



*«Московский государственный технический университет  
имени Н.Э. Баумана»*

*(МГТУ им. Н.Э. Баумана)*

---

**ФАКУЛЬТЕТ** Информатика и системы управления

**КАФЕДРА** Программное обеспечение ЭВМ и информационные технологии

## **РАСЧЕТНО - ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту на тему:**

Информационная система контроля перемещения транспортных средств с опасными грузами в случае чрезвычайного происшествия.

Студент \_\_\_\_\_ ( Пахомов А. А. )  
(ф.и.о.)

Руководитель курсового проекта \_\_\_\_\_ ( Волкова Л. Л. )  
(ф.и.о.)

Москва, 2014

МГТУ им. Н.Э. Баумана

# Реферат

РПЗ 30с., 17 рис., 4 источника

Объектом исследования является анализ опасности, связанной с перевозкой опасных грузов в определённом месте в определённое время.

Цель работы – ускорение устранения последствий аварий, угрожающих загрязнением окружающей среды.

В процессе работы разработан программный комплекс, состоящий из базы данных, карты и приложения пользовательского уровня.

Работа с базой данных предполагает выборку данных для предоставления пользователю.

Работа с картой предполагает прокладывание маршрутов для водителя.

## Оглавление

Введение .....	5
1 Аналитический раздел.....	6
1.1 Описание предметной области .....	6
1.2 Требования к ПО.....	7
1.3 Определение бизнес-правил.....	8
1.4 Определение сущностей, связей между сущностями и атрибутов сущностей.....	8
1.5 ER-диаграмма .....	9
2 Конструкторская часть.....	10
2.1. Обоснование выбора СУБД .....	10
2.2 Определение типов данных для столбцов таблиц базы данных Course_DB .....	10
2.3 Ограничения целостности .....	10
2.3.1 Первичные ключи .....	10
2.3.2 Внешние ключи.....	10
2.4 Диаграмма базы данных.....	11
2.5 Алгоритм прокладывания маршрутов.....	11
3 Технологическая часть.....	14
3.1 Обоснование выбора языка и среды разработки .....	14
3.2 Взаимодействие с базой данных .....	14
3.3 Запросы к базе данных .....	15
3.4 Технологические особенности проекта .....	15
3.4.1. Класс IMap .....	17
3.4.2. Класс IDataHandler.....	18
3.5 Пользовательский интерфейс .....	23
4 Руководство пользователя .....	26
4.1 Анализ опасности .....	26
4.2 Заполнение путевого листа .....	26
4.3 Добавление .....	27

4.4 Удаление .....	28
4.4 Поиск информации .....	28
Заключение.....	29
Список литературы.....	30

## **Введение**

При чрезвычайном происшествии необходимо скорейшее реагирование специальных служб. При участии в аварии автомобиля, перевозящего опасный груз, в зависимости от типа этого груза, может быть необходимо немедленное принятие радикальных мер.

Поэтому существует задача контроля перемещения транспортных средств с опасными грузами для оперативного определения местоположения водителя в определённый момент времени.

На данный момент не существуют реализации подобной системы, поэтому разработка является актуальной.

# 1 Аналитический раздел

## 1.1 Описание предметной области

Каждый водитель, перевозящий опасные грузы, перед отправкой обязан заполнять путевой лист. В нём указывается контакт водителя, перевозимый груз и посещаемые населённые пункты (как минимум начальный и конечный).

На основании этих данных и данных, полученных посредством анализа карты, можно составить оценки времени, которое потребуется водителю для пересечения каждого посещаемого населённого пункта по пути из начального в конечный. Рассчитанную информацию можно хранить в базе данных и использовать для анализа степени опасности происшествия.

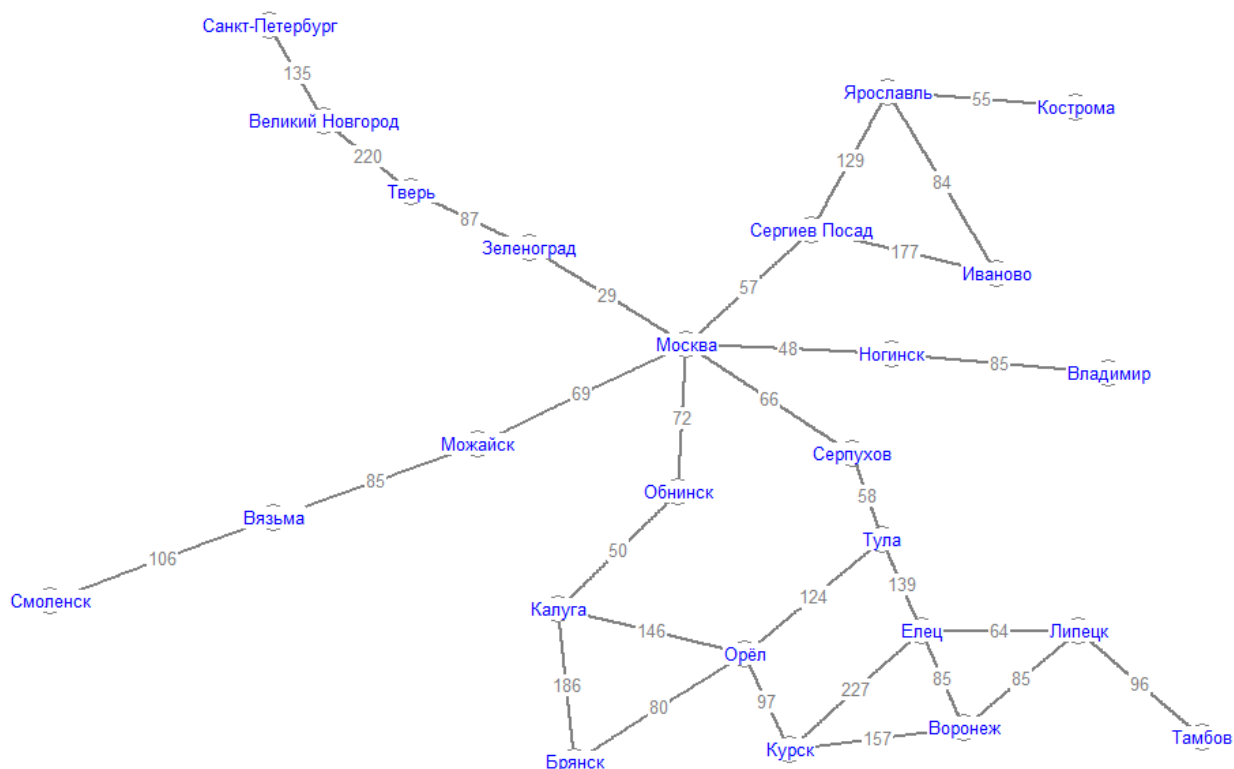
При поступлении информации об аварии: времени и месте, используя хранимые данные можно быстро определить, какие транспортные средства могли находиться вблизи заданной точки, и предоставить данные о возможной утечке опасного вещества, категории чрезвычайного происшествия и рекомендации по устранению опасности.

В данном курсовом проекте используются следующие классы опасности грузов (табл. 1.1), соответствующие ГОСТ 12.1.007-76:

Таблица 1.1.  
Классы опасности.

Класс опасности	Степень опасности
I	чрезвычайно опасные вещества
II	высокоопасные вещества
III	умеренно опасные вещества
IV	малоопасные вещества

В качестве карты использовался граф (рис. 1.1). Вес каждого ребра показывает среднее время в минутах, необходимое для преодоления расстояния между городами.



## 1.2 Требования к ПО

По результатам анализа предметной области можно определить следующий необходимый функционал приложения:

- предоставление формы для заполнения путевых листов;
- определение приблизительного времени нахождения водителя в каждом городе;
- определение местоположения водителя в любое указанное время в рамках зарегистрированной на это время перевозки (в том числе текущее местоположение) по данным базы;
- удаление завершённых перевозок и перевозок, зарегистрированных ранее указанного времени;
- возможность изменения данных в базе;
- расчёт маршрутов.

Пользователями базы данных являются операторы МЧС. При регистрации новой перевозки на водителя его предыдущие перевозки не удаляются автоматически ввиду возможности их последующей надобности. При удалении водителя удаляется вся информация о его перевозках.

Основной задачей базы данных является хранение оценок времени, на основании которых производится определение местонахождения водителя. Каждый водитель перед отправлением подаёт путевой лист, на основании которого производится расчёт и заполнение базы данными о новой перевозке. Основными типами информации в базе данных являются текст и время.

### 1.3 Определение бизнес-правил

Чтобы определить структуру базы данных, контролировать и влиять на операции с ней, необходимо определить бизнес-правила.

1. Каждый водитель должен иметь имя, фамилию и номер мобильного телефона, эти поля являются обязательными.
2. Водитель может не иметь отчества.
3. На водителя может быть не зарегистрирована ни одна перевозка.
4. Груз должен иметь название, уровень опасности и описание действий при аварии.
5. Количество посещаемых водителем городов равно как минимум двум.
6. Водитель может в рамках одной перевозки посетить один и тот же город несколько раз.
7. Маршрут может быть «специфическим» (например, из Воронежа в Москву через Курск).
8. Множество перевозок может быть зарегистрировано на одного водителя.
9. Несколько водителей не могут иметь один и тот же номер телефона.
10. Водители могут быть полными тёзками.
11. Водители не могут обмениваться номерами телефонов.

### 1.4 Определение сущностей, связей между сущностями и атрибутов сущностей

Для определения количества таблиц в базе данных обратимся к результатам изучения предметной области.

Создадим таблицу Transits как базовую таблицу о каждой перевозке, содержащую основной ID водителя и груза. Для определения параметров перевозки (время, города), создадим таблицу Routes. Связь между таблицами Transits и Routes «один к одному». Кроме того, для хранения рассчитанных оценок создадим таблицу TransitStadies, которая имеет связь с таблицей Transits «многие к одному». Для разбиения карты на области создадим таблицу Regions. Города хранятся в таблице Cities и имеют связь с таблицей Regions «многие к одному». Для хранения информации о водителе создадим таблицу Drivers, содержащую ID водителя и его ФИО. Все контакты хранятся в таблице Contacts, связанной с Drivers отношением «многие к одному». Для хранения информации о грузах создадим таблицу Consignments.



## 1.5 ER-диаграмма

ER-диаграмма базы данных, составленная в результате анализа предметной области и определения целей создания приложения, показана на рис. 1.2.

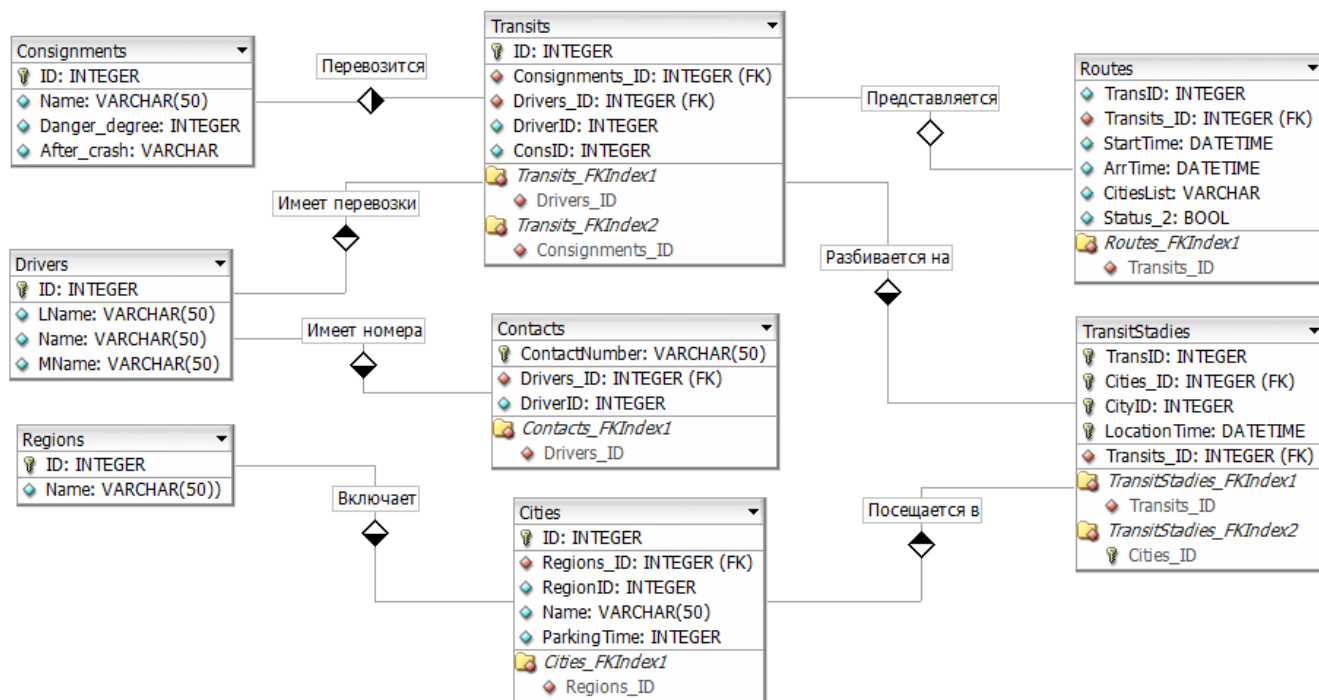


Рисунок 1.2. ER-диаграмма

## **2 Конструкторская часть**

### **2.1. Обоснование выбора СУБД**

В качестве используемой СУБД была выбрана система MS SQL Server 2008R2.

Данная система является одной из наиболее распространённых в отрасли, а также является лёгкой в работе благодаря значительной интеграции со стороны связки C# + .Net Framework (LINQ to SQL и другие способы доступа к данным). В конечном счёте именно интеграция с C# и .Net Framework оказалась решающим фактором в выборе СУБД, поскольку вся логика, связанная с работой с БД, сосредоточена в клиентском приложении; сама же база лишь выступает средством хранения данных, не осуществляя никакого их анализа помимо валидации при записи.

### **2.2 Определение типов данных для столбцов таблиц базы данных Course\_DB**

Рекомендуемые типы данных для всех столбцов приведены на рис. 1.2.

### **2.3 Ограничения целостности**

#### **2.3.1 Первичные ключи**

Все таблицы содержат первичные ключи для быстрого доступа к строкам таблицы по номеру первичного ключа, а также для организации связи между таблицами.

#### **2.3.2 Внешние ключи**

С помощью внешних ключей осуществляется связь между сущностями. Каждый столбец таблицы, содержащий указатель на элемент другой таблицы, является внешним ключом. Внешние ключи осуществляют реализацию связей «один-ко-многим» и «один-к-одному».

## 2.4 Диаграмма базы данных

На рис. 2.1 показана диаграмма базы данных, составленная в результате описания типов данных в базе и составления запроса создания базы.

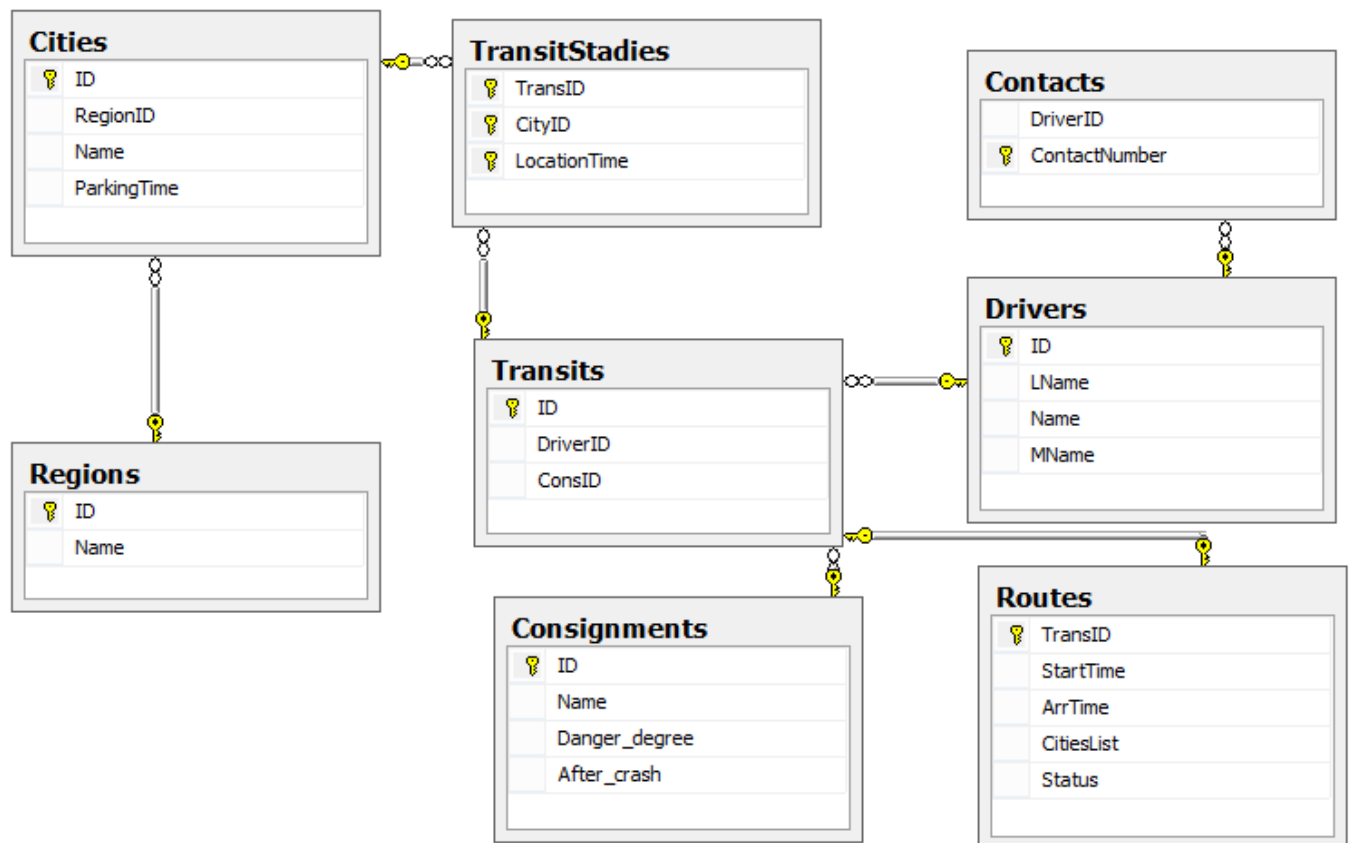


Рисунок 2.1. Диаграмма базы данных

## 2.5 Алгоритм прокладывания маршрутов

Алгоритм Дейкстры находит кратчайшие расстояния во взвешенном ориентированном графе без петель. Его действие легко распространяется на взвешенный неориентированный граф с введением пометки посещённых вершин. Маршрут можно определить, если ввести ассоциативный массив, в котором будет храниться вершина и информация о предшествующей ей вершине. Тогда на обратном ходе можно восстановить путь из начальной вершины в конечную.

Ниже приведено описание алгоритма на псевдокоде:

- $V$  — множество вершин графа
- $E$  — множество ребер графа
- $w[ij]$  — вес (длина) ребра  $ij$
- $a$  — вершина, расстояния от которой ищутся
- $U$  — множество посещенных вершин
- $d[u]$  — по окончании работы алгоритма равно длине кратчайшего пути из  $a$  до вершины  $u$
- $p[u]$  — по окончании работы алгоритма содержит кратчайший путь из  $a$  в  $u$

Присвоим  $d[a] \leftarrow 0, p[a] \leftarrow a$

Для всех  $u \in V$  отличных от  $a$

присвоим  $d[u] \leftarrow \infty$

Пока  $\exists v \notin U$

Пусть  $v \notin U$  — вершина с минимальным  $d[v]$

занесём  $v$  в  $U$

Для всех  $u \notin U$  таких, что  $vu \in E$

если  $d[u] > d[v] + w[vu]$  то

изменим  $d[u] \leftarrow d[v] + w[vu]$

изменим  $p[u] \leftarrow p[v], u$

На рис.2.2. приведена структурная схема алгоритма Дейкстры.

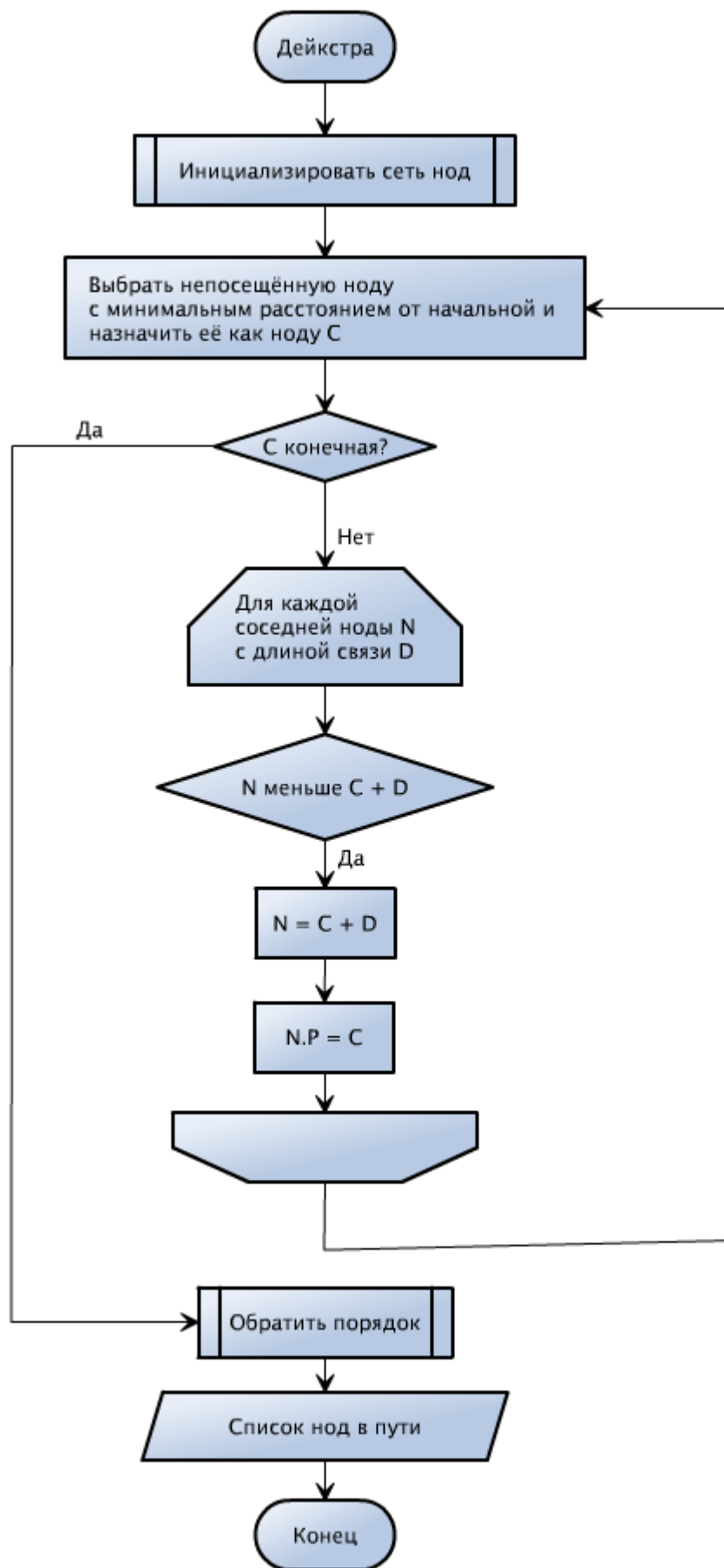


Рисунок 2.2. Схема алгоритма Дейкстры

## **3 Технологическая часть**

### **3.1 Обоснование выбора языка и среды разработки**

Для решения поставленной задачи была выбрана интегральная среда разработки (ИСР) Microsoft Visual Studio.NET поскольку:

- в ней доступны широкие возможности языка C#;
- она имеет все необходимые средства и инструменты для разработки данного проекта;
- удобна в использовании, гибка в настройке;
- в ней имеется возможность использования готовых библиотек классов;
- позволяет импортировать исходный код и проекты для ИСР других ОС.

Так как ИСР выбрана Microsoft Visual Studio.NET, соответственно языком программирования выбран C#, поскольку:

- «родной» язык для создания приложений в среде .NET;
- подлинная объектная ориентированность (всякая языковая сущность претендует на то, чтобы быть объектом);
- компонентно-ориентированное программирование;
- безопасный (по сравнению с языками C и C++) код;
- унифицированная система типизации;
- поддержка событийно-ориентированного программирования;

Также использование данного языка облегчает работу с базой данных благодаря механизму LINQ to SQL, позволяющему работать с реляционной БД через её отображение на объектно-сущностную модель.

### **3.2 Взаимодействие с базой данных**

Язык C# предлагает несколько способов взаимодействия с базой данных. В данной работе используются технология LINQ to SQL.

В LINQ to SQL модель данных реляционной базы данных сопоставляется объектной модели, выраженной средствами языка программирования. При запуске приложения LINQ to SQL преобразует запросы LINQ из объектной модели в SQL и отправляет их в базу данных для выполнения. Когда база данных возвращает результаты, LINQ to SQL преобразует их обратно в объекты.

При помощи этой технологии вкупе с возможностями языка C# можно создавать компактные решения однотипных задач. Например, получить количество записей в любой таблице или последнюю запись из таблицы, что значительно сокращает время отладки приложения.

### **3.3 Запросы к базе данных**

Вся логика работы с базой данных реализована в классе QueriesHandler (см. Рисунок 3.3.2.). Основная задача анализа опасности подразумевает выборку из нескольких таблиц базы с рядом проверок и ограничений.

Результатом анализа является следующая информация:

1. наименование опасного вещества;
2. уровень опасности;
3. предположительное местонахождение;
4. предписанные действия;
5. имя и фамилия водителя;
6. контактные номера водителя;
7. предположительное время нахождения определённом месте.

Помимо получения информации о результатах анализа опасности, пользователь может получить подробную информацию о каждом водителе, а именно:

1. фамилию, имя, отчество;
2. контактные телефоны;
3. ID в базе данных;
4. предположительное текущее местонахождение;
5. информацию о последней перевозке:
  - а. начальный и конечный пункты;
  - б. наименование груза (опасность помечается цветом);
  - в. время отправления (при незавершённой перевозке);
  - г. время прибытия (предположительное при незавершённой перевозке).

### **3.4 Технологические особенности проекта**

Архитектура данного проекта подразумевает его расширяемость: гибкое взаимодействие с базой данных и с картой. Другими словами приложение должно «уметь» переключаться с одной

базы на другую, с минимальными изменениями в коде, чтобы в дальнейшем этот процесс можно было автоматизировать. Так же и с картой.

В связи с этим были разработаны интерфейсные классы `IDataHandler` и `IMap`, описывающие функционал, который обязаны предоставлять классы, реализующие эти интерфейсы. Классы пространства имён (рис. 3. 2) пользовательского приложения взаимодействуют именно с интерфейсами, «не зная» ничего об их реализации. Таким образом достигается отделение интерфейса от реализации (рис. 3.1).

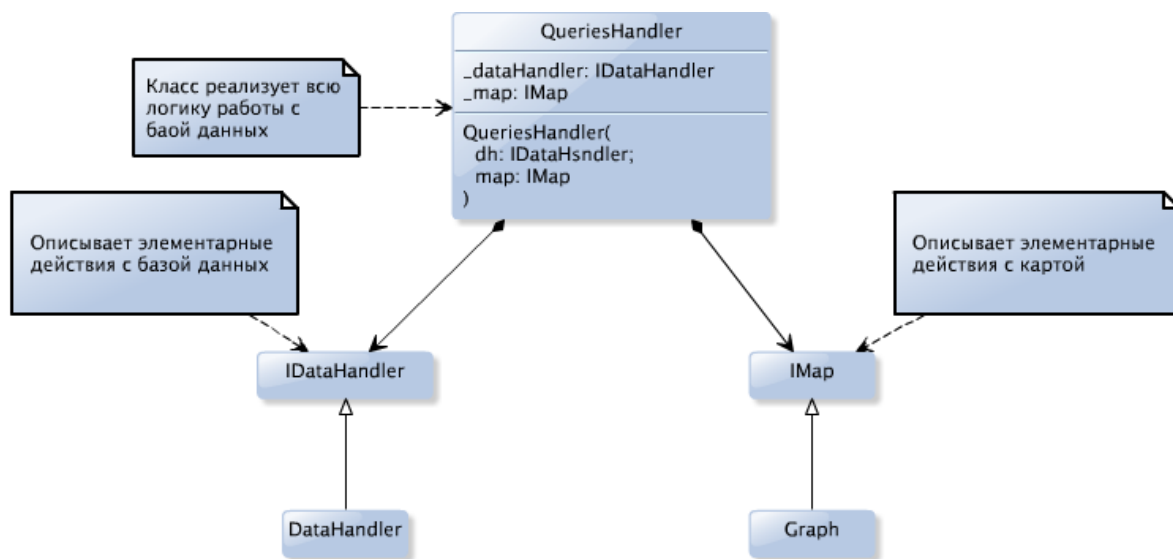


Рисунок 3.1. Гибкость приложения

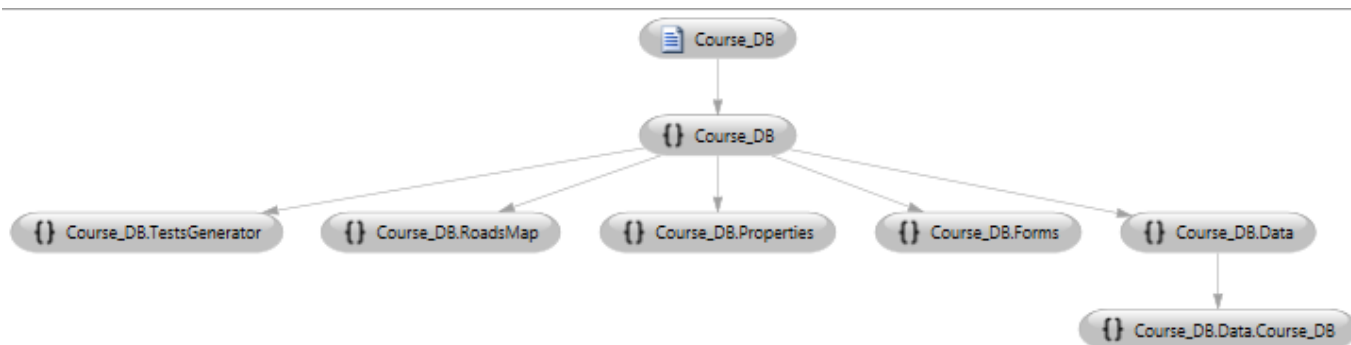


Рисунок 3.2. Пространства имён приложения



### 3.4.1. Класс IMap

```
/// <summary>
/// Интерфейс для карты. Классы-реализации должны предоставлять пользователю структуру карты,
возможность её обновить
/// и возможность находить кратчайшие пути с возможным объездом множества населённых пунктов.
/// </summary>
public interface IMap
{
    /// <summary>
    /// Ключ - ID основного города
    /// Значение - ассоциативный массив, ключ которого - ID города, смежного с основным,
    /// а значение - время в минутах, необходимое для переезда в город из основного
    /// </summary>
    Dictionary<int, Dictionary<int, int>> RoadMap { get; }

    /// <summary>
    /// Перезагрузить карту
    /// </summary>
    void Reload();

    /// <summary>
    /// Ключ записи словаря - идентификационный номер города в базе данных,
    /// значение записи словаря - время поездки из пункта отправления
    /// </summary>
    /// <param name="start">ID начального пункта</param>
    /// <param name="goal">ID конечного пункта</param>
    /// <param name="startValue">Время, которое автомобиль был в пути до прибытия в начальный
пункт (по умолчанию 0)</param>
    /// <returns></returns>
    List<KeyValuePair<int, int>> GetShortTrack(int start, int goal, int startValue = 0);

    /// <summary>
    /// Ключ записи словаря - идентификационный номер города в базе данных,
    /// значение записи словаря - время поездки из пункта отправления
    /// </summary>
    /// <param name="start">ID начального пункта</param>
    /// <param name="goal">ID конечного пункта</param>
    /// <param name="dont_drive">Вектор городов, которые нужно объехать</param>
    /// <param name="startValue">Время, которое автомобиль был в пути до прибытия в начальный
пункт (по умолчанию 0)</param>
    /// <returns></returns>
    List<KeyValuePair<int, int>> GetShortTrackWithoutPoints(int start, int goal, double[]
dont_drive, int startValue = 0);
}
```

### 3.4.2. Класс IDataHandler

```
/// <summary>
/// Предоставляет необходимый функционал для работы с данными, абстрагируясь от способа их
хранения
/// </summary>
public interface IDataHandler
{
    List<int> MissedDriverIDs { get; }

    /// <summary>
    /// Добавляет новый контакт
    /// </summary>
    /// <param name="driverID">Идентификатор владельца контакта</param>
    /// <param name="contact">Номер телефона</param>
    void AddNewContact(int driverID, string contact);

    /// <summary>
    /// Удалить все контакты, записанные за водителя
    /// </summary>
    /// <param name="driverID">ID водителя</param>
    void DelContacts(int driverID);

    /// <summary>
    /// Добавить нового водителя
    /// </summary>
    /// <param name="lName">Фамилия</param>
    /// <param name="name">Имя</param>
    /// <param name="mName">Отчество, если есть</param>
    /// <returns>ID добавленного водителя</returns>
    int AddNewDriver(string lName, string name, string mName);

    /// <summary>
    /// Удалить водителя
    /// </summary>
    /// <param name="driverID"></param>
    void DelDriver(int driverID);

    /// <summary>
    /// Добавить новый груз
    /// </summary>
    /// <param name="name">Наименование</param>
    /// <param name="dangerDegree">Уровень опасности</param>
    /// <param name="afterCrash">Действия при аварии</param>
    /// <returns>ID добавленного груза</returns>
    int AddNewConsignment(string name, int dangerDegree, string afterCrash);

    /// <summary>
    /// Добавить новую перевозку
    /// </summary>
    /// <param name="driverID"></param>
    /// <param name="consignmentID"></param>
    /// <returns>ID добавленной перевозки</returns>
    int AddNewTransit(int driverID, int consignmentID);

    /// <summary>
    /// Удалить перевозку
    /// </summary>
    /// <param name="transID"></param>
    void DelTransit(int transID);

    /// <summary>
    /// Добавить новый маршрут
```

```

    /// </summary>
    /// <param name="transitID"></param>
    /// <param name="startTime">Время отправления</param>
    /// <param name="arrTime">Время прибытия (отмечается по прибытии)</param>
    /// <param name="cities">Список проезжаемых городов</param>
    /// <param name="status">Завершена ли перевозка</param>
    void AddNewRoute(int transitID, DateTime startTime, DateTime arrTime, string cities, bool
status);

    /// <summary>
    /// Удалить информацию о маршруте
    /// </summary>
    /// <param name="transID">ID перевозки</param>
    void DelRoute(int transID);

    /// <summary>
    /// Добавить новый населённый пункт
    /// </summary>
    /// <param name="name">Название</param>
    /// <param name="parkingTime">Время стоянки</param>
    /// <param name="regionID">ID региона</param>
    /// <returns>ID добавленного населённого пункта</returns>
    int AddNewCity(string name, int parkingTime, int regionID);

    /// <summary>
    /// Добавить регион
    /// </summary>
    /// <param name="name">Название региона</param>
    /// <returns>ID добавленного региона</returns>
    int AddNewRegion(string name);

    /// <summary>
    /// Добавить новую стадию перевозки. Стадии перевозки нужны для быстрого поиска
информации обо всех водителях в конкретном месте в конкретное время.
    /// </summary>
    /// <param name="transitID">ID перевозки</param>
    /// <param name="cityID">ID населённого пункта</param>
    /// <param name="noticedTime">Время нахождения</param>
    void AddNewTransitStady(int transitID, int cityID, DateTime noticedTime);

    /// <summary>
    /// Обновляет уровень опасности и действия после аварии.
    /// </summary>
    /// <param name="consID"></param>
    /// <param name="dangerDegree"></param>
    /// <param name="afterCrash"></param>
    void SetConsignmentParameters(int consID, int dangerDegree, string afterCrash);

    /// <summary>
    /// Устанавливает время прибытия и статус завершения для перевозки.
    /// </summary>
    /// <param name="transID"></param>
    /// <param name="start"></param>
    /// <param name="arr"></param>
    void SetEndingStatus(int transID, DateTime start, DateTime arr);

    /// <summary>
    /// Удалить все стадии перевозки с ID этой перевозки
    /// </summary>
    /// <param name="transID"></param>
    void DeleteStadiesByTransitID(int transID);

    /// <summary>
    /// Возвращает последнюю запись в таблице

```

```

    /// </summary>
    /// <typeparam name="T">Типом должен являться пользовательский класс, описывающий таблицу
БД</typeparam>
    /// <returns></returns>
    T GetLastInTable<T>();

    /// <summary>
    ///
    /// </summary>
    /// <typeparam name="T"></typeparam>
    /// <returns>Количество записей в таблице</returns>
    int GetTableLength<T>();

    /// <summary>
    /// Существует ли в базе данных такой номер телефона
    /// </summary>
    bool HasPhoneNumber(string contact);

    /// <summary>
    /// Возвращает массив идентификаторов водителей с заданными ФИО
    /// </summary>
    int[] FindDrivers(string lName, string name, string mName);

    /// <summary>
    /// Возвращает ID водителя по его номеру телефона
    /// </summary>
    int DriverWithPhoneNumber(string contact);

    string GetDriversFullName(int driverID);
    List<string> GetDriversFullNames();

    /// <summary>
    /// Возвращает номер населённого пункта по его названию.
    /// </summary>
    /// <param name="name"></param>
    /// <returns></returns>
    int GetCityID(string name);

    string GetCityName(int cityID);

    /// <summary>
    ///
    /// </summary>
    /// <param name="name"></param>
    /// <returns>Номер региона (или округа, если быть точным)</returns>
    int GetRegionID(string name);

    /// <summary>
    /// Найти идентификатор груза по названию груза.
    /// </summary>
    /// <param name="name"></param>
    /// <returns></returns>
    int GetConsignmentID(string name);

    /// <summary>
    /// Найти ID перевозки по ID водителя и времени отправления
    /// </summary>
    /// <param name="driverID"></param>
    /// <param name="start"></param>
    /// <returns></returns>
    int GetTransitID(int driverID, DateTime start);

    /// <summary>
    /// Найти ID всех перевозок в заданный промежуток времени в заданном городе

```

```

/// </summary>
/// <returns></returns>
List<int> GetTransitIDs(DateTime start, DateTime until, int placeID);

/// <summary>
///
/// </summary>
/// <param name="start"></param>
/// <param name="until"></param>
/// <param name="placeID"></param>
/// <returns></returns>
List<int> GetTransitIDs(int driverID);

/// <summary>
/// Получить список из ID перевозок, зарегистрированных ранее указанного времени
/// </summary>
/// <param name="time">"Верхнее" время регистрации</param>
/// <returns></returns>
List<int> TransitsBefore(DateTime time);

/// <summary>
/// Получить список из ID завершённых перевозок
/// </summary>
/// <returns></returns>
List<int> EndedTransits();

/// <summary>
///
/// </summary>
/// <returns>Список названий всех городов</returns>
List<string> GetCitiesNames();

/// <summary>
///
/// </summary>
/// <param name="regionID"></param>
/// <returns>Список идентификаторов городов в регионе (округе)</returns>
List<int> GetCitiesInRegion(int regionID);

/// <summary>
///
/// </summary>
/// <returns>Список названий всех грузов</returns>
List<string> GetConsignmentsNames();

/// <summary>
///
/// </summary>
/// <returns>Список всех контактов</returns>
List<string> GetNumbers();

/// <summary>
///
/// </summary>
/// <returns>Список названий всех регионов</returns>
List<string> GetRegionsNames();

/// <summary>
/// Позволяет узнать продолжительность остановки/стоянки в минутах в заданном городе
/// </summary>
/// <param name="cityID"></param>
/// <returns>Количество минут стоянки</returns>
int GetParkingMinutesOfTheCity(int cityID);

```

```

/// <summary>
/// Определить водителя по номеру перевозки
/// </summary>
/// <param name="transID">ID перевозки</param>
/// <returns>ID водителя</returns>
int GetDriverID(int transID);

/// <summary>
/// Определить груз по номеру перевозки
/// </summary>
/// <param name="transID"></param>
/// <returns></returns>
int GetConsignmentID(int transID);

/// <summary>
/// Получить имя водителя
/// </summary>
/// <param name="driverID">ID водителя</param>
/// <returns></returns>
string GetDriverName(int driverID);

/// <summary>
/// Получить фамилию водителя
/// </summary>
/// <param name="driverID">ID водителя</param>
/// <returns></returns>
string GetDriverSurname(int driverID);

/// <summary>
/// Получить контактные номера водителя
/// </summary>
/// <param name="driverID">ID водителя</param>
/// <returns>Список номеров</returns>
List<string> GetDriverNumbers(int driverID);

/// <summary>
/// Получить название груза
/// </summary>
/// <param name="consID">ID груза</param>
/// <returns></returns>
string GetConsignmentName(int consID);

/// <summary>
/// Получить описание действий при аварии
/// </summary>
/// <param name="consID">ID груза</param>
/// <returns></returns>
string GetAfterCrashInfo(int consID);

/// <summary>
/// Получить уровень опасности груза
/// </summary>
/// <param name="consID">ID груза</param>
/// <returns></returns>
int GetDangerDegree(int consID);

/// <summary>
/// Получить время нахождения водителя в городе в рамках перевозки
/// </summary>
/// <param name="transID">ID перевозки</param>
/// <param name="cityID">ID города</param>
/// <returns>Оцененное время, когда водитель будет в этом городе</returns>
List<DateTime> GetLocationTime(int transID, int cityID);

```

```

    /// <summary>
    /// Получить время окончания завершённой перевозки
    /// </summary>
    /// <param name="transID">ID перевозки</param>
    /// <returns></returns>
    DateTime GetArrival(int transID);

    /// <summary>
    /// Узнать города, посещаемые водителем в рамках данной перевозки
    /// </summary>
    /// <param name="transID">ID перевозки</param>
    /// <returns>Строка, в которой через пробел записаны ID городов в порядке их
    посещения</returns>
    string GetVisitsCities(int transID);

    /// <summary>
    /// Получить текущее местонахождение водителя
    /// </summary>
    /// <param name="transID">ID перевозки</param>
    /// <returns></returns>
    int GetCurrentLocation(int transID);

    /// <summary>
    /// Получить статус перевозки
    /// </summary>
    /// <param name="transID">ID перевозки</param>
    /// <returns></returns>
    bool GetStatus(int transID);
}

```

### 3.5 Пользовательский интерфейс

Пользовательский интерфейс должен иметь удобный формат для организации запросов в базу, редактирования данных базы, удаления и добавления новых элементов в базу данных.

Каждая из операций работы с базой должна быть максимально удобной и интуитивно понятной. Пользователь не должен иметь возможности ввода запроса, который может принести вред базе, такой как удаление базы и т.п.

В главном окне приложения пользователь сразу имеет возможность анализировать опасность. При запросах, выполняющихся долгое время, появляется полоса загрузки, которая «показывает» пользователю, что система на самом деле работает. Главное окно приложения показано на рис. 3.3.

Рисунок 3.3. Главное окно приложения

По завершении анализа отображается окно результата анализа (рис. 3.4). Грузы в выпадающем списке отсортированы по классу опасности (сначала идут наиболее опасные). Цвет (на рисунке салатовый) отражает уровень опасности перевозимого груза.

Рисунок 3.4. Результат анализа

Форма для заполнения путевого листа представлена на рис. 3.5.



Заполнить путевой лист

\* Наименование груза (вещества): Бензин

\* Пункт отправления: Владимир

Промежуточные пункты: Курск Ещё

\* Пункт прибытия: Ногинск

\* Телефон водителя: (111) 111-1111

\* Дата отправления: 15.05.2014 19:52

Добавить

Рисунок 3.5. Новый путевой лист

Форма установки статуса завершения перевозки показана на рис. 3.6.

Установить статус завершения для перевозки

\* Телефон водителя: (111) 111-1111

\* Дата отправления: 15.05.2014 19:57

\* Дата прибытия: 15.05.2014 23:57

Установить

Рисунок 3.6. Установка статуса завершения перевозки

Форма предоставления пользователю информации о водителях показана на рис. 3.7.

Найти информацию о водителе

Водитель: Васинский Павел Алексеевич

Телефоны: 4999999609 ID 391

Предположительное место нахождения: Москва

**Последняя перевозка** Откуда: Москва Куда: Ярославль

Статус: Не завершена Время отправления: 17.05.2014 15:33

Груз: Бензин Время прибытия (предположительное): 18.05.2014 04:41

Рисунок 3.7. Информация о водителях

## 4 Руководство пользователя

### 4.1 Анализ опасности

В поле ввода места аварии начните вводить первые буквы названия города и выберите из выпавшего списка необходимый (рис. 4.1). Для указания региона, нажмите на «Регион» и повторите действия.

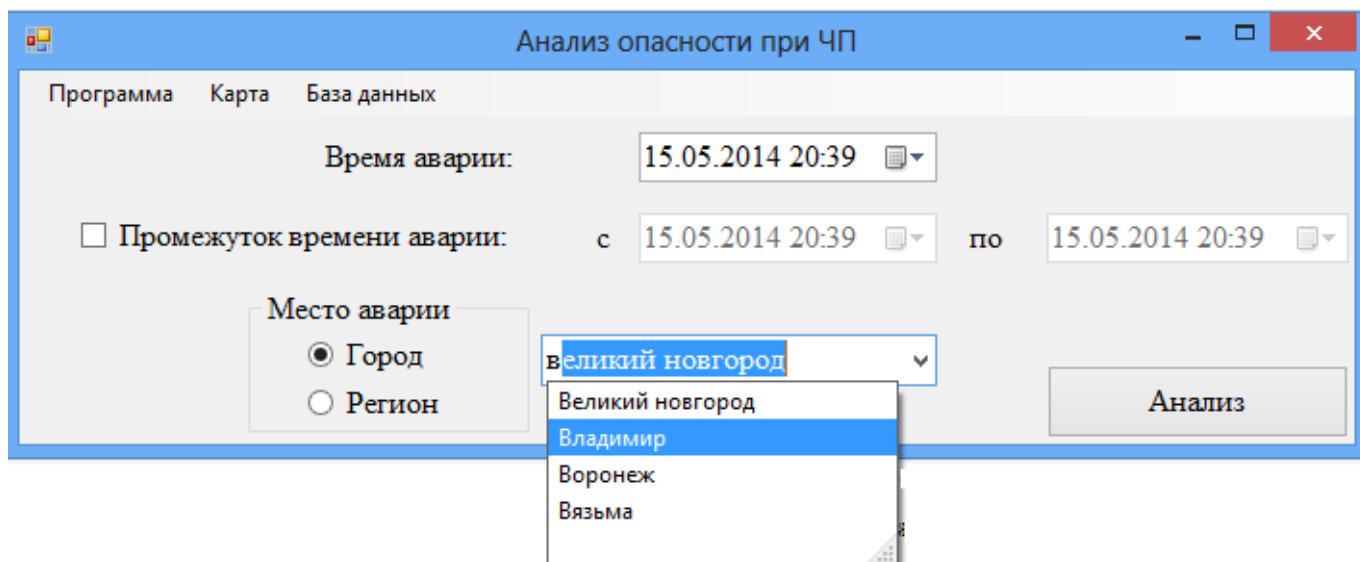


Рисунок 4.1. Анализ

Для указания промежутка времени аварии, нажмите на «Промежуток времени аварии» и введите данные. Затем нажмите «Анализ» и подождите. Процесс загрузки возможных перевозок, находящихся в заданном месте в заданное время, отображается в полосе загрузки над кнопкой «Анализ».

### 4.2 Заполнение путевого листа

Для заполнения выполните Программа – Новый путевой лист. При заполнении путевого листа данные не заносятся в базу данных. Возможно только выборка из неё, поэтому необходимо вводить существующие данные, иначе могут быть получены различные предупреждения (рис. 4.2-4).

Город, указанный в поле «Промежуточные пункты» учитывается. Если нужно больше промежуточных пунктов, нажмите кнопку «Ещё» и вводите следующий город. Промежуточные пункты должны вводиться в порядке их посещения.

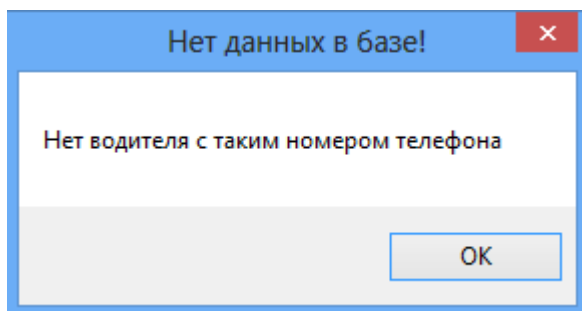


Рисунок 4.2. Отсутствие водителя

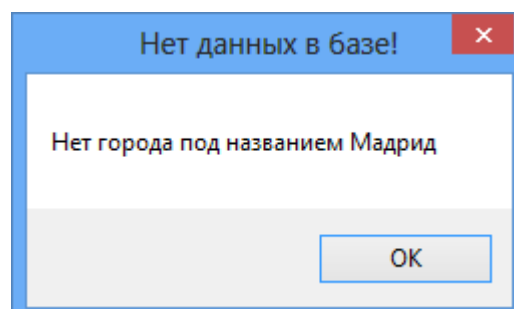


Рисунок 4.3. Отсутствие города

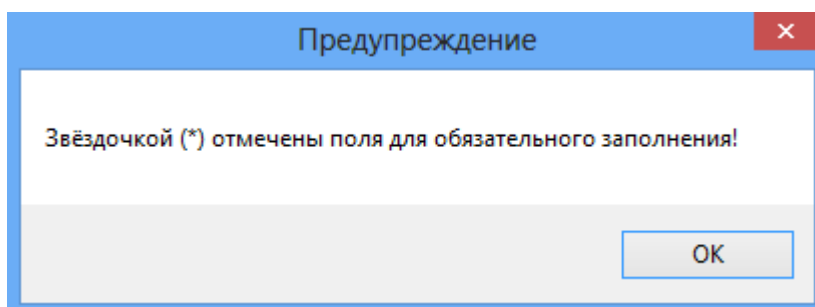


Рисунок 4.4. Заполнены не все поля

### 4.3 Добавление

Чтобы добавить водителя, выполните База данных – Добавить – Водителя. В появившемся окне заполните как минимум поля, отмеченные «\*», и нажмите «Добавить».

Если водитель с таким именем уже зарегистрирован, то откроется окно обработки ситуации с тёзками (рис. 4.5). При нажатии кнопки «Добавить номер к существующему», если найденный водитель один, то произойдёт добавление, если больше, чем один, то будет предложено выбрать водителя, к которому необходимо добавить контакт, по его предыдущему номеру телефона.

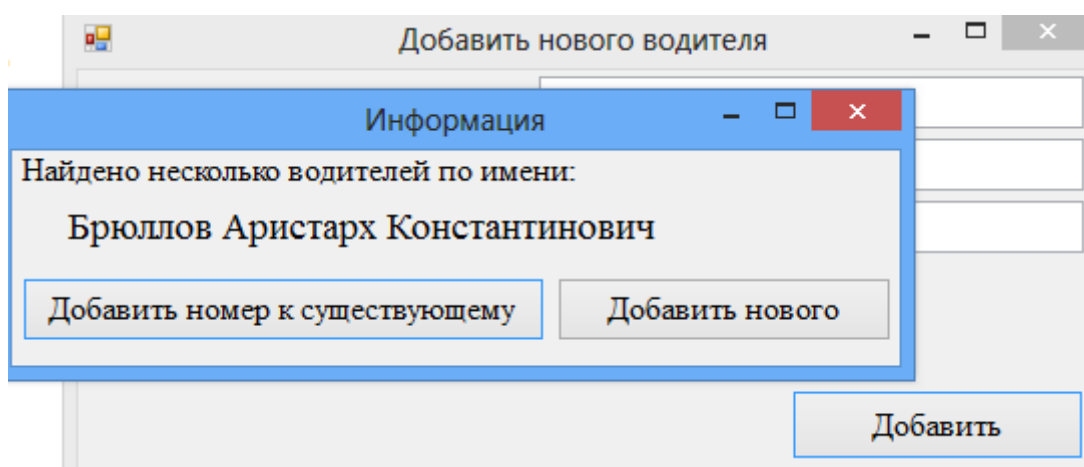


Рисунок 4.5. Обработка ситуации с тёзками

Для добавления груза выполните База данных – Добавить – Груз. Если груз с таким именем уже существует, произойдёт замена его класса опасности и действий при аварии на введённые.

#### 4.4 Удаление

Для удаления необходимо обладать правами администратора. При попытке удаления чего-либо система запросит у пользователя пароль (рис. 4.6).

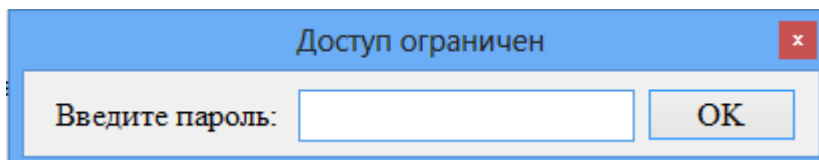


Рисунок 4.6. Запрос пароля.

Система позволяет удалить водителя, перевозки ранее заданного времени и завершённые перевозки.

Удаление водителя происходит по его номеру телефона. При этом удаляются все зарегистрированные на него перевозки и контактные номера.

#### 4.4 Поиск информации

Для получения информации о водителе нажмите База данных – Найти – Информацию о водителе. В появившемся окне (Рисунок 3.7) в поле имени водителя начните вводить его имя и из выпадающего списка выбирайте необходимого водителя. Нажмите на номер его телефона, чтобы развернуть список и посмотреть все номера.

## **Заключение**

В результате проделанной работы разработан программный комплекс, позволяющий оперативно найти все перевозки опасных грузов, осуществляющиеся в заданный момент времени близи заданного месте. Спроектирована и реализована в СУБД MS SQL Server 2008 R2 реляционная база данных, осуществляющая хранение данных для анализа. Смоделирована карта на основе графовой структуры. Разработано приложение с графическим интерфейсом, реализующее безопасный доступ пользователя к данным базы, их изменение, удаление, и добавление новых данных.

Разработанная гибкая архитектура приложения позволяет расширять его с минимальными изменениями, что важно, потому что систему планируется модернизировать. В дальнейшем будет осуществлена попытка перейти на другую модель хранения данных, возможно, NoSQL Document Data Model. В перспективе возможна клиент-серверная связь с базой данных, чтобы работать с одной централизованной системой и принимать данные от различных баз данных. Также будет осуществлена попытка перехода на реальную карту с возможностью задания места аварии координатами на карте, а не только конкретным городом или областью. Что касается временных оценок, будет произведён более подробный анализ этой предметной области и попытка найти метод расчёта времени, затрачиваемого на переезд, близкого к реальному. Возможно, для этого придётся разрабатывать обучаемый модуль (например, нейросеть).

## **Список литературы**

1. MSDN Library. Документация библиотеки .NET. Microsoft corp.

Web: <http://msdn.microsoft.com/ru-ru/library/default.aspx>

2. Stackoverflow. Система вопросов и ответов. USA, Stack Overflow Internet Services, inc.

Web: <http://www.stackoverflow.com>

3. Wikipedia. Электронная энциклопедия. Wikimedia Foundation, inc.

Web: <http://www.wikipedia.org/>

4. Просуков Е.А. Базы данных. Курс лекций, МГТУ 2013-2014.