СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc27183574)

[1. ПОСТРОЕНИЕ ИНФОЛОГИЧЕСКОЙ КОНЦЕПТУАЛЬНОЙ МОДЕЛИ 6](#_Toc27183575)

[1.1 Анализ предметной области и выявление необходимого набора сущностей 6](#_Toc27183576)

[1.2 Обоснование требуемого набора атрибутов для каждой сущности и выделение идентифицирующих атрибутов 7](#_Toc27183577)

[1.3 Определение связей между объектами 10](#_Toc27183578)

[1.4 Описание полученной модели на языке инфологического проектирования 10](#_Toc27183579)

[2. ПОСТРОЕНИЕ СХЕМЫ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ 12](#_Toc27183580)

[2.1 Построение набора необходимых отношений базы данных 12](#_Toc27183581)

[2.2 Задание первичных и внешних ключей определенных отношений 13](#_Toc27183582)

[2.3 Третья нормальная форма 13](#_Toc27183583)

[2.4 Определение ограничений целостности для внешних ключей отношений и для отношений в целом 14](#_Toc27183584)

[2.5 Графическое представление связей между внешними и первичными ключами 15](#_Toc27183585)

[3. СОЗДАНИЕ СПРОЕКТИРОВАННОЙ БАЗЫ ДАННЫХ 16](#_Toc27183586)

[4 ЗАПИСЬ ВЫРАЖЕНИЙ, УКАЗАННЫХ В ВАРИАНТЕ ЗАДАНИЯ ТИПОВ ЗАПРОСОВ НА ЯЗЫКЕ SQL 21](#_Toc27183587)

[5 ВЫБОР И ОСНОВАНИЕ СРЕДСТВ РАЗРАБОТКИ ПРИЛОЖЕНИЯ 23](#_Toc27183588)

[6 РЕАЛИЗАЦИЯ ЗАКОНЧЕННОГО ПРИЛОЖЕНИЯ, РАБОТАЮЩЕГО С СОЗДАННОЙ БАЗОЙ ДАННЫХ 26](#_Toc27183589)

[6.1 Разработка и построение интерфейса главной и рабочих форм 26](#_Toc27183590)

[6.2 Построение главного меню и кнопок панели инструментов 26](#_Toc27183591)

[6.3 Выполнение программного кода на Microsoft Visual C# 26](#_Toc27183592)

[ЗАКЛЮЧЕНИЕ 28](#_Toc27183593)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 29](#_Toc27183594)

[ПРИЛОЖЕНИЕ А 30](#_Toc27183595)

[ПРИЛОЖЕНИЕ Б 31](#_Toc27183596)

[ПРИЛОЖЕНИЕ В 32](#_Toc27183597)

[ПРИЛОЖЕНИЕ Г 35](#_Toc27183598)

## ВВЕДЕНИЕ

Современное производство невозможно без эффективного управления. Важной составляющей являются системы обработки информации, от которых во многом зависит эффективность работы любого предприятия ли учреждения. Такая система должна:

•   обеспечивать получение общих и/или детализированных отчетов по итогам работы;

•   позволять легко определять тенденции изменения важнейших показателей;

•   обеспечивать получение информации, критической по времени, без существенных задержек;

•   выполнять точный и полный анализ данных.

Среди наиболее ярких представителей систем управления базами данных можно отметить: Lotus Approach, Microsoft Access, Borland dBase, Borland Paradox, Microsoft Visual FoxPro, Microsoft Visual Basic, а также баз данных Microsoft SQL Server и Oracle, используемые в приложениях, построенных по технологии «клиент-сервер». Фактически, у любой современной СУБД существует аналог, выпускаемый другой компанией, имеющий аналогичную область применения и возможности, любое приложение способно работать со многими форматами представления данных, осуществлять экспорт и импорт данных благодаря наличию большого числа конвертеров. Общепринятыми, также, являются технологи, позволяющие использовать возможности других приложений, например, текстовых процессоров, пакетов построения графиков и т.п., и встроенные версии языков высокого уровня (чаще – диалекты SQL или VBA) и средства визуального программирования интерфейсов разрабатываемых приложений. Поэтому уже не имеет существенного значения, на каком языке и на основе какого пакета написано конкретное приложение, и какой формат данных в нем используется. Более того, стандартом фактически стала «быстрая разработка приложений», или RAD (от английского Rapid Application Development), основанная на широко декларируемом в литературе «открытом подходе», то есть необходимость и возможность использования различных прикладных программ и технологий для разработки более гибких и мощных систем обработки данных. Поэтому в одном ряду с «классическими» СУБД все чаще упоминаются языки программирования Visual Basic  и Visual C++, которые позволяют быстро создавать необходимые компоненты приложений, критичные по скорости работы, которые трудно, а иногда невозможно разработать средствами «классических» СУБД. Современный подход к управлению базами данных подразумевает также широкое использование технологии «клиент-сервер».

Таким образом, на сегодняшний день разработчик не связан рамками какого-либо конкретного пакета, а в зависимости от поставленной задачи может использовать самые разные приложения. Поэтому, более важным представляется общее направление развития СУБД и других средств разработки приложений в настоящее время.

Основные идеи современной информационной технологии базируются на концепции, согласно которой данные должны быть организованы в базы данных с целью адекватного отображения изменяющегося реального мира и удовлетворения информационных потребностей пользователей. Эти базы данных создаются и функционируют под управлением специальных программных комплексов, называемых системами управления базами данных (СУБД).

Увеличение объема и структурной сложности хранимых данных, расширение круга пользователей информационных систем привели к широкому распространению наиболее удобных и сравнительно простых для понимания реляционных (табличных) СУБД. Для обеспечения одновременного доступа к данным множества пользователей, нередко расположенных достаточно далеко друг от друга и от места хранения баз данных, созданы сетевые мультипользовательские версии БД основанных на реляционной структуре. В них тем или иным путем решаются специфические проблемы параллельных процессов, целостности (правильности) и безопасности данных, а также санкционирования доступа.

В данной курсовой работе поставлена задача базы данных и создания информационной системы автотранспортного предприятия города.

Для создания информационной базы данных будет использоваться СУДБ MsSQL. Для создания приложения – среда Visual Studio 2019.

## ПОСТРОЕНИЕ ИНФОЛОГИЧЕСКОЙ КОНЦЕПТУАЛЬНОЙ МОДЕЛИ

* 1. Анализ предметной области и выявление необходимого набора сущностей

В ходе анализа задания и разработке базы данных были выявлены основные сущности:

* Сущность *Автомобиль* описывает основные характеристики автомобиля, такие как цвет, вместительность, дату списания и дату прибытия, номер автомобиля.
* Сущность *Сотрудник* описывает всех сотрудников, которые работают или работали в автотранспортном предприятии. Характеризуются полным именем, возраста и номером телефона.
* Сущность *Путь* описывает все пути движения транспорта. Характеризуются названием пути, временем в пути, длинной пути, начальной и конечно точной движения.
* Сущность *Модель* описывает основные характеристики данной модели транспортного средства, такие как тип кузова, тип автомобиля, название модели и средний расход.
* Подсущность *Цех* описывает помещение, в котором содержаться автомобили и работаю сотрудники.
* Подсущность *Распределение* описывает распределение рабочих и автомобилей на маршрут. Характеризуется датой распределения на маршрут и датой окончания работы на маршруте.
* Подсущность *Производитель* описывает производителя автомобиля. Характеризуется наименованием производителя.
* Подсущность *Профессия* описывает профессии сотрудников. Характеризуется наименование профессии.
* Сущность *Ремонт* описывает все ремонты автомобиля. Характеризуется датой ремонта, рабочим, который ремонтировал и работой, которую выполнил рабочий.
* Подсущность *Тип автомобиля* описывает типы существующих автомобилей. Характеризуется наименованием типа авто.
* Подсущность *Тип ремонта* описывает тип ремонтов, выполняемых на автомобилях. Характеризуется наименованием выполняемого ремонта и стоимостью.
  1. Обоснование требуемого набора атрибутов для каждой сущности и выделение идентифицирующих атрибутов

Для построения инфологической концептуальной модели необходимо для каждой сущности, выявленной в предыдущем пункте, определить требуемый набор атрибутов. Атрибутом является поименованная характеристика сущности. Его наименование должно быть уникальным для конкретного типа сущности, но может быть одинаковым для различного типа сущностей. Атрибуты используются для определения того, какая информация должна быть собрана о сущности.

В таблице 1.1 представлены сущности, определенные для них атрибуты и описание атрибутов, а также ключи.

**Таблица 1.1** – Описание сущнzостей

|  |  |  |  |
| --- | --- | --- | --- |
| **Таблица** | **Поле** | **Ключ** | **Описание** |
| Avto (автомобиль) | Avto\_Key | PK | Идентификационный номер таблицы "Avto". |
| Department\_Key | FK | Идентификационный номер таблицы "Department". |
| Brand\_Key | FK | Идентификационный номер таблицы " Brand ". |
| Namber |  | Номер автомобиля. |
| Year\_of\_release |  | Дата производства авто. |
| Color |  | Цвет авто. |
| Additional\_info |  | Вместительность авто. |
| Cancellation\_date |  | Дата списания авто. |
| Receipt\_date |  | Дата поступления авто на автопредприятие. |
| When\_cancellation |  | Дата, когда авто было списано. |
| When\_sell |  | Дата, когда авто было продано. |
| Sell\_cost |  | Стоимость продажи авто. |
| Image\_byte |  | Фотография автомобиля. |
| Department  (цех) | Department\_Key | PK | Идентификационный номер таблицы "Department". |
| Name |  | Наименование. |

Продолжение таблицы 1.1

|  |  |  |  |
| --- | --- | --- | --- |
| **Таблица** | **Поле** | **Ключ** | **Описание** |
| Brand  (Модель) | Brand\_key | PK | Идентификационный номер таблицы " Brand ". |
| Manufacturer\_key | FK | Идентификационный номер таблицы " Manufacturer ". |
| Type\_of\_avto\_key | FK | Идентификационный номер таблицы " Type\_of\_avto ". |
| Body\_type |  | Тип кузова. |
| Name |  | Название модели. |
| Expenses |  | Средний расход. |
| Division  (Распределение на маршрут) | Division\_key | PK | Идентификационный номер таблицы " Division ". |
| Avto\_worker\_key | FK | Идентификационный номер таблицы " Avto\_Worker ". |
| Ways\_key | FK | Идентификационный номер таблицы " Ways ". |
| Date\_start |  | Дата распределения на маршрут. |
| Date\_end |  | Дата снятия с маршрута. |
| Manufacturer  (Производитель) | Manufacturer\_key | PK | Идентификационный номер таблицы "Manufacturer". |
| Name |  | Наименование производителя. |
| Profession  (Профессия) | Profession\_key | PK | Идентификационный номер таблицы "Profession". |
| Name |  | Наименование профессии. |
| Repair  (Ремонт) | Repair\_Key | PK | Идентификационный номер таблицы "Repair". |
| Avto\_key | FK | Идентификационный номер таблицы "Avto". |
| Type\_of\_repair\_Key | FK | Идентификационный номер таблицы "Type\_of\_repair". |
| Work\_data\_Key | FK | Идентификационный номер таблицы "Work\_data". |
| Date |  | Дата ремонта. |

Продолжение таблицы 1.1

|  |  |  |  |
| --- | --- | --- | --- |
| **Таблица** | **Поле** | **Ключ** | **Описание** |
| Type\_of\_avto  (Тип авто) | Type\_of\_avto\_key | PK | Идентификационный номер таблицы "Type\_of\_avto". |
| Name |  | Наименование типа авто. |
| Avto\_worker  (Распределение на авто) | Avto\_worker\_key | PK | Идентификационный номер таблицы "Avto\_worker". |
| Work\_data\_Key | FK | Идентификационный номер таблицы "Work\_data". |
| Avto\_key | FK | Идентификационный номер таблицы "Avto". |
| Date\_start |  | Дата начала работы на автомобиле. |
| Date\_end |  | Дата окончания работы на автомобиле. |
| Type\_of\_repair  (Тип ремонта) | Type\_of\_repair\_Key | PK | Идентификационный номер таблицы "Type\_of\_repair". |
| Name |  | Наименование типа ремонта. |
| Cost |  | Стоимость ремонта. |
| Ways  (Маршрут) | Ways\_key | PK | Идентификационный номер таблицы "Ways". |
| Name |  | Наименование пути. |
| Lenght |  | Длинна пути. |
| Time\_in\_way |  | Время в пути. |
| Cost |  | Стоимость. |
| Start\_point |  | Начальная точка маршрута. |
| End\_point |  | Конечная точка маршрута. |
| Work\_data  (Распределение на работу) | Work\_data\_Key | PK | Идентификационный номер таблицы "Work\_data". |
| Department\_key | FK | Идентификационный номер таблицы "Department". |
| Worker\_key | FK | Идентификационный номер таблицы "Worker". |
| Profession\_key | FK | Идентификационный номер таблицы "Profession". |
| Date\_start |  | Дата начала работы. |
| Date\_end |  | Дата окончания работы. |

Окончание таблицы 1.1

|  |  |  |  |
| --- | --- | --- | --- |
| **Таблица** | **Поле** | **Ключ** | **Описание** |
| Worker  (Сотрудник) | Worker\_key | PK | Идентификационный номер таблицы "Worker". |
| Fullname |  | Полное имя сотрудника. |
| Age |  | Возраст сотрудника. |
| Namber |  | Номер сотрудника. |

* 1. Определение связей между объектами

Кроме атрибутов каждой сущности модель данных должна определять связи между сущностями. На концептуальном уровне связи представляют собой простые ассоциации между сущностями.

Связь – это ассоциирование двух или более сущностей. Если бы назначением базы данных было только хранение отдельных, не связанных между собой данных, то ее структура могла бы быть очень простой. Однако, одно из основных требований к организации базы данных – это обеспечение возможности отыскания одних сущностей по значениям других, для чего необходимо установить между ними определенные связи. А так как в реальных базах данных нередко содержатся десятки или даже сотни сущностей, то между ними может быть установлено великое множество связей. Наличие такого множества связей и определяет сложность инфологических моделей.

Для реализации информационной системы регистратуры поликлиники необходимо установить все связи между объектами. А именно, нужно рассмотреть всю информационную систему в совокупности и определить взаимное влияние объектов, составляющих систему.

Проследить отношения, в которых состоят таблицы базы данных можно по схеме, изображенной на рисунке A.1 в приложении A.

* 1. Описание полученной модели на языке инфологического проектирования

Проектирование инфологической модели предметной области – частично формализованное описание объектов предметной области в терминах некоторой семантической модели, например, в терминах ER-модели (*англ.* entity-relationship model).

По правилам построения ER-диаграмм сущность изображается в виде прямоугольника. Связь изображается линией, которая связывает две сущности, участвующие в отношении. Степень конца связи указывается графически, множественность связи изображается в виде «вилки» на конце связи. Модальность связи так же изображается графически — необязательность связи помечается кружком на конце связи. Атрибуты сущности записываются внутри прямоугольника, изображающего сущность.

Таким образом, на основании результатов предыдущих пунктов, получим ER-диаграмму, проектируемой базы данных, представленную в приложении A, рисунке А.1.

## ПОСТРОЕНИЕ СХЕМЫ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

* 1. **Построение набора необходимых отношений базы данных**

Чтобы построить схему реляционной базы данных необходимо определить совокупность отношений, которые составляют базу данных. Эта совокупность отношений будет содержать всю информацию, которая должна храниться в базе данных.

В предыдущем пункте описана инфологическая концептуальная модель базы данных «Автопредприятия», построенной с помощью IDEF1X. На основе полученной концептуальной модели можно определить набор необходимых отношений базы данных. На рисунке 2.1 представлены отношения для базы данных.

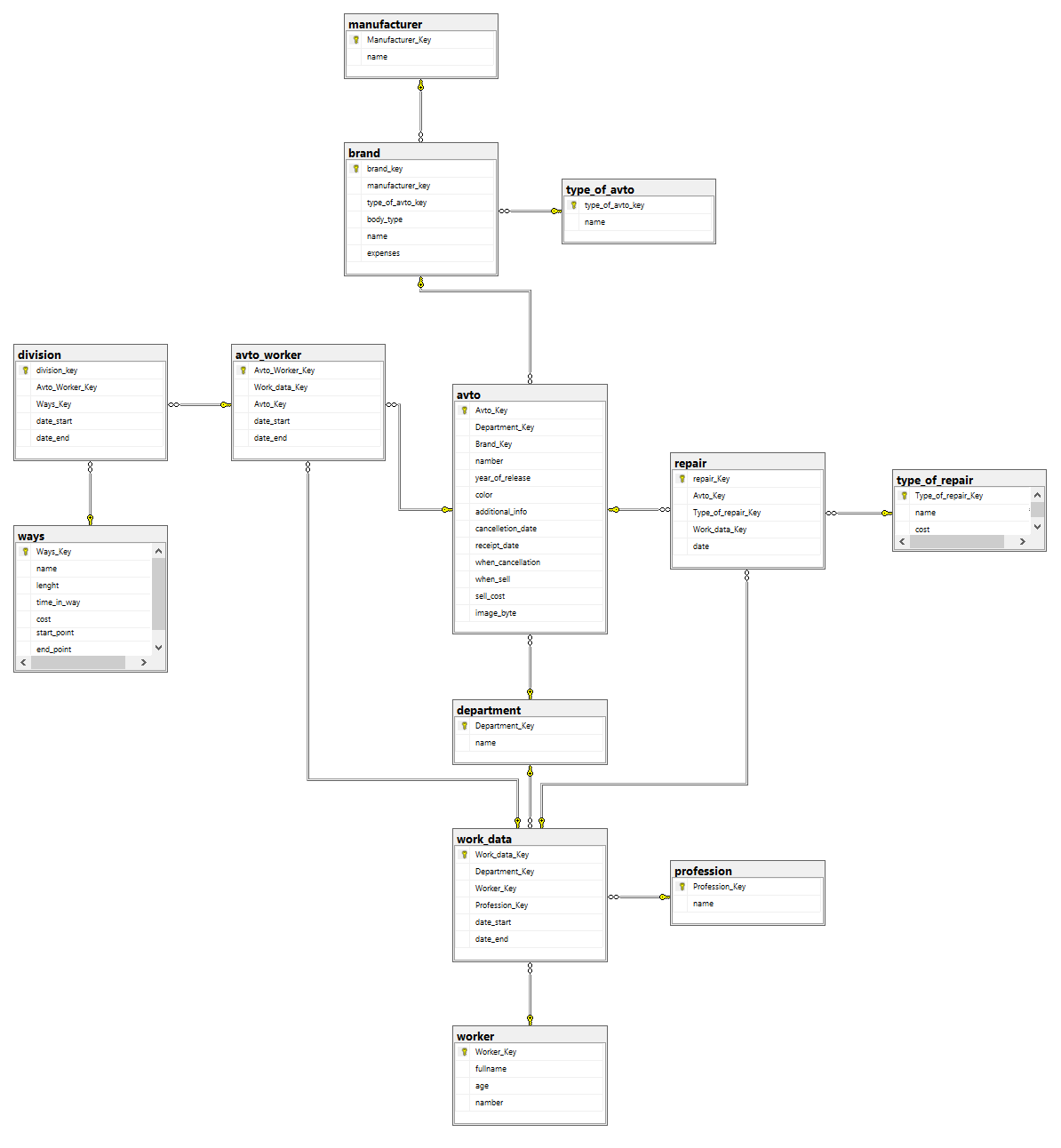


Рисунок 2.1 – Набор необходимых отношений базы данных

## **Задание первичных и внешних ключей определенных отношений**

Поле первичного ключа служит уникальным определением записи. Оно также служит для связи таблиц. В связанных таблицах первичный ключ родительской таблицы становится внешним ключом в дочерней таблице. Внешний ключ дочерней таблицы отсылает к сведениям родительской таблицы.

Первичные ключи будут иметь постфикс PK, вторичные – FK. Также в подавляющем числе случаев названия вторичных ключей будут соответствовать названию связанных таблиц (без множественного числа), при несоответствии укажем название связанной таблицы во избежание путаницы. Также следует упомянуть, что в контексте данной работы первичные ключи будут являться единственным полем в таблице.

Первичные и вторичные ключи представлены в пункте 1.2 (таблица 1.1).

В дальнейшем построении схемы реляционной базы данных ключи будут служить для организации связей между отношениями.

* 1. Третья нормальная форма

Процесс преобразования базы данных к виду, отвечающему нормальным формам, называется нормализацией. Нормализация предназначена для приведения структуры базы данных к виду, обеспечивающему минимальную избыточность, то есть нормализация не имеет целью уменьшение или увеличение производительности работы или же уменьшение, или увеличение объёма БД. Конечной целью нормализации является уменьшение потенциальной противоречивости хранимой в БД информации.

Нормальная форма — свойство [отношения](http://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_(%D1%80%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C)) в [реляционной модели данных](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), характеризующее его с точки зрения избыточности, которая потенциально может привести к логически ошибочным результатам выборки или изменения данных. Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение.

Процесс преобразования отношений базы данных к виду, отвечающему нормальным формам, называется нормализацией.

Так как все атрибуты наших отношений атомарны, а каждое отношение имеет первичный ключ, то это означает, что отношения базы находятся в первой нормальной форме (1НФ).

Так как зависимости неключевых атрибутов от части составного ключа отсутствуют (все ключи в вышеописанных отношениях несоставные), а отношения базы находятся в 1НФ, то можно утверждать, что отношения базы удовлетворяют требованиям второй нормальной формы (2НФ).

Так как в отношениях базы отсутствуют зависимости неключевых атрибутов от других неключевых атрибутов, а присутствие 2НФ указано выше, можно сказать, что отношения базы находятся в третьей нормальной форме (3НФ).

Таким образом, отношения БД находятся в 3НФ.

* 1. Определение ограничений целостности для внешних ключей отношений и для отношений в целом

Ограничение целостности отношений заключается в том, что в любом отношении должны отсутствовать записи с одним и тем же значением первичного ключа. Конкретно требование состоит в том, что любая запись любого отношения должна быть отличной от любой другой записи этого отношения. Это требование автоматически удовлетворяется, если в системе не нарушаются базовые свойства отношений.

Ограничение целостности для внешних ключей состоит в том, что значение внешнего ключа должно быть равным значению первичного ключа цели; либо быть полностью неопределенным, т.е. каждое значение атрибута, участвующего во внешнем ключе должно быть неопределенным.

Условиями целостности называется набор правил, используемых для поддержания допустимых межтабличных связей и запрета на случайное изменение или удаление связанных данных. Следует устанавливать целостность данных только при выполнении следующих условий: связываемое поле из главной таблицы является полем первичного ключа и имеет уникальный индекс, связанные поля имеют один и тот же тип данных.

Для автоматического обновления связанных полей (удаления записей) при обновлении (удалении) в главной таблице, следует устанавливать обеспечение целостности данных и каскадное обновление связанных полей (каскадное удаление связанных записей).

Ограничение целостности, накладываемые на разрабатываемую систему:

* ключевое поле отношения должно быть уникальным;
* внешний ключ должен быть повторяющимся, то есть соответствовать уникальному ключу в своем отношении.

Для удовлетворения требования ограничения целостности для внешних ключей отношений и для отношений в целом необходимо, чтобы выполнялось соответствие между типами вводимых данных и типами столбцов в таблицах, а также чтобы были заполнены все обязательные поля в таблицах, т.е. те поля которые не могут содержать значения NULL.

* 1. Графическое представление связей между внешними и первичными ключами

По результатам нормализации, определении первичных и внешних ключей, связей между сущностями, была получена схема реляционной базы данных, представленная в приложении Б на рисунке Б.1. На ней изображаются все отношения базы данных, а также связей между внешними и первичными ключами. Первичные ключи обозначаются буквами PK (*от англ.* Primary Key – первичный ключ), внешние ключи обозначаются FK (*от англ.* Foreign Key – внешний ключ).

## СОЗДАНИЕ СПРОЕКТИРОВАННОЙ БАЗЫ ДАННЫХ

Для реализации спроектированной базы данных была выбрана система управления базами данных MS SQL Server Express 2016. Это обусловлено тем, что, во-первых, данная СУБД получилась широкое распространение, а во-вторых, имеются свободно распространяемые сборки, а в-третьих, Microsoft и другие компании производят большое число программных средств разработки, позволяющих разрабатывать бизнес-приложения с использованием баз данных Microsoft SQL Server. Но так как это бесплатная версия СУБД в ней не имеется важная для нас функция «Планировщика задач».

Произведем описание структуру каждой из таблиц с описанием типа полей.

Таблица *Avto* хранит данные автомобилей. Её структура приведена в таблице 3.1.

Таблица 3.1 – Характеристика атрибутов таблицы *Avto*.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| Avto\_Key | INT | Идентификационный номер таблицы "Avto". |
| Department\_Key | INT | Идентификационный номер таблицы "Department". |
| Brand\_Key | INT | Идентификационный номер таблицы " Brand ". |
| Namber | VARCHAR | Номер автомобиля. |
| Year\_of\_release | DATE | Дата производства авто. |
| Color | VARCHAR | Цвет авто. |
| Additional\_info | INT | Вместительность авто. |
| Cancellation\_date | DATE | Дата списания авто. |
| Receipt\_date | DATE | Дата поступления авто на автопредприятие. |
| When\_cancellation | DATE | Дата, когда авто было списано. |
| When\_sell | DATE | Дата, когда авто было продано. |
| Sell\_cost | FLOAT | Стоимость продажи авто. |
| Image\_byte | IMAGE | Фотография автомобиля. |

Таблица *Department* содержит наименование цехов. Её структура приведена в таблице 3.2.

Таблица 3.2 – Характеристика атрибутов таблицы *Department*.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| Department\_Key | INT | Идентификационный номер таблицы "Department". |
| Name | VARCHAR | Наименование. |

Таблица *Brand* содержит модели транспортных средств. Её структура приведена в таблице 3.3.

**Таблица 3.3 –** Характеристика атрибутов таблицы *Brand*.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| Brand\_key | INT | Идентификационный номер таблицы " Brand ". |
| Manufacturer\_key | INT | Идентификационный номер таблицы " Manufacturer ". |
| Type\_of\_avto\_key | INT | Идентификационный номер таблицы " Type\_of\_avto ". |
| Body\_type | VARCHAR | Тип кузова. |
| Name | VARCHAR | Название модели. |
| Expenses | FLOAT | Средний расход. |

Таблица *Division* содержит сведения о распределении на маршрут. Её структура приведена в таблице 3.4.

**Таблица 3.4 –** Характеристика атрибутов таблицы *Division*.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| Division\_key | INT | Идентификационный номер таблицы " Division ". |
| Avto\_worker\_key | INT | Идентификационный номер таблицы " Avto\_Worker ". |
| Ways\_key | INT | Идентификационный номер таблицы " Ways ". |
| Date\_start | DATE | Дата распределения на маршрут. |
| Date\_end | DATE | Дата снятия с маршрута. |

Таблица *Manufacturer* содержит список производителей. Её структура приведена в таблице 3.5.

**Таблица 3.5 –** Характеристика атрибутов таблицы *Manufacturer*.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| Manufacturer\_key | INT | Идентификационный номер таблицы "Manufacturer". |
| Name | VARCHAR | Наименование производителя. |

Таблица *Profession* содержит список профессий. Её структура приведена в таблице 3.6.

**Таблица 3.6 –** Характеристика атрибутов таблицы *Profession*.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| Profession\_key | INT | Идентификационный номер таблицы "Profession". |
| Name | VARCHAR | Наименование профессии. |

Таблица *Repair* содержит информацию о ремонте автомобиля. Её структура приведена в таблице 3.7.

**Таблица 3.7** – Характеристика атрибутов таблицы *Repair*.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| Repair\_Key | INT | Идентификационный номер таблицы "Repair". |
| Avto\_key | INT | Идентификационный номер таблицы "Avto". |
| Type\_of\_repair\_Key | INT | Идентификационный номер таблицы "Type\_of\_repair". |
| Work\_data\_Key | INT | Идентификационный номер таблицы "Work\_data". |
| Date | DATE | Дата ремонта. |

Таблица *Type\_of\_avto* содержит список типов автомобилей. Её структура приведена в таблице 3.8.

**Таблица 3.8 –** Характеристика атрибутов таблицы *Type\_of\_avto* .

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| Type\_of\_avto\_key | INT | Идентификационный номер таблицы "Type\_of\_avto". |
| Name | VARCHAR | Наименование типа авто. |

Таблица *Avto\_worker* содержит информацию о начале и окончании работы сотрудника на автомобиле. Её структура приведена в таблице 3.9.

**Таблица 3.9** – Характеристика атрибутов таблицы *Avto\_worker*.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| Avto\_worker\_key | INT | Идентификационный номер таблицы "Avto\_worker". |
| Work\_data\_Key | INT | Идентификационный номер таблицы "Work\_data". |
| Avto\_key | INT | Идентификационный номер таблицы "Avto". |
| Date\_start | DATE | Дата начала работы на автомобиле. |
| Date\_end | DATE | Дата окончания работы на автомобиле. |

Таблица *Type\_of\_repair* содержит информациб о видах ремонта. Её структура приведена в таблице 3.10.

**Таблица 3.10** – Характеристика атрибутов таблицы *Type\_of\_repair* .

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| Type\_of\_repair\_Key | INT | Идентификационный номер таблицы "Type\_of\_repair". |
| Name | VARCHAR | Наименование типа ремонта. |
| Cost | FLOAT | Стоимость ремонта. |

Таблица *Ways* содержит сведения о маршрутах. Её структура приведена в таблице 3.11.

**Таблица 3.11** – Характеристика атрибутов таблицы *Ways*.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| Ways\_key | INT | Идентификационный номер таблицы "Ways". |
| Name | VARCHAR | Наименование пути. |
| Lenght | FLOAT | Длинна пути. |
| Time\_in\_way | TIME | Время в пути. |
| Cost | FLOAT | Стоимость. |
| Start\_point | VARCHAR | Начальная точка маршрута. |
| End\_point | VARCHAR | Конечная точка маршрута. |

Таблица *Work\_data* хранит информацию о распределении на маршруты. Её структура приведена в таблице 3.12.

**Таблица 3.12** – Характеристика атрибутов таблицы *Work\_data*.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| Work\_data\_Key | INT | Идентификационный номер таблицы "Work\_data". |
| Department\_key | INT | Идентификационный номер таблицы "Department". |
| Worker\_key | INT | Идентификационный номер таблицы "Worker". |
| Profession\_key | INT | Идентификационный номер таблицы "Profession". |
| Date\_start | DATE | Дата начала работы. |
| Date\_end | DATE | Дата окончания работы. |

Таблица *Worker* хранит в себе названия улиц, в которых проживают поциенты. Её структура приведена в таблице 3.13.

**Таблица 3.13** **–** Характеристика атрибутов таблицы *Worker*.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| Worker\_key | INT | Идентификационный номер таблицы "Worker". |
| Fullname | VARCHAR | Полное имя сотрудника. |
| Age | INT | Возраст сотрудника. |
| Namber | VARCHAR | Номер сотрудника. |

## 4 ЗАПИСЬ ВЫРАЖЕНИЙ, УКАЗАННЫХ В ВАРИАНТЕ ЗАДАНИЯ ТИПОВ ЗАПРОСОВ НА ЯЗЫКЕ SQL

1. Функция получения перечня и общее количество по предприятию, по указанной автомашине.
2. create function dbo.Hello1(@avtokey int)
3. returns table
4. as
5. return(select a.Avto\_Key, d.fullname, d.age, d.namber from avto a, avto\_worker b, work\_data c, worker d where a.Avto\_Key=b.Avto\_Key and b.Work\_data\_Key=c.Work\_data\_Key and c.Worker\_Key=d.Worker\_Key and b.Avto\_Key=@avtokey);
6. Запрос получает список распределения водителей по автомобилям.
7. select a.Avto\_Key, d.fullname, d.age, d.namber from avto a, avto\_worker b, work\_data c, worker d where a.Avto\_Key=b.Avto\_Key and b.Work\_data\_Key=c.Work\_data\_Key and c.Worker\_Key=d.Worker\_Key;
8. Запрос получает список распределения пассажирского автотранспорта по маршрутам.
9. select a.namber, c.name, n.name from avto a, brand b, type\_of\_avto c, avto\_worker d, division m, ways n where a.Brand\_Key=b.brand\_key and b.type\_of\_avto\_key=c.type\_of\_avto\_key and a.Avto\_Key=d.Avto\_Key and d.Avto\_Worker\_Key=m.Avto\_Worker\_Key and m.Ways\_Key=n.Ways\_Key and c.name IN('Car','Taxi','Route taxi', 'Bus');
10. Функция, получающая данные о числе ремонтов и их стоимость для автотранспорта определенной категории, отдельной марки автотранспорта или указанной автомашины за период времени.
11. create procedure zad5
12. @datestart date,
13. @dateend date,
14. @avtoname varchar(20),
15. @avtotype varchar(20),
16. @avtoman varchar(20)
17. as
18. if(@avtoname!=null)
19. begin
20. select count(b.Avto\_key), sum(c.cost) from avto a, repair b, type\_of\_repair c, brand d, type\_of\_avto m
21. where a.Avto\_Key=b.Avto\_Key and b.Type\_of\_repair\_Key=c.Type\_of\_repair\_Key and a.Brand\_Key=d.brand\_key and d.type\_of\_avto\_key=m.type\_of\_avto\_key
22. and (b.date>=@datestart or b.date<=@dateend) and a.Avto\_Key=@avtoname;
23. end;
24. else if(@avtotype!=null)
25. begin
26. select count(b.Avto\_key), sum(c.cost) from avto a, repair b, type\_of\_repair c, brand d, type\_of\_avto m
27. where a.Avto\_Key=b.Avto\_Key and b.Type\_of\_repair\_Key=c.Type\_of\_repair\_Key and a.Brand\_Key=d.brand\_key and d.type\_of\_avto\_key=m.type\_of\_avto\_key
28. and (b.date>=@datestart or b.date<=@dateend) and d.name=@avtotype;
29. end;
30. else if(@avtotype!=null)
31. begin
32. select count(b.Avto\_key), sum(c.cost) from avto a, repair b, type\_of\_repair c, brand d, type\_of\_avto m, manufacturer k
33. where a.Avto\_Key=b.Avto\_Key and b.Type\_of\_repair\_Key=c.Type\_of\_repair\_Key and a.Brand\_Key=d.brand\_key and d.type\_of\_avto\_key=m.type\_of\_avto\_key and k.Manufacturer\_Key=d.manufacturer\_key
34. and (b.date>=@datestart or b.date<=@dateend) and k.name=@avtoman;
35. end;
36. Запрос получает данные о распределении автотранспорта на предприятии.
37. select a.namber, b.name from avto a, department b where a.Department\_Key=b.Department\_Key
38. Запрос получает данные о грузоперевозках, выполненных указанной автомашиной за указанный период времени.
39. select a.namber, d.name, d.lenght, d.start\_point, d.end\_point, d.lenght, d.time\_in\_way, d.cost from avto a, avto\_worker b, division c, ways d where a.Avto\_Key=b.Avto\_Key and b.Avto\_Worker\_Key=c.Avto\_Worker\_Key and c.Ways\_Key=d.Ways\_Key;

Прочие запросы и процедуры реализованы с помощью Linq-запросов, ввиду актуальности и скорости реализации кода.

## 5 ВЫБОР И ОСНОВАНИЕ СРЕДСТВ РАЗРАБОТКИ ПРИЛОЖЕНИЯ

Выбор СУБД является сложной задачей и должен основываться, в первую очередь, на потребностях с точки зрения информационной системы и пользователей. Определяющими здесь являются вид программного продукта и категория пользователей (или профессиональные программисты, или конечные пользователи, или и то, и другое). Другими показателями, влияющими на выбор СУБД, являются [2]:

* + удобство и простота использования;
  + качество средств разработки, защиты и контроля базы данных;
  + уровень коммуникационных средств в случае применения ее в сетях;
  + фирма-разработчик;
  + стоимость.

Система MsSQL Server позволяет обращаться к данным из любого приложения, разработанного с применением технологий Microsoft .NET и Visual Studio. MsSQL Server обеспечивает высочайший уровень безопасности, надежности и масштабируемости для критически важных приложений. Чтобы использовать новые возможности, постоянно возникающие в быстро меняющемся деловом мире, предприятиям нужно быть способными быстро создавать и развертывать решения, управляемые данными. MsSQL Server позволяет сократить затраты времени и средств, требуемые на управление и развертывание таких приложений. Также следует учесть, что фирма-разработчик данной СУБД является также разработчиком самой распространенной ОС. В финансовом плане важным фактором является то, что существуют бесплатные сборки данной СУБД (Express).

Для реализации приложения была выбрана среда разработки Microsoft Visual Studio 2019, в качестве языка программирования – C#.

Достоинства платформы .NET [1]:

1) Вся платформа .NET основана на единой объектно-ориентированной модели. Все сервисы, интерфейсы и объекты, которые платформа предоставляет разработчику объединены в единую иерархию классов. Другими словами, все, что может вам потребоваться при создании приложений под платформу .NET будет всегда у вас под рукой. Причем, все это сгруппировано очень удобно и интуитивно понятно.

2) Приложение, написанное на любом .NET-совместимом языке, является межплатформенным. Дело в том, что приложение, написанное, скажем, на том же C#, не зависит от платформы, на которой будет выполняться, но зато зависит от наличия платформы .NET.

3) В состав платформы .NET входит "сборщик мусора", который освобождает ресурсы. Таким образом, приложения защищены от утечки памяти и от необходимости освобождать ресурсы. Это делает программирование более легким и более безопасным.

6) Приложения .NET используют безопасные типы, что повышает их надежность и совместимость.

7) .NET приложения могут быть сертифицированы на безопасность. Это является особенность промежуточного кода, в который преобразуются все .NET приложения.

8) Абсолютно все ошибки обрабатываются механизмом исключительных ситуаций. Это позволяет избежать разногласий, которые иногда возникают при программировании под Win32.

9) Повторное использование кода стало еще удобнее. Это связано с тем, что промежуточный язык MSIL не зависит от языка программирования. Например, вы можете написать программу на C#, а патч к ней писать уже, скажем, на J#.

Для разработки пользовательского интерфейса выбран фреймворк Bootstrap 4.

**Bootstrap — это открытый и бесплатный HTML, CSS и JS фреймворк, который используется веб-разработчиками для быстрой вёрстки адаптивных дизайнов сайтов и веб-приложений.**

Фреймворк Bootstrap используется по всему миру не только независимыми разработчиками, но иногда и целыми компаниями.

Основная область его применения – это разработка frontend сайтов и интерфейсов админок. Среди аналогичных систем (Foundation, UIkit, Semantic UI, InK и др.) фреймворк Bootstrap является самым популярным [4].

Основные преимущества Bootstrap:

* Адаптивность (mobile first), высокая скорость и оптимизация, стандартизация интерфейсов — динамичные макеты Bootstrap качественно отображаются на самых разных устройствах без необходимости внесения изменений в разметку.
* Дизайн — единые шаблоны и стилевое оформление элементов макета и всех страниц на сайте в целом. И при этом Bootstrap кросс-браузерный и хорошо отображается во всех браузерах Safari, Firefox, IE, EDGE и тех, что на основе Chromium (движок Blink на основе Webkit: Яндекс.Браузер, Опера, Гугл Хром). Регулярное обновление и дополнение фреймворка самыми современными возможностями HTML и CSS вносит некоторые ограничения в использовании с IE7 и IE8 — не забудьте проверить.
* Простота и открытость — использовать Bootstrap настолько просто, что с ним справляются даже школьники и начинающие веб-разработчики, а открытый исходный код позволяет самому участвовать в разработке, модифицировать под свои нужды или просто пользоваться хорошим бесплатным решением.

При этом код HTML, JavaScript и CSS в Bootstrap продуман и рассмотрен под микроскопом сотнями разработчиков со всего мира — все для того, чтобы рядовые вебмастера и верстальщики могли легко и просто настроить сетку сайта или встроить необходимые элементы в интерфейс.

Также, в Bootstrap используется динамический язык стилей LESS, которые расширяет возможности CSS: разработчики могут управлять цветами, создавать вложенные колонки и переменные [5].

## 6 РЕАЛИЗАЦИЯ ЗАКОНЧЕННОГО ПРИЛОЖЕНИЯ, РАБОТАЮЩЕГО С СОЗДАННОЙ БАЗОЙ ДАННЫХ

* 1. **Разработка и построение интерфейса главной и рабочих форм**

Прежде всего пользователь заходя на сайт попадает на главную страницу, на которой расположен перечень автмашин по сей день работающих на автопредприятии.

Главная страница представлена виде страницы сайта с возможностью перехода по представленным автомашинам и просмотру полной информации об авто, также на главной странице находиться ссылки для выполнения данного действия.

Все основные формы и виды выполнены в отдельных страницах сайта с собственным функционалом и с использованием интерактивных полей ввода для взаимодействия с пользователем.

При проектировании приложения были учтены все возможные случаи некорректной работы программы, поэтому большинство нештатных ситуаций сопровождается оповещениями с описанием проблемы.

Скриншоты главной и основных страниц представлены в приложении В.

* 1. Построение главного меню и кнопок панели инструментов

Навигационная панель сайта представлена шестью пунктами: Home, Avto, Brand, Worker, Way, Repair. Данные пункты выполнены виде навигационной панели сайта, с использование выпадающего списка.

* 1. **Выполнение программного кода на Microsoft Visual C#**

Опишем работу приложения с базой данных. Все необходимые интерфейсы для работы с базами данных находятся в классе *Startup*. Подключение к базе данных начинается с формирования строки подключения представленной в файле *appsettings.json* и последующим созданием контекста на основе данной строки:

**Листинг 6.1** – Листинг файла *appsettings.json*

1. "ConnectionStrings":
2. {
3. "DefaultConnection":
4. "Server=(localdb)\\MSSQLLocalDB;Database=KursachTry3;Trusted\_Connection=True;MultipleActiveResultSets=true"
5. }

**Листинг 6.2** – Создание контекста на основе строки подключения

1. public void ConfigureServices(IServiceCollection services)
2. {
3. string connection = Configuration.GetConnectionString("DefaultConnection");
4. services.AddDbContext<BDModel.AvtoModel>(options => options.UseSqlServer(connection));
5. services.Configure<CookiePolicyOptions>(options =>
6. {
7. // This lambda determines whether user consent for non-essential cookies is needed for a given request.
8. options.CheckConsentNeeded = context => true;
9. options.MinimumSameSitePolicy = SameSiteMode.None;
10. });
11. services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version\_2\_1);}

Все запросы по работе с базой данных обращаются к контексту и строятся с помощью LINQ to Entity [6]. В следствии чего упрощается построение запросов, вызов процедур и фильтрация результатов.

Хранение и обработку настроек приложения обеспечивает класс *AppSettings*, который позволяет загружать, сохранять и изменять настройки программы.

## ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была реализована база данных информационной системы автотранспортного предприятия, а также сайт, а также эффективно работающее с этой базой данных приложение. Программный продукт реализован при помощи языков разметки HTML, CSS с использованием фреймворка Bootstrap, языка программирования C# с использованием платформы .NET и фреймворка Entity.

Разработанная база данных удовлетворяет всем требованиям, предъявленным в задании, и позволяет без проблем хранить и извлекать требуемую информацию.

Созданный сайт позволяет упростить работу с информаций для работников автотранспортного предприятия, позволяет систематизировать всю необходимую информацию. Также приложение упрощает операции по удалению, изменению, добавлению данных.

В процессе выполнения данной курсовой работы были закреплены навыки в программировании на ASP.NET, проектировании баз данных и реализации их в MS SQL Server 2016.

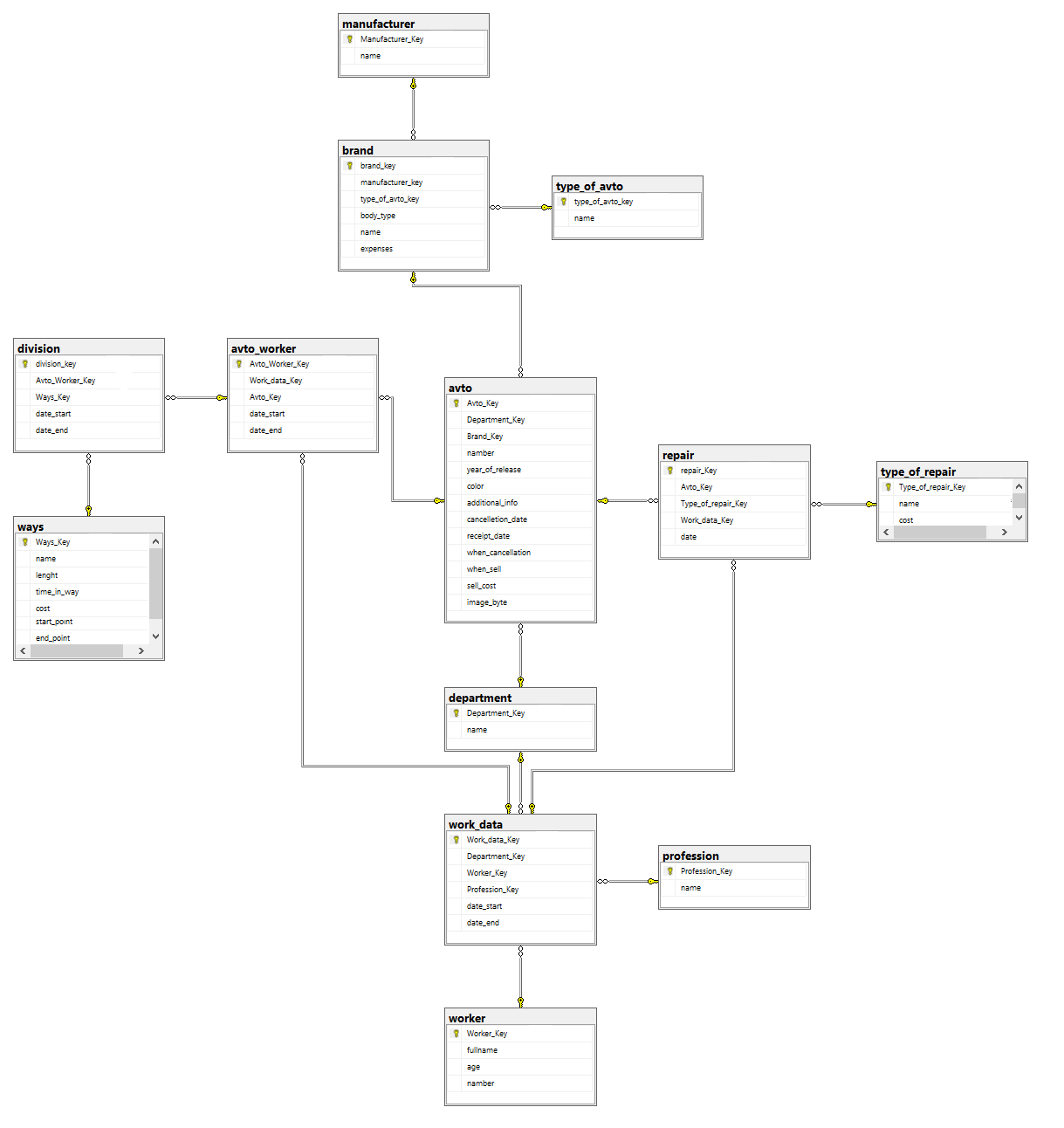
## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Data flow diagram - Wikipedia, the free encyclopedia / Многоязычная общедоступная свободно распространяемая энциклопедия, публикуемая в Интернете Википедия. Режим доступа: http://en.wikipedia.org/wiki/Data\_flow\_diagram; Дата доступа: 23.09.2019
2. Entity-relationship model – Wikipedia, the free encyclopedia / Многоязычная общедоступная свободно распространяемая энциклопедия, публикуемая в Интернете Википедия. Режим доступа: http://en.wikipedia.org/wiki/Entity-relationship\_model; Дата доступа: 23.09.2019
3. MySQL 5.0 Reference Manual [Электронный ресурс] / Официальный сайт MySQL. Документация по MySQL. Режим доступа: http://dev.mysql.com/doc/refman/5.0/en/index.html; Дата доступа: 23.09.2019
4. Что такое Bootstrap и зачем он нужен? [Электронный ресурс] / Ресурс изучения технологий Bootstrap, HTML, CSS, JavaScript и др.: https://itchief.ru/bootstrap/introduction; Дата доступа: 23.09.2019
5. Что такое Bootstrap? [Электронный ресурс] / Интернет работа: https://blogwork.ru/chto-takoe-bootstrap; Дата доступа: 23.09.2019
6. Раттц Д. LINQ язык интегрированных запросов в C# 2008 для профессионалов. Пер. с англ. – М.: Вильямс, 2008. – 645с.: ил.; Дата доступа: 23.09.2019

## ПРИЛОЖЕНИЕ А

(Обязательное)

КОНЦЕПТУАЛЬНАЯ СХЕМА БД



**Рисунок A.1** – ER-диаграмма проектируемой базы данных

## ПРИЛОЖЕНИЕ Б

(Обязательное)

## СХЕМА РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

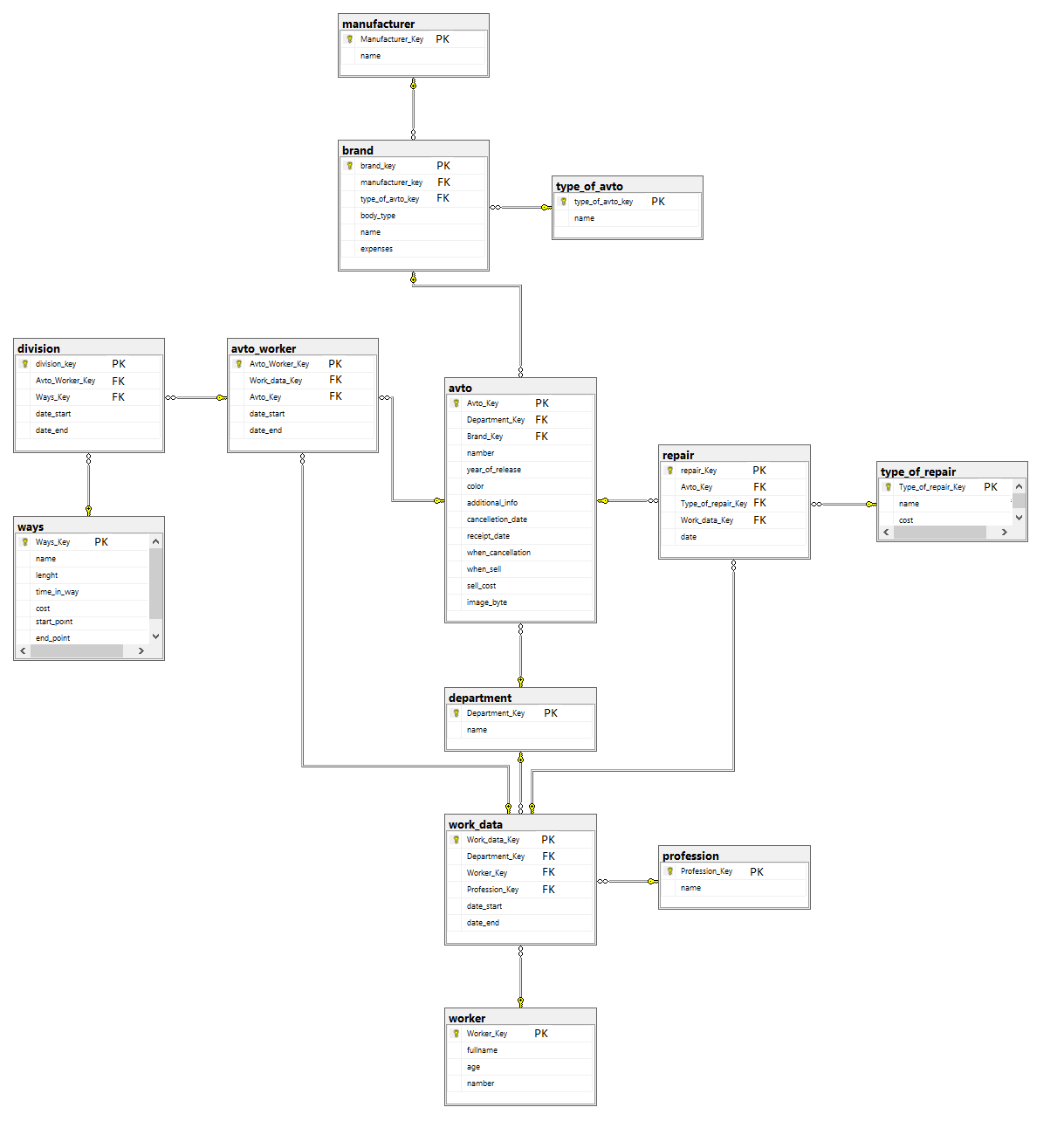
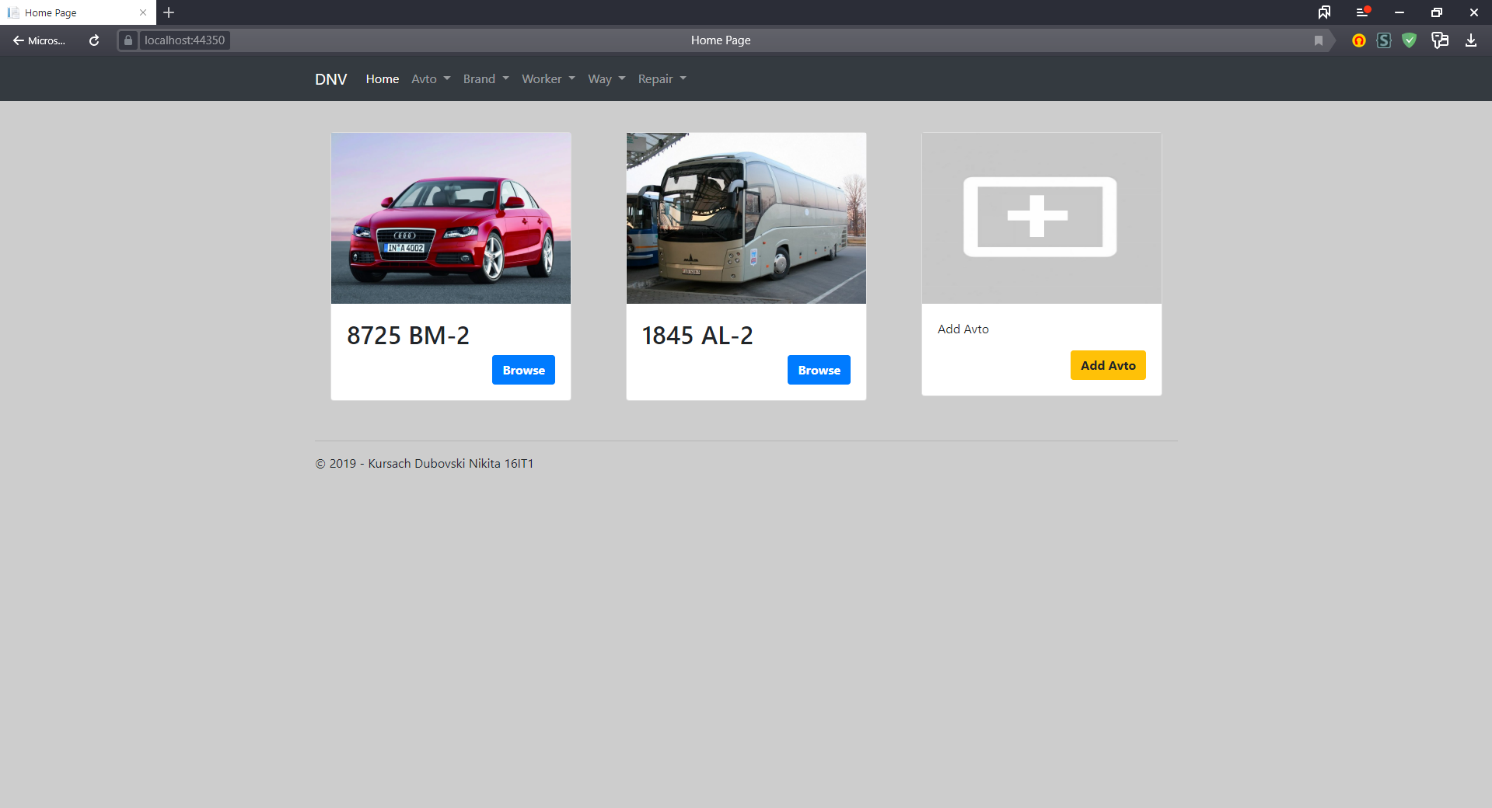
****

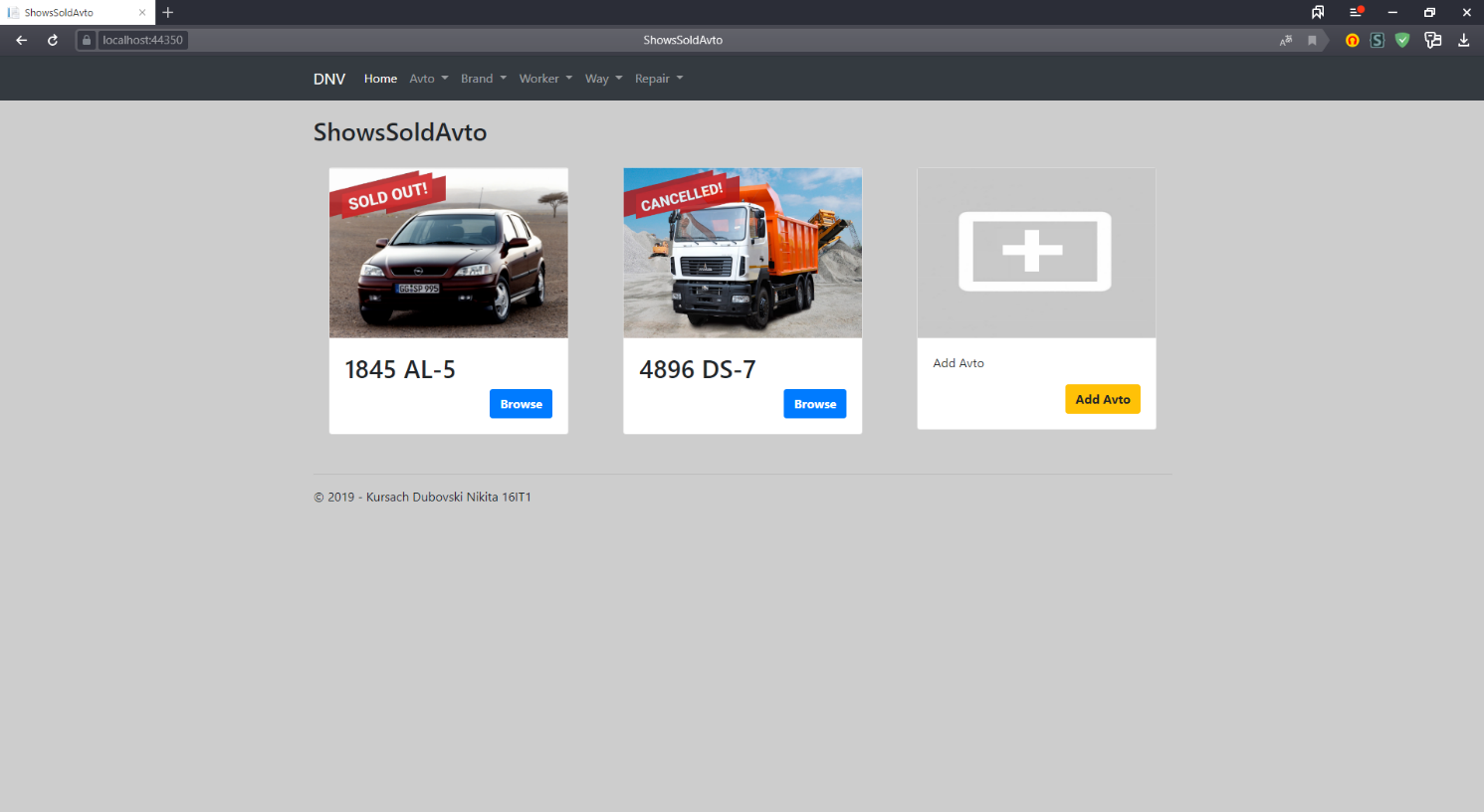
Рисунок Б.1 – Схема реляционной базы данных

## ПРИЛОЖЕНИЕ В

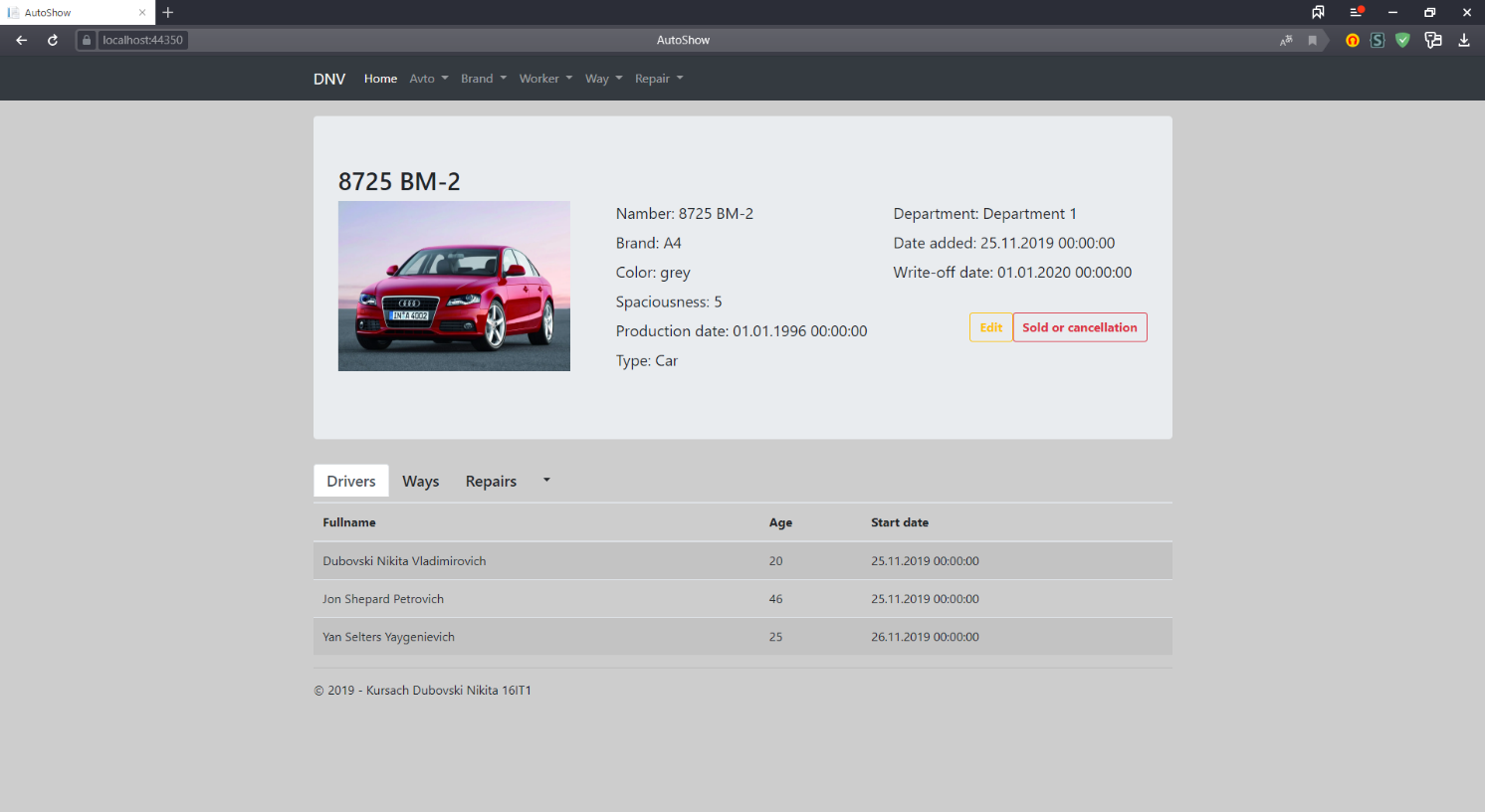
## ГЛАВНАЯ И РАБОЧИЕ ФОРМЫ ПРИЛОЖЕНИЯ



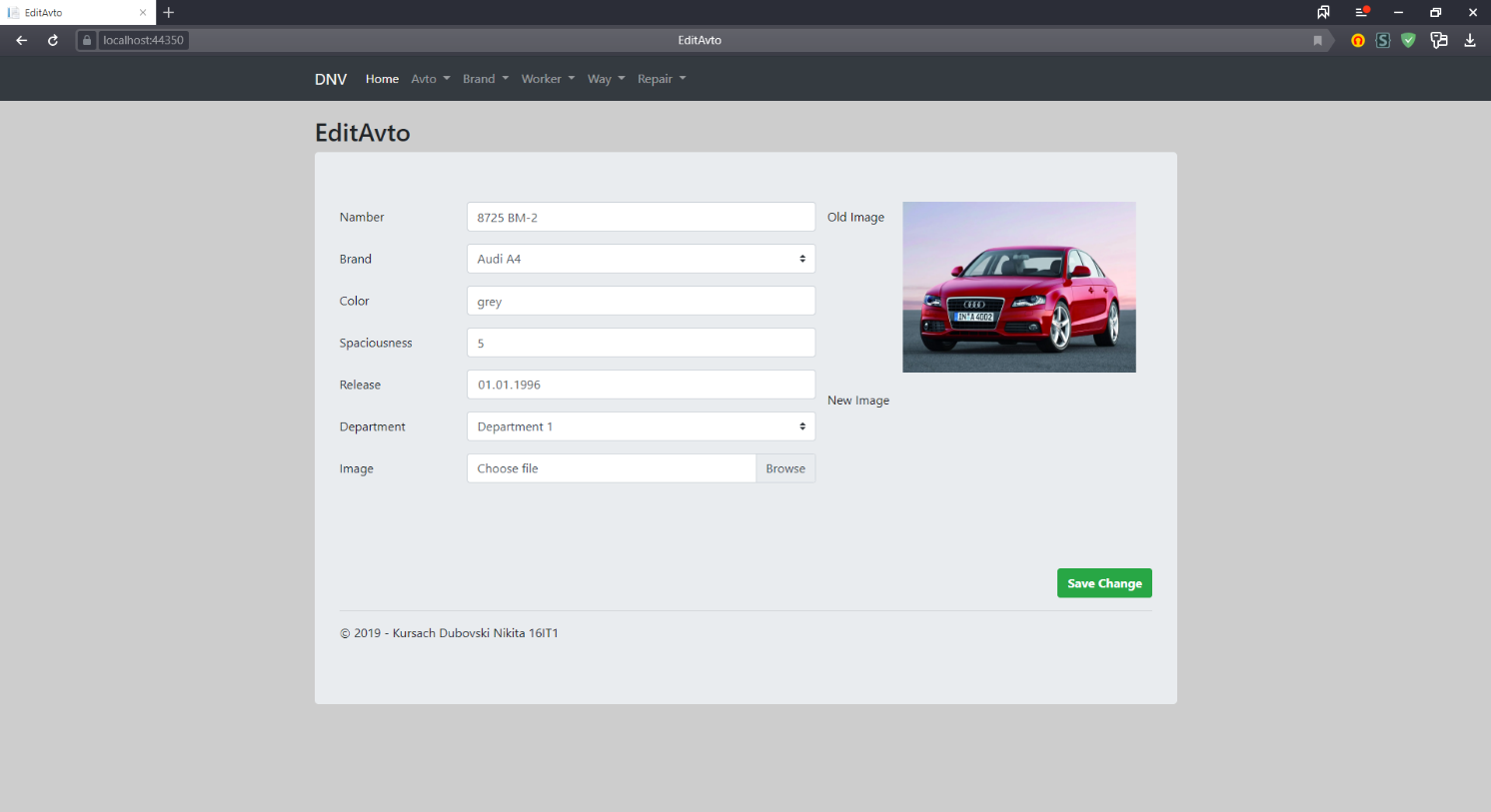
**Рисунок В.1 –** Главная страница

****

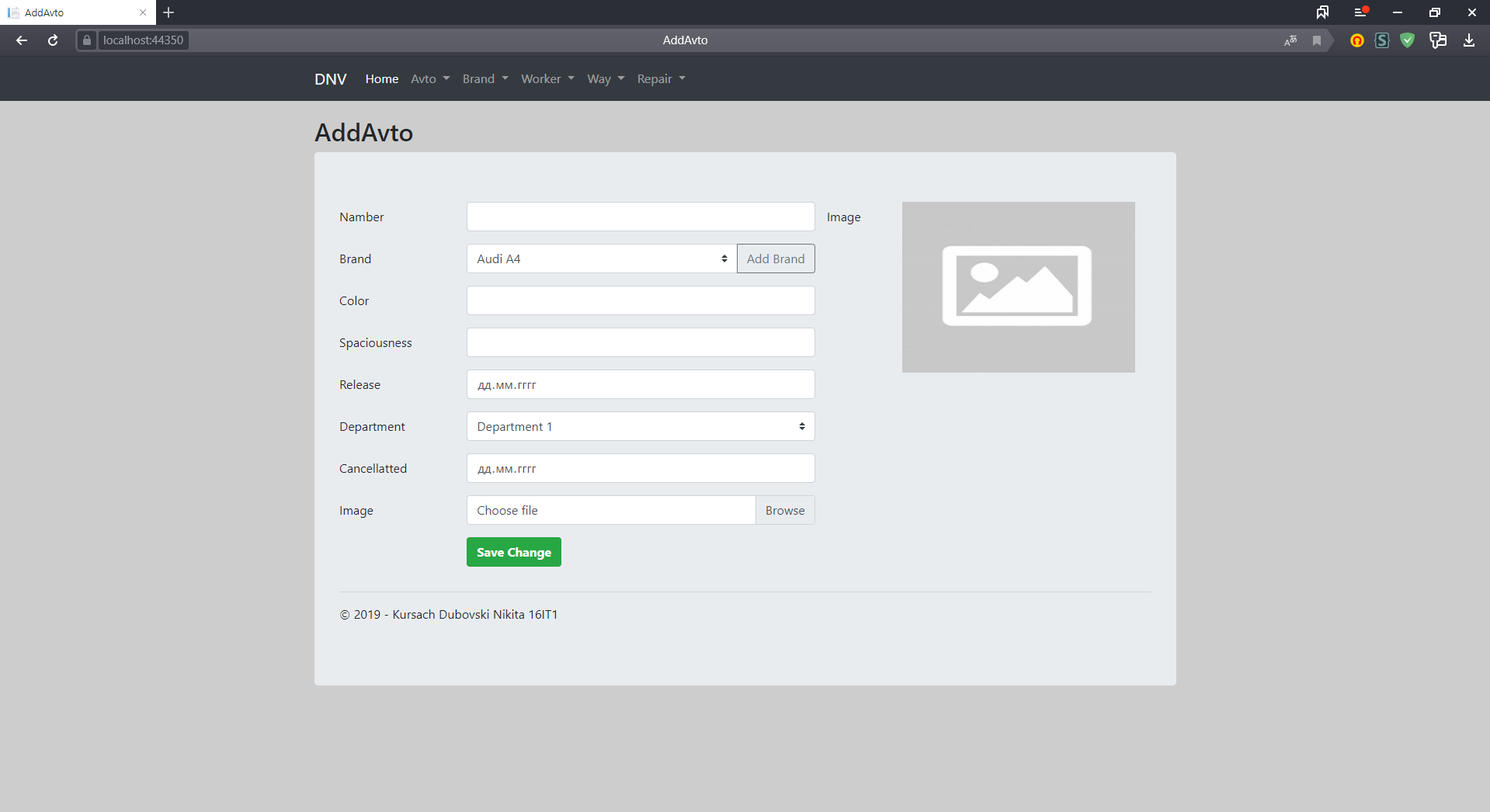
**Рисунок В.2 –** Страница списанных или проданных автомашин

****

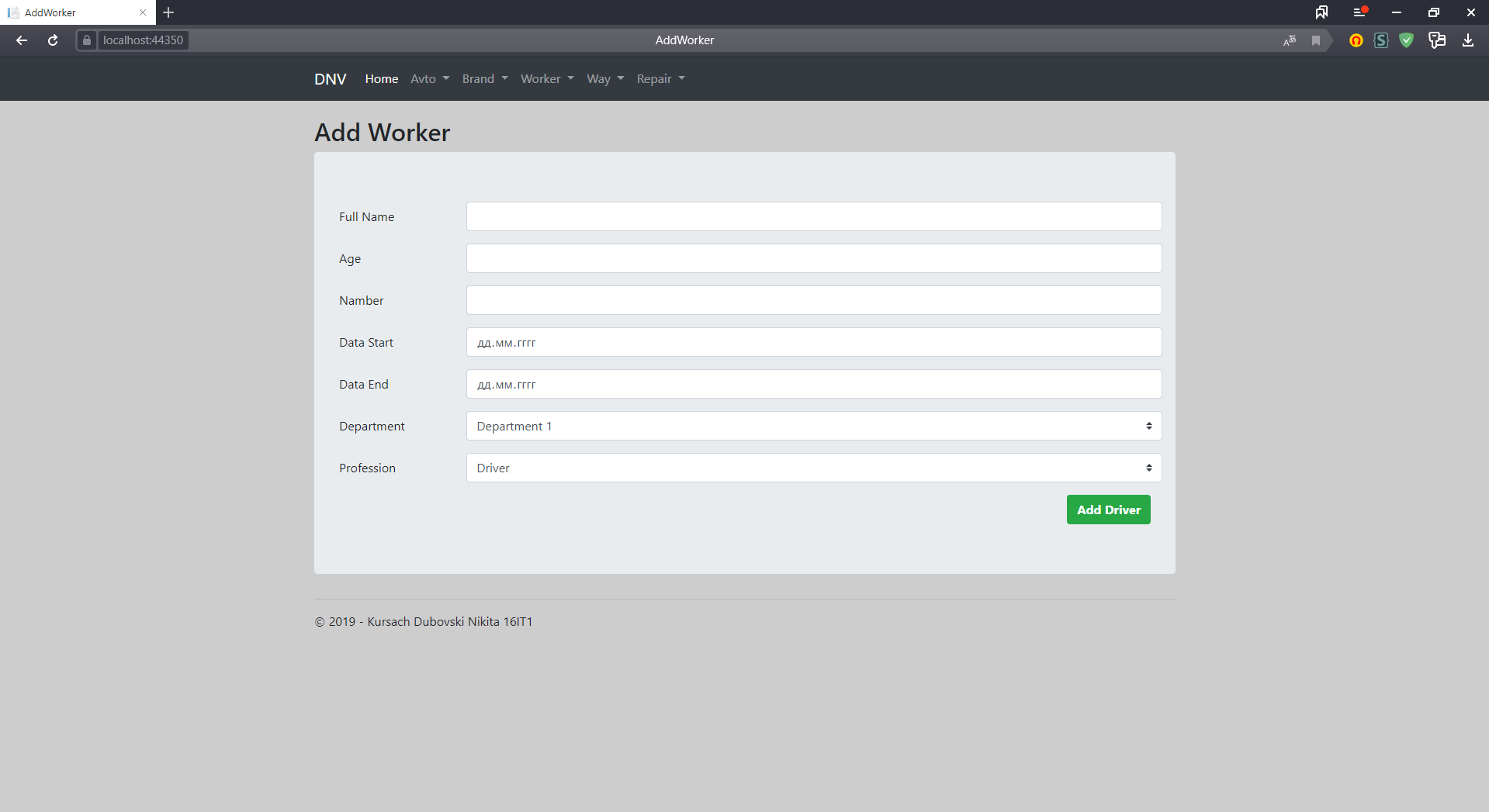
**Рисунок В.3 –** Просмотр информации об автомашине

****

**Рисунок В.4 –** Редактирование информации автомашины

****

**Рисунок В.5 –** Добавление автомашины

****

**Рисунок В.6 –** Добавление сотрудника

## ПРИЛОЖЕНИЕ Г

## ЛИСТИНГ ПРОГРАММЫ