



КУРСОВОЙ ПРОЕКТ

**Тема: разработка программного модуля информационной системы,
предназначенной для эффективного управления парком автомобилей
Специальность 09.02.07 Информационные системы и программирование**

Выполнил студент(ка) группы 31ИС-21 _____ **С. В. Савченко**

Руководитель _____ **В.Ю. Назаров**

Москва 2023



УТВЕРЖДАЮ
Зам. директора КМПО

С.Ф. Гасанов

« ____ » _____ 2023 г.

ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ

по дисциплине: МДК.01.01 Разработка программных модулей
Специальность 09.02.07 Информационные системы и программирование

Студент(ка) группы 31ИС-21 Савченко Сергей

**ТЕМА: «Разработка программного модуля информационной системы,
предназначенной для эффективного управления парком автомобилей.»**

Дата выдачи задания « ____ » _____ 2023 г.

Срок сдачи работы « ____ » _____ 2023 г.

**Федеральное государственное бюджетное образовательное учреждение
высшего образования
«РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА И ГОСУДАРСТВЕННОЙ
СЛУЖБЫ ПРИ
ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»
КОЛЛЕДЖ МНОГОУРОВНЕВОГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**Задание
на курсовой проект**

Дисциплина: МДК.01.01 Разработка программных модулей

Тема: Разработка программного модуля «Система управления парком автомобилей»

Специальность: 09.02.07 Информационные системы и программирование

Группа: 31ИС-21

ФИО студента Савченко С. В.

ФИО руководителя Назаров В.Ю.

1. Проанализировать предметную область
2. Проанализировать готовые решения
3. Подготовить техническое задание
4. Подготовить план тестирования
5. Обосновать выбор инструментов и средств разработки
6. Описать реализацию технического задания
7. Выполнить тестирование

Задание выдано «_____» _____ 2023 г.

Срок выполнения «_____» _____ 2023 г.

Сроки защиты _____

Преподаватель: _____

Задание получил: _____

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ	6
1.1 Описание предметной области	6
1.2 Обзор и анализ существующих программных решений	7
2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ	12
2.1 Техническое задание	12
2.2 Общее назначение системы	16
3. РЕАЛИЗАЦИЯ ПРОЕКТА СИСТЕМЫ	19
3.1 Описание кодом основных функций модуля Authorization	19
3.2 Описание кодом основных функций модуля ChooseWindow	20
3.3 Описание основных функций модуля Admin	21
3.3.1 Описание кодом основных методов класса AdminDatabase	21
3.3.2 Описание кодом основных методов класса AdminExcel	22
3.4 Описание кодом основных функций модуля Driver	25
3.5 Тестирование приложения	27
3.5.1 Тестирование окна авторизации	27
3.5.2 Тестирование интерфейса админа	28
3.5.3 Тестирование интерфейса пользователя	29
ЗАКЛЮЧЕНИЕ	30
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	31
Приложение 1. Добавление записи в таблицу	33
Приложение 2. Удаление записи из таблицы	34
Приложение 3. Редактирование существующей записи в таблице	35
Приложение 4. План тестирования	36

ВВЕДЕНИЕ

Эффективное управление парком автомобилей представляет собой ключевой аспект для различных транспортных компаний и предприятий. В условиях динамичного развития транспортной отрасли и необходимости эффективного использования автопарка, вопросы автоматизации и оптимизации управления транспортными средствами становятся крайне важными.

Создание подобной системы обладает высокой актуальностью в современном бизнес-контексте, поскольку предприятия и организации сталкиваются с рядом вызовов, такими как необходимость проведения анализов и составления отчётностей, оптимизация ресурсов, улучшение качества обслуживания техники и многие другие, которые можно эффективно решать с использованием такой системы.

Цель данного курсового проекта - разработка программного модуля информационной системы, предназначенной для эффективного управления парком автомобилей. Такая система должна обеспечивать комплексное взаимодействие с транспортными средствами, оптимизировать процессы логистики, обеспечивать техническое обслуживание и контроль за безопасностью, с учетом современных стандартов и требований, возможность создания различных отчётов,

Актуальность проблемы создания модуля информационной системы для управления автопарком обусловлена не только ростом объемов автомобильных перевозок, но и стремительным развитием технологий, позволяющих сделать управление транспортными ресурсами более эффективным и прозрачным. Компании, занимающиеся автотранспортной деятельностью, сталкиваются с необходимостью современного подхода к управлению автопарком, который включает в себя автоматизацию, аналитику и оптимизацию процессов.

К задачам курсового проекта по созданию модуля информационной системы для управления парком автомобилей относятся:

Анализ требований и функциональности:

- Изучение потребностей и требований заказчика к информационной системе.
- Составление списка функциональных требований к программному модулю.
- Анализ существующих аналогов и определение основных возможностей.

Проектирование системы:

- Разработка структуры базы данных для хранения информации о парке автомобилей.
- Проектирование пользовательского интерфейса для удобного взаимодействия с системой.
- Определение архитектуры программного модуля.

Разработка основного функционала:

- Создание модуля для регистрации и учета автомобилей в парке.
- Разработка функционала для управления состоянием автомобилей (ремонт, техобслуживание).
- Внедрение механизмов мониторинга за расходом топлива и техническим состоянием.

Безопасность и доступ:

- Внедрение системы аутентификации и авторизации пользователей.
- Обеспечение защиты данных и обеспечение конфиденциальности информации.

Тестирование:

- Разработка тестовых случаев для проверки функционала.

Документирование:

- Написание технической документации по разработанному программному модулю.

- Подготовка руководства пользователя для использования системы.

Оптимизация и улучшение производительности:

- Идентификация узких мест в работе системы и их оптимизация.

В данном курсовом проекте объектом исследования будет являться управление парком автомобилей, а предметом исследования является разработка и реализация приложения для управления парком автомобилей с использованием средств для программирования и создания базы данных. Проект включает создание графического интерфейса, обеспечивающий удобный просмотр, добавление, удаление, редактирование данных о транспортных средствах, маршрутах, водителях, а также интеграция функционала анализа данных о маршрутах отдельного водителя или транспортного средства.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Описание предметной области

Транспортные компании осуществляют перевозки различных грузов и/или пассажиров, оперируя автопарками, водителями, и маршрутами.

Информационная система транспортной компании предназначена для эффективного управления операционной деятельностью, отслеживания движения автотранспорта, анализа данных и принятия решений.

Основные компоненты предметной области – это:

- Водители: Регистрация и учет водителей, их личной и профессиональной информации, включая данные о водительских правах, медицинских осмотрах и прочее.
- Автопарк: Учет транспортных средств, их техническое обслуживание, топливо и технические характеристики.
- Маршруты: Планирование, создание и управление маршрутами, определение географических точек и расписания движения.
- Грузы и Пассажиры: Регистрация и отслеживание грузов, пассажиров, услуг и их перемещение через систему.
- Операционная деятельность: Отслеживание текущих перевозок, статусов, контроль исполнения заказов и учет времени в пути.

Основные потребности бизнеса — это:

- Оптимизация маршрутов: Минимизация времени в пути и расходов на топливо.
- Планирование ресурсов: Равномерное распределение загрузки по водителям и автомобилям.
- Учет операций: Отслеживание движения транспорта, регистрация выполненных задач и загруженности автопарка.

- Безопасность и соблюдение законов: Ведение документации по водителям, автомобилям и выполненным маршрутам для соблюдения нормативов и требований.

1.2 Обзор и анализ существующих программных решений

Обзор и анализ существующих программных решений в области информационных систем управления парком автомобилей транспортной компании выявляет разнообразие инструментов, предназначенных для повышения эффективности управления автопарком и оптимизации операционных процессов. Важно отметить, что выбор программного решения должен быть основан на конкретных потребностях и характеристиках транспортной компании. Стоит рассмотреть некоторые из существующих программных решений:

Fleet Complete:

Особенности: управление данными о водителях, машинах, рейсах и техническом обслуживании, генерация отчетов по эффективности водителей и техническому состоянию машин, предусматривает мобильную версию приложения.

Преимущества:

- Интегрированный подход к управлению всеми аспектами автопарка.
- Гибкая конфигурация и адаптация под различные типы автопарков.
- Эффективное управление техническим обслуживанием и ресурсами.

Недостатки:

- Возможно, избыточность функций, если не требуется GPS-отслеживание.

Далее представлен интерфейс Fleet Complete (Рисунок 1).

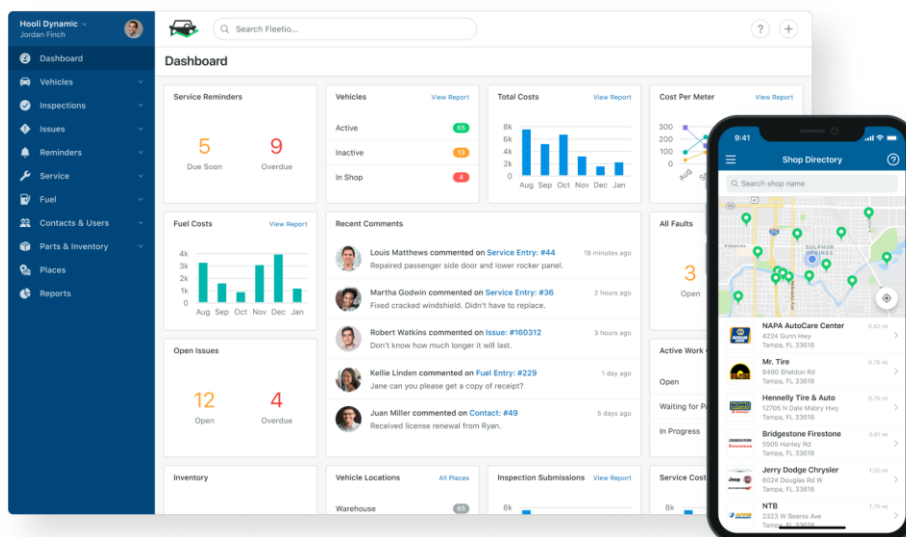


Рисунок 1 - Fleet Complete

Geotab:

Особенности: Geotab предоставляет комплексное решение для мониторинга и управления автопарком. Включает в себя GPS-отслеживание, датчики топлива, мониторинг вождения, отчеты о техническом состоянии и другие функции.

Преимущества:

- Обширные возможности аналитики, геозонирования, отслеживания топлива.

Недостатки:

- Высокие затраты на внедрение и обслуживание.

Далее представлен интерфейс Geotab (Рисунок 2).

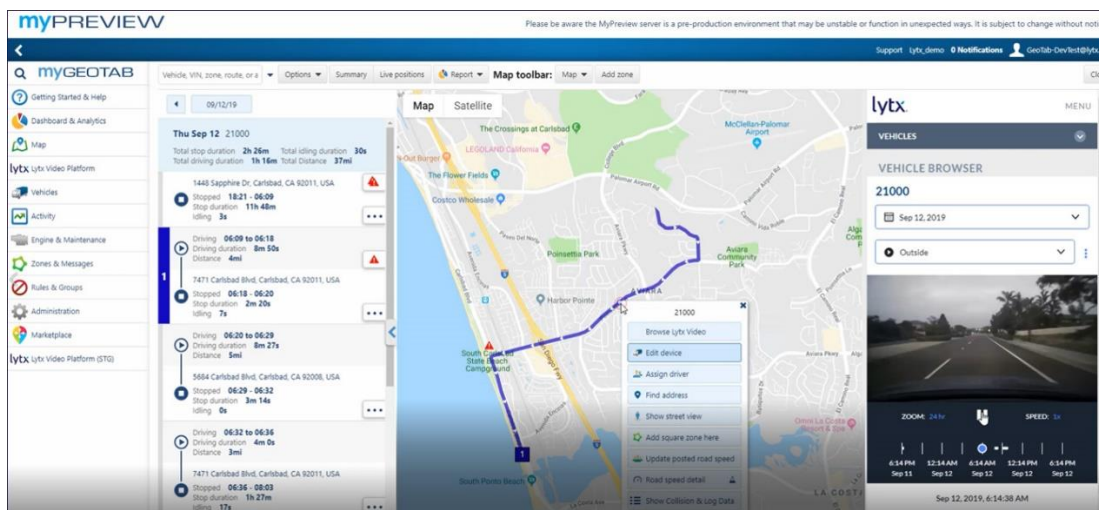


Рисунок 2 - Geotab

Verizon Connect:

Особенности: предоставляет решение для маршрутизации, мониторинга топливопотребления и анализа водительского стиля, предусматривает мобильную версию приложения.

Преимущества:

- Гибкие настройки.
- Многофункциональность.
- Возможности интеграции.

Недостатки:

- Высокие расходы на внедрение и обслуживание.

Далее представлен интерфейс Verizon Connect (Рисунок 3).

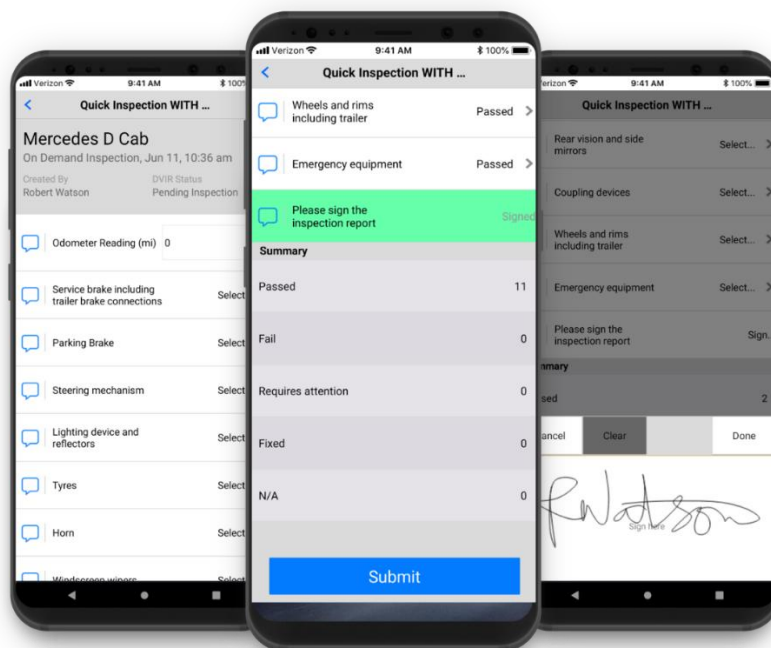


Рисунок 3 - Verizon Connect

Open-source решения:

Существуют также open-source решения, такие как Traccar, которые предоставляют базовые функции мониторинга автопарка и могут быть адаптированы под конкретные потребности компании.

Преимущества:

- Бесплатность.
- Возможность настройки под конкретные требования.

Недостатки:

- Ограниченные функциональные возможности по сравнению с коммерческими решениями.

Далее представлен интерфейс Traccar (Рисунок 4).

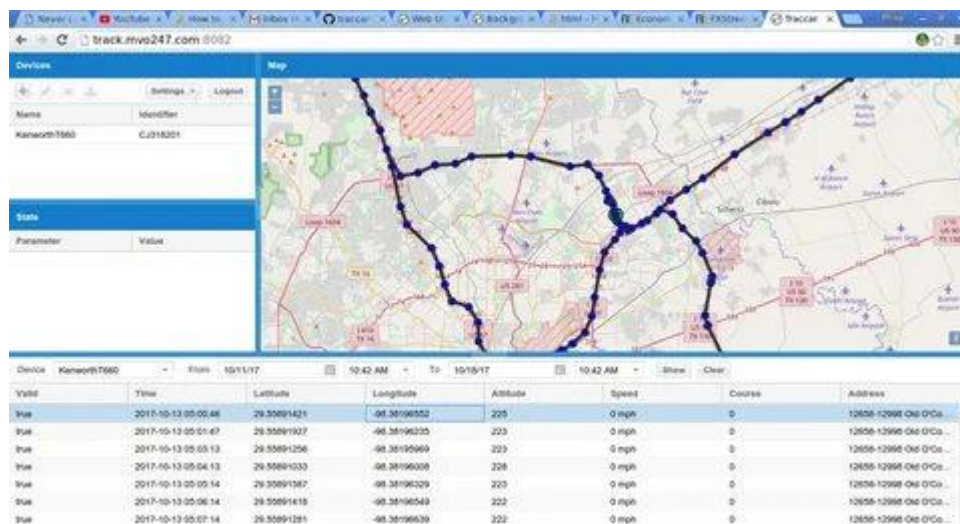


Рисунок 4 - Traccar

На основании анализа решений вывод о том, что включает разрабатываемый модуль.

Программный модуль для информационной системы для управления парком автомобилей должен включать в себя возможность управления данными о водителях, машинах, рейсах и техническом обслуживании, возможность генерации отчетов о проделанных маршрутах и техническому состоянию машин, медицинскому состоянию водителей.

2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ

2.1 Техническое задание

Целью разработки модуля информационной системы «Управление парком автомобилей» является создание графического приложения для управления данными в базе данных, а также самой базы данных для данного модуля информационной системы. Программа предоставляет пользователю удобный интерфейс для добавления, удаления и просмотра данных в таблицах базы данных.

Перечень требований к функциям, выполняемым системой – это:

Авторизация пользователя, к ней относятся:

- Создание 2 уровней допуска пользователя:
 - «admin» - имеет права на добавление, удаление и редактирование любых данных, создавать отчёты на основании этих данных;
 - «driver» – может просматривать свои личные данные и информацию о своих выездах;
- Возможность ввода логина пользователя при запуске приложения.
- Проверка учетных данных для доступа к приложению.
- В случае успешной авторизации, в зависимости от пользователя, открывается главное окно приложения для конкретного пользователя.

Главное окно:

- Пользователь «admin»:
 - Пользовательский интерфейс с кнопками "Добавить данные", "Удалить данные" и "Редактировать данные":
 - Окно добавления данных:
 - Форма для ввода данных согласно полям таблицы базы данных.

- Кнопка "Добавить" для выполнения операции вставки данных в таблицу.
- Кнопка "Отмена" для закрытия окна без добавления данных.
- Окно удаления данных:
 - Форма для выбора записи для удаления по полю ID, уникальному для каждой таблицы
 - Кнопка "Удалить" для выполнения операции удаления данных из таблицы.
 - Кнопка "Отмена" для закрытия окна без удаления данных.
- Окно редактирования данных:
 - Форма для редактирования данных согласно полям таблицы базы данных.
 - Кнопка "Подтвердить" для подтверждения операции редактирования данных.
 - Кнопка "Отмена" для закрытия окна без редактирования данных.
- Таблица для отображения данных из выбранной таблицы базы данных.
 - Выбор таблицы для вывода данных
 - Кнопка «Вывести данные» для подтверждения вывода

Далее приведена диаграмма вариантов использования для требуемого программного модуля (Рисунок 5).

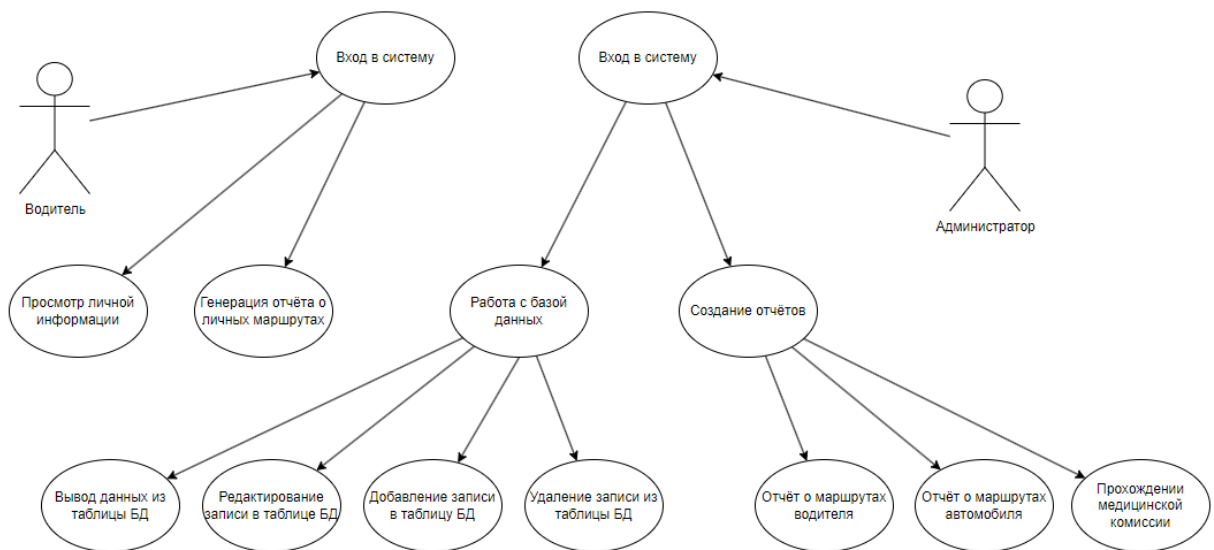


Рисунок 5 - Диаграмма вариантов использования

Интеграция с базой данных:

- Подключение к базе данных MySQL.
- Выполнение запросов SELECT, INSERT, DELETE и так далее для работы с данными.

Дополнительные функциональные требования (по желанию):

- Автоматическая подгонка размеров элементов интерфейса под размер окна.

Технические требования:

- Использование языка программирования Python.
- Использование библиотеки PyQt6 для создания графического интерфейса.
- Подключение к базе данных MySQL для выполнения запросов.
- Обработка ошибок при выполнении запросов к базе данных.
- Валидация вводимых данных в формах добавления и удаления.

Требования к базе данных:

- В базе данных программа создать следующие таблицы:
- Auto – таблица для хранения данных об автомобилях
- Driver – таблица для хранения данных о водителях

- Routs – таблица для хранения данных о маршрутах автотранспорта и водителей
- Technical service – таблица для хранения данных о техническом обслуживании автотранспорта

Далее приведена ER-диаграмма для требуемой базы данных (Рисунок 6).

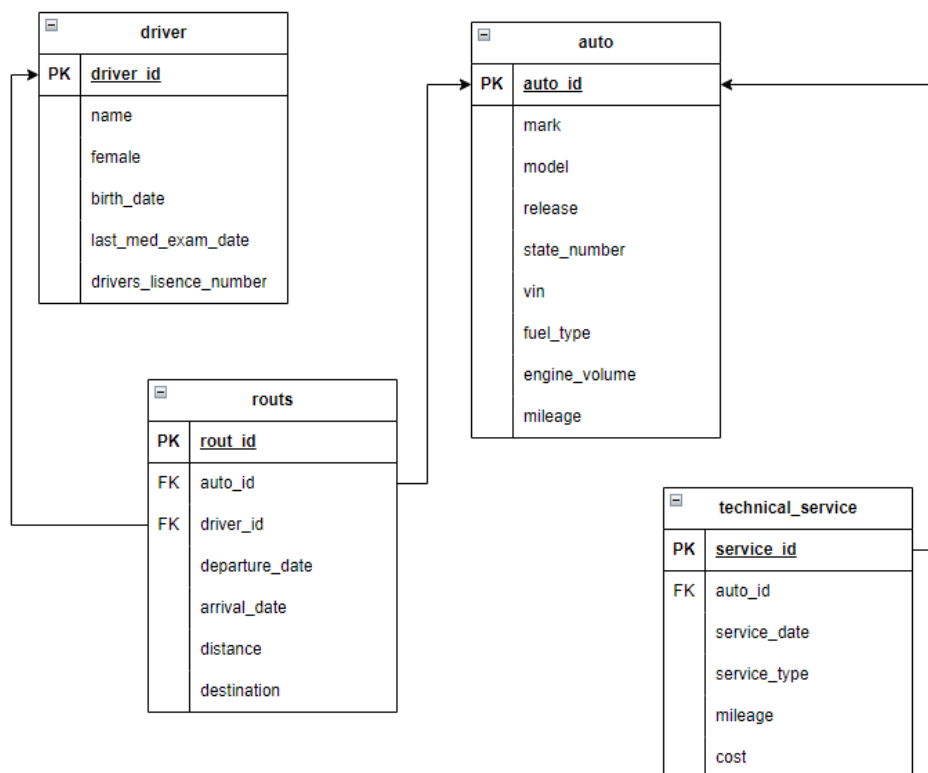


Рисунок 6 - ER-диаграмма

Далее приведена диаграмма классов для разрабатываемого программного модуля (Рисунок 7).

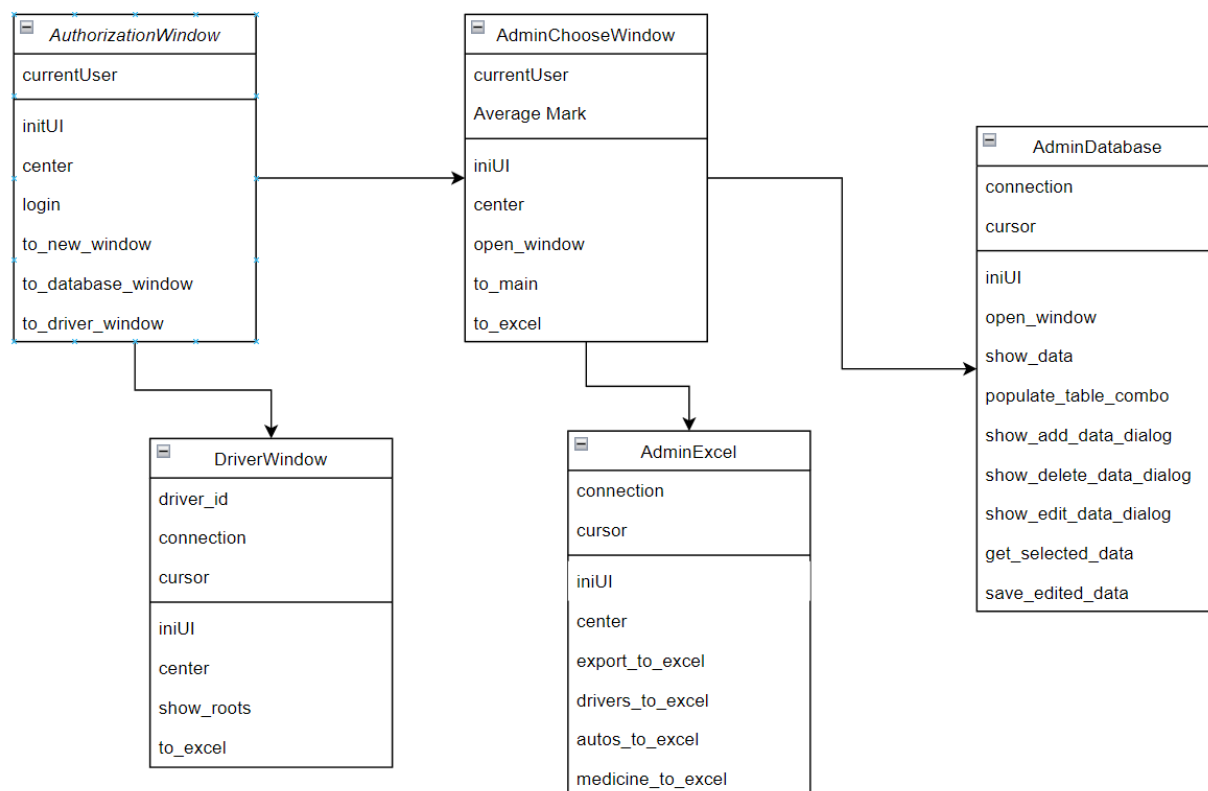


Рисунок 7 - Диаграмма классов

2.2 Общее назначение системы

Основной назначением разрабатываемого модуля является обеспечение комплексного контроля и анализа операций автопарка, повышение эффективности использования ресурсов и обеспечение безопасности и удобства для пользователей.

Ключевые функции системы:

Учет водителей:

- Добавление, редактирование и удаление информации о водителях.
- Возможность просмотра статистики и маршрутов каждого водителя.

Управление автомобилями:

- Регистрация и поддержка данных о каждом автомобиле в парке.
- Возможность добавления новых автомобилей и отслеживание их состояния.

Маршрутное планирование:

- Просмотр, добавление и редактирование данных о маршрутах.
- Мониторинг прохождения маршрутов, контроль расстояния.

Формирование отчетов:

- Создание отчетов по различным критериям, таким как пробег, расход топлива и статистика по водителям.
- Анализ данных для принятия решений и оптимизации работы автопарка.

Экспорт в Excel:

- Возможность экспорта данных о маршрутах и отчетов в формат Excel для удобного анализа и предоставления отчетности.

Система безопасности:

- Реализация механизма аутентификации и авторизации для защиты от несанкционированного доступа к данным.
- Обеспечение безопасной обработки и хранения конфиденциальной информации.

Удобный интерфейс:

- Создание интуитивно понятного и стильного пользовательского интерфейса для комфортного взаимодействия с системой.
- Легкое переключение между различными модулями и функционалом.

Ожидаемые результаты:

- Оптимизация управления ресурсами:
- Повышение эффективности использования автопарка.
- Уменьшение времени на планирование и мониторинг маршрутов.

Улучшение безопасности и контроля:

- Обеспечение надежной системы учета и контроля за водителями и автомобилями.

- Своевременное выявление и реагирование на нежелательные события.

Автоматизация процессов:

- Сокращение ручной работы при ведении данных и формировании отчетов.
- Улучшение точности и достоверности данных.

Удовлетворение потребностей пользователей:

- Обеспечение удобного и понятного интерфейса для различных категорий пользователей.
- Внедрение новых функций и улучшений в ответ на потребности и обратную связь пользователей.

3. РЕАЛИЗАЦИЯ ПРОЕКТА СИСТЕМЫ

3.1 Описание кодом основных функций модуля Authorization

Модуль MainWindow предназначен для авторизации пользователя в системе. Данный модуль содержит ряд функций для выполнения этой задачи (Рисунок 8).

```
def login(self):
    username = self.username_input.text().strip()
    password = self.password_input.text().strip()
    try:
        if username == 'admin' and password == 'admin':
            self.to_admin_window()
        elif username.startswith('driver') and username.split()[1].isdigit() and username == password:
            self.to_driver_window()
        else:
            QMessageBox.warning(self, 'Предупреждение об ошибке', 'Не существующий логин или пароль')
    except:
        QMessageBox.warning(self, 'Предупреждение об ошибке', 'Проверьте введенные значения')

def to_new_window(self, user):
    if self.currentUser:
        self.currentUser.close()
    self.currentUser = user
    self.currentUser.show()
    self.username_input.clear()
    self.password_input.clear()

def to_admin_window(self):
    self.to_new_window(AdminChooseWindow())

def to_driver_window(self):
    self.to_new_window(DriverWindow(self.username_input.text().split()[1]))
```

Рисунок 8 – Код функций для авторизации пользователя

Функция login организует принятие на вход логина и пароля, проверку их на корректность, при вводе некорректных или несуществующих данных пользователь получает уведомление о неправильных данных. В случае подтверждения данных, запустится одна из функций: to_admin_window или to_driver_window, в зависимости от введенного логина и пароля.

Переменная «currentUser», используемая в функции to_new_window служит для передачи ссылки на класс, который будет инициализирован в случае открытия того или иного окна.

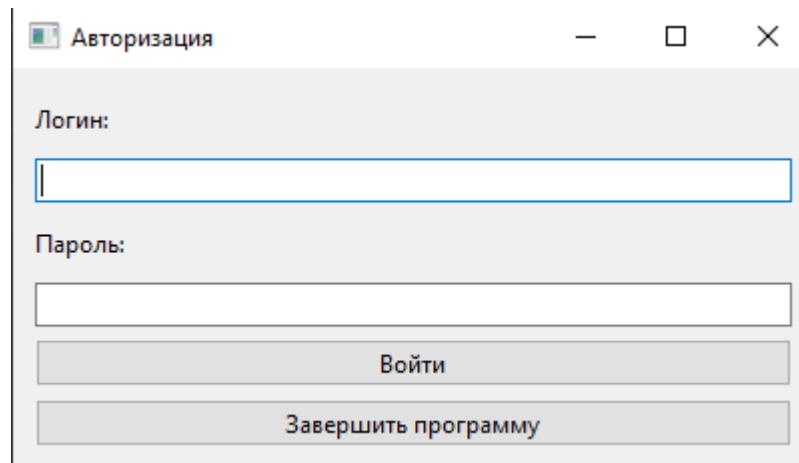


Рисунок 9 – Окно авторизации

3.2 Описание кодом основных функций модуля ChooseWindow

Модуль ChooseWindow предназначен для определения дальнейшего направления работы администратора: работа с базой данных или создание отчётов. Далее приведены функции, обеспечивающие выполнение требуемых задач (Рисунок 10).

```
def open_window(self, window_class):
    try:
        if self.currentUser:
            self.currentUser.close()
        self.currentUser = window_class
        self.currentUser.show()
    except Exception as e:
        QMessageBox.warning(self, "Предупреждение об ошибке", f'Ошибка: {str(e)}')

def to_main(self):
    self.open_window(AdminDatabase())

def to_excel(self):
    self.open_window(AdminExcel())
```

Рисунок 10 – Код функций окна для работы администратора

В модуле ChooseWindow переменная «currentUser» используется для передачи ссылки на класс, инициализируемый при запуске нового модуля, зависящего от выбора администратора.

3.3 Описание основных функций модуля Admin

Данный модуль предназначен для работы администратора в системе. Он содержит два основных класса: AdminDatabase и AdminExcel.

3.3.1 Описание кодом основных методов класса AdminDatabase

При инициализации класса происходит подключение к базе данных для дальнейшей работы с ней (Рисунок 11).

```
class AdminDatabase(QWidget):  
    def __init__(self):  
        self.connection = mysql.connector.connect(  
            host="localhost",  
            user="root",  
            password="",  
            database="inform_sys"  
        )  
  
        self.cursor = self.connection.cursor()  
        super().__init__()
```

Рисунок 11 – AdminDatabase. Подключение к базе

Класс AdminDatabase содержит 4 основные функции для взаимодействия с базой данных:

- show_data – предназначена для вывода информации из выбранной в специальном окне таблицы базы данных (Рисунок 12).
- show_add_data_dialog – предназначена для добавления новой записи в выбранную таблицу базы данных (Приложение 1).
- show_delete_data_dialog – предназначена для удаления записи из выбранной таблицы базы данных по ключевому полю «id» (Приложение 2). Полное название поля «id» индивидуально для каждой из таблиц базы данных.
- show_edit_data_dialog – предназначена редактирования выделенной записи в выбранной таблице базы данных (Приложение 3).

```

def show_data(self):
    self.data_table.clear()
    self.data_table.setRowCount(0)

    selected_table = self.table_selector.currentText()

    self.cursor.execute(f"SELECT * FROM {selected_table}")
    data = self.cursor.fetchall()

    if data:
        num_rows = len(data)
        num_cols = len(data[0])

        self.data_table.setRowCount(num_rows)
        self.data_table.setColumnCount(num_cols)

        column_headers = [description[0] for description in self.cursor.description]
        self.data_table.setHorizontalHeaderLabels(column_headers)

        for row in range(num_rows):
            for col in range(num_cols):
                item = QTableWidgetItem(str(data[row][col]))
                self.data_table.setItem(row, col, item)

```

Рисунок 12 – Код функции show_data

3.3.2 Описание кодом основных методов класса AdminExcel

При инициализации класса происходит подключение к базе данных для дальнейшей работы с ней (Рисунок 13).

```

class AdminExcel(QWidget):
    def __init__(self):
        self.connection = mysql.connector.connect(
            host="localhost",
            user="root",
            password="",
            database="inform_sys"
        )

        self.cursor = self.connection.cursor()
        super().__init__()

```

Рисунок 13 – AdminExcel. Подключение к базе

Класс AdminExcel содержит 4 основные функции формирования отчётов, основываясь на данных из таблиц базы данных:

– drivers_to_excel (Рисунок 14).

```
def drivers_to_excel(self):
    def export():
        selected_table = drivers_cb.currentText()
        self.cursor.execute(
            f'select rout_id, concat(mark, " ", model), concat(female, " ", name), '
            f'departure_date, arrival_date, distance, destination from routs '
            f'inner join driver on driver.driver_id = routs.driver_id '
            f'inner join auto on auto.auto_id = routs.auto_id '
            f'where driver.driver_id = {selected_table.split()[0]}')
        data = self.cursor.fetchall()
        headers = ["id рейса", "Авто", "Водитель", "Дата выезда", "Дата прибытия", "Расстояние", "Пункт назначения"]
        file_name = f'Маршруты {selected_table.split()[2]}.xlsx'
        self.export_to_excel(data, headers, file_name)
        dialog.close()

    dialog = QDialog(self)
    dialog.setWindowTitle('Маршруты')
    drivers_cb = QComboBox()
    self.cursor.execute('select driver_id, name, female from driver')
    driver_info = [list(sub_list) for sub_list in self.cursor.fetchall()]
    drivers_cb.addItem(f'{str(item[0])} - {item[2]} {item[1]}' for item in driver_info)
    show_table = QPushButton('Вывести в excel')
    drivers_label = QLabel('Выберите водителя:')
    dr_layout = QHBoxLayout()
    dr_layout.addWidget(drivers_label)
    dr_layout.addWidget(drivers_cb)
    show_table.clicked.connect(export)
    layout = QVBoxLayout()
    layout.addLayout(dr_layout)
    layout.addWidget(show_table)
    dialog.setLayout(layout)
    result = dialog.exec()
```

Рисунок 14 – Код функции drivers_to_excel

Функция предназначена выборки данных для составления отчёта о маршрутах одного конкретного водителя, а также для создания диалогового окна для выбора водителя для составления отчёта.

– autos_to_excel.

Код функции сильно идентичен коду функции drivers_to_excel (Рисунок 14). Меняются лишь заголовки таблиц, запрос в базу данных и название файла для сохранения.

Функция предназначена выборки данных для составления отчёта о маршрутах одного конкретного автомобиля, а также для создания диалогового окна для выбора авто для составления отчёта.

– medicine_to_excel (Рисунок 16).

```

def medicine_to_excel(self):
    self.cursor.execute(
        f'select driver_id, name, female, drivers_license_number, last_med_exam_date, '
        f'date_add(last_med_exam_date , INTERVAL 90 DAY) as next_med_exam_date, '
        f'if(datediff(now(), date_add(last_med_exam_date , INTERVAL 90 DAY)) > 0, "Не пройдена", "Пройдена") as completed_status '
        f'from driver '
    )

    data = self.cursor.fetchall()
    headers = ["id водителя", "Имя", "Фамилия", "Номер водительского удостоверения",
               "Дата последней мед.комиссии", "Дата следующей мед.комиссии", "Статус прохождения мед.комиссии"]

    file_name = 'Медицинские комиссии.xlsx'
    self.export_to_excel(data, headers, file_name)

```

Рисунок 16 – Код функции medicine_to_excel

Функция предназначена для выборки данных для составления отчёта о прохождении мед. комиссии водителями транспортных средств, определения, была ли пройдена медкомиссия, установленная на запланированную дату.

– export_to_excel (Рисунок 17).

```

def export_to_excel(self, data, headers, file_name):
    try:
        workbook = openpyxl.Workbook()
        worksheet = workbook.active
        for col_idx, header in enumerate(headers, start=1):
            column_letter = get_column_letter(col_idx)
            worksheet[f"{column_letter}1"] = header
            worksheet[f"{column_letter}1"].alignment = Alignment(horizontal='center')
        for row_idx, row_data in enumerate(data, start=2):
            for col_idx, value in enumerate(row_data, start=1):
                column_letter = get_column_letter(col_idx)
                cell = f"{column_letter}{row_idx}"
                worksheet[cell] = value
        for column in worksheet.columns:
            max_length = 0
            column = [cell for cell in column]
            for cell in column:
                try:
                    if len(str(cell.value)) > max_length: max_length = len(cell.value)
                except: pass
            adjusted_width = (max_length + 2)
            worksheet.column_dimensions[column[0].column_letter].width = adjusted_width
        workbook.save(file_name)
        QMessageBox.information(self, 'Успешно', 'Отчёт создан!')
    except: QMessageBox.warning(self, 'Ошибка', 'Ошибка при экспорте данных')

```

Рисунок 17 – Код функции export_to_excel

Функция является единым шаблоном для функций drivers_to_excel, autos_to_excel, medicine_to_excel, который принимает данные и экспортирует их в excel.

3.4 Описание кодом основных функций модуля Driver

Данный модуль предназначен для того, чтобы дать обычным пользователям, то есть водителям, возможность просматривать свои личные маршруты. При инициализации класса данного модуля производится подключение к базе данных (Рисунок 18). Так же функция `__init__` принимает на вход параметр `driver_id`, отмечающий за идентификацию пользователя в системе.

```
class DriverWindow(QWidget):
    def __init__(self, driver_id):
        super().__init__()
        self.driver_id = int(driver_id)

        self.connection = mysql.connector.connect(
            host="localhost",
            user="root",
            password="",
            database="inform_sys"
        )

        self.cursor = self.connection.cursor()
```

Рисунок 18 – DriverWindow. Подключение к базе

В классе `DriverWindow` существует функция для просмотра личной информации, которая автоматически выводится через диалоговое окно при входе пользователем в систему (Рисунок 19).

```
def init_ui(self):
    self.cursor.execute(f"DESCRIBE driver")
    columns = [column[0] for column in self.cursor.fetchall()][:-1]

    self.cursor.execute(f"SELECT * FROM driver where driver_id = {self.driver_id}")
    data = [element for sub_list in self.cursor.fetchall() for element in sub_list]

    self.cursor.execute(
        f'select ((YEAR(now()) - YEAR(birth_date)) - ((DATE_FORMAT(now(), "00-%m-%d") < DATE_FORMAT(birth_date, "00-%m-%d")))) as age '
        f'from driver where driver_id = {self.driver_id}')
    age = str(self.cursor.fetchall()[0][0])

    intro_layout = QHBoxLayout()
    fio_and_age = QLabel(f'{data[2]} {data[1]}, {age} {"род" if age[-1] == "1" else "года" if age[-1] in "234" else "лет"}')
    intro_layout.addWidget(fio_and_age)

    form_layout = QFormLayout()
    for column in list(zip(columns, data)):
        label = QLabel(column[0])
        data = QLabel(str(column[1]))
        form_layout.addRow(label, data)
```

Рисунок 19 – Код функции `init_ui` для вывода личной информации

Для просмотра пользователем своих рейсов существует функция `show_roots`. Ниже представлена часть кода, отвечающая за вывод информации о личных маршрутах на экран (Рисунок 20).

```
self.cursor.execute(f"SELECT * FROM routs where driver_id = {self.driver_id}")
data = self.cursor.fetchall()

if data:
    routs_table.setRowCount(num_rows)
    routs_table.setColumnCount(num_cols)

    column_headers = [description[0] for description in self.cursor.description]
    routs_table.setHorizontalHeaderLabels(column_headers)

    for row in range(num_rows):
        for col in range(num_cols):
            item = QTableWidgetItem(str(data[row][col]))
            routs_table.setItem(row, col, item)

    buttons_layout.addWidget(to_excel_button)
else:
    no_roots = QLabel('Информация по вашим маршрутам отсутствует')
    dialog_layout.addWidget(no_roots)
```

Рисунок 20 – Часть кода функции `show_roots`

Для вывода информации о своих маршрутах в excel существует функция `to_excel` (Рисунок 21).

```
def to_excel(self):
    workbook = openpyxl.Workbook()
    worksheet = workbook.active

    for col_idx, header in enumerate(["id рейса", "Авто", "Водитель", "Дата выезда", "Дата прибытия", "Расстояние", "Пункт назначения"], start=1):
        column_letter = get_column_letter(col_idx)
        cell = f"{column_letter}1"
        worksheet[cell] = header
        worksheet[cell].alignment = Alignment(horizontal='center')

    self.cursor.execute(
        f'select rout_id, auto_id, routs.driver_id, departure_date, arrival_date, distance, destination from routs '
        f'inner join driver on driver.driver_id = routs.driver_id '
        f'where driver.driver_id = {self.driver_id}')
    data = self.cursor.fetchall()

    for row_idx, row_data in enumerate([list(item) for item in data], start=2):
        for col_idx, value in enumerate(row_data, start=1):
            column_letter = get_column_letter(col_idx)
            cell = f"{column_letter}{row_idx}"
            worksheet[cell] = value
    workbook.save(f'Мои маршруты.xlsx')
```

Рисунок 21 – Код функции `to_excel`

3.5 Тестирование приложения

Протестируем программу по плану тестирования (Приложение 4).

3.5.1 Тестирование окна авторизации

При вводе неверных данных пользователь должен увидеть уведомление о некорректном вводе (Рисунок 22).

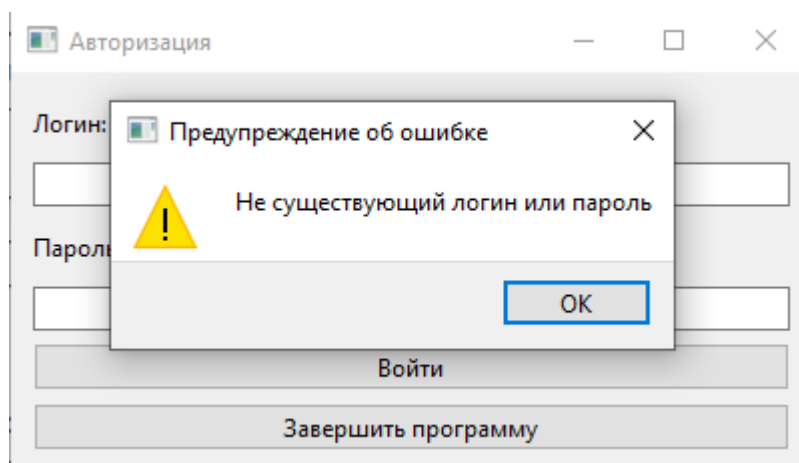


Рисунок 22 – Тестирование авторизации. Некорректные данные

Результатом является сообщение об ошибке и отчистка полей для ввода. При вводе верных данных администратора должно перенаправить на окно выбора действия (Рисунок 23).

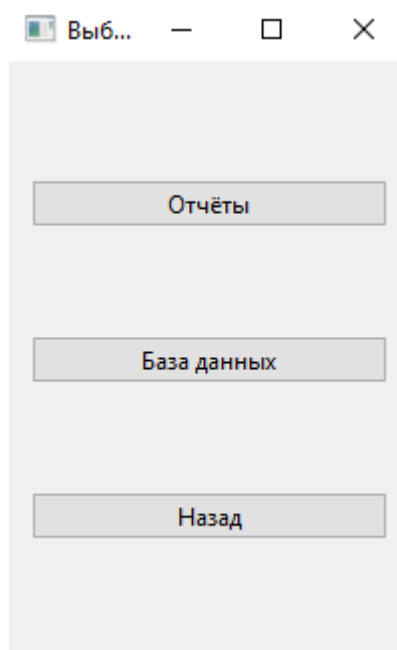


Рисунок 23 – Тестирование авторизации. Окно выбора действия

3.5.2 Тестирование интерфейса админа

В случае отсутствия сбоев, администратор сможет перейти на одно из двух окон (Рисунок 24) и (Рисунок 25).

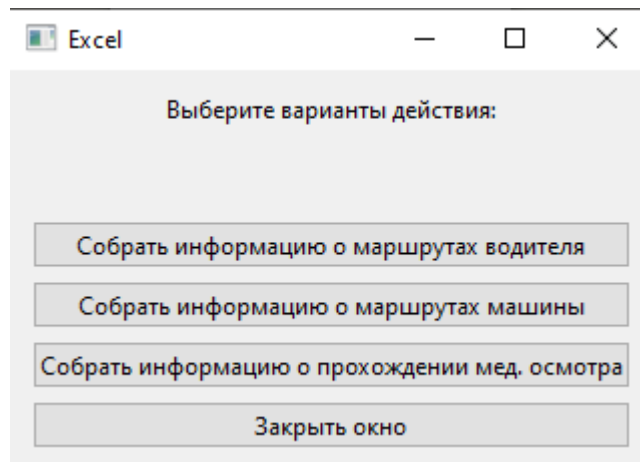


Рисунок 24 – Создание отчётов

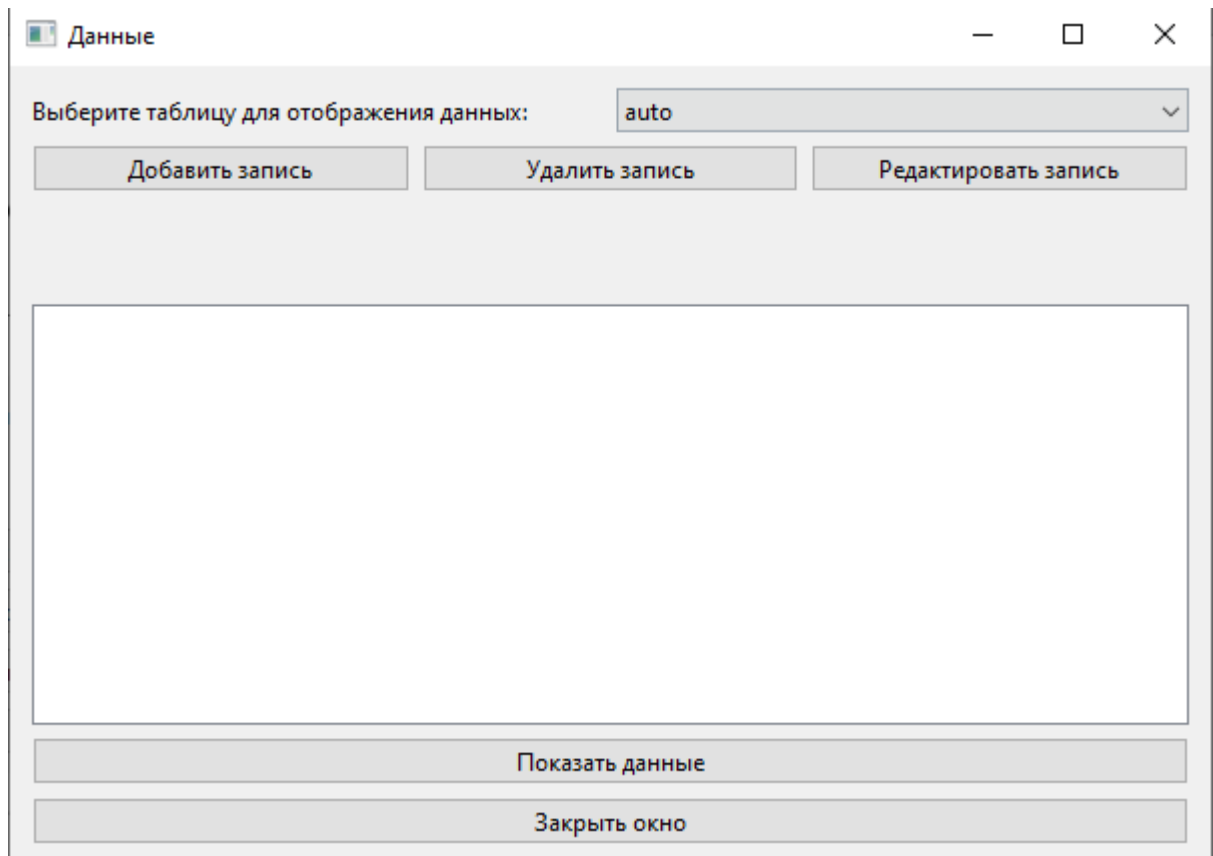
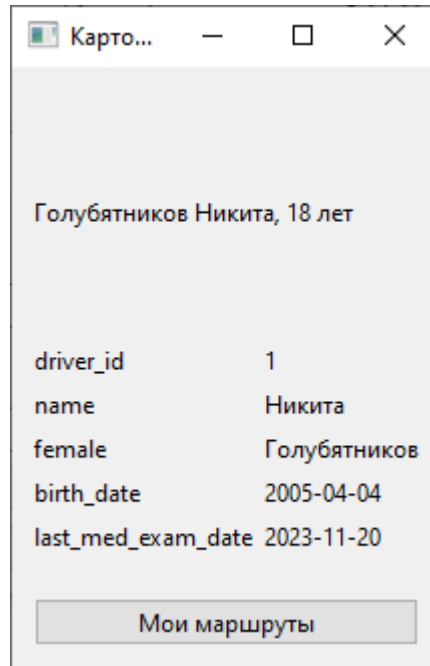


Рисунок 25 – Работа с базой данных

В случае введения верных данных пользователем, он попадет на окно с личной информацией и кнопкой, дающей возможность просмотреть свои маршруты (Рисунок 26).

3.5.3 Тестирование интерфейса пользователя



Карто...

Голубятников Никита, 18 лет

driver_id 1

name Никита

female Голубятников

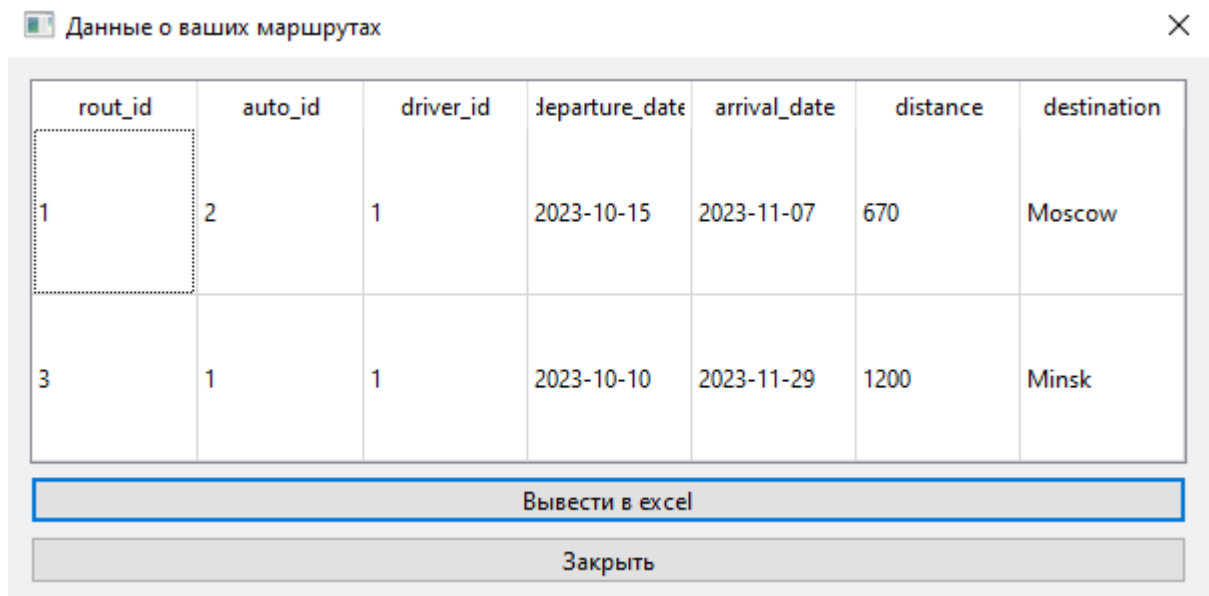
birth_date 2005-04-04

last_med_exam_date 2023-11-20

Мои маршруты

Рисунок 26 – Карточка водителя

При нажатии на кнопку «Мои маршруты» откроется окно с информацией о маршрутах пользователя (Рисунок 27).



rout_id	auto_id	driver_id	departure_date	arrival_date	distance	destination
1	2	1	2023-10-15	2023-11-07	670	Moscow
3	1	1	2023-10-10	2023-11-29	1200	Minsk

Вывести в excel

Заккрыть

Рисунок 27 – Маршруты пользователя

При нажатии на кнопку «Вывести в excel» создастся excel файл с названием «Мои маршруты.xlsx», содержащий информацию с рисунка 27.

ЗАКЛЮЧЕНИЕ

В рамках данного курсового проекта был создан программный модуль для управления парком автомобилей. В ходе разработки был осуществлен анализ требований и функциональности, связанных с управлением парком автомобилей, что позволило определить необходимые функции и особенности модуля.

Были разработаны и реализованы несколько ключевых функций, включая управление информацией об автомобилях, их состоянии, пробеге и техническом обслуживании. Модуль также предоставляет возможность просмотра статистических данных для качественного анализа работы парка автомобилей.

Модуль отличается интуитивно понятным интерфейсом и простотой использования, что позволяет сотрудникам эффективно работать с ним и улучшить общую производительность бизнеса. Он обеспечивает удобство, эффективность и надежность при работе с данными, а также способствует оптимизации бизнес-процессов, связанных с управлением автопарком.

В результате тестирования модуля была подтверждена его эффективность. Разработанный программный продукт предоставляет комплексное решение для управления автопарком, содействуя увеличению производительности и снижению операционных издержек.

Разработанный программный модуль информационной системы может быть использован в различных сферах, таких как логистика, транспортные компании и прокат автомобилей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Нормативно-правовые источники

1. ГОСТ 7.32–2017 Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления: Стандартинформ, 2017
2. ГОСТ 7.1—2003 Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления
3. ГОСТ 7.9—95 Система стандартов по информации, библиотечному и издательскому делу. Реферат и аннотация. Общие требования
4. ГОСТ 7.11—2004 (ИСО 832:1994) Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Сокращение слов и словосочетаний на иностранных европейских языках
5. ГОСТ 7.12—93 Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Сокращение слов на русском языке. Общие требования и правила
6. ГОСТ 7.80—2000 Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Заголовок. Общие требования и правила составления
7. ГОСТ 7.82—2001 Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления

Учебники, учебные пособия, статьи.

8. Калитин С.В. Инструментальные средства информационных систем: учебное пособие / С.В. Калитин. – Москва: Издательство Солон-Пресс, 2021. – 104 с.
9. Голицына, О.Л. Информационные системы: Учебное пособие /

О.Л. Голицына, Н.В. Максимов, И.И. Попов. - М.: Форум, 2016. - 352 с.

10. Панов А. И. Сетевое программирование на языке Python - М.: ДМК Пресс, 2017. - 246 с.

11. Томашевский В. А., Коновалов В. Г. Предметно-ориентированное прикладное программирование. - М.: Горячая Линия – Телеком, 2017. - 280 с.

12. Шалевич П. Загрузка данных и управление контентом в Интернете на PHP и MySQL. - М.: Трудовая Академия, 2018. - 384 с.

13. Оставько В. М., Карлунин А. И. Организация и технологии хранения данных в реляционных СУБД: Учебное пособие. - М.: БИНОМ. Лаборатория знаний, 2017. - 328 с.

14. Горишный, В. М. Основы разработки программного обеспечения: Учебное пособие. – Изд. 2-е, испр. и доп. – СПб.: БХВ-Петербург, 2018. – 400 с.

15. Медведев И. В. Сравнение предметно-ориентированного подхода и объектно-ориентированного подхода в разработке программного обеспечения // Прикладная информатика. 2019. №2 (50). 218–223 с.

16. Муратов С. В., Агошков А. О. Организация интерфейсов с предметно-ориентированными языками // Прикладная информатика. 2019. №2 (50). 224

17. Краснов И. В., Муравьев С. В., Лысицин Ю. В. Проектирование систем распределенной обработки баз данных: монография. Волгоград: Изд-во ВолгГТУ, 2020. 95 с.

18. Румбахер Р., Эттингер У. Программирование на 99%. - М.: Драйзер-Медиа, 2017. - 600 с.

19. Егiazаров Ю. И., Хлебushкин А. А. Проектирование и реализация информационных систем на СУБД FoxPro: Учебное пособие. – СПб.: Политехника, 2008. – 256 с.

20. Каплун А. В., Ахиненко П. А. Основы проектирования инфокоммуникационных систем: учебное пособие. М.: ПИК, 2020. 161 с.

```
def show_add_data_dialog(self):
    selected_table = self.table_selector.currentText()
    self.cursor.execute(f"DESCRIBE {selected_table}")
    columns = [column[0] for column in self.cursor.fetchall()]
    form_layout = QFormLayout()
    input_fields = []

    for column in columns:
        label = QLabel(column)
        input_field = QLineEdit()
        form_layout.addRow(label, input_field)
        input_fields.append(input_field)

    dialog = QDialog(self)
    dialog.setWindowTitle('Ввод данных')
    dialog.setLayout(form_layout)

    def add_data():
        try:
            values = [field.text() for field in input_fields]
            if len(values) == len(columns):
                query = f"INSERT INTO {selected_table} ({', '.join(columns)}) VALUES ({', '.join(['%s'] * len(columns))})"
                self.cursor.execute(query, values)
                self.connection.commit()

                QMessageBox.information(self, "Успешно", "Данные добавлены успешно!")
                self.show_data()
                dialog.accept()
            else:
                QMessageBox.warning(self, 'Предупреждение об ошибке',
                                    'Количество введенных значений не совпадает с количеством столбцов')
        except:
            QMessageBox.warning(self, 'Предупреждение об ошибке', 'Ошибка!')

    add_button = QPushButton('Добавить')
    form_layout.addRow(add_button)
    add_button.clicked.connect(add_data)
    result = dialog.exec()
```

```
def show_delete_data_dialog(self):
    selected_table = self.table_selector.currentText()

    form_layout = QFormLayout()
    id_input = QLineEdit()
    delete_button = QPushButton('Удалить')
    form_layout.addRow("Введите ID для удаления данных:", id_input)
    form_layout.addRow(delete_button)

    dialog = QDialog(self)
    dialog.setWindowTitle('Удаление данных')
    dialog.setLayout(form_layout)

    def delete_data():
        self.cursor.execute(f"DESCRIBE {selected_table}")
        id_column = [column[0] for column in self.cursor.fetchall()][0]

        id_value = id_input.text()

        query = f"DELETE FROM {selected_table} WHERE {id_column} = {id_value}"
        self.cursor.execute(query)
        self.connection.commit()

        QMessageBox.information(self, "Успешно", "Данные удалены успешно!")
        self.show_data()
        dialog.accept()

    delete_button.clicked.connect(delete_data)

    result = dialog.exec()
```

Приложение 3. Редактирование существующей записи в таблице

```
def show_edit_data_dialog(self):
    selected_data = self.get_selected_data()

    if selected_data:
        edit_dialog = QDialog(self)
        edit_dialog.setWindowTitle('Редактировать запись')
        layout = QFormLayout()

        input_fields = []
        for key, value in selected_data.items():
            label = QLabel(key.capitalize())
            edit_line = QLineEdit(self)
            if layout.rowCount() == 0:
                edit_line.setEnabled(False)
            edit_line.setText(str(value))
            layout.addRow(label, edit_line)
            input_fields.append(edit_line)

        save_button = QPushButton('Сохранить')
        save_button.clicked.connect(lambda: self.save_edited_data(selected_data, input_fields, edit_dialog))
        layout.addRow(save_button)

        edit_dialog.setLayout(layout)

        result = edit_dialog.exec()
    else:
        QMessageBox.warning(self, 'Предупреждение об ошибке', 'Выберите запись для редактирования')

def get_selected_data(self):
    selected_items = self.data_table.selectedItems()
    if selected_items:
        selected_row = selected_items[0].row()
        selected_data = {}
        for col in range(self.data_table.columnCount()):
            header = self.data_table.horizontalHeaderItem(col).text()
            item = self.data_table.item(selected_row, col)
            selected_data[header] = item.text()
        return selected_data

def save_edited_data(self, selected_data, input_fields, dialog):
    try:
        selected_id = list(selected_data.values())[0]
        field_id = list(selected_data.keys())[0]
        if not selected_id:
            raise ValueError("Отсутствует уникальный идентификатор для редактирования.")

        update_values = {key: field.text() for key, field in zip(selected_data.keys(), input_fields)}

        set_clause = ", ".join([f"{key} = %s" for key in update_values.keys()])
        query = f"UPDATE {self.table_selector.currentText()} SET {set_clause} WHERE {field_id} = {selected_id}"

        self.cursor.execute(query, tuple(update_values.values()))
        self.connection.commit()

        QMessageBox.information(self, "Успешно!", "Данные успешно отредактированы")
        self.show_data()
        dialog.accept()
    except:
        QMessageBox.warning(self, 'Предупреждение об ошибке', 'Ошибка при редактировании данных')
```

Приложение 4. План тестирования

№	Наименование функциональности	Наименование поля	Тестовый набор	Результат тестирования
1	Авторизация	Логин и пароль	Ввод корректных данных	Вход в систему под введенными данными
		Логин и пароль	Ввод неподходящих данных	Сообщение об ошибке: «Проверьте введенные значения»
2	Добавление записи в базу данных	Ввод данных, в соответствии с наименованиями полей	Численное значение поля id	Сообщение: «Данные успешно добавлены»
		Ввод данных, в соответствии с наименованиями полей	Ввод неподходящих данных	Сообщение об ошибке: «проверьте введенные данные»
3	Удаление записи из базы данных	Ключевое поле id, уникальное для каждой таблицы	Численное значение поля id	Сообщение: «Данные успешно удалены»
		Ключевое поле id, уникальное для каждой таблицы	Ключевое поле id, не существующее в таблице	Сообщение об ошибке: «проверьте введенное значение поля id»
4	Редактирование существующей записи	Выделенная запись для редактирования	Ввод корректных данных, подходящих под наименование поля, кроме поля id(поле id редактировать нельзя)	Сообщение: «Данные успешно отредактированы»
		Выделенная запись для редактирования	Не выделение ни одной из записей	Сообщение об ошибке: «проверьте введенное значение поля id»