

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

**«Дальневосточный федеральный университет»**

(ДВФУ)

**ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ**

**(ШКОЛА)**

**Департамент программной инженерии и искусственного интеллекта**

**КУРСОВАЯ РАБОТА**

по дисциплине «Проектирование и разработка баз данных»

по образовательной программе подготовки бакалавров по направлению

09.03.04 «Программная инженерия», профиль «Программная инженерия»

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | |  |  | Студенты группы Б9121-09.03.04 | | | | | | | |
|  |  |  | | | | Башинский Р. А. | | | |
|  | | | | | | |  |  | (подпись) | | | |  | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | |  |  | Проверил: | | | | | | | |
|  | | | | | | |  |  |  | | | | | | | |
|  | | | | | | |  |  | ученая степень, должность | | | | |  |  | |
|  | | | | | | |  |  | (подпись) | | | | |  | (ФИО) | |
|  | | | | | | | | | | | | | | | | |
| Регистрационный № | | | | | | |  |  | Защищен с оценкой | | | | | | | |
|  | | | | | | |  |  |  | | | | | | | |
|  | | | |  |  | |  |  | « |  | » |  |  | | | 2023 г. |
| (подпись) | | | |  | (ФИО) | |  |  |  | | | | | | | |
| « |  | » |  | | | 2023 г. |  |  |  | | | | | | | |

г. Владивосток

2023

Оглавление

[Введение 3](#_Toc156929151)

[1 Анализ предметной области 4](#_Toc156929152)

[2 Инфологическое проектирование 6](#_Toc156929153)

[3 Даталогическое проектирование 15](#_Toc156929154)

[3.1 Выбор СУБД 15](#_Toc156929155)

[3.1.1 Возможности СУБД 15](#_Toc156929156)

[3.1.2 Элементы СУБД 16](#_Toc156929157)

[3.1.3 Ограничения СУБД 16](#_Toc156929158)

[3.1.4 Типы данных СУБД 17](#_Toc156929159)

[3.2 Конструирование таблиц 17](#_Toc156929160)

[4 Реализация и тестирование 22](#_Toc156929161)

[4.1 Программа для водителей 22](#_Toc156929162)

[4.2 Программа для инспекторов 25](#_Toc156929163)

[4.3 Программа для администраторов 25](#_Toc156929164)

[Заключение 25](#_Toc156929165)

[Список литературы 25](#_Toc156929166)

# Введение

ГАИ города производит регистрацию автомобилей. Инспектора дорожно-патрульной службы постоянно следят за безопасностью дорожного движения. В случае нарушения правил дорожного движения к водителям применяются меры взысканий. Виды нарушений и меры взысканий определяются Кодексом об административных правонарушениях.

Для того, чтобы эффективно и доступно хранить многочисленные данные о нарушениях, водителях, автомобилях и штрафах необходимо иметь систему, которая предоставит такие возможности. Возможности такой не должны быть ограничены одним хранением, также данная система должна эти данные обрабатывать.

Цель: необходимо спроектировать базу данных ГАИ, информация которой будет использоваться для подведения статистики совершаемых водителями нарушений правил дорожного движения; выявления водителей, многократно совершающих нарушения правил дорожного движения; определения наиболее аварийных районов города, размера штрафа за совершенное нарушение и др.

Задачи:

1. Провести анализ предметной области с целью выделить основные термины предметной области и их смыл, связи между ними.
2. Провести анализ предметной области с целью выделить основные термины предметной области и их смыл, связи между ними.
3. Построить даталогическую модель, чтобы отобразить информационно-логическую модель в среду конкретной СУБД.
4. Провести тестирование реализованной базы данных.

# 1 Анализ предметной области

Процессы, происходящие в рамках данной предметной области, включают в себя регистрацию автомобилей в ГАИ, контроль соблюдения ПДД водителями, наложение взысканий за нарушения ПДД и учет оплаченных штрафов.

Информация, которую необходимо хранить в БД, включает данные о водителях, автомобилях и нарушениях ПДД, а также о взысканиях с водителей.

Таким образом, для работы с базой данных необходимо обозначить следующие роли:

1. Водитель (пользователь)
2. Инспектор (оператор)
3. Администратор

Для данных групп прав требуется выделить определенные возможности по взаимодействию с информацией, такие как “Просмотр”, “Создание”, “Редактирование” и “Удаление”. Каждая из операций над данными внутри БД подразумевает свою принадлежность к одной из этих групп взаимодействий. Таким образом, каждая роль будет иметь свой набор функций, который позволит разграничить доступ к данным для каждого объекта взаимодействия с базой данных.

Для роли “Водитель (пользователь)” определены действия и процессы, принадлежащие исключительно группе “Просмотр”. Каждый Водитель, совершивший нарушение правил дорожного движения, должен иметь информацию о том, какие штрафы и взыскания, по каким причинам и в каком месте он получил. В первую очередь — это необходимо для более удобной оплаты штрафов, что двусторонне упрощает работу как для Инспекторов ГАИ, так и для Водителей. Во вторую – можно будет формировать историю нарушений для каждого Водителя, что позволит в будущем как-то поощрять более добросовестных водителей (меньшие взыскания, ограничение предупреждением и т. п.). Также, в возможности Водителя входит просмотр списка зарегистрированных на его имя автомобилей.

Для роли “Инспектор (оператор)” определены действия и процессы, принадлежащие группам “Просмотр” и “Создание”. Это значит, что база дынных ГАИ подразумевает более удобную фиксацию нарушений во время рабочего дня Инспекторов. Инспектор может на месте создать протокол нарушения, в виде бумажного документа, и продублировать его, внеся в систему базы данных ГАИ. Если данный алгоритм действий станет обязательным, то это позволит избавиться от коррупции со стороны Инспекторов, а также позволит Водителям оплатить штраф на месте, используя не только наличные деньги, но и безналичный расчёт. В процесс фиксации нарушений будет внесена прозрачность, в лице большей открытости и информативности. Инспектора могут регистрировать и добавлять в БД новые автомобили водителей.

Для роли “Администратор” определены действия и процессы групп “Просмотр”, “Редактирование” и “Удаление”. Роль Администратора необходима для возможности изменить или удалить данные об определенных взысканиях. Возможны ситуации, когда Водитель мог оспорить решение Инспектора в суде, оплатить всего лишь часть взыскания, оплатить взыскание, но не закрыть его из-за ошибки системы, или получить незаслуженное взыскание из-за ошибки системы. Для того, чтобы как-то справляться с возникшими последствиями, вытекающих из данных ситуаций, нужно иметь определенный контингент лиц, который в праве по официальному документальному запросу\обращению и т. д. изменить информацию о взыскании или удалить взыскание из базы данных ГАИ.

Таблица 1 - Роли и права доступа.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Права  Роль | Просмотр | Создание | Редактирование | Удаление |
| Водитель | ✓ | ✖ | ✖ | ✖ |
| Инспектор | ✓ | ✓ | ✖ | ✖ |
| Администратор | ✓ | ✖ | ✓ | ✓ |

# 2 Инфологическое проектирование

Все процессы и данные так или иначе неразрывно связаны друг с другом, поэтому для понимания этих процессов, а дак же для их реализации, в частности решения задач предметной области, необходимо построить модель базы данных, в которой можно выделить следующие объекты:

1. Car – машина, которая прикреплена к водителю:

registration number

* Выдается при регистрации автомобиля в ГАИ
* Представляет собой целое число (INTEGER), отличное от нуля
* Первичное поле
* Необходим для однозначного определения автомобиля

brand

* Зависит от того, какая кампания произвела автомобиль
* Представляет собой строку кириллических и латинских знаков длиной в 100 символов
* Не может быть пустым полем
* Необходимо для определения соответствия регистрационному номеру

model

* Зависит от производителя
* Представляет собой строку кириллических и латинских знаков длиной не более 100 символов
* Не может быть пустым полем
* Необходимо для определения соответствия регистрационному номеру

color

* Зависит от производителя
* Представляет собой строку кириллических и латинских знаков длиной не более 100 символов
* Не может быть пустым полем
* Необходимо для определения соответствия регистрационному номеру

manufacture year

* Зависит от производителя
* Представляет собой структуру данных “YEAR” (4 цифры)
* Не может быть пустым полем
* Необходимо для определения соответствия регистрационному номеру

registration date

* Зависит от даты обращения в ГАИ
* Представляет собой структуру данных “DATE” (dd.mm.yyyy)
* Не может быть пустым полем
* Необходимо для определения соответствия регистрационному номеру

owner license

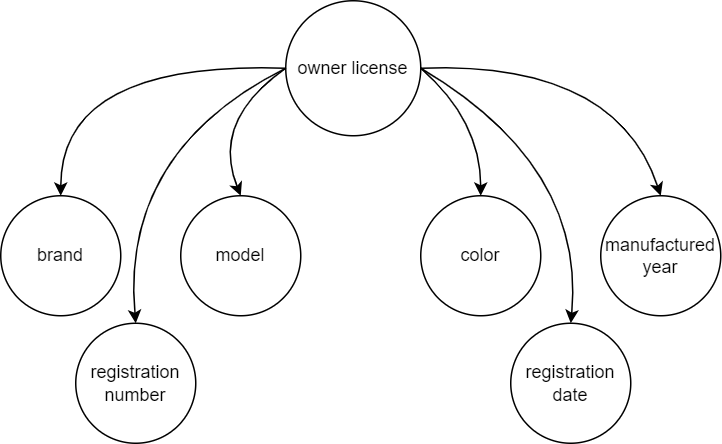
* Зависит от того, кто владеет машиной
* Представляет собой целое число (INTEGER), отличное от нуля
* Не может быть пустым полем
* Необходимо для однозначного определения владельца автомобиля

Рисунок 1 – схема отношений полей объекта Car.

1. Driver – водитель, который передвигается по дорогам на машине, соблюдая правила ПДД:

license number

* Определяется в ГАИ при выдаче прав на вождение
* Представляет собой целое число (INTEGER), отличное от нуля
* Первичное поле
* Необходимо для однозначного определения водителя

passport

* Определяется при получении документа
* Представляет собой целое число (INTEGER), отличное от нуля
* Не может быть пустым
* Необходимо для однозначного определения личности водителя

full name

* Определяется при рождении
* Представляет собой строку кириллических и латинских знаков длиной не более 255 символов
* Не может быть пустым полем
* Необходимо для определения соответствия номеру лицензии

address

* Определяется при прописке\регистрации по месту жительства
* Представляет собой текстовое поле (TEXT), состоящее из кириллических и латинских знаков
* Необходимо для административных процессов (письма, извещения)

phone number

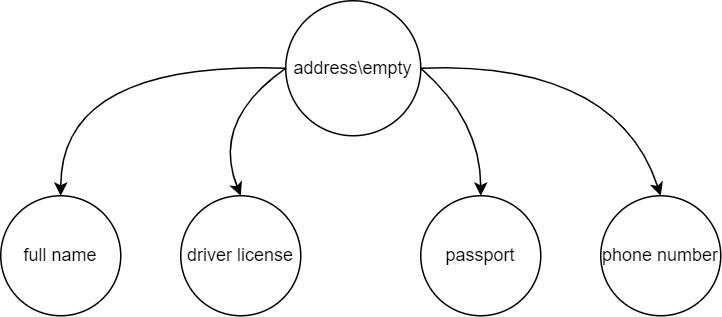
* Определяется оператором сотовой связи при покупке sim-карты
* Представляет собой целое число (INTEGER), отличное от нуля.
* Необходимо для административных процессов (sms, извещения)

Рисунок 2 – схема отношений полей объекта Driver.

1. Fine – взыскание, которое направлено на водителя за нарушение ПДД:

id

* Определяется автоматически при составлении протокола
* Представляет собой целое число (INTEGER), отличное от нуля
* Необходимо для однозначного определения протокола
* Первичное поле

violation code

* Определяется от конкретного нарушения.
* Представляет собой целое число (INTEGER), отличное от нуля
* Необходимо для однозначного определения нарушения

driver license

* Определяется при составлении протокола
* Зависит от того, кто совершил правонарушение
* Представляет собой целое число (INTEGER), отличное от нуля
* Необходимо для однозначного определения водителя, совершившего нарушение

inspector number

* Определяется ГАИ
* Зависит от того, какой инспектор составляет протокол
* Представляет собой целое число (INTEGER), отличное от нуля
* Необходимо для однозначного определения инспектора

date

* Определяется мирозданием
* Зависит от того, в какой день составлен протокол
* Представляет собой структуру данных “DATE” (dd.mm.yyyy)
* Необходимо для однозначного определения даты составления протокола

time

* Определяется мирозданием
* Зависит от того, в какое время составлен протокол
* Представляет собой структуру данных “TIMESTAMP” (hh:mm:ss)
* Необходимо для однозначного определения времени составления протокола

area

* Определяется городским устройством
* Зависит от района, в котором зафиксировано нарушение
* Представляет собой не пустое текстовое поле
* Необходима для определения места составления протокола

payment state

* Определяется ситуацией
* Зависит от оплаты
* Представляет собой булево значение “BOOLEAN”
* Необходимо для определения статуса оплаты взыскания

payment size

* Определяется законодательством
* Зависит от решения инспектора
* Представляет собой дробное поле “FLOAT”
* Необходимо для определения размера взыскания

deprivation size

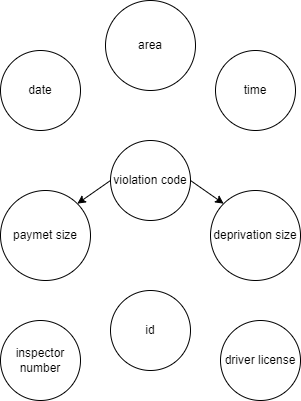
* Определяется законодательством
* Зависит от решения инспектора
* Представляет собой целочисленное число “TINYINT”, которое больше нуля, но меньше 36
* Необходимо для определения количества месяцев, на которое лишили водителя прав

Рисунок 3 – отношение полей объекта Fine.

Таблица Fine не является полностью нормализованной ввиду технического ограничения, вызванного будущей реализацией сервисов. Данное техническое ограничение может быть обусловлено особенностями фреймворков для описания front-end составляющей как web-приложений, так и приложений desktop формата.

1. Violation - нарушение, по факту которого составляется взыскание:

code

* Определяется законодательством
* Представляет собой целое число (INTEGER), отличное от нуля
* Необходимо для однозначного определения нарушения
* Первичное поле

type

* Определяется законодательством
* Представляет собой строку кириллических и латинских знаков длиной не более 100 символов
* Необходимо для определения типа нарушения

payment diapason

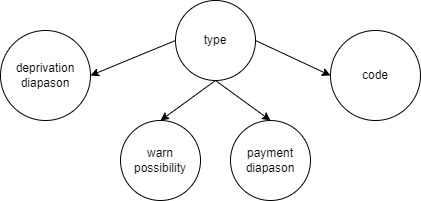
* Определяется законодательством
* Представляет собой строку кириллических и латинских знаков длиной не более 100 символов
* Необходимо для определения диапазона возможной оплаты

warn possibility

* Определяется законодательством
* Представляет собой булево значение “BOOLEAN”
* Необходимо для определения возможности вынесения предупреждения без штрафа

deprivation diapason

* Определяется законодательством
* Представляет собой строку кириллических и латинских знаков длиной не более 100 символов
* Необходимо для определения диапазона месяцев лишения прав

Рисунок 4 – отношение полей объекта Violation.

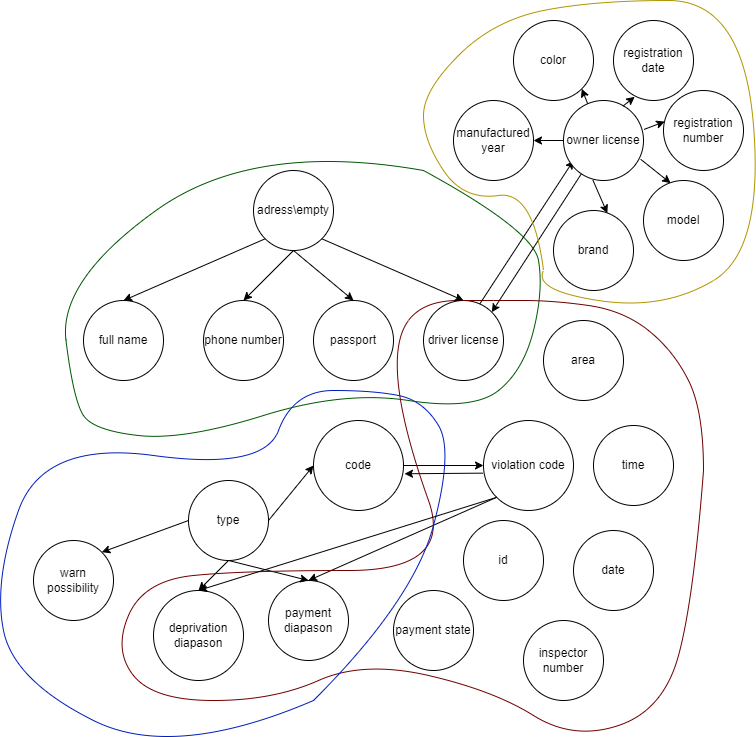


Рисунок 5 – глобальные отношения полей.

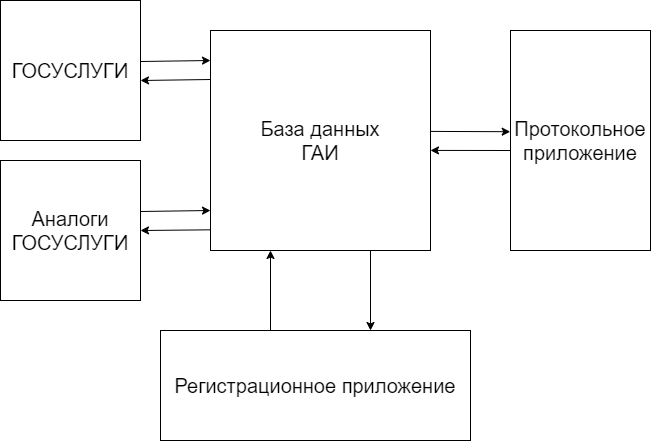
База данных подразумевает наличие структуры и ролей, работающих над ней. Это означает, что в общем, первозданном виде база данных ГАИ выступает неким “ядром” для сторонних сервисов. То есть функции, которые должны быть реализованы в рамках ролей базы данных, могут быть исполнены разным программным обеспечением. Например, просмотр данных о штрафах может исполнять сервис, аналогичный онлайн сервису “ГОСУСЛУГИ”. Для регистрации новых авто и составления протоколов штрафов могут быть задействованы две отдельные друг от друга программы. Роль, описанная при проектировании базы данных, лишь обобщает общий набор функций, но не регламентирует полное взаимодействие стороннего или внешнего программного обеспечения с данной базой данных. Однако, набор функций внутри стороннего программного обеспечения не может выходить за рамки роли, описанной при проектировании.

Рисунок 6 – диаграмма связи стороннего ПО и БД

# 3 Даталогическое проектирование

## **3.1 Выбор СУБД**

Для реализации построенной модели данных претендентом является СУБД MySQL.

MySQL относится к СУБД реляционного типа. Поддерживает формирование запросов на языке SQL, а также предоставляет как графические, так и CLI инструменты для взаимодействия с СУБД.

## **3.1.1 Возможности СУБД**

Основные возможности MySQL:

1. Поддержка различных типов данных: MySQL поддерживает все основные типы данных, включая числа, строки, даты, время, бинарные данные и т. д. Это позволяет разработчикам создавать базы данных с необходимой им структурой.

2. Транзакции: MySQL обеспечивает поддержку транзакций, что позволяет разработчикам выполнять несколько операций с базой данных как одну атомарную операцию. Это повышает надежность и стабильность работы приложения.

3. Индексы: MySQL предлагает широкий спектр индексов для оптимизации производительности запросов и ускорения поиска данных. Это особенно полезно в больших базах данных с большим количеством записей.

4. Безопасность: MySQL предоставляет механизмы безопасности для защиты данных от несанкционированного доступа, такие как аутентификация пользователей и шифрование данных.

5. Репликация: MySQL поддерживает различные типы репликации, что позволяет создавать кластеры с высокой доступностью и масштабируемостью.

## **3.1.2 Элементы СУБД**

Данная СУБД также может быть представлена как набор отдельных элементов, которые формируют ее функционал. К основным элементам баз данных в MySQL относятся:

• Таблицы — представляют собой коллекции данных с информацией об одном типе объектов.

• Поля — определяют столбцы в таблице, которые содержат информацию.

• Записи — это строки в таблице, содержащие значения полей.

Индексы - используются для ускорения поиска и сортировки данных.

• Триггеры — позволяют автоматически выполнять определенные действия при изменении данных в таблице.

• Хранимые процедуры и функции - позволяют выполнять сложные операции с данными без необходимости писать код на языке программирования.

• Представления - позволяют пользователям видеть подмножество данных из одной или нескольких таблиц без необходимости создавать копию этих данных.

## **3.1.3 Ограничения СУБД**

В MySQL существуют следующие ограничения:

• Максимальный размер таблицы составляет 65 535 \* 65 536 байт (около 64 ГБ).

• Максимальное количество полей в таблице равно 65 535.

• Максимальное количество символов в строке константы равно 1024.

• Максимальный размер строки равен 65 535 байт.

• Максимальная длина имени таблицы, индекса или триггера равна 64 байтам.

• Максимально допустимое число символов в имени базы данных равно 64.

## **3.1.4 Типы данных СУБД**

СУБД MySQL предоставляет множество различных типов данных, которые очень сильно упрощают решение даже самых сложных задач. Например, в качестве основных типов данных в MySQL можно отметить:

• INT - целочисленный тип данных, который может хранить числа от -2^31 до 2^31-1.

• FLOAT - число с плавающей точкой, которое может хранить числа с точностью до 7 цифр.

• DOUBLE - число с двойной точностью, которое может хранить числа до 15–16 значащих цифр.

• DECIMAL - десятичное число, которое позволяет хранить числа с заданной точностью и масштабом.

• DATE - тип данных для хранения даты и времени.

• TIME - тип данных для хранения времени.

• DATETIME - тип данных, объединяющий дату и время.

• TIMESTAMP - тип данных с автоматической установкой текущего времени при добавлении новой записи.

• BLOB - двоичный большой объект, используемый для хранения больших объемов данных, таких как изображения, аудио и видео.

• VARCHAR - строковый тип данных переменной длины.

• CHAR - строковый тип данных фиксированной длины.

## **3.2 Конструирование таблиц**

Каждой из составленных таблиц составим в соответствие SLQ запрос, который будет напрямую посылаться в СУБД и инициализировать таблицу с необходимыми параметрами.

Для каждой из таблиц будут указаны поля и их типы данных, в соответствии с главой 2 настоящей курсовой работы. Таким же образом, на основании логических заключений и схем отношений полей таблиц в главе 2, для каждой из таблиц будет выведена комбинация уникальности полей для однозначного определения каждой записи внутри базы данных.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 7 – SQL запрос инициализации таблицы Drivers

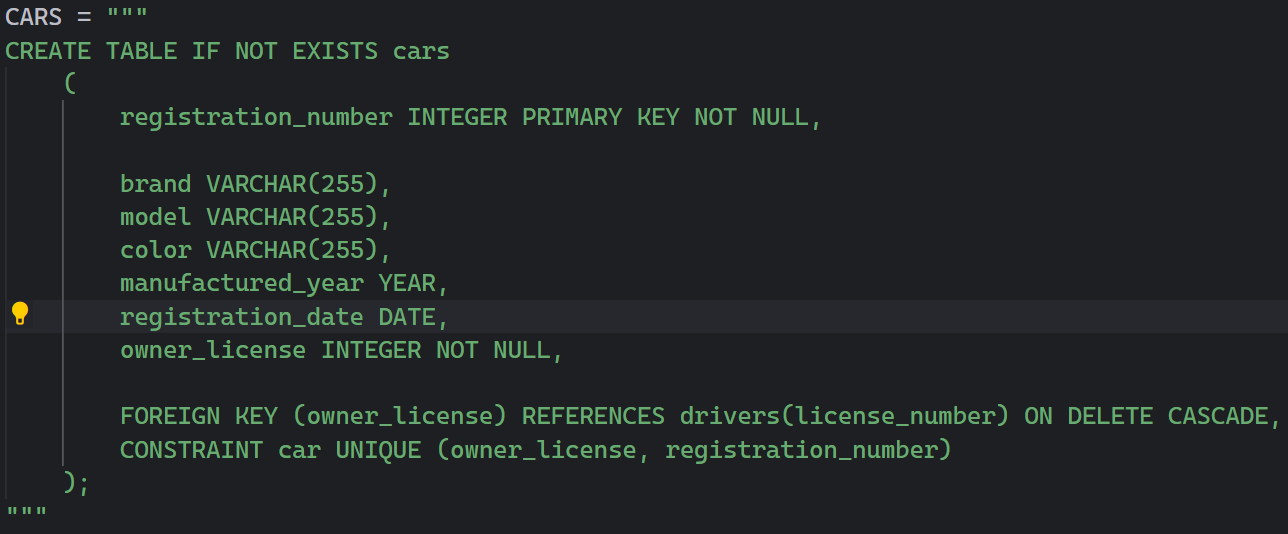
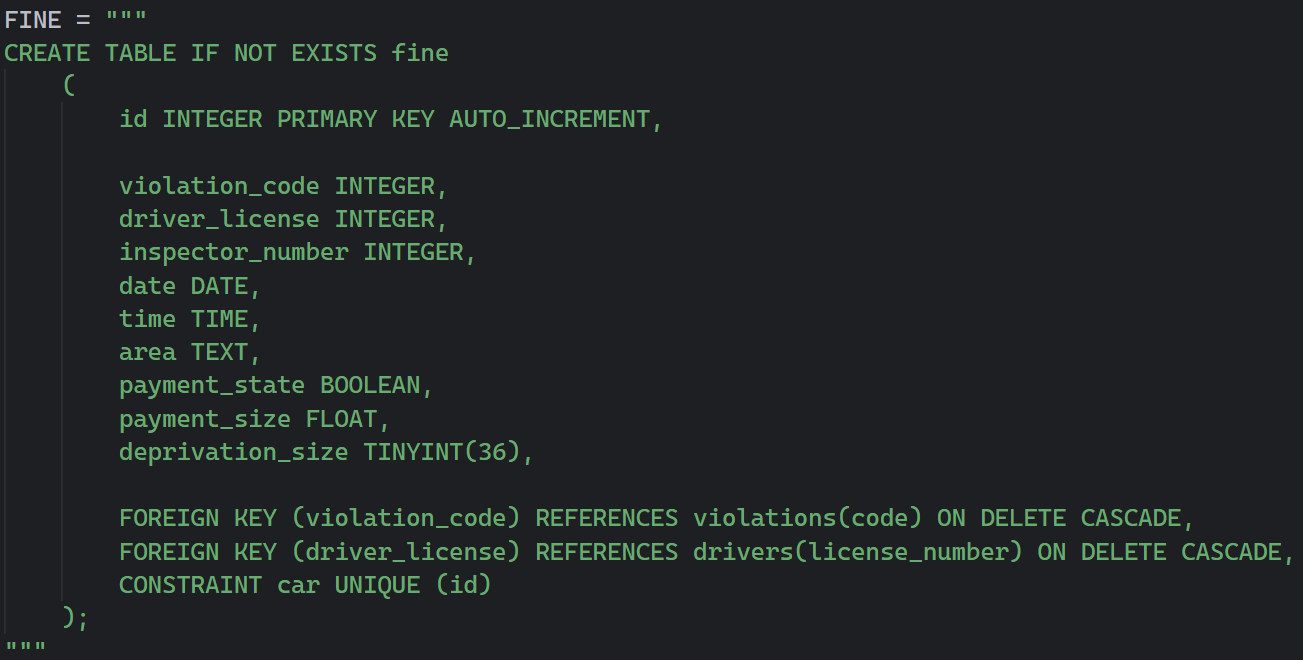


Рисунок 8 – SQL запрос инициализации таблицы Cars

Рисунок 9 – SQL запрос инициализации таблицы Fine

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 10 – SQL запрос инициализации таблицы Violations

Для таблицы Drivers (Рисунок 7) установлены следующие параметры:

* Первичный ключ – поле license\_number
* Комбинация уникальности – поля license\_number и phone\_number

Для таблицы Cars (Рисунок 8) установлены следующие параметры:

* Первичный ключ – поле registration\_number
* Комбинация уникальности – поля owner\_license и registration\_number
* Внешний ключ – owner\_license <=> drivers.license\_number
* Каскадное удаление

Для таблицы Fine (Рисунок 9) установлены следующие параметры:

* Первичный ключ – поле id
* Комбинация уникальности – поле id
* Внешний ключ – driver\_license <=> drivers.license\_number
* Внешний ключ – violation\_code <=> violations.code
* Каскадное удаление

Для таблицы Violations (Рисунок 10) установлены следующие параметры:

* Первичный ключ – поле code
* Комбинация уникальности – поле code

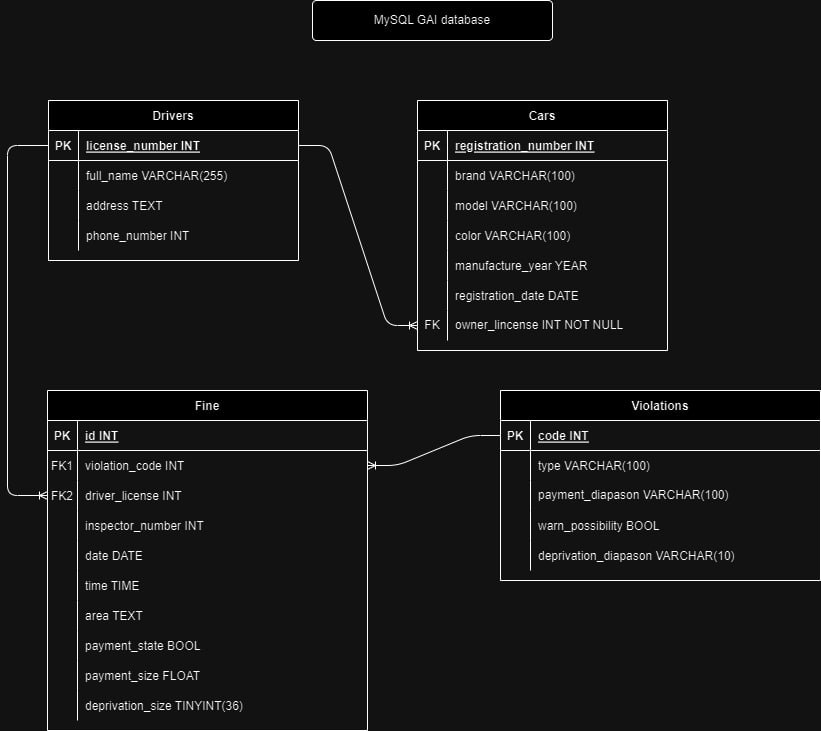
В соответствии с построенными таблицами можно создать и продемонстрировать внутренне представление структуры базы данных внутри СУБД MySQL.

Рисунок 11 – Схема отношений таблиц базы данных

Согласно данной схеме – база данных является компактной, полностью связанной, нормализованной (за исключением таблицы Fine) и инициализированной по всем правилам и критериям.

Будущее взаимодействие с базой данных будет реализовываться через API или, если делать задачу проще, через ORM (Объектно-реляционную модель). Такое решение позволит снять лишнюю нагрузку с разработчиков базы данных и перенаправить ее на сторонних разработчиков, которые будут интегрировать ORM (или API) в свой продукт.

В лице продукта может выступать любой сторонний сайт учреждения или программа, которые были целенаправленно спроектированы специально для взаимодействия с базой данных ГАИ.

# 4 Реализация и тестирование

Для демонстрации эффективной работы созданной БД была спроектирована программа-имитатор на базе WEB-приложения, которая отвечает за исполнение функций сразу трёх сервисов:

1. Программа создания протоколов для инспекторов ГАИ;
2. Программа просмотра состояния штрафов и автопарка для Водителей;
3. Программа изменения, удаления и корректировки записей базы данных для администраторов.

В данном разделе будет представленная реализация взаимодействия с базой данных с параллельной отработкой сценария тестирования, что гарантирует корректность работы программы и целостность диалогового повествования жизненного цикла программы.

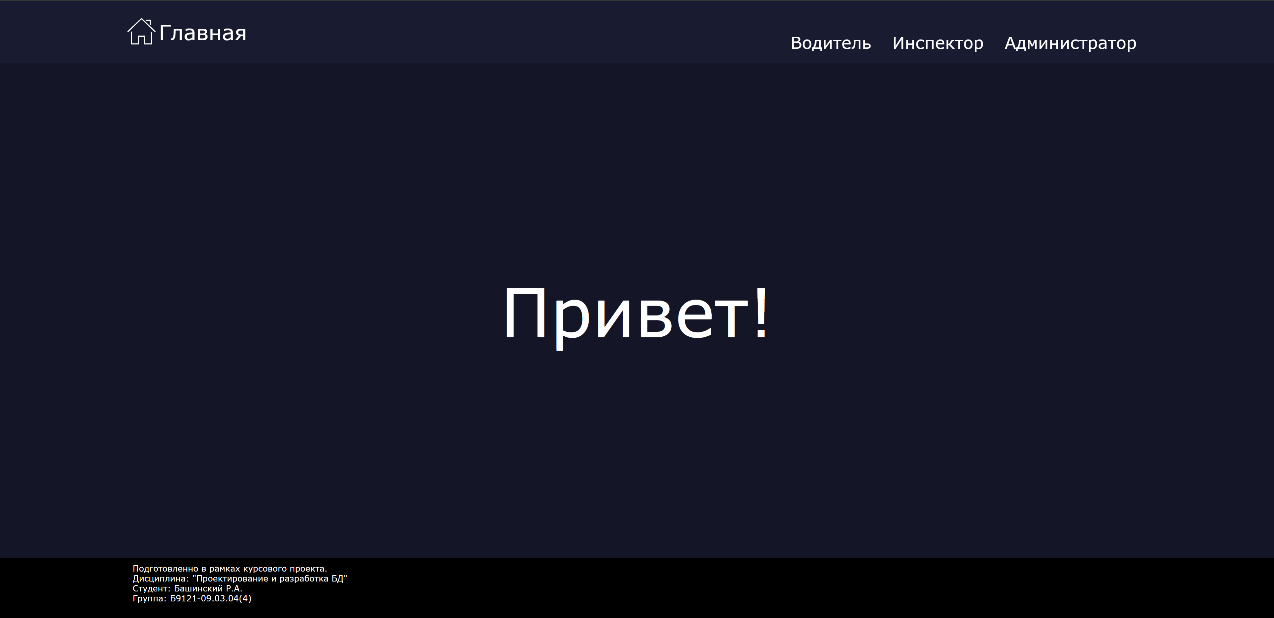
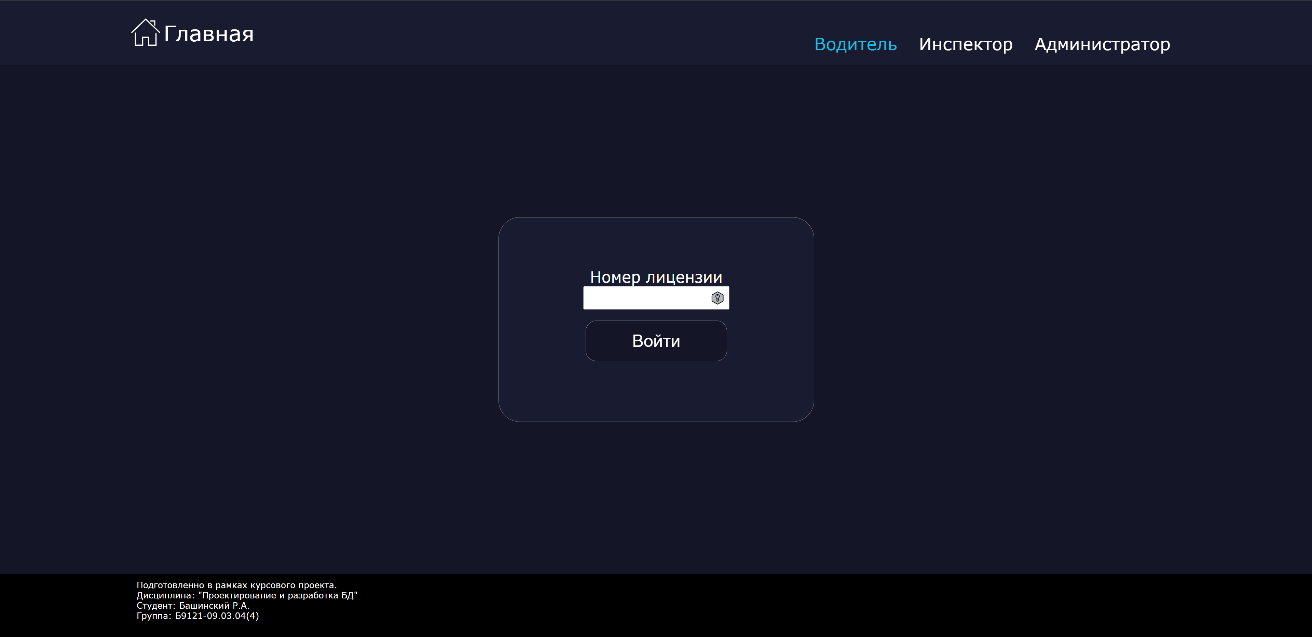
При первом запуске программы (сайта) вас поприветствуют. Изначально необходимо обратить внимание на верхнее пользовательское меню и выбрать нужную программу, которая и будет имитировать интересующие нас действия.

Рисунок 12 – Диалог 1: Приветствие

## **4.1 Программа для водителей**

Для того, чтобы начать работу с программой имитатором “Водитель”, необходимо выбрать соответствующий названию пункт в меню.

Рисунок 13 – Диалог 2: Авторизация для водителя

В данном диалоговом окне появится поле записи номера водительского удостоверения (лицензии). Необходимо вписать свой номер лицензии для авторизации и просмотра данных о машинах и штрафах.

После ввода номера лицензии нужно нажать на кнопку входа, далее страница переведет пользователя в его личный кабинет. В случае, если в системе не будет найдено совпадений (не будет найдено такого номера лицензии), на экране появится уведомление об ошибке “404”.

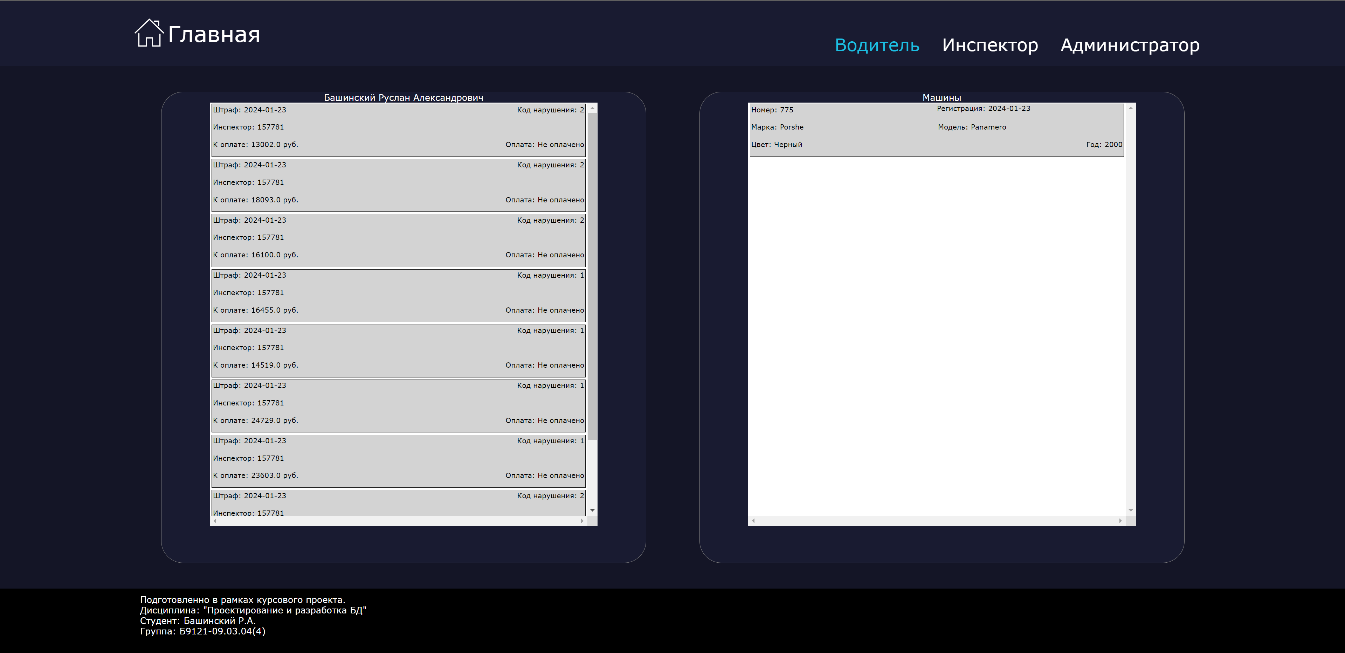


Рисунок 14 – Диалог 3: Личный кабинет пользователя

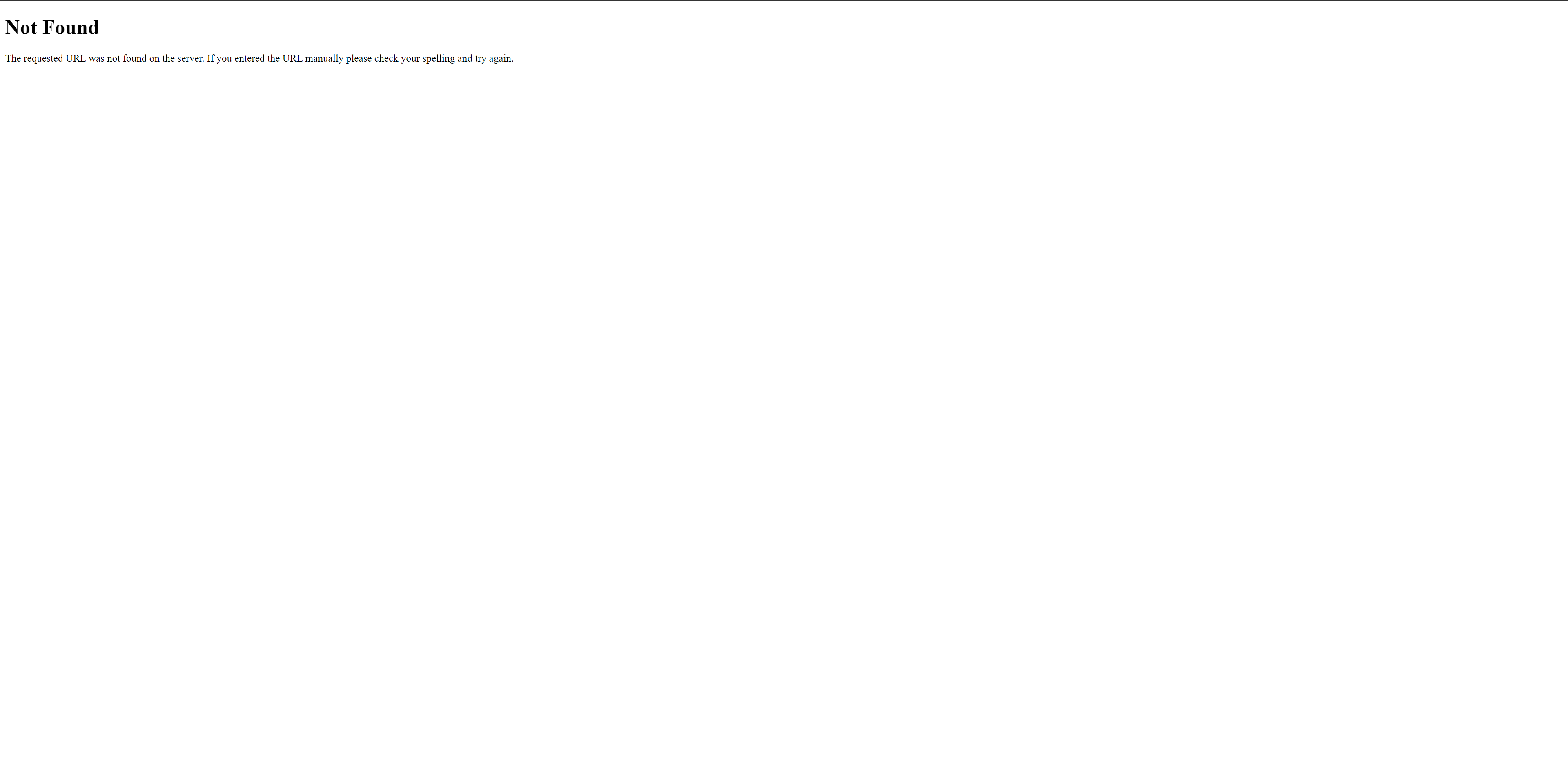


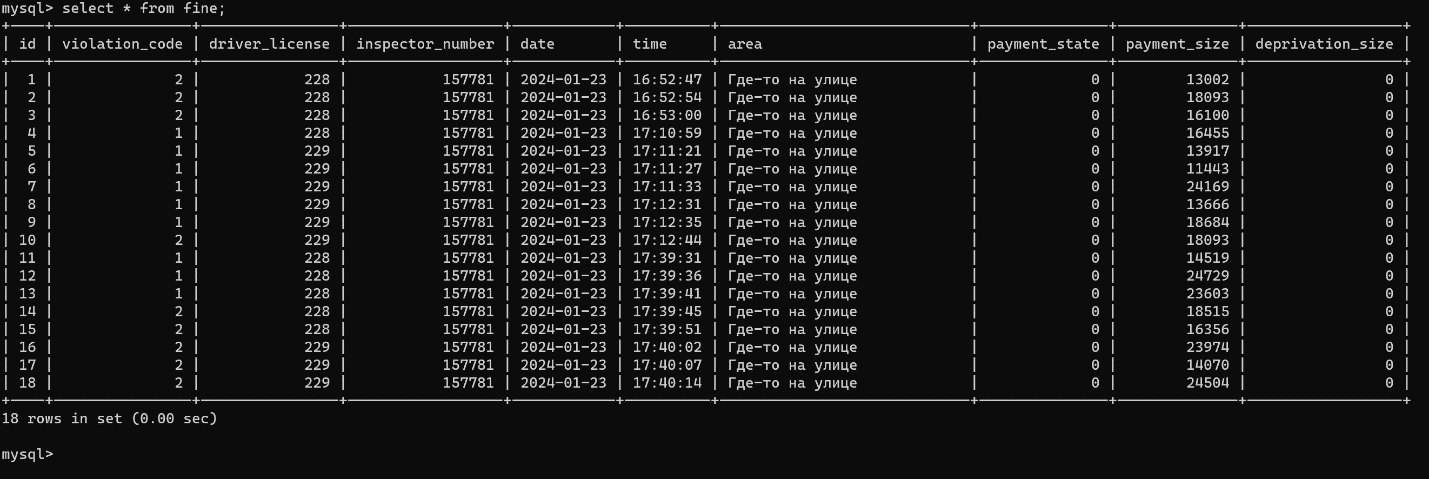
Рисунок 15 – Диалог 4: Ошибка 404

Изображение выглядит как текст, снимок экрана, Шрифт, логотип

Автоматически созданное описаниеВ случае, если кто-то решит ввести неправильные данные в поле ввода, программа сообщит ему об этом, а также укажет на причину ошибки.

Рисунок 15 – Диалог 5: Ввод некорректных данных

Как видно из диалоговых ситуаций, программа штатно отрабатывает, способна выполнить проверку целостности данных, а также проверку существование данных, которые пользователь пытается получить по запросу.

Обращаясь к самой базе данных, мы действительно видим, что при попытке получить данные о водителе “Башинский Руслан Александрович”, программа выдает именно тот результат, который ожидался.

Изображение выглядит как текст, снимок экрана, линия, черный

Автоматически созданное описаниеРисунок 16 – Записи в таблице Fines

Рисунок 17 – Записи в таблице Drivers

## **4.2 Программа для инспекторов**

## **4.3 Программа для администраторов**

# Заключение

# Список литературы