# ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

**"ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ"**

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе по курсу

«Базы данных»

Тема работы:

«Типографии города Донецка»

Руководители:

Щедрин С.В

Незамова Л.В

Рычка О.В

Разработал:

ст.гр. ПИ-18б

Моргунов А.Г

Донецк — 2020

РЕФЕРАТ

Пояснительная записка содержит: 83 страниц, 64 рисунка, 3 источника, 4 приложения.

Объектами исследования являются типографии города Донецка.

Цель работы - разработка системы для учета деятельности типографий города Донецка.

Результатом работы является база данных и клиентское приложение для взаимодействия с ней. Система может выполнять такие функции как: добавление, удаление, поиск записей в таблицах и справочниках БД, составление однотабличного и многотабличного отчетов в Microsoft Excel, составление HTML публикации на основе таблицы, создание различных типов диаграмм основанных на статистических данных, выполнение запросов различного уровня сложности.

Для разработки системы использована СУБД PostgreSQL.

SQL, C#, POSTGRESQL, ТАБЛИЦА, БАЗА ДАННЫХ, СУБД, ТИПОГРАФИЯ, SQL-ЗАПРОС, ВИДЫ БУМАГИ, БАНКИ

СОДЕРЖАНИЕ

ВВЕДЕНИЕ…………………………………………………………………………..5

1 ПРЕДПРОЕКТНОЕ ИССЛЕДОВАНИЕ………………………………………..6

1.1 Проблемы автоматизации учета деятельности типографий горда Донецка………………………………………………………………………………6

1.2 Разработка информационной системы «Учет деятельности Типографий города Донецка» ……………………………………………………………………6

1.2.1 Анализ предметной области…………………………………………………6

1.2.2 Описание входных документов…………………………………………..…7

1.2.3 Описание выходных документов……………………………………………7

1.2.4 Список ограничений…………………………………………………………7

1.3 Назначение и функции системы ………………………………………………..8

2 ТЕХНИЧЕСКОЕ ПРОЕКТИРОВАНИЕ…………………………………………9

2.1 Инфологическое проектирование……………………………………………..9

2.2 Даталогическое проектирование……………………………………………..10

2.3 Физическое проектирование…………………………………………………10

3 РАЗРАБОТКА ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ …………………………..15

3.1 Создание проекта……………………………………………………………..15

3.2 Создание базы данных…………………………………………………….…15

3.3 Создание таблиц………………………………………………………………15

3.4 Схема БД………………………………………………………………………17

3.5 Разработка пользовательского интерфейса …………………………………17

3.6 Разработка процедур обработки данных (модулей) ……………………… 26

3.7 Тестирование и отладка БД и ИС……………………………………………28

3.8 Разработка эксплуатационной документации БД и ИС………….…………29

ВЫВОДЫ………………………………………………………………………….30

СПИСОК ЛИТЕРАТУРЫ…………………………………………………………31

ПРИЛОЖЕНИЕ А. Техническое задание ………………………………..………32

ПРИЛОЖЕНИЕ Б. SQL-запросы…………………………………………………33

ПРИЛОЖЕНИЕ В. Руководство пользователя……………..……………………50

ПРИЛОЖЕНИЕ Г. Текст программы……………………………………….……51

ВВЕДЕНИЕ

В современном мире технологии все больше и больше вливаются в жизнь людей. В том числе и в бизнес – сферу. Многие предприниматели и организации используют современные методы обработки информации. Основным средством для быстрой обработки информации являются базы данных. Это средство позволяет обрабатывать, выводить, хранить и манипулировать огромными объемами данных. Поэтому такое средство востребовано и используется повсеместно.

Система управления базами данных(СУБД) – это программный комплекс, обеспечивающий централизованное хранение данных и предоставляющий приложениям услуги по обработке данных.

Совокупность данных, хранимых под управлением СУБД, называется базой данных. В оригинальном английском варианте словосочетание data base означает «основание, состоящее из данных». Этот смысл несколько искажается в русском словосочетании «база данных». На самом деле это – фундамент, на котором строятся приложения и который состоит из данных. Действительно, данные (а следовательно, база данных) являются очень существенной частью практически любой информационной системы[1]

Целью разработки является создание базы данных типографий города Донецка с помощью СУБД PostgreSQL и создание клиентского приложение для взаимодействия с этой базой данных на языке программирования C#

Разработанная система может применятся для учета деятельности типографий города Донецка

1. ПРЕДПРОЕКТНОЕ ИССЛЕДОВАНИЕ

1.1 Проблемы автоматизации учета деятельности типографий горда Донецка

Работа типографий подразумевает учет огромного числа данных. В них входят: информация о типографиях, работниках, заказчиках, заказах, информация о различных изделиях и их видах и т.д. Также финансовый вопрос стоит не на последнем месте. Из-за ограниченного бюджета необходимо максимально сократить расходы на учет информации, но при этом не потерять надежность и качество хранения информации. Без автоматизации невероятно сложно и дорого следить за всем. Следовательно автоматизация упрощает и уменьшает затраты на обработку и хранение информации. Также благодаря переходу на хранение данных с помощью БД облегчается доступ и контроль за информацией.

1.2 Разработка информационной системы «Учет деятельности Типографий города Донецка

1.2.1 Анализ предметной области

Благодаря анализу можно заключить, что для учета деятельности типографий нуобходимы такие данные название типографии, тип собственности, район, адрес типографии, телефон типографии, год открытия, фио заказчика (физ.или юр. лицо), адрес заказчика. дата рождения/регистрации, телефон заказчика, расчетный счет заказчика (номер р/с),. банк размещения р/c, вид изделия заказчика, название издания, титульная страница (эскиз автора), фио принявшего заказ в типографии, цена экземпляра изделия, тираж, количество листов в изделии, формат листов, тип бумаги, плотность бумаги, дата принятия заказа, дата выполнения заказа(план), дата выполнения заказа(факт), предоплата (принятый аванс), дополнительные сведения о заказе.

1.2.2 Описание входных документов

Ввод данных производится с помощью формы добавления записи, а также генерации записей. Для изменения данных используется форма обновления записи.

1.2.3 Описание выходных документов

Входные данные могут быть различными:

- Excel таблица с результатом запроса

-HTML-страница с выбранными данными

-Диаграммы

1.2.4 Список ограничений

При разработке базы данных были выявлены следующие ограничения:

-Год открытия типографии должен быть больше 1980 и меньше 2020

-Дата принятия заказа не может быть меньше года основания типографии

-Дата выполнения заказа (план) и (факт) должны быть больше даты принятия заказа

- Тираж заказа начинается с 50 штук

-Цена одного экземпляра изделия начинается с 50 у/е

-Количество листов издания больше 0

-Предоплата больше или равна 0

-Дата рождения заказчика должна быть меньше 2002 года

-Плотность бумаги должна быть больше 40 и меньше 350

1.3 Назначение и функции системы

Разрабатываемая программа предназначена для учета деятельности типографий города Донецка. Благодаря своим возможностям она облегчает работу с данными.

К функциям системы относятся: хранение данных, поиск данных, управление данными, группировка данных, сохранение целостности данных, генерация псевдоинформации.

1. ТЕХНИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

2.1 Инфологическое проектирование

Инфологическая модель (информационно-логическая модель) — ориентированная на человека и не зависимая от типа СУБД модель предметной области, определяющая совокупности информационных объектов, их атрибутов и отношений между объектами, динамику изменений предметной области, а также характер информационных потребностей пользователей. Инфологическая модель предметной области может быть описана моделью "сущность—связь" (моделью Чена), в основе которой лежит деление реального мира на отдельные различимые сущности, находящиеся в определенных связях друг с другом, причем обе категории — сущность и связь полагаются первичными, неопределенными понятиями.[2]

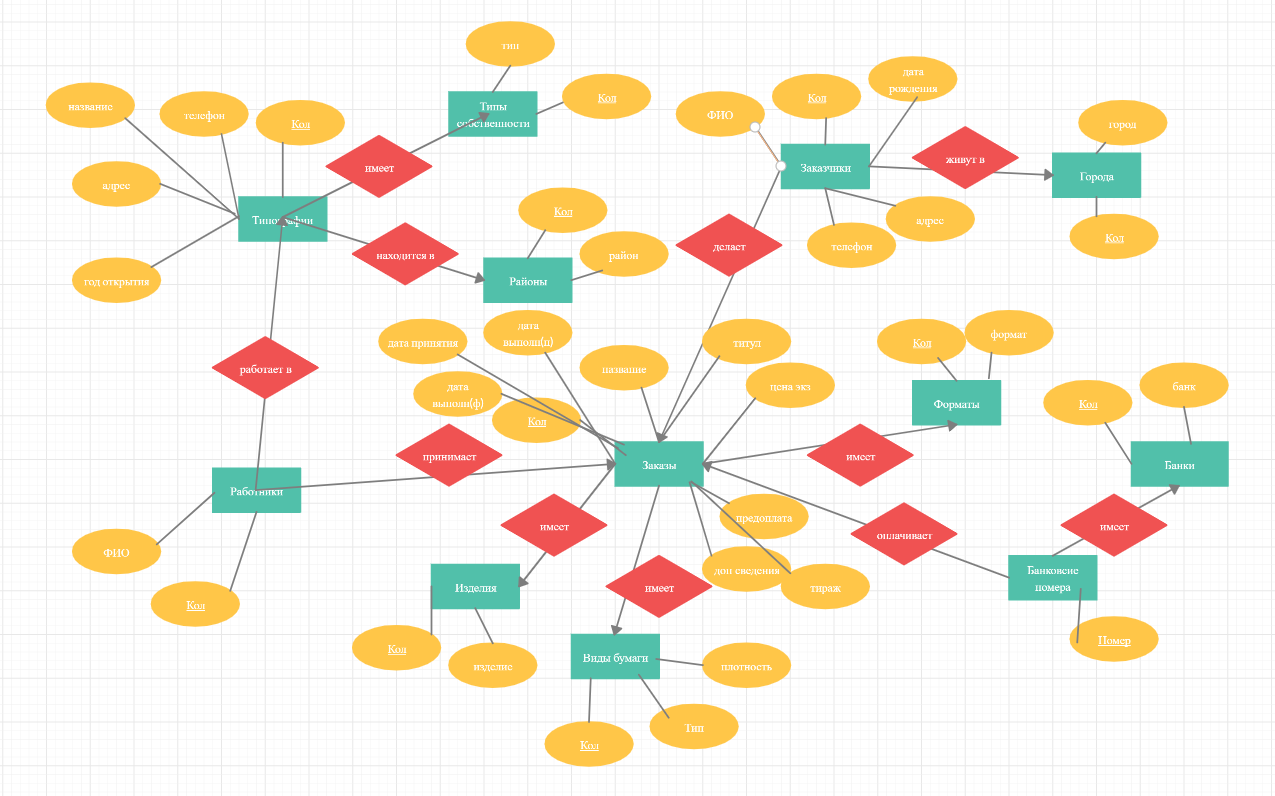


Рисунок 2.1 – ER диаграмма

2.2 Даталогическое проектирование

Даталогическое проектирование является проектированием логической структуры БД, что означает определение всех информационных единиц и связей между ними, задание их имен и типов, а также некоторых количественных характеристик (например, длины поля).

При проектировании логической структуры БД, осуществляется преобразование исходной инфологической модели в модель данных, поддерживаемую конкретной СУБД, и проверка адекватности полученной даталогической модели отображаемой предметной области.

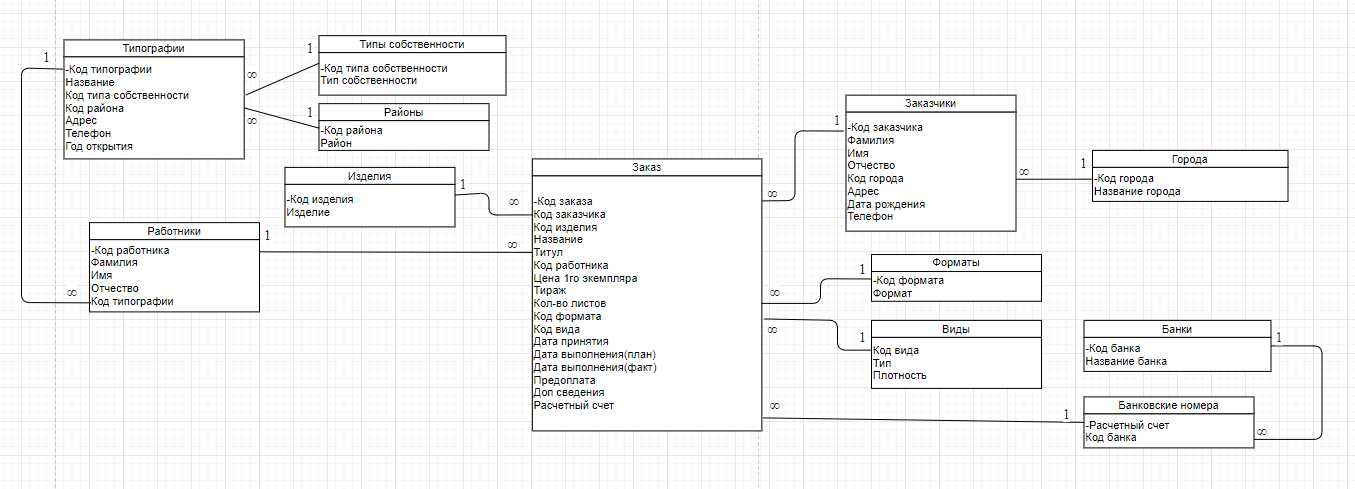


Рисунок 2.2 – логическая схема

2.3 Физическое проектирование

Физическое проектирование — создание [схемы базы данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) для конкретной [СУБД](https://ru.wikipedia.org/wiki/%D0%A1%D0%A3%D0%91%D0%94). Специфика конкретной СУБД может включать в себя ограничения на именование объектов базы данных, ограничения на поддерживаемые типы данных и т. п. Кроме того, специфика конкретной СУБД при физическом проектировании включает выбор решений, связанных с физической средой хранения данных (выбор методов управления дисковой памятью, разделение БД по файлам и устройствам, методов доступа к данным), создание индексов и т. д.[3]

С учетом возможностей и особенностей PostgreSQL было проведено физическое проектирования. В результате которого были выведены скрипты на языке SQL, которые описывают все ограничения, типы и связи между таблицами.(Рис 2.3-2.18)

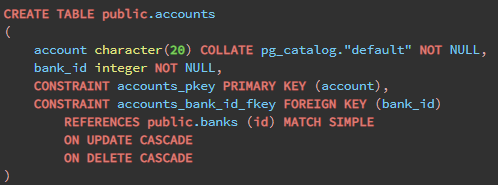


Рисунок 2.3 Срипт SQL описывающий таблицу банковские счета

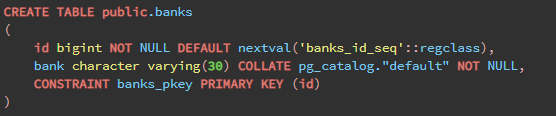


Рисунок 2.4 Срипт SQL описывающий таблицу банки

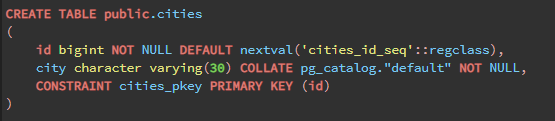


Рисунок 2.5 Срипт SQL описывающий таблицу города

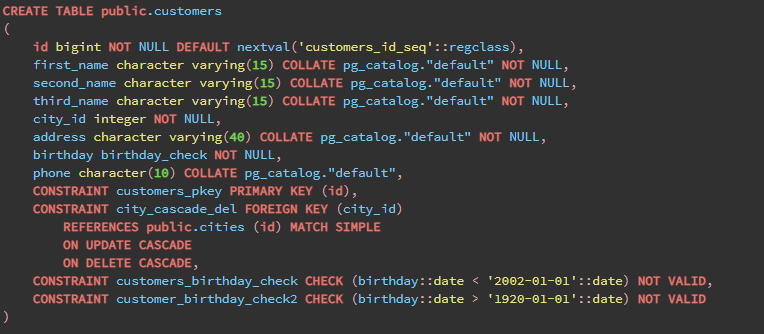


Рисунок 2.6 Срипт SQL описывающий таблицу заказчики

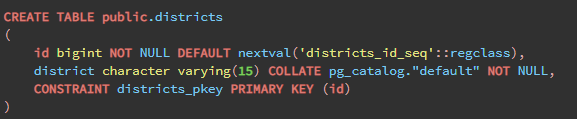


Рисунок 2.7 Срипт SQL описывающий таблицу районы

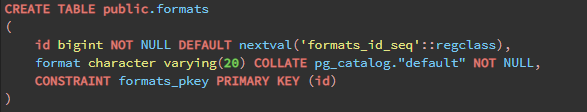


Рисунок 2.8 Срипт SQL описывающий таблицу форматы

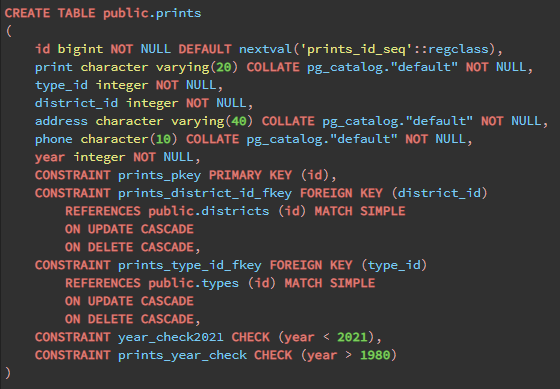


Рисунок 2.9 Срипт SQL описывающий таблицу типографии



Рисунок 2.10 Срипт SQL описывающий таблицу заказы

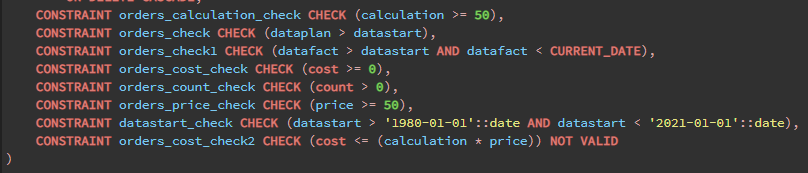


Рисунок 2.10, лист 2

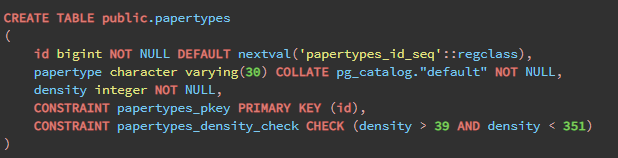


Рисунок 2.11 Срипт SQL описывающий таблицу виды бумаги

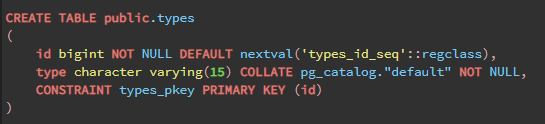


Рисунок 2.12 Срипт SQL описывающий таблицу типы собственности

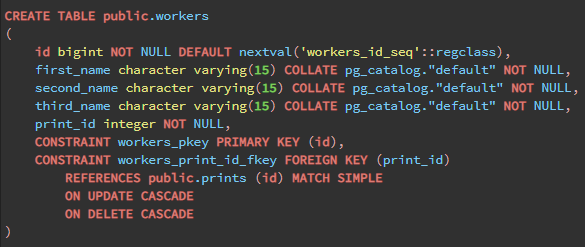


Рисунок 2.13 Срипт SQL описывающий таблицу работники

1. РАЗРАБОТКА ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ

3.1 Создание проекта

Для разработки клиентской части системы выбран язык C# и среда разработки Visual Studio 2019.

Процесс создания и подготовки проекта для работой с БД включает в себя. Создание нового проекта на языке C#. Подключение библиотеки Npgsql для связывания серверной части системы с клиентской.

3.2 Создание БД

Для работы с сервером используется среда PG admin 4. В ней упрощены и автоматизированы некоторые взаимодействия с базой данных.

Для начала работы нужно авторизоваться. После этого создаем новую базу данных. После того как база данных создана можно приступать к созданию таблиц

3.3 Создание таблиц

При помощи инструмента Query Tool мы можем воспользоваться командой CREATE TABLE, которая позволяет создавать таблицы.

Благодаря физическому проектированию у нас уже есть код для создания каждой таблицы. Остается только запустить его. Важно что сначала создаются таблицы без внешних ключей, а уже потом таблицы, связанные с ними.

После создания таблиц база данных выглядит так(Рис 3.1)

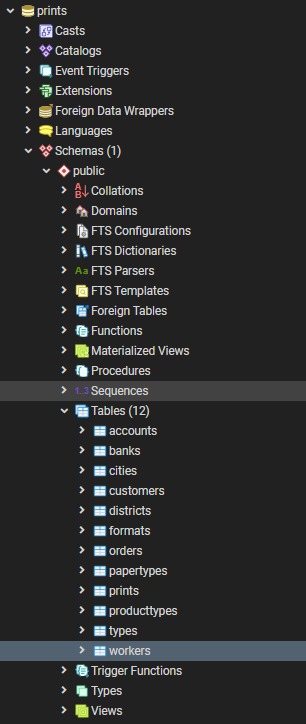
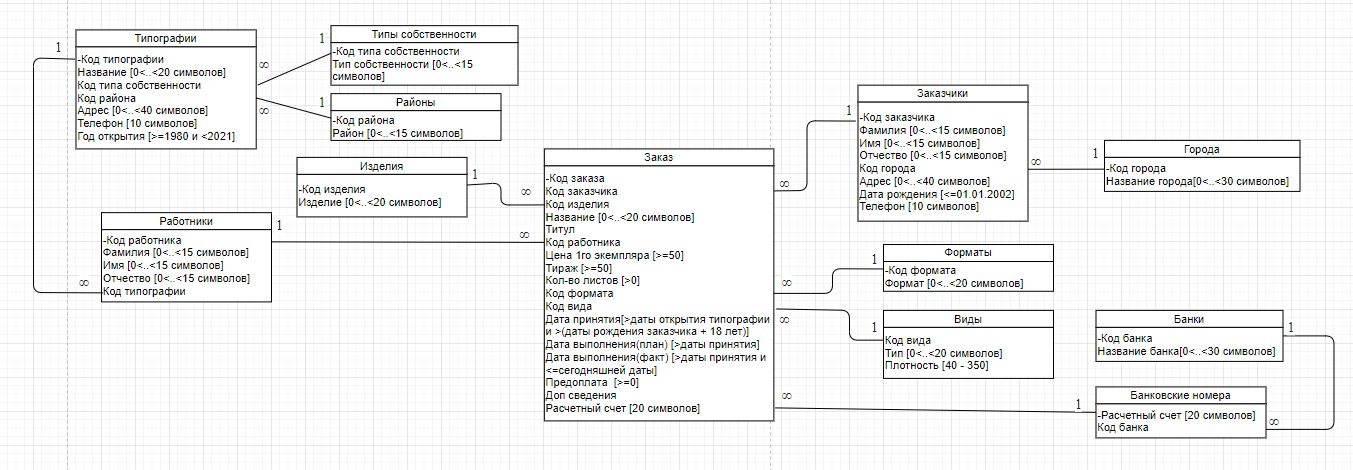


Рисунок 3.1 – База данных после создания таблиц

3.4 Схема БД

Схема БД создается чтобы сохранить целостность данных(Рис 3.2)

Рисунок 3.2 – схема БД



3.5 Разработка пользовательского интерфейса

При запуске программы пользователя встречает окно авторизации (Рис 3.3)

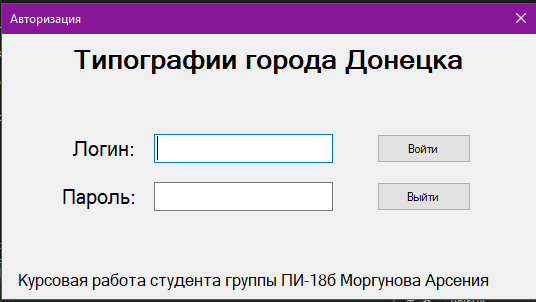


Рисунок 3.3 –форма авторизации

После авторизации пользователь попадает на главную форму, которая по умолчанию заполняется таблицей типографии (Рис 3.4)

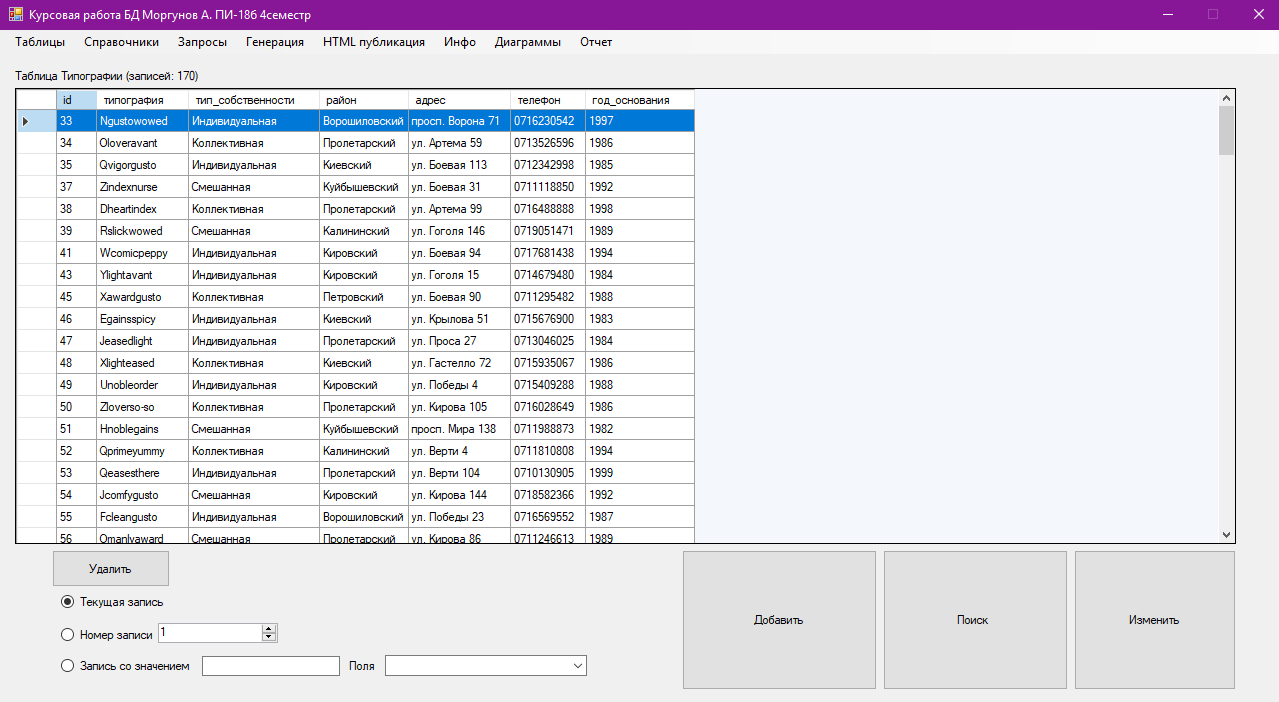


Рисунок 3.4 – Главная форма

Для выбора отображаемой таблицы в пунктах меню есть кнопки таблицы и справочники, при нажатии на которые появится выпадающий список таблиц и справочников соответственно (Рис 3.5-3.6)

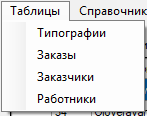


Рисунок 3.5 – Выпадающий список таблиц

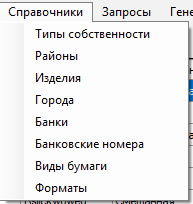


Рисунок 3.6 – Выпадающий список справочников

При нажатии на кнопку добавить появится форма с добавлением записи в ту таблицу, которая сейчас отображается на главной форме. Формы добавления представлены ниже(Рис 3.7-3.12)

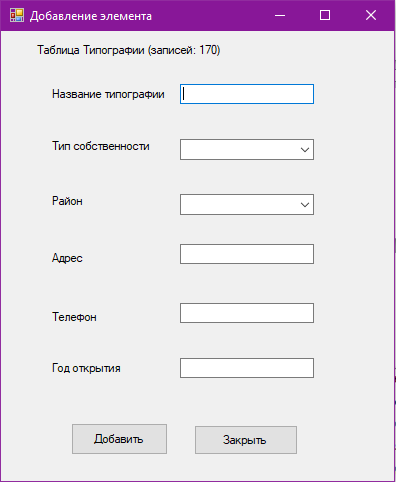


Рисунок 3.7 – Добавление в таблицу типографии

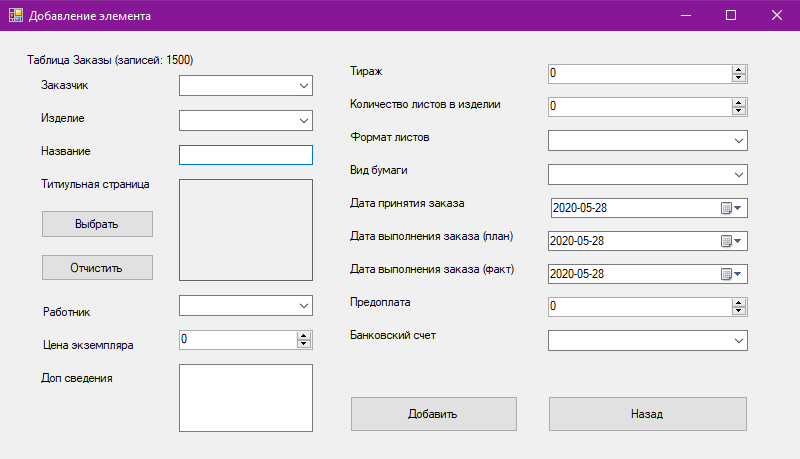


Рисунок 3.8 – Добавление в таблицу заказы

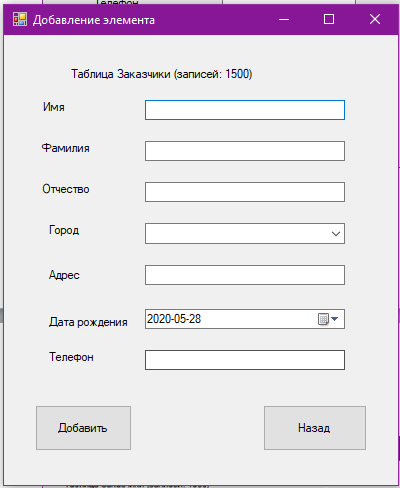


Рисунок 3.9 – Добавление в таблицу заказчики

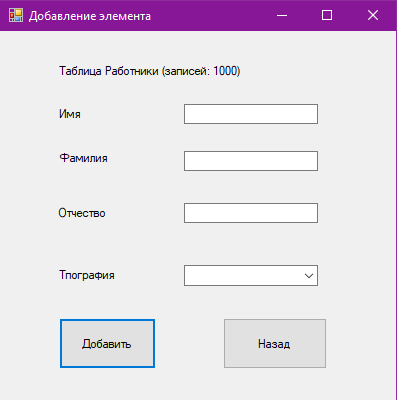


Рисунок 3.10 – Добавление в таблицу работники

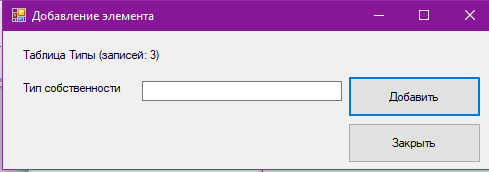


Рисунок 3.11 – Добавление в таблицу типы собственности

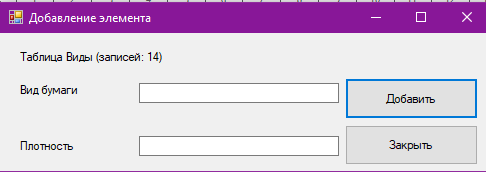


Рисунок 3.12 – Добавление в таблицу виды собственности

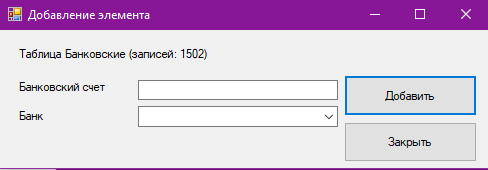


Рисунок 3.13 – Добавление в таблицу банковские счета

Для изменения записи нужно выделить ее на главной форме и нажать кнопку изменить. После этого появится форма добавления записей, но уже заполненная(Рис 3.14). Для изменения данных нужно нажать кнопку изменить.

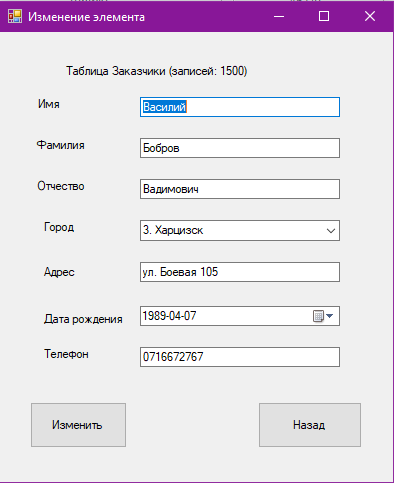


Рисунок 3.14 – Форма изменения записи таблицы заказчики

Удаление происходит с помощью кнопки удалить на главной форме. При этом можно выбрать желаемый тип удаления(Рис 3.15)

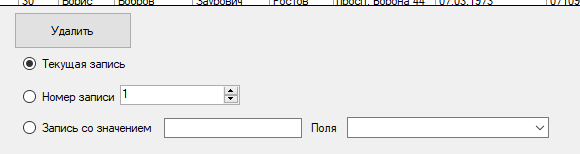


Рисунок 3.15 – Кнопка удалить и переключатели типа удаления

Реализованы запросы. Чтобы открыть форму запросов необходимо нажать на пункт верхнего меню запросы. Откроется форма, в которой можно выбрать запрос(Рис 3.16)

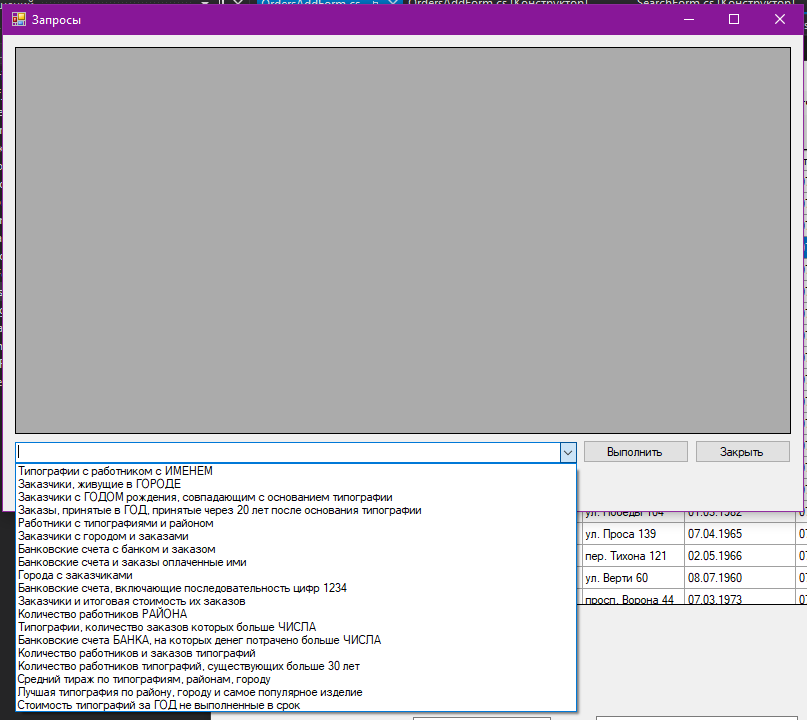


Рисунок 3.16 – Форма запросов и выбор запроса

Для того чтобы открыть диаграмму нужно нажать на пункт верхнего меню диаграммы и выбрать интересующую. Вид диаграмм(Рис 3.17-3.19)

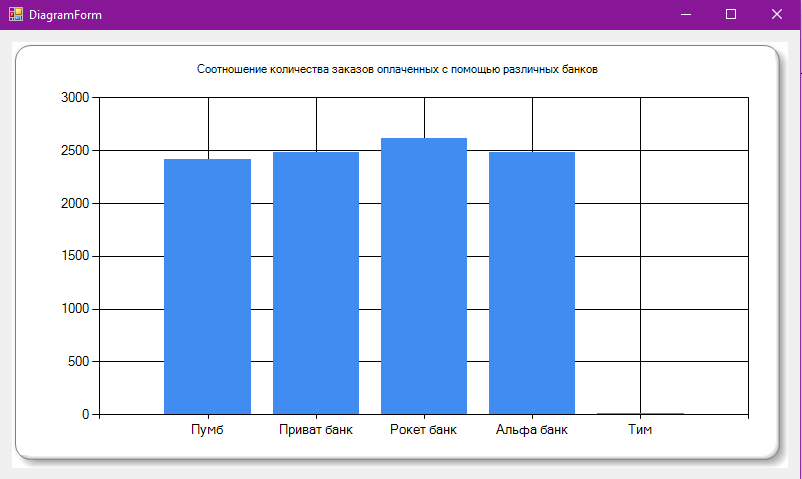


Рисунок 3.17 – Диаграмма количество заказов на банк

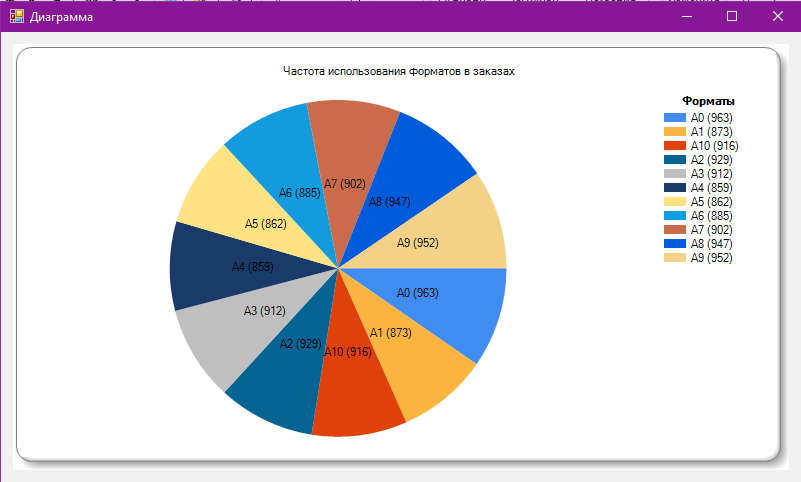


Рисунок 3.18 – Диаграмма частота использования форматов

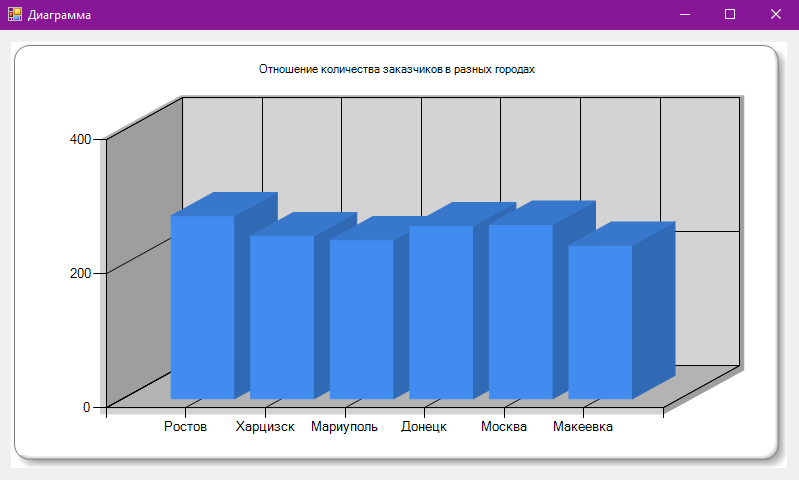


Рисунок 3.19 – Диаграмма количество заказчиков в городах

Для вызова html публикации нажмите на пункт верхнего меню html-публикация и выберите таблицу. Результат(Рис 3.20)



Рисунок 3.20 – html – публикация таблицы типографии

Также реализованы однотабличные (Рис 3.21) и многотабличные (Рис 3.22) Excel отчеты

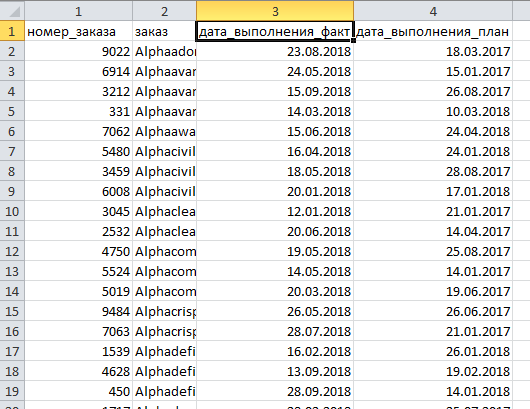


Рисунок 3.21 – Однотабличный отчет

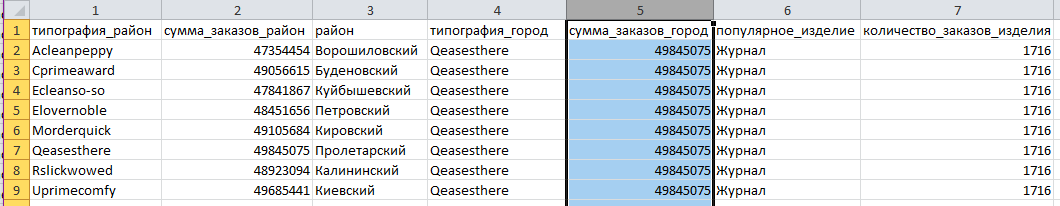


Рисунок 3.22 – Многотабличный отчет

3.6 Процедуры обработки данных

Для вывода информации на главную форму используется метод Select()(Рис.3.23)



Рисунок 3.23 – Метод Select()

Для экспорта в html(Рис.3.24) также создан отдельный метод

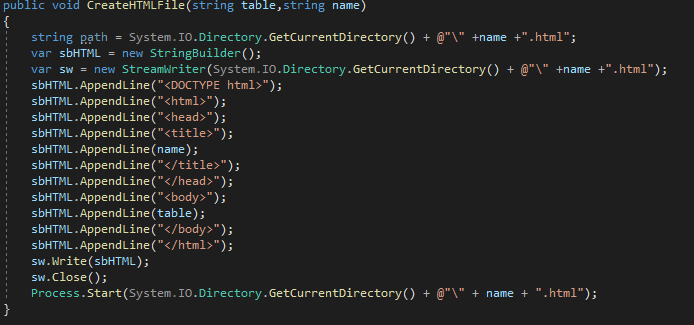


Рисунок 3.24 – Экспорт в html

Функциональность остальных возможностей системы заключается в комбинировании функционала отдельных методов и кода.

3.7 Тестирование и отладка БД и ИС

При вводе некорректных данных программа уведомляет пользователя и выдает ошибку(Рис 3.25)

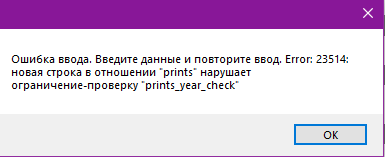


Рисунок 3.25 – неправильный ввод

Если пользователь ввел не все данные, то система оповестит его об этом(Рис 3.26).

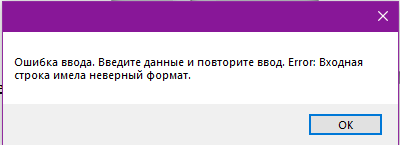


Рисунок 3.26 – неполный ввод

При попытке пользователя сгенерировать справочник система выдаст ошибку(Рис 3.27)

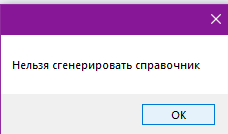


Рисунок 3.27 – ошибка при генерации справочника

3.8 Разработка эксплуатационной документации БД и ИС

В приложении B присутствует руководство пользователя, в котором описан функционал и взаимодействие с системой.

ВЫВОДЫ

Результат работы – система, реализующая систему для учета деятельности типографий города Донецка, реализованная при помощи языка программирования C# и системы управления базами данных PostgreSQL.

К преимуществам программы относятся: удобный пользовательский интерфейс, реализация и управление базой данных с помощью клиентского интерфейса, возможность добавлять, удалять, редактировать, выбирать данные из таблиц. Возможен экспорт данных в html и excel. Безопасность для пользователя от случайного удаления данных. Возможен вывод диаграмм со статистическими данными. Просмотр результатов запросов различной сложности.

Недостатки: при слишком большом количестве обрабатываемых записей заметны подтормаживания системы.

СПИСОК ЛИТЕРАТУРЫ

1. Основы технологий баз данных: учеб. пособие/ Б.А. Новиков, Е.А.Горшкова; под ред. Е.В. Рогова. – М.:ДМК Пресс, 2019. -240с
2. Инфологическая модель [Электронный ресурс] // Инфологическая модель. – режим доступа: <http://wiki.mvtom.ru/index.php/%D0%98%D0%BD%D1%84%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5>
3. Физическое проектирование [Электронный ресурс] // Физическое проектирование. – режим доступа: <https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%B1%D0%B0%D0%B7_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85#%D0%A4%D0%B8%D0%B7%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5>

# ПРИЛОЖЕНИЕ А

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

# ПРИЛОЖЕНИЕ Б

SQL-ЗАПРОСЫ

# Симметричное внутреннее соединение с условием:

1.Заказчики с **указанным годом** рождения, который совпадает с годом основания типографии

select

cust.first\_name as имя,

cust.second\_name as фамилия,

cust.third\_name as отчество,

extract(year from cust.birthday) as год\_рождения,

p.year as год\_основаия,

p.print as типография

from customers as cust,

orders as o,

workers as w,

prints as p

where cust.id = o.customer\_id and

o.worker\_id=w.id and

w.print\_id=p.id and

extract(year from cust.birthday)=p.year and

p.year = **ГОД**

order by p.print

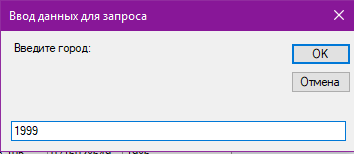


Рисунок Б.1 – Ввод данных для запроса 1

Запрос с параметром 1999

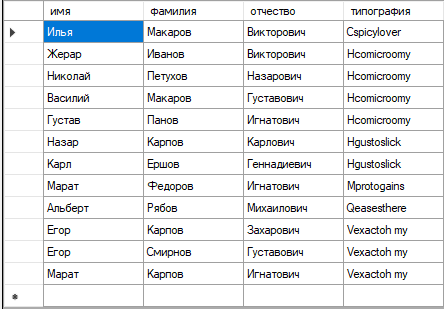


Рисунок Б.2 – Запрос 1

2.Заказы с **указанным годом** принятия заказа, принятые через 20 лет после основания типографии

select

o.id as номер\_заказа,

o.publication as название

from workers as w,

prints as p,

orders as o

where p.id=w.print\_id and

o.worker\_id=w.id and

(extract(year from o.datastart)-20)=p.year and

extract(year from o.datastart)=**ГОД**

order by p.year

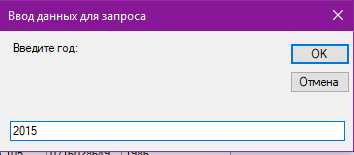


Рисунок Б.3 – Ввод данных для запроса 2

Запрос с параметром 2015

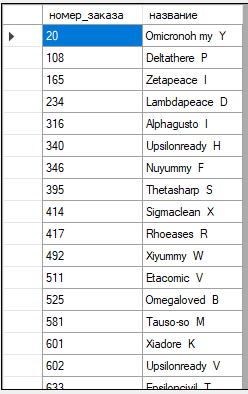


Рисунок Б.4 – Запрос 2

3.Заказчики, которые живут в **указанном городе**

select cust.first\_name as имя,

cust.second\_name as фамилия,

cust.third\_name as отчество,

city.city as город

from customers as cust,

cities as city

where cust.city\_id=city.id and

city.city=**ГОРОД**

order by 4

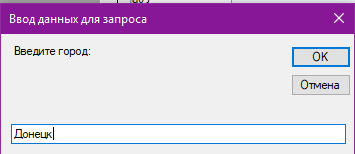


Рисунок Б.5 – Ввод данных для запроса 3

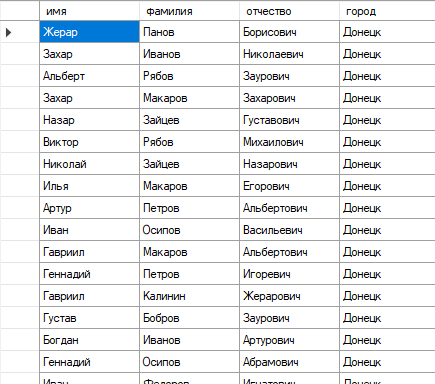
Запрос с параметром «Донецк» 

Рисунок Б.6 – Запрос 3

4.Типографии, в которых есть работник с **указанным** **именем**

select

p.print as типография,

count(w.first\_name) as количество\_работников\_с\_именем

from prints as p,

workers as w

where p.id=w.print\_id and

w.first\_name=**ИМЯ**

group by p.print

order by 1

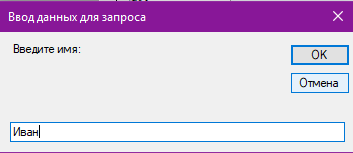


Рисунок Б.7 – Ввод данных для запроса 4

Запрос с параметром «Иван»

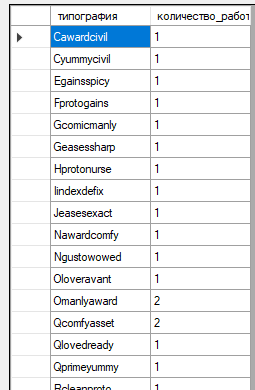


Рисунок Б.8 – Запрос 4

# **Симметричное внутреннее соединение без условия**:

1. Работники с типографиями в которых они работают и районом расположения типографии

select

w.first\_name as имя,

w.second\_name as фамилия,

w.third\_name as отчество,

p.print as типография ,

d.district as район

from workers as w,

prints as p,

districts as d

where p.id=w.print\_id and

p.district\_id=d.id

order by 1,2,3

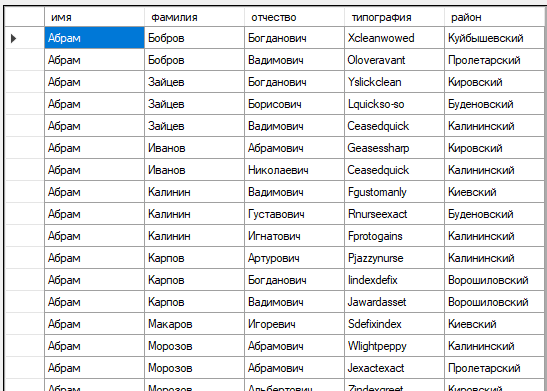


Рисунок Б.9 – Запрос 5

2. заказчики с городами проживания и количеством заказов

select cust.id,

cust.first\_name as имя

,cust.second\_name as фамилия,

cust.third\_name as отчество,

c.city as город,

count(o.\*) as количество\_заказов

from customers as cust,

cities as c,

orders as o

where cust.city\_id=c.id and

o.customer\_id=cust.id

group by 1,2,3,4,5

order by 6 desc

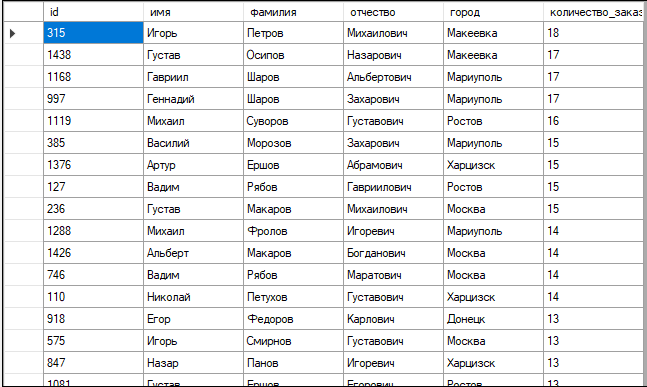


Рисунок Б.10 – Запрос 6

3.банковские номера с банком и количеством заказов оплаченных ими

select

a.account as банковский\_счет,

b.bank as банк,

count(o.\*) as количество\_заказов

from accounts as a,

banks as b,

orders as o

where a.account=o.account and

a.bank\_id = b.id

group by 1,2

order by 3 desc

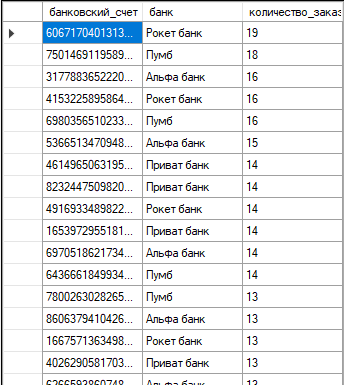


Рисунок Б.11 – Запрос 7

# Левое внешнее соединение

номера бановских счетов и заказы которые были оплачены ими

select

a.account as банковский\_счет,

o.id as номер\_заказа

from accounts as a

left join orders as o on o.account=a.account

order by 2 desc

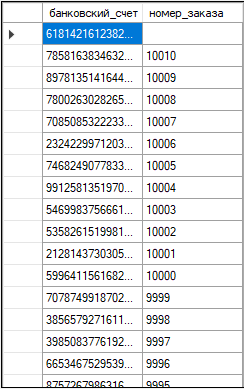


Рисунок Б.12 – Запрос 8

# Правое внешнее соединение

все города и заказчики в городах

select c.id,c.city as Город,

cust.first\_name as имя,

cust.second\_name as фамилия,

cust.third\_name as отчество

from customers as cust

right join cities as c on c.id=cust.city\_id

order by 1 desc



Рисунок Б.13 – Запрос 9

# Запрос на запросе по принципу левого соединения

информация о банковских счетах, которые включают в себя последовательность цифр 1234

select aa.account,

b.bank

from (select account,bank\_id from accounts where account like '%1234%') as aa

left join banks as b on aa.bank\_id=b.id

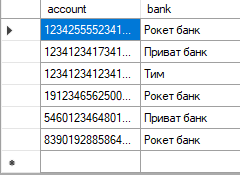


Рисунок Б.14 – Запрос 10

# Итоговый запрос без условия

Общая стоимость заказов каждого заказчика

select cust.id as номер\_заказчика,

cust.first\_name as имя,

cust.second\_name as фамилия,

cust.third\_name as отчество,

sum(o.cost\*o.calculation) as денег\_потрачено

from customers as cust,

orders as o

where o.customer\_id=cust.id

group by 1,2,3,4

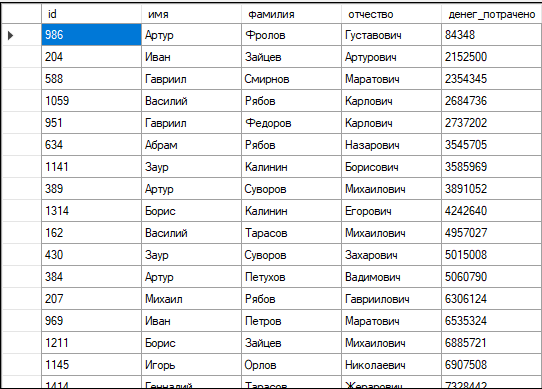
order by 5

Рисунок Б.15 – Запрос 11

# Итоговый запрос с условием на данные

Количество работников задействованных в типографиях данного **РАЙОНА**

select d.district as район,

count (w.\*) as работников

from prints as p,

workers as w,

districts as d

where d.id=p.district\_id and

p.id=w.print\_id and

d.district=**РАЙОН**

group by 1

order by 2

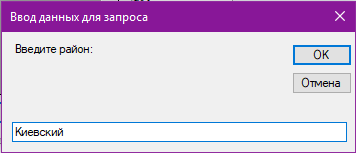


Рисунок Б.16 – Ввод данных для запроса 12

результат запроса с параметром: Киевский

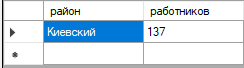


Рисунок Б.17 – Запрос 12

# Итоговый запрос с условием на группы

Типографии, количество заказов в которых больше **ЧИСЛА**

select p.id as номер\_типографии,

p.print as типография,

count(o.\*) as количество\_заказов

from prints as p,

workers as w,

orders as o

where p.id=w.print\_id and

o.worker\_id=w.id

group by 1,2

having(count(o.\*)>**ЧИСЛО**)

order by 3

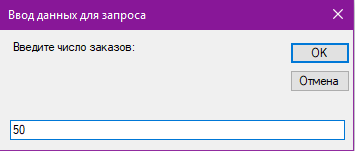


Рисунок Б.18 – Ввод данных для запроса 13

Результат запроса с параметром 50

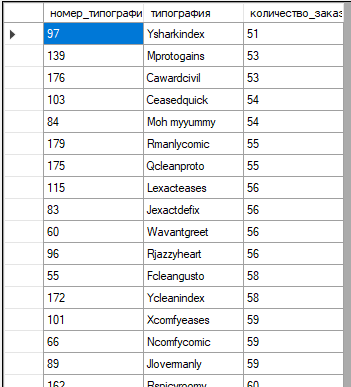


Рисунок Б.19 – Запрос 13

# Итоговый запрос с условием на данные и на группы

Банковские счета **БАНКА**, на которых денег потрачено больше **ЧИСЛА**

select a.account as банковский\_счет,

sum(o.cost\*o.calculation) as денег\_потрачено

from accounts as a,

banks as b,

orders as o

where o.account=a.account and

a.bank\_id = b.id and

b.bank=**БАНК**

group by 1

having (sum(o.cost\*o.calculation)>**ЧИСЛО**)

order by 2

запрос с параметрами Пумб , 100000000

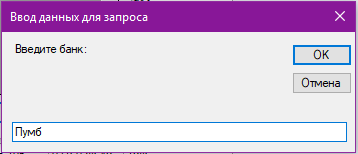


Рисунок Б.20 – Ввод данных для запроса 14

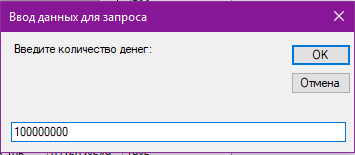


Рисунок Б.21 – Ввод данных для запроса 14

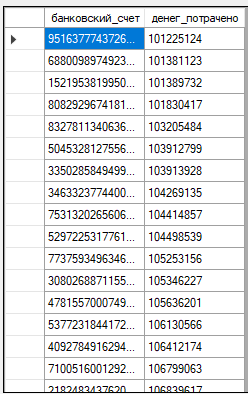


Рисунок Б.22 – Запрос 14

# Запрос на запросе по принципу итогового запроса

Количество работников и количество заказов всех типографий

select p.id as номер\_типографии,

p.print as типография ,

count (w.\*) as количество\_работников,

p.возраст,

p.год\_основания

from workers as w,

(select id,print,year as год\_основания,(extract(year from current\_date)-year) as возраст from prints where (extract(year from current\_date)-year>=30)) as p

where p.id=w.print\_id

group by 1,2,4,5

order by 4 desc,3

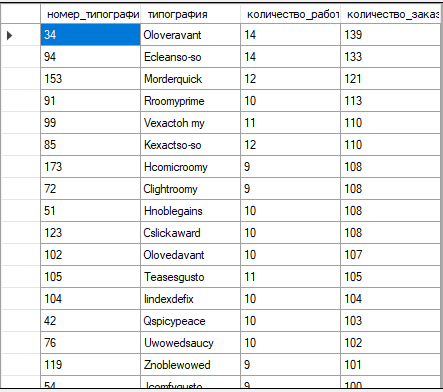


Рисунок Б.23 – Запрос 15

# Запрос с подзапросом

Количество работников , которые работают в типографиях, существующих более 30 лет

select p.id as номер\_типографии,

p.print as типография ,

count (w.\*) as количество\_работников,

p.возраст,

p.год\_основания

from workers as w,

(select id,print,year as год\_основания,(extract(year from current\_date)-year) as возраст from prints where (extract(year from current\_date)-year>=30)) as p

where p.id=w.print\_id

group by 1,2,4,5

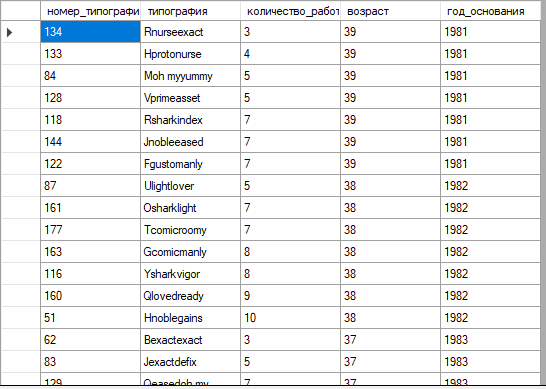
order by 4 desc,3

Рисунок Б.24 – Запрос 16

# ПРИЛОЖЕНИЕ В

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

При запуске пользователю необходимо авторизоваться.

После этого пользователь попадает на главную форму.

Для того, чтобы вывести любую таблицу на главную форму нужно нажать пункт меню таблицы и выбрать нужную таблицу в выпадающем списке.

Для того, чтобы вывести любой справочник на главную форму нужно нажать пункт меню справочники и выбрать нужный справочник в выпадающем списке.

Для того чтобы увидеть результаты запросов нужно нажать пункт меню запросы. В открывшейся форме выбрать запрос и нажать кнопку выполнить.

Для того, чтобы сгенерировать записи в таблицу выберите любую таблицу(не справочник) и нажмите кнопку генерация. Введите нужное число и подтвердите действие.

Для того чтобы сделать html публикацию нужно нажать на пункт меню html публикация и выбрать таблицу в выпадающем списке

Для того чтобы посмотреть на диаграммы нажмите на пункт меню диаграммы. И выберите из выпадающего списка.

Для того чтобы экспортировать запрос в Excel нужно нажать на пункт меню отчет и выбрать запрос в выпадающем списке

Для того чтобы добавить запись нужно выбрать таблицу и нажать кнопку добавить. После этого заполнить поля и нажать кнопку добавить.

Для того чтобы изменить запись нужно выбрать запись нажать кнопку изменить и изменить поля с данными на те, что вы хотите. После нажмите кнопку изменить.

Для того, чтобы выполнить поиск нужно нажать на кнопку поиск. Откроется форма поиска. В ней выберите нужное поле и введите значение для поиска

Для удаления записи выберите режим удаления. При текущей записи вы удалите выделенную запись. При номере записи вы ужалите запись с номером указанным в поле ввода. При записи со значением вы удалите поля со значениями равными указанным.

# ПРИЛОЖЕНИЕ Г

ТЕКСТ ПРОГРАММЫ

using Npgsql;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Reflection;

using Excel = Microsoft.Office.Interop.Excel;

using System.Web.UI;

using System.Web.UI.HtmlControls;

using System.IO;

using System.Diagnostics;

namespace Prints\_Of\_Donets\_KursovayaBD\_

{

public partial class MainForm : Form

{

private string connstring = String.Format("Server={0};Port={1};"

+ "User Id={2};Password={3};Database={4}", "localhost", 5432, "postgres", "sava000", "prints");

public NpgsqlConnection conn;

public string sql;

public string selectSql;

public NpgsqlCommand cmd;

public DataTable dt;

private int rowIndex = -1;

public MainForm()

{

InitializeComponent();

}

public bool FindTableType()//показывает тип таблицы показанной пользователю true - таблица false - справочник

{

string[] words = label1.Text.Split(' ');

switch (words[0])

{

case "Таблица":

{

return true;

break;

}

case "Справочник":

{

return false;

break;

}

}

return true;

}

public string FindTable() //показывает какая таблица показана пользователю

{

string[] words = label1.Text.Split(' ');

switch (words[1])

{

case "Типографии":

{

return "prints";

break;

}

case "Заказы":

{

return "orders";

break;

}

case "Заказчики":

{

return "customers";

break;

}

case "Работники":

{

return "workers";

break;

}

case "Банковские":

{

return "accounts";

break;

}

case "Типы":

{

return "types";

break;

}

case "Районы":

{

return "districts";

break;

}

case "Города":

{

return "cities";

break;

}

case "Банки":

{

return "banks";

break;

}

case "Изделия":

{

return "producttypes";

break;

}

case "Виды":

{

return "papertypes";

break;

}

case "Форматы":

{

return "formats";

break;

}

}

return "";

}

private void Form1\_Load(object sender, EventArgs e)//при загрузке главной формы

{

conn = new NpgsqlConnection(connstring);//присоединение к БД

//select

selectSql = @"select id,print as типография,(select type from types where id=type\_id) as тип\_собственности," +

"(select district from districts where id=district\_id) as район,address as адрес, phone as телефон, year as год\_основания from prints";

Select();

//select

//вывод количества записей и названия таблицы с которой работаем на форму

sql = @"select count(\*) from prints";

ChangeLabel("Таблица Типографии");

//конец вывод количества записей и названия таблицы с которой работаем на форму

//заполнение ComboBox

int i = 1;

cellComboBox.Items.Clear();

foreach (DataColumn item in dt.Columns)

{

if (CheckBytea(i - 1))

{

cellComboBox.Items.Add(i + ") " + item);

}

i++;

}

//заполнение ComboBox

}

public void ChangeLabel(string tableName)//записывает с какой таблицей работаем и сколько в ней записей

{

try

{

conn.Open();

cmd = new NpgsqlCommand(sql, conn);

label1.Text = tableName +" (записей: " + cmd.ExecuteScalar() + ")";

conn.Close();

}

catch (Exception ex)

{

conn.Close();

MessageBox.Show("Ошибка при вычислении количества записей. Error: " + ex.Message);

}

}

public void Select()

{

try

{

conn.Open();

cmd = new NpgsqlCommand(selectSql, conn);

dt = new DataTable();

dt.Load(cmd.ExecuteReader());

conn.Close();

dgvData.DataSource = null;

dgvData.DataSource = dt;

}

catch(Exception ex)

{

conn.Close();

MessageBox.Show("Error: " + ex.Message);

}

}

public string NameConverterForDelete(string column)

{

switch (column)

{

case "типография":

{

try

{

conn.Open();

cmd = new NpgsqlCommand(@"select count(\*) filter (where print\_id="+ ""+") from workers ", conn);

conn.Close();

}

catch (Exception ex)

{

conn.Close();

MessageBox.Show("Error: " + ex.Message);

}

return "print";

}

case "тип\_собственности":

return "type\_id";

case "год\_основания":

return "year";

default:

break;

}

return column;

}

public bool CheckBytea(int i)//проверка типа столбца на bytea

{

switch (dt.Columns[i].DataType.ToString())

{

case "System.Int64":

{

break;

}

case "System.Int32":

{

break;

}

case "System.String":

{

break;

}

case "System.DateTime":

{

break;

}

default:

{

return false;

}

break;

}

return true;

}

public string CheckComboBox(int i)//проверка типа вводимого значения для ComboBox на MainForm

{

//valueTextBox.Text = dt.Columns[i].DataType.ToString();

switch (dt.Columns[i].DataType.ToString())

{

case "System.Int64":

{

return valueTextBox.Text;

break;

}

case "System.Int32":

{

return valueTextBox.Text;

break;

}

case "System.String":

{

return "'"+ valueTextBox.Text+"'";

break;

}

case "System.DateTime":

{

return "'" + valueTextBox.Text + "'";

break;

}

default:

{

valueTextBox.Text = dt.Columns[i].DataType.ToString();

return "error";

}

break;

}

//return dt.Columns[i].DataType.ToString();

}

private void dgvData\_CellClick(object sender, DataGridViewCellEventArgs e)//при нажатии на строку в таблице

{

if (e.RowIndex >= 0)

{

rowIndex = e.RowIndex;

//nameTextBox.Text = dgvData.Rows[e.RowIndex].Cells["name"].Value.ToString();

}

}

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

{

PrintsAddForm addForm = new PrintsAddForm();

addForm.Visible = true;

//int result = 0;

//rowIndex = -1;

//try

//{

// conn.Open();

// sql = @"select \* from print\_insert(:\_name)";

// cmd = new NpgsqlCommand(sql, conn);

// // cmd.Parameters.AddWithValue("\_name", nameTextBox.Text);

// result = (int)cmd.ExecuteScalar();

// conn.Close();

// if (result == 1)

// {

// MessageBox.Show("Inserted succesfully");

// Select();

// }

// else

// {

// MessageBox.Show("Inserted fail");

// }

//}

//catch (Exception ex)

//{

// conn.Close();

// MessageBox.Show("Inserted fail. Error: " + ex.Message);

//}

}

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

{

//int result = 0;

//if (rowIndex < 0)

//{

// MessageBox.Show("Please choose student to update");

// return;

//}

//try

//{

// conn.Open();

// sql = @"select \* from print\_update(:\_id,:\_name)";

// cmd = new NpgsqlCommand(sql, conn);

// cmd.Parameters.AddWithValue("\_id", int.Parse(dgvData.Rows[rowIndex].Cells["id"].Value.ToString()));

// //cmd.Parameters.AddWithValue("\_name", nameTextBox.Text);

// result = (int)cmd.ExecuteScalar();

// conn.Close();

// if (result == 1)

// {

// MessageBox.Show("Updated succesfully");

// Select();

// }

// else

// {

// MessageBox.Show("Updated fail");

// }

//}

//catch (Exception ex)

//{

// conn.Close();

// MessageBox.Show("Update fail. Error: " + ex.Message);

//}

//result = 0;

}

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

{

if (rowIndex < 0)

{

MessageBox.Show("Выберите строку для удаления");

return;

}

try

{

conn.Open();

sql = @"select \* from st\_delete(:\_id)";

cmd = new NpgsqlCommand(sql, conn);

cmd.Parameters.AddWithValue("\_id", int.Parse(dgvData.Rows[rowIndex].Cells["id"].Value.ToString()));

if ((int)cmd.ExecuteScalar() == 1)

{

MessageBox.Show("Удалено успешно");

rowIndex = -1;

}

conn.Close();

Select();

}

catch (Exception ex)

{

conn.Close();

MessageBox.Show("Ошибка при удалении. Error: " + ex.Message);

}

}

private void MainForm\_FormClosed(object sender, FormClosedEventArgs e)//при закрытии главной формы

{

Application.Exit();

}

private void buttonSearch\_Click\_1(object sender, EventArgs e)

{

string[] words = label1.Text.Split(' ');

SearchForm searchForm;

switch (words[1])

{

case "Типографии":

{

searchForm = new SearchForm(dt,cellComboBox.Items, FindTable(),label1.Text,dgvData.Columns);

searchForm.Show();

break;

}

case "Заказы":

{

searchForm = new SearchForm(dt, cellComboBox.Items, FindTable(), label1.Text, dgvData.Columns);

searchForm.Show();

break;

}

case "Заказчики":

{

searchForm = new SearchForm(dt, cellComboBox.Items, FindTable(), label1.Text, dgvData.Columns);

searchForm.Show();

break;

}

case "Работники":

{

searchForm = new SearchForm(dt, cellComboBox.Items,FindTable(), label1.Text, dgvData.Columns);

searchForm.Show();

break;

}

default:

{

MessageBox.Show("Поиск выполняется только для таблиц");

break;

}

}

}

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

{

//select

selectSql = @"select id,print as типография,(select type from types where id=type\_id) as тип\_собственности,"+

"(select district from districts where id=district\_id) as район,address as адрес, phone as телефон, year as год\_основания from prints";

Select();

//select

//count

sql = @"select count(\*) from prints";

ChangeLabel("Таблица Типографии");

//count

//заполнение ComboBox

int i = 1;

cellComboBox.Items.Clear();

foreach (DataColumn item in dt.Columns)

{

if (CheckBytea(i-1))

{

cellComboBox.Items.Add(i + ") " + item);

}

i++;

}

//конец заполнение ComboBox

}

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

{

//select

selectSql = @"select id, customer\_id as индекс\_заказчика, (select producttype from producttypes where id=producttype\_id) as изделие," +

" publication as название\_издания,picture as титульная\_страница, worker\_id as индекс\_работника, price as цена\_экземпляра, " +

"calculation as тираж, count as количество\_листов, (select format from formats where id=format\_id) as формат, (select papertype from papertypes where id = papertype\_id) as вид\_бумаги" +

", datastart as дата\_принятия, dataplan as дата\_выполнения\_план, datafact as дата\_выполнения\_факт, cost as предоплата, comment as доп\_сведения" +

",account as банковский\_номер from orders limit 1000";

Select();

//select

//count

sql = @"select count(\*) from orders";

ChangeLabel("Таблица Заказы");

//count

//заполнение ComboBox

int i = 1;

cellComboBox.Items.Clear();

foreach (DataColumn item in dt.Columns)

{

if (CheckBytea(i-1))

{

cellComboBox.Items.Add(i + ") " + item);

}

i++;

}

//заполнение ComboBox

}

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

{

//select

selectSql = @"select id,district as Район from districts";

Select();

//select

cellComboBox.Items.Clear();

//count

sql = @"select count(\*) from districts";

ChangeLabel("Справочник Районы");

//count

}

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

{

//select

selectSql = @"select id,type as Тип\_Собственности from types";

Select();

//select

cellComboBox.Items.Clear();

//count

sql = @"select count(\*) from types";

ChangeLabel("Справочник Типы собственности");

//count

}

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

{

//select

selectSql = @"select id,producttype as Изделие from producttypes";

Select();

//select

cellComboBox.Items.Clear();

//count

sql = @"select count(\*) from producttypes";

ChangeLabel("Справочник Изделия");

//count

}

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

{

//select

selectSql = @"select id,city as Город from cities";

Select();

//select

cellComboBox.Items.Clear();

//count

sql = @"select count(\*) from cities";

ChangeLabel("Справочник Города");

//count

}

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

{

//select

selectSql = @"select id,bank as Банк from banks";

Select();

//select

cellComboBox.Items.Clear();

//count

sql = @"select count(\*) from banks";

ChangeLabel("Справочник Банки");

//count

}

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

{

//select

selectSql = @"select \* from accounts\_select()";

Select();

//select

cellComboBox.Items.Clear();

//count

sql = @"select count(\*) from accounts";

ChangeLabel("Справочник Банковские номера");

//count

}

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

{

//select

selectSql = @"select id,papertype as Вид\_бумаги,density from papertypes";

Select();

//select

cellComboBox.Items.Clear();

//count

sql = @"select count(\*) from papertypes";

ChangeLabel("Справочник Виды бумаги");

//count

}

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

{

//select

selectSql = @"select id,format as Формат from formats";

Select();

//select

cellComboBox.Items.Clear();

//count

sql = @"select count(\*) from formats";

ChangeLabel("Справочник Форматы");

//count

}

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

{

//select

selectSql = @"select id,first\_name as имя, second\_name as фамилия, third\_name as отчество,(select city from cities where id=city\_id) as город," +

"address as адрес, birthday as дата\_рождения, phone as телефон from customers";

Select();

//select

//count

sql = @"select count(\*) from customers";

ChangeLabel("Таблица Заказчики");

//count

//заполнение ComboBox

int i = 1;

cellComboBox.Items.Clear();

foreach (DataColumn item in dt.Columns)

{

if (CheckBytea(i - 1))

{

cellComboBox.Items.Add(i + ") " + item);

}

i++;

}

//конец заполнение ComboBox

}

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

{

//select

selectSql = @"select id, first\_name as имя, second\_name as фамилия, third\_name as отчество,(select print from prints where id = print\_id) as типография from workers";

Select();

//select

//count

sql = @"select count(\*) from workers";

ChangeLabel("Таблица Работники");

//count

//заполнение ComboBox

int i = 1;

cellComboBox.Items.Clear();

foreach (DataColumn item in dt.Columns)

{

if (CheckBytea(i - 1))

{

cellComboBox.Items.Add(i + ") " + item);

}

i++;

}

//конец заполнение ComboBox

}

private void insertButton\_Click\_1(object sender, EventArgs e)

{

string[] words = label1.Text.Split(' ');

switch (words[1])

{

case "Типографии":

{

//return "prints";

PrintsAddForm printsAdd = new PrintsAddForm();

printsAdd.isUpdate = false;

printsAdd.Visible = true;

//MessageBox.Show(""+rowIndex);

break;

}

case "Заказы":

{

//return "orders";

OrdersAddForm ordersAdd = new OrdersAddForm();

ordersAdd.isUpdate = false;

ordersAdd.Visible = true;

break;

}

case "Заказчики":

{

//return "customers";

CustomersAddForm customersAdd = new CustomersAddForm();

customersAdd.isUpdate = false;

customersAdd.Visible = true;

break;

}

case "Работники":

{

//return "workers";

WorkersAddForm workersAdd = new WorkersAddForm();

workersAdd.isUpdate = false;

workersAdd.Visible = true;

break;

}

case "Банковские":

{

//return "accounts";

DirectoryAddForm directoryAdd = new DirectoryAddForm(true,true,false,"Банковский счет","Банк","", words[1], FindTable(),0,"");

directoryAdd.isUpdate = false;

directoryAdd.Visible = true;

directoryAdd.directory = FindTable();

break;

}

case "Типы":

{

//return "types";

DirectoryAddForm directoryAdd = new DirectoryAddForm(true, false, false, "Тип собственности", "", "", words[1], FindTable(), 0, "");

directoryAdd.isUpdate = false;

directoryAdd.Visible = true;

directoryAdd.directory = FindTable();

break;

}

case "Районы":

{

//return "districts";

DirectoryAddForm directoryAdd = new DirectoryAddForm(true, false, false, "Район", "", "", words[1], FindTable(), 0, "");

directoryAdd.isUpdate = false;

directoryAdd.Visible = true;

directoryAdd.directory = FindTable();

break;

}

case "Города":

{

//return "cities";

DirectoryAddForm directoryAdd = new DirectoryAddForm(true, false, false, "Город", "", "", words[1], FindTable(), 0, "");

directoryAdd.isUpdate = false;

directoryAdd.Visible = true;

directoryAdd.directory = FindTable();

break;

}

case "Банки":

{

//return "banks";

DirectoryAddForm directoryAdd = new DirectoryAddForm(true, false, false, "Банк", "", "", words[1], FindTable(), 0, "");

directoryAdd.isUpdate = false;

directoryAdd.Visible = true;

directoryAdd.directory = FindTable();

break;

}

case "Изделия":

{

//return "producttypes";

DirectoryAddForm directoryAdd = new DirectoryAddForm(true, false, false, "Изделие", "", "", words[1], FindTable(), 0, "");

directoryAdd.isUpdate = false;

directoryAdd.Visible = true;

directoryAdd.directory = FindTable();

break;

}

case "Виды":

{

//return "papertypes";

DirectoryAddForm directoryAdd = new DirectoryAddForm(true, false, true, "Вид бумаги", "", "Плотность", words[1], FindTable(), 0, "");

directoryAdd.isUpdate = false;

directoryAdd.Visible = true;

directoryAdd.directory = FindTable();

break;

}

case "Форматы":

{

//return "formats";

DirectoryAddForm directoryAdd = new DirectoryAddForm(true, false, false, "Формат", "", "", words[1], FindTable(), 0, "");

directoryAdd.isUpdate = false;

directoryAdd.Visible = true;

directoryAdd.directory = FindTable();

break;

}

}

}

private void deleteButton\_Click\_1(object sender, EventArgs e)

{

if (radioButton1.Checked == true)//если помечено удаление текущей записи

{

if (rowIndex < 0)//если не выбрана запись

{

MessageBox.Show("Выберите типографию которую хотите удалить");

return;

}

try

{

conn.Open();

//проверка каскадного удаления для таблицы работники

cmd = new NpgsqlCommand("select count(\*) from workers as w, prints as p where p.id=w.print\_id and p.id=" +

dgvData.Rows[rowIndex].Cells["id"].Value.ToString(), conn);

string a = "";

a += "Из таблицы работники будет удалено "+cmd.ExecuteScalar()+" записей. ";

//конец проверка каскадного удаления для таблицы работники

//проверка каскадного удаления для таблицы заказы

cmd = new NpgsqlCommand("select count(\*) from workers as w, prints as p,orders as o where p.id=w.print\_id and w.id=o.worker\_id and p.id="+

dgvData.Rows[rowIndex].Cells["id"].Value.ToString(), conn);

a+= "Из таблицы заказы будет удалено " + cmd.ExecuteScalar() + " записей.";

MessageBox.Show(a);

//конец проверка каскадного удаления для таблицы заказы

//удаление записи

//sql = @"delete from prints where id=" + dgvData.Rows[rowIndex].Cells["id"].Value.ToString();

//cmd = new NpgsqlCommand(sql, conn);

//dt.Load(cmd.ExecuteReader());

//MessageBox.Show("Удаление успешно");

//конец удаление записи

conn.Close();

rowIndex = -1;//переход к состоянию "ни одна запись не выбрана"

//select

selectSql = @"select id,print as типография,(select type from types where id=type\_id) as тип\_собственности," +

"(select district from districts where id=district\_id) as район,address as адрес, phone as телефон, year as год\_основания from prints";

Select();

//select

}

catch (Exception ex)

{

conn.Close();

MessageBox.Show("Ошибка при удалении. Error: " + ex.Message);

}

}

else if (radioButton2.Checked == true)//если помечено удаление по номеру записи

{

try

{

conn.Open();

//удаление записи по id

sql = @"delete from prints where id=" + numericUpDown1.Value;

cmd = new NpgsqlCommand(sql, conn);

dt.Load(cmd.ExecuteReader());

MessageBox.Show("Удаление успешно");

//конец удаление записи по id

conn.Close();

//select

selectSql = @"select id,print as типография,(select type from types where id=type\_id) as тип\_собственности," +

"(select district from districts where id=district\_id) as район,address as адрес, phone as телефон, year as год\_основания from prints";

Select();

//select

}

catch (Exception ex)

{

conn.Close();

MessageBox.Show("Ошибка при удалении. Error: " + ex.Message);

}

}

else//если помечено удаление со значением поля

{

try

{

conn.Open();

//удаление поля со значением

MessageBox.Show("delete from prints where " + NameConverterForDelete( dt.Columns[cellComboBox.SelectedIndex].ColumnName) +

"=" + CheckComboBox(cellComboBox.SelectedIndex));

sql = @"delete from prints where " + NameConverterForDelete(dt.Columns[cellComboBox.SelectedIndex].ColumnName) +

"=" + CheckComboBox(cellComboBox.SelectedIndex);

cmd = new NpgsqlCommand(sql, conn);

dt.Load(cmd.ExecuteReader());

MessageBox.Show("Удаление успешно");

//конец удаление поля со значением

conn.Close();

CheckComboBox(cellComboBox.SelectedIndex);

//select

selectSql = @"select id,print as типография,(select type from types where id=type\_id) as тип\_собственности," +

"(select district from districts where id=district\_id) as район,address as адрес, phone as телефон, year as год\_основания from prints";

Select();

//select

}

catch (Exception ex)

{

conn.Close();

MessageBox.Show("Ошибка при удалении. Error: " + ex.Message);

}

}

}

private void генерацияToolStripMenuItem\_Click(object sender, EventArgs e)//----------------------------генерация--------------------------------

{

string[] streets = { "просп. Мира", "ул. Артема", "ул. Кирова", "ул. Победы", "ул. Проса", "ул. Верти", "просп. Ворона",

"пер. Тихона","ул. Боевая", "ул. Гоголя", "ул. Гастелло", "ул. Крылова" };//улицы для случайной генерации

string[] first\_names = { "Борис", "Абрам", "Альберт", "Богдан", "Вадим", "Василий", "Гавриил", "Виктор", "Густав", "Геннадий", "Егор", "Игорь", "Заур", "Игнат", "Иван", "Жерар"

,"Захар","Илья","Карл","Назар","Николай","Артур","Михаил","Марат"};//имена

string[] second\_names = { "Рябов", "Смирнов", "Иванов", "Петров", "Морозов", "Зайцев", "Тарасов", "Федоров", "Орлов", "Бобров", "Фролов", "Ершов", "Карпов", "Шаров", "Панов",

"Суворов", "Осипов", "Петухов", "Калинин", "Макаров" };//фамилии

string[] third\_names = { "Борисович", "Абрамович", "Альбертович", "Богданович", "Вадимович", "Васильевич", "Гавриилович", "Викторович", "Густавович", "Геннадиевич",

"Егорович", "Игоревич", "Заурович", "Игнатович", "Иванович", "Жерарович","Захарович","Карлович","Назарович","Николаевич","Артурович","Михаилович","Маратович"};//отчества

string[] alphabet = { "A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z" };//Англ алфавит

string[] greekAlphabet = { "Alpha", "Beta", "Gamma", "Delta", "Epsilon", "Zeta", "Eta", "Theta", "Kappa", "Lambda",

"Mu", "Nu", "Xi", "Omicron", "Pi", "Rho", "Sigma", "Tau", "Upsilon", "Omega"};//греческий алфавит

string[] nameParts = { "ta-da", "so-so", "ready", "loved", "lover", "order", "slick", "sharp", "shark", "civil", "wowed", "eased", "adore", "roomy", "oh my"

, "gusto", "peppy", "vigor", "comic", "dress", "heart", "light", "exact", "index", "peace", "noble", "eases","gains", "award", "quick", "prime"

, "eases", "there", "yummy", "crisp", "jazzy", "saucy", "greet", "proto", "clean", "avant", "defix", "asset", "comfy", "spicy", "nurse", "manly"};//части названий

string[] words = label1.Text.Split(' ');//проверка названия таблицы

var random = new Random();

switch (words[1])

{

case "Банковские":

{

string result = Microsoft.VisualBasic.Interaction.InputBox("Сколько записей сгенерировать:", "Генерация справочника банковские номера", "");

if (result != "")//если нажали отмена и строка не пустая

{

//формирование sql запроса insert

if (true)

{

sql = @"insert into accounts( account,bank\_id) values";

sql += "('" + random.Next(11111, 99999) + random.Next(11111, 99999) +

random.Next(11111, 99999) + random.Next(11111, 99999) + "'," + random.Next(1, 5) + ")";

for (int i = 1; i < int.Parse(result); i++)

{

sql += ",('" + random.Next(11111, 99999) + random.Next(11111, 99999) +

random.Next(11111, 99999) + random.Next(11111, 99999) + "'," + random.Next(1, 5) + ")";

}

}

//конец формирование sql запроса insert

try

{

conn.Open();

//запрос insert

cmd = new NpgsqlCommand(sql, conn);

dt.Load(cmd.ExecuteReader());

//конец запрос insert

conn.Close();

//select

selectSql = @"select account as банковский\_номер, (select bank from banks where id=bank\_id) as bank from accounts";

Select();

//select

}

catch (Exception ex)

{

conn.Close();

MessageBox.Show("Ошибка при удалении. Error: " + ex.Message);

}

}

break;

}

case "Типографии":

{

string result = Microsoft.VisualBasic.Interaction.InputBox("Сколько записей сгенерировать:", "Генерация таблицы Типографии", "");

if (result != "")//если нажали отмена и строка не пустая

{

//формирование sql запроса insert

if (true)

{

sql = "INSERT INTO prints( print, type\_id, district\_id, address, phone, year) values" +

"('" +

alphabet[random.Next(0, 26)] + nameParts[random.Next(1, nameParts.Length)] + nameParts[random.Next(1, nameParts.Length)] + "', " +//print

random.Next(1, 4) + ", " +//type id

random.Next(1, 9) + ", '" +//district id

streets[random.Next(0, streets.Length)] + " " + random.Next(1, 150) + "', " +//address

"'071" + random.Next(0, 10) + random.Next(0, 10) + random.Next(0, 10) + random.Next(0, 10) + random.Next(0, 10) + random.Next(0, 10) + random.Next(0, 10) + "', " +//phone

random.Next(1981, 2000) + ")";//year

for (int i = 2; i < int.Parse(result) + 1; i++)

{

sql += ",('" +

alphabet[random.Next(0, 26)] + nameParts[random.Next(1, nameParts.Length)] + nameParts[random.Next(1, nameParts.Length)] + "', " +//print

random.Next(1, 4) + ", " +//type id

random.Next(1, 9) + ", '" +//district id

streets[random.Next(0, streets.Length)] + " " + random.Next(1, 150) + "', " +//address

"'071" + random.Next(0, 10) + random.Next(0, 10) + random.Next(0, 10) + random.Next(0, 10) + random.Next(0, 10) + random.Next(0, 10) + random.Next(0, 10) + "', " +//phone

random.Next(1981, 2000) + ")";//year

}

}

//конец формирование sql запроса insert

try

{

conn.Open();

//запрос insert

sql = @"" + sql + ";";

cmd = new NpgsqlCommand(sql, conn);

dt.Load(cmd.ExecuteReader());

//конец запрос insert

conn.Close();

//select

selectSql = @"select id,print as типография,(select type from types where id=type\_id) as тип\_собственности," +

"(select district from districts where id=district\_id) as район, address as адрес,phone as телефон, year as год from prints";

Select();

//select

//count

sql = "select count(\*) from prints";

ChangeLabel("Таблица Заказы");

//count

MessageBox.Show("Генерация успешно выполнена");

}

catch (Exception ex)

{

conn.Close();

MessageBox.Show("Ошибка при генерации. Error: " + ex.Message);

}

}

break;

}

case "Заказы":

{

string[] acc = { "a" };//номера аккаунтов

DataTable dataTable;//промежуточная таблица данных

int[] customers= { 0};//массив индексов пользователей

int[] producttypes = { 0 };//массив видов изделий

int[] workers = { 0 };//массив работников

int[] formats = { 0 };//форматов

int[] papertypes = { 0 };//видов бумаги

int o;//счетчик для циклов foreach

string result = Microsoft.VisualBasic.Interaction.InputBox("Сколько записей сгенерировать?", "Генерация таблицы Заказы", "");

if (result != "")//если нажали ок(и строка не пустая)

{

try

{

if (true)//заполнение массивов с индексами

{

conn.Open();

//вытягивание аккаунотов из справочника аккаунты

dataTable = new DataTable();

sql = @"select account from accounts";

cmd = new NpgsqlCommand(sql, conn);

dataTable.Load(cmd.ExecuteReader());

acc = new string[dataTable.Rows.Count];

o = 0;

foreach (DataRow dr in dataTable.Rows)

{

acc[o] = dr.ItemArray[0].ToString();

//MessageBox.Show("acc["+i+"]: "+acc[i]);

o += 1;

}

//конец вытягивание аккаунотов из справочника аккаунты

//вытягивание id из таблицы заказчики

cmd = new NpgsqlCommand("select id from customers", conn);

dataTable = new DataTable();

dataTable.Load(cmd.ExecuteReader());

customers = new int[dataTable.Rows.Count];

o = 0;

foreach (DataRow dr in dataTable.Rows)

{

customers[o] = int.Parse(dr.ItemArray[0].ToString());

//MessageBox.Show("customers[" + o + "]: " + customers[o]);

o += 1;

}

//конец вытягивание id из таблицы заказчики

//вытягивание id из таблицы изделия

cmd = new NpgsqlCommand("select id from producttypes", conn);

dataTable = new DataTable();

dataTable.Load(cmd.ExecuteReader());

producttypes = new int[dataTable.Rows.Count];

o = 0;

foreach (DataRow dr in dataTable.Rows)

{

producttypes[o] = int.Parse(dr.ItemArray[0].ToString());

//MessageBox.Show("producttypes[" + o + "]: " + producttypes[o]);

o += 1;

}

//конец вытягивание id из таблицы изделия

//вытягивание id из таблицы работники

cmd = new NpgsqlCommand("select id from workers", conn);

dataTable = new DataTable();

dataTable.Load(cmd.ExecuteReader());

workers = new int[dataTable.Rows.Count];

o = 0;

foreach (DataRow dr in dataTable.Rows)

{

workers[o] = int.Parse(dr.ItemArray[0].ToString());

//MessageBox.Show("worekrs[" + o + "]: " + workers[o]);

o += 1;

}

//конец вытягивание id из таблицы работники

//вытягивание id из таблицы форматы

cmd = new NpgsqlCommand("select id from formats", conn);

dataTable = new DataTable();

dataTable.Load(cmd.ExecuteReader());

formats = new int[dataTable.Rows.Count];

o = 0;

foreach (DataRow dr in dataTable.Rows)

{

formats[o] = int.Parse(dr.ItemArray[0].ToString());

//MessageBox.Show("formats[" + o + "]: " + formats[o]);

o += 1;

}

//конец вытягивание id из таблицы форматы

//вытягивание id из таблицы виды бумаги

cmd = new NpgsqlCommand("select id from papertypes", conn);

dataTable = new DataTable();

dataTable.Load(cmd.ExecuteReader());

papertypes = new int[dataTable.Rows.Count];

o = 0;

foreach (DataRow dr in dataTable.Rows)

{

papertypes[o] = int.Parse(dr.ItemArray[0].ToString());

//MessageBox.Show("papertypes[" + o + "]: " + papertypes[o]);

o += 1;

}

//конец вытягивание id из таблицы виды бумаги

conn.Close();

}//заполнение массивов с индексами

}

catch (Exception ex)

{

conn.Close();

MessageBox.Show("Deleted fail. Error: " + ex.Message);

}

//формирование sql запроса insert

if (true)

{

sql = "INSERT INTO orders(customer\_id, producttype\_id, publication, worker\_id, price, calculation, count, format\_id, papertype\_id, datastart, dataplan, datafact, cost, account) VALUES" +

"(" + " " +//id

customers[random.Next(0, customers.Length)] + ", " +//customer id

producttypes[random.Next(0, producttypes.Length)] + ", '" +//producttype id

greekAlphabet[random.Next(0, greekAlphabet.Length)] + nameParts[random.Next(0, nameParts.Length)] + " " +

alphabet[random.Next(0, alphabet.Length)] + "', " +//publication

workers[random.Next(0, workers.Length)] + ", " +//worker id

random.Next(100, 5000) + ", " +//price

random.Next(100, 5000) + ", " +//calculation

random.Next(100, 5000) + ", " +//count

formats[random.Next(0, formats.Length)] + ", " +//format id

papertypes[random.Next(0, papertypes.Length)] + ", '" +//papertype id

random.Next(2015, 2017) + "-0" + random.Next(1, 10) + "-" + random.Next(10, 29) + "', '" +//datastart

random.Next(2017, 2020) + "-0" + random.Next(1, 10) + "-" + random.Next(10, 29) + "', '" +//dataplan

random.Next(2017, 2020) + "-0" + random.Next(1, 10) + "-" + random.Next(10, 29) + "', " +//datafact

random.Next(0, 10000) + ", '" +//cost

acc[random.Next(0, acc.Length)] + "')";//account

for (int i = 2; i < int.Parse(result) + 1; i++)

{

sql += ",(" + " " +//id

customers[random.Next(0, customers.Length)] + ", " +//customer id

producttypes[random.Next(0, producttypes.Length)] + ", '" +//producttype id

greekAlphabet[random.Next(0, greekAlphabet.Length)] + nameParts[random.Next(0, nameParts.Length)] + " " +

alphabet[random.Next(0, alphabet.Length)] + "', " +//publication

workers[random.Next(0, workers.Length)] + ", " +//worker id

random.Next(100, 5000) + ", " +//price

random.Next(100, 5000) + ", " +//calculation

random.Next(100, 5000) + ", " +//count

formats[random.Next(0, formats.Length)] + ", " +//format id

papertypes[random.Next(0, papertypes.Length)] + ", '" +//papertype id

random.Next(2015, 2017) + "-0" + random.Next(1, 10) + "-" + random.Next(10, 29) + "', '" +//datastart

random.Next(2017, 2020) + "-0" + random.Next(1, 10) + "-" + random.Next(10, 29) + "', '" +//dataplan

random.Next(2017, 2020) + "-0" + random.Next(1, 10) + "-" + random.Next(10, 29) + "', " +//datafact

random.Next(0, 10000) + ", '" +//cost

acc[random.Next(0, acc.Length)] + "')";//account

}

}

//конец формирование sql запроса insert

try

{

conn.Open();

//запрос insert

sql = @"" + sql + ";";

cmd = new NpgsqlCommand(sql, conn);

dt.Load(cmd.ExecuteReader());

//конец запрос insert

conn.Close();

//select

selectSql = @"select id, customer\_id as индекс\_заказчика, (select producttype from producttypes where id=producttype\_id) as изделие," +

" publication as название\_издания,picture as титульная\_страница," +

"calculation as тираж, count as количество\_листов, (select format from formats where id=format\_id) as формат, (select papertype from papertypes where id = papertype\_id) as вид\_бумаги" +

", datastart as дата\_принятия, dataplan as дата\_выполнения\_план, datafact as дата\_выполнения\_факт, cost as предоплата, comment as доп\_сведения" +

",account as банковский\_номер from orders limit 1000";

Select();

//select

//count

sql = "select count(\*) from orders";

ChangeLabel("Таблица Заказы");

//count

MessageBox.Show("Генерация успешно выполнена");

}

catch (Exception ex)

{

conn.Close();

MessageBox.Show("Ошибка при генерации. Error: " + ex.Message);

}

}

else//если нажали отмена или строка пустая

{

MessageBox.Show("Генерация отменена!", "Оповещение", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

break;

}

case "Заказчики":

{

//"prints";

string result = Microsoft.VisualBasic.Interaction.InputBox("Сколько записей сгенерировать:", "Генерация таблицы Заказчики", "");

if (result != "")//если нажали отмена и строка не пустая

{

try

{

selectSql = @"select \* from customers";

Select();

}

catch (Exception ex)

{

conn.Close();

MessageBox.Show("Ошибка при отчистке таблицы. Error: " + ex.Message);

}

//формирование sql запроса insert

if (true)

{

sql = "INSERT INTO customers( first\_name, second\_name, third\_name, city\_id, address, birthday, phone) VALUES" +

"(" + " '" +//id

first\_names[random.Next(0, first\_names.Length)] + "', '" +//first name

second\_names[random.Next(0, second\_names.Length)] + "', '" +//second name

third\_names[random.Next(0, third\_names.Length)] + "', " +// third name

random.Next(1, 7) + ", '" +//city id

streets[random.Next(0, streets.Length)] + " " + random.Next(1, 150) + "', " +//address

"'" + random.Next(1950, 2002) + "-0" + random.Next(1, 10) + "-0" + random.Next(1, 28) + "', " +//birthday

"'071" + random.Next(0, 10) + random.Next(0, 10) + random.Next(0, 10) + random.Next(0, 10) + random.Next(0, 10) + random.Next(0, 10) + random.Next(0, 10) + "') ";//phone

for (int i = 2; i < int.Parse(result) + 1; i++)

{

sql += ",(" + " '" +//id

first\_names[random.Next(0, first\_names.Length)] + "', '" +//first name

second\_names[random.Next(0, second\_names.Length)] + "', '" +//second name

third\_names[random.Next(0, third\_names.Length)] + "', " +// third name

random.Next(1, 7) + ", '" +//city id

streets[random.Next(0, streets.Length)] + " " + random.Next(1, 150) + "', " +//address

"'" + random.Next(1950, 2002) + "-0" + random.Next(1, 10) + "-0" + random.Next(1, 28) + "', " +//birthday

"'071" + random.Next(0, 10) + random.Next(0, 10) + random.Next(0, 10) + random.Next(0, 10) + random.Next(0, 10) + random.Next(0, 10) + random.Next(0, 10) + "') ";//phone

}

}

//конец формирование sql запроса insert

try

{

conn.Open();

//запрос insert

sql = @"" + sql + ";";

cmd = new NpgsqlCommand(sql, conn);

dt.Load(cmd.ExecuteReader());

//конец запрос insert

conn.Close();

//select

selectSql = @"select id,first\_name as имя, second\_name as фамилия, third\_name as отчество,(select city from cities where id=city\_id) as город," +

"address as адрес, birthday as дата\_рождения, phone as телефон from customers";

Select();

//select

//count

sql = "select count(\*) from customers";

ChangeLabel("Таблица Заказчики");

//count

MessageBox.Show("Генерация успешно выполнена");

}

catch (Exception ex)

{

conn.Close();

MessageBox.Show("Ошибка при генерации. Error: " + ex.Message);

}

}

else//если нажали отмена или строка пустая

{

MessageBox.Show("Генерация отменена!", "Оповещение", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

break;

}

case "Работники":

{

string result = Microsoft.VisualBasic.Interaction.InputBox("Сколько записей сгенерировать:", "Генерация таблицы Работники", "");

if (result != "")//если нажали отмена и строка не пустая

{

int printsCount=0;

try

{

conn.Open();

//sql = @"select count(\*) from workers";

//cmd = new NpgsqlCommand(sql, conn);

//MessageBox.Show("Будет удалено " + cmd.ExecuteScalar() + " записей из таблицы Работники", "Предупреждение", MessageBoxButtons.OKCancel);

sql = @"select count(\*) from prints";

cmd = new NpgsqlCommand(sql, conn);

printsCount = int.Parse(cmd.ExecuteScalar().ToString());

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

conn.Close();

//conn.Open();

//sql = @"delete from workers";

//cmd = new NpgsqlCommand(sql, conn);

//dt.Load(cmd.ExecuteReader());

////MessageBox.Show("Delete succesfully");

//conn.Close();

//select

selectSql = @"select \* from workers";

Select();

//select

}

catch (Exception ex)

{

conn.Close();

MessageBox.Show("Ошибка при отчистке таблицы. Error: " + ex.Message);

}

//формирование sql запроса insert

if (true)

{

sql = "INSERT INTO public.workers( first\_name, second\_name, third\_name, print\_id) VALUES " +

"(" + " '" +//id

first\_names[random.Next(0, first\_names.Length)] + "', '" +//first name

second\_names[random.Next(0, second\_names.Length)] + "', '" +//second name

third\_names[random.Next(0, third\_names.Length)] + "', " +// third name

random.Next(32, 32 + printsCount) + ")";//print id

for (int i = 2; i < int.Parse(result) + 1; i++)

{

sql += ",(" + "'" +//id

first\_names[random.Next(0, first\_names.Length)] + "', '" +//first name

second\_names[random.Next(0, second\_names.Length)] + "', '" +//second name

third\_names[random.Next(0, third\_names.Length)] + "', " +// third name

random.Next(32, 32 + printsCount) + ")";//print id

}

}

//конец формирование sql запроса insert

try

{

conn.Open();

//запрос insert

sql = @"" + sql + ";";

cmd = new NpgsqlCommand(sql, conn);

dt.Load(cmd.ExecuteReader());

//конец запрос insert

conn.Close();

//select

selectSql = @"select id,first\_name as имя, second\_name as фамилия, third\_name as отчество,(select print from prints where id=print\_id) as типография from workers";

Select();

//select

//count

sql = "select count(\*) from workers";

ChangeLabel("Таблица Работники");

//count

MessageBox.Show("Генерация успешно выполнена");

}

catch (Exception ex)

{

conn.Close();

MessageBox.Show("Ошибка при генерации. Error: " + ex.Message);

}

}

else//если нажали отмена или строка пустая

{

MessageBox.Show("Генерация отменена!", "Оповещение", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

break;

}

default:

{

{

MessageBox.Show("Нельзя сгенерировать справочник");

}

break;

}

}

}

private void panel1\_Paint(object sender, PaintEventArgs e)

{

}

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

{

CheckComboBox(cellComboBox.SelectedIndex);

}

private void updateButton\_Click\_1(object sender, EventArgs e)

{

string[] words = label1.Text.Split(' ');

if (rowIndex >= 0)

switch (words[1])

{

case "Типографии":

{

//return "prints";

PrintsAddForm addForm = new PrintsAddForm();

addForm.isUpdate = true;

addForm.id = int.Parse(dt.Rows[rowIndex].ItemArray[0].ToString());

addForm.Visible = true;

break;

}

case "Заказы":

{

//return "orders";

OrdersAddForm ordersAdd = new OrdersAddForm();

ordersAdd.isUpdate = true;

ordersAdd.id = int.Parse(dt.Rows[rowIndex].ItemArray[0].ToString());

ordersAdd.Visible = true;

break;

}

case "Заказчики":

{

//return "customers";

CustomersAddForm customersAdd = new CustomersAddForm();

customersAdd.isUpdate = true;

customersAdd.id = int.Parse(dt.Rows[rowIndex].ItemArray[0].ToString());

customersAdd.Visible = true;

break;

}

case "Работники":

{

//return "workers";

WorkersAddForm workersAdd = new WorkersAddForm();

workersAdd.isUpdate = true;

workersAdd.id= int.Parse(dt.Rows[rowIndex].ItemArray[0].ToString());

workersAdd.Visible = true;

break;

}

case "Банковские":

{

//return "accounts";

DirectoryAddForm directoryAdd = new DirectoryAddForm(true, true, false, "Банковский счет", "Банк", "", words[1], FindTable(),0, dt.Rows[rowIndex].ItemArray[0].ToString());

directoryAdd.isUpdate = true;

directoryAdd.Visible = true;

directoryAdd.directory = FindTable();

break;

}

case "Типы":

{

//return "types";

DirectoryAddForm directoryAdd = new DirectoryAddForm(true, false, false, "Тип собственности", "", "", words[1], FindTable(), int.Parse(dt.Rows[rowIndex].ItemArray[0].ToString()), "");

directoryAdd.isUpdate = true;

directoryAdd.Visible = true;

directoryAdd.directory = FindTable();

break;

}

case "Районы":

{

//return "districts";

DirectoryAddForm directoryAdd = new DirectoryAddForm(true, false, false, "Район", "", "", words[1], FindTable(), int.Parse(dt.Rows[rowIndex].ItemArray[0].ToString()), "");

directoryAdd.isUpdate = true;

directoryAdd.Visible = true;

directoryAdd.directory = FindTable();

break;

}

case "Города":

{

//return "cities";

DirectoryAddForm directoryAdd = new DirectoryAddForm(true, false, false, "Город", "", "", words[1], FindTable(), int.Parse(dt.Rows[rowIndex].ItemArray[0].ToString()), "");

directoryAdd.isUpdate = true;

directoryAdd.Visible = true;

directoryAdd.directory = FindTable();

break;

}

case "Банки":

{

//return "banks";

DirectoryAddForm directoryAdd = new DirectoryAddForm(true, false, false, "Банк", "", "", words[1], FindTable(), int.Parse(dt.Rows[rowIndex].ItemArray[0].ToString()), "");

directoryAdd.isUpdate = true;

directoryAdd.Visible = true;

directoryAdd.directory = FindTable();

break;

}

case "Изделия":

{

//return "producttypes";

DirectoryAddForm directoryAdd = new DirectoryAddForm(true, false, false, "Изделие", "", "", words[1], FindTable(), int.Parse(dt.Rows[rowIndex].ItemArray[0].ToString()), "");

directoryAdd.isUpdate = true;

directoryAdd.Visible = true;

directoryAdd.directory = FindTable();

break;

}

case "Виды":

{

//return "papertypes";

DirectoryAddForm directoryAdd = new DirectoryAddForm(true, false, true, "Вид бумаги", "", "Плотность", words[1], FindTable(), int.Parse(dt.Rows[rowIndex].ItemArray[0].ToString()), "");

directoryAdd.isUpdate = true;

directoryAdd.Visible = true;

directoryAdd.directory = FindTable();

break;

}

case "Форматы":

{

//return "formats";

DirectoryAddForm directoryAdd = new DirectoryAddForm(true, false, false, "Формат", "", "", words[1], FindTable(), int.Parse(dt.Rows[rowIndex].ItemArray[0].ToString()),"");

directoryAdd.isUpdate = true;

directoryAdd.Visible = true;

directoryAdd.directory = FindTable();

break;

}

}

else MessageBox.Show("Выберите запись");

}

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

{

SelectForm selectForm = new SelectForm();

selectForm.Show();

}

private void изделияToolStripMenuItem2\_Click(object sender, EventArgs e)

{

DiagramForm diagramForm = new DiagramForm();

diagramForm.type = 3;

diagramForm.Show();

}

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

{

DiagramForm diagramForm = new DiagramForm();

diagramForm.type = 2;

diagramForm.Show();

}

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

{

DiagramForm diagramForm = new DiagramForm();

diagramForm.type = 1;

diagramForm.Show();

}

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

{

string path = System.IO.Directory.GetCurrentDirectory() + @"\" + "Save\_value.xlsx";

Excel.Application excelapp = new Excel.Application();

Excel.Workbook workbook = excelapp.Workbooks.Add();

Excel.Worksheet worksheet = workbook.ActiveSheet;

for (int i= 1; i < dgvData.RowCount + 1; i++)

{

for (int j = 1; j < dgvData.ColumnCount+1; j++)

{

worksheet.Rows[i].Columns[j] = dgvData.Rows[i - 1].Cells[j - 1].Value;

}

}

excelapp.AlertBeforeOverwriting = false;

workbook.SaveAs(path);

excelapp.Quit();

}

private void средийРазмерТиражейПоТипографииToolStripMenuItem\_Click(object sender, EventArgs e)//Средний тираж по типографиям, районам, городу

{

DataTable dt4 = new DataTable();

string path = System.IO.Directory.GetCurrentDirectory() + @"\" + "Средний\_тираж.xlsx";

sql = @"select p.print as типография,avg(o.calculation) as средний\_тираж\_типография,d.district as район,dist.средний\_тираж\_район,cit.средний\_тираж\_город" +

" from districts as d, prints as p, workers as w, orders as o, ( select d.district,avg(o.calculation) as средний\_тираж\_район from districts as d," +

" prints as p, workers as w, orders as o where p.id=w.print\_id and d.id=p.district\_id and o.worker\_id = w.id group by 1 order by 1) as dist," +

" (select avg(o.calculation) as средний\_тираж\_город from districts as d, prints as p, workers as w, orders as o where p.id=w.print\_id and " +

" d.id=p.district\_id and o.worker\_id = w.id order by 1) as cit where p.id=w.print\_id and d.id=p.district\_id and o.worker\_id = w.id and dist.district = d.district" +

" group by 1,3,4,5 order by 3,1";

try

{

AuthorizationForm.mainForm.conn.Open();

AuthorizationForm.mainForm.cmd = new NpgsqlCommand(sql, AuthorizationForm.mainForm.conn);

dt4 = new DataTable();

dt4.Load(AuthorizationForm.mainForm.cmd.ExecuteReader());

AuthorizationForm.mainForm.conn.Close();

}

catch (Exception ex)

{

AuthorizationForm.mainForm.conn.Close();

MessageBox.Show("Error: " + ex.Message);

}

Excel.Application excelapp = new Excel.Application();

Excel.Workbook workbook = excelapp.Workbooks.Add();

Excel.Worksheet worksheet = workbook.ActiveSheet;

for (int j = 1; j < dt4.Columns.Count + 1; j++)

{

worksheet.Rows[1].Columns[j] = dt4.Columns[j - 1].ColumnName;

}

for (int i = 2; i < dt4.Rows.Count + 2; i++)

{

for (int j = 1; j < dt4.Columns.Count + 1; j++)

{

worksheet.Rows[i].Columns[j] = dt4.Rows[i - 2].ItemArray[j - 1];

}

}

excelapp.AlertBeforeOverwriting = false;

workbook.SaveAs(path);

excelapp.Quit();

}

private void стреднийРазмерТиражейПоРайонуToolStripMenuItem\_Click(object sender, EventArgs e)//Лучшая типография по району, городу и самое популярное изделие

{

DataTable dt4 = new DataTable();

string path = System.IO.Directory.GetCurrentDirectory() + @"\" + "Лучшая\_типография.xlsx";

sql = @"select p.print as типография\_район,o.cost\*o.calculation as сумма\_заказов\_район,d.district as район,maxxx.print as типография\_город," +

"maxxx.maxcd as сумма\_заказов\_город,prmax.изделие as популярное\_изделие , prmax.количество\_заказов\_изделия from districts as d," +

" prints as p, workers as w, orders as o, ( select d.district,max(o.cost\*o.calculation) as maxcd from districts as d," +

" prints as p, workers as w, orders as o where p.id=w.print\_id and d.id=p.district\_id and o.worker\_id = w.id " +

" group by 1 order by 1) as maxd, ( select p.print,max(o.cost\*o.calculation) as maxcd from districts as d, " +

" prints as p, workers as w, orders as o,( select max(o.cost\*o.calculation) as maxcc from districts as d," +

" prints as p, workers as w, orders as o where p.id=w.print\_id and d.id=p.district\_id and" +

" o.worker\_id = w.id order by 1) as maxc where p.id=w.print\_id and d.id=p.district\_id and o.worker\_id = w.id and" +

" maxc.maxcc=(o.cost\*o.calculation) group by 1 order by 1) as maxxx, (select pr.producttype as изделие,o.producttype\_id,count(o.\*) as количество\_заказов\_изделия from districts as d, " +

" prints as p, workers as w, orders as o, producttypes as pr where p.id=w.print\_id and d.id=p.district\_id and o.worker\_id = w.id and" +

" o.producttype\_id=pr.id group by 1,2 order by 3 desc limit 1) as prmax " +

" where p.id=w.print\_id and d.id=p.district\_id and o.worker\_id = w.id and maxd.maxcd=(o.cost\*o.calculation) and maxd.district=d.district order by 1";

try

{

AuthorizationForm.mainForm.conn.Open();

AuthorizationForm.mainForm.cmd = new NpgsqlCommand(sql, AuthorizationForm.mainForm.conn);

dt4 = new DataTable();

dt4.Load(AuthorizationForm.mainForm.cmd.ExecuteReader());

AuthorizationForm.mainForm.conn.Close();

}

catch (Exception ex)

{

AuthorizationForm.mainForm.conn.Close();

MessageBox.Show("Error: " + ex.Message);

}

Excel.Application excelapp = new Excel.Application();

Excel.Workbook workbook = excelapp.Workbooks.Add();

Excel.Worksheet worksheet = workbook.ActiveSheet;

for (int j = 1; j < dt4.Columns.Count + 1; j++)

{

worksheet.Rows[1].Columns[j] = dt4.Columns[j - 1].ColumnName;

}

for (int i = 2; i < dt4.Rows.Count + 2; i++)

{

for (int j = 1; j < dt4.Columns.Count + 1; j++)

{

worksheet.Rows[i].Columns[j] = dt4.Rows[i - 2].ItemArray[j - 1];

}

}

excelapp.AlertBeforeOverwriting = false;

workbook.SaveAs(path);

excelapp.Quit();

}

private void стреднийРазмерТиражейПоГородуToolStripMenuItem\_Click(object sender, EventArgs e)//Стоимость заказов типографий за ГОД не выполненные в срок

{

DataTable dt4 = new DataTable();

string path = System.IO.Directory.GetCurrentDirectory() + @"\" + "Стоимость\_заказов.xlsx";

string result = Microsoft.VisualBasic.Interaction.InputBox("Введите год:", "Ввод данных для запроса", "");

sql = @"select id as номер\_заказа,publication заказ,datafact дата\_выполнения\_факт,dataplan as дата\_выполнения\_план from orders" +

" where datafact>dataplan and extract(year from datafact) =" + result + " order by 2";

try

{

AuthorizationForm.mainForm.conn.Open();

AuthorizationForm.mainForm.cmd = new NpgsqlCommand(sql, AuthorizationForm.mainForm.conn);

dt4 = new DataTable();

dt4.Load(AuthorizationForm.mainForm.cmd.ExecuteReader());

AuthorizationForm.mainForm.conn.Close();

}

catch (Exception ex)

{

AuthorizationForm.mainForm.conn.Close();

MessageBox.Show("Error: " + ex.Message);

}

Excel.Application excelapp = new Excel.Application();

Excel.Workbook workbook = excelapp.Workbooks.Add();

Excel.Worksheet worksheet = workbook.ActiveSheet;

for (int j = 1; j < dt4.Columns.Count + 1; j++)

{

worksheet.Rows[1].Columns[j] = dt4.Columns[j - 1].ColumnName;

}

for (int i = 2; i < dt4.Rows.Count + 2; i++)

{

for (int j = 1; j < dt4.Columns.Count + 1; j++)

{

worksheet.Rows[i].Columns[j] = dt4.Rows[i - 2].ItemArray[j - 1];

}

}

excelapp.AlertBeforeOverwriting = false;

workbook.SaveAs(path);

excelapp.Quit();

}

public void CreateHTMLFile(string table,string name)

{

string path = System.IO.Directory.GetCurrentDirectory() + @"\" +name +".html";

var sbHTML = new StringBuilder();

var sw = new StreamWriter(System.IO.Directory.GetCurrentDirectory() + @"\" +name +".html");

sbHTML.AppendLine("<DOCTYPE html>");

sbHTML.AppendLine("<html>");

sbHTML.AppendLine("<head>");

sbHTML.AppendLine("<title>");

sbHTML.AppendLine(name);

sbHTML.AppendLine("</title>");

sbHTML.AppendLine("</head>");

sbHTML.AppendLine("<body>");

sbHTML.AppendLine(table);

sbHTML.AppendLine("</body>");

sbHTML.AppendLine("</html>");

sw.Write(sbHTML);

sw.Close();

Process.Start(System.IO.Directory.GetCurrentDirectory() + @"\" + name + ".html");

}

private void типографииToolStripMenuItem1\_Click(object sender, EventArgs e)

{

var dt = new DataTable();

sql = @"select id,print as типография,(select type from types where id=type\_id) as тип\_собственности," +

"(select district from districts where id=district\_id) as район,address as адрес, phone as телефон, year as год\_основания from prints";

try

{

AuthorizationForm.mainForm.conn.Open();

AuthorizationForm.mainForm.cmd = new NpgsqlCommand(sql, AuthorizationForm.mainForm.conn);

dt = new DataTable();

dt.Load(AuthorizationForm.mainForm.cmd.ExecuteReader());

AuthorizationForm.mainForm.conn.Close();

}

catch (Exception ex)

{

AuthorizationForm.mainForm.conn.Close();

MessageBox.Show("Error: " + ex.Message);

}

CreateHTMLFile(Extensions.GetHtml(dt),"типографии");

}

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

{

var dt = new DataTable();

sql = @"select id, customer\_id as индекс\_заказчика, (select producttype from producttypes where id=producttype\_id) as изделие," +

" publication as название\_издания,picture as титульная\_страница, worker\_id as индекс\_работника, price as цена\_экземпляра, " +

"calculation as тираж, count as количество\_листов, (select format from formats where id=format\_id) as формат, (select papertype from papertypes where id = papertype\_id) as вид\_бумаги" +

", datastart as дата\_принятия, dataplan as дата\_выполнения\_план, datafact as дата\_выполнения\_факт, cost as предоплата, comment as доп\_сведения" +

",account as банковский\_номер from orders limit 1000";

try

{

AuthorizationForm.mainForm.conn.Open();

AuthorizationForm.mainForm.cmd = new NpgsqlCommand(sql, AuthorizationForm.mainForm.conn);

dt = new DataTable();

dt.Load(AuthorizationForm.mainForm.cmd.ExecuteReader());

AuthorizationForm.mainForm.conn.Close();

}

catch (Exception ex)

{

AuthorizationForm.mainForm.conn.Close();

MessageBox.Show("Error: " + ex.Message);

}

CreateHTMLFile(Extensions.GetHtml(dt), "заказы");

}

private void заказчикиToolStripMenuItem1\_Click(object sender, EventArgs e)

{

var dt = new DataTable();

sql = @"select id,first\_name as имя, second\_name as фамилия, third\_name as отчество,(select city from cities where id=city\_id) as город," +

"address as адрес, birthday as дата\_рождения, phone as телефон from customers";

try

{

AuthorizationForm.mainForm.conn.Open();

AuthorizationForm.mainForm.cmd = new NpgsqlCommand(sql, AuthorizationForm.mainForm.conn);

dt = new DataTable();

dt.Load(AuthorizationForm.mainForm.cmd.ExecuteReader());

AuthorizationForm.mainForm.conn.Close();

}

catch (Exception ex)

{

AuthorizationForm.mainForm.conn.Close();

MessageBox.Show("Error: " + ex.Message);

}

CreateHTMLFile(Extensions.GetHtml(dt), "заказчики");

}

private void работникиToolStripMenuItem1\_Click(object sender, EventArgs e)

{

var dt = new DataTable();

sql = @"select id, first\_name as имя, second\_name as фамилия, third\_name as отчество,(select print from prints where id = print\_id) as типография from workers";

try

{

AuthorizationForm.mainForm.conn.Open();

AuthorizationForm.mainForm.cmd = new NpgsqlCommand(sql, AuthorizationForm.mainForm.conn);

dt = new DataTable();

dt.Load(AuthorizationForm.mainForm.cmd.ExecuteReader());

AuthorizationForm.mainForm.conn.Close();

}

catch (Exception ex)

{

AuthorizationForm.mainForm.conn.Close();

MessageBox.Show("Error: " + ex.Message);

}

CreateHTMLFile(Extensions.GetHtml(dt), "работники");

}

private void типыСобственностиToolStripMenuItem1\_Click(object sender, EventArgs e)

{

var dt = new DataTable();

sql = @"select id,type as Тип\_Собственности from types";

try

{

AuthorizationForm.mainForm.conn.Open();

AuthorizationForm.mainForm.cmd = new NpgsqlCommand(sql, AuthorizationForm.mainForm.conn);

dt = new DataTable();

dt.Load(AuthorizationForm.mainForm.cmd.ExecuteReader());

AuthorizationForm.mainForm.conn.Close();

}

catch (Exception ex)

{

AuthorizationForm.mainForm.conn.Close();

MessageBox.Show("Error: " + ex.Message);

}

CreateHTMLFile(Extensions.GetHtml(dt), "типы\_собственности");

}

private void районыToolStripMenuItem1\_Click(object sender, EventArgs e)

{

var dt = new DataTable();

sql = @"select id,district as Район from districts";

try

{

AuthorizationForm.mainForm.conn.Open();

AuthorizationForm.mainForm.cmd = new NpgsqlCommand(sql, AuthorizationForm.mainForm.conn);

dt = new DataTable();

dt.Load(AuthorizationForm.mainForm.cmd.ExecuteReader());

AuthorizationForm.mainForm.conn.Close();

}

catch (Exception ex)

{

AuthorizationForm.mainForm.conn.Close();

MessageBox.Show("Error: " + ex.Message);

}

CreateHTMLFile(Extensions.GetHtml(dt), "районы");

}

private void изделияToolStripMenuItem1\_Click(object sender, EventArgs e)

{

var dt = new DataTable();

sql = @"select id,format as Формат from formats";

try

{

AuthorizationForm.mainForm.conn.Open();

AuthorizationForm.mainForm.cmd = new NpgsqlCommand(sql, AuthorizationForm.mainForm.conn);

dt = new DataTable();

dt.Load(AuthorizationForm.mainForm.cmd.ExecuteReader());

AuthorizationForm.mainForm.conn.Close();

}

catch (Exception ex)

{

AuthorizationForm.mainForm.conn.Close();

MessageBox.Show("Error: " + ex.Message);

}

CreateHTMLFile(Extensions.GetHtml(dt), "изделия");

}

private void городаToolStripMenuItem1\_Click(object sender, EventArgs e)

{

var dt = new DataTable();

sql = @"select id,city as Город from cities";

try

{

AuthorizationForm.mainForm.conn.Open();

AuthorizationForm.mainForm.cmd = new NpgsqlCommand(sql, AuthorizationForm.mainForm.conn);

dt = new DataTable();

dt.Load(AuthorizationForm.mainForm.cmd.ExecuteReader());

AuthorizationForm.mainForm.conn.Close();

}

catch (Exception ex)

{

AuthorizationForm.mainForm.conn.Close();

MessageBox.Show("Error: " + ex.Message);

}

CreateHTMLFile(Extensions.GetHtml(dt), "города");

}

private void банкиToolStripMenuItem1\_Click(object sender, EventArgs e)

{

var dt = new DataTable();

sql = @"select id,bank as Банк from banks";

try

{

AuthorizationForm.mainForm.conn.Open();

AuthorizationForm.mainForm.cmd = new NpgsqlCommand(sql, AuthorizationForm.mainForm.conn);

dt = new DataTable();

dt.Load(AuthorizationForm.mainForm.cmd.ExecuteReader());

AuthorizationForm.mainForm.conn.Close();

}

catch (Exception ex)

{

AuthorizationForm.mainForm.conn.Close();

MessageBox.Show("Error: " + ex.Message);

}

CreateHTMLFile(Extensions.GetHtml(dt), "банки");

}

private void банковскиеНомераToolStripMenuItem1\_Click(object sender, EventArgs e)

{

var dt = new DataTable();

sql = @"select \* from accounts\_select()";

try

{

AuthorizationForm.mainForm.conn.Open();

AuthorizationForm.mainForm.cmd = new NpgsqlCommand(sql, AuthorizationForm.mainForm.conn);

dt = new DataTable();

dt.Load(AuthorizationForm.mainForm.cmd.ExecuteReader());

AuthorizationForm.mainForm.conn.Close();

}

catch (Exception ex)

{

AuthorizationForm.mainForm.conn.Close();

MessageBox.Show("Error: " + ex.Message);

}

CreateHTMLFile(Extensions.GetHtml(dt), "банковские\_ноиера");

}

private void видыБумагиToolStripMenuItem1\_Click(object sender, EventArgs e)

{

var dt = new DataTable();

sql = @"select id,papertype as Вид\_бумаги,density from papertypes";

try

{

AuthorizationForm.mainForm.conn.Open();

AuthorizationForm.mainForm.cmd = new NpgsqlCommand(sql, AuthorizationForm.mainForm.conn);

dt = new DataTable();

dt.Load(AuthorizationForm.mainForm.cmd.ExecuteReader());

AuthorizationForm.mainForm.conn.Close();

}

catch (Exception ex)

{

AuthorizationForm.mainForm.conn.Close();

MessageBox.Show("Error: " + ex.Message);

}

CreateHTMLFile(Extensions.GetHtml(dt), "виды\_бумаги");

}

private void форматыToolStripMenuItem1\_Click(object sender, EventArgs e)

{

var dt = new DataTable();

sql = @"select id,format as Формат from formats";

try

{

AuthorizationForm.mainForm.conn.Open();

AuthorizationForm.mainForm.cmd = new NpgsqlCommand(sql, AuthorizationForm.mainForm.conn);

dt = new DataTable();

dt.Load(AuthorizationForm.mainForm.cmd.ExecuteReader());

AuthorizationForm.mainForm.conn.Close();

}

catch (Exception ex)

{

AuthorizationForm.mainForm.conn.Close();

MessageBox.Show("Error: " + ex.Message);

}

CreateHTMLFile(Extensions.GetHtml(dt), "форматы");

}

}

public static class Extensions

{

public static String GetHtml(this DataTable dataTable)

{

StringBuilder sbControlHtml = new StringBuilder();

using (StringWriter stringWriter = new StringWriter())

{

using (HtmlTextWriter htmlWriter = new HtmlTextWriter(stringWriter))

{

using (var htmlTable = new HtmlTable())

{

htmlTable.Border = 1;

// Add table header row

using (var headerRow = new HtmlTableRow())

{

foreach (DataColumn dataColumn in dataTable.Columns)

{

using (var htmlColumn = new HtmlTableCell())

{

htmlColumn.InnerText = dataColumn.ColumnName;

headerRow.Cells.Add(htmlColumn);

}

}

htmlTable.Rows.Add(headerRow);

}

// Add data rows

foreach (DataRow row in dataTable.Rows)

{

using (var htmlRow = new HtmlTableRow())

{

foreach (DataColumn column in dataTable.Columns)

{

using (var htmlColumn = new HtmlTableCell())

{

htmlColumn.InnerText = row[column].ToString();

htmlRow.Cells.Add(htmlColumn);

}

}

htmlTable.Rows.Add(htmlRow);

}

}

htmlTable.RenderControl(htmlWriter);

sbControlHtml.Append(stringWriter.ToString());

}

}

}

return sbControlHtml.ToString();

}

}

}