

Министерство науки и высшего образования РФ  
Федеральное государственное автономное образовательное  
учреждение высшего образования  
«Казанский (Приволжский) федеральный университет»

Институт Вычислительной математики и информационных технологий  
Кафедра системного анализа и информационных технологий

Направление: 02.03.02 – Фундаментальная информатика  
и информационные технологии  
Профиль: Системный анализ и информационные технологии

**КУРСОВАЯ РАБОТА ПО КУРСУ «ТЕХНОЛОГИИ БАЗ ДАННЫХ»**

Разработка базы данных «Раздельный сбор и вывоз бытовых отходов» и  
клиентского приложения для доступа к ней

Студент 3 курса

Группа 09-832

«\_\_\_» \_\_\_\_\_ 2020 г. \_\_\_\_\_ Кузьмина В.А.

Руководитель

к.ф.-м.н., доцент КСАИТ

«\_\_\_» \_\_\_\_\_ 2020 г. \_\_\_\_\_ Андрианова А.А.

Казань-2020

## Содержание

Содержание .....	2
Введение .....	3
Описание реляционной модели данных .....	4
Запросы к базе данных.....	7
Серверные процедуры и функции .....	10
1. Рейтинг водителей .....	10
2. Корректность расписания .....	11
Клиентское приложение .....	14
1. Авторизация.....	14
2. Страница водителя.....	16
2.1. Просмотр расписания .....	16
2.2. Добавление или изменение времени вывоза.....	17
3. Страница администратора.....	18
3.1. Редактор записи таблицы «Расписание» .....	19
3.2. Добавление новой записи в расписание .....	20
3.3. Редактор записи таблицы «Экипажи» .....	21
3.3. Добавление нового экипажа .....	21
4. Завершение работы в приложении.....	22
Заключение .....	23
Список литературы .....	24
Приложение 1. Сценарий создания базы данных .....	25
Приложение 2. Программный код клиентского приложения .....	31

## **Введение**

Приложение баз данных «Раздельный сбор и вывоз бытовых отходов» предназначено для контроля расписания вывоза мусора и регулировки совместной работы транспортных компаний и компаний по переработке отходов. Базы данных содержат необходимую информацию о компаниях, пунктах сбора, хранилищах, заводах. Клиентское приложение предназначено для администраторов транспортной компании и водителей мусоровозов. Администраторы получают доступ к таблице с расписанием и к таблицам экипажей, а водителям предоставлен ограниченный доступ к таблице с расписанием. Пользователи могут смотреть, редактировать существующие записи или добавлять новые.

Данное приложение может стать основой для создания информационной системы по поддержке переработки вторсырья. Этого можно достичь с помощью организации раздельного сбора мусора, его вывоза в хранилища и дальнейшего распределения на соответствующие заводы.

База данных была создана на основе СУБД Microsoft SQL Server [1, 2] в приложении DataGrip [3, 4]. А клиентское приложение разработано в формате Windows Forms в приложении Visual Studio на языке C# [5, 6].

## Описание реляционной модели данных

На **Рисунок 1** в свободном формате представлены все таблицы данных и логические связи между ними.

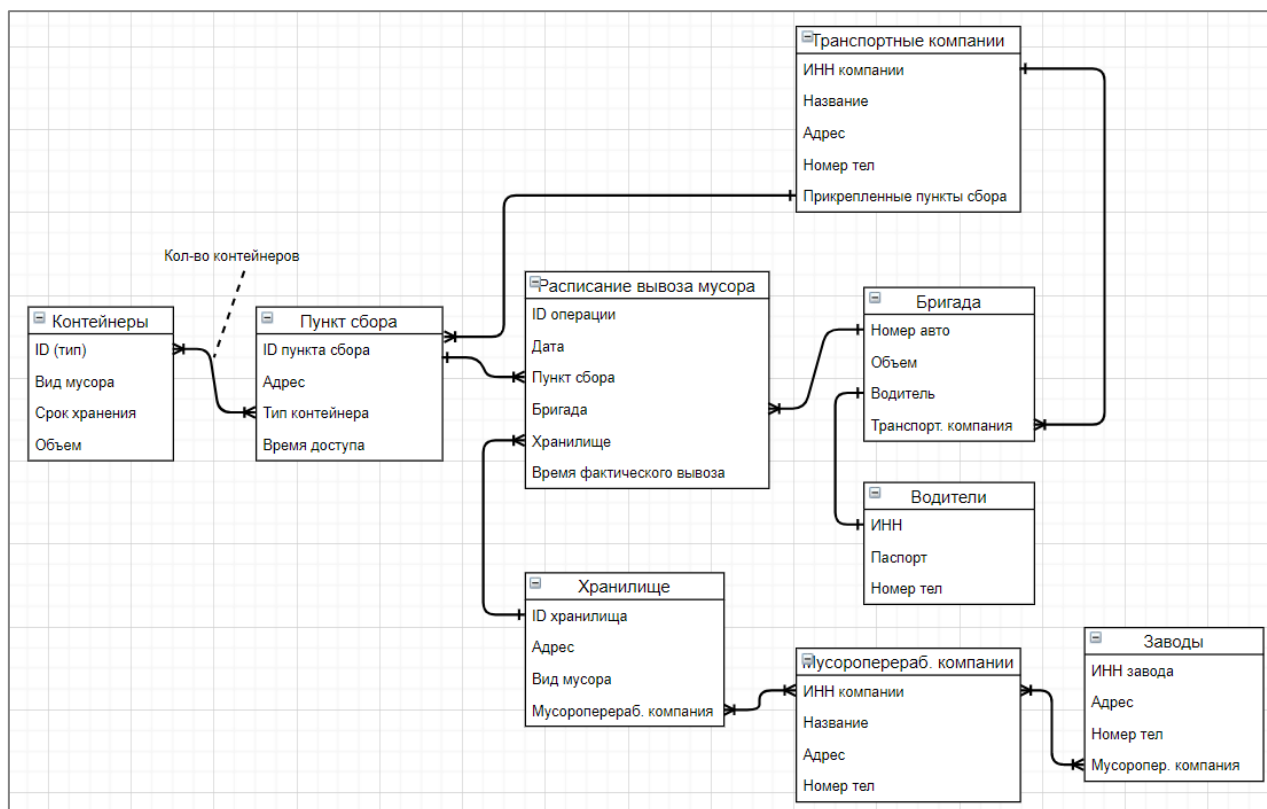


Рисунок 1 - Реляционная модель данных в свободном формате

- 1) «Расписание» — это основополагающая динамическая таблица, в которой распределено, кто-когда-откуда должен вывозить мусор. В основном записи состоят из ключей, указывающих на элементы других таблиц.
- 2) «Пункт сбора» хранит адрес пункта, типы установленных на нем контейнеров и ссылку на транспортную компанию, обслуживающую данный пункт.
- 3) «Контейнеры» — эта таблица с информацией о предназначении контейнера и особенностях хранимого в нем сырья.

- 4) «Бригада» (экипаж) соответствует парам машина-водитель, в первую очередь это номер автомобиля и указание компании, которой принадлежит этот экипаж.
- 5) «Водитель» хранит дополнительную информацию о человеке.
- 6) Таблица «Транспортные компании» содержит общую информацию о компаниях, за которыми могут быть закреплены пункты сбора мусора и бригады.
- 7) «Хранилище» - это таблица с описанием складов, где могут храниться бытовые отходы до их переработки.
- 8) «Заводы» - таблица с общей информацией о заводах.
- 9) Таблица «Мусороперерабатывающие компании» содержит общую информацию о компаниях, которым могут принадлежать хранилища и заводы.

В дальнейшем, при создании триггеров и при оформлении клиентского приложения, в базу данных были добавлены таблицы «Рейтинг» и «Логин администратора».

Итоговый вид реляционной модели с выделенными первичными и вторичными ключами, таблицами связей для множественной связи и типами данных представлен на **Рисунок 2**.

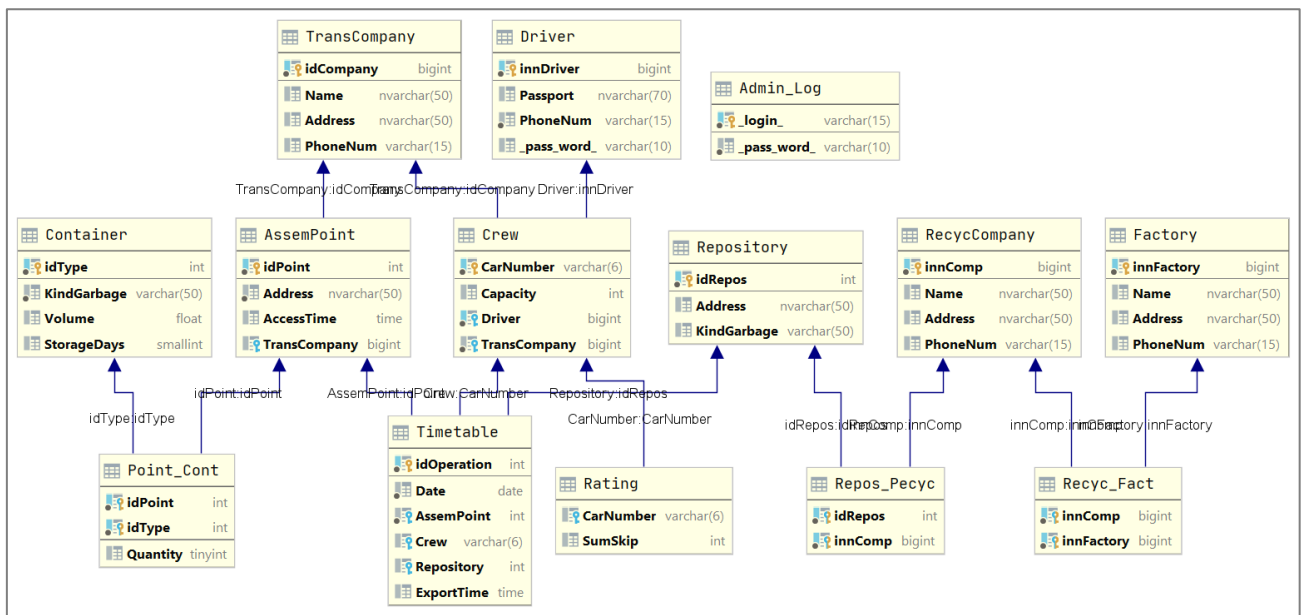


Рисунок 2 - Реляционная модель данных

## Запросы к базе данных

- 1) Запрос на создание таблицы с указанием, какой вид мусора собирается на каждом участке. [Таблица 1]

```
select Address, KindGarbage from AssemPoint
    inner join Point_Cont on AssemPoint.idPoint =
Point_Cont.idPoint
    inner join Container C on C.idType = Point_Cont.idType
order by Address
```

Данный запрос использует такие операции реляционной алгебры, как проекция и полусоединение, а также упорядочивает записи по конкретному атрибуту.

Таблица 1 - Результат запроса на вид собираемого мусора на участках

	Address	KindGarbage
1	Авангардная, 148	battery
2	Агрономическая, 80А	paper
3	Агрономическая, 80А	textile
4	Ботаническая, 14	plastic
5	Оренбургский тракт, 138Б	paper
6	Оренбургский тракт, 2	polyethylene
7	Проспект Победы, 17	paper

- 2) Поиск в таблице «Расписание» некорректных записей, в которых указаны пункт сбора и бригада, принадлежащие разным транспортным компаниям. Вывод ID таких записей, адреса пункта сбора и номера мусоровоза. В дальнейшем эту корректность записей будет проверять специальный триггер. [Таблица 2]

```
select idOperation, Address, Crew from Timetable
    inner join AssemPoint AP on AP.idPoint =
Timetable.AssemPoint
    inner join Crew C on C.CarNumber = Timetable.Crew
where not C.TransCompany=Ap.TransCompany or
AP.TransCompany is null
```

В запросе применяются операции селекции, проекции и полусоединения.

Таблица 2 - Результат запроса на некорректные записи в расписании

	idOperation	Address	Crew
1	6	Проспект Победы, 17	w777hy
2	10	Ботаническая, 14	c386at
3	14	Авангардная, 148	c386at

- 3) Запрос на поиск записей на прошлые даты, в которых не заполнено поле «Время вывоза» и подсчет количества таких записей с точки зрения соответствующих им бригадам. Выводится таблица с указанием номера машины экипажа, мобильным телефоном водителя и количеством неподтвержденных операций вывоза мусора. Этот запрос помогает создать рейтинг водителей по их добросовестности в отношении работы и заполнения графика. [Таблица 3]

```
select Crew, PhoneNum, count(*) as kolvo from Timetable
inner join Crew on Timetable.Crew = Crew.CarNumber
inner join Driver on Driver.innDriver = Crew.Driver
where Date < GETDATE() and ExportTime is null
group by Crew, PhoneNum
```

Данный запрос использует следующие операции реляционной алгебры: селекция, проекция, полусоединение и группировка.

Таблица 3 - Результат запроса на количество незаполненных записей водителем

	Crew	PhoneNum	kolvo
1	c386at	8-880-700-56-56	2
2	m223rr	8-888-888-00-01	3
3	w777hy	8-775-111-12-34	3

- 4) Поиск мусоровозов, объем которых меньше, чем объем контейнеров на участках транспортной компании, владеющей этим экипажем. С помощью этого запроса можно предположить, какие машины следует заменить более вместительными. [Таблица 4]

```
select CarNumber, Capacity, sum(Volume) as cont_volumes from
Crew
inner join AssemPoint AP on Crew.TransCompany =
AP.TransCompany
inner join Point_Cont PC on AP.idPoint = PC.idPoint
inner join Container C on C.idType = PC.idType
```



```
group by CarNumber, Capacity having
sum(Volume)>Capacity
```

Данный запрос использует следующие операции реляционной алгебры: проекция, полусоединение и группировка с наложенным условием на группы.

Таблица 4 - Результат запроса на сравнение объема машины и контейнеров

	CarNumber	Capacity	cont_volumes
1	a007aa	500	820
2	c386at	600	780
3	m223rr	380	1820

- 5) Запрос на получение списка хранилищ мусора, невостребованных в определенный промежуток времени (например, в ноябре).

Пояснение: невостребованных – значит тех, на которые не отвозился мусор.

```
select idRepos from Repository
where not exists(select Repository from Timetable
where Date>'2020-10-31' and Date<'2020-
11-30' and idRepos=Repository)
```

В этом запросе используются операции селекции, проекции и разности.

- 6) Поиск таких пунктов сбора, у которых не указано время доступа для их обслуживания, но мусор с них хоть раз вывозился в определенный промежуток времени. Этот запрос позволяет понять, в какое время двор открыт, чтобы к данному адресу в таблице дописать действующее время доступа.

```
select distinct AssemPoint from Timetable
where ExportTime>'14:00:00.0000000'
and exists(select idPoint from AssemPoint
where AccessTime is null and
AssemPoint=idPoint)
```

Данный запрос использует такие операции реляционной алгебры, как проекция, селекция, полусоединение и пересечение, а также дополнительное ограничение на наличие дубликатов.

## Серверные процедуры и функции

Для того чтобы некоторые данные в таблицах обновлялись автоматически и не допускались ошибки при добавлении некорректных записей, в базе данных содержится несколько функций и триггеров. По назначению их можно разделить на 2 смысловые группы.

### 1. Рейтинг водителей

Таблица «Рейтинг» хранит информацию о том, сколько раз каждый водитель не заполнял поле «Время вывоза» в расписании. Водитель представлен как номер его машины. Чем меньше пропусков, тем добросовестнее водитель и тем выше он в рейтинге. [Таблица 5]

Таблица 5 - Рейтинг экипажей

	CarNumber	SumSkip
1	a007aa	0
2	b666ib	0
3	c386at	2
4	m223rr	3
5	w777hy	3

Данная таблица должна обновляться каждый раз после того, как водитель вносит информацию о времени вывоза мусора с пункта сбора или удаляет данную информацию. Для этого был создан специальный триггер.

```
create trigger triggerRating on Timetable after UPDATE
as
begin
    declare @car VARCHAR(6), @extimenew TIME, @extimelast
    TIME, @date DATE;

    set @car = (select Crew from inserted);
    set @extimenew = (select ExportTime from inserted);
    set @extimelast = (select ExportTime from deleted);
    set @date = (select Date from inserted);

    if exists(select * from Rating where CarNumber = @car)
    begin
        if @date <= GETDATE() and @extimelast is null and
        @extimenew is not null
            update Rating set SumSkip = SumSkip - 1
            where CarNumber = @car;
```

```

        if @date <= GETDATE() and @extimelast is not null
and @extimenew is null
            update Rating set SumSkip = SumSkip + 1
            where CarNumber = @car;
        end;

    else
        begin
            if @date <= GETDATE() and @extimelast is null and
@extimenew is not null
                insert into Rating (CarNumber, SumSkip) values
(@car,0);
            if @date <= GETDATE() and @extimelast is not null
and @extimenew is null
                insert into Rating (CarNumber, SumSkip) values
(@car,1);
        end
    end
end

```

Этот триггер вызывается при обновлении данных в таблице «Расписание». Сначала в локальные переменные сохраняются старые значения обновляемой записи. После этого посылается запрос на получение из рейтинга записи с номером машины, соответствующим экипажу обновляемой записи расписания. Если такая запись найдена, значит в таблице «Рейтинг» уже есть информация о данном водителе и его рейтинг либо уменьшится на 1 при добавлении времени вывоза, либо увеличится на 1 при удалении информации о времени.

Если записи о таком водителе в рейтинге еще не было, тогда в таблицу «Рейтинг» добавится информация о данном водителе. Его изначальный рейтинг будет либо равен 0 (при наличии времени вывоза в записи), либо 1 (при незаполненном поле времени вывоза).

## 2. Корректность расписания

Для того чтобы проверять добавляемые в расписание записи на их корректность, создана отдельная функция.

```

create function IsNoteCorrect(@date DATE, @point INT, @crew
VARCHAR(6), @repos INT, @extime TIME) returns int
as
begin

```

```

        if exists(select * from AssemPoint where
AssemPoint.idPoint=@point)

                and exists(select * from Crew where
Crew.CarNumber=@crew)

                        and exists(select * from Repository where
Repository.idRepos=@repos)

                                and exists(select TransCompany from AssemPoint where
idPoint=@point
                                                and exists(select TransCompany from Crew
where CarNumber=@crew
                                                                and
Crew.TransCompany=AssemPoint.TransCompany))
                                        and @date>=GETDATE()
                                        and @extime is null
                                return 1;
        return 0;
end

```

Под корректностью записи понимается:

- 1) наличие указанных пункта сбора, хранилища и экипажа в соответствующих таблицах базы;
- 2) принадлежность указанных пункта сбора и экипажа одной транспортной компании;
- 3) текущая дата или будущая дата (нельзя вставлять в расписание записи на уже прошедшие даты);
- 4) пустое поле времени вывоза.

Если запись соответствует всем четырем параметрам, то функция возвращает 1, иначе 0.

Данная функция используется в триггере, срабатывающем при вставке новой записи в таблицу «Расписание»:

```

create trigger triggerNote on Timetable for INSERT
as
begin
        declare @date DATE, @point INT, @crew VARCHAR(6), @repos
INT, @extime TIME;

```

```

        set @date = (select Date from inserted);
        set @point = (select AssemPoint from inserted);
        set @crew = (select Crew from inserted);
        set @repos = (select Repository from inserted);
        set @extime = (select ExportTime from inserted);

        if dbo.IsNoteCorrect (@date,@point, @crew, @repos,
@extime) = 0
            begin
                print 'DateError: Note is not correct';
                rollback;
            end
        else
            begin
                update Rating set SumSkip = SumSkip+1
                where CarNumber=@crew;
            end
        end
    end

```

Если новая запись не прошла проверку на корректность в функции «IsNoteCorrect», то выводится соответствующее уведомление и транзакция откатывается. В ином случае запись в расписание вставляется, и обновляется таблица «Рейтинг»: количество пропусков данного водителя увеличивается на 1.

Для того чтобы не допустить заполнение поля «Время вывоза» в расписании на будущие даты, создан еще один триггер.

```

create trigger triggerExtime on Timetable for UPDATE
as
begin
    declare @extimenew TIME, @date DATE;

    set @extimenew = (select ExportTime from inserted);
    set @date = (select Date from inserted);

    if @date>GETDATE() and @extimenew is not null
        begin
            print 'DateError: Changes are not possible';
            rollback;
        end
    end

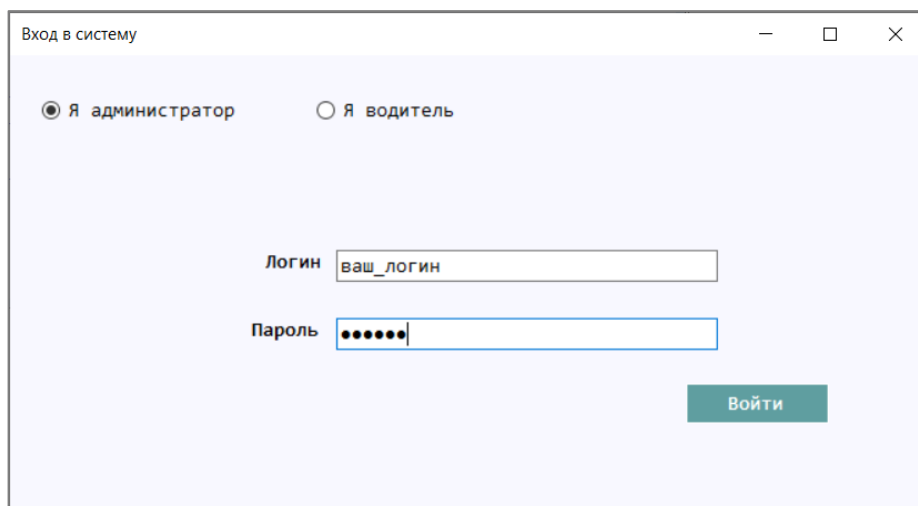
```

Если запись соответствует будущей дате, то такое обновление не допускается и выводится уведомление об ошибке.

## Клиентское приложение

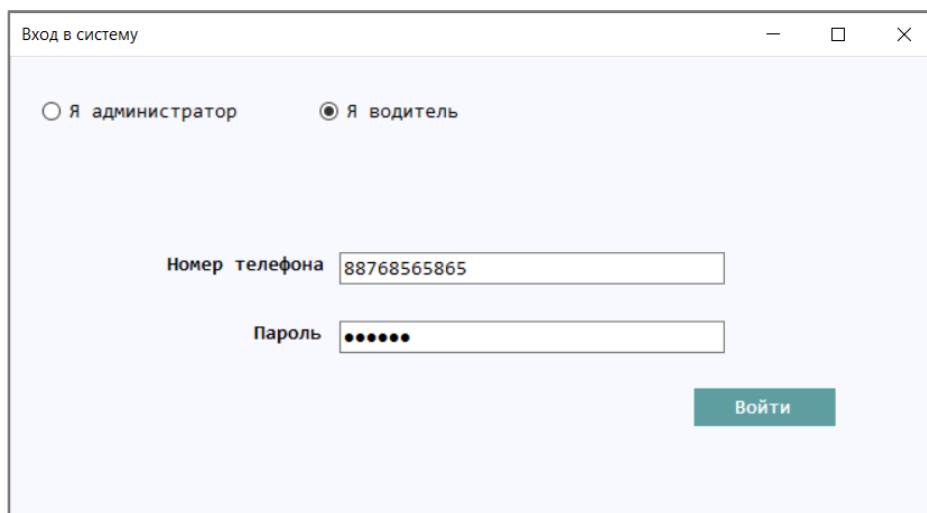
### 1. Авторизация

Полнота доступа к базе данных зависит от привилегий пользователя, поэтому при входе в приложение необходимо указать роль (администратор - Рисунок 3, водитель - Рисунок 4). После этого нужно ввести логин/мобильный телефон и пароль.



The screenshot shows a window titled "Вход в систему" (Login to the system). At the top, there are two radio buttons: "Я администратор" (I am administrator) which is selected, and "Я водитель" (I am driver). Below these, there are two input fields: "Логин" (Login) with the placeholder text "ваш\_логин" (your login), and "Пароль" (Password) with masked characters "••••••". A green button labeled "Войти" (Login) is located at the bottom right.

Рисунок 3 - Вход в качестве администратора



The screenshot shows a window titled "Вход в систему" (Login to the system). At the top, there are two radio buttons: "Я администратор" (I am administrator) and "Я водитель" (I am driver) which is selected. Below these, there are two input fields: "Номер телефона" (Phone number) with the value "88768565865", and "Пароль" (Password) with masked characters "••••••". A green button labeled "Войти" (Login) is located at the bottom right.

Рисунок 4 - Вход в качестве водителя

После нажатия на кнопку «Войти» при правильно введенных данных откроется новая форма, в другом случае появится уведомление об ошибке. (Рисунок 5)

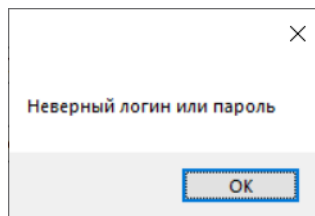


Рисунок 5 - Уведомление о некорректных данных для входа

При работе с логинами и паролями всегда возникает проблема взлома системы с помощью SQL-инъекций, потому что можно легко догадаться, какой запрос будет отправлен к базе данных, и попытаться изменить его в своих интересах. Для того чтобы вместо реального логина злоумышленник не смог вставить часть строки для запроса, необходимо передавать параметры в строку запроса не напрямую, а через параметры.

```
OleDbCommand com_admin = new OleDbCommand  
    ("select _pass_word_ from Admin_Log  
where _login_ = ?", con);  
com_admin.Parameters.AddWithValue("@log", login);
```

В данном случае модуль «Parameters» автоматически предотвращает вставку в строку переменной «login» в ее изначальном виде, если она представляет собой часть запроса.

Для того чтобы убедиться в безопасности базы данных, была проведена попытка добавить в таблицу новую запись пользователя со своим логином и паролем:

```
login = "'1'; insert into Admin_Log(_login_, _pass_word_)  
values('1', '1')";  
  
OleDbCommand com_admin = new OleDbCommand  
    ("select _pass_word_ from Admin_Log  
where _login_ = ?", con);  
com_admin.Parameters.AddWithValue("@log", login);
```

Но команда не была выполнена, потому что войти с логином = '1' и паролем = '1' в дальнейшем не удалось.

## 2. Страница водителя

Водителю будет доступна только та информация из баз данных, которая соответствует его экипажу, поэтому в верхнем углу формы закреплён номер машины. С помощью приложения водитель может либо посмотреть свое расписание в определенный промежуток времени, либо добавить время вывоза мусора к одной из записей. (Рисунок 6)

The screenshot shows a web application window titled "Личный кабинет" (Personal Cabinet). At the top, there is a text field labeled "Номер машины:" (Car number:) containing the value "a007aa". Below this, there is a section for viewing the schedule. It starts with the text "Посмотреть мое расписание с" (View my schedule from), followed by a date picker set to "26 декабря 2020 г." (26 December 2020), then the word "до" (to), another date picker set to "26 декабря 2020 г.", and a green button labeled "Посмотреть" (View). Below this section is a large, empty rectangular box. At the bottom, there is a section for adding or changing waste removal time. It starts with the text "Заполнить время вывоза мусора или изменить существующее:" (Fill in the waste removal time or change the existing one:), followed by a date picker set to "26 декабря 2020 г.", a dropdown menu, a time input field showing "00", a colon separator, another time input field showing "00", and a green button labeled "Изменить" (Change).

Рисунок 6 - Страница водителя

### 2.1. Просмотр расписания

С помощью специальных выпадающих календарей нужно указать границы временного промежутка. После нажатия на кнопку «Посмотреть» ниже отобразится таблица с расписанием для данной бригады. (Рисунок 7)



Посмотреть мое расписание с 1 октября 2020 г. до 19 ноября 2020 г. Посмотреть

	Дата	Адрес пункта вывоза	Адрес хранилища	Время вывоза
▶	01.10.20	Оренбургский тракт, 138Б	Нижний Новгород, Шувал...	16:30:00.0000000
	01.10.20	Оренбургский тракт, 2	Бугульма, ул Строитель...	12:15:00.0000000
	10.11.20	Оренбургский тракт, 138Б	Бугульма, ул Строитель...	10:30:00.0000000
	17.11.20	Оренбургский тракт, 2	Пермь, ул Соликамская,...	09:30:00.0000000

Рисунок 7 - Просмотр расписания

## 2.2. Добавление или изменение времени вывоза

Сначала нужно выбрать дату с помощью специального календаря. Если выбранная дата отсутствует в расписании, то об этом появится уведомление. (Рисунок 8)

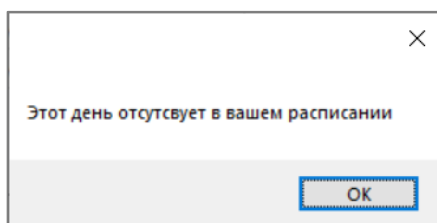


Рисунок 8 - Уведомление о неверно выбранной дате

Если была выбрана существующая дата, то в соседнем окне с выпадающим списком появятся все адреса, которые обслуживала в тот день эта бригада. (Рисунок 9)

Заполнить время вывоза мусора или изменить существующее:

1 октября 2020 г. ▼ 00 : 00 Изменить

Оренбургский тракт, 138Б  
Оренбургский тракт, 2

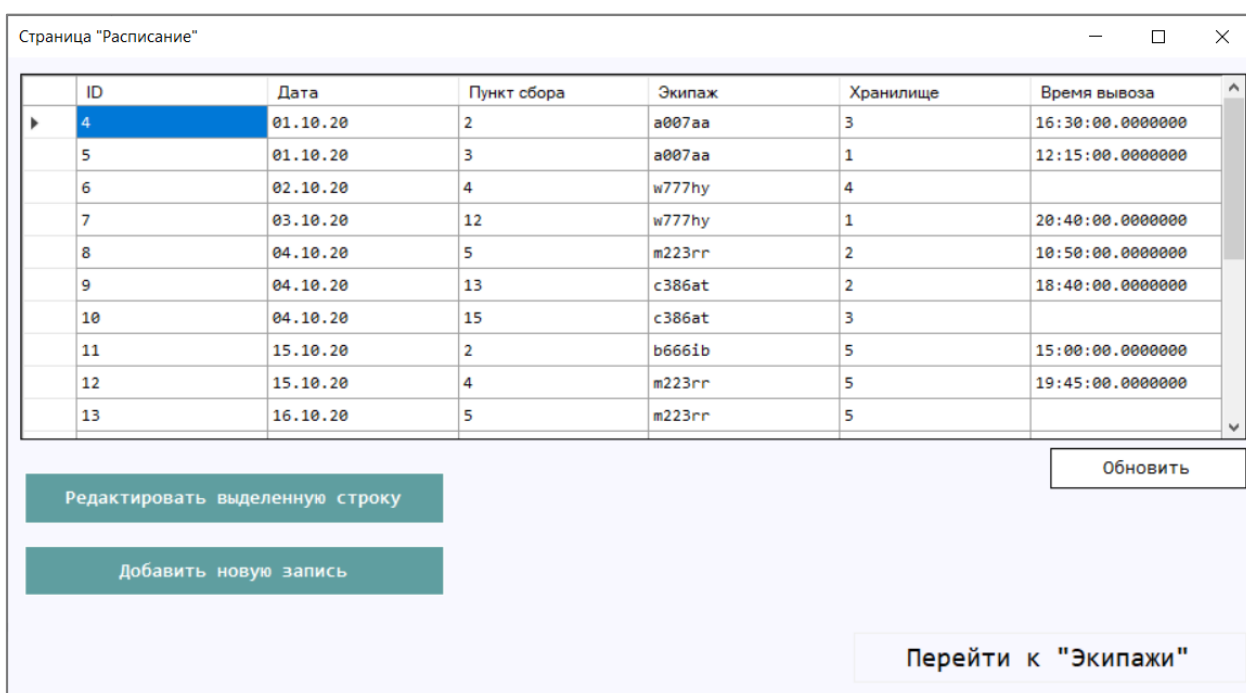
Рисунок 9 - Добавление времени вывоза

Затем необходимо вписать время (часы : минуты) и нажать кнопку «Изменить». После обновления таблицы расписания время вывоза будет добавлено/изменено.

Если вдруг пользователь заполнил не все поля, то об этом появятся соответствующие предупреждения, и запрос выполнен не будет.

### 3. Страница администратора

Изначально на странице администратора открыт раздел «Расписание», которое можно посмотреть, изменить, добавить в него новые записи (Рисунок 10). Перейти в раздел «Экипажи» можно с помощью кнопки в правом нижнем углу окна (Рисунок 11).



ID	Дата	Пункт сбора	Экипаж	Хранилище	Время вывоза
4	01.10.20	2	a007aa	3	16:30:00.000000
5	01.10.20	3	a007aa	1	12:15:00.000000
6	02.10.20	4	w777hy	4	
7	03.10.20	12	w777hy	1	20:40:00.000000
8	04.10.20	5	m223rr	2	10:50:00.000000
9	04.10.20	13	c386at	2	18:40:00.000000
10	04.10.20	15	c386at	3	
11	15.10.20	2	b666ib	5	15:00:00.000000
12	15.10.20	4	m223rr	5	19:45:00.000000
13	16.10.20	5	m223rr	5	

Редактировать выделенную строку

Добавить новую запись

Обновить

Перейти к "Экипажи"

Рисунок 10 - Страница администратора, раздел "Расписание"

Страница "Экипажи" — □ ×

	Номер машины	Объем	Трансп. компания	ИНН водителя	Паспорт водителя	Моб. телефон водителя
▶	a007aa	500	2	121615241111	9102 465738, выд...	8-000-000-34-23
	b666ib		2	123496867563		8-905-549-37-66
	c386at	600	15	144987600392		8-880-700-56-56
	m223rr	380	10	127564888888	9232 509892, выд...	8-888-888-00-01
	w777hy	450	14	144444443788	9412 636876	8-775-111-12-34

Редактировать выделенную строку

Обновить

Добавить новый экипаж

Перейти к "Расписание"

Рисунок 11 - Страница администратора, раздел "Экипажи"

### 3.1. Редактор записи таблицы «Расписание»

Чтобы изменить запись в расписании, необходимо ее выделить и нажать на соответствующую кнопку. После чего откроется специальное окно. (Рисунок 12)

Редактор записи в расписании

ID записи = 4

Дата

Пункт сбора (откуда)

Экипаж

Хранилище (куда)

1 октября 2020 г. ▾

Оренбургский тракт, 1385 ▾

a007aa ▾

Нижний Новгород, Шуваловский ▾

Отмена

Применить

Рисунок 12 - Редактор записи в расписании

Изначально все поля заполнены исходными параметрами выбранной записи. В выпадающих списках можно выбрать другие возможные варианты. Если вдруг выбранный новый пункт сбора не совпадает с выбранным экипажем, то появится уведомление. (Рисунок 13)

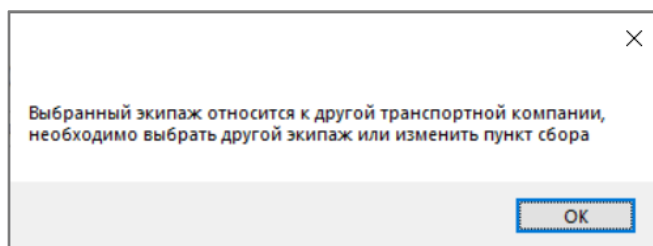


Рисунок 13 - Уведомление о несовместимости выбранного экипажа и адреса

После внесения необходимых изменений нужно нажать на «Применить». После этого либо высветится уведомление о возможных ошибках в заполнении, которые надо будет исправить, либо окно редактора закроется, что говорит об успешной операции изменения. Для того чтобы увидеть изменения в таблице, необходимо нажать на кнопку «Обновить».

### 3.2. Добавление новой записи в расписание

При выборе «Добавить новую запись» откроется новое окно, выпадающие списки которого заполнены всеми возможными вариантами. (Рисунок 14)

Рисунок 14 - Добавление новой записи в расписание

Для того чтобы добавить запись, необходимо указать все параметры. Если вдруг выбранный пункт сбора и экипаж будут принадлежать разным транспортным компаниям, то высветится окошко с предупреждением.

### 3.3. Редактор записи таблицы «Экипажи»

Чтобы изменить информацию о некоторой бригаде, необходимо выделить ее строку и нажать на соответствующую кнопку «Редактировать выделенную строку». После чего откроется специальное окно. (Рисунок 15)

Рисунок 15 - Редактор данных об экипаже

Изначально все поля заполнены исходными данными выбранной строки. Содержание текстовых полей можно изменить, а в выпадающем списке можно выбрать другую транспортную компанию из всех возможных. Поля объема и мобильного телефона ограничены в возможных вводимых данных, в них нельзя вписать никакие символы, кроме цифр. После нажатия на кнопку «Применить» все параметры экипажа заменятся новыми данными.

### 3.3. Добавление нового экипажа

При нажатии на «Добавить новый экипаж» откроется новое окно. (Рисунок 16)

Редактор экипажа

Машина

Номер\*      Объем      Трансп. компания\*

Водитель

ИНН\*      Паспортные данные      Моб. телефон\*

Отмена      Применить

Рисунок 16 - Добавление нового экипажа

Поля, помеченные звездочкой, обязательны к заполнению, остальные поля могут остаться пустыми. Перед записью нового экипажа в базу данных, будет проверено их наличие и их корректность. При возникновении одной из ошибок высветится соответствующее уведомление. (Рисунок 17)

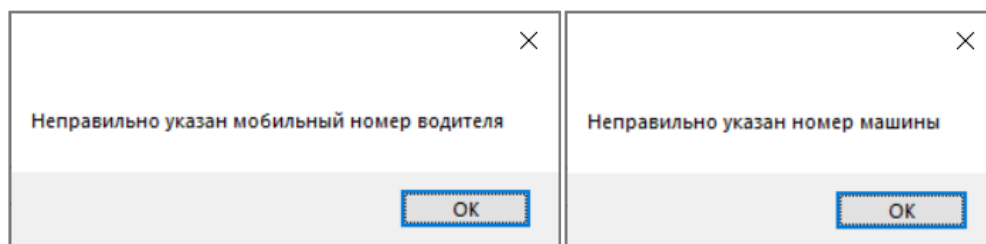


Рисунок 17 - Уведомления о некорректности данных для экипажа

#### 4. Завершение работы в приложении

После закрытия основной страницы пользователя вновь появится форма авторизации для повторного входа. Для выхода из приложения нужно закрыть форму «Вход в систему».

## **Заключение**

На примере создания базы данных «Раздельный сбор и вывоз бытовых отходов» и клиентского приложения для нее были получены следующие навыки:

- 1) формирование реляционной модели данных
- 2) управление базой с помощью SQL-запросов
- 3) получение необходимой информации из базы при помощи запросов с использованием операций реляционной алгебры
- 4) написание функций и триггеров для базы данных на языке SQL
- 5) установление связи с базой данных из приложения для клиента
- 6) использование программных средств для работы с полученной информацией из базы данных и преобразование этой информации в формат, доступный пользователю.

В результате проделанной работы создано самостоятельное приложение баз данных, которое так же может быть одной из частей большой информационной системы, занимающейся организацией процесса раздельного сбора мусора в некотором населенном пункте.

## Список литературы

1. Официальный сайт СУБД Microsoft SQL Server  
<https://www.microsoft.com/ru-ru/sql-server/sql-server-2019>
2. Документация по Microsoft SQL Server  
<https://docs.microsoft.com/ru-ru/sql/?view=sql-server-ver15>
3. Официальный сайт платформы Datagrip  
<https://www.jetbrains.com/ru-ru/datagrip/>
4. Документация по платформе Datagrip  
<https://www.jetbrains.com/help/datagrip/meet-the-product.html>
5. Официальный сайт платформы Visual Studio  
<https://visualstudio.microsoft.com/ru/>
6. Документация по разделу Visual Studio WinForms  
<https://docs.microsoft.com/ru-ru/dotnet/desktop/winforms/windows-forms-overview?view=netframeworkdesktop-4.8>



## Приложение 1. Сценарий создания базы данных

```
create table Admin_Log
(
    _login_          varchar(15) not null
        constraint Admin_Log_pk
            primary key nonclustered,
    _pass_word_      varchar(10) not null
)
go

create unique index Admin_Log__login__uindex
on Admin_Log (_login_)
go

create table Container
(
    idType          int identity
        constraint PK_Container_idType
            primary key,
    KindGarbage      varchar(50) not null,
    Volume           float,
    StorageDays      smallint
)
go

create table Driver
(
    innDriver        bigint          not null
        constraint PK_Driver_innDriver
            primary key,
    Passport          nvarchar(70),
    PhoneNum          varchar(15) not null
        constraint check_num
            check ([PhoneNum] like '[8]-[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9]'),
    _pass_word_      varchar(10)
)
go

create table Factory
(
    innFactory        bigint not null
        constraint PK_Factory_innFactory
            primary key,
    Name              nvarchar(50),
    Address           nvarchar(50),
    PhoneNum          varchar(15)
        constraint check_num_fact
            check ([PhoneNum] like '[8]-[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9]')
)
```

```

go

create table RecycCompany
(
    innComp bigint not null
        constraint PK_RecycCompany_innCompany
            primary key,
    Name      nvarchar(50),
    Address   nvarchar(50),
    PhoneNum  varchar(15)
        constraint check_num_recyc
            check ([PhoneNum] like '[8]-[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9]')
)
go

create table Recyc_Fact
(
    innComp      bigint not null
        constraint FK_RecycCompany_innCompany
            references RecycCompany,
    innFactory   bigint not null
        constraint FK_Factoryory_innFactory
            references Factory,
    constraint PR_Recyc_Fact
        primary key (innComp, innFactory)
)
go

create table Repository
(
    idRepos      int identity
        constraint PK_Repository_idRepository
            primary key,
    Address       nvarchar(50),
    KindGarbage  varchar(50)
)
go

create table Repos_Pecyc
(
    idRepos int not null
        constraint FK_Repos_Recyc
            references Repository
        constraint FK_table1_Repository_idRepository
            references Repository,
    innComp bigint not null
        constraint FK_Repos_Recyc_2
            references RecycCompany
        constraint FK_table1_RecycCompany_innCompany
            references RecycCompany,
    constraint PK_Repos_Pecyc
        primary key (idRepos, innComp)
)

```

```

)
go

create table TransCompany
(
    idCompany bigint identity
        constraint PK_TransCompany_innCompany
            primary key,
    Name        nvarchar(50),
    Address     nvarchar(50),
    PhoneNum    varchar(15)
        constraint check_num_trans
            check ([PhoneNum] like '[8]-[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9]')
)
go

create table AssemPoint
(
    idPoint      int identity
        constraint PK_AssemPoint_idPoint
            primary key,
    Address      nvarchar(50) not null,
    AccessTime   time,
    TransCompany bigint
        constraint FK_AssemPoint_TransCompany_idCompany
            references TransCompany
)
go

create table Crew
(
    CarNumber    varchar(6) not null
        constraint PK_Crew_CarNumber
            primary key,
    Capacity     int,
    Driver       bigint      not null
        constraint KEY_Crew_Driver
            unique
        constraint u_driver
            unique
        constraint FK_Crew
            references Driver,
    TransCompany bigint      not null
        constraint FK_Crew_TransCompany_innCompany
            references TransCompany
)
go

create table Point_Cont
(
    idPoint      int not null
        constraint FK_Point_Cont

```

```

        references AssemPoint,
idType    int not null
        constraint FK_Point_Cont_Container_idType
            references Container,
Quantity tinyint,
constraint PK_Point_Cont
    primary key (idType, idPoint)
)
go

create table Rating
(
    CarNumber varchar(6)
        constraint fk_rating
            references Crew,
    SumSkip    int
)
go

create table Timetable
(
    idOperation int identity
        constraint PK_Timetable_idOperation
            primary key,
    Date         date not null,
    AssemPoint   int not null
        constraint FK_Timetable_AssemPoint_idPoint
            references AssemPoint,
    Crew         varchar(6)
        constraint FK_Timetable_Crew_CarNumber
            references Crew,
    Repository   int
        constraint FK_Timetable_Repository_idRepository
            references Repository,
    ExportTime   time
)
go

create trigger triggerExtime
on Timetable
for UPDATE
as
begin
    --создание необх переменных
    declare @extimenew TIME, @date DATE;

    set @extimenew = (select ExportTime from inserted);
    set @date = (select Date from inserted);

    if @date > GETDATE() and @extimenew is not null --хотят
поставить время на еще не совершенную операцию
    begin
        print 'DateError: Changes are not possible';
    end
end

```

```

        rollback;
    end
end
go

create trigger triggerNote
    on Timetable
    for INSERT
    as
begin
    --создание необх переменных
    declare @date DATE, @point INT, @crew VARCHAR(6), @repos
    INT, @extime TIME;

    --чтение данных новой записи в переменные
    set @date = (select Date from inserted);
    set @point = (select AssemPoint from inserted);
    set @crew = (select Crew from inserted);
    set @repos = (select Repository from inserted);
    set @extime = (select ExportTime from inserted);

    if dbo.IsNoteCorrect(@date, @point, @crew, @repos, @extime)
= 0
        begin
            print 'DateError: Note is not correct';
            rollback;
        end
    else
        if @extime is null --прибавляем пока пропуск в таблицу
рейтинга
            begin
                update Rating
                set SumSkip = SumSkip + 1
                where CarNumber = @crew
            end
end
go

create trigger triggerRating
    on Timetable
    after UPDATE
    as
begin
    declare @car VARCHAR(6), @extimenew TIME, @extimelast
    TIME, @date DATE;

    set @car = (select Crew from inserted);
    set @extimenew = (select ExportTime from inserted);
    set @extimelast = (select ExportTime from deleted);
    set @date = (select Date from inserted);

    if exists(select * from Rating where CarNumber = @car)
        begin

```

```

        if @date <= GETDATE() and @extimelast is null and
@extimenew is not null --если только вписали время вывоза
            update Rating
            set SumSkip = SumSkip - 1
            where CarNumber = @car;
        if @date <= GETDATE() and @extimelast is not null
and @extimenew is null --если удалили время вывоза
            update Rating
            set SumSkip = SumSkip + 1
            where CarNumber = @car;
    end;
else --добавляем новую запись
    begin
        if @date <= GETDATE() and @extimelast is null and
@extimenew is not null
            insert into Rating (CarNumber, SumSkip) values
(@car, 0);
        if @date <= GETDATE() and @extimelast is not null
and @extimenew is null
            insert into Rating (CarNumber, SumSkip) values
(@car, 1);
    end
end
go

create function IsNoteCorrect(@date DATE, @point INT, @crew
VARCHAR(6), @repos INT, @extime TIME) returns int
as
begin
    if exists(select * from AssemPoint where AssemPoint.idPoint
= @point)
        and exists(select * from Crew where Crew.CarNumber =
@crew)
            and exists(select * from Repository where
Repository.idRepos = @repos)
                and exists(select TransCompany
from AssemPoint
where idPoint = @point
and exists(select TransCompany
from Crew
where CarNumber = @crew
and Crew.TransCompany =
AssemPoint.TransCompany))
                    and @date >= GETDATE() --чтобы записи на старые даты не
вставляли
                        and @extime is null --чтобы заранее не заполняли время
фактич вывоза
                            return 1;
                                return 0;
end
go

```

## Приложение 2. Программный код клиентского приложения

### Форма «Вход в систему»:

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.OleDb;    // для связи с БД
using System.Windows.Forms;

namespace AppForDB_Recycling
{
    public partial class Вход_в_систему : Form
    {
        public Вход_в_систему()
        {
            InitializeComponent();

            //попытка авторизации
            private void but_Try_login_Click(object sender,
            EventArgs e)
            {
                if (text_Login.Text == "" || text_Password.Text ==
                "")
                {
                    MessageBox.Show("Заполните поля \"Логин\" и
                    \"Пароль\"");
                    return;
                }

                string login = text_Login.Text;
                string password = text_Password.Text;

                // проверка логина-пароля администратора
                if (rbut_I_am_admin.Checked)
                {
                    string forconnect =
                    "Provider=SQLNCLI11.1;Integrated Security=SSPI;Persist Security
                    Info=False;User ID=\"\";Initial Catalog=Recycling;Data
                    Source=(local);Initial File Name=\"\";Server SPN=\"\"";
                    OleDbConnection con = new
                    OleDbConnection(forconnect);
                    con.Open();

                    OleDbCommand com_admin = new
                    OleDbCommand("select _pass_word_ from Admin_Log where _login_ =
                    ?", con);

                    com_admin.Parameters.AddWithValue("@log", login);
                    login = "'1'"; insert into Admin_Log(_login_,
                    _pass_word_) values('1', '1')";
```

```

        OleDbDataReader cursor_admin =
com_admin.ExecuteReader();
        string bd_password = "";
        while (cursor_admin.Read())
            bd_password = cursor_admin[0].ToString();

        cursor_admin.Close();
        con.Close();

        if (password != bd_password)
        {
            MessageBox.Show("Неверный логин или
пароль");
            return;
        }
        else
        {
            Страница_админа newform = new
Страница_админа(this);
            this.Hide();
            text_Login.Text = "";
            text_Password.Text = "";
            newform.Show();
        }
    }

    // проверка логина-пароля водителя
    if (rbut_I_am_driver.Checked)
    {
        long login_phone = Convert.ToInt64(login);
        if (login_phone >= 800000000000 && login_phone <=
899999999999)
            login = string.Format("{0:##-###-###-##-##}",
login_phone);
        else
        {
            MessageBox.Show("Некорректный номер
телефона");
            return;
        }

        string forconnect = "...";
        OleDbConnection con = new
OleDbConnection(forconnect);
        con.Open();

        OleDbCommand com_driver = new
OleDbCommand("select * from Driver", con);
        OleDbDataReader cursor_driver =
com_driver.ExecuteReader();
        while (cursor_driver.Read())
        {

```



```

        if ((cursor_driver["Phonenum"].ToString() ==
login) && (cursor_driver["_pass_word_"].ToString() == password))
        {
            //ищу соответсвующий номер машины по
номеру телефона водителя
            OleDbCommand com_crew = new
OleDbCommand("select CarNumber, Driver from Crew", con);
            OleDbDataReader cursor_crew =
com_crew.ExecuteReader();
            string crew_num = "";
            while (cursor_crew.Read())
            {
                if (cursor_crew["Driver"].ToString()
== cursor_driver["innDriver"].ToString())
                    crew_num =
cursor_crew["CarNumber"].ToString();
            }
            cursor_driver.Close();
            cursor_crew.Close();
            con.Close();

            Страница_водителя newform = new
Страница_водителя(crew_num, this);
            text_Login.Text = "";
            text_Password.Text = "";
            this.Hide();
            newform.Show();
            return;
        }
    }
    //если мы здесь, значит данные неправильные
    cursor_driver.Close();
    con.Close();
    MessageBox.Show("Неверный номер телефона или
пароль");
}
}

private void rbut_I_am_driver_CheckedChanged(object
sender, EventArgs e)
{
    label_log.Text = "Номер телефона";
}

private void rbut_I_am_admin_CheckedChanged(object
sender, EventArgs e)
{
    label_log.Text = "Логин";
}
}
}

```

## Форма «Страница водителя»:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.OleDb;
using System.Windows.Forms;

namespace AppForDB_Recycling
{
    public partial class Страница_водителя : Form
    {
        /*для пользователя (водителя):
        - посмотреть свое расписание в определенный
        промежуток дней (табл Расписание)
        - заполнить время вывоза (табл Расписание)*/

        public Страница_водителя(string car, Form Login)
        {
            InitializeComponent();
            Login_form = Login;

            label_carnum.Text = label_carnum.Text + car;
            label_carnum.Tag = car;
        }

        // показ таблицы с расписанием на определенные даты
        private void but_show_table_Click(object sender,
            EventArgs e)
        {
            datagrid_Timetab.Rows.Clear(); //старое удаляю

            string forconnect = "...";
            OleDbConnection con = new
            OleDbConnection(forconnect);
            con.Open();

            OleDbCommand com_timetable = new
            OleDbCommand("select Date, AP.Address, R2.Address, ExportTime from
            Timetable " +

            "inner join AssemPoint AP on Timetable.AssemPoint = AP.idPoint " +

            "inner join Repository R2 on Timetable.Repository = R2.idRepos " +

            "where Crew = ? and Date >= ? and Date <= ?", con);
            com_timetable.Parameters.AddWithValue("@car",
            label_carnum.Tag.ToString());
        }
    }
}
```

```

        DateTime from = new DateTime(date_from.Value.Year,
date_from.Value.Month, date_from.Value.Day, 0, 0, 0);
        DateTime to = new DateTime(date_to.Value.Year,
date_to.Value.Month, date_to.Value.Day, 0, 0, 0);
        com_timetable.Parameters.AddWithValue("@date_from",
from);
        com_timetable.Parameters.AddWithValue("@date_to",
to);

        //создаю таблицу без привязки данных
datagrid_Timetab.ColumnCount = 4;
datagrid_Timetab.Columns[0].Name = "Дата";
datagrid_Timetab.Columns[1].Name = "Адрес пункта
вывоза";
datagrid_Timetab.Columns[2].Name = "Адрес
хранилища";
datagrid_Timetab.Columns[3].Name = "Время вывоза";

        OleDbDataReader cursor_tt =
com_timetable.ExecuteReader();
        while (cursor_tt.Read())

datagrid_Timetab.Rows.Add(cursor_tt[0].ToString().Substring(0,
8), cursor_tt[1], cursor_tt[2], cursor_tt[3]);

        cursor_tt.Close();
        con.Close();
    }

    Dictionary<object, object> id_addr = new
Dictionary<object, object>(); //будет хранить адреса и их id

    // добавление в конкретную запись бд время вывоза
    private void but_add_extime_Click(object sender,
EventArgs e)
    {
        if (comboBox_address.Text == "")
        {
            MessageBox.Show("Выберите адрес");
            return;
        }
        if (text_extime_hour.Text == "" ||
text_extime_min.Text == "")
        {
            MessageBox.Show("Введите время вывоза");
            return;
        }

        string forconnect = "...";
        OleDbConnection con = new
OleDbConnection(forconnect);
        con.Open();

```

```

        //обработка значений, по которым будут изменения в
таблице
        DateTime date = new DateTime(date_extime.Value.Year,
date_extime.Value.Month, date_extime.Value.Day, 0, 0, 0);
        int point =
Convert.ToInt32(id_addr[comboBox_address.Text]);
        if (text_extime_hour.Text.Length == 1)
            text_extime_hour.Text = "0" +
text_extime_hour.Text;
        string extime = text_extime_hour.Text + ":" +
text_extime_min.Text + ":00.00000000";

        OleDbCommand com_update = new OleDbCommand("update
Timetable set ExportTime = ? where Date = ? and AssemPoint = ?
and Crew = ?", con);
        com_update.Parameters.AddWithValue("@extime",
extime);
        com_update.Parameters.AddWithValue("@date",
date.ToString());
        com_update.Parameters.AddWithValue("@point", point);
        com_update.Parameters.AddWithValue("@crew",
label_carnum.Tag.ToString());

        com_update.ExecuteNonQuery();
        con.Close();

        //удаление всех заполненных полей
comboBox_address.Items.Clear();
text_extime_hour.Text = "00";
text_extime_min.Text = "00";
    }

    // заполнение доступных адресов в выбранную дату
private void date_extime_ValueChanged(object sender,
EventArgs e)
{
    //очистить предыдущие адреса
comboBox_address.Items.Clear();
id_addr.Clear();

    if (date_extime.Value > DateTime.Now)
    {
        MessageBox.Show("Нельзя редактировать записи на
будущие даты");
        return;
    }

    string forconnect = "...";
    OleDbConnection con = new
OleDbConnection(forconnect);

```

```

        con.Open();

        OleDbCommand com_check_date = new
OleDbCommand("select AssemPoint, Address from Timetable " +
"inner join AssemPoint AP on AP.idPoint = Timetable.AssemPoint " +
+
"where Crew = ? and Date = ?", con);
        com_check_date.Parameters.AddWithValue("@car",
label_carnum.Tag.ToString());

        string str_date_ex =
date_extime.Value.Year.ToString() + '-' +
date_extime.Value.Month.ToString() + '-' +
date_extime.Value.Day.ToString();
        com_check_date.Parameters.AddWithValue("@date",
str_date_ex);

        OleDbDataReader cursor_addr =
com_check_date.ExecuteReader(); //здесь адреса пунктов в ту дату
(если есть)

        while (cursor_addr.Read())
            id_addr.Add(cursor_addr[1], cursor_addr[0]);
//key = "address", value = his id

        if (id_addr.Count == 0)
        {
            MessageBox.Show("Этот день отсутствует в вашем
расписании");
            con.Close();
            return;
        }
        //заполняем адресами той даты список, чтобы водитель
выбрал 1 из них
        foreach (object item in id_addr.Keys)
            comboBox_address.Items.Add(item);

        con.Close();
    }

    private Form Login_form = new Form(); // главная форма
    private void Form_closed(object sender,
FormClosedEventArgs e)
    {
        // возвращение главной формы
        Login_form.Show();
    }

```

```

        // ограничение на ввод некорректных символов в поля с
        // числовыми данными
        private void text_extime_hour_KeyPress(object sender,
        KeyPressEventArgs e)
        {
            //берем каждый введенный символ и пишем только те,
            //которые будут цифрами или кл удаления
            char sym = e.KeyChar;
            if (!Char.IsDigit(sym) && sym != 8)
                e.Handled = true;
        }

        private void text_extime_min_KeyPress(object sender,
        KeyPressEventArgs e)
        {
            char sym = e.KeyChar;
            if (!Char.IsDigit(sym) && sym != 8)
                e.Handled = true;
        }
    }
}

```

### Форма «Страница администратора»:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.OleDb;
using System.Windows.Forms;

namespace AppForDB_Recycling
{
    public partial class Страница_админа : Form
    {
        /*для админа:
        - добавить/изменить запись в расписании
        - добавить/изменить экипаж = водитель+машина*/

        public Страница_админа(Form Login)
        {
            InitializeComponent();
            Login_form = Login;

            // Tag = 0 (Расписание), Tag = 1 (Экипажи)
            this.Tag = 0;
            Update_Table_Time();
        }

        // обновление таблицы "Расписание"
        private void Update_Table_Time()

```

```

        {
            string forconnect = "...";
            OleDbConnection con = new
OleDbConnection(forconnect);
            con.Open();

            OleDbCommand com_time = new OleDbCommand("select *
from Timetable", con);
            datagrid_table.Rows.Clear();
            datagrid_table.ColumnCount = 6;
            datagrid_table.Columns[0].Name = "ID";
            datagrid_table.Columns[1].Name = "Дата";
            datagrid_table.Columns[2].Name = "Пункт сбора";
            datagrid_table.Columns[3].Name = "Экипаж";
            datagrid_table.Columns[4].Name = "Хранилище";
            datagrid_table.Columns[5].Name = "Время вывоза";

            OleDbDataReader cursor_time =
com_time.ExecuteReader();
            while (cursor_time.Read())
                datagrid_table.Rows.Add(cursor_time[0],
cursor_time[1].ToString().Substring(0, 8), cursor_time[2],
cursor_time[3],
cursor_time[4], cursor_time[5].ToString());

            cursor_time.Close();
            con.Close();
        }

        // обновление таблицы "Экипажи"
        private void Update_Table_Crew()
        {
            string forconnect = "...";
            OleDbConnection con = new
OleDbConnection(forconnect);
            con.Open();

            OleDbCommand com_crew = new OleDbCommand("select
CarNumber, Capacity, TransCompany, innDriver, Passport, PhoneNum
from Crew " +
                                                    "inner join
Driver D on D.innDriver = Crew.Driver", con);
            datagrid_table.Rows.Clear();
            datagrid_table.ColumnCount = 6;
            datagrid_table.Columns[0].Name = "Номер машины";
            datagrid_table.Columns[1].Name = "Объем";
            datagrid_table.Columns[2].Name = "Трансп. компания";
            datagrid_table.Columns[3].Name = "ИНН водителя";
            datagrid_table.Columns[4].Name = "Паспорт водителя";
            datagrid_table.Columns[5].Name = "Моб. телефон
водителя";

```

```

        OleDbDataReader cursor_crew =
com_crew.ExecuteReader();
        while (cursor_crew.Read())
            datagrid_table.Rows.Add(cursor_crew[0],
cursor_crew[1], cursor_crew[2], cursor_crew[3], cursor_crew[4],
cursor_crew[5]);

        cursor_crew.Close();
        con.Close();
    }

    private void but_update_table_Click(object sender,
EventArgs e)
    {
        if ((int)this.Tag == 0)
            Update_Table_Time();
        else
            Update_Table_Crew();
    }

    // переход на специальную форму обновления определенной
записи
    private void but_update_str_Click(object sender,
EventArgs e)
    {
        if ((int)this.Tag == 0)
        {
            int choosed_id =
(int)datagrid_table.CurrentRow.Cells[0].Value;
            Редактор_расписания newform = new
Редактор_расписания(choosed_id);
            newform.Show();
        }
        else
        {
            string carnumber =
datagrid_table.CurrentRow.Cells[0].Value.ToString();
            string inn =
datagrid_table.CurrentRow.Cells[3].Value.ToString();
            Редактор_экипажа newform = new
Редактор_экипажа(carnumber, inn);
            newform.Show();
        }
    }

    // переход на специальную форму добавления новой записи
    private void but_insert_str_Click(object sender,
EventArgs e)
    {
        if ((int)this.Tag == 0)
        {

```



```

        Редактор_расписания newform = new
Редактор_расписания(-1);
        newform.Show();
    }
    else
    {
        Редактор_экипажа newform = new
Редактор_экипажа("", "");
        newform.Show();
    }
}

// Из раздела "Расписания" в "Экипажи" и наоборот
private void but_change_crew_time_Click(object sender,
EventArgs e)
{
    // Tag = 0 (Расписание), Tag = 1 (Экипажи)
    if ((int)this.Tag == 0)
    {
        this.Tag = 1;
        this.Text = "Страница \"Экипажи\"";
        but_change_crew_time.Text = "Перейти к
\"Расписание\"";
        but_insert_str.Text = "Добавить новый экипаж";
        Update_Table_Crew();
        return;
    }

    if ((int)this.Tag == 1)
    {
        this.Tag = 0;
        this.Text = "Страница \"Расписание\"";
        but_change_crew_time.Text = "Перейти к
\"Экипажи\"";
        but_insert_str.Text = "Добавить новую запись";
        Update_Table_Time();
    }
}

private Form Login_form = new Form(); // главная форма
private void Form_closed(object sender,
FormClosedEventArgs e)
{
    // возвращение главной формы
    Login_form.Show();
}
}
}

```

## Форма «Редактор записи расписания»:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.OleDb;
using System.Windows.Forms;

namespace AppForDB_Recycling
{
    public partial class Редактор_расписания : Form
    {
        public Редактор_расписания(int id)
        {
            InitializeComponent();
            string forconnect = "...";
            OleDbConnection con = new
OleDbConnection(forconnect);
            con.Open();

            //заполнение списков с выбором
            OleDbCommand com_assem = new OleDbCommand("select
Address from AssemPoint", con);
            OleDbCommand com_crew = new OleDbCommand("select
CarNumber from Crew", con);
            OleDbCommand com_repos = new OleDbCommand("select
Address from Repository", con);

            OleDbDataReader cursor_assem =
com_assem.ExecuteReader();
            while (cursor_assem.Read())
                comboBox_assem.Items.Add(cursor_assem[0]);
            cursor_assem.Close();
            OleDbDataReader cursor_crew =
com_crew.ExecuteReader();
            while (cursor_crew.Read())
                comboBox_crew.Items.Add(cursor_crew[0]);
            cursor_crew.Close();
            OleDbDataReader cursor_repos =
com_repos.ExecuteReader();
            while (cursor_repos.Read())
                comboBox_repos.Items.Add(cursor_repos[0]);
            cursor_repos.Close();

            //если будем новое создавать, а не старое редачить
            if (id == -1)
            {
                this.Text = "Добавление_записи";
                but_done.Text = "Добавить";
                label_id.Visible = false;
                label_id.Tag = -1;
            }
        }
    }
}
```

```

        con.Close();
        return;
    }

    label_id.Text = label_id.Text + id.ToString();
    label_id.Tag = id;

    //установка значений как в первоначальной записи
    OleDbCommand com_thisstr = new OleDbCommand("select
idOperation, Date, AP.Address, Crew, R2.Address from Timetable "
+
"inner join AssemPoint AP on Timetable.AssemPoint = AP.idPoint "
+
"inner join Repository R2 on Timetable.Repository = R2.idRepos "
+
"where idOperation = ?", con);
    com_thisstr.Parameters.AddWithValue("@id", id);

    OleDbDataReader cursor_str =
com_thisstr.ExecuteReader();
    while (cursor_str.Read())
    {
        comboBox_assem.Text = cursor_str[2].ToString();
        comboBox_crew.Text =
cursor_str["Crew"].ToString();
        comboBox_repos.Text = cursor_str[4].ToString();
        date_operation.Value =
(DateTime)cursor_str["Date"];
    }
    cursor_str.Close();
    con.Close();
}

private void but_cancel_Click(object sender, EventArgs e)
{
    this.Close();
}

// внесение изменений в базу данных
private void but_done_Click(object sender, EventArgs e)
{
    if (date_operation.Value < DateTime.Now)
    {
        MessageBox.Show("Нельзя добавить запись на
старые даты");
        return;
    }
    if (comboBox_assem.Text == "" || comboBox_crew.Text
== "" || comboBox_repos.Text == "")

```

```

        {
            MessageBox.Show("Есть пустые данные, необходимо
выбрать все параметры записи");
            return;
        }

        string forconnect = "...";
        OleDbConnection con = new
OleDbConnection(forconnect);
        con.Open();

        //поиск id адресов
        int id_assem = 0; int id_repos = 0;
        OleDbCommand com_assem = new OleDbCommand("select
idPoint, Address from AssemPoint", con);
        OleDbCommand com_repos = new OleDbCommand("select
idRepos, Address from Repository", con);

        OleDbDataReader cursor_assem =
com_assem.ExecuteReader();
        while (cursor_assem.Read())
            if (comboBox_assem.Text ==
cursor_assem["Address"].ToString())
                id_assem = (int)cursor_assem["idPoint"];
        cursor_assem.Close();

        OleDbDataReader cursor_repos =
com_repos.ExecuteReader();
        while (cursor_repos.Read())
            if (comboBox_repos.Text ==
cursor_repos["Address"].ToString())
                id_repos = (int)cursor_repos["idRepos"];
        cursor_repos.Close();

        if ((int)label_id.Tag == -1) //если добавляем новую
запись
        {
            OleDbCommand com_add = new OleDbCommand("insert
into Timetable (Date, AssemPoint, Crew, Repository) values (?,
?, ?, ?)", con);

            com_add.Parameters.AddWithValue("@date",
date_operation.Value);
            com_add.Parameters.AddWithValue("@assem",
id_assem);
            com_add.Parameters.AddWithValue("@crew",
comboBox_crew.Text);
            com_add.Parameters.AddWithValue("@repos",
id_repos);

            com_add.ExecuteNonQuery();
        }
        else //если редачим старую запись

```

```

        {
            OleDbCommand com_update = new
OleDbCommand("update Timetable set Date = ?, AssemPoint = ?,
Crew = ?, Repository = ? " +

"where idOperation = ?", con);
            com_update.Parameters.AddWithValue("@date",
date_operation.Value);
            com_update.Parameters.AddWithValue("@assem",
id_assem);
            com_update.Parameters.AddWithValue("@crew",
comboBox_crew.Text);
            com_update.Parameters.AddWithValue("@repos",
id_repos);
            com_update.Parameters.AddWithValue("@id",
(int)label_id.Tag);

            com_update.ExecuteScalar();
        }

        con.Close();
        this.Close();
    }

    private void comboBox_assem_change(object sender,
EventArgs e)
    {
        //адрес и машина должны быть одной компании
        comboBox_crew.Items.Clear();

        string forconnect = "...";
        OleDbConnection con = new
OleDbConnection(forconnect);
        con.Open();

        long id_transcom = 0;
        bool is_transcom = true;
        OleDbCommand com_assem = new OleDbCommand("select
Address, TransCompany from AssemPoint", con);
        OleDbDataReader cursor_assem =
com_assem.ExecuteReader();
        while (cursor_assem.Read())
        {
            if (comboBox_assem.Text ==
cursor_assem["Address"].ToString())
            {
                // если компания не указана
                if (cursor_assem["TransCompany"].ToString()
== "")
                {
                    is_transcom = false;
                    break;

```

```

        }
        id_transcom =
(long)cursor_assem["TransCompany"];
    }
}
cursor_assem.Close();

// если компания не указана, тогда с данного адреса
не увозят мусор
    if (!is_transcom)
    {
        MessageBox.Show("Сейчас нет экипажей, которые
обслуживают данный пункт сбора");
        con.Close();
        return;
    }

//добавим только те машины, которые принадлежат этой
компании
    bool right_chooosed = false;

    OleDbCommand com_crew = new OleDbCommand("select
CarNumber, TransCompany from Crew", con);
    OleDbDataReader cursor_crew =
com_crew.ExecuteReader();
    while (cursor_crew.Read())
    {
        if ((long)cursor_crew["TransCompany"] ==
id_transcom)
        {
            comboBox_crew.Items.Add(cursor_crew["CarNumber"]);
            //соответ-т ли выделенный вариант одному из
возможных
            if (cursor_crew["CarNumber"].ToString() ==
comboBox_crew.Text)
                right_chooosed = true;
        }
    }
    cursor_crew.Close();
    con.Close();

    if (comboBox_crew.Text == "") // еще не выбрали
машину
        return;
    if (!right_chooosed) // если нет совпадений со
выбранной машиной
    {
        //значит надо убрать текущий вариант
        comboBox_crew.Text = "";
        MessageBox.Show("Выбранный экипаж относится к
другой транспортной компании, необходимо выбрать другой экипаж
или изменить пункт сбора");
    }

```

```

        }
    }

    private void date_operation_ValueChanged(object sender,
EventArgs e)
    {
        // нельзя добавлять новую запись на старые даты
        if ((date_operation.Value < DateTime.Now) &&
((int)label_id.Tag == -1))
        {
            MessageBox.Show("Нельзя добавить запись на
старые даты");
            return;
        }
    }
}
}
}

```

### Форма «Редактор данных об экипаже»:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.OleDb;
using System.Windows.Forms;

namespace AppForDB_Recycling
{
    public partial class Редактор_экипажа : Form
    {
        public Редактор_экипажа(string carnum, string inn)
        {
            InitializeComponent();
            string forconnect = "...";
            OleDbConnection con = new
OleDbConnection(forconnect);
            con.Open();

            if (carnum == "") // если новый экипаж добавлять
            {
                OleDbCommand com_trans = new
OleDbCommand("select Name from TransCompany", con);
                OleDbDataReader cursor_trans =
com_trans.ExecuteReader();
                while (cursor_trans.Read())
                    comboBox_transp.Items.Add(cursor_trans[0]);

                cursor_trans.Close();
            }
        }
    }
}

```

```

        con.Close();

        label_carnum.Tag = -1;
        return;
    }

    //если передадим сущ экипаж
    label_carnum.Text = label_carnum.Text + ": " +
carnum;

    label_carnum.Tag = carnum;
    textBox_carnum.Visible = false;
    label_carnum_title.Visible = false;

    label_driver_inn.Text = label_driver_inn.Text + ",
ИИН: " + inn;
    label_driver_inn.Tag = inn;
    textBox_inn.Visible = false;
    label_driver_inn_title.Visible = false;

    //заполнение исходными данными сначала
    OleDbCommand com_crew = new OleDbCommand("select
Capacity, TransCompany, innDriver, Passport, PhoneNum from Crew
" +
                                                                    "inner join
Driver D on D.innDriver = Crew.Driver " +
                                                                    "where
CarNumber = ?", con);
    com_crew.Parameters.AddWithValue("@carnum", carnum);

    OleDbDataReader cursor_crew =
com_crew.ExecuteReader();
    while (cursor_crew.Read())
    {
        textBox_capacity.Text =
cursor_crew["Capacity"].ToString();
        comboBox_transp.Tag =
cursor_crew["TransCompany"]; //id пока
        textBox_passport.Text =
cursor_crew["Passport"].ToString();

        //приведение моб номера к нормальному виду без -
        textBox_mobile.Text =
cursor_crew["PhoneNum"].ToString().Replace("-", string.Empty);
    }
    cursor_crew.Close();

    //заполнение трансп компаний
    OleDbCommand com_transp = new OleDbCommand("select
idCompany, Name from TransCompany", con);
    OleDbDataReader cursor_transp =
com_transp.ExecuteReader();
    while (cursor_transp.Read())
    {

```



```

        comboBox_transp.Items.Add(cursor_transp[1]);
        if (cursor_transp[0].ToString() ==
comboBox_transp.Tag.ToString())
            comboBox_transp.Text =
cursor_transp[1].ToString(); //назв-е исх компании
    }
    cursor_transp.Close();
    con.Close();
}

private void but_cancel_Click(object sender, EventArgs e)
{
    this.Close();
}

// внесение изменений в бд
private void but_done_Click(object sender, EventArgs e)
{
    // проверка на корректность данных
    if (comboBox_transp.Text == "" ||
textBox_mobile.Text == "")
    {
        MessageBox.Show("Необходимо указать все
параметры записи, помеченные звездочкой");
        return;
    }
    if (textBox_mobile.Text.Length != 11)
    {
        MessageBox.Show("Неправильно указан мобильный
номер водителя");
        return;
    }

    //поиск id трансп. компании
    long id_trans = 0;
    string forconnect = "...";
    OleDbConnection con = new
OleDbConnection(forconnect);
    con.Open();

    OleDbCommand com_trans = new OleDbCommand("select
idCompany, Name from TransCompany", con);
    OleDbDataReader cursor_trans =
com_trans.ExecuteReader();
    while (cursor_trans.Read())
        if (cursor_trans["Name"].ToString() ==
comboBox_transp.Text)
            id_trans = (long)cursor_trans["idCompany"];
    cursor_trans.Close();

    //преобразование моб номера

```

```

        string mobile = string.Format("{0:##-###-###-##-##}",
textBox_mobile.Text);

        //преобразование мб пустых необязательных данных
        object passport = null;
        if (textBox_passport.Text != "")
            passport = textBox_passport.Text;
        object capacity = null;
        if (textBox_capacity.Text != "")
            capacity = textBox_capacity.Text;

        if (label_carnum.Tag.ToString() == "-1") //если
добавляем новую запись
        {
            // проверка на корректность данных
            if (textBox_inn.Text == "")
            {
                MessageBox.Show("Необходимо указать все
параметры записи, помеченные звездочкой");
                return;
            }
            if (textBox_inn.Text.Length != 12)
            {
                MessageBox.Show("Неправильно указан ИНН
водителя");
                return;
            }
            if (textBox_carnum.Text.Length != 6)
            {
                MessageBox.Show("Неправильно указан номер
машины");
                con.Close();
                return;
            }

            OleDbCommand com_add_driver = new
OleDbCommand("insert into Driver (innDriver, Passport, PhoneNum)
values (?, ?, ?)", con);
            com_add_driver.Parameters.AddWithValue("@inn",
Convert.ToInt64(textBox_inn.Text));

            com_add_driver.Parameters.AddWithValue("@passport", passport);
            com_add_driver.Parameters.AddWithValue("@phone",
mobile);

            com_add_driver.ExecuteNonQuery();

            OleDbCommand com_add_crew = new
OleDbCommand("insert into Crew (CarNumber, Capacity, Driver,
TransCompany) values (?, ?, ?, ?)", con);
            com_add_crew.Parameters.AddWithValue("@carnum",
textBox_carnum.Text);
            com_add_crew.Parameters.AddWithValue("@capac",
capacity);

```

```

        com_add_crew.Parameters.AddWithValue("@driver",
Convert.ToInt64(textBox_inn.Text));
        com_add_crew.Parameters.AddWithValue("@transp",
id_trans);
        com_add_crew.ExecuteScalar();
    }
    else //если передадим старую
    {
        OleDbCommand com_update_driver = new
OleDbCommand("update Driver set Passport = ?, PhoneNum = ? " +
"where innDriver = ?", con);

com_update_driver.Parameters.AddWithValue("@passport",
passport);

com_update_driver.Parameters.AddWithValue("@phone", mobile);

com_update_driver.Parameters.AddWithValue("@inn",
Convert.ToInt64(label_driver_inn.Tag));
        com_update_driver.ExecuteScalar();

        OleDbCommand com_update_crew = new
OleDbCommand("update Crew set Capacity = ?, TransCompany = ? " +
"where CarNumber = ?", con);

com_update_crew.Parameters.AddWithValue("@capac", capacity);

com_update_crew.Parameters.AddWithValue("@transp", id_trans);

com_update_crew.Parameters.AddWithValue("@carnum",
label_carnum.Tag.ToString());
        com_update_crew.ExecuteScalar();
    }

    con.Close();
    this.Close();
}

// ограничение на ввод некорректных символов в поля с
числовыми данными
private void textBox_capacity_KeyPress(object sender,
KeyPressEventArgs e)
{
    //берем каждый введенный символ и пишем только те,
которые будут цифрами или кл удаления
    char sym = e.KeyChar;
    if (!Char.IsDigit(sym) && sym != 8)
        e.Handled = true;
}

```

```

        private void textBox_inn_KeyPress(object sender,
KeyPressEventArgs e)
        {
            char sym = e.KeyChar;
            if (!Char.IsDigit(sym) && sym != 8)
                e.Handled = true;
        }

        private void textBox_mobile_KeyPress(object sender,
KeyPressEventArgs e)
        {
            char sym = e.KeyChar;
            if (!Char.IsDigit(sym) && sym != 8)
                e.Handled = true;
        }
    }
}

```