Министерство просвещения ПМР

ГОУ СПО «Тираспольский техникум информатики и права»

Курсовая работа

по дисциплине «Разработка программных модулей»

на тему: Разработка информационной системы деятельности диспетчера автобусного парка

Выполнил обучающийся

Сиглов Артем Витальевич

Специальность:2.09.02.07 Информационные системы и программирование

Руководитель

Преподаватель высшей

квалификационной категории

Балашова Юлия Владимировна

(оценка)

(подпись)

Тирасполь 2024

Оглавление

**ВВЕДЕНИЕ**

Глава 1 Теоретические основы деятельности диспетчера автобусного парка

1.1Характеристика деятельности диспетчера автобусного парка

1.2Анализ существующих решений деятельности диспетчера автобусного парка

1.3Выбор модели реализации информационной системы

Глава 2 Практическая реализация разработки информационной системы деятельности диспетчера автобусного парка

2.1Постановка задачи

2.2Техническое задание на разработку информационной системы деятельности автобусного парка

2.3Документация на программный продукт

2.4Тестирование информационной системы деятельности диспетчера автобусного парка

Заключение

Список литературы

Приложения

**ВВЕДЕНИЕ**

На сегодняшний день общественный транспорт играет большую роль и пользуется спросом как в крупных городах - мегаполисах, так и на периферии. Перевозки могут выполняться такими видами подвижного состава, как троллейбусы, автобусы, трамваи, легковые автомобили и электропоезда.

Среди всех видов пассажирского автомобильного транспорта преимущественное развитие получает автобусный, являющийся наиболее массовым видом пассажирского транспорта общего пользования. Перевозка автобусами может осуществляться по городским, пригородным, междугородним и международным маршрутам. К пригородным автобусным перевозкам относятся перевозки, осуществляемые за пределы города (населенного пункта) на расстояние до 50 км включительно. Пригородные зоны входят в сферу постоянных трудовых, деловых и культурно-бытовых связей с городом, характер которых зависит от экономики и географии пригородных районов.

Актуальность курсовой работы заключается в необходимости проектирования и реализации информационной системы по пассажира перевозкам, что позволит автоматизировать работу транспортной компании.

Цель работы разработать информационную систему управления пассажира перевозками.

Основные задачи транспорта - качественное, своевременное обслуживание и полное удовлетворение потребностей населения и народного хозяйства в перевозках, а также повышение экономической эффективности его работы.

Для выполнения курсовой работы были поставлены следующие задачи:

* Исследовать предметную область.
* Выявить проблемы и недостатки транспортной отрасли.
* Проанализировать существующие системы, их возможности и недостатки.
* Разработать информационную систему деятельности диспетчера автобусного парка.

Объектом исследования является деятельность диспетчера автобусного парка.

Предметом исследования является фирма, осуществляющая пассажирские пригородные, междугородные и международные перевозки. Для решения поставленных задач использовались методы: анализ (при описании предметной области); синтез (при реализации информационной системы); формализации (при проектировании информационной системы).

Разработанная информационная система позволит:

* хранить сведения о водителях, маршрутах, характеристиках автобусов;
* следить за соблюдением графиков движения;
* отслеживать автомобили в реальном времени;
* информировать клиентов о местонахождении груза.

ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ДЕЯТЕЛЬНОСТИ ДИСПЕТЧЕРА АВТОБУСНОГО ПАРКА

# 1.1 Характеристика деятельности диспетчера автобусного парка

Пассажирские перевозки - это организованная перевозка пассажиров и багажа, при помощи специально оборудованных транспортных средств, (такси, микроавтобус, автобус, троллейбус, трамвай, поезд, самолёт, теплоход) на определённые расстояния.

Пассажир - это человек, который перевозится в каком-либо пассажирском транспортном средстве, на основании гласного или не гласного договора и не является членом экипажа.

Пассажирское транспортное средство - это специально оборудованный транспорт, перевозящий пассажиров и багаж.

Основными видами пассажирских перевозок являются автомобильные перевозки (такси, микроавтобусы, маршрутные такси, автобусы), железнодорожные перевозки (электрички, поезда дальнего следования), морские и речные перевозки (корабли, паромы), авиаперевозки (вертолёты, самолёты).

Пассажирские перевозки автотранспортом являются самыми распространёнными в мире. К автотранспорту относятся перевозки на автомобилях, микроавтобусах, автобусах. Ежедневно миллиарды пассажиров по всему миру пользуются услугами общественного и коммерческого автомобильного транспорта. Шаговая доступность дешевизна, комфорт и удобство использования — это основные преимущества автотранспортных перевозок.

Автобусный парк содержит определенное количество подвижных единиц. На маршруты выходят иногда одни и те же автобусы. Автобусы из автобусного парка ездят по определённым, заранее спроектированным маршрутам. Все автобусы должны быть в исправном стоянии. За техническим состоянием автобусов наблюдают тех. служащие или механики.

Водители отвечают за работоспособность автобуса, закрепленного за ними. Каждому водителю администрация автобусного парка должна предоставить автобус - транспортное средство, при помощи которого водитель должен исполнять свои служебные обязанности. Каждому водителю назначается конкретный автобус, за которым водитель обязан следить, но на практике, это не всегда так. В случае поломки автобуса, водителю могут предоставить другой автобус, в то время как, в случае невыхода на работу водителя, на его автобусе, по распоряжение начальства, может ездить другой водитель.

Диспетчер закрепляет за автобусами маршруты, определяет в какую смену выходит водитель и на каком автобусе. Указывает маршрут и время возвращения в парк. Каждый маршрут имеет пункт отправления и назначения.

# 1.2 Анализ существующих решений деятельности диспетчера автобусного парка

После изучения предметной области были найдены следующие аналоги программ, программа «Автопарк» и «ТМ корпоративные поездки».

Программа «Автопарк» предназначена для комплексной автоматизации авто предприятия. Программа позволяет вести учет всего, что связано с автопарком: ТО, ремонты, ДТП, ГСМ, резина, комплектующие, документы на автомобили, путевые листы; прием и обработку поступающих заказов, учет работы и взаиморасчеты с водителями, учет работы операторов и диспетчеров. Она ведет учет договоров проката, платежей и взаиморасчетов с клиентами, историю каждого автомобиля, формирует все необходимые для работы документы и отчеты.

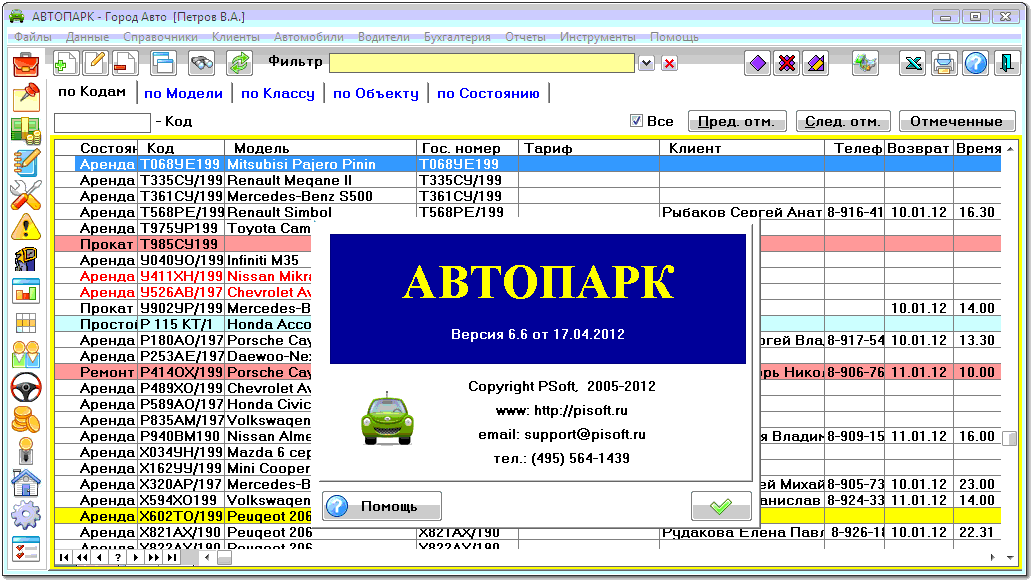


Рис. 1 Процесс работы программы

Программа ТМ: Корпоративные поездки — современное решение, которое позволяет произвести автоматизацию собственного автопарка предприятия. Благодаря цифровизации процесса заказа можно в реальном времени собирать информацию о положении автомобилей и сократить холостой пробег. Внедрение программного комплекса позволяет оптимизировать работу и осуществлять управление корпоративным автопарком эффективно

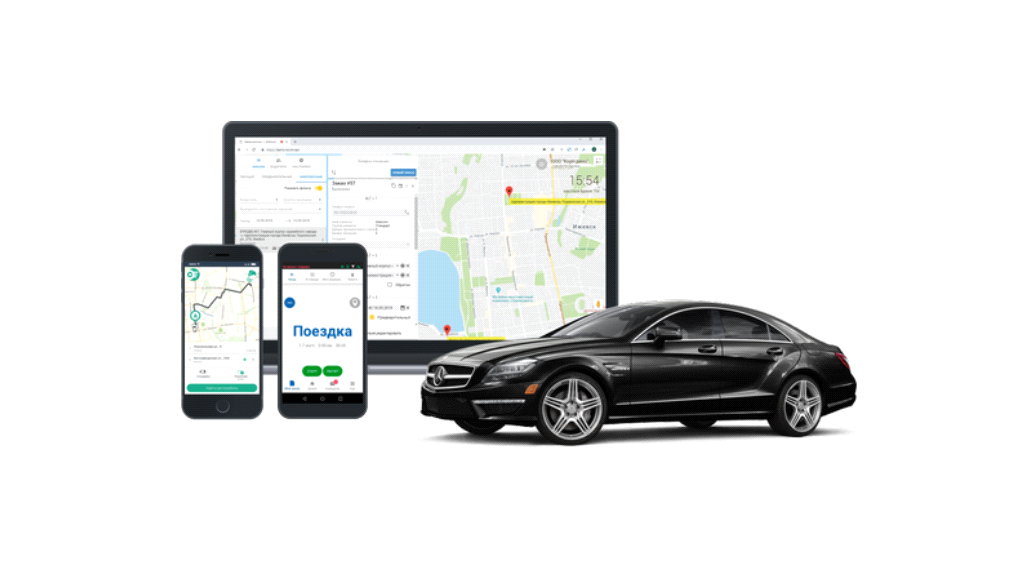


Рис. 2 ТМ корпоративные поездки.

Внешний вид и содержание печатаемых документов может настраиваться конечными пользователями. Предусмотрено резервирование автомобилей; имеется наглядный график изменения состояний автомобилей (шахматка). Гибкая настройка тарифов: тарифы, зависящие от числа дней проката, почасовые тарифы, тарифы за перепрокат, тарифы выходного дня и пр. Имеется поддержка тарифных планов для корпоративных клиентов, автоматизация взаиморасчетов с клиентами. Программа имеет развитую систему разделения доступа сотрудников к различным функциям и отчетам.

Программа Автопарк является сетевой и позволят работать с общей базой данных одновременно с нескольких рабочих мест: операторы, диспетчера, бухгалтеры, руководители. Возможен доступ к базе данных программы из удаленных филиалов через интернет.

1. модели автомобилей;
2. состояния автомобилей;
3. график занятости автомобилей;
4. учет ремонтов, то, дтп автомобилей;
5. учет путевых листов;
6. учет оплаты гсм;
7. водители**,**графики работы водителей;
8. взаиморасчеты с водителями;
9. тарифы**,**тарифные планы.

Клиенты

1. Категории клиентов.
2. Учет контактов с клиентами.
3. Работа с корпоративными клиентами.
4. Улицы и районы.
5. Дополнительные услуги.
6. Сотрудники и разделение доступа.
7. Категории сотрудников.
8. Журнал операций.
9. Фирмы.
10. Договора проката.
11. Резервирование автомобилей.
12. Кассовая книга.
13. Задания сотрудников.
14. Отчеты по автомобилям.
15. Отчеты по договорам.
16. Отчеты по кассе
17. Настройка печатаемых документов.
18. Параметры настройки программы
19. Использование фильтров.
20. Автоматическое резервирование данных.
21. Программа ТМ корпоративные поездки имеет следующие преимущества:
22. Мобильное приложение для сотрудников.
23. Мобильное приложение для водителей.
24. Кабинет организации.
25. GPS-трекинг.
26. Аналитическая платформа.
27. Заказ поездок.
28. Кабинет клиента.
29. Карты и поиск адресов.
30. Лимит на поездки.
31. Прием звонков роботом.
32. Рабочее место ТМ.
33. Управление составом автопарка.

# 1.3 Выбор модели реализации программного продукта

При анализе предметной области и реализации поставленной задачи необходимо сделать выбор модели реализации программного продукта.

Объектно-ориентированное программирование (ООП) – методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования.

При анализе поставленных задач можно сделать вывод, что для реализации данного программного продукта наилучшим моделью программирования является ООП на языке С#.

Описание выявленных сущностей

При анализе предметной области были выявлены следующие сущности:

1. Водитель.

2.Маршрут.

3.Автобус.

Сущность «Водитель» отвечает за информацию о водителях автобусного парка, а именно: ключ, фамилия, имя, отчество, стаж, класс, оклад, маршрут, график. Необходимо реализовать работу с данной информацией, а именно отображение информации, добавление информации, и удаление.

Сущность «Маршрут» отвечает за информацию о маршрутах автобусного парка: номер маршрута, начало маршрута, конец маршрута, время отправления, время прибытия, гос. номер автобуса, который закреплен за маршрутом, ключ водителя, который закреплен за автобусом. Предусмотреть добавление новой информации, корректировка и удаление.

Сущность «Автобус» отвечает за информацию об автобусах в автобусном парке: ключ гос. номер, тип, вместимость, исправен.

Предусмотреть списание старого автобуса, добавление нового автобуса.

ГЛАВА 2. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ РАЗРАБОТКИ ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЕЯТЕЛЬНОСТИ ДИСПЕТЧЕРА АВТОБУСНОГО ПАРКА

# 2.1 Постановка задачи

Основная деятельность диспетчера автобусного парка - это получение и передача информации от водителей автобуса, слежение за их перемещениями, в том числе производит запись поступающих данных.

Информация, хранимая в информационной системе автобусного парка:

* информация о маршрутах;
* информация о водителях;
* сведения об автобусах;
* расписание работы водителей.

Информация о маршрутах. Маршрут-путь объекта из точки А в точку В.

Информация о маршруте должна содержать:

* номер маршрута;
* название начального и конечного пункта движения;
* время начала и конца движения;
* интервал движения и протяженность в минутах.

Информация о водителе:

* фамилия;
* имя;
* отчество;
* стаж;
* класс;
* оклад;
* маршрут;
* график.

Сведения об автобусе должны содержать:

* тип;
* вместимость;
* гос. номер.

В базе должен храниться график работы водителей в виде массива дней выхода на работу.

Диспетчеру автопарка могут потребоваться следующие сведения:

1. список водителей, работающих на определенном маршруте с указанием графика их работы;
2. какие автобусы обслуживают данный маршрут;
3. протяженность маршрутов и их минимальная и максимальная протяженность
4. на каком маршруте работает водитель с максимальным стажем;
5. какова общая протяженность маршрутов, обслуживаемых автопарком;
6. какие водители не вышли на линию по причине неисправности автобуса.

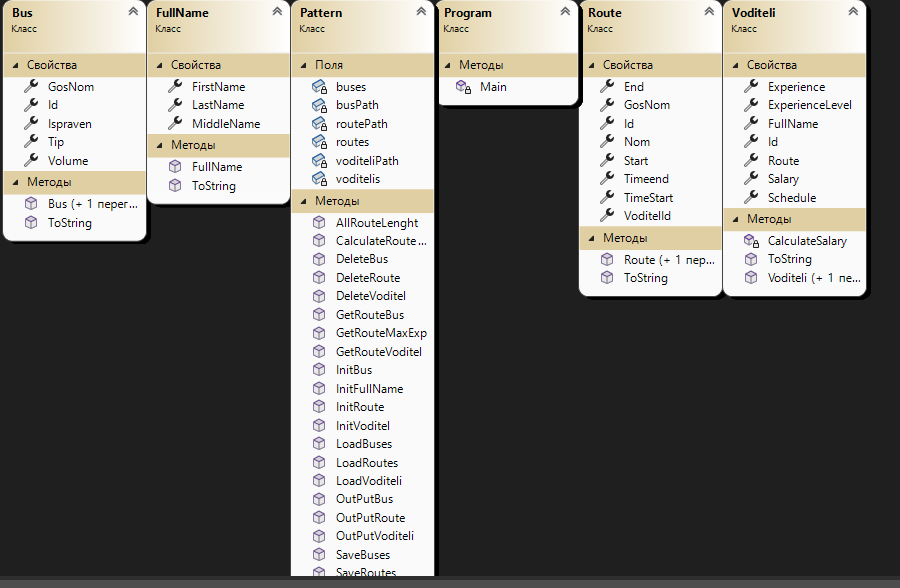


Рис. 3 UML Диаграмма классов

2.2 Техническое задание на разработку автоматизированного рабочего места технолога общественного питания

Цель: разработка информационной системы для автоматизации деятельности диспетчера автобусного парка.

Исходные данные: информация о водителях, маршрутах, автобусах.

Организация маршрутных пассажира перевозок обратилась с задачей — автоматизировать деятельность диспетчера автобусного парка. Из-за одновременной работы нескольких человек над одним документом, а также других особенностей Excel часто возникали ошибки в учете данных, которые в дальнейшем трудно было найти и исправить. С ростом числа маршрутов, увеличилось и количество таких ошибок.

Реализация

В начале работы над проектом изучена специфика учета данных автобусного парка. Она обусловлена не только сферой деятельности диспетчера автобусного парка, но и особенностями работы в учреждении, соблюдением определенных правил. Речь идет о различных дополнительных факторах.

Из-за специфического учета данных об автобусном парке конфигурации организации не подошли бы, и было принято решение — разработать информационную систему для автоматизации деятельности диспетчера автобусного парка. После тестирования новой информационной системы были исправлены и настроены дополнительные нюансы, которые были выявлены в ходе тестирования.

Информационная система поддерживает функционал для автоматизации деятельности диспетчера автобусного парка.

Созданы файлы для хранения информации о маршрутах, автобусах и водителях.

Настроено консольное приложение для диспетчеров, где их действия максимально упрощены: они выбирают нужный пункт меню, в котором описаны возможные действия в информационной системе

Реализованы все пожелания организации пассажира перевозок по формированию дополнительных справочников.

Важный момент в том, что внедрение новой программы никак не усложнило работу диспетчеров. Новый функционал максимально прост, и работа в данной системе учета не сложнее работы с таблицами Excel. Кроме того, новая информационная система незамедлительно сигнализирует об ошибках ввода данных, благодаря чему сведено к минимуму влияние человеческого фактора на систему учета.

Результаты

По результатам проекта разработана и настроена специализированная информационная система для автоматизации деятельности диспетчера автобусного парка. Новая система автоматизации учета позволила уйти от неэффективной работы в Excel и избежать множества ошибок. Информационная система стабильно работает и отвечает всем требованиям учреждения.

Для реализации поставленной задачи выбрана платформа «Microsoft Visual Studio», которая является интегрированной средой разработки для автоматизации различных областей экономической деятельности. В конкретный программный продукт, входящий в систему программ «Microsoft Visual Studio», включаются те функции и возможности, которые отвечают назначению этого продукта.

Все составляющие системы программ «Microsoft Visual Studio» можно разделить на Технологическую платформу (оболочку) и Конфигурации. Технологическая платформа представляет собой набор различных библиотек, используемых для автоматизации экономической деятельности и не зависящих от конкретного законодательства и методологии учета. Конфигурации являются собственно прикладными решениями. Каждая конфигурация ориентирована на автоматизацию определенной сферы экономической деятельности и, разумеется, отвечает принятому законодательству.

Плюсом разделения системы на оболочку и конфигурацию является открытость программы и возможность изменения конфигурации самим пользователем или организациями, специализирующимися на внедрении и поддержке программных продуктов фирмы (Microsoft Visual Studio), адаптируя ее к потребностям данного предприятия. Кроме этого, пользователь может создавать новые программные модули и структуры, однако их создание требует глубоких знаний самой системы и наличия хотя бы базового образования в области программирования.

Возможности по изменению конфигурации позволяют обеспечить максимальное соответствие системы индивидуальным особенностям учета в конкретной организации, обеспечивают возможность развития системы вместе с компанией, а также расширяют функциональность программы путем добавления к ней новых компонентов.

Формирование структуры представления и хранения данных в комплексной информационной системе, отвечает требованиям компании, и позволяет реализовать программу для автоматизации деятельности диспетчера автобусного парка.

Разрабатываемая информационная система должна удовлетворять следующим требованиям:

* содержать информацию о водителях, маршрутах и автобусах и оперировать ей при деятельности диспетчера;
* иметь удобный пользовательский интерфейс;
* иметь возможность модернизации с учётом пожеланий пользователя;
* слабо зависеть от вычислительной мощности ПК.

Системные требования

1. 32-разрядный сервер:

* Процессор Intel Pentium IV/Xeon 2,4 ГГц и выше
* Оперативная память 1024 Мб и выше
* Жесткий диск 40Гб и выше
* Устройство чтения компакт-дисков
* USB-порт
* SVGA-видеокарта

1. 64-разрядный сервер:

* Процессор с архитектурой x86-64 (Intel с поддержкой EM64T, AMD с поддержкой AMD64).
* Оперативная память 2048 Мб и выше
* Жесткий диск 40Гб и выше
* Устройство чтения компакт-дисков
* USB-порт
* SVGA-видеокарта

Изучив структуру, можно использовать полученные данные для создания программы автоматизации деятельности диспетчера автобусного парка. Информационная система будет разработана на базе «Microsoft Visual Studio». В качестве технологии проектирования был выбран язык программирования «С#».

# 2.3 Документация на программный продукт

Класс – некая сущность, которая задает некоторое общее поведение для объектов. Таким образом, любой объект может принадлежать или не принадлежать определенному классу, то есть обладать или не обладать поведением, которое данный класс подразумевает. В программной реализации курсовой работы используются такие классы, как: Voditeli, Route, Bus, FullName.

Класс Voditeli используется для описания информации о водителе. В классе используются поля: стаж experience типа int, оклад salary типа int. Данный класс содержит метод ToString() предназначенный для вывода информации о объекте.

Класс Route используется для описания информации о маршрутах. В классе используются поля: время отправление и время прибытия типа datetime.

Класс Bus используется для описания информации об автобусе. В классе используется поле: исправен типа bool.

Описание программной реализации

Программный продукт был реализован с использованием объектно-ориентированного программирования, то есть в ней используются классы, каждый из которых описывает некоторую сущность предметной области.

Программная реализация курсовой работы была выполнена в интегрированной среде разработки Microsoft Visual Studio 2019 на языке программирования высокого уровня C#.

# Руководства пользователя

Прежде чем получить доступ к программе, пользователю необходимо ввести логин и пароль, как показано на Рисунке 4. Это обеспечивает безопасность и контроль доступа к функционалу. После успешной аутентификации система разрешает доступ к основному интерфейсу программы.

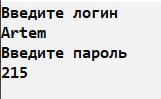


Рис. 4 Авторизация в системе пользователя  
Пользователь может легко взаимодействовать с программой благодаря созданному консольному приложению. После запуска отображается основное окно программы, изображенное на Рисунке 5, что обеспечивает удобство использования.

Чтобы добавить элемент необходимо ввести с клавиатуры значение – «1», для вывода информации необходимо ввести – «2», для удаления ввести – «3», для выхода из программы необходимо ввести – «0».

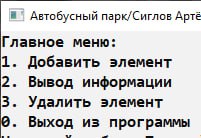


Рис. 5 Главное меню

Меню вывода информации на Рисунке 6 предоставляет пользователю различные опции для вывода информации о системе или организации, вероятно, связанной с транспортом или автобусными маршрутами. Давайте разберем каждый пункт:

1. Вывести список водителей на маршруте с графиком работы: Позволяет пользователю получить информацию о водителях, которые обслуживают определенный маршрут, а также их рабочий график.
2. Вывести список автобусов, обслуживающих данный маршрут: Предоставляет данные о всех автобусах, которые используются для обслуживания конкретного маршрута.
3. Вывести протяженность маршрутов и их минимальную и максимальную протяженность: Позволяет получить информацию о длине каждого маршрута, а также находит минимальную и максимальную длину среди всех маршрутов.
4. Вывести маршрут водителя с максимальным стажем: Предоставляет информацию о маршруте, на котором работает водитель с наибольшим стажем.
5. Вывести общую протяженность маршрутов: Позволяет узнать общую протяженность всех маршрутов в системе.
6. Вывести список водителей, не вышедших на линию из-за неисправности автобуса: Предоставляет информацию о водителях, которые не смогли выехать на маршрут из-за неисправности автобуса.
7. Вывести список водителей: Отображает полный список всех водителей, зарегистрированных в системе.
8. Вывести список автобусов: Предоставляет информацию о всех автобусах, которые есть в системе.
9. Вывести список маршрутов: Отображает полный список всех маршрутов, доступных в системе.
10. Выход в главное меню: Позволяет вернуться в главное меню программы для выбора других опций или завершения работы.

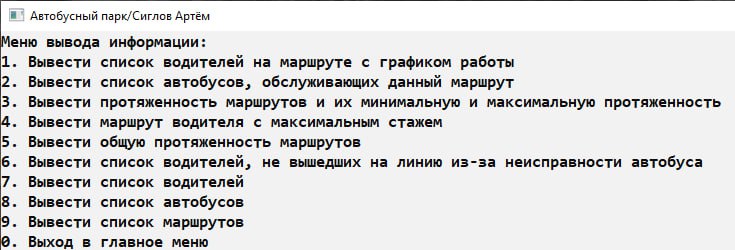


Рис. 6 Меню для вывода информации

Меню удаления на Рисунке 7 предоставляет пользователю опции для удаления элементов из системы. Рассмотрим каждый пункт:

1. удалить водителя: Позволяет пользователю удалить информацию о конкретном водителе из системы. Это может включать удаление его личных данных, рабочего графика и другой связанной информации.
2. удалить автобус: Предоставляет возможность удалить информацию о конкретном автобусе из списка. Это может включать удаление данных о модели автобуса, его техническом состоянии и прочей связанной информации.
3. удалить маршрут: Позволяет пользователю удалить определенный маршрут из системы. Это может включать удаление информации о маршруте, его длине, расписании и других связанных данных.
4. выйти в главное меню: Дает возможность вернуться в главное меню программы для выбора других опций или завершения работы.

Эти опции предоставляют пользователю удобный способ управления содержимым системы, позволяя добавлять, изменять и удалять информацию в соответствии с текущими потребностями и требованиями.

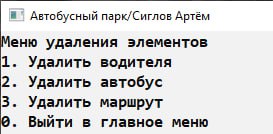


Рис. 7 Меню удаления элементов  
Руководство программиста

Класс Bus представляет собой описание автобуса и его характеристик. На Рисунке 8 описывается классBus.

Давайте разберем его по частям:

1. Класс Bus:

Этот класс описывает автобус и его свойства.

Он помечен атрибутом [Serializable], что означает, что объекты этого класса можно сериализовать (преобразовывать в последовательность байтов для сохранения в файле или передачи по сети).

1. Свойства:

Id (int): Идентификатор автобуса.

GosNom (string): Государственный номер автобуса.

Tip (string): Тип автобуса.

Volume (int): Вместимость автобуса.

Ispraven (bool): Показатель исправности автобуса.

1. Конструкторы:

public Bus(): Конструктор по умолчанию, который создает экземпляр класса с пустыми значениями свойств.

public Bus(int id, string gosNom, string tip, int volume, bool ispraven): Конструктор с параметрами, позволяющий инициализировать свойства объекта при его создании.

1. Методы:

public override string ToString(): Метод, который переопределяет стандартный метод ToString() и возвращает строку, содержащую значения всех свойств объекта, что удобно для вывода информации о объекте.

Этот класс предоставляет удобный способ хранения информации о различных автобусах, их характеристиках и текущем состоянии.

|  |
| --- |
| [Serializable]  public class Bus  {  public int Id { get; set; }  public string GosNom { get; set; }  public string Tip { get; set; }  public int Volume { get; set; }  public bool Ispraven { get; set; }    public Bus() { }    public Bus(int id, string gosNom, string tip, int volume, bool ispraven)  {  Id = id;  GosNom = gosNom;  Tip = tip;  Volume = volume;  Ispraven = ispraven;  }  public override string ToString()  {  return $"Id: {Id}, GosNom: {GosNom}, Tip: {Tip}, Volume: {Volume}, Ispraven: {Ispraven}";  }  }  } |

Рис. 8 Описание класса Bus

Класс Pattern, представляет собой некоторый шаблон для управления информацией о водителях, маршрутах и автобусах. На Рисунке 9 описывается классPattern.

1. Поля класса:

voditelis, routes, buses: списки объектов классов Voditeli, Route, Bus соответственно. В них хранятся информация о водителях, маршрутах и автобусах.

DirectoryInfoVoditeli, DirectoryInfoRoute, DirectoryInfoBus: пути к файлам, где хранится информация о водителях, маршрутах и автобусах.

1. Методы и конструкторы:

InitFullName(): метод для создания экземпляра класса FullName (который, предположительно, представляет ФИО).

InitVoditel(), InitRoute(), InitBus(): методы для создания объектов классов Voditeli, Route, Bus.

GetRouteVoditel(), GetRouteBus(): методы для вывода информации о водителях и автобусах на определенном маршруте.

CalculateRouteLength(): метод для расчета максимальной и минимальной протяженности маршрута в минутах.

GetRouteMaxExp(): метод для вывода информации о маршруте, на котором работает водитель с максимальным стажем.

AllRouteLenght(): метод для расчета общей протяженности всех маршрутов в минутах.

VoditelNoJobBusFailure(): метод для вывода информации о водителях, не вышедших на работу из-за неисправности автобуса.

DeleteVoditel(), DeleteBus(), DeleteRoute(): методы для удаления водителя, автобуса и маршрута соответственно.

OutPutVoditeli(), OutPutBus(), OutPutRoute(): методы для вывода информации о водителях, автобусах и маршрутах.

1. Методы для работы с файлами:

SaveVoditeli(), SaveRoutes(), SaveBuses(): методы для сохранения списков в файлы.

LoadVoditeli(), LoadRoutes(), LoadBuses(): методы для загрузки списков из файлов.

Этот класс предоставляет функциональность для работы с данными о водителях, маршрутах и автобусах, их создания, изменения, удаления и сохранения в файлы. Он также позволяет проводить различные аналитические расчеты и получать информацию о работе системы.

|  |
| --- |
| public class Pattern  {  List<Voditeli> voditelis = new List<Voditeli>();  List<Route> routes = new List<Route>();  List<Bus> buses = new List<Bus>();  DirectoryInfo DirectoryInfoVoditeli = new DirectoryInfo("Voditeli.bin");  DirectoryInfo DirectoryInfoRoute = new DirectoryInfo("Route.bin");  DirectoryInfo DirectoryInfoBus = new DirectoryInfo("Bus.bin");  public static FullName InitFullName()  {  Console.WriteLine("Фамилия:");  string firstName = Console.ReadLine();  Console.WriteLine("Имя:");  string lastName = Console.ReadLine();  Console.WriteLine("Отчество:");  string middleName = Console.ReadLine();  return new FullName(firstName, lastName, middleName);  }  public void InitVoditel()  {  try  {  LoadVoditeli(DirectoryInfoVoditeli.FullName);  Console.WriteLine("Id:");  int id = int.Parse(Console.ReadLine());  Console.WriteLine("Полное имя:");  FullName fullName = InitFullName();  Console.WriteLine("Стаж:");  double exp = double.Parse(Console.ReadLine());  Console.WriteLine("Класс:");  int expLevel = int.Parse(Console.ReadLine());  Console.WriteLine("Маршрут:");  int route = int.Parse(Console.ReadLine());  Console.WriteLine("График:");  int daystart = int.Parse(Console.ReadLine());  Console.WriteLine("Сколько дней работает водитель?");  int dayJob = int.Parse(Console.ReadLine());  DayOfWeek[] schedule = new DayOfWeek[dayJob];  for (int i = 0; i < schedule.Length; i++)  {  schedule[i] = DateTime.FromOADate(daystart + i).DayOfWeek;  }  voditelis.Add(new Voditeli(id, fullName, exp, expLevel, route, schedule));  SaveVoditeli(DirectoryInfoVoditeli.FullName);  }  catch { InitVoditel(); }  }  public void InitRoute()  {  try  {  LoadBuses(DirectoryInfoBus.FullName);  LoadVoditeli(DirectoryInfoVoditeli.FullName);  Console.WriteLine("Id:");  int id = int.Parse(Console.ReadLine());  Console.WriteLine("Nom:");  int nom = int.Parse(Console.ReadLine());  Console.WriteLine("Start:");  string start = Console.ReadLine();  Console.WriteLine("End:");  string end = Console.ReadLine();  Console.WriteLine("Time Start:");  DateTime timeStart = DateTime.Parse(Console.ReadLine());  Console.WriteLine("Time End:");  DateTime timeEnd = DateTime.Parse(Console.ReadLine());  Console.WriteLine("Выберите автобус для маршрута:");  foreach (var bus in buses)  {  Console.WriteLine(bus.GosNom);  }  Console.WriteLine("Введите индекс:");  int busIndex = int.Parse(Console.ReadLine());  string gosNom = buses[busIndex - 1].GosNom;  Console.WriteLine("Выберите водителя для маршрута:");  foreach (var driver in voditelis)  {  Console.WriteLine(driver.Id);  }  Console.WriteLine("Введите индекс:");  int driverIndex = int.Parse(Console.ReadLine());  int voditelId = voditelis[driverIndex - 1].Id;  routes.Add(new Route(id, nom, start, end, timeStart, timeEnd, gosNom, voditelId));  SaveRoutes(DirectoryInfoRoute.FullName);  }  catch { InitRoute(); }  }  public void InitBus()  {  try  {  LoadBuses(DirectoryInfoBus.FullName);  Console.WriteLine("Id:");  int id = int.Parse(Console.ReadLine());  Console.WriteLine("гос.Номер:");  string gosnom = Console.ReadLine();  Console.WriteLine("Тип:");  string tip = Console.ReadLine();  Console.WriteLine("Вместимость:");  int volume = int.Parse(Console.ReadLine());  Console.WriteLine("Исправен:");  bool ispraven = bool.Parse(Console.ReadLine());  buses.Add(new Bus(id, gosnom, tip, volume, ispraven));  SaveBuses(DirectoryInfoBus.FullName);  }  catch { InitBus(); }  }  public void GetRouteVoditel()  {  LoadVoditeli(DirectoryInfoVoditeli.FullName);  LoadRoutes(DirectoryInfoRoute.FullName);  foreach (var item in routes)  {  Console.WriteLine("Маршрут-" + item.Nom);  }  Console.WriteLine("Введите номер маршрута, чтобы узнать информацию о водителях:");  int nom = int.Parse(Console.ReadLine());  foreach (var driver in voditelis)  {  if (driver.Route == nom) { Console.WriteLine( driver.ToString()); }  }  }  public void GetRouteBus()  {  LoadBuses(DirectoryInfoBus.FullName);  LoadRoutes(DirectoryInfoRoute.FullName);  foreach (var item in routes)  {  Console.WriteLine($"Маршрут-{item.Nom}");  }  Console.WriteLine("Введите номер маршрута, чтобы узнать автобусы:");  int nom = int.Parse(Console.ReadLine());  foreach (var route in routes)  {  if (route.Nom == nom)  {  Console.WriteLine($"Автобусы на маршруте {route.Nom}:");  foreach (var item in buses)  {  if (route.GosNom == item.GosNom)  {  Console.WriteLine(item.GosNom);  }  }  }  }  }  public void CalculateRouteLength()  {  LoadRoutes(DirectoryInfoRoute.FullName);  int max = 0;  int min = int.MaxValue;  foreach (var item in routes)  {  TimeSpan time = item.Timeend - item.TimeStart;  int routeLenght = (int)time.TotalMinutes;  if (routeLenght > max) { max = routeLenght; }  if (routeLenght < min) { min = routeLenght; }  Console.WriteLine($"Route {item.Nom}: {routeLenght} minutes");  }  Console.WriteLine($"Minimum route length: {min} minutes");  Console.WriteLine($"Maximum route length: {max} minutes");  }  public Route GetRouteMaxExp()  {  LoadVoditeli(DirectoryInfoVoditeli.FullName);  LoadRoutes(DirectoryInfoRoute.FullName);  Voditeli voditeli = null;  double maxExp = 0;  foreach (var driver in voditelis)  {  if (driver.Experience > maxExp) { maxExp = driver.Experience; }  if (driver.Experience == maxExp) { voditeli = driver; }  }  foreach (var route in routes)  {  if (route.VoditelId == voditeli.Id)  {  return route;  }  }  return null;  }  public int AllRouteLenght()  {  LoadRoutes(DirectoryInfoRoute.FullName);  int allLenght = 0;  foreach (var route in routes)  {  TimeSpan duration = route.Timeend - route.TimeStart;  allLenght += (int)duration.TotalMinutes;  }  return allLenght;  }  public void VoditelNoJobBusFailure()  {  LoadVoditeli(DirectoryInfoVoditeli.FullName);  LoadRoutes(DirectoryInfoRoute.FullName);  LoadBuses(DirectoryInfoBus.FullName);  foreach (var bus in buses)  {  if (bus.Ispraven == false)  {  foreach (var route in routes)  {  if (route.GosNom == bus.GosNom)  {  foreach (var driver in voditelis)  {  if (driver.Id == route.VoditelId)  {  Console.WriteLine(driver);  }  }  }  }  }  }  }  public void DeleteVoditel()  {  try  {  LoadVoditeli(DirectoryInfoVoditeli.FullName);  Voditeli voditeli = null;  foreach (var driver in voditelis)  {  Console.WriteLine(driver.Id);  }  int id = int.Parse(Console.ReadLine());  foreach (var driver in voditelis)  {  if (id == driver.Id)  {  voditeli=driver;  }  }  voditelis.Remove(voditeli);  SaveVoditeli(DirectoryInfoVoditeli.FullName);  }  catch { }  }  public void DeleteBus()  {  LoadBuses(DirectoryInfoBus.FullName);  Bus bus1 = null;  foreach (var bus in buses)  {  Console.WriteLine(bus.Id);  }  int id = int.Parse(Console.ReadLine());  foreach (var bus in buses)  {  if (id == bus.Id)  {  bus1=bus;  }  }  buses.Remove(bus1);  SaveBuses(DirectoryInfoBus.FullName);  }  public void DeleteRoute()  {  LoadRoutes(DirectoryInfoRoute.FullName);  Route route1 = null;  foreach (var route in routes)  {  Console.WriteLine(route.Id);  }  int id = int.Parse(Console.ReadLine());  foreach (var route in routes)  {  if (id == route.Id)  {  route1=route;  }  }  routes.Remove(route1);  SaveRoutes(DirectoryInfoRoute.FullName);  }  public void OutPutVoditeli()  {  LoadVoditeli(DirectoryInfoVoditeli.FullName);  foreach (var driver in voditelis)  {  Console.WriteLine( driver.ToString());  }  }  public void OutPutBus()  {  LoadBuses(DirectoryInfoBus.FullName);  foreach (var bus in buses)  {  Console.WriteLine( bus.ToString());  }  }  public void OutPutRoute()  {  LoadRoutes(DirectoryInfoRoute.FullName);  foreach (var route in routes)  {  Console.WriteLine( route.ToString());  }  }  public void SaveVoditeli(string fileName)  {  BinaryFormatter formatter = new BinaryFormatter();  using (FileStream fs = new FileStream(fileName, FileMode.Create))  {  formatter.Serialize(fs, voditelis);  }  }  public void LoadVoditeli(string fileName)  {  BinaryFormatter formatter = new BinaryFormatter();  using (FileStream fs = new FileStream(fileName, FileMode.Open))  {  voditelis = (List<Voditeli>)formatter.Deserialize(fs);  }  }  public void SaveRoutes(string fileName)  {  BinaryFormatter formatter = new BinaryFormatter();  using (FileStream fs = new FileStream(fileName, FileMode.Create))  {  formatter.Serialize(fs, routes);  }  }  public void LoadRoutes(string fileName)  {  BinaryFormatter formatter = new BinaryFormatter();  using (FileStream fs = new FileStream(fileName, FileMode.Open))  {  routes = (List<Route>)formatter.Deserialize(fs);  }  }  public void SaveBuses(string fileName)  {  BinaryFormatter formatter = new BinaryFormatter();  using (FileStream fs = new FileStream(fileName, FileMode.Create))  {  formatter.Serialize(fs, buses);  }  }  public void LoadBuses(string fileName)  {  BinaryFormatter formatter = new BinaryFormatter();  using (FileStream fs = new FileStream(fileName, FileMode.Open))  {  buses = (List<Bus>)formatter.Deserialize(fs);  }  }  } |

Рис. 9 Описание классаPattern.

Класс Route представляет собой описание маршрута. На Рисунке 10 описывается класс Route. Давайте рассмотрим его основные элементы:

1. Поля класса:

Id: идентификатор маршрута.

Nom: номер маршрута.

Start: пункт отправления.

End: пункт назначения.

TimeStart: время начала движения.

Timeend: время конца движения.

GosNom: государственный номер автомобиля, работающего на маршруте.

VoditelId: идентификатор водителя, работающего на маршруте.

1. Конструкторы:

Конструктор по умолчанию, который инициализирует все поля класса.

Конструктор с параметрами, который позволяет установить значения всех полей класса при создании объекта.

1. Методы:

ToString(): метод, который возвращает строковое представление объекта класса, содержащее значения всех его полей.

Этот класс позволяет создавать объекты, представляющие конкретные маршруты, задавать им различные параметры, такие как номер, пункты отправления и назначения, время начала и конца движения, а также государственный номер автомобиля и идентификатор водителя. Метод ToString() позволяет получить информацию о маршруте в удобном для чтения формате.

|  |
| --- |
| [Serializable]  public class Route  {  public int Id { get; set; }  public int Nom { get; set; }  public string Start { get; set; }  public string End { get; set; }  public DateTime TimeStart { get; set; }  public DateTime Timeend { get; set; }  public string GosNom { get; set; }  public int VoditelId { get; set; }  public Route() { }  public Route(int id, int nom, string start, string end, DateTime timestart, DateTime timeend, string gosNom, int voditelId)  {  Id = id;  Nom = nom;  Start = start;  End = end;  TimeStart = timestart;  Timeend = timeend;  GosNom = gosNom;  VoditelId = voditelId;  }  public override string ToString()  {  return $"Id-{Id} Nom-{Nom} Start-{Start} End-{End} Time Start-{TimeStart} TimeEnd-{Timeend} Gosnom-{GosNom} Voditel Id-{VoditelId}";  }  } |

Рис. 10 Описание класса Route

Класс FullName представляет собой описание полного имени с учетом фамилии, имени и, возможно, отчества. На Рисунке 11 описывается класс FullName. Рассмотрим его основные аспекты:

1. Поля класса:

FirstName: фамилия.

LastName: имя.

MiddleName: отчество (может быть null).

1. Конструктор:

Конструктор с параметрами, позволяющий инициализировать поля класса при создании объекта.

1. Методы:

ToString(): метод, возвращающий строковое представление полного имени, включая фамилию, имя и, если оно есть, отчество.

Этот класс позволяет создать объекты, представляющие полное имя человека, и получить информацию о нем в удобном для чтения формате. Метод ToString() используется для получения строкового представления объекта класса, содержащего значения его полей.

|  |
| --- |
| [Serializable]  public class FullName  {  public FullName(string firstName, string lastName, string middleName)  {  FirstName = firstName;  LastName = lastName;  MiddleName = middleName;  }  public string FirstName { get; set; }  public string LastName { get; set; }  public string MiddleName { get; set; } = null;  public override string ToString()  {  return $"{FirstName} {LastName} {MiddleName}";  }  } |

Рис. 11 Описание класса FullName

Класс Voditeli представляет водителя и содержит информацию о его идентификаторе, полном имени, стаже, классе, доходе, маршруте и графике работы. На Рисунке 12 описывается класс Voditeli. Рассмотрим основные аспекты этого класса:

1. Поля класса:

Id: идентификатор водителя.

FullName: объект типа FullName, представляющий полное имя водителя.

Experience: стаж работы водителя.

ExperienceLevel: класс водителя (в зависимости от стажа).

Salary: доход водителя, рассчитывается в зависимости от стажа и класса.

Route: номер маршрута, на котором работает водитель.

Schedule: массив дней недели, представляющий график работы водителя.

1. Конструкторы:

Конструктор без параметров.

Конструктор с параметрами, позволяющий инициализировать поля класса при создании объекта. При инициализации также происходит расчет зарплаты.

1. Методы:

CalculateSalary(): метод для расчета зарплаты водителя в зависимости от его стажа и класса.

ToString(): метод, возвращающий строковое представление объекта, содержащее информацию о всех свойствах класса.

Этот класс позволяет создавать объекты, представляющие водителей, и получать информацию о них в удобном для чтения формате. Метод ToString() используется для получения строкового представления объекта класса, содержащего значения его полей.

|  |
| --- |
| [Serializable]  public class Voditeli  {  public int Id { get; set; }  public FullName FullName { get; set; }  public double Experience { get; set; }  public int ExperienceLevel { get; set; }  public double Salary { get; set; }  public int Route { get; set; }  public DayOfWeek[] Schedule { get; set; }  public Voditeli() { }  public Voditeli(int id, FullName fullName, double experience, int experienceLevel, int route, DayOfWeek[] schedule)  {  Id = id;  FullName = fullName;  Experience = experience;  ExperienceLevel = experienceLevel;  Route = route;  Schedule = schedule;  CalculateSalary();  }  private void CalculateSalary()  {  if (Experience < 5 || ExperienceLevel == 3)  {  Salary = 3600;  }  else if (Experience > 5 && Experience < 10 || ExperienceLevel == 2)  {  Salary = 4200;  }  else if (Experience > 10 || ExperienceLevel == 1)  {  Salary = 5400;  }  else  {  Salary = 1500;  }  }  public override string ToString()  {  return $"Id-{Id} FIO: {FullName} Стаж-{Experience} Класс-{ExperienceLevel} Доход-{Salary} Маршрут-{Route} График-{string.Join(", ", Schedule)}";  }  } |

Рис. 12 Описание класса Voditeli.

Класс Program представляет точку входа в программу и содержит логику пользовательского интерфейса через консоль. Рассмотрим основные аспекты этого класса:

1. Метод Main():

Метод Main() является точкой входа в программу. Здесь инициализируются основные объекты, устанавливается цвет консоли и запускается главное меню.

1. Обработка действий пользователя:

Пользователь может выбирать различные действия, вводя соответствующие символы с клавиатуры.

Для каждого действия предусмотрено свое подменю, где пользователь может выполнить определенные операции.

1. Методы добавления, вывода и удаления элементов:

Реализованы методы для добавления водителей, автобусов и маршрутов, вывода информации о них и удаления.

При выборе соответствующего пункта меню вызываются соответствующие методы класса Pattern, который реализует логику работы с данными.

1. Обработка исключений:

Реализована обработка исключений для обеспечения корректной работы программы даже при возникновении ошибок.

При возникновении исключения выводится соответствующее сообщение, а затем пользователю предлагается повторить попытку ввода.

1. Выход из программы:

При вводе символа '0' в любом из подменю происходит возврат в главное меню.

При вводе символа '0' в главном меню программа завершает свою работу.

# 2.4 Тестирование информационной системы деятельности диспетчера автобусного парка

Этот класс обеспечивает взаимодействие с пользователем через консольный интерфейс, позволяя ему выполнять различные операции с данными о водителях, автобусах и маршрутах.

Пункт «Добавить элемент» представляют собой как показано на Рисунке 13 процесс добавления нового водителя в систему. Рассмотрим каждый шаг:

1. Id: Пользователю предлагается ввести уникальный идентификатор (Id) для нового водителя. Этот идентификатор может быть использован для однозначной идентификации водителя в системе.
2. Полное имя: Пользователю предоставляется возможность ввести полное имя нового водителя, включая фамилию, имя и отчество.
3. Фамилия, Имя, Отчество: Пользователь последовательно вводит фамилию, имя и отчество нового водителя.
4. Стаж: Затем пользователю предлагается указать стаж работы нового водителя. Это может быть количество лет или месяцев, проведенных им на работе.
5. Класс: Пользователю предлагается указать класс нового водителя. Это может быть уровень квалификации или категория, определяющая его компетенцию.
6. Маршрут: Пользователю предоставляется возможность выбрать маршрут, который будет обслуживать новый водитель. Это связывает водителя с конкретным маршрутом, который он будет выполнять.
7. График: Затем пользователю предлагается указать график работы нового водителя. Представляет собой дни недели или смены, в которые водитель будет работать.
8. Сколько дней: Это запрашивается для уточнения количества дней в графике работы нового водителя.

После ввода всех необходимых данных система добавляет нового водителя в список, что позволяет использовать его информацию для управления и организации работы водителей и маршрутов в системе.

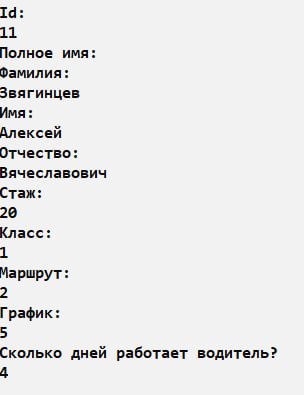


Рис.13 Добавление водителя

Представленная информация на Рисунке 14 сообщает о том, как получить список автобусов, обслуживающих определенный маршрут.

1. Маршрут-2: Сначала пользователю предоставляется информация о том, что будет выведен список автобусов, обслуживающих маршрут с идентификатором 2.
2. Введите номер маршрута, чтобы узнать автобусы: Пользователю предлагается ввести номер маршрута, для которого он хочет получить список автобусов.
3. 2: Пользователь вводит номер маршрута, заинтересовавшего его.
4. Автобусы на маршруте 2: Система выводит список автобусов, которые обслуживают указанный маршрут. В данном случае, на маршруте 2 имеется один автобус с номером T807HC.
5. Enter чтобы продолжить: Пользователю предлагается нажать клавишу Enter, чтобы продолжить работу с программой.

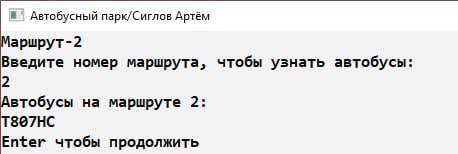


Рис. 14 Список автобусов, работающих на выбранном маршруте

Представленная информация на Рисунке 15 содержит данные о различных сотрудниках. Каждый сотрудник имеет уникальный идентификатор (Id), который используется для их однозначной идентификации. Каждый сотрудник также имеет следующие характеристики:

1. ФИО (Фамилия, Имя, Отчество): полные данные о человеке.
2. Стаж работы: количество лет или месяцев, проведенных на работе.
3. Класс: уровень или категория сотрудника.
4. Доход: заработная плата или другие финансовые показатели.
5. Маршрут: номер или идентификатор маршрута, который сотрудник выполняет или к которому он привязан.
6. График: дни недели, в которые сотрудник работает или имеет рабочие обязанности.

Эта информация, вероятно, используется для управления и организации рабочего процесса, определения расписания работы сотрудников, учета заработной платы и мониторинга производительности. Пользователю, предоставляется возможность просмотра и управления этими данными через консольное приложение, с помощью определенных команд или меню.

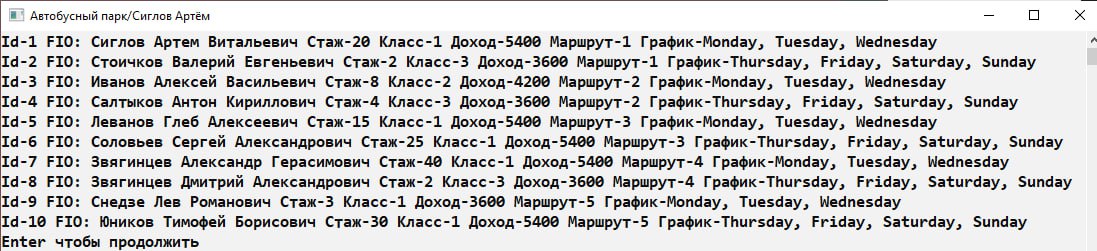


Рис. 15 Вывод информации о всех водителях

# Заключение

Разработал информационную систему управления пассажира перевозками.

В ходе выполнения курсовой работы были выполнены следующие задачи:

* Исследовать предметную область.
* Выявить проблемы и недостатки транспортной отрасли.
* Проанализировать существующие системы, их возможности и недостатки.
* Разработать информационную систему деятельности диспетчера автобусного парка.

# Список литературы