

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-186
Моргунов А.Г

Донецк — 2020

РЕФЕРАТ

Пояснительная записка содержит: 90 страниц, 64 рисунка, 4 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 однотоабличного и многотоабличного 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 города 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 приложения для взаимодействия с этой базой данных на языке программирования C#

Разработанная система может применяться 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 необходимы такие данные: название типографии, тип собственности, район, адрес типографии, телефон типографии, год открытия, ф.и.о заказчика (физ.или юр. лицо), адрес заказчика, дата рождения/регистрации, телефон заказчика, расчетный счет заказчика (номер р/с), банк размещения р/с, вид изделия заказчика, название издания, титульная страница (эскиз автора), ф.и.о принявшего заказ в типографии, цена экземпляра изделия, тираж, количество листов в издании, 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.

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.

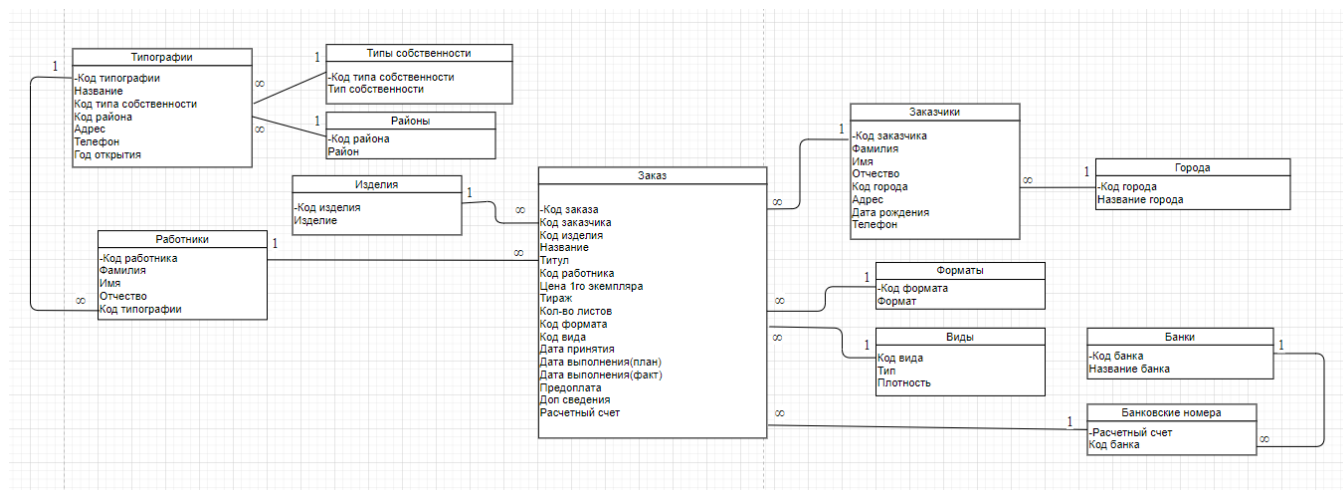


Рисунок 2.2 – логическая схема

2.3 Физическое проектирование

Физическое проектирование — создание схемы базы данных для конкретной СУБД. 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)

```
CREATE TABLE public.accounts
(
    account character(20) COLLATE pg_catalog."default" NOT NULL,
    bank_id integer NOT NULL,
    CONSTRAINT accounts_pkey PRIMARY KEY (account),
    CONSTRAINT accounts_bank_id_fkey FOREIGN KEY (bank_id)
        REFERENCES public.banks (id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
)
```

Рисунок 2.3 Скрипт SQL описывающий таблицу банковские счета

```
CREATE TABLE public.banks
(
    id bigint NOT NULL DEFAULT nextval('banks_id_seq'::regclass),
    bank character varying(30) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT banks_pkey PRIMARY KEY (id)
)
```

Рисунок 2.4 Скрипт SQL описывающий таблицу банки

```
CREATE TABLE public.cities
(
    id bigint NOT NULL DEFAULT nextval('cities_id_seq'::regclass),
    city character varying(30) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT cities_pkey PRIMARY KEY (id)
)
```

Рисунок 2.5 Скрипт SQL описывающий таблицу города

```
CREATE TABLE public.customers
(
    id bigint NOT NULL DEFAULT nextval('customers_id_seq'::regclass),
    first_name character varying(15) COLLATE pg_catalog."default" NOT NULL,
    second_name character varying(15) COLLATE pg_catalog."default" NOT NULL,
    third_name character varying(15) COLLATE pg_catalog."default" NOT NULL,
    city_id integer NOT NULL,
    address character varying(40) COLLATE pg_catalog."default" NOT NULL,
    birthday birthday_check NOT NULL,
    phone character(10) COLLATE pg_catalog."default",
    CONSTRAINT customers_pkey PRIMARY KEY (id),
    CONSTRAINT city_cascade_del FOREIGN KEY (city_id)
        REFERENCES public.cities (id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT customers_birthday_check CHECK (birthday::date < '2002-01-01'::date) NOT VALID,
    CONSTRAINT customer_birthday_check2 CHECK (birthday::date > '1920-01-01'::date) NOT VALID
)
```

Рисунок 2.6 Скрипт SQL описывающий таблицу заказчики

```
CREATE TABLE public.districts
(
    id bigint NOT NULL DEFAULT nextval('districts_id_seq'::regclass),
    district character varying(15) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT districts_pkey PRIMARY KEY (id)
)
```

Рисунок 2.7 Скрипт SQL описывающий таблицу районы

```
CREATE TABLE public.formats
(
    id bigint NOT NULL DEFAULT nextval('formats_id_seq'::regclass),
    format character varying(20) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT formats_pkey PRIMARY KEY (id)
)
```

Рисунок 2.8 Скрипт SQL описывающий таблицу форматы

```
CREATE TABLE public.prints
(
    id bigint NOT NULL DEFAULT nextval('prints_id_seq'::regclass),
    print character varying(20) COLLATE pg_catalog."default" NOT NULL,
    type_id integer NOT NULL,
    district_id integer NOT NULL,
    address character varying(40) COLLATE pg_catalog."default" NOT NULL,
    phone character(10) COLLATE pg_catalog."default" NOT NULL,
    year integer NOT NULL,
    CONSTRAINT prints_pkey PRIMARY KEY (id),
    CONSTRAINT prints_district_id_fkey FOREIGN KEY (district_id)
        REFERENCES public.districts (id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT prints_type_id_fkey FOREIGN KEY (type_id)
        REFERENCES public.types (id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT year_check2021 CHECK (year < 2021),
    CONSTRAINT prints_year_check CHECK (year > 1980)
)
```

Рисунок 2.9 Скрипт SQL описывающий таблицу типографии

```

CREATE TABLE public.orders
(
    id bigint NOT NULL DEFAULT nextval('orders_id_seq'::regclass),
    customer_id integer NOT NULL,
    producttype_id integer NOT NULL,
    publication character varying(20) COLLATE pg_catalog."default",
    picture bytea,
    worker_id integer NOT NULL,
    price integer NOT NULL,
    calculation integer NOT NULL,
    count integer,
    format_id integer NOT NULL,
    papertype_id integer NOT NULL,
    datastart date NOT NULL,
    dataplan date NOT NULL,
    datafact date NOT NULL,
    cost integer NOT NULL,
    comment text COLLATE pg_catalog."default",
    account character(20) COLLATE pg_catalog."default",
    CONSTRAINT orders_pkey PRIMARY KEY (id),
    CONSTRAINT orders_account_fkey FOREIGN KEY (account)
        REFERENCES public.accounts (account) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT orders_customer_id_fkey FOREIGN KEY (customer_id)
        REFERENCES public.customers (id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT orders_format_id_fkey FOREIGN KEY (format_id)
        REFERENCES public.formats (id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT orders_papertype_id_fkey FOREIGN KEY (papertype_id)
        REFERENCES public.papertypes (id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT orders_producttype_id_fkey FOREIGN KEY (producttype_id)
        REFERENCES public.producttypes (id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT orders_worker_id_fkey FOREIGN KEY (worker_id)
        REFERENCES public.workers (id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,

```

Рисунок 2.10 Скрипт SQL описывающий таблицу заказы

```

CONSTRAINT orders_calculation_check CHECK (calculation >= 50),
CONSTRAINT orders_check CHECK (dataplan > datastart),
CONSTRAINT orders_check1 CHECK (datafact > datastart AND datafact < CURRENT_DATE),
CONSTRAINT orders_cost_check CHECK (cost >= 0),
CONSTRAINT orders_count_check CHECK (count > 0),
CONSTRAINT orders_price_check CHECK (price >= 50),
CONSTRAINT datastart_check CHECK (datastart > '1980-01-01'::date AND datastart < '2021-01-01'::date),
CONSTRAINT orders_cost_check2 CHECK (cost <= (calculation * price)) NOT VALID
)

```

Рисунок 2.10, лист 2

```

CREATE TABLE public.papertypes
(
    id bigint NOT NULL DEFAULT nextval('papertypes_id_seq'::regclass),
    papertype character varying(30) COLLATE pg_catalog."default" NOT NULL,
    density integer NOT NULL,
    CONSTRAINT papertypes_pkey PRIMARY KEY (id),
    CONSTRAINT papertypes_density_check CHECK (density > 39 AND density < 351)
)

```

Рисунок 2.11 Скрипт SQL описывающий таблицу виды бумаги

```

CREATE TABLE public.types
(
    id bigint NOT NULL DEFAULT nextval('types_id_seq'::regclass),
    type character varying(15) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT types_pkey PRIMARY KEY (id)
)

```

Рисунок 2.12 Скрипт SQL описывающий таблицу типы собственности

```

CREATE TABLE public.workers
(
    id bigint NOT NULL DEFAULT nextval('workers_id_seq'::regclass),
    first_name character varying(15) COLLATE pg_catalog."default" NOT NULL,
    second_name character varying(15) COLLATE pg_catalog."default" NOT NULL,
    third_name character varying(15) COLLATE pg_catalog."default" NOT NULL,
    print_id integer NOT NULL,
    CONSTRAINT workers_pkey PRIMARY KEY (id),
    CONSTRAINT workers_print_id_fkey FOREIGN KEY (print_id)
        REFERENCES public.prints (id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
)

```

Рисунок 2.13 Скрипт SQL описывающий таблицу работники

3. 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 работы 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 приступить 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)

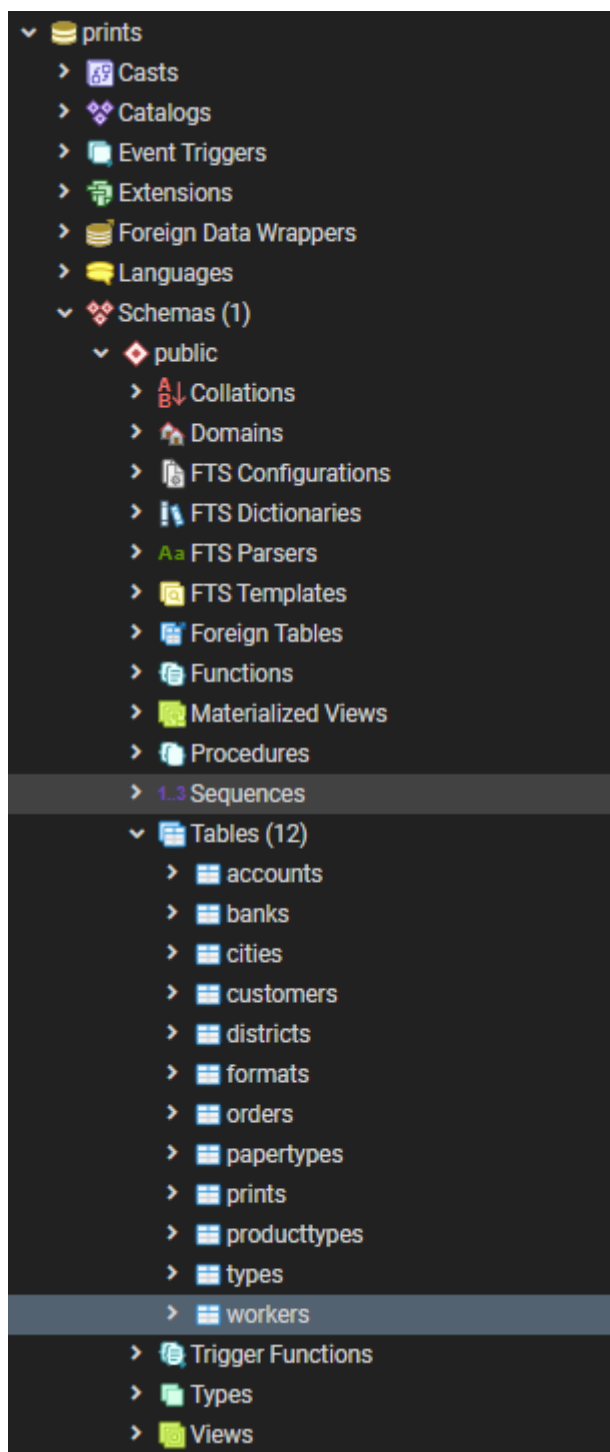


Рисунок 3.1 – База данных после создания таблиц

3.4 Схема БД

Схема БД создается чтобы сохранить целостность данных(Рис 3.2)

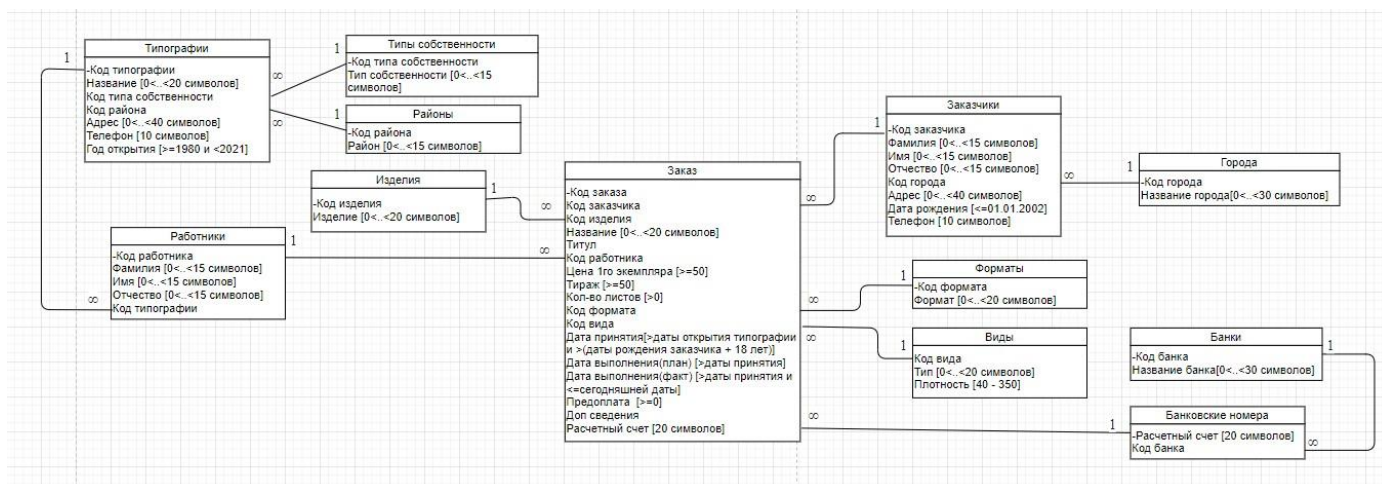


Рисунок 3.2 – схема БД

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)

Курсовая работа БД Моргунов А. ПИ-186 4семестр

Таблицы Справочники Запросы Генерация HTML публикация Инфо Диаграммы Отчет

Таблица Типографии (записей: 170)

id	типография	тип_собственности	район	адрес	телефон	год_основания
33	Ngustowowed	Индивидуальная	Ворошиловский	просп. Ворона 71	0716230542	1997
34	Oloveravant	Коллективная	Пролетарский	ул. Артема 59	0713526596	1986
35	Qvigogusto	Индивидуальная	Киевский	ул. Боевая 113	0712342998	1985
37	Zindexnurse	Смешанная	Куйбышевский	ул. Боевая 31	0711118850	1992
38	Dheartindex	Коллективная	Пролетарский	ул. Артема 99	0716488888	1998
39	Ralickwowed	Смешанная	Калининский	ул. Гоголя 146	0719051471	1989
41	Womicpeppy	Индивидуальная	Кировский	ул. Боевая 94	0717681438	1994
43	Ylightavant	Индивидуальная	Кировский	ул. Гоголя 15	0714679480	1984
45	Xawardgusto	Коллективная	Петровский	ул. Боевая 90	0711295482	1988
46	Egainsspicy	Индивидуальная	Киевский	ул. Крылова 51	0715676900	1983
47	Jeasedlight	Индивидуальная	Пролетарский	ул. Проса 27	0713046025	1984
48	Xlighteased	Коллективная	Киевский	ул. Гастелло 72	0715935067	1986
49	Unobleorder	Индивидуальная	Кировский	ул. Победы 4	0715409288	1988
50	Zloverso-oo	Коллективная	Пролетарский	ул. Кирова 105	0716028649	1986
51	Hnoblegains	Смешанная	Куйбышевский	просп. Мира 138	0711988873	1982
52	Qprimeyummy	Коллективная	Калининский	ул. Верти 4	0711810808	1994
53	Qeasesthere	Индивидуальная	Пролетарский	ул. Верти 104	0710130905	1999
54	Jcomfygusto	Смешанная	Кировский	ул. Кирова 144	0718582366	1992
55	Fcleangusto	Индивидуальная	Ворошиловский	ул. Победы 23	0716569552	1987
56	Omanhaward	Смешанная	Пролетарский	ул. Кирова 86	0711246613	1989

Удалить

☒ Текущая запись

☐ Номер записи

☐ Запись со значением Поля

Добавить Поиск Изменить

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)

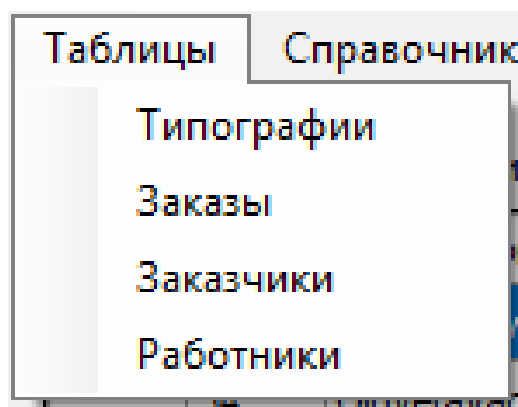


Рисунок 3.5 – Выпадающий список таблиц

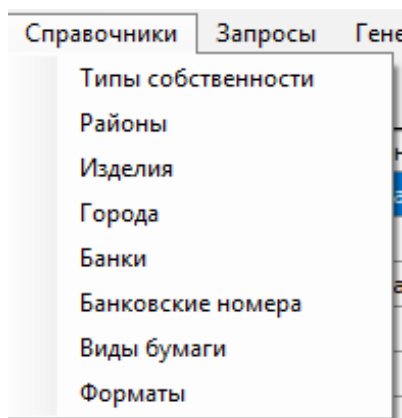


Рисунок 3.6 – Выпадающий список справочников

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

Добавление элемента

Таблица Заказы (записей: 1500)

Заказчик

Изделие

Название

Титульная страница

Выбрать

Отчистить

Работник

Цена экземпляра

Доп сведения

Тираж

Количество листов в изделии

Формат листов

Вид бумаги

Дата принятия заказа

Дата выполнения заказа (план)

Дата выполнения заказа (факт)

Предоплата

Банковский счет

Добавить

Назад

Рисунок 3.8 – Добавление в таблицу заказы

Добавление элемента

Таблица Заказчики (записей: 1500)

Имя

Фамилия

Отчество

Город

Адрес

Дата рождения

Телефон

Добавить

Назад

Рисунок 3.9 – Добавление в таблицу заказчики

Добавление элемента

Таблица Работники (записей: 1000)

Имя

Фамилия

Отчество

Тпография

Рисунок 3.10 – Добавление в таблицу работники

Добавление элемента

Таблица Типы (записей: 3)

Тип собственности

Рисунок 3.11 – Добавление в таблицу типы собственности

Добавление элемента

Таблица Виды (записей: 14)

Вид бумаги

Плотность

Рисунок 3.12 – Добавление в таблицу виды собственности

The screenshot shows a dialog box titled 'Добавление элемента' (Add Element) with a purple header bar. Below the header, it says 'Таблица Банковские (записей: 1502)'. There are two input fields: 'Банковский счет' (Bank account) and 'Банк' (Bank). To the right of these fields are two buttons: 'Добавить' (Add) and 'Заккрыть' (Close). The 'Добавить' button is highlighted with a blue border.

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.

The screenshot shows a dialog box titled 'Изменение элемента' (Change Element) with a purple header bar. Below the header, it says 'Таблица Заказчики (записей: 1500)'. There are several input fields: 'Имя' (Name) with the value 'Василий', 'Фамилия' (Surname) with 'Бобров', 'Отчество' (Patronymic) with 'Вадимович', 'Город' (City) with a dropdown menu showing 'З. Харцизск', 'Адрес' (Address) with 'ул. Боевая 105', 'Дата рождения' (Date of birth) with '1989-04-07' and a calendar icon, and 'Телефон' (Phone) with '0716672767'. At the bottom, there are two buttons: 'Изменить' (Change) and 'Назад' (Back).

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)

ул. проса	д.ч.	д.ч.
ул. Проса 139	07.04.1965	0'
пер. Тихона 121	02.05.1966	0'
ул. Верти 60	08.07.1960	0'
просп. Ворона 44	07.03.1973	0'

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)

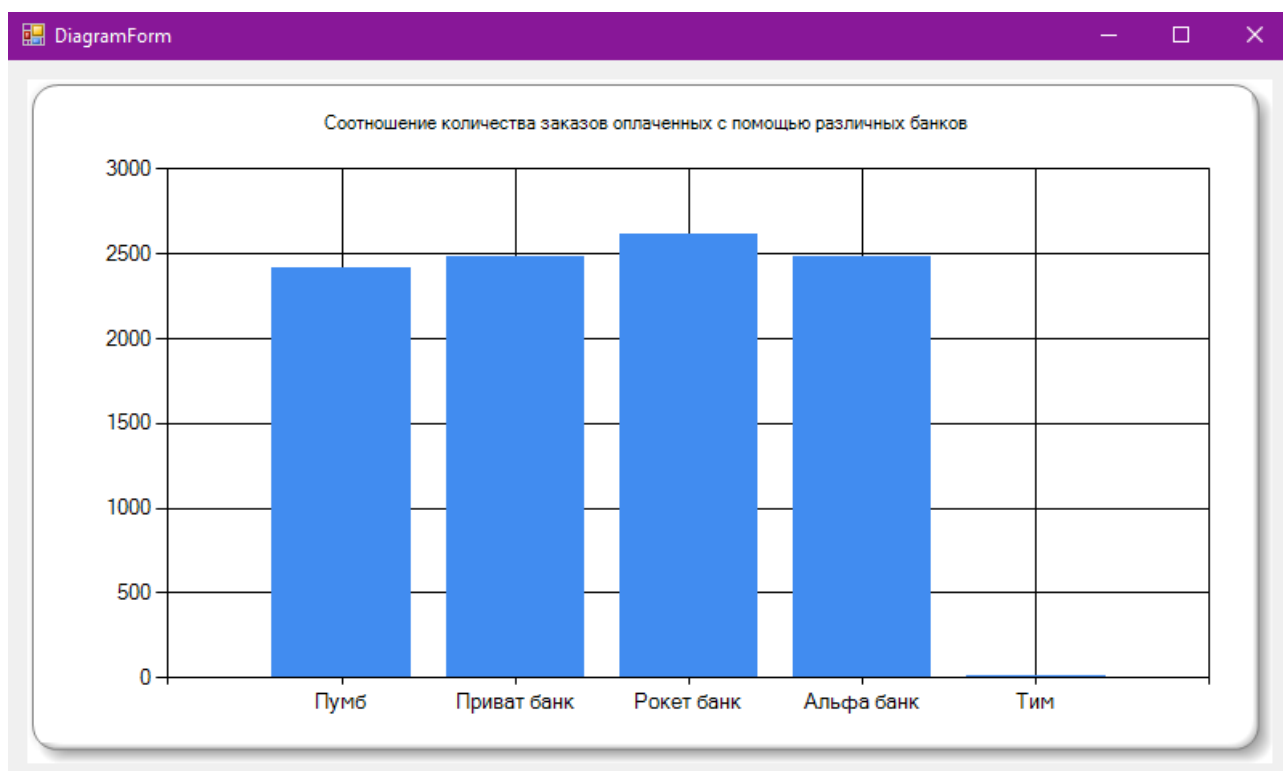


Рисунок 3.17 – Диаграмма количество заказов на банк

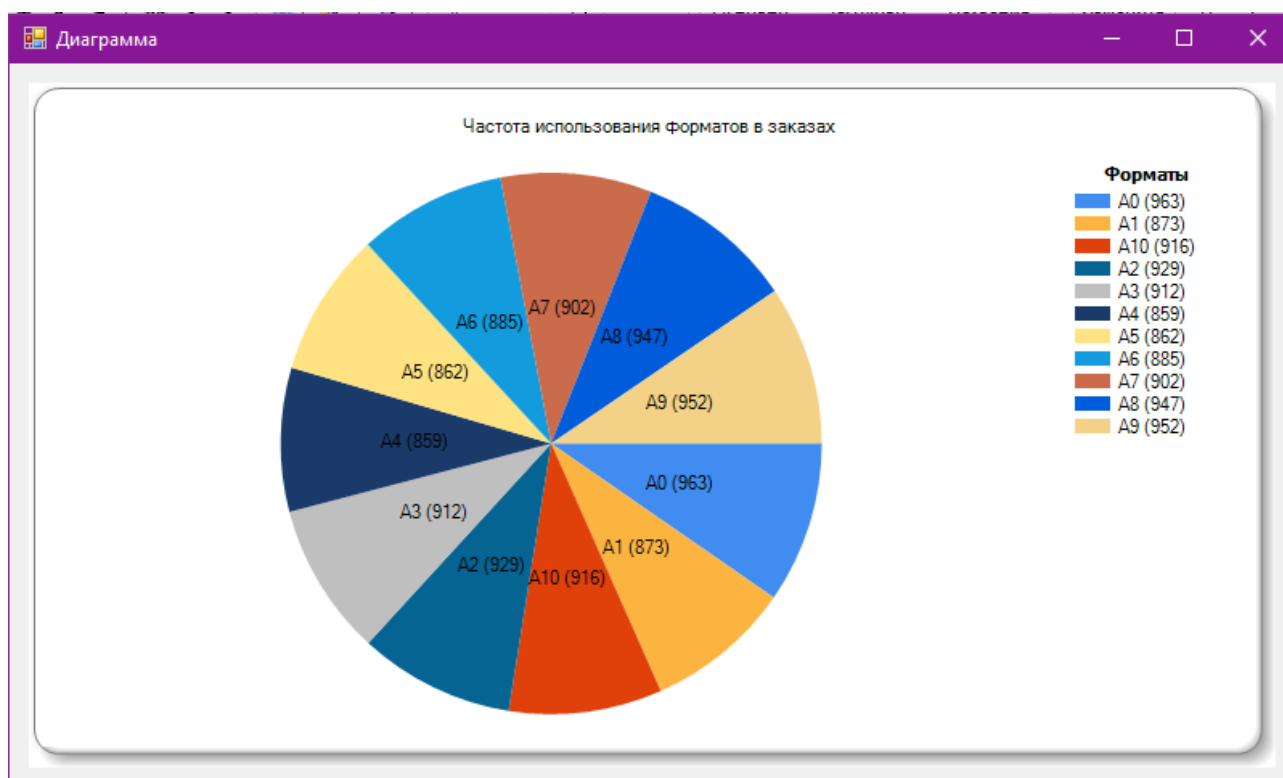


Рисунок 3.18 – Диаграмма частота использования форматов

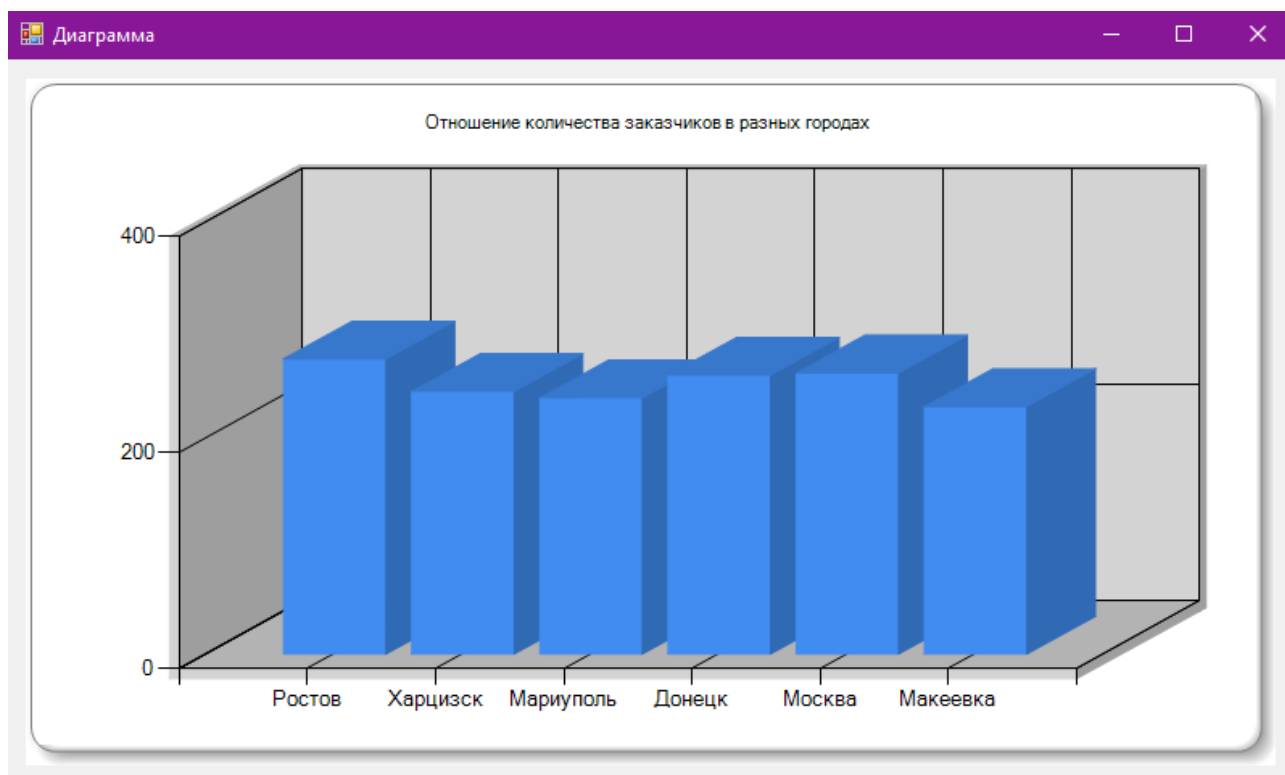


Рисунок 3.19 – Диаграмма количество заказчиков в городах

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

id	типография	тип_собственности	район	адрес	телефон	год_основания
33	Ngustowowed	Индивидуальная	Ворошиловский	просп. Ворона 71	0716230542	1997
34	Oloveravant	Коллективная	Пролетарский	ул. Артема 59	0713526596	1986
35	Qvigorgusto	Индивидуальная	Киевский	ул. Боевая 113	0712342998	1985
37	Zindexnurse	Смешанная	Куйбышевский	ул. Боевая 31	0711118850	1992
38	Dheartindex	Коллективная	Пролетарский	ул. Артема 99	0716488888	1998
39	Rslickwowed	Смешанная	Калининский	ул. Гоголя 146	0719051471	1989
41	Wcomicpeppy	Индивидуальная	Кировский	ул. Боевая 94	0717681438	1994
43	Ylightavant	Индивидуальная	Кировский	ул. Гоголя 15	0714679480	1984
45	Xawardgusto	Коллективная	Петровский	ул. Боевая 90	0711295482	1988
46	Egainsspicy	Индивидуальная	Киевский	ул. Крылова 51	0715676900	1983
47	Jeasedlight	Индивидуальная	Пролетарский	ул. Проса 27	0713046025	1984
48	Xlighteased	Коллективная	Киевский	ул. Гастелло 72	0715935067	1986
49	Unobleorder	Индивидуальная	Кировский	ул. Победы 4	0715409288	1988
50	Zloverso-so	Коллективная	Пролетарский	ул. Кирова 105	0716028649	1986
51	Hnoblegains	Смешанная	Куйбышевский	просп. Мира 138	0711988873	1982
52	Qprimeyummy	Коллективная	Калининский	ул. Верти 4	0711810808	1994

Рисунок 3.20 – html – публикация таблицы типографии

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

	1	2	3	4
1	номер_заказа	заказ	дата_выполнения_факт	дата_выполнения_план
2	9022	Alphaadon	23.08.2018	18.03.2017
3	6914	Alphaavar	24.05.2018	15.01.2017
4	3212	Alphaavar	15.09.2018	26.08.2017
5	331	Alphaavar	14.03.2018	10.03.2018
6	7062	Alphaawa	15.06.2018	24.04.2018
7	5480	Alphacivil	16.04.2018	24.01.2018
8	3459	Alphacivil	18.05.2018	28.08.2017
9	6008	Alphacivil	20.01.2018	17.01.2018
10	3045	Alphaclea	12.01.2018	21.01.2017
11	2532	Alphaclea	20.06.2018	14.04.2017
12	4750	Alphacom	19.05.2018	25.08.2017
13	5524	Alphacom	14.05.2018	14.01.2017
14	5019	Alphacom	20.03.2018	19.06.2017
15	9484	Alphacrisp	26.05.2018	26.06.2017
16	7063	Alphacrisp	28.07.2018	21.01.2017
17	1539	Alphadefi	16.02.2018	26.01.2018
18	4628	Alphadefi	13.09.2018	19.02.2018
19	450	Alphadefi	28.09.2018	14.01.2018

Рисунок 3.21 – Однотабличный отчет

	1	2	3	4	5	6	7
1	типография_район	сумма_заказов_район	район	типография_город	сумма_заказов_город	популярное_изделие	количество_заказов_изделия
2	Acleanpeppy	47354454	Ворошиловский	Qeasesthere	49845075	Журнал	1716
3	Cprimeaward	49056615	Буденовский	Qeasesthere	49845075	Журнал	1716
4	Ecleanso-so	47841867	Куйбышевский	Qeasesthere	49845075	Журнал	1716
5	Eloverbable	48451656	Петровский	Qeasesthere	49845075	Журнал	1716
6	Morderquick	49105684	Кировский	Qeasesthere	49845075	Журнал	1716
7	Qeasesthere	49845075	Пролетарский	Qeasesthere	49845075	Журнал	1716
8	Rslickwowed	48923094	Калининский	Qeasesthere	49845075	Журнал	1716
9	Uprimecomfy	49685441	Киевский	Qeasesthere	49845075	Журнал	1716

Рисунок 3.22 – Многотабличный отчет

3.6 Процедуры обработки данных

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

```

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);
    }
}

```

Рисунок 3.23 – Метод Select()

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

```

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");
}

```

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)

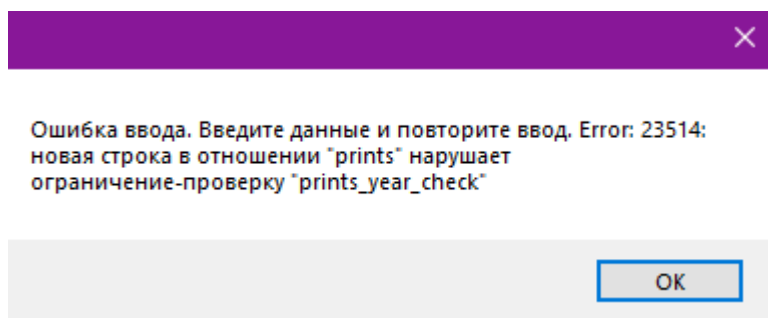


Рисунок 3.25 – неправильный ввод

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

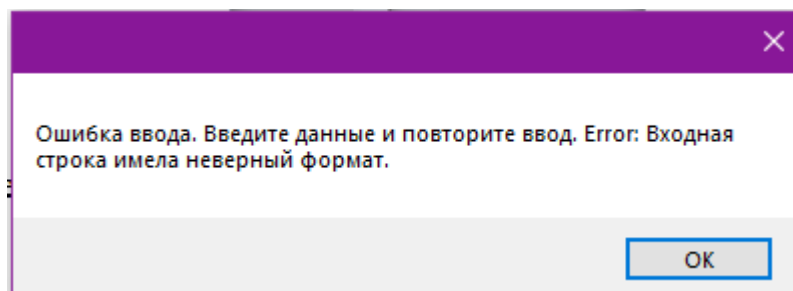


Рисунок 3.26 – неполный ввод

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

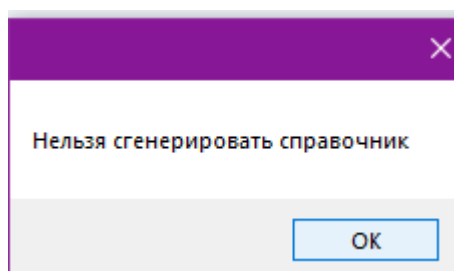


Рисунок 3.27 – ошибка при генерации справочника

3.8 Разработка эксплуатационной документации БД и ИС

В приложении В 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
```

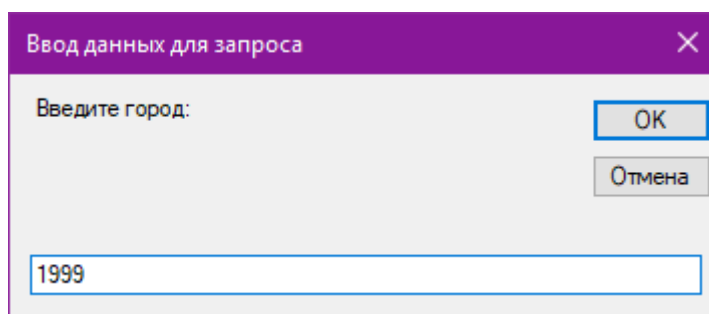


Рисунок Б.1 – Ввод данных для запроса 1

Запрос с параметром 1999

	имя	фамилия	отчество	типография
►	Илья	Макаров	Викторович	Cspicylover
	Жерар	Иванов	Викторович	Hcomicroomy
	Николай	Петухов	Назарович	Hcomicroomy
	Василий	Макаров	Густавович	Hcomicroomy
	Густав	Панов	Игнатович	Hcomicroomy
	Назар	Карпов	Карлович	Hgustoslick
	Карл	Ершов	Геннадиевич	Hgustoslick
	Марат	Федоров	Игнатович	Mprotogains
	Альберт	Рябов	Михаилович	Qeasesthere
	Егор	Карпов	Захарович	Vexactoh my
	Егор	Смирнов	Густавович	Vexactoh my
	Марат	Карпов	Игнатович	Vexactoh my
*				

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

	номер_заказа	название
▶	20	Omicronoh my Y
	108	Deltathere P
	165	Zetapeace I
	234	Lambdapeace D
	316	Alphagusto I
	340	Upsilonready H
	346	Nuyummy F
	395	Thetasharp S
	414	Sigmaclean X
	417	Rhoeases R
	492	Xiyummy W
	511	Etacomic V
	525	Omegaloved B
	581	Tauso-so M
	601	Xiadore K
	602	Upsilonready V
	633	Epsiloncivil T

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

```

Ввод данных для запроса

Введите город:

Донецк

OK

Отмена

Рисунок Б.5 – Ввод данных для запроса 3

Запрос с параметром «Донецк»

	имя	фамилия	отчество	город
►	Жерар	Панов	Борисович	Донецк
	Захар	Иванов	Николаевич	Донецк
	Альберт	Рябов	Заурович	Донецк
	Захар	Макаров	Захарович	Донецк
	Назар	Зайцев	Густавович	Донецк
	Виктор	Рябов	Михаилович	Донецк
	Николай	Зайцев	Назарович	Донецк
	Илья	Макаров	Егорович	Донецк
	Артур	Петров	Альбертович	Донецк
	Иван	Осипов	Васильевич	Донецк
	Гавриил	Макаров	Альбертович	Донецк
	Геннадий	Петров	Игоревич	Донецк
	Гавриил	Калинин	Жерарович	Донецк
	Густав	Бобров	Заурович	Донецк
	Богдан	Иванов	Артурович	Донецк
	Геннадий	Осипов	Абрамович	Донецк
	Иван	Осипов	Иванович	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»

	типография	количество_работ
►	Cawardcivil	1
	Cyummycivil	1
	Egainsspicy	1
	Fprotogains	1
	Gcomicmanly	1
	Geasessharp	1
	Hprotonurse	1
	lindexdefix	1
	Jeasesexact	1
	Nawardcomfy	1
	Ngustowowed	1
	Oloveravant	1
	Omanlyaward	2
	Qcomfyasset	2
	Qlovedready	1
	Qprimeyummy	1
	Rleasantto	1

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
```

	имя	фамилия	отчество	типография	район
►	Абрам	Бобров	Богданович	Xcleanwowed	Куйбышевский
	Абрам	Бобров	Вадимович	Oloveravant	Пролетарский
	Абрам	Зайцев	Богданович	Yslickclean	Кировский
	Абрам	Зайцев	Борисович	Lquickso-so	Буденовский
	Абрам	Зайцев	Вадимович	Ceasedquick	Калининский
	Абрам	Иванов	Абрамович	Geasessharp	Кировский
	Абрам	Иванов	Николаевич	Ceasedquick	Калининский
	Абрам	Калинин	Вадимович	Fgustomanly	Киевский
	Абрам	Калинин	Густавович	Rnurseexact	Буденовский
	Абрам	Калинин	Игнатович	Fprotogains	Калининский
	Абрам	Карпов	Артурович	Pjazzynurse	Калининский
	Абрам	Карпов	Богданович	lindexdefix	Ворошиловский
	Абрам	Карпов	Вадимович	Jawardasset	Ворошиловский
	Абрам	Макаров	Игоревич	Sdefixindex	Киевский
	Абрам	Морозов	Абрамович	Wlightpeppy	Калининский
	Абрам	Морозов	Абрамович	Jexactexact	Пролетарский
	Абрам	Морозов	Альбертович	Zindexexact	Кировский

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

```

	id	имя	фамилия	отчество	город	количество_заказ
►	315	Игорь	Петров	Михайлович	Макеевка	18
	1438	Густав	Осипов	Назарович	Макеевка	17
	1168	Гавриил	Шаров	Альбертович	Мариуполь	17
	997	Геннадий	Шаров	Захарович	Мариуполь	17
	1119	Михаил	Суворов	Густавович	Ростов	16
	385	Василий	Морозов	Захарович	Мариуполь	15
	1376	Артур	Ершов	Абрамович	Харцизск	15
	127	Вадим	Рябов	Гаврилович	Ростов	15
	236	Густав	Макаров	Михайлович	Москва	15
	1288	Михаил	Фролов	Игоревич	Мариуполь	14
	1426	Альберт	Макаров	Богданович	Москва	14
	746	Вадим	Рябов	Маратович	Москва	14
	110	Николай	Петухов	Густавович	Харцизск	14
	918	Егор	Федоров	Карлович	Донецк	13
	575	Игорь	Смирнов	Густавович	Москва	13
	847	Назар	Панов	Игоревич	Харцизск	13
	1081	Густав	Ершов	Егорович	Ростов	13

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

```

	банковский_счет	банк	количество_заказ
►	6067170401313...	Рокет банк	19
	7501469119589...	Пумб	18
	3177883652220...	Альфа банк	16
	4153225895864...	Рокет банк	16
	6980356510233...	Пумб	16
	5366513470948...	Альфа банк	15
	4614965063195...	Приват банк	14
	8232447509820...	Приват банк	14
	4916933489822...	Рокет банк	14
	1653972955181...	Приват банк	14
	6970518621734...	Альфа банк	14
	6436661849934...	Пумб	14
	7800263028265...	Пумб	13
	8606379410426...	Альфа банк	13
	1667571363498...	Рокет банк	13
	4026290581703...	Приват банк	13
	6766507860740...	Альфа банк	13

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
```


	банковский_счет	номер_заказа
▶	6181421612382...	
	7858163834632...	10010
	8978135141644...	10009
	7800263028265...	10008
	7085085322233...	10007
	2324229971203...	10006
	7468249077833...	10005
	9912581351970...	10004
	5469983756661...	10003
	5358261519981...	10002
	2128143730305...	10001
	5996411561682...	10000
	7078749918702...	9999
	3856579271611...	9998
	3985083776192...	9997
	6653467529539...	9996
	8757267986316...	9995

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
```

	id	Город	имя	фамилия	отчество
►	7	Вашингтон			
	6	Мариуполь	Густав	Петухов	Игоревич
	6	Мариуполь	Заур	Федоров	Михаилович
	6	Мариуполь	Илья	Тарасов	Игоревич
	6	Мариуполь	Богдан	Макаров	Маратович
	6	Мариуполь	Назар	Фролов	Васильевич
	6	Мариуполь	Артур	Зайцев	Егорович
	6	Мариуполь	Богдан	Рябов	Жерарович
	6	Мариуполь	Захар	Бобров	Васильевич
	6	Мариуполь	Жерар	Зайцев	Заурович
	6	Мариуполь	Абрам	Смирнов	Геннадиевич
	6	Мариуполь	Василий	Смирнов	Михаилович
	6	Мариуполь	Виктор	Калинин	Альбертович
	6	Мариуполь	Артур	Фролов	Иванович
	6	Мариуполь	Борис	Орлов	Вадимович
	6	Мариуполь	Гавриил	Калинин	Маратович
	6	Мариуполь	Виктор	Смирнов	Викторович

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
```

	account	bank
►	1234255552341...	Рокет банк
	1234123417341...	Приват банк
	1234123412341...	Тим
	1912346562500...	Рокет банк
	5460123464801...	Приват банк
	8390192885864...	Рокет банк
*		

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

```

	id	имя	фамилия	отчество	денег_потрачено
►	986	Артур	Фролов	Густавович	84348
	204	Иван	Зайцев	Артурович	2152500
	588	Гавриил	Смирнов	Маратович	2354345
	1059	Василий	Рябов	Карлович	2684736
	951	Гавриил	Федоров	Карлович	2737202
	634	Абрам	Рябов	Назарович	3545705
	1141	Заур	Калинин	Борисович	3585969
	389	Артур	Суворов	Михаилович	3891052
	1314	Борис	Калинин	Егорович	4242640
	162	Василий	Тарасов	Михаилович	4957027
	430	Заур	Суворов	Захарович	5015008
	384	Артур	Петухов	Вадимович	5060790
	207	Михаил	Рябов	Гавриилович	6306124
	969	Иван	Петров	Маратович	6535324
	1211	Борис	Зайцев	Михаилович	6885721
	1145	Игорь	Орлов	Николаевич	6907508
	1414	Геннадий	Тарасов	Женарович	7328442

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

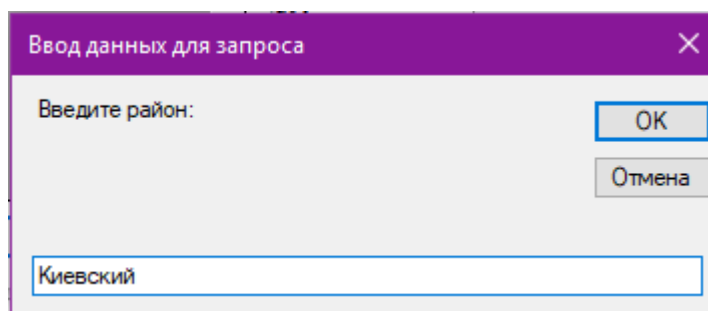


Рисунок Б.16 – Ввод данных для запроса 12

результат запроса с параметром: Киевский

	район	работников
▶	Киевский	137
*		

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

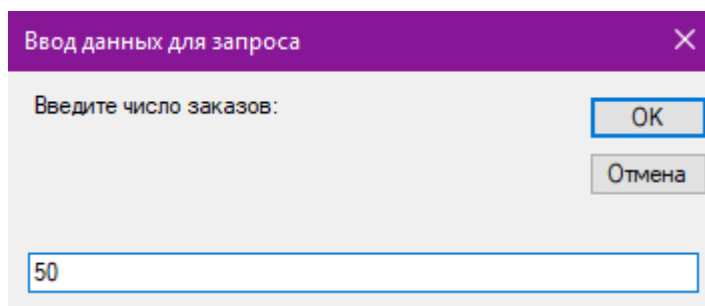


Рисунок Б.18 – Ввод данных для запроса 13

Результат запроса с параметром 50

	номер_типографи	типография	количество_зака:
►	97	Ysharkindex	51
	139	Mprotogains	53
	176	Cawardcivil	53
	103	Ceasedquick	54
	84	Moh myyummy	54
	179	Rmanlycomic	55
	175	Qcleanproto	55
	115	Lexacteases	56
	83	Jexactdefix	56
	60	Wavantgreet	56
	96	Rjazyheart	56
	55	Fcleangusto	58
	172	Ycleanindex	58
	101	Xcomfyases	59
	66	Ncomfycomic	59
	89	Jlovemanly	59
	162	Benicunomu	60

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

Ввод данных для запроса

Введите банк:

Пумб

ОК

Отмена

Рисунок Б.20 – Ввод данных для запроса 14

Ввод данных для запроса

Введите количество денег:

100000000

ОК

Отмена

Рисунок Б.21 – Ввод данных для запроса 14

	банковский_счет	денег_потрачено
►	9516377743726...	101225124
	6880098974923...	101381123
	1521953819950...	101389732
	8082929674181...	101830417
	8327811340636...	103205484
	5045328127556...	103912799
	3350285849499...	103913928
	3463323774400...	104269135
	7531320265606...	104414857
	5297225317761...	104498539
	7737593496346...	105253156
	3080268871155...	105346227
	4781557000749...	105636201
	5377231844172...	106130566
	4092784916294...	106412174
	7100516001292...	106799063
	2182483437620	106839617

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

```

	номер_типографии	типография	количество_работ	количество_заказ
▶	34	Oloveravant	14	139
	94	Ecleanso-so	14	133
	153	Morderquick	12	121
	91	Rroomyprime	10	113
	99	Vexactoh my	11	110
	85	Kexactso-so	12	110
	173	Hcomicroomy	9	108
	72	Clightroomy	9	108
	51	Hnoble gains	10	108
	123	Cslickaward	10	108
	102	Olovedavant	10	107
	105	Teasesgusto	11	105
	104	lindexdefix	10	104
	42	Qspicypeace	10	103
	76	Uwowedsaucy	10	102
	119	Znoblewowed	9	101
	54	Iconfugusto	9	100

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

	номер_типографи	типография	количество_работ	возраст	год_основания
►	134	Rnurseexact	3	39	1981
	133	Hprotonurse	4	39	1981
	84	Moh myyummy	5	39	1981
	128	Vprimeasset	5	39	1981
	118	Rsharkindex	7	39	1981
	144	Jnobleleased	7	39	1981
	122	Fgustomanly	7	39	1981
	87	Ulightlover	5	38	1982
	161	Osharklight	7	38	1982
	177	Tcomicroomy	7	38	1982
	163	Gcomicmanly	8	38	1982
	116	Ysharkvigor	8	38	1982
	160	Qlovedready	9	38	1982
	51	Hnoble gains	10	38	1982
	62	Bexactexact	3	37	1983
	83	Jexactdefix	5	37	1983
	129	Oleasedeh my	7	37	1983

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 удалите 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_KurovayaBD_
{
    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);//присоедине
    ние 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"].V
alue.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_Cli
ck(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_Clic
    k(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(objec
    t 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,"Банковск
ий 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.T
oString(), 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.T
oString(), 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.T
oString();

        //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].C
olumnName) +
        "=" +
        CheckComboBox(cellComboBox.SelectedInd
ex));
        sql = @"delete from prints where
" +
        NameConverterForDelete(dt.Columns[cellCo
mboBox.SelectedIndex].ColumnName) +
        "=" +
        CheckComboBox(cellComboBox.SelectedInd
ex);
        cmd = new NpgsqlCommand(sql,

```

```

conn);
        dt.Load(cmd.ExecuteReader());
        MessageBox.Show("Удаление
успешно");
        //конец удаление поля со
значением

        conn.Close();

        CheckComboBox(cellComboBox.SelectedInd
ex);

        //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", "Наз
арович", "Николаевич", "Артурович", "Миха
илович", "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
аккаунтов 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
аккаунтов 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

```



```

        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

            }
        }
        //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

```

```

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,

```



```

    }

    private void panel1_Paint(object sender,
PaintEventArgs e)
    {

    }

    private void
cellComboBox_SelectedIndexChanged(object
sender, EventArgs e)
    {

CheckComboBox(cellComboBox.SelectedInd
ex);
    }

    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].To
String());
                    addForm.Visible = true;
                    break;
                }
                case "Заказы":
                {
                    //return "orders";
                    OrdersAddForm ordersAdd

```

```

= new OrdersAddForm();
                    ordersAdd.isUpdate = true;
                    ordersAdd.id =
int.Parse(dt.Rows[rowIndex].ItemArray[0].To
String());
                    ordersAdd.Visible = true;

                    break;
                }
                case "Заказчики":
                {
                    //return "customers";
                    CustomersAddForm
customersAdd = new CustomersAddForm();
                    customersAdd.isUpdate =
true;
                    customersAdd.id =
int.Parse(dt.Rows[rowIndex].ItemArray[0].To
String());
                    customersAdd.Visible =
true;

                    break;
                }
                case "Работники":
                {
                    //return "workers";
                    WorkersAddForm
workersAdd = new WorkersAddForm();
                    workersAdd.isUpdate =
true;
                    workersAdd.id=
int.Parse(dt.Rows[rowIndex].ItemArray[0].To
String());
                    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
частотаИспользованияФорматовToolStrip
MenuItem_Click(object sender, EventArgs e)
    {
        DiagramForm diagramForm = new
        DiagramForm();
        diagramForm.type = 2;
        diagramForm.Show();
    }

```

```

    private void
количествоЗаказовНаБанкToolStripMenuIte
m_Click(object sender, EventArgs e)
    {
        DiagramForm diagramForm = new
        DiagramForm();
        diagramForm.type = 1;
        diagramForm.Show();
    }

    private void
городаСЗаказчикамиToolStripMenuItem_Cl
ick(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
средийРазмерТиражейПоТипографииToolStrip
MenuItem_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.средни
й_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
среднийРазмерТиражейПоРайонуToolStripMenuItem_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

```

```

среднийРазмерТиражейПоГородуToolStripMenuItem_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.GetCurren
            tDirectory() + @"\" + 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.GetCurren
        tDirectory() + @"\" + 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.E
xecuteReader());

                AuthorizationForm.mainForm.conn.Close();
            }
            catch (Exception ex)
            {
                AuthorizationForm.mainForm.conn.Close();
                MessageBox.Show("Error: " +
ex.Message);
            }

            CreateHTMLFile(Extensions.GetHtml(dt),"ти
пографии");
        }

        private void
заToolStripMenuItem1_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.E
xecuteReader());

                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.E
xecuteReader());

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.E
xecuteReader());

AuthorizationForm.mainForm.conn.Close();
    }
    catch (Exception ex)
    {

AuthorizationForm.mainForm.conn.Close();
        MessageBox.Show("Error: " +
ex.Message);
    }

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

    private void
типыСобственностиToolStripMenuItem1_Cl
ick(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.E
xecuteReader());

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.E
xecuteReader());

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.E
xecuteReader());

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.E
xecuteReader());

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.E
xecuteReader());

AuthorizationForm.mainForm.conn.Close();
        }
        catch (Exception ex)
        {

AuthorizationForm.mainForm.conn.Close();
            MessageBox.Show("Error: " +
ex.Message);
        }

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

    private void
банковскиеНомераToolStripMenuItem1_Cli
ck(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.E
xecuteReader());

AuthorizationForm.mainForm.conn.Close();
        }
        catch (Exception ex)
        {

AuthorizationForm.mainForm.conn.Close();
            MessageBox.Show("Error: " +
ex.Message);
        }

CreateHTMLFile(Extensions.GetHtml(dt),
"банковские_номера");
    }

    private void
видыБумагиToolStripMenuItem1_Click(obje
ct 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.E
xecuteReader());

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.E
xecuteReader());

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();  
}  
}  
}
```