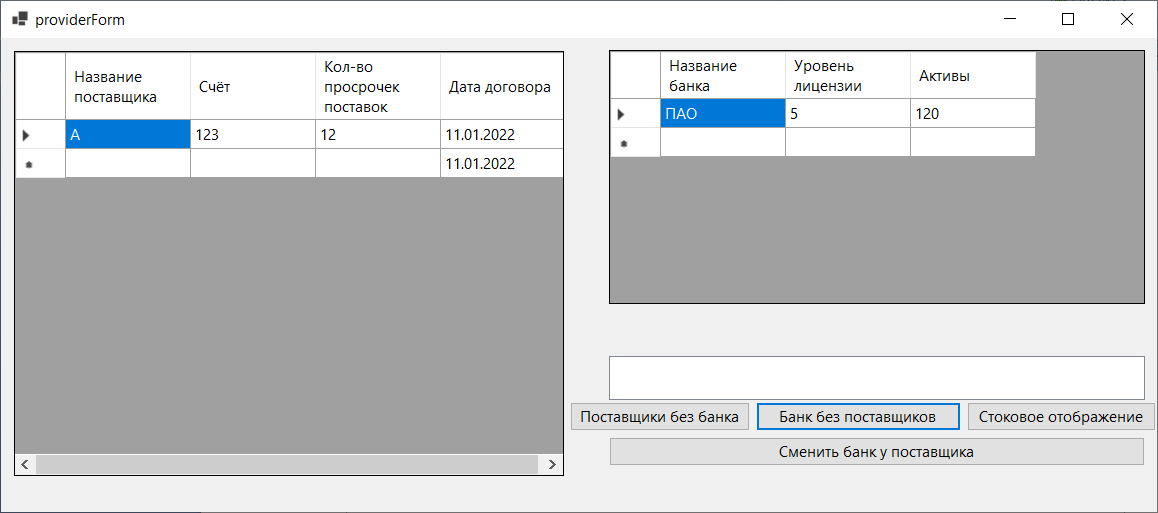


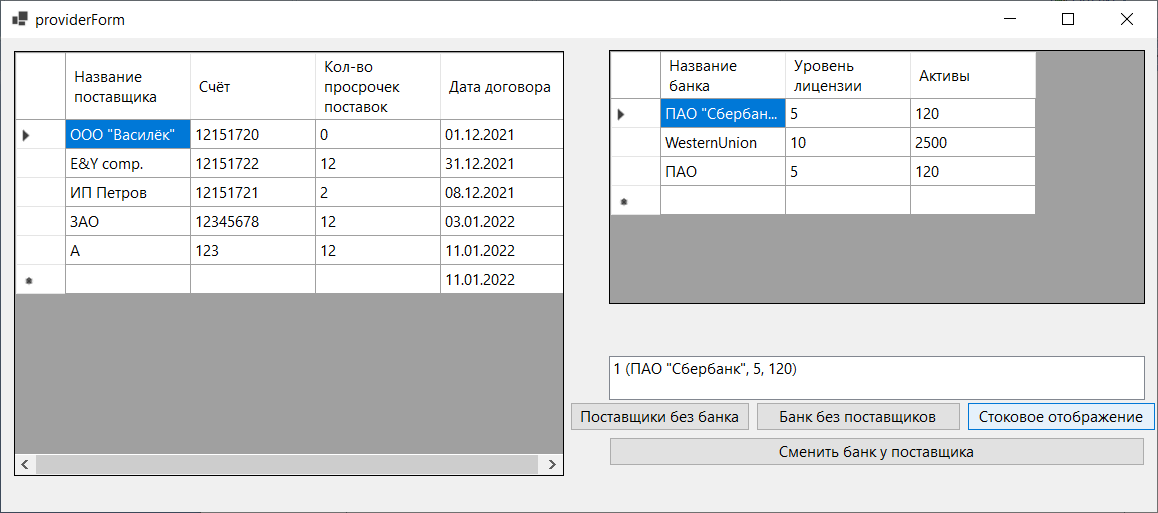
1. Исполнение пользовательских запросов





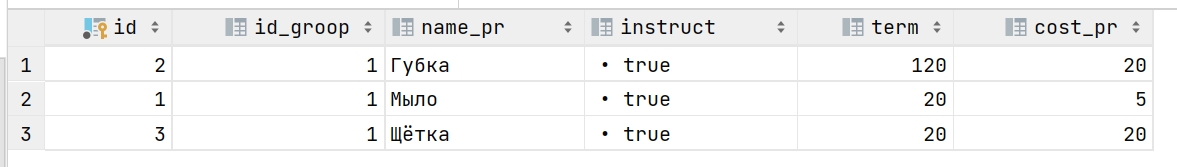






1. Добавление данных в таблицы поставщик, банк, продукты

Входные данные

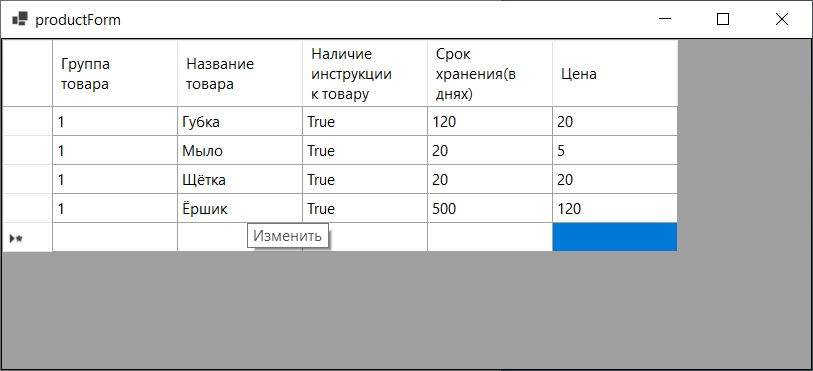


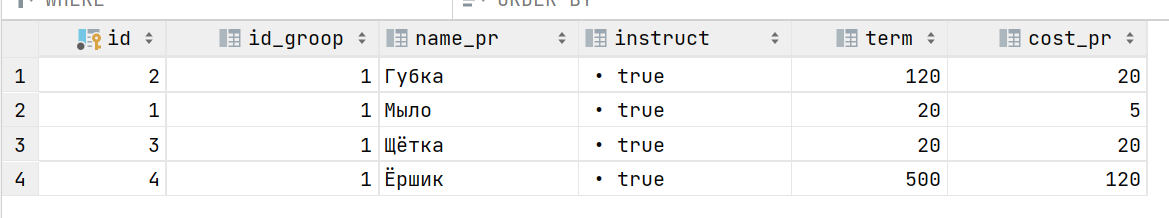
{1; Ёршик; true; 500; 120}

Ожидаемый результат

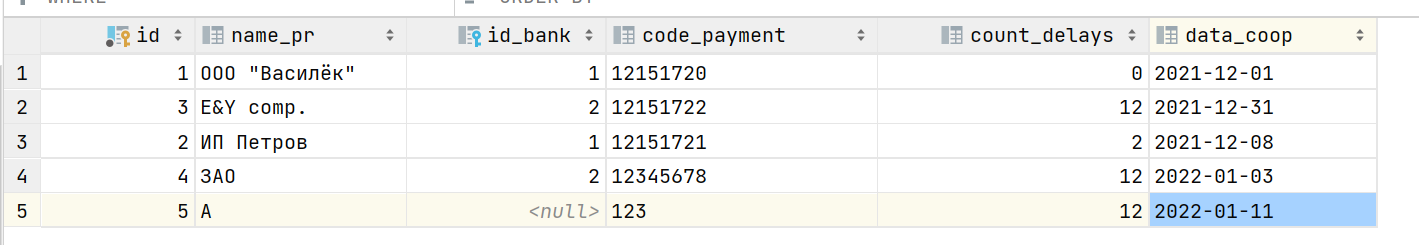
Создание новой записи

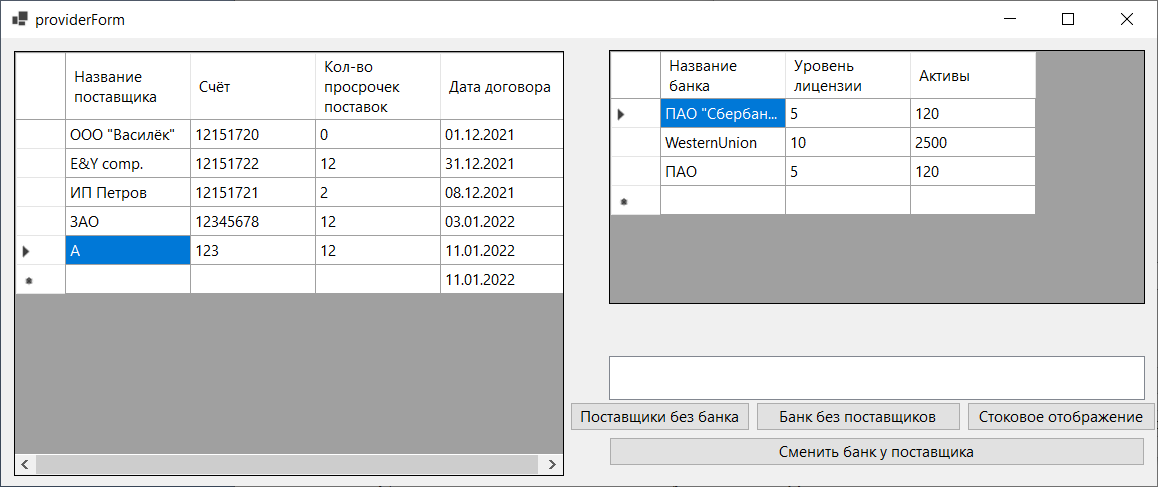
Выходные данные





1. Изменение данных в таблице поставщик, банк, продукты

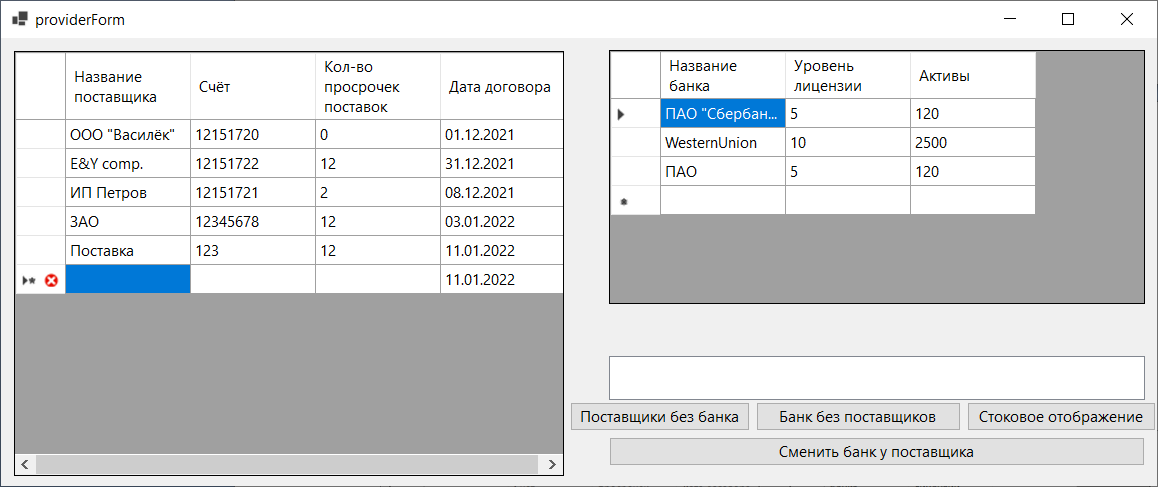
Входные данные 

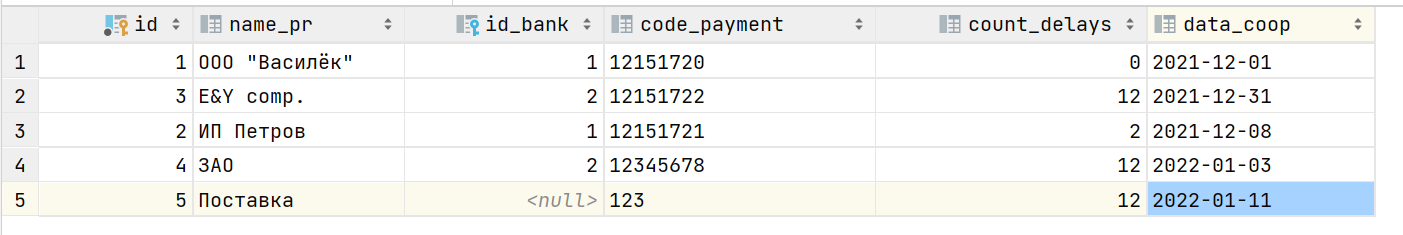


Ожидаемый результат

Смена имени у поставщика «А» на имя «Поставка»

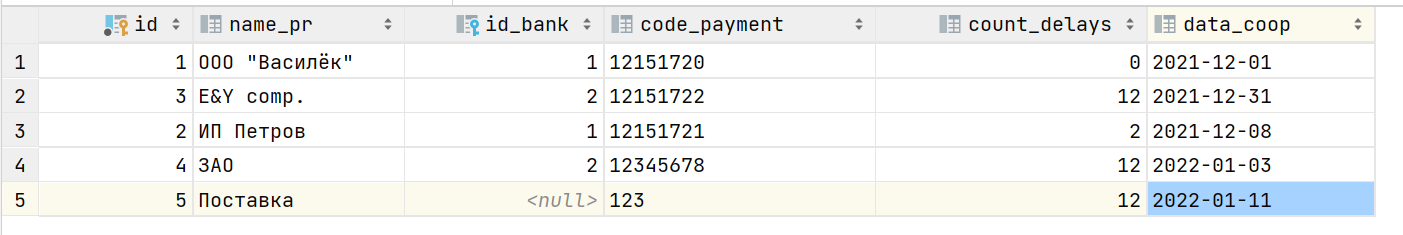
Выходные данные





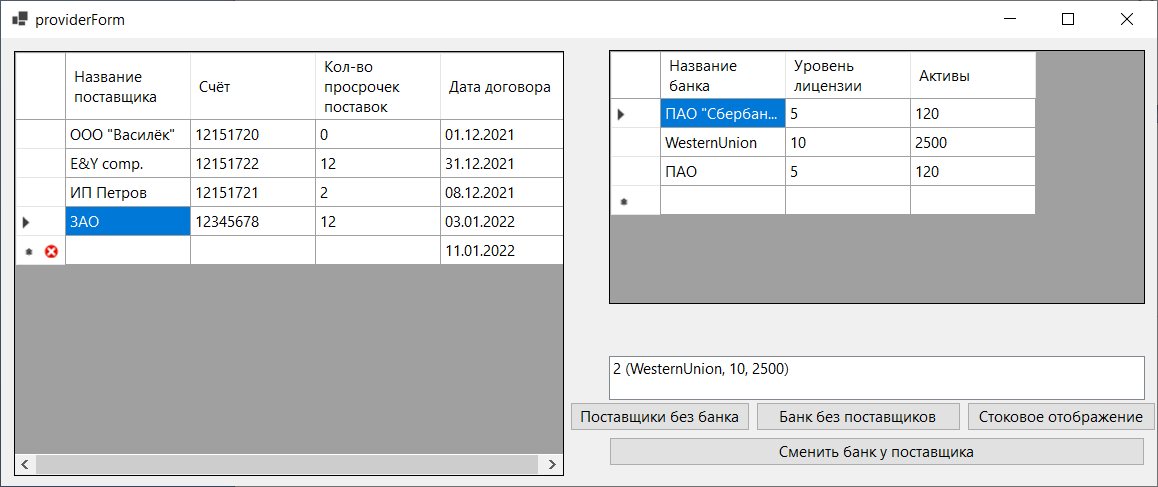
1. Удаление данных в таблице поставщик, банк, продукты

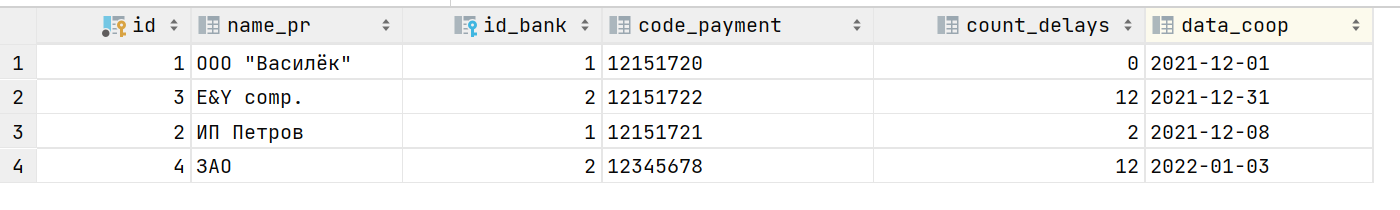
Входные данные

Ожидаемый результат

Удаление записи Поставка

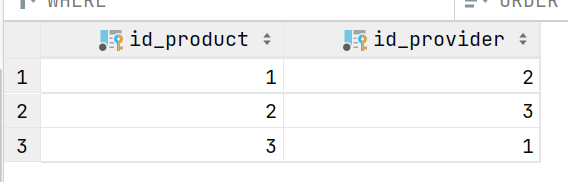
Выходные данные



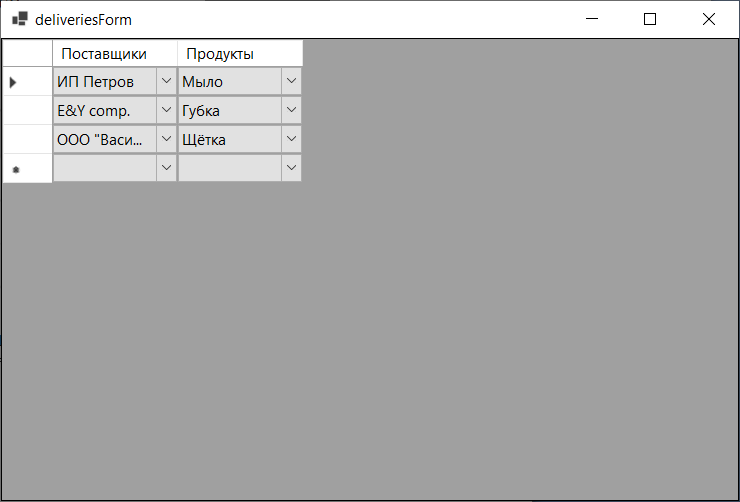


1. Добавление записей в таблице поставки

Входные данные



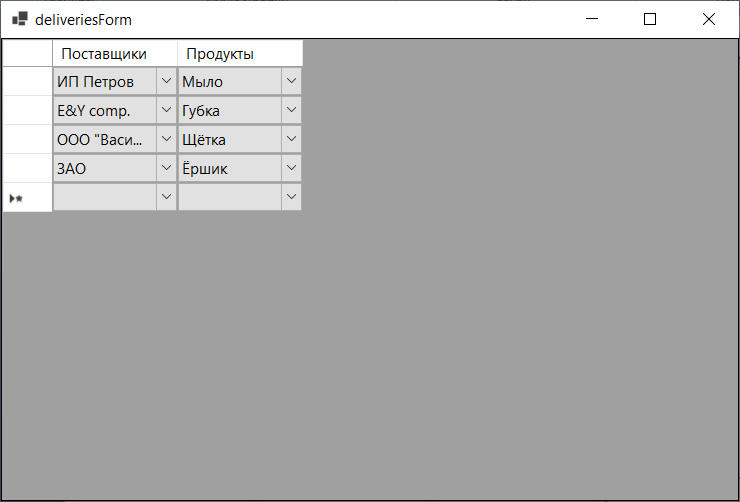
SELECT *\** FROM delays JOIN provider ON delays.id\_provider = provider.id JOIN product p on delays.id\_product = p.id;

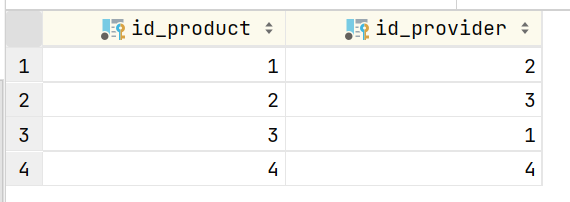


Ожидаемый результат

Создание новой записи

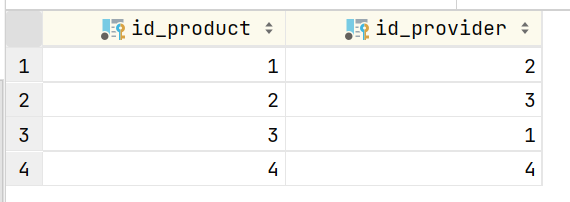
Выходные данные

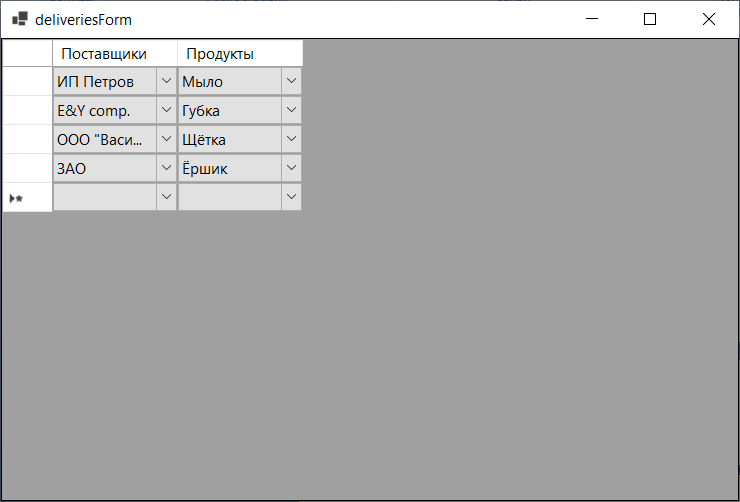




1. Изменение записей в таблице поставки

Входные данные

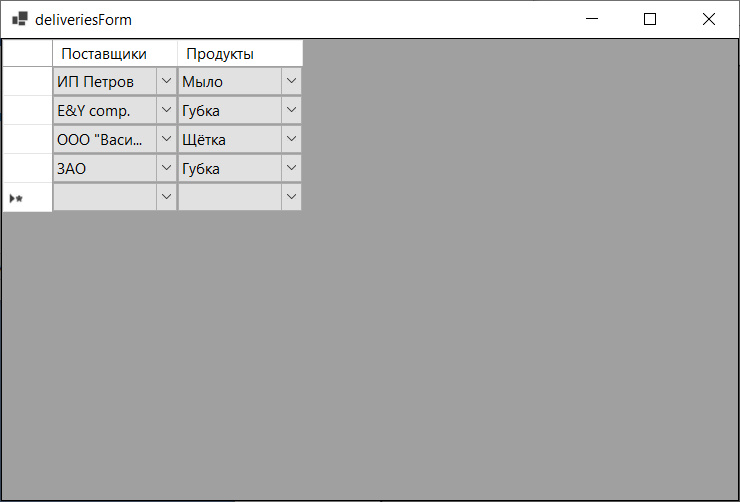


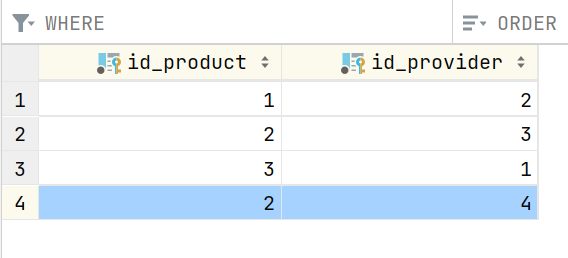


Ожидаемый результат

Изменение записи

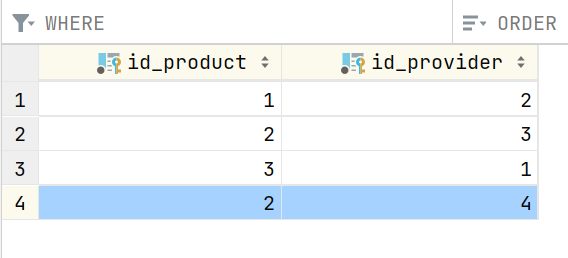
Выходные данные



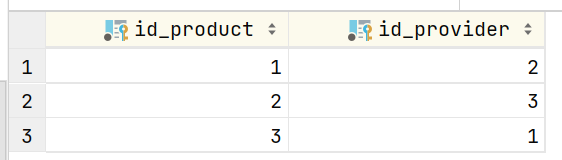


1. Удаление записей в таблице поставки

Входные данные



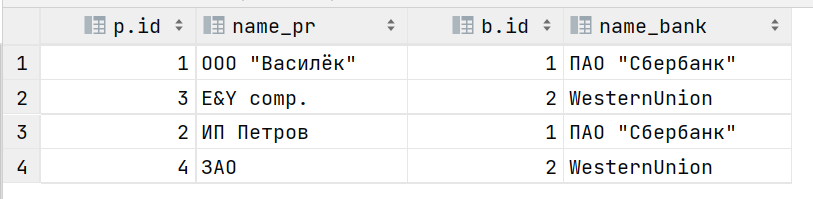
Выходные данные

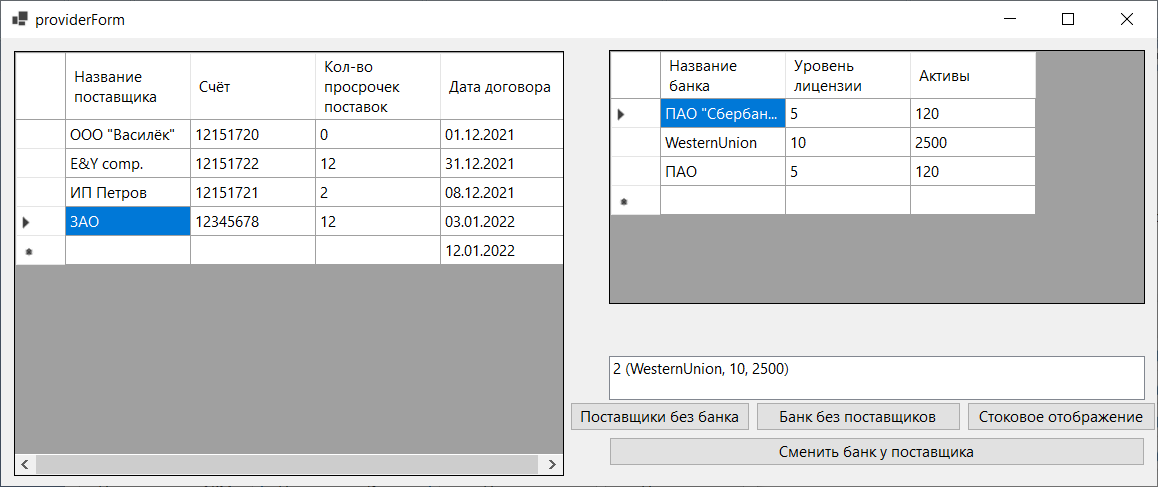


1. Изменение связи для поставщика

Входные данные

SELECT p.id, p.name\_pr, b.id, b.name\_bank FROM provider p JOIN bank b on p.id\_bank = b.id

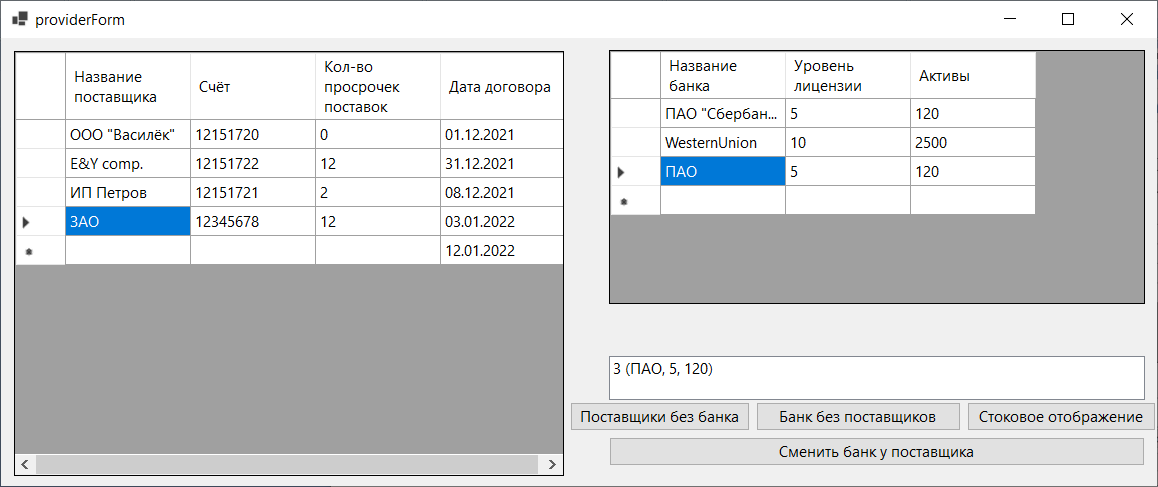


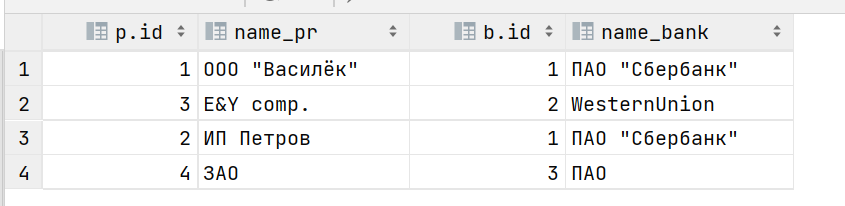


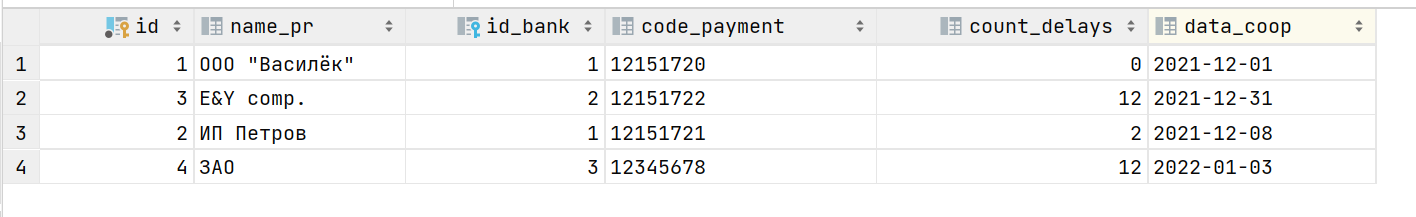
Ожидаемый результат

Смена банка у поставщика «ЗАО» на банк «ПАО»

Выходные данные







Вывод: Программа прошла все тесты и работает корректно.