Министерство образования и науки Российской Федерации Федеральное государственное автономное образовательное   
учреждение высшего образования   
«Российский государственный профессионально-педагогический университет»   
Институт инженерно-педагогического образования

Кафедра информационных систем и технологий

**КУРСОВАЯ РАБОТА**

**по дисциплине: «Объектно-ориентированные технологии»**

**Вариант 6**

**Выполнил:**  Парфенов Н.А.

**Группа:**  ИТк-301

**Проверил:**

к. п. н. Толстова Н.С.

Екатеринбург 2020

**ОГЛАВЛЕНИЕ**

[Задание на курсовую работу 3](#_Toc41995602)

[Введение 4](#_Toc41995603)

[1 Аналитическая часть (проектная часть) 5](#_Toc41995604)

[1.1 Характеристика предметной области 5](#_Toc41995605)

[1.2 Постановка задачи 5](#_Toc41995606)

[1.3 Проектирование Windows-приложения 6](#_Toc41995607)

[1.4 Проектирование базы данных 11](#_Toc41995608)

[2 Практическая часть 13](#_Toc41995609)

[2.1 Разработка базы данных 13](#_Toc41995610)

[2.2 Описание Windows-приложения 17](#_Toc41995611)

# Задание на курсовую работу

Требуется разработать программную систему, предназначенную для диспетчера автобусного парка. Такая система должна обеспечивать хранение сведений о водителях, о маршрутах и характеристиках автобусов.

Каждый водитель характеризуется паспортными данными, классом, стажем работы и окладом, причем оклад зависит от класса и стажа работы. Маршрут автобуса характеризуется номером маршрута, названием начального и конечного пункта движения, временем начала и конца движения, интервалом движения и протяженностью в минутах (время движения по всему маршруту). Характеристиками автобуса являются: его тип, вместимость и государственный номерной знак. Каждый водитель закреплен за отдельным автобусом и работает на определенном маршруте, но в случае поломки своего автобуса или болезни другого водителя может пересесть на другую машину. В базе должен храниться график работы водителей в виде массива дней выхода на работу. Необходимо предусмотреть возможность корректировки списков в случаях поступления на работу нового водителя, списания старого автобуса, введения нового маршрута или изменения старого и т.п.

# Введение

Транспортная сеть это большая, живая и сложная система, без которой невозможно представить современное общество.

Бумажный документооборот в подобной структуре — это непозволительный атавизм.

OS Windows на данный момент является самой популярной операционной системой. Электронный документооборот должен быть реализован именно на данной системе.

Цельюкурсовой работы является закрепление материала, пройденного на предмете «Объектно-ориентированные технологии»

Задача курсовой работы — разработать Windows приложение для работы с внешними источниками данных (БД) на языке высокоуровневого программирования С# в многоязычной интегрированной среде Microsoft Visual Studio решающее проблему бумажного документооборота в сфере транспортных систем.

В процессе разработки были использованы:

* Microsoft Visual Studio для разработки Windows Forms приложений на языке программирования C#,
* Microsoft Visual Studio Code для разработки html шаблонов для отчетов приложения.
* NoSQL Booster for MongoDB в качестве СУБД для разработки базы данных.

Разработанное Windows приложение на практике может использоваться диспетчерами автобусных парков для учета персонала, автопарка и контроля маршрутов.

# 1 Аналитическая часть (проектная часть)

## 1.1 Характеристика предметной области

Основной задачей проектной деятельности является хранение и редактирование базы данных, содержащей информацию о автобусах, водителях и маршрутах автобусного парка.

Приложение предназначено для двух категорий пользователей: сотрудники автобусного парка предоставляющие отчеты начальству и следящими за ходом процесса, директора автобусных парков, для внесения изменений в базу данных.

## 1.2 Постановка задачи

Необходимо разработать структуру и модели базы данных, для реализации Windows приложения для электронного документооборота в сфере общественного транспорта.

В приложении нужно предусмотреть возможность добавления и редактирования записей в базу данных в категориях:

* Водитель:
* Автобус;
* Транспортный маршрут;

Также должна быть реализована система ранжирования доступа к базе данных: разделение ролей и возможностей пользователей.

В том числе необходимо реализовать возможность составления и печати бумажных отчетов, для предоставления в инстанции, использующие бумажный документооборот.

## 1.3 Проектирование Windows-приложения

#### 1.3.1 Описание ролей пользователей и функциональных возможностей приложения.

В приложении предусмотрено две пользовательские роли:

* Диспетчер: пользователь, имеющий возможность читать данные из базы данных, следить за актуальностью и выводить в печать отчеты по категориям.
* Администратор: пользователь с доступом к редактированию базы данных. Доступные реализованные функции: добавить запись, удалить запись и редактировать запись.

Разработанные формы позволяют максимально эффективно использовать пользовательский интерфейс для учета и редактирования базы данных. Реализованные функции:

* добавление,
* удаление,
* редактирование записей в базе данных,
* удаленный доступ к централизованной базе,
* вывод наглядного отчета по категориям с возможностью печати прямо из приложения.

#### 1.3.2 Архитектура приложения

При разработке приложения были реализованы различные экраны:

Приложение встречает диалоговым окном авторизации (Рисунок 1).

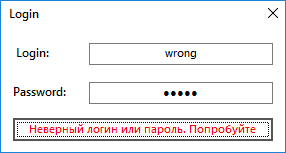


Рисунок — Окно авторизации "Login"

После успешной аутентификации нас встречает главный экран (Рисунок 2).

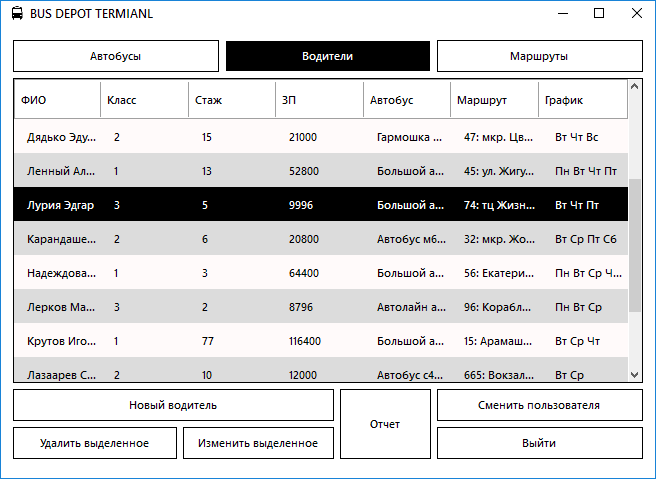


Рисунок — Главный экран приложения.

Наверху главного экрана присутствуют кнопки, реализующие систему «вкладок» в приложении. При нажатии открывается соответствующее представление базы данных (Рисунок 3, 4).

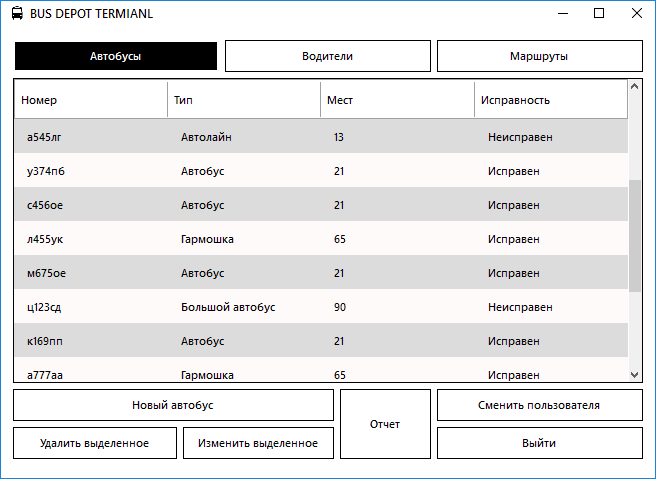


Рисунок — Главный экран приложения.

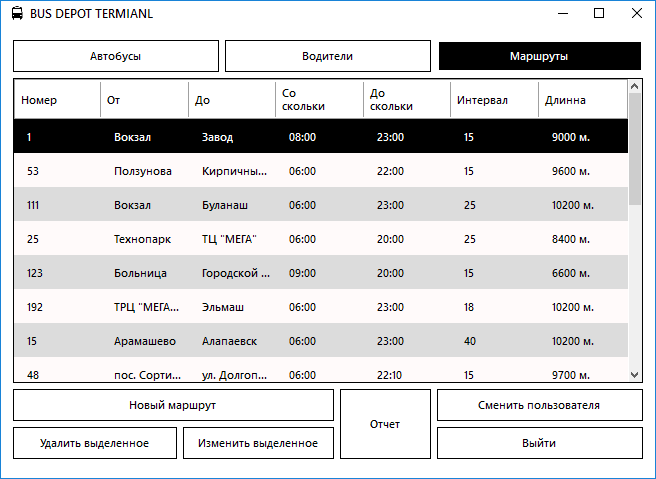


Рисунок —Главный экран приложения.

Для всех категорий базы данных предусмотрены экраны «удалить», «создать» и «удалить» (Рисунок 5, 6, 7).

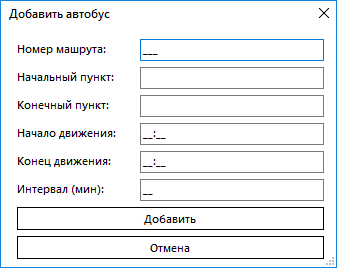


Рисунок —Экран «Добавить маршрут»

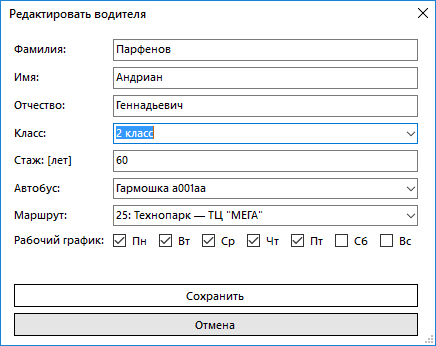


Рисунок 6 —Экран «Редактировать водителя»

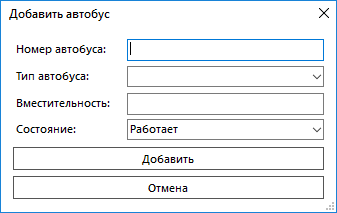


Рисунок 7 —Экран «Добавить автобус»

На экране «Добавить водителя» и «Редактировать водителя» реализован выбор автобусов и маршрутов из существующих посредством стандартных дропдаунов Windows Forms Combo Box.

#### 1.3.3 Выбор средств реализации

Для разработки приложений с применением технологии Windows Forms в большинстве случаев используется студия разработки от компании Microsoft — Visual Studio, из-за визуального редактора форм, что позволяет не работать с XML разметкой и воспользоваться удобными инструментами проектирования интерфейсов при помощи перетаскивания функциональных блоков мышью.

Также в Visual Studio интегрирован компилятор C# и менеджер пакетов NuGet для удобства установки необходимых драйверов и пактов разработки, таких как драйвер для работы с базой данных Mongo Data Base.

Для предоставления отчетов по категориям я выбрал стек HTML CSS, что обеспечило максимальную простоту и гибкость разработки шаблонов будущих отчетов. Разработкой HTML шаблонов занимался в IDE от Microsoft — Visual Studio Code.

**MongoDB** — документно-ориентированная система управления базами данных с открытым исходным кодом, не требующая описания схемы таблиц. Классифицирована как NoSQL, использует JSON-подобные документы и схему базы данных. Написана на языке C++. Используется в веб-разработке, в частности, в рамках JavaScript-ориентированного стека MEAN.

Ранее мне приходилось работать с данной СУБД в проектах на Django Python и Node.js. Во время работы я смог оценить удобство и простоту разработки моделей для не реляционной базы данных.

## 1.4 Проектирование базы данных

Для реализации задачи необходимо спроектировать три таблицы и продумать между ними связь по ключевым полям.

В качестве ключевых полей при реализации базы данных был выбран автогенерируемый, встроенный в MongoDB атрибут ObjectId.

Связываться коллекции будут через коллекцию «Водитель». У нее будут поля BusId и RouteId, что позволит обеспечить связь между всеми коллекциями базы данных.

#### 1.4.1 Структура входной и выходной информации

Для контроля структуры входной и выходной информации при работе с базой данных принято решение разработать модель базы данных, которая будет являться классом языка программирования C#. Помещение данных в экземпляр класса, с предопределенными полями позволит избежать проблем с различными типами данных и нарушением целостности структуры документов коллекций.

#### 1.4.2 Обоснование выбора СУБД

**MongoDB** — документно-ориентированная система управления базами данных с открытым исходным кодом, не требующая описания схемы таблиц. Если сравнивать с реляционными БД и с оглядкой на конкретно MySQL — MongoDB идеально вписывается там, где структура данных заранее неизвестна.

Из-за небольшого времени на разработку было принято решение выбрать именно данную СУБД, ведь она позволяет исправить многие ошибки уже во время выполнения проекта.

Также немаловажной причиной выбора является большой личный опыт работы с данной СУБД.

# 2 Практическая часть

## 2.1 Разработка базы данных

Для реализации работы с базой данных на удаленном сервере под управлением Ubuntu 18.04 Lts был развернут сервер MongoDb Server и реализованы два пользователя с разными правами:

Admin: пользователь с правами readwrite.

Reader: пользователь с правами readonly.

Система авторизации в последующем будет реализована именно посредствам MongoDB.

Далее необходимо установить драйвер работы с удаленными СУБД в наш C# проект, а также реализовать модели базы данных в виде классов.

#### 2.1.1 Реализация интерфейса базы данных

В проекте создан файл MongoTools.cs содержащий класс для связи и реализации основных операций с удаленной БД.

Основные методы класса (Рисунок 8, 9, 10, 11)

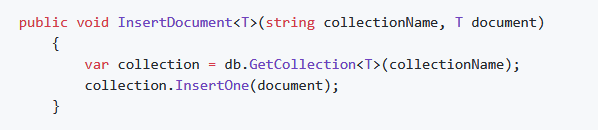


Рисунок 8 — Метод «Вставить документ в коллекцию»

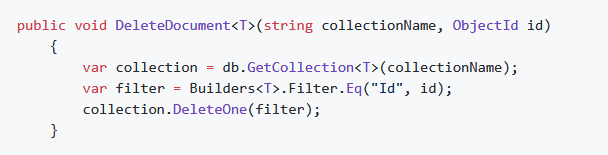


Рисунок 9 — Метод «Удалить документ из коллекции»

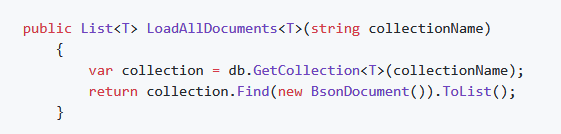


Рисунок 10 — Метод «Получить все элементы коллекции»

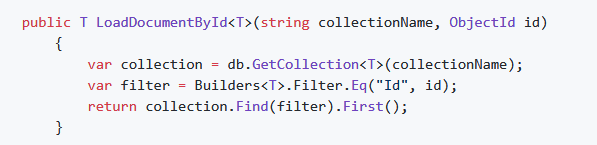


Рисунок 11 — Метод «Получить элемент коллекции»

#### 2.1.2 Модель базы данных

Модель «Водитель» (Рисунок 12) включает в себя поля:

* Уникальный ключ
* Фамилия
* Имя
* Отчество
* Класс
* Стаж
* Зарплата
* Уникальный ключ элемента таблицы «Автобус»
* Уникальный ключ элемента таблицы «Маршрут»

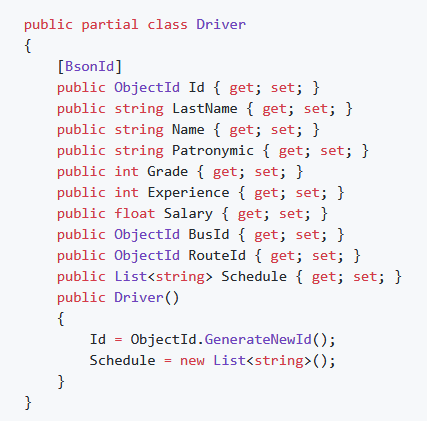


Рисунок 12 — Модель коллекции «Водитель»

Модель «Маршрут» (Рисунок 13) включает в себя поля:

* Уникальный ключ
* Номер
* Начальная станция
* Конечная станция
* Начальное время
* Конечное время
* Интервал движения
* Протяженность маршрута

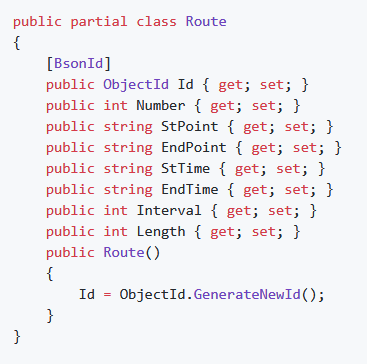


Рисунок 13 — Модель коллекции «Маршрут»

Модель «Автобус» (Рисунок 14) включает в себя поля:

* Уникальный ключ
* Номер
* Тип автобуса
* Вместительность
* Исправность

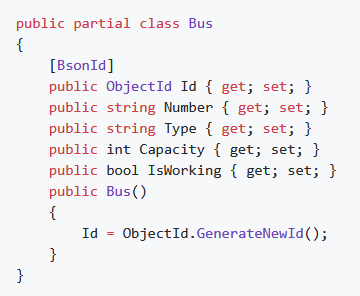


Рисунок 14 — Модель коллекции «Автобус»

## 2.2 Описание Windows-приложения

#### 2.2.1 Подключение к удаленной базе данных

Пользователя встречает экран аутентификации. После нажатии кнопки Login данные из инпутов передаются в функцию-слушатель loginBtn\_Click (Рисунок 15)

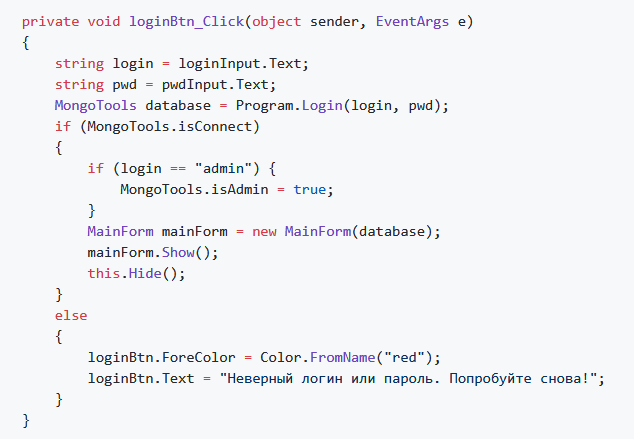


Рисунок 15 — Обработчик кнопки на экране аутентификации.

Приложение передает строки логина и пароля в метод класса Program — Login, где происходит попытка создания экземпляра подключения к базе данных. (Рисунок 16)

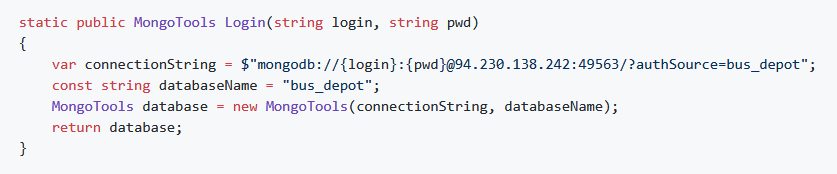


Рисунок 16 — Создание подключения к базе данных с использованием логина и пароля.

Далее управление передается обратно в обработчик кнопки экрана аутентификации. Там происходит проверка корректности подключения. Поле isConnect создается в классе MongoTools при создании экземпляра базы данных. (Рисунок 17)

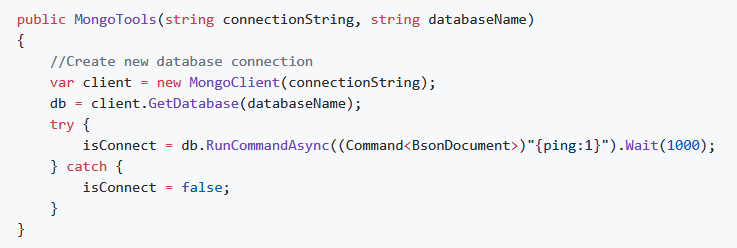


Рисунок 17 — Создание экземпляра базы данных и проверка корректности подключения.

#### 2.2.2. Создание экземпляра коллекции

Реализацию добавления документа в коллекцию базы данных рассмотрим на примере самой маленькой коллекции «Автобус». При нажатии на кнопку «Добавить» срабатывает обработчик события нажатия на эту кнопку (Рисунок 18). Если выполняется условие, проводящее валидацию ввода данных на экране, то создается экземпляр класса «Автобус», его поля заполняются данными, введенными пользователем на экране. После этого экземпляр передается в метод класса MongoTools, добавляющий элемент в коллекцию.

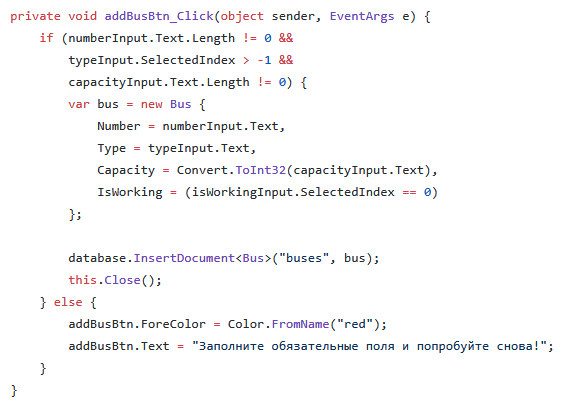


Рисунок 18 —Обработчик нажатия на кнопку «Добавить» экрана «Добавление автобуса».

#### 2.2.3. Формирование отчета

Для отображения отчета используется HTML шаблон вставляемый в элемент управления WebBrowser формы «Отчет».

Предварительно создан шаблон HTML средствами Visual Studio Code. При нажатии на кнопку «Отчет» в форму передается подключение к базе данных, информация о выбранной категории и данные DataGridView. Из этих данных генерируется HTML отчет. (Рисунок 18, 19, 20)

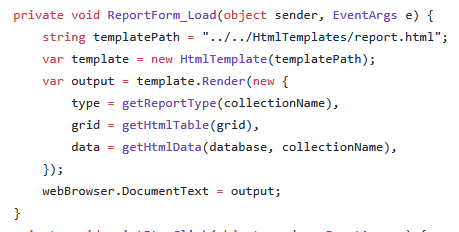


Рисунок 19 — Создание HTML-отчета.

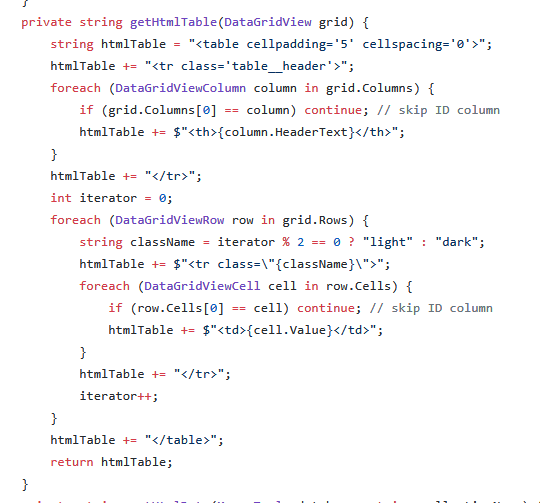


Рисунок 20 — Создание HTML-таблицы из данных DataGridView



Рисунок 21 — Создание HTML-данных для отчета о таблице «Водители».

Пример HTML-отчета (Рисунок 22).

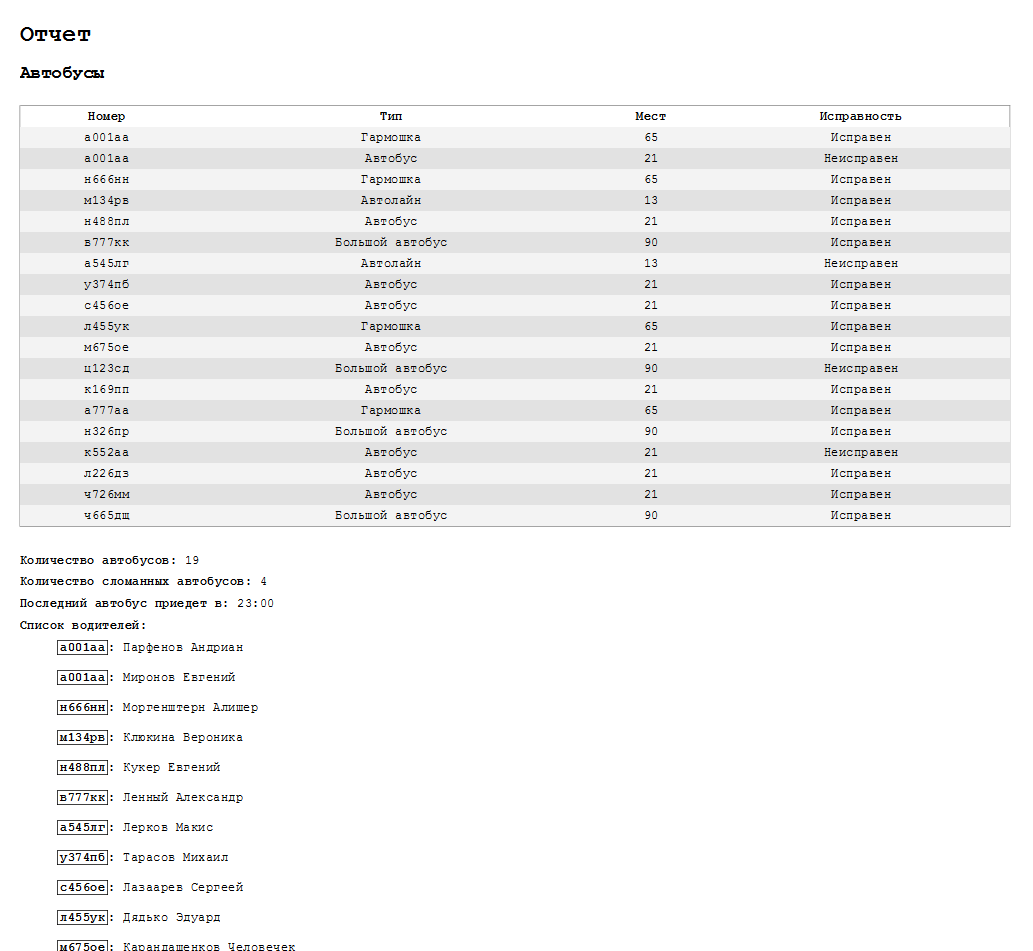


Рисунок 22 — Пример отчета.

Также в отчете реализована функция передачи на печать.