БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники (ФИТР)

Кафедра программного обеспечения информационных систем и технологий

КУРСОВОЙ ПРОЕКТ

по дисциплине «Базы данных»

Тема: «Разработка приложения с базой данных: Информационная система Организация грузоперевозок »

|  |  |
| --- | --- |
| Исполнитель: | студент 3 курса группа 30701117  Друзик Алексей Николаевич |
| Руководитель проекта: | доцент Остапенко Инесса Витальевна |

Минск 2020

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и технологий

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

по дисциплине «Базы данных»

Тема: «Разработка приложения с базой данных: Информационная система Организация грузоперевозок »

|  |  |
| --- | --- |
| Исполнитель: | студент 3 курса группа 30701117  Друзик Алексей Николаевич |
| Руководитель проекта: | доцент Остапенко Инесса Витальевна |

Минск 2020

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 5](#_Toc9196595)

[**1 Проектирование базы данных** 6](#_Toc9196596)

[1.1 Построение концептуальной модели 6](#_Toc9196597)

[1.2 Построение логической модели 7](#_Toc9196598)

[1.3 Построение физической модели 12](#_Toc9196599)

[**2Разработка приложения для работы с базой данных** 13](#_Toc9196600)

[2.1 Назначение 13](#_Toc9196601)

[2.2 Структура приложения 13](#_Toc9196602)

[2.3 Описание приложения 14](#_Toc9196603)

[2.4 Тестирование приложения 16](#_Toc9196604)

[2.5 Руководство пользователя 17](#_Toc9196605)

[**ЗАКЛЮЧЕНИЕ** 22](#_Toc9196606)

[**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ** 23](#_Toc9196607)

[**ПРИЛОЖЕНИЕ** 24](#_Toc9196608)

# ВВЕДЕНИЕ

В настоящее время многие процессы нужно автоматизировать, поэтому создаются приложения, которые работают с базой данных. Благодаря таким приложениям работа многих диспетчеров становится проще, так как в один клик можно посмотреть данные из таблиц, добавить, редактировать или удалить данные из база данных.

Базаданных служит для упрощенного взиамодействия между различными данными, для их редактирования и просмотра. Это очень полезное обеспечение, особенно, когда говорят о самых различных информационных системах.

Говоря о теме данного проекта, мы не можем не использовать базы данных в нём, иначе приложение будет неэффективным и небезопасным. При проектировании баз данных по теме «Информационная система Организация грузоперевозок» в первую очередь нужно подумать про удобство использования и помощи для упрощения работы учета водителей.

В данном приложении должна быть спроектирована таблица журнала, где ведётся учет всех заказов и водителей, которые выполняют заказы. Такая таблица с соответствующей функциональностью значительно упрощает работу ведения учёта.

Таблицы и поля должны быть спроектированы таким образом, чтобы обеспечивалась целостность данных и их непротиворечивость.

# 1 Проектирование базы данных

Процесс проектирования включает в себя следующие этапы:

* концептуальное проектирование;
* логическое проектирование;
* физическое проектирование.

## 1.1 Построение концептуальной модели

Задачей концептуального проектирования является: выделение основных объектов предметной области, которые реализуются в виде сущностей, определение свойств сущностей и связей между ними.

Основные сущности с атрибутами:

* Водитель: уникальный идентификатор водителя, фамилия, имя, контактные данные.
* Услуга предоставления авто: уникальный идентификатор, марка, номерной знак.
* Место прибытия: уникальный идентификатор, название страны, название региона, название города.
* Клиент: уникальный идентификатор, фамилия, имя, контактные данные.
* Журнал заказов: уникальные идентификаторы заказа, города, услуги, водителя, клиента,дата выполения заказа, стоимость.

Для добавления записи в журнал заказов нужно будет указать дату выполнения заказа, стоимость, число мест в автомобиле, услугу, клиента, водителя, город. Затем, нужно сохранить изменения в базу данных. Также можно будет добавлять, редактировать и удалять во всех таблицах данные. Создавать отчёт о каждой поездке, а также можно фильтровать по времени и стоимости заказа. На рисунке 1 изображена ER-модель базы данных.

ER-модель - модель данных, позволяющая описывать концептуальные схемы предметной области. С её помощью можно выделить ключевые сущности и обозначить связи, которые могут устанавливаться между этими сущностями.

На ER-диаграмме(рисунок1) сущности обозначены прямоугольниками, а их атрибуты овалами.

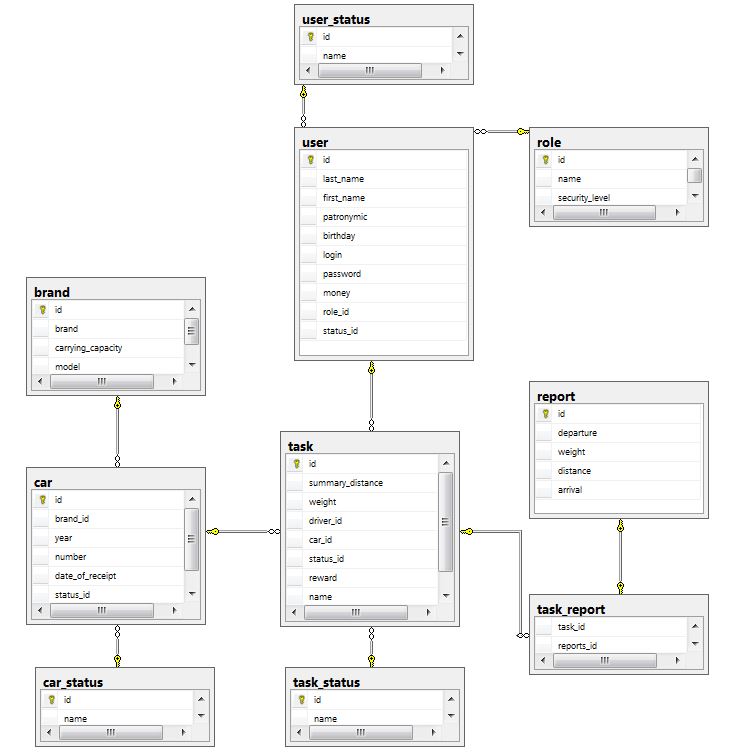


Рисунок 1 – ER-диаграмма базы данных

## 1.2 Построение логической модели

Логическое проектирование – это процесс конструирования информационной модели на основе существующих моделей данных, не зависимо от используемой СУБД и других условий физической реализации.

На таблице приведенной ниже логическая модель:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Таблица | Поле | Описание | Тип данных (длина) | Ограничения | Значение по умолч. | Допускается  NULL |
| brand | id | Идентификатор брэнда | int | PK, identity, autoincrement | Нет | Нет |
| brand | Брэнд | varchar(50) |  | Нет | Нет |
| carrying\_capacity | Грузоподъемность | float | FK, Ref service(id) | Нет | Нет |
| model | Модель | varchar(50) | FK, Ref voditel(id) | Нет | Нет |
| car | id | Идентификатор авто | int | PK, identity, autoincrement | Нет | Нет |
| brand\_id | Брэнд | int | FK, Ref brand(id) | Нет | Нет |
| year | Год | datetime | До 50 символов | Нет | Нет |
| number | Номер | varchar(50) | До 50 символов | Нет | Нет |
| date\_of\_receipt | Дата выпуска | datetime |  | Нет | Нет |
| status\_id | Статус | int | FK, Ref car\_status(id) | Нет | Нет |
| car\_status | id | Идентификатор статуса авто | int | PK, identity, autoincrement | Нет | Нет |
| name | Имя | varchar(50) | До 50 символов | Нет | Нет |
| user | id | Идентификатор водителя | int | PK, identity, autoincrement | Нет | Нет |
| last\_name | Фамилия | varchar(50) | До 50 символов | Нет | Нет |
| first\_name | Имя | varchar(50) | До 50 символов | Нет | Нет |
| patronymic | Отчество | varchar(50) | До 50 символов | Нет | Нет |
| birthday | День рождения | datetime | Нет | Нет | Нет |
| login | Логин | varchar(50) | До 50 символов | Нет | Нет |
| password | Пароль | varchar(50) | До 50 символов | Нет | Нет |
| money | Деньги | float |  | Нет | Нет |
| role\_id | Роль | int | FK, Ref role(id) | Нет | Нет |
| status\_id | Статус | int | FK, Ref user\_status(id) | Нет | Нет |
| user\_status | id | Идентификатор статуса юзера | int | PK, identity, autoincrement | Нет | Нет |
| name | Имя | varchar(50) | До 50 символов | Нет | Нет |
| role | id | Идентификатор роли | int | PK, identity, autoincrement | Нет | Нет |
| name | Имя | varchar(50) | До 50 символов | Нет | Нет |
| security\_level | Уровень | int | 1, 2, 3 | Нет | Нет |
| task | id | Идентификатор задачи | int | PK, identity, autoincrement | Нет | Нет |
| summary\_distance | Дисстанция | float |  | Нет | Нет |
| weight | Вес | float |  | Нет | Нет |
| driver\_id | Водитель | int | FK, Ref user (id) | Нет | Нет |
| car\_id | Авто | int | FK, Ref car(id) | Нет | Нет |
| status\_id | Статус | int | FK, Ref task\_status(id) | Нет | Нет |
| reward | Награда | float |  | Нет | Нет |
| name | Имя | varchar(50) | До 50 символов | Нет | Нет |
| task\_status | id | Идентификатор статуса задачи | int | PK, identity, autoincrement | Нет | Нет |
| name | Имя | varchar(50) | До 50 символов | Нет | Нет |
| report | id | Идентификатор отчета | int | PK, identity, autoincrement | Нет | Нет |
| departure | Отправление | datetime |  | Нет | Нет |
| weight | Вес | float |  | Нет | Нет |
| distance | Дисстанция | float |  | Нет | Нет |
| arrival | Прибытие | datetime |  | Нет | Нет |
| task\_report | task\_id | Задача | int | FK, Ref task(id) | Нет | Нет |
| report\_id | Отчет | int | FK, Ref report(id) | Нет | Нет |

Перечень представлений, хранимых процедур, функций пользователя и триггеров:

full\_user\_view – представление содержащее информацию о водителе:

**GO  
CREATE VIEW** full\_user\_view([id], [first\_name], [last\_name], [patronymic], [birthday], [login], [password], [money], [role\_id], [status\_id], [**name**]) **AS  
SELECT** u.id, u.first\_name, u.last\_name, u.patronymic, u.birthday, u.login, u.password, u.money, u.role\_id, u.status\_id, s.name **FROM**[carriages\_system].[dbo].[**user**] **AS** u  
**left join**user\_status **AS** s  
**ON** u.status\_id = s.id

carriages\_min\_max\_distance\_view – представление содержащее информацию о среднем километраже выездов каждого водителя за два года:

**GO  
CREATE VIEW** cariages\_min\_max\_distance\_view ([driverId], [distance])  
**AS SELECT** (driver1.last\_name + **' '** + driver1.first\_name + **' '** + driver1.patronymic), *AVG*(distance)  
**FROM** (**SELECT** *\** **FROM** task **as** t  
 **LEFT JOIN** (**SELECT** departure, distance, arrival, task\_id, reports\_id **FROM** report **AS** r  
 **LEFT JOIN** task\_report **AS** tr  
 **ON** tr.reports\_id = r.id) **AS** rtr  
 **ON** rtr.task\_id = t.id) **AS** journal1  
**INNER JOIN**[carriages\_system].[dbo].[**user**] **AS** driver1  
**ON** journal1.driver\_id = driver1.id  
**WHERE** getdate() > DATEPART(m, DATEADD(m, -24, getdate())) **GROUP BY** (driver1.last\_name + **' '** + driver1.first\_name + **' '** + driver1.patronymic);

carriages\_car\_count\_view – представление содержащее информацию об использовании автомобилей за два года:

**GO  
CREATE VIEW** cariages\_car\_count\_view ([**name**], **departureCount**)  
**AS  
SELECT** (b.brand + **' '** + b.model + **' '** + result2.number), *COUNT*(departure) **as** departureCount **FROM** [carriages\_system].[dbo].[Brand] **AS** b  
**INNER JOIN**(**SELECT** car1.brand\_id, car1.number, journal1.\*  
**FROM** (**SELECT** \* **FROM** task **as** t  
 **LEFT JOIN** (**SELECT** departure, distance, arrival, task\_id, reports\_id **FROM** report **AS** r  
 **LEFT JOIN** task\_report **AS** tr  
 **ON** tr.reports\_id = r.id) **AS** rtr  
 **ON** rtr.task\_id = t.id) **AS** journal1  
**LEFT JOIN**[carriages\_system].[dbo].[Car] **AS** car1  
**ON** journal1.car\_id = car1.id) **AS** result2  
**ON** result2.brand\_id = b.id  
**WHERE** getdate() > DATEPART(m, DATEADD(m, -24, getdate())) **GROUP BY** (b.brand + **' '** + b.model + **' '** + result2.number)  
  
 GetFreeTasks() - функция возвращает все свободные задачи:

**GO  
CREATE** FUNCTION GetFreeTasks()  
RETURNS **TABLE  
AS**RETURN  
**SELECT** *\****FROM** [task] **AS** t  
**WHERE** t.status\_id = 1

GetMineFinishedTasks() - функция возвращает все успешно завершенные задачи текущего водителя:

**GO  
CREATE** FUNCTION GetMineFinishedTasks(@driver\_id **int**)  
RETURNS **TABLE  
AS**RETURN  
**SELECT** *\****FROM** [task] **AS** t  
**WHERE** t.driver\_id = @driver\_id **AND** t.status\_id = 5

GetMineCurrentTask() - функция возвращает текущую задачу текущего водителя:**GO  
CREATE** FUNCTION GetMineCurrentTask(@driver\_id **int**)  
RETURNS **TABLE  
AS**RETURN  
**SELECT** *\****FROM** [task] **AS** t  
**WHERE** t.driver\_id = @driver\_id **AND** t.status\_id = 2

GetReportsByDrivarIdAndTaskId() - функция возвращает все отчеты для текущей задачи и текущего водителя:

**GO  
CREATE** FUNCTION GetReportsByDrivarIdAndTaskId(@driver\_id **int**, @task\_id **int**)  
RETURNS **TABLE  
AS**RETURN  
**SELECT** *\****FROM** [task\_report] **AS** tr  
**LEFT JOIN**[task] **as** t  
**ON** t.id = tr.task\_id  
**WHERE** t.driver\_id = @driver\_id **AND** t.id = @task\_id

GetReportsForActiveTask() - функция возвращает все отчеты для текущей активной задачи и текущего водителя:

**GO  
CREATE** FUNCTION getReportsForActiveTask(@driver\_id **int**)  
RETURNS **TABLE  
AS**RETURN  
**SELECT** *\** **FROM** report **WHERE** id **IN**(**SELECT** reports\_id **FROM** task\_report **where** task\_id = (**SELECT** top 1 id **FROM** task **WHERE** driver\_id = @driver\_id **AND** status\_id = 2))

## ADD\_REPORT - процедура добавления нового отчета:

**GO  
CREATE PROCEDURE** ADD\_REPORT  
(@current\_task\_id **int OUTPUT**, @current\_user\_id **int OUTPUT**, @departure DATETIME **OUTPUT**, @weight **int OUTPUT**, @distance **int OUTPUT**, @arrival DATETIME **OUTPUT**)  
**AS  
BEGIN  
DECLARE** @current\_report\_id **int**;  
**SET** @current\_report\_id = (**SELECT** IDENT\_CURRENT(**'report'**) + 1);  
**SET** IDENTITY\_INSERT report **ON**;  
**INSERT INTO** [dbo].[report] ([id],[departure],[weight],[distance], [arrival])  
 **VALUES** (@current\_report\_id, @departure, @weight, @distance, @arrival);  
**INSERT INTO** [dbo].[task\_report] ([task\_id],[reports\_id])  
**VALUES** (@current\_task\_id, @current\_report\_id);  
**SET** IDENTITY\_INSERT report OFF ;  
**SELECT** *\** **FROM** dbo.GetReportsForActiveTask(@current\_user\_id) **ORDER BY** id  
**END**

## TAKE\_TASK - процедура назначения задачи свободному водителю:

**GO  
CREATE PROCEDURE** TAKE\_TASK  
(@chosen\_task\_id **int OUTPUT**, @current\_user\_id **int OUTPUT**)  
**AS  
BEGIN  
DECLARE** @random\_free\_car\_id **int**;  
**SET** @random\_free\_car\_id = (**SELECT** TOP 1 id **FROM** car **WHERE** status\_id = 1);  
**UPDATE** task **SET** status\_id = 2, car\_id = @random\_free\_car\_id, driver\_id = @current\_user\_id **WHERE** id = @chosen\_task\_id;  
**UPDATE** [carriages\_system].[dbo].[**user**] **SET** status\_id = 2 **WHERE** id = @current\_user\_id;  
**SELECT** *\** **FROM** dbo.GetFreeTasks() **ORDER BY** id;  
**END**

VALIDATE \_TASK\_TO\_REJECTED - процедура отклонения задачи администратором:**GO  
CREATE PROCEDURE** VALIDATE\_TASK\_TO\_REJECTED  
(@chosen\_task\_id **int OUTPUT**)  
**AS  
BEGIN  
DECLARE** @current\_user\_id **int**;  
**SET** @current\_user\_id = (**SELECT** TOP 1 driver\_id **FROM** task **WHERE** id = @chosen\_task\_id);  
**UPDATE** task **SET** status\_id = 4 **WHERE** id = @chosen\_task\_id;  
**UPDATE** [carriages\_system].[dbo].[**user**] **SET** status\_id = 1 **WHERE** id = @current\_user\_id;  
**SELECT** *\** **FROM** task **ORDER BY** id;  
**END**

VALIDATE\_TASK\_TO\_FINISHED - процедура подтверждения завершения задачи администратором:**GO  
CREATE PROCEDURE** VALIDATE\_TASK\_TO\_FINISHED  
(@chosen\_task\_id **int OUTPUT**)  
**AS  
BEGIN  
DECLARE** @reward **float**;  
**DECLARE** @adminAmount **float**;  
**DECLARE** @driverAmount **float**;  
**DECLARE** @newAdminAmount **float**;  
**DECLARE** @newDriverAmount **float**;  
**DECLARE** @current\_user\_id **int**;  
**SET** @current\_user\_id = (**SELECT** TOP 1 driver\_id **FROM** task **WHERE** id = @chosen\_task\_id);  
**SET** @reward = (**SELECT** t.reward **FROM** task **as** t **WHERE** t.id = @chosen\_task\_id);  
**SET** @adminAmount = (**SELECT** u.money **FROM** [carriages\_system].[dbo].[**user**] **as** u **WHERE** u.id = 1);  
**SET** @driverAmount = (**SELECT** u.money **FROM** [carriages\_system].[dbo].[**user**] **as** u **WHERE** u.id = @current\_user\_id);  
**SET** @newAdminAmount = @adminAmount - @reward;  
**SET** @newDriverAmount = @driverAmount + @reward;  
**UPDATE** task **SET** status\_id = 5 **WHERE** id = @chosen\_task\_id;  
**UPDATE** [carriages\_system].[dbo].[**user**] **SET** money = @newAdminAmount **WHERE** id = 1;  
**UPDATE** [carriages\_system].[dbo].[**user**] **SET** status\_id = 1, money = @newDriverAmount **WHERE** id = @current\_user\_id;  
**SELECT** *\** **FROM** task **ORDER BY** id;  
**END**

audit\_report - триггер аудита изменения отчетов:

**GO  
CREATE** Trigger audit\_report **ON** [dbo].[report] AFTER  
**UPDATE**, **DELETE  
AS  
BEGIN  
DECLARE** @idOld **INT**, @idNew **INT  
DECLARE** @carIdOld **INT**, @carIdNew **INT  
DECLARE** @driverIdOld **INT**, @driverIdNew **INT  
DECLARE** @departureOld SMALLDATETIME, @departureNew SMALLDATETIME  
**DECLARE** @weightOld **DECIMAL**(18, 2), @weightNew **DECIMAL**(18, 2)  
**DECLARE** @distanceOld **DECIMAL**(18, 2), @distanceNew **DECIMAL**(18, 2)  
**DECLARE** @arrivalOld SMALLDATETIME, @arrivalNew SMALLDATETIME  
**SET** @idOld = (**select** [id] **from** deleted)  
**SET** @idNew = (**select** [id] **from** inserted)  
**SET** @departureOld = (**select** [departure] **from** deleted)  
**SET** @departureNew = (**select** [departure] **from** inserted)  
**SET** @weightOld = (**select** [weight] **from** deleted)  
**SET** @weightNew = (**select** [weight] **from** inserted)  
**SET** @distanceOld = (**select** [distance] **from** deleted)  
**SET** @distanceNew = (**select** [distance] **from** inserted)  
**SET** @arrivalOld = (**select** [arrival] **from** deleted)  
**SET** @arrivalNew = (**select** [arrival] **from** inserted);  
**INSERT INTO** [carriages\_system].[dbo].[audit\_report\_data] **VALUES**(@idOld, @idNew, GETDATE(), @departureOld, @departureNew,  
@weightOld, @weightNew, @distanceOld, @distanceNew, @arrivalOld, @arrivalNew)  
**END**check\_arrival - триггер для корректировки ввода даты:

**GO  
CREATE** TRIGGER check\_arrival **ON** [dbo].[report] AFTER **INSERT  
AS  
BEGIN  
DECLARE** @insertedId **INT  
DECLARE** @departure SMALLDATETIME  
**DECLARE** @arrival SMALLDATETIME  
**SET** @insertedId = (**SELECT** id **FROM** inserted)  
**SET** @departure = (**SELECT** departure **FROM** inserted)  
**SET** @arrival = (**SELECT** arrival **FROM** inserted)  
if @departure > @arrival  
**begin  
UPDATE** [dbo].[report]  
**SET** arrival = @departure **WHERE** id = @insertedId  
**end  
END**check\_departure - триггер для невозможности добавления отчета в день дальнобойщика - 08.31.2019:

**GO  
CREATE** TRIGGER check\_departure **ON** [dbo].[report]  
AFTER **INSERT**, **UPDATE  
AS  
BEGIN  
DECLARE** @departure SMALLDATETIME  
**SET** @departure = (**SELECT** departure **FROM** inserted)  
IF (8 = datepart(m, @departure) **AND** 31 = datepart(d, @departure))  
**BEGIN  
ROLLBACK TRANSACTION**PRINT **'День дальнобойщика, вы не можете задавать в БД запись на этот день!'  
END  
ELSE** PRINT **'Строка вставлена/изменена'  
END**

## 1.3 Построение физической модели

Физическое проектирование – это процедура создания описания конкретной реализации БД с описанием структуры хранения данных, методов доступа к данным.

Физическая модель обеспечивается SQL SERVER MANAGEMENT STUDIO по умолчанию: один вид файлов хранит саму базу данных, а другой вид файлов хранит все транзакции базы данных, количество и размеры данных файлов установлены по умолчанию.

Данные об файловой структуры базы данных отображены на рисунке 1.1 и рисунке 1.2.

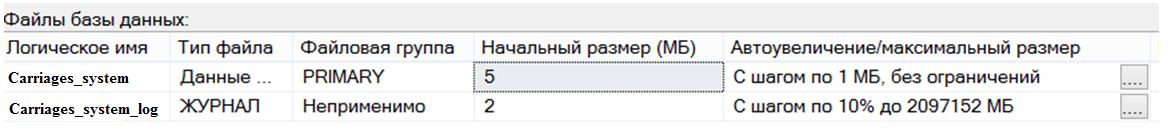
+

Рисунок 1.1 – Данные об файловой структуре

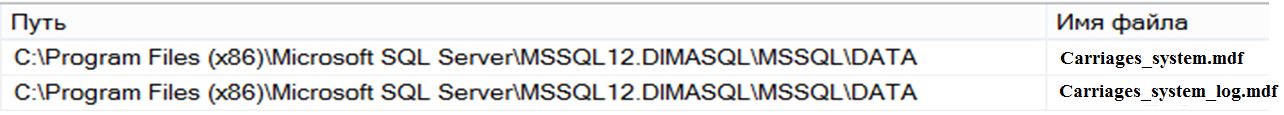


Рисунок 1.2 – Путь хранения базы данных

# Разработка приложения для работы с базой данных

## 2.1 Назначение

Разрабатываемое приложение будет служить для работы с базой данных «Автобаза – информационная система учета работы транспорта и водителей».

Приложение должно содержать функционал по добавлению, изменению, удалению информации об водителях, автомобилях, дате выполнения, городе, журнале ведения учёта заказов. Приложение должно максимально упростить взаимодействие пользователя и базы данных.

Предметная область представлена следующей информацией:

* водители;
* автомобили;
* задачи;
* ежедневные отчеты;
* статусы;
* журнал учёта.

## 2.2 Структура приложения

Приложение представляет собой автоматизированную информационную систему учета работы транспорта и водителей.

Приложение было разработано в виде приложения WindowsForms в среде Microsoft Visual Studio 2019.

В приложении для подключения и работы с базой данных используются встроенные инструменты MSVisualStudio:

* DataSet;
* DataView;
* DataGridView;
* SqlConnection;
* SqlCommand.

## 2.3 Описание приложения

Для хранения таблиц «brand», «car», «car\_status», «user», «user\_status», «role», «task», «task\_status», «report», «task\_report», а также для хранения представлений при необходимости был использован объект DataSet (рис. 2.1).

Для получения таблиц для DataSet был использован класс DataAdapter.

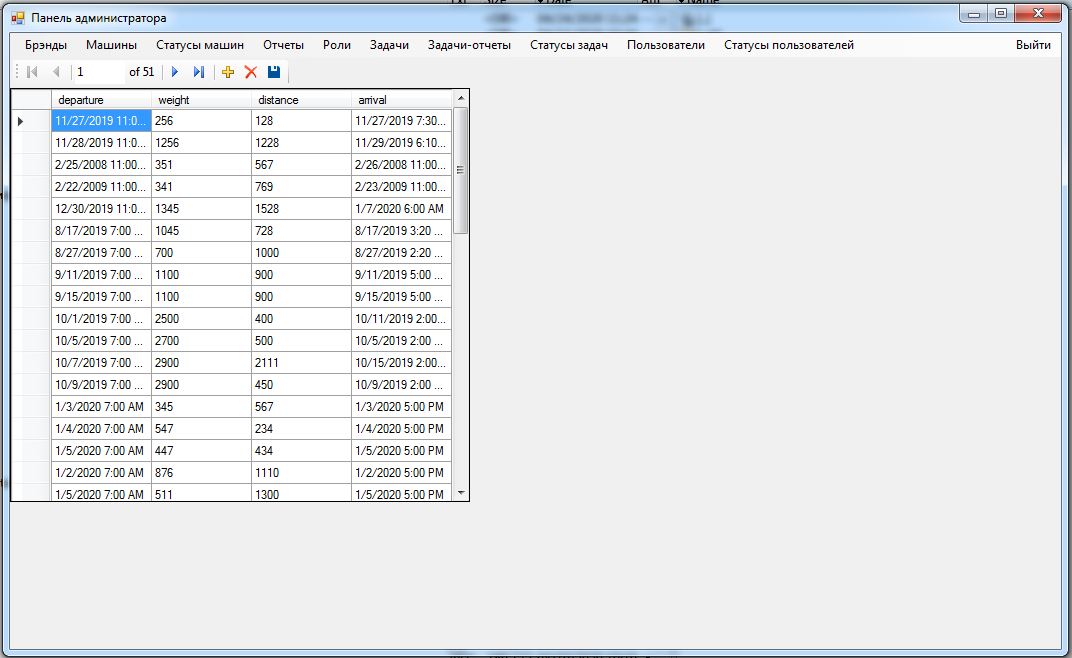


Рисунок 2.1 – использование DataSet

Для отдельного хранения каждой таблицы был использован класс DataTable, который содержит такие таблицы, как «Cities», «Clients», «Drivers», «Services», «Journal».

Код использования DataTable для таблицы «task»:

privatevoid Form1\_Load(object sender, EventArgs e)

{

this. reportTableAdapter.Fill(this.taskDataSet. report);

}

Для фильтрации данных в таблицах используется класс DataView. Он содержит свойство RowFilter, при присваивании значении которому, таблица, для которого был создан DataView будет отфильтрован согласно строке фильтрации.

Пример использования фильтрации для представления «Search»:

privatevoid button2\_Click(object sender, EventArgs e)

{

double start, end;

bool s = double.TryParse(textBox1.Text, out start);

if (!s) start = 1;

s = double.TryParse(textBox2.Text, out end);

if (!s) end = 100;

this.reportTableAdapter.FillByCost(this.taskDataSet.task, report, end);

}

Для отображения содержимого базы данных в приложении используется компонент DataGridView. **Этот компонент применяется ко всем таблицам приложения** «brand», «car», «car\_status», «user», «user\_status», «role», «task», «task\_status», «report», «task\_report», а также представлениям.

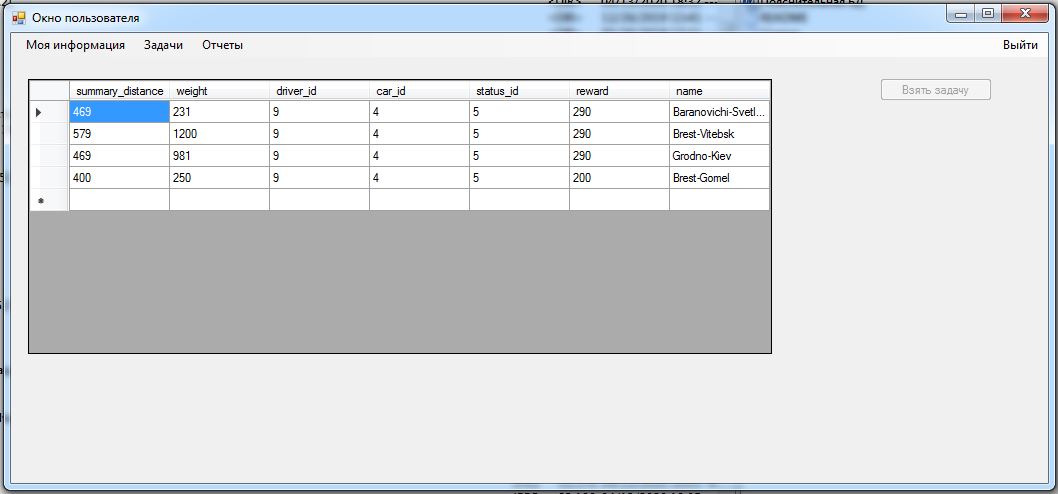


Рисунок 2.2 – применение DataGridViewк таблице «task»

Для работы с полями, содержащими дату и время использовался компонент dateTimePicker. Был применен для поля DateTimeв формах для работы с таблицами «Searchreport».

privatevoid button1\_Click(object sender, EventArgs e)

{

string start = "" + dateTimePicker1.Value;

string end = "" + dateTimePicker2.Value;

this.reportTableAdapter.FillByDate(this.journalDataSet. report, start.Substring(0, 10) , end.Substring(0, 10));

}

Для оформления отчета и создания справки в приложении использовался компонент ReportViewer. С помощью этого компонента были созданы формы «TaskReport», « DaylyTaskReport », «UserReport ». Для отображения отчета на этом компоненте был создан элемент «Report1».Также был офрмлен отчёт с параметром, в качестве параметра выступает дата, водитель.

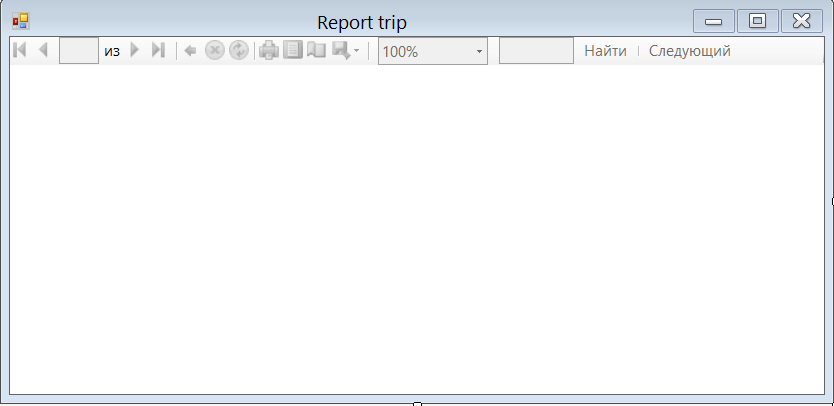


Рисунок 2.3 – форма отчета о ежедневном рейсе

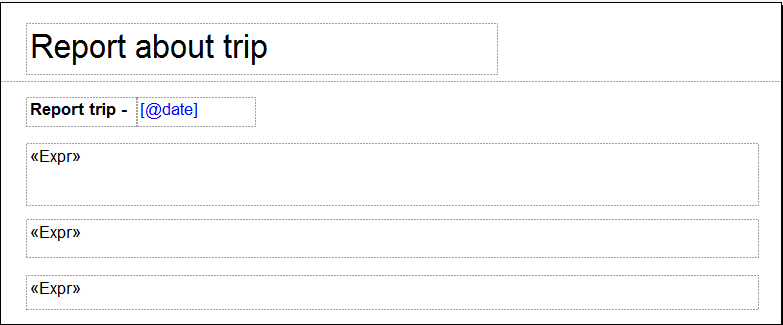


Рисунок 2.4 – эскиз отчета

## 2.4 Тестирование приложения

Тестирование созданного программного продукта будем производить на следующей конфигурации компьютера: Intel® Core™ i7-6720@ 2.60 GHz, 16.00 Гб RAM, ОС Windows 7. Для того, чтобы приложение запустилось на пк должно быть установлено:

* MSVisualStudio 2019;
* SQL Server 2014;
* SQL Management Studio 2014.

В процессе создания программы были устранены серьезные ошибки, носящие важное влияние на работу приложения. В конечном продукте явных ошибок обнаружено не было.

## 2.5 Руководство пользователя

Программа написана на языке C# в среде MS Visual Studio 2019. Функционал реализован в соответствии с требованиями к проекту.

После запуска приложения пользователь попадает на главное окно входа в систему (рис. 2.5). Есть два вида пользователей - администратор и пользователь.

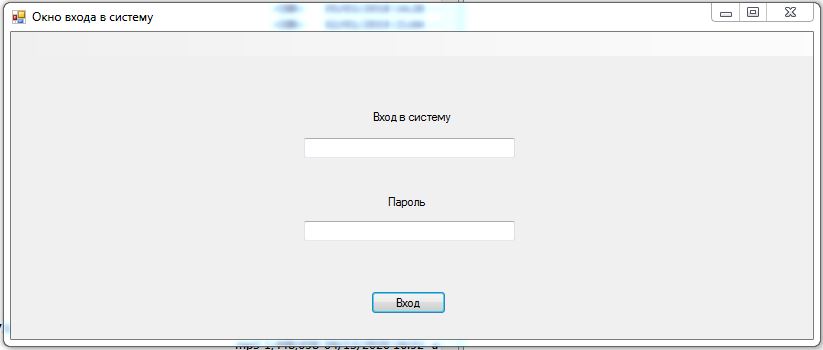


Рисунок 2.5 – главное окно

С главного окна пользователя можно просматривать свою текущую информацию, информацию о суммарном километраже других водителей, суммарном количестве выезда машин, зарегестрированных в системе, а так же перейти на вкладку с задачами и отчетами (рис. 2.6).

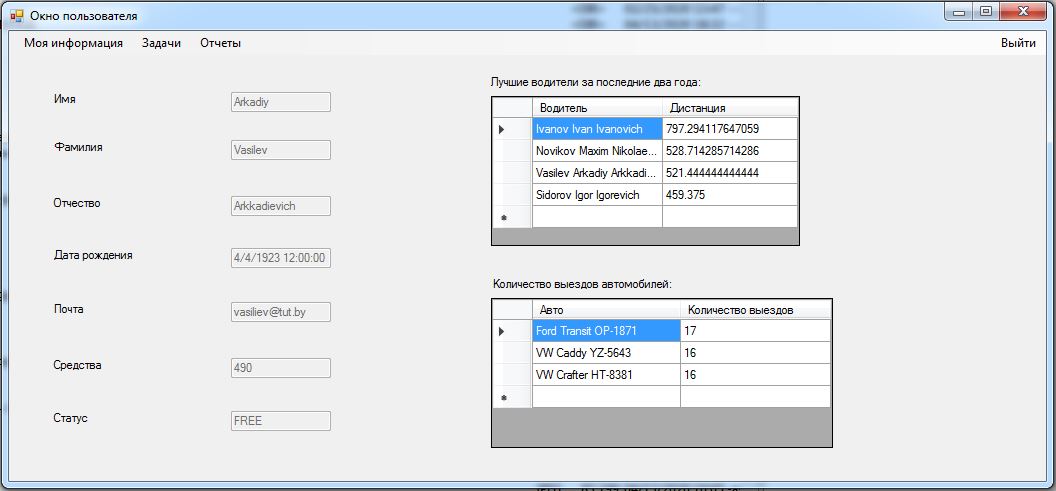


Рисунок 2.6 – начальное окно со стороны пользователя

Чтобы перейти на окно с задачми пользователь кликнуть на вкладку “задачи” и выбрать необходимую опцию. Доступны подвкладки - ''Все задачи'', ''Текущая задача'', ''Свободные задачи'' (рис. 2.7). Водитель в этом окне может просматривать свои успешно завершенные задачи и текущую задачу. В случае если у водителя нету задачи, то он может выбрать любую доступную, кликнув по ней в таблице, при этом кнопка ''Взять задачу'' станет активной. После того, как пользователь выбрал задачу, в главном окне меняется его статус на “Занят”.

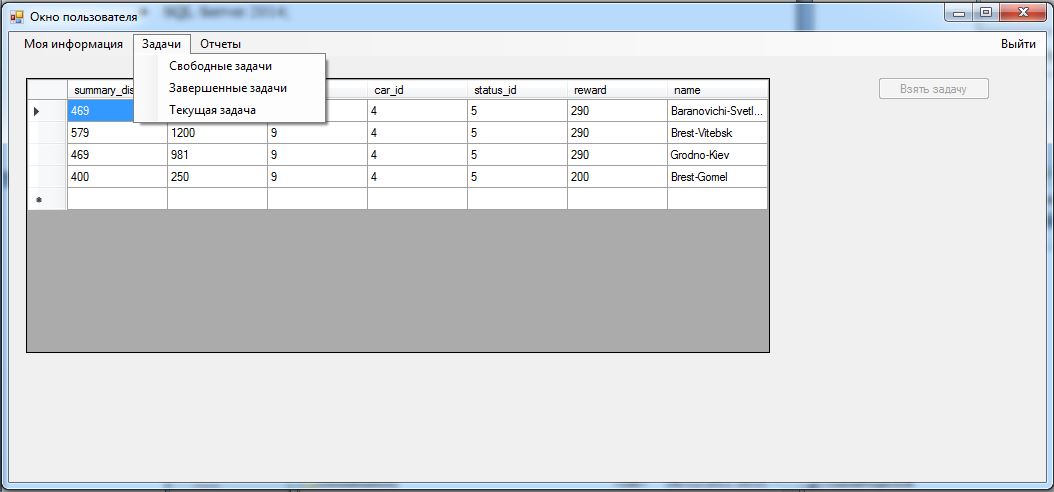


Рисунок 2.7 – окно с задачами со стороны пользователя

Кликнув по вкладке “Отчеты” пользователь попадает в окно с отчетами (рис. 2.8). В случае, если у пользователя есть активная задача, то этом окне пользователь может видеть и добавлять ежедневные отчеты. При добавлении отчета, все поля должны быть заполнены, иначе пользователь увидит сообщение. Когда добавлена, как минимум одна задача, кнопка “Завершить задачу” становится активной и пользователь может завершить задачу кликнув по ней.

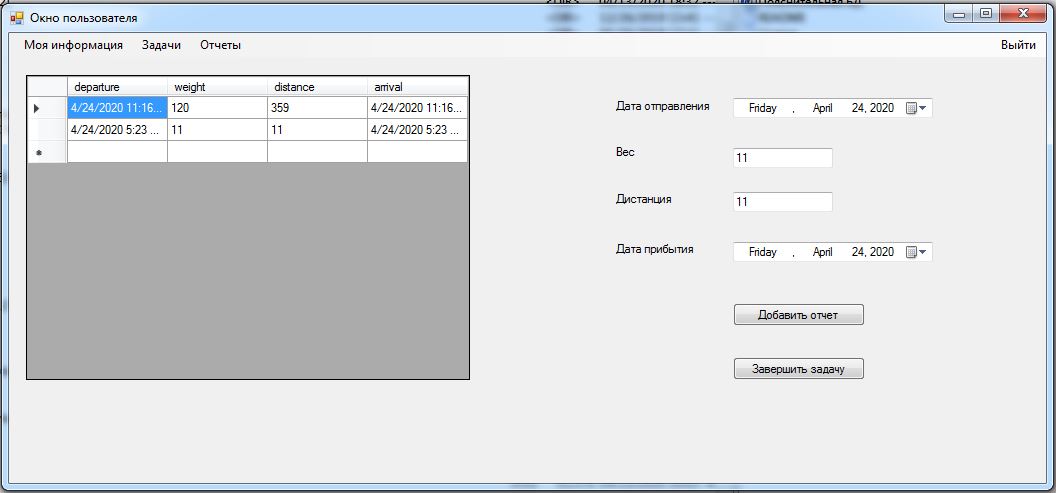


Рисунок 2.8 – окно просмотра и добавления отчетов

С окна администратора можно работать с каждой таблицей БД - "Брэнды", "Машины", "Статусы машин", "Отчеты", "Роли", "Задачи", "Статусы задач", "Пользователи", "Статусы пользователей". Предусмотрена возможность со стороны администратора просматривать записи, добавлять новые записи, редактировать существующие записи, удалять записи. После входа в систему в качестве администратора, пользователь попадает на вкладку с брэндами (рис. 2.9).

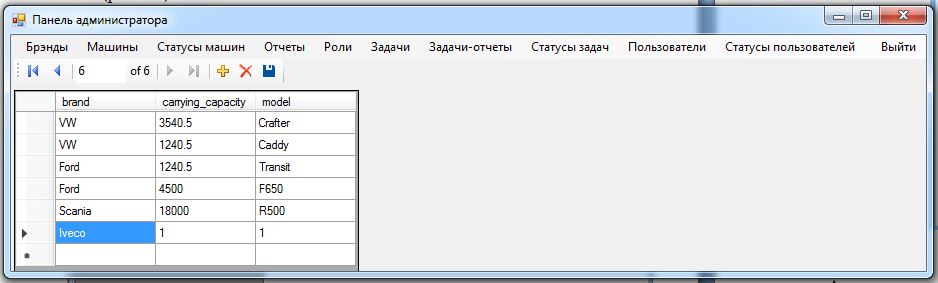


Рисунок 2.9 – начальное окно со стороны администратора

Кликнув по вкладке "Машины" админ попадает в окно с машинами (рис. 2.10).

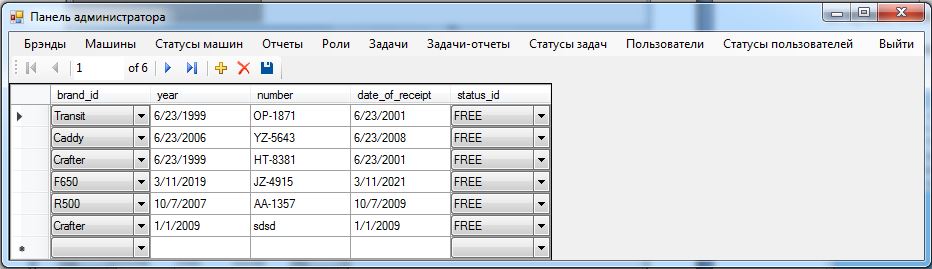


Рисунок 2.10 – окно работы с таблицей "Машины" со стороны администратора

Кликнув по вкладке "Статусы машин" админ попадает в окно со статусами (рис. 2.11).

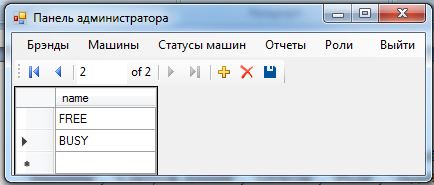


Рисунок 2.11 – окно работы с таблицей "Статусы машин" со стороны админа

Кликнув по вкладке "Отчеты" админ попадает в окно с отчетами (рис. 2.12).

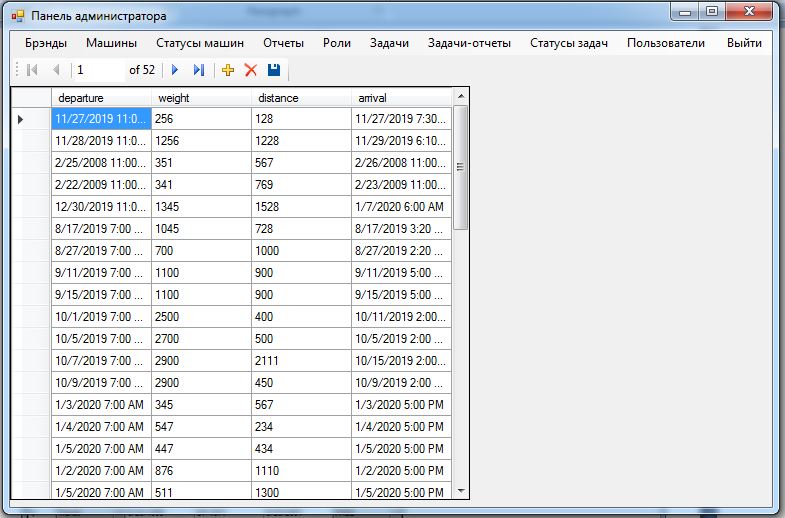


Рисунок 2.12 – окно работы с таблицей "Отчеты" со стороны админа

Кликнув по вкладке "Роли" админ попадает в окно со статусами (рис. 2.13).

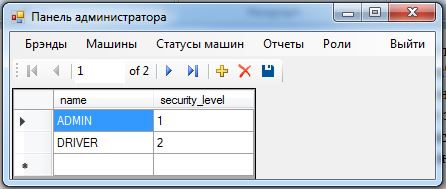


Рисунок 2.13 – окно работы с таблицей "Роли" со стороны админа

Кликнув по вкладке "Задачи" админ попадает в окно со статусами (рис. 2.14). Вся основная работа администратора происходит именно в этой вкдадке. В этой вкладке администратор может валидировать завершенные со стороны водителя задачи. Возможно два варианта - откланить или подтвердить. В случае отклонения задачи, статус водителя поменяется на "Свободен", статус задачи поменяется на "Отклонено" и ему не будут переведены средства. В случае подтверждения задачи, статус водителя поменяется на "Завершен", статус задачи поменяется на "Завершено" и ему будут переведены средства.

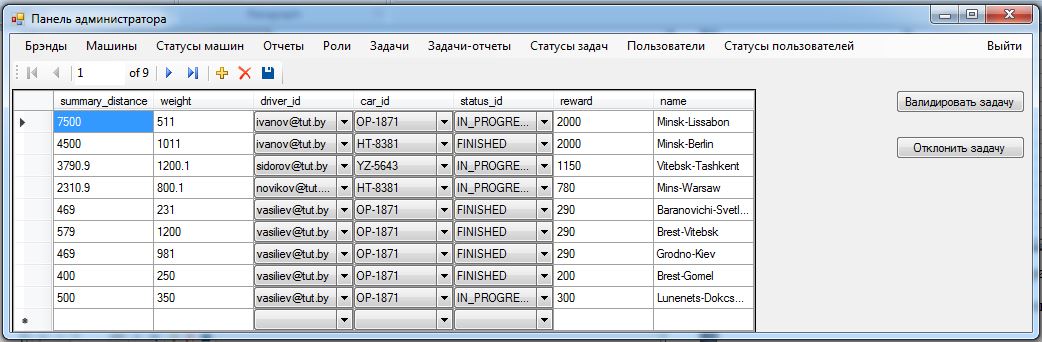


Рисунок 2.14 – окно работы с таблицей "Задачи" со стороны админа

После выбора задачи для валидации и нажатии на кнопку "Валидировать задачу", появлется сообщение об выполнении успешной операции. (рис. 2.15).

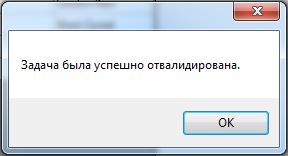


Рисунок 2.15 – окно работы с таблицей "Статусы задач" со стороны админа

Кликнув по вкладке "Статусы задач" админ попадает в окно со статусами (рис. 2.16).

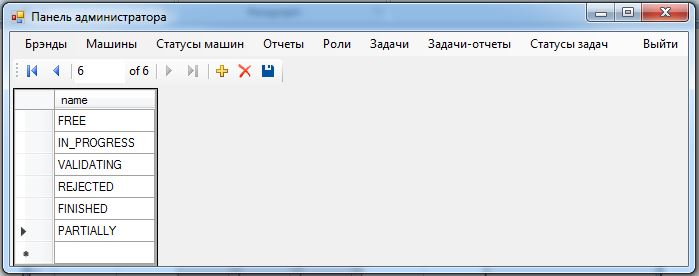


Рисунок 2.16 – окно работы с таблицей "Статусы задач" со стороны админа

Кликнув по вкладке "Пользователи" админ попадает в окно со статусами (рис. 2.17).

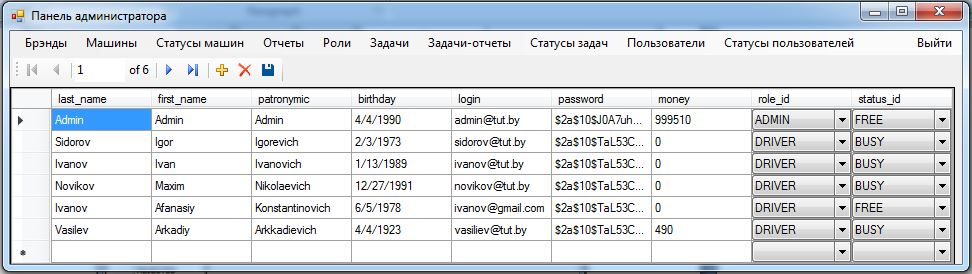


Рисунок 2.17 – окно работы с таблицей "Пользователи" со стороны админа

Кликнув по вкладке "Статусы задач" админ попадает в окно со статусами (рис. 2.18).

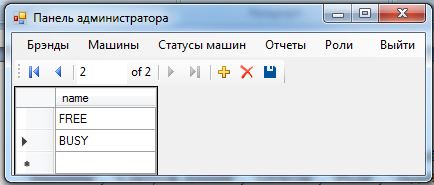


Рисунок 2.18 – окно работы с таблицей "Статусы задач" со стороны админа

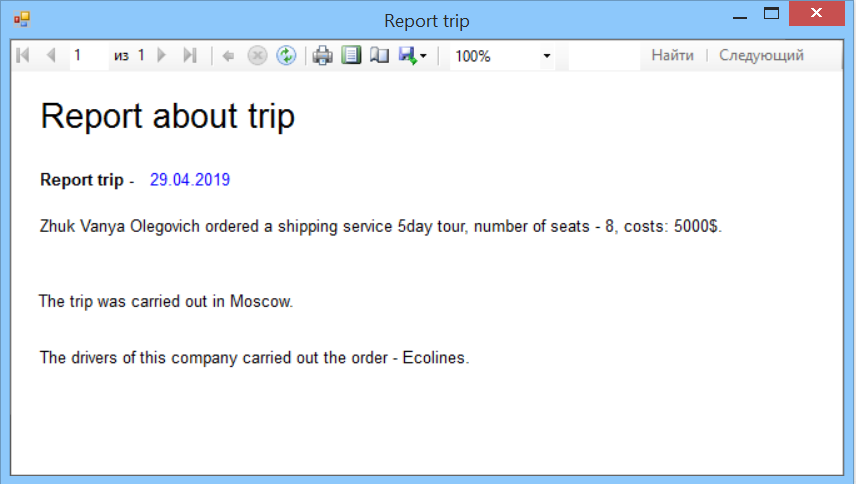


Рисунок 2.10 – окно отчёта с параметром

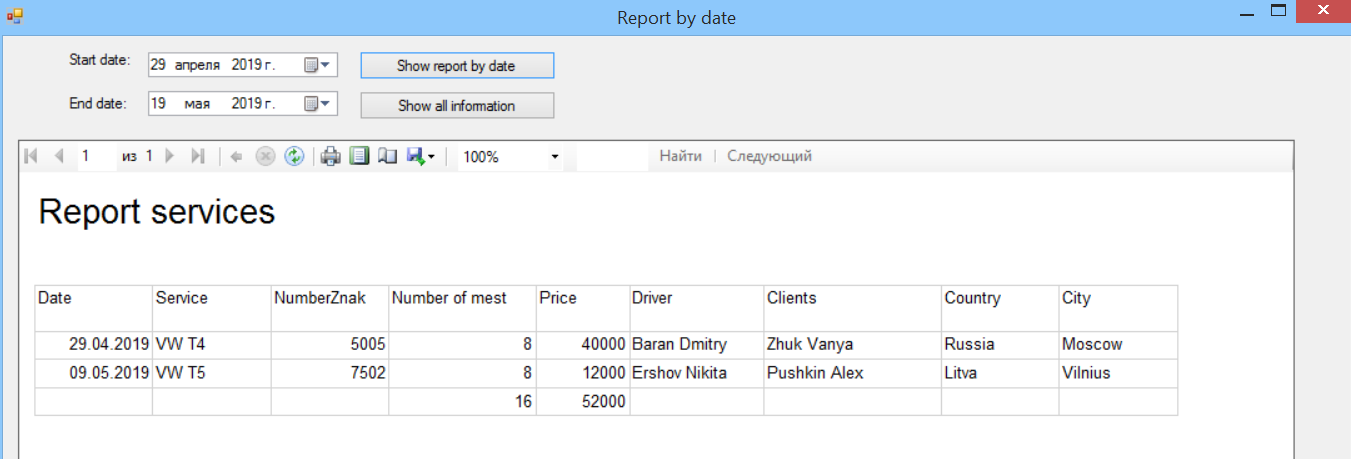


Рисунок 2.11 – окно отчёта

Нажав на главной форме на кнопку “Searchbycost” открывается окно поиска по двум параметрам:

* по цене;
* по периоду времени.

Выбрав период даты, можно увидеть журнал в пределах выбранных дат (рис. 2.12)

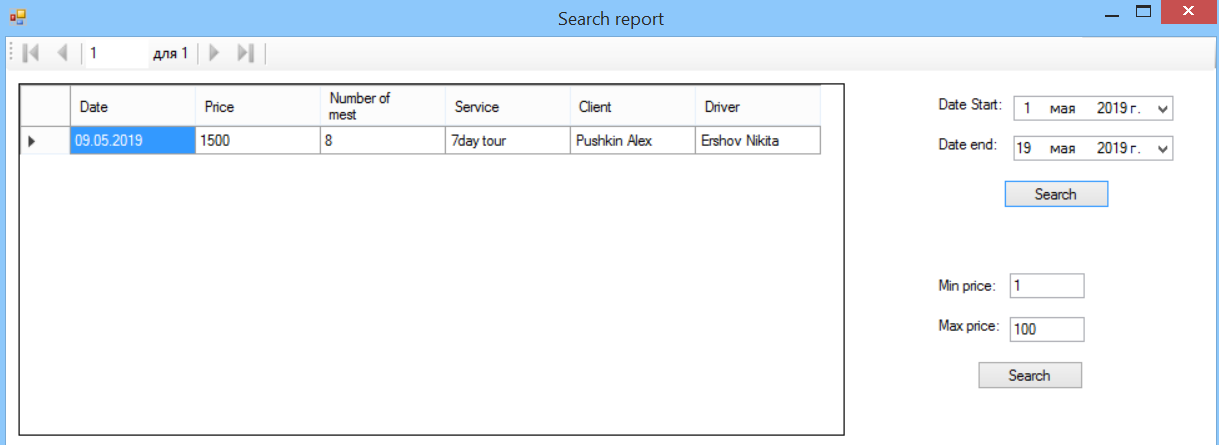


Рисунок 2.12 – окно журнала за выбранный период

Выбрав минимальную и максимульную стоимость услуги, можно увидеть журнал в пределах выбранных значений (рис. 2.13)

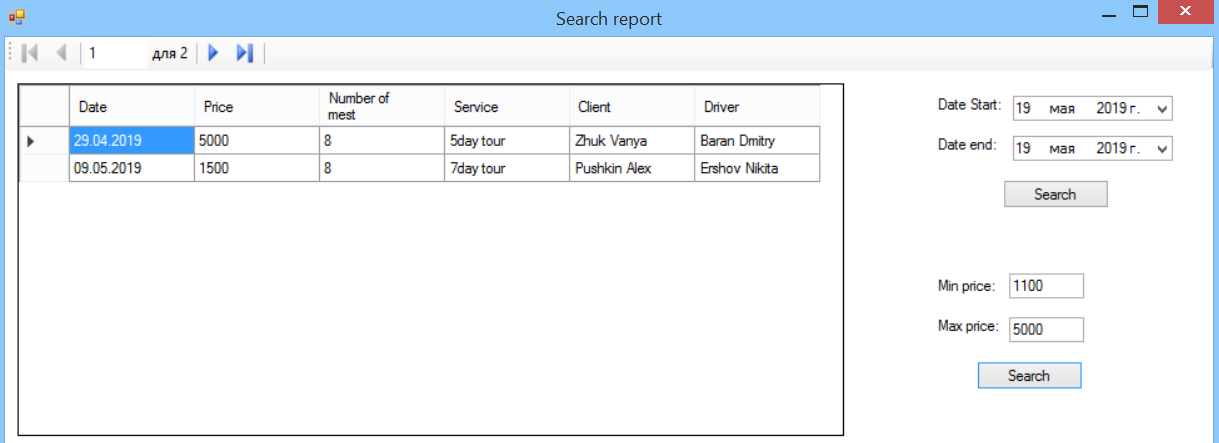


Рисунок 2.13 – окно журнала в предалах ввёдных значений стоимости

Для закрытия приложения нужно нажать на крестик в правом верхнем углу главного окна. После закрытия приложения все изменённые данные будут сохранены в базу данных.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения данной курсовой работы была спроектирована база данных и разработано приложение для работы с этой базой данных. Исходя из цели, было разработано приложение отображения информации о водителе, услуге предоставляемой, клиентах, городах прибытия и для редактирования журнала учёта водителей.

Разработанное приложение можно использовать на производствах, логистических и туристических фирмах. Благодаря простоте использования это приложение значительно упрощает и ускоряет работу по ведению учёта водителей.

Приложение имеет пользовательский интерфейс, разработанный посредством использования средств, предоставляемых MicrosoftVisualStudio. В связи с этим, данных продукт способен работать на любых машинах, поддерживающих операционную систему Windows и не обладающих сверхмощной производительностью.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. «Понимание SQL» — Мартин Грубер. – Яз. Русск. – Москва, 1993.
2. «Изучаем SQL» — Алан Бьюли. – Яз. Русск.
3. Wikipedia [Электронный ресурс]. – Wikipedia, 20.03.2019. –https://ru.wikipedia.org/wiki/SQL-Server, - Загл. С экрана. – Яз. Русск.
4. Sql-language[Электронный ресурс]. -sql-language, 17.05.2018 –**Язык запросов SQL**https://sql-language.ru/query-select.html, свободный, - Загл. С экрана. – Яз. Русск.
5. Metanit [Электронный ресурс]. – Metanit.com, 19.05.2019 - <https://metanit.com/sharp/general.php>, - Загл. С экрана. – Яз. Русск.

# ПРИЛОЖЕНИЕ

Листинг программы:

Форма входа в систему:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CourseProjectDB

{

public partial class Main : Form

{

const int ADMIN\_ROLE\_ID = 1;

const int USER\_ROLE\_ID = 2;

public Main()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

Connection connection = new Connection();

String login = textBox1.Text;

String query = "select \* from [carriages\_system].[dbo].[FullUserView] where login = '" + login + "'";

SqlDataAdapter sqlDataAdapter = connection.getConnection(query);

DataTable dataTable = new DataTable();

sqlDataAdapter.Fill(dataTable);

int i = dataTable.Select().Length;

if (i == 1)

{

DataRow row = dataTable.Rows[0];

String email = row["login"].ToString();

String password = row["password"].ToString();

if (textBox1.Text == email && textBox2.Text == password)

{

int role\_id = Int32.Parse(row["role\_id"].ToString());

if (role\_id == USER\_ROLE\_ID)

{

UserMainForm userForm = new UserMainForm(this, dataTable);

userForm.Show();

this.Hide();

}

else

{

AdminMainForm userForm = new AdminMainForm(this, dataTable);

userForm.Show();

this.Hide();

}

}

else

{

MessageBox.Show("Неправильный логин или пароль");

}

}

else

{

MessageBox.Show("Неправильный логин или пароль");

}

}

}

}

Форма пользователя:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CourseProjectDB

{

public partial class UserMainForm : Form

{

Connection connection = new Connection();

Main main = null;

DataTable currentUserDataTable = null;

DataTable currentTasksTable = null;

DataTable freeTasksDataTable = null;

bool userIsActive = false;

public UserMainForm(Main main, DataTable dataTable)

{

InitializeComponent();

this.main = main;

this.currentUserDataTable = dataTable;

DataRow row = dataTable.Rows[0];

textBox11.Text = row["first\_name"].ToString();

textBox12.Text = row["last\_name"].ToString();

textBox13.Text = row["patronymic"].ToString();

textBox14.Text = row["birthday"].ToString();

textBox15.Text = row["login"].ToString();

textBox16.Text = row["money"].ToString();

textBox17.Text = row["name"].ToString();

panel1.BringToFront();

panel2.SendToBack();

int userId = (int)row["id"];

SqlDataAdapter sqlDataAdapter = this.connection.getConnection("SELECT \* from GetMineCurrentTasks(" + userId + ")");

DataTable currentTasksTable = new DataTable();

sqlDataAdapter.Fill(currentTasksTable);

this.currentTasksTable = currentTasksTable;

sqlDataAdapter = this.connection.getConnection("SELECT \* FROM cariages\_min\_max\_distance\_view ORDER BY distance DESC;");

DataTable maxDistanceTable = new DataTable();

sqlDataAdapter.Fill(maxDistanceTable);

dataGridView3.Columns.Add("newColumnName", "Водитель");

dataGridView3.Columns.Add("newColumnName", "Дистанция");

DataGridViewColumn column1 = dataGridView3.Columns[0];

DataGridViewColumn column2 = dataGridView3.Columns[1];

column1.Width = 130;

column2.Width = 135;

for (int i = 0; i < maxDistanceTable.Rows.Count; i++)

{

dataGridView3.Rows.Add();

dataGridView3.Rows[i].Cells[0].Value = maxDistanceTable.Rows[i]["driverId"].ToString();

dataGridView3.Rows[i].Cells[1].Value = maxDistanceTable.Rows[i]["distance"].ToString();

}

sqlDataAdapter = this.connection.getConnection("SELECT \* FROM cariages\_car\_count\_view;");

DataTable maxDeparturesTable = new DataTable();

sqlDataAdapter.Fill(maxDeparturesTable);

dataGridView4.Columns.Add("newColumnName", "Авто");

dataGridView4.Columns.Add("newColumnName", "Количество выездов");

DataGridViewColumn column3 = dataGridView4.Columns[0];

DataGridViewColumn column4 = dataGridView4.Columns[1];

column3.Width = 148;

column4.Width = 150;

for (int i = 0; i < maxDeparturesTable.Rows.Count; i++)

{

dataGridView4.Rows.Add();

dataGridView4.Rows[i].Cells[0].Value = maxDeparturesTable.Rows[i]["name"].ToString();

dataGridView4.Rows[i].Cells[1].Value = maxDeparturesTable.Rows[i]["departureCount"].ToString();

}

if (currentTasksTable.Rows.Count == 1)

{

this.userIsActive = true;

button2.Enabled = false;

}

}

private void UserForm\_Load(object sender, EventArgs e)

{

panel1.BringToFront();

panel1.Location = new Point(0, 30);

panel2.SendToBack();

panel3.SendToBack();

panel1.Visible = true;

panel3.Visible = false;

panel2.Visible = false;

}

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

{

main.Show();

this.Hide();

}

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

{

}

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

{

SqlDataAdapter sqlDataAdapter = this.connection.getConnection("SELECT \* from GetFreeTasks()");

DataTable freeTasksDataTable = new DataTable();

sqlDataAdapter.Fill(freeTasksDataTable);

dataGridView1.DataSource = freeTasksDataTable;

dataGridView1.Columns[0].Visible = false;

this.freeTasksDataTable = freeTasksDataTable;

panel1.SendToBack();

panel2.BringToFront();

panel2.Location = new Point(0, 30);

panel3.SendToBack();

panel1.Visible = false;

panel2.Visible = true;

panel3.Visible = false;

}

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

{

SqlDataAdapter sqlDataAdapter = this.connection.getConnection("Select \* from car");

DataTable dataTable = new DataTable();

sqlDataAdapter.Fill(dataTable);

panel1.BringToFront();

panel1.Location = new Point(0, 30);

panel2.SendToBack();

panel3.SendToBack();

panel1.Visible = true;

panel3.Visible = false;

panel2.Visible = false;

//dataGridView1.DataSource = dataTable;

}

private void label1\_Click(object sender, EventArgs e)

{

}

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

{

DataRow row = this.currentUserDataTable.Rows[0];

int userId = (int) row["id"];

SqlDataAdapter sqlDataAdapter = this.connection.getConnection("SELECT \* from GetMineFinishedTasks(" + userId + ")");

DataTable dataTable = new DataTable();

sqlDataAdapter.Fill(dataTable);

dataGridView1.DataSource = dataTable;

dataGridView1.Columns[0].Visible = false;

panel1.SendToBack();

panel2.BringToFront();

panel2.Location = new Point(0, 30);

panel3.SendToBack();

panel1.Visible = false;

panel2.Visible = true;

panel3.Visible = false;

}

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

{

DataRow row = this.currentUserDataTable.Rows[0];

int userId = (int)row["id"];

SqlDataAdapter sqlDataAdapter = this.connection.getConnection("SELECT \* from GetReportsForActiveTask(" + userId + ")");

DataTable dataTable = new DataTable();

sqlDataAdapter.Fill(dataTable);

dataGridView2.DataSource = dataTable;

dataGridView2.Columns[0].Visible = false;

SqlDataAdapter sqlDataAdapter2 = this.connection.getConnection("SELECT \* from GetMineCurrentTasks(" + userId + ")");

DataTable currentTasksTable = new DataTable();

sqlDataAdapter2.Fill(currentTasksTable);

this.currentTasksTable = currentTasksTable;

panel1.SendToBack();

panel2.SendToBack();

panel3.BringToFront();

panel3.Location = new Point(0, 30);

panel1.Visible = false;

panel2.Visible = false;

panel3.Visible = true;

if (dataTable.Rows.Count == 0)

{

button3.Enabled = false;

}

}

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

{

panel1.SendToBack();

panel2.SendToBack();

panel3.BringToFront();

panel3.Location = new Point(0, 30);

panel1.Visible = false;

panel2.Visible = false;

panel3.Visible = true;

}

private void текущаяЗадачаToolStripMenuItem1\_Click(object sender, EventArgs e)

{

DataRow row = this.currentUserDataTable.Rows[0];

int userId = (int)row["id"];

SqlDataAdapter sqlDataAdapter = this.connection.getConnection("SELECT \* from GetMineCurrentTasks(" + userId + ")");

DataTable dataTable = new DataTable();

sqlDataAdapter.Fill(dataTable);

dataGridView1.DataSource = dataTable;

dataGridView1.Columns[0].Visible = false;

panel1.SendToBack();

panel2.BringToFront();

panel2.Location = new Point(0, 30);

panel3.SendToBack();

panel1.Visible = false;

panel2.Visible = true;

panel3.Visible = false;

}

private void label1\_Click\_1(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

int taskId = default;

int weight;

int distance;

DateTime departure;

DateTime arrival;

if (this.currentTasksTable.Rows.Count == 1)

{

DataRow currentTasksTableRow = this.currentTasksTable.Rows[0];

taskId = (int)currentTasksTableRow["id"];

if (textBox41.Text != "" && textBox42.Text != "")

{

DataRow userRow = currentUserDataTable.Rows[0];

int userId = (int)userRow["id"];

weight = Int32.Parse(textBox41.Text);

distance = Int32.Parse(textBox42.Text);

departure = dateTimePicker1.Value;

arrival = dateTimePicker1.Value;

DataTable dt = new DataTable();

SqlConnection myConn = new SqlConnection("Data Source=ADRUZIK-PC\\SQLEXPRESS;Initial Catalog=carriages\_system;Integrated Security=True;User ID=root;Password=root;");

myConn.Open();

SqlCommand myCmd = new SqlCommand("ADD\_REPORT", myConn);

myCmd.CommandType = CommandType.StoredProcedure;

SqlParameter param1 = new SqlParameter("@current\_task\_id", taskId);

SqlParameter param2 = new SqlParameter("@current\_user\_id", userId);

SqlParameter param3 = new SqlParameter("@departure", departure);

SqlParameter param4 = new SqlParameter("@weight", weight);

SqlParameter param5 = new SqlParameter("@distance", distance);

SqlParameter param6 = new SqlParameter("@arrival", arrival);

myCmd.Parameters.Add(param1);

myCmd.Parameters.Add(param2);

myCmd.Parameters.Add(param3);

myCmd.Parameters.Add(param4);

myCmd.Parameters.Add(param5);

myCmd.Parameters.Add(param6);

SqlDataAdapter da = new SqlDataAdapter(myCmd);

da.Fill(dt);

dataGridView2.DataSource = dt;

button3.Enabled = true;

MessageBox.Show("Отчет был успешно добавлен.");

}

else

{

MessageBox.Show("Заполните все поля!");

}

} else

{

MessageBox.Show("У пользователя нету активных задач!");

}

}

private void button2\_Click(object sender, EventArgs e)

{

int rowIndex = dataGridView1.CurrentCell.RowIndex;

int numberOfSelectedRows = dataGridView1.SelectedRows.Count;

if (numberOfSelectedRows == 1)

{

DataRow freeTaskRow = this.freeTasksDataTable.Rows[0];

int taskId = (int)freeTaskRow["id"];

DataRow userRow = this.currentUserDataTable.Rows[0];

int userId = (int)userRow["id"];

DataTable dt = new DataTable();

SqlConnection myConn = new SqlConnection("Data Source=ADRUZIK-PC\\SQLEXPRESS;Initial Catalog=carriages\_system;Integrated Security=True;User ID=root;Password=root;");

myConn.Open();

SqlCommand myCmd = new SqlCommand("TAKE\_TASK", myConn);

myCmd.CommandType = CommandType.StoredProcedure;

SqlParameter param1 = new SqlParameter("@chosen\_task\_id", taskId);

SqlParameter param2 = new SqlParameter("@current\_user\_id", userId);

myCmd.Parameters.Add(param1);

myCmd.Parameters.Add(param2);

SqlDataAdapter da = new SqlDataAdapter(myCmd);

da.Fill(dt);

dataGridView1.DataSource = dt;

String query = "select \* from [carriages\_system].[dbo].[FullUserView] where id = " + userId;

SqlDataAdapter sqlDataAdapter = this.connection.getConnection(query);

DataTable currentUserDataTable = new DataTable();

sqlDataAdapter.Fill(currentUserDataTable);

this.currentUserDataTable = currentUserDataTable;

DataRow row = this.currentUserDataTable.Rows[0];

textBox17.Text = row["name"].ToString();

MessageBox.Show("Задача успешно выбрана для выполнения!");

button2.Enabled = false;

} else

{

MessageBox.Show("Выберите задачу!");

}

}

private void button3\_Click(object sender, EventArgs e)

{

if (this.currentTasksTable.Rows.Count == 1)

{

DataRow currentTasksTableRow = this.currentTasksTable.Rows[0];

int taskId = (int)currentTasksTableRow["id"];

DataTable dt = new DataTable();

SqlConnection myConn = new SqlConnection("Data Source=ADRUZIK-PC\\SQLEXPRESS;Initial Catalog=carriages\_system;Integrated Security=True;User ID=root;Password=root;");

myConn.Open();

SqlCommand myCmd = new SqlCommand("SEND\_TO\_VALIDATION\_TASK", myConn);

myCmd.CommandType = CommandType.StoredProcedure;

SqlParameter param1 = new SqlParameter("@chosen\_task\_id", taskId);

myCmd.Parameters.Add(param1);

SqlDataAdapter da = new SqlDataAdapter(myCmd);

da.Fill(dt);

dataGridView1.DataSource = dt;

button1.Enabled = false;

button3.Enabled = false;

MessageBox.Show("Задача отправлена на валидацию администратором!");

} else

{

MessageBox.Show("У вас нету активной задачи или нету ни одного отчета!");

}

}

}

}

Форма администратора:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CourseProjectDB

{

public partial class AdminMainForm : Form

{

Main main = null;

DataTable dataTable = null;

public AdminMainForm(Main main, DataTable dataTable)

{

InitializeComponent();

this.main = main;

this.dataTable = dataTable;

}

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

{

main.Show();

this.Hide();

}

private void AdminMainForm\_Load(object sender, EventArgs e)

{

// This line of code loads data into the 'carriages\_systemDataSet.FullUserView' table. You can move, or remove it, as needed.

//this.fullUserViewTableAdapter.Fill(this.carriages\_systemDataSet.FullUserView);

// This line of code loads data into the 'carriages\_systemDataSet.user\_status' table. You can move, or remove it, as needed.

this.user\_statusTableAdapter.Fill(this.carriages\_systemDataSet.user\_status);

// This line of code loads data into the 'carriages\_systemDataSet.user' table. You can move, or remove it, as needed.

this.userTableAdapter.Fill(this.carriages\_systemDataSet.user);

// This line of code loads data into the 'carriages\_systemDataSet.user' table. You can move, or remove it, as needed.

this.userTableAdapter.Fill(this.carriages\_systemDataSet.user);

// This line of code loads data into the 'carriages\_systemDataSet.task\_status' table. You can move, or remove it, as needed.

this.task\_statusTableAdapter.Fill(this.carriages\_systemDataSet.task\_status);

// This line of code loads data into the 'carriages\_systemDataSet.task\_report' table. You can move, or remove it, as needed.

this.task\_reportTableAdapter.Fill(this.carriages\_systemDataSet.task\_report);

// This line of code loads data into the 'carriages\_systemDataSet.task' table. You can move, or remove it, as needed.

this.taskTableAdapter.Fill(this.carriages\_systemDataSet.task);

// This line of code loads data into the 'carriages\_systemDataSet.role' table. You can move, or remove it, as needed.

this.roleTableAdapter.Fill(this.carriages\_systemDataSet.role);

// This line of code loads data into the 'carriages\_systemDataSet.report' table. You can move, or remove it, as needed.

this.reportTableAdapter.Fill(this.carriages\_systemDataSet.report);

// This line of code loads data into the 'carriages\_systemDataSet.car\_status' table. You can move, or remove it, as needed.

this.car\_statusTableAdapter.Fill(this.carriages\_systemDataSet.car\_status);

// This line of code loads data into the 'carriages\_systemDataSet.car' table. You can move, or remove it, as needed.

this.carTableAdapter.Fill(this.carriages\_systemDataSet.car);

// This line of code loads data into the 'carriages\_systemDataSet.brand' table. You can move, or remove it, as needed.

this.brandTableAdapter.Fill(this.carriages\_systemDataSet.brand);

panel1.BringToFront();

panel2.SendToBack();

panel3.SendToBack();

panel4.SendToBack();

panel5.SendToBack();

panel6.SendToBack();

panel7.SendToBack();

panel8.SendToBack();

panel9.SendToBack();

panel10.SendToBack();

panel11.SendToBack();

}

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

{

try

{

this.Validate();

this.brandBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.carriages\_systemDataSet);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void toolStripButton7\_Click(object sender, EventArgs e)

{

try

{

this.Validate();

this.carBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.carriages\_systemDataSet);

}

catch (System.Data.SqlClient.SqlException ex)

{

MessageBox.Show(ex.Message);

}

}

private void toolStripButton14\_Click(object sender, EventArgs e)

{

try

{

this.Validate();

this.carstatusBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.carriages\_systemDataSet);

}

catch (System.Data.SqlClient.SqlException ex)

{

MessageBox.Show(ex.Message);

}

}

private void toolStripButton21\_Click(object sender, EventArgs e)

{

try

{

this.Validate();

this.reportBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.carriages\_systemDataSet);

}

catch (System.Data.SqlClient.SqlException ex)

{

MessageBox.Show(ex.Message);

}

}

private void toolStripButton28\_Click(object sender, EventArgs e)

{

try

{

this.Validate();

this.roleBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.carriages\_systemDataSet);

}

catch (System.Data.SqlClient.SqlException ex)

{

MessageBox.Show(ex.Message);

}

}

private void toolStripButton35\_Click(object sender, EventArgs e)

{

try

{

this.Validate();

this.taskBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.carriages\_systemDataSet);

}

catch (System.Data.SqlClient.SqlException ex)

{

MessageBox.Show(ex.Message);

}

}

private void toolStripButton42\_Click(object sender, EventArgs e)

{

try

{

this.Validate();

this.taskreportBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.carriages\_systemDataSet);

}

catch (System.Data.SqlClient.SqlException ex)

{

MessageBox.Show(ex.Message);

}

}

private void toolStripButton49\_Click(object sender, EventArgs e)

{

try

{

this.Validate();

this.taskstatusBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.carriages\_systemDataSet);

}

catch (System.Data.SqlClient.SqlException ex)

{

MessageBox.Show(ex.Message);

}

}

private void toolStripButton56\_Click(object sender, EventArgs e)

{

try

{

this.Validate();

this.userBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.carriages\_systemDataSet);

}

catch (System.Data.SqlClient.SqlException ex)

{

MessageBox.Show(ex.Message);

}

}

private void toolStripButton63\_Click(object sender, EventArgs e)

{

try

{

this.Validate();

this.userstatusBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.carriages\_systemDataSet);

}

catch (System.Data.SqlClient.SqlException ex)

{

MessageBox.Show(ex.Message);

}

}

private void toolStripButton70\_Click(object sender, EventArgs e)

{

try

{

this.Validate();

this.fullUserViewBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.carriages\_systemDataSet);

}

catch (System.Data.SqlClient.SqlException ex)

{

MessageBox.Show(ex.Message);

}

}

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

{

panel1.BringToFront();

panel2.SendToBack();

panel3.SendToBack();

panel4.SendToBack();

panel5.SendToBack();

panel6.SendToBack();

panel7.SendToBack();

panel8.SendToBack();

panel9.SendToBack();

panel10.SendToBack();

panel11.SendToBack();

}

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

{

panel1.SendToBack();

panel2.BringToFront();

panel3.SendToBack();

panel4.SendToBack();

panel5.SendToBack();

panel6.SendToBack();

panel7.SendToBack();

panel8.SendToBack();

panel9.SendToBack();

panel10.SendToBack();

panel11.SendToBack();

}

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

{

panel1.SendToBack();

panel2.SendToBack();

panel3.BringToFront();

panel4.SendToBack();

panel5.SendToBack();

panel6.SendToBack();

panel7.SendToBack();

panel8.SendToBack();

panel9.SendToBack();

panel10.SendToBack();

panel11.SendToBack();

}

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

{

panel1.SendToBack();

panel2.SendToBack();

panel3.SendToBack();

panel4.BringToFront();

panel5.SendToBack();

panel6.SendToBack();

panel7.SendToBack();

panel8.SendToBack();

panel9.SendToBack();

panel10.SendToBack();

panel11.SendToBack();

}

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

{

panel1.SendToBack();

panel2.SendToBack();

panel3.SendToBack();

panel4.SendToBack();

panel5.SendToBack();

panel6.SendToBack();

panel7.SendToBack();

panel8.BringToFront();

panel9.SendToBack();

panel10.SendToBack();

panel11.SendToBack();

}

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

{

panel1.SendToBack();

panel2.SendToBack();

panel3.SendToBack();

panel4.SendToBack();

panel5.SendToBack();

panel6.SendToBack();

panel7.BringToFront();

panel8.SendToBack();

panel9.SendToBack();

panel10.SendToBack();

panel11.SendToBack();

}

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

{

panel1.SendToBack();

panel2.SendToBack();

panel3.SendToBack();

panel4.SendToBack();

panel5.SendToBack();

panel6.BringToFront();

panel7.SendToBack();

panel8.SendToBack();

panel9.SendToBack();

panel10.SendToBack();

panel11.SendToBack();

}

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

{

panel1.SendToBack();

panel2.SendToBack();

panel3.SendToBack();

panel4.SendToBack();

panel5.BringToFront();

panel6.SendToBack();

panel7.SendToBack();

panel8.SendToBack();

panel9.SendToBack();

panel10.SendToBack();

panel11.SendToBack();

}

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

{

panel1.SendToBack();

panel2.SendToBack();

panel3.SendToBack();

panel4.SendToBack();

panel5.SendToBack();

panel6.SendToBack();

panel7.SendToBack();

panel8.SendToBack();

panel9.SendToBack();

panel10.BringToFront();

panel11.SendToBack();

}

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

{

panel1.SendToBack();

panel2.SendToBack();

panel3.SendToBack();

panel4.SendToBack();

panel5.SendToBack();

panel6.SendToBack();

panel7.SendToBack();

panel8.SendToBack();

panel9.BringToFront();

panel10.SendToBack();

panel11.SendToBack();

}

private void button1\_Click(object sender, EventArgs e)

{

int rowIndex = dataGridView5.CurrentCell.RowIndex;

if (rowIndex != 0)

{

int selectedrowindex = dataGridView5.SelectedCells[0].RowIndex;

DataGridViewRow selectedRow = dataGridView5.Rows[selectedrowindex];

int taskId = (int)selectedRow.Cells[1].Value;

int taskStatus = (int)selectedRow.Cells[5].Value;

if (taskStatus == 3)

{

DataTable dt = new DataTable();

SqlConnection myConn = new SqlConnection("Data Source=ADRUZIK-PC\\SQLEXPRESS;Initial Catalog=carriages\_system;Integrated Security=True;User ID=root;Password=root;");

myConn.Open();

SqlCommand myCmd = new SqlCommand("VALIDATE\_TASK\_TO\_FINISHED", myConn);

myCmd.CommandType = CommandType.StoredProcedure;

SqlParameter param1 = new SqlParameter("@chosen\_task\_id", taskId);

myCmd.Parameters.Add(param1);

SqlDataAdapter da = new SqlDataAdapter(myCmd);

da.Fill(dt);

dataGridView5.DataSource = dt;

this.userTableAdapter.Fill(this.carriages\_systemDataSet.user);

try

{

this.Validate();

this.fullUserViewBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.carriages\_systemDataSet);

}

catch (System.Data.SqlClient.SqlException ex)

{

MessageBox.Show(ex.Message);

}

MessageBox.Show("Задача была успешно отвалидирована.");

}

else

{

MessageBox.Show("Выберите задычу, со статусом 'VALIDATING'!");

}

}

else

{

MessageBox.Show("Выберите задачу!");

}

}

private void button2\_Click(object sender, EventArgs e)

{

int numberOfSelectedRows = dataGridView5.SelectedRows.Count;

if (numberOfSelectedRows == 1)

{

int selectedrowindex = dataGridView5.SelectedCells[0].RowIndex;

DataGridViewRow selectedRow = dataGridView5.Rows[selectedrowindex];

int taskId = (int) selectedRow.Cells[1].Value;

int taskStatus = (int) selectedRow.Cells[5].Value;

if (taskStatus == 3)

{

DataTable dt = new DataTable();

SqlConnection myConn = new SqlConnection("Data Source=ADRUZIK-PC\\SQLEXPRESS;Initial Catalog=carriages\_system;Integrated Security=True;User ID=root;Password=root;");

myConn.Open();

SqlCommand myCmd = new SqlCommand("VALIDATE\_TASK\_TO\_REJECTED", myConn);

myCmd.CommandType = CommandType.StoredProcedure;

SqlParameter param1 = new SqlParameter("@chosen\_task\_id", taskId);

myCmd.Parameters.Add(param1);

SqlDataAdapter da = new SqlDataAdapter(myCmd);

da.Fill(dt);

dataGridView5.DataSource = dt;

MessageBox.Show("Задача успешно отклонена.");

} else

{

MessageBox.Show("Выберите задычу, со статусом 'VALIDATING'!");

}

}

else

{

MessageBox.Show("Выберите задачу!");

}

}

}

}