

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.02 «Информационные системы и технологии»

Направленность Информационные системы и технологии

БАКАЛАВРСКАЯ РАБОТА

на тему

**Программные средства для управления
доступом автотранспорта на закрытые
территории**

Студент

(подпись, дата)

Тишин Максим Евгеньевич

(ФИО полностью)

Руководитель

(подпись, дата)

Балашова И. Ю.

(фамилия, инициалы)

Нормоконтролёр

(подпись, дата)

Такташкин Д.В.

(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой

(подпись)

Козлов А.Ю.

(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК

(подпись)

Попова Н.А.

(фамилия, инициалы)

Пенза, 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____ А.Ю. Козлов
«25» декабря 2023 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент _____ Тишин Максим Евгеньевич _____ гр. _____ 20ВИ1 факультета _____ ВТ _____
<Фамилия, имя, отчество полностью>

2. Тема работы _____ Программные средства для управления доступом автотранспорта
на закрытые территории _____

Тема утверждена приказом ПГУ № _____ от " _____ " _____ 20 _____

3. Руководитель работы _____ к.т.н., доцент Балашова И.Ю.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Назначение: Разработка программных средств с целью автоматизации
управления доступом транспортных средств на закрытые территории

Реализуемые функции:

а) захват видеопотока с камеры;

б) предварительная обработка изображения;

в) локализация регистрационного знака в кадре;

г) фрагментация и бинаризация изображения;

д) распознавание символов регистрационного номера;

е) сравнение распознанной строки с белым списком;

ж) формирование белого списка;

з) формирование списка владельцев автомобилей.

Инструментальные средства разработки: Microsoft SQL Server 2019, Microsoft
Visual Studio Code, Visual Paradigm, ProjectLibre, Miro, Tesseract OCR

Языковые средства разработки: Python 3, SQL, UML.

Среда выполнения: ОС семейства Windows, Unix.

5. Перечень подлежащих разработке вопросов _____
- 1) Анализ предметной области и постановка задачи;
- 2) Проектирование информационной системы;
- 3) Разработка программных средств;
- 4) Тестирование программных средств

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	08.04.2024 – 13.04.2024	
2	Проектирование программных средств	20%	15.04.2024 – 25.04.2024	
3	Реализация программных средств	30%	26.04.2024 – 12.05.2024	
4	Тестирование программных средств	20%	13.05.2024 – 22.05.2024	
5	Оформление пояснительной записки	20%	23.05.2024 – 03.06.2024	

Дата выдачи задания «21» декабря 2023г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 72 листа, 44 рисунка, 6 таблиц, 23 использованных источника, 5 приложений.

КОМПЬЮТЕРНОЕ ЗРЕНИЕ, КАСКАДНЫЕ КЛАССИФИКАТОРЫ, РАСПОЗНАВАНИЕ АВТОМОБИЛЬНЫХ НОМЕРОВ, КЛИЕНТ-СЕРВЕРНАЯ АРХИТЕКТУРА, MS SQL SERVER, PYTHON, TESSERACT

Объектом разработки являются программные средства для управления доступом автотранспорта на закрытые территории в форме desktop-приложения.

Цель работы – разработка прототипа приложения, который может быть использован для автоматизации доступа автотранспорта на закрытые территории.

В качестве СУБД была использована MS SQL Server 19.

Разработка клиентской части производилась на языке программирования Python. В качестве среды разработки использовалась Microsoft Visual Studio Code.

					ПГУ 09.03.02 – 08ВР201.15 ПЗ			
Изм.	Лист	№ докум.	Подп.	Дата	Программные средства для управления доступом автотранспорта на закрытые территории. Пояснительная записка	Лит.	Лист	Листов
Разраб.		Тишин М.Е.					4	72
Пров.		Балашова И.Ю.				гр. 20ВИ1		
Реценз.								
Н. контр.		Такташкин Д.В.						
Утв.		Козлов А.Ю.						

Содержание

Введение.....	7
1 Анализ программных средств управления доступом автотранспорта на закрытые территории	8
1.1 Анализ предметной области	8
1.2 Анализ готовых решений средств контроля и управления доступом....	10
1.3 Постановка задачи	11
1.4 Методы предварительной обработки изображения	12
1.5 Методы обнаружения объектов на изображении.....	13
1.6 Методы распознавания символов	16
1.7 Планирование и оценка бюджета проекта	17
1.8 Анализ бизнес-процессов предметной области.....	19
1.9 Анализ требований	22
1.9.1 Анализ функциональных требований.....	22
1.9.2 Анализ нефункциональных требований.....	24
1.10 Выбор инструментальных средств.....	25
1.11 Системные программные средства	26
2 Проектирование системы управления доступом автотранспорта	28
2.1 Базовая архитектура системы.....	28
2.2 Логическая модель базы данных.....	29
2.3 Проектирование пользовательского интерфейса приложения	31
2.4 Проектирование программных средств для управления доступом автотранспорта на закрытые территории.....	33
3 Реализация и тестирование приложения для управления доступом автотранспорта на закрытые территории	35
3.1 Физическая модель базы данных	35
3.3 Реализация программных средств	36
3.4 Тестирование	39
3.4.1 Выбор режима тестирования	39
3.4.2 Функциональное тестирование	40

3.5 Метрики кода	41
Заключение	45
Список использованных источников	46
Приложение А (справочное) Глоссарий	49
Приложение Б (рекомендуемое) Листинг скрипта создания базы данных на языке SQL.....	51
Приложение В (рекомендуемое) Листинг клиентского приложения на языке Python	54
Приложение Г (рекомендуемое) Результаты функционального тестирования.....	65
Приложение Д (рекомендуемое) Диаграмма Ганта.....	71

Введение

Автомобиль является одним из наиболее распространенных средств передвижения. Стоянку автотранспорта стремятся осуществлять на территориях безопасных, контролируемых, что в свою очередь связано с расходами на охрану, которые могут уменьшиться благодаря применению автоматизации.

Чтобы повысить безопасность закрытых территорий, предпринимаются меры, направленные на автоматизацию контроля передвижения транспортных средств (ТС). Подобные системы следят за соблюдением правил дорожного движения без участия человека. Комплекс средств, использующихся в системах контроля, обычно включают в себя фото и видеоаппаратуру, программное обеспечение компьютерного зрения.

Идентификация ТС осуществляется с помощью присвоения ему государственного регистрационного номера (ГРН), расположенным на государственном регистрационном знаке (ГРЗ). Их обнаружение и распознавание может осуществляться в автоматическом режиме с применением технических средств компьютерного зрения [1]. Использоваться подобные системы могут на территории жилых комплексов, складов, бизнес-центров, торгово-развлекательных центров, производственных предприятий, автомобильных стоянок.

Разработка программных средств прототипа системы доступа автотранспорта на закрытые территории является целью данной работы. Для достижения цели необходимо решить следующие задачи:

- провести анализ предметной области, принципов работы аналогичных систем;
- изучить методы предварительной обработки изображения, обнаружения объектов, методы автоматического распознавания символов;
- с учетом полученной информации спроектировать, разработать и протестировать прототип приложения, который может быть применен для организации автоматизации пропускного пункта для автомобилей.

1 Анализ программных средств управления доступом автотранспорта на закрытые территории

1.1 Анализ предметной области

Формат ГРЗ регламентируется ГОСТом Р 50577-93. ГРН гражданского автомобиля состоит из серии, регистрационного номера и кода региона РФ, в котором он был зарегистрирован.

В серии используются только кириллические буквы, имеющие аналог в латинице: А, В, С, Е, Н, К, М, О, Р, Т, Х, У. Регистрационный номер представляет собой 3 цифры, кроме комбинации «000» [2]. Код регионов устанавливаются в соответствии с приказами МВД России №776 от 5 октября 2017 года [3], №130 от 11 марта 2020 года [4], №553 от 26 июля 2022 года [5], №810 от 31 октября 2022 года [6]. Пример формата стандартных ГРЗ представлен на рисунке 1.

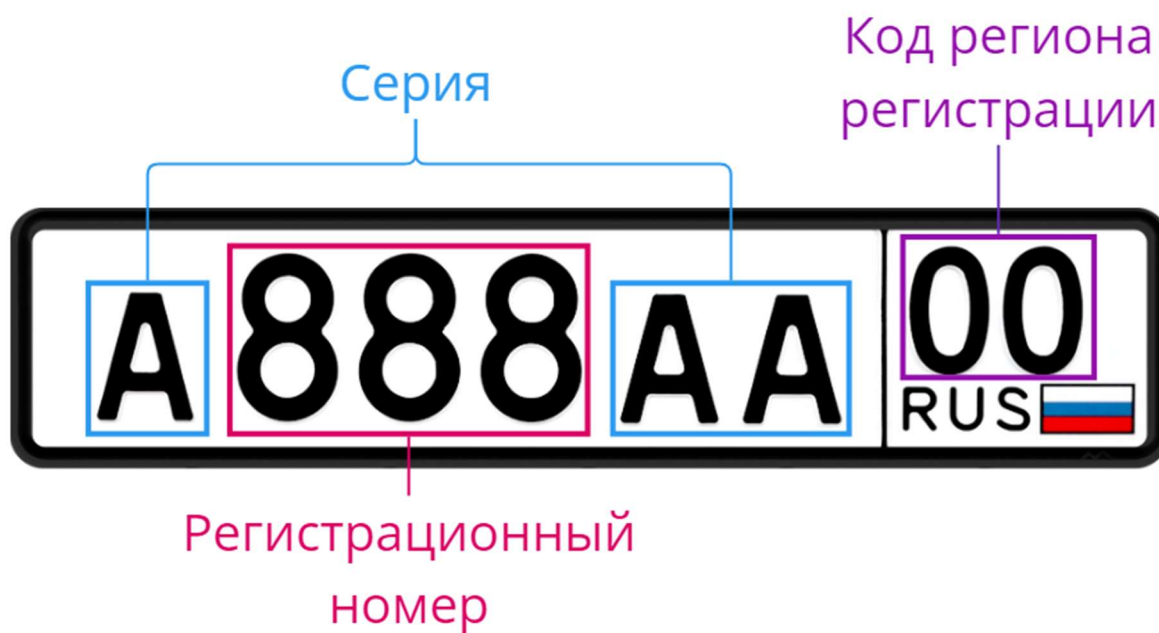


Рисунок 1 – Формат стандартного ГРЗ

Автоматическое обнаружение ГРЗ (Automatic number plate recognition – ANPR) является частью технологии компьютерного зрения (Computer vision – CV). CV средства служат для классификации и обнаружения объектов на изображении. Например, без компьютерного зрения была бы невозможна

реализация автопилота на современных автомобилях, средств контроля дорожного движения.

Оптическое распознавание символов (Optical character recognition – OCR) осуществляет автоматическое чтение символов с изображения (рукописного, печатного текста) в текстовые данные.

Архитектура систем распознавания ГРН в упрощенном виде представлена на рисунке 2.

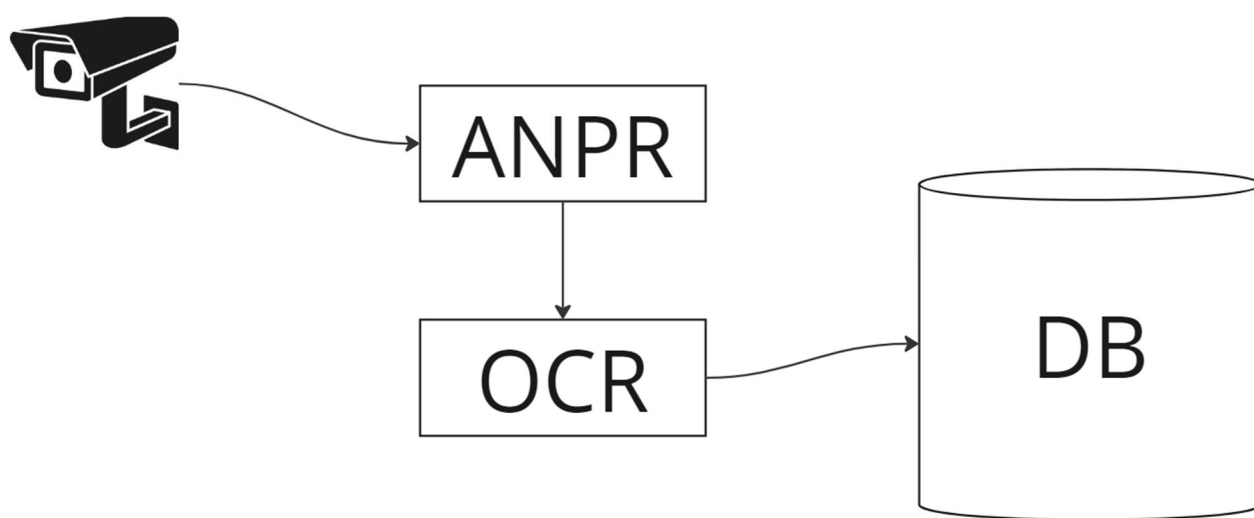


Рисунок 2 – Архитектура систем управления доступом автотранспорта

Кадр видеопотока захватывается и поступает для обнаружения в модуль обнаружения ГРЗ. После чего изображение обрезаается и поступает в модуль OCR, где происходит считывание графической информации и ее интерпретация в строковый вид.

Задачи обнаружения и распознавания осложняется внешними условиями, рамка ГРЗ может быть загрязнена или неравномерно освещена. Эти проблемы предстоит решить на уровне предварительной обработки, для увеличения точности и отказоустойчивости модулей локализации и классификации номерного знака, оптического распознавания символов государственного номера [7].

1.2 Анализ готовых решений средств контроля и управления доступом

На рынке представлено достаточно большое число решений для широкого круга спроса. Среди целевых объектов:

- склады;
- стройки;
- логистические центры;
- дворы жилых комплексов;
- территории садовых некоммерческих товариществ;
- территории производственных предприятий;
- территории торговых развлекательных комплексов.

Продукт «PASS24» холдинга «ЛАНИТ» в сфере СКУД помимо разработки занимается аудитом, монтажом аппаратуры, пусконаладочными работами и сервисным обслуживанием. Имеется калькулятор решений для объектов разного класса. В формировании конфигурации предоставляют возможность выбора класса объекта, количество КПП и проходных. Предлагаемый функционал:

- терминал распознавания лиц;
- видеонаблюдение;
- распознавание автомобильных номеров;
- мобильное бюро пропусков;
- бронирование машиномест в приложении;
- система контроля и управления доступом;
- учет рабочего времени.

Компания «Ангелы Айти» с системой «Smart Gate» предлагает аппаратно-программный комплекс, не предусматривающий использования стороннего ПО и механизмов, поддерживаются только распространяемые компанией комплексы. Помимо видеокамеры, проезд в обязательном порядке оснащается ИК-модулем. Комплекс разворачивается на базе ЭВМ «Orange». Данные предлагается хранить на облачном сервере, а локальный сервер выйдет в

полмиллиона рублей. В минимальной комплектации продукт оценивается в 180 тысяч рублей.

Корпорация «СКАЙПРОС» с системами безопасности широкого спектра назначений. В сфере средств управления доступом автотранспорта занимается распространением как копий ПО, так и интеграцией готовых решений, чаще всего входящих в комплекс более обширных систем. Стоимость только копии программного обеспечения в коробочной форме распространения в минимальной комплектации начинается с 13 тысяч рублей, стоимость аппаратного комплекса не указывается.

Проведенный анализ рынка подобных систем показывает их разнообразность и комплексность. Помимо автоматизации доступа на территорию транспортных средств разработчики предлагают введение автоматизации проходных с помощью электронных ключей, сбор статистики.

Высокая стоимость ограничивает привлекательность данных программных решений для пользователя. В большинстве решений присутствует навязанная избыточность программно-аппаратного комплекса, что заставляет заинтересованных лиц переплачивать за продукты, в которых нет необходимости.

Производители, понимая, что не всех клиентов интересует полный комплекс предлагаемых систем, допускают вариативность конфигурации.

1.3 Постановка задачи

Задачей работы является проектирование, разработка и отладка программных средств прототипа приложения, которое может быть использовано для автоматизации пропускного режима автомобилей на закрытые территории и быть пригодно для дальнейшей модификации.

Основная функция разрабатываемого ПО – определение ГРН транспортного средства. Общая последовательность предлагаемого алгоритма:

- захват видеопотока с камеры;
- выделение кадра из потока;

- цветовые манипуляции над изображением;
- обнаружение в кадре рамки с ГРЗ;
- фрагментация;
- бинаризация фрагмента;
- считывание строки ГРН;
- ликвидация возможных ошибок чтения.

Помимо самого распознавания, должно быть реализовано хранение и изменение записей в базе данных. Приложение должно быть легко в освоении для конечного пользователя, не обладающего опытом в использовании сложных технических средств.

1.4 Методы предварительной обработки изображения

Кадры видеопотока с камеры наружного наблюдения не всегда пригодны к последующему OCR. Распознавание символов может быть осложнено шумами, помехами излишней информацией. Предварительная обработка преследует две цели – уменьшить влияние шумов, и общий объем информации для оптимизации процесса.

Гауссово размытие – преобразование изображения, путем свертки пикселей по Гауссовой функции. Используется для уменьшения детализации изображения, его упрощения, работает как фильтр низких частот и ликвидирует мелкие детали изображения. С применением размытия на изображении остается меньшее количество границ объектов.

Бинаризация изображения – процесс преобразования полутонового или цветного изображения в монохромное (черно-белое), благодаря чему значительно уменьшается количество обрабатываемой информации.

Порог бинаризации может быть вычислен методом Оцу. Вычисляется гистограмма, отражающая значение яркости пикселей и их количество. Гистограмма разбивается на два класса – фон и объект. Тот класс, что встречается чаще, считается фоновым, остальные значения приписываются к объектному [8]. Для этого выполняется поэлементное преобразование вида:

$$s = \begin{cases} 1, & r \geq \Delta, \\ 0, & r \leq \Delta, \end{cases}$$

где Δ – пороговое значение яркости;

r – значение яркости пикселя до обработки;

s – значение яркости пикселя после обработки.

1.5 Методы обнаружения объектов на изображении

Гистограмма направленных градиентов (Histogram of Oriented Gradients – HOG) – один из фундаментальных приемов CV, фрагменты которого используются во многих алгоритмах. Метод базируется на определении дескриптора признаков – фрагмента изображения в виде упрощенном виде. Вычисление HOG начинается с расчета дескриптора. Расчет происходит путем вычисления горизонтальных и вертикальных градиентов, используя оператор Собеля – дискретный дифференциальный оператор, который вычисляет приближенный вектор градиента яркости изображения в точке с помощью формул:

$$g = \sqrt{g_x^2 + g_y^2} ; \quad \theta = \sqrt{\arctan \frac{g_x}{g_y}},$$

где g – величина приближенного градиента;

θ – направление приближенного градиента;

g_x, g_y – изображения, каждая точка которых содержит приближенные производные по x и по y соответственно.

И используя ядра свертки, которые могут применяться для размытия по Гауссу, выделения границ, увеличения резкости, тиснения изображений. Свертка ядра представляется формулой:

$$g(x, y) = \omega * f(x, y) = \sum_{\Delta x=-a}^a \sum_{\Delta y=-b}^b \omega(\Delta x, \Delta y) f(x - \Delta x, y - \Delta y),$$

где $g(x, y)$ – свернутое изображение;

ω – ядро свертки;

$f(x, y)$ – исходное изображение;

a, b – границы свертки.

Пиксель представляется как трехмерный вектор. Пара полученных матриц – направления и величины градиента участвует в формировании конечной гистограммы градиентов. Значение, которое попадает в ячейку выбирается на основе величины, а сама ячейка (интервал) гистограммы выбирается на основе направления. Значения, записанные в ячейки, суммируются [10].

Обнаружение объектов с использованием каскадных классификаторов – эффективный метод обнаружения объектов, основанный на машинном обучении, описанный Полом Виолой и Майклом Джонсом в их статье «Rapid Object Detection using a Boosted Cascade of Simple Features» в 2001 году. Метод аккумулирует в себе концепции функций подобных вейвлетам Хаара (рисунок 3), интегральных образов, и алгоритма «AdaBoost».

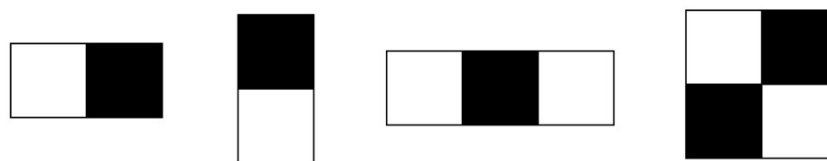


Рисунок 3 – Признаки Хаара

Признак Хаара производит значение из разности суммы интенсивности светлых пикселей и суммы интенсивности темных пикселей. Пример вычисления приведен на рисунке 4.

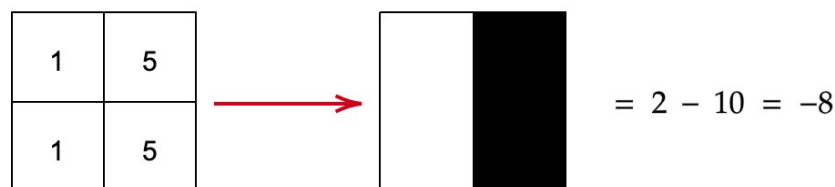


Рисунок 4 – Вычисление признака Хаара

На рисунке выше приведен пример фрагмента изображения размером 2×2 . Пара светлых пикселей слева со значениями интенсивности, равным одному, у каждого и пара темных со значениями интенсивности равными пяти. Сумма

интенсивности светлой пары равна двум, а сумма темной десяти, таким образом, разность сумм равна восьми [9].

Интегральный образ – представление изображения, позволяющее вычислить суммы значений пикселей в прямоугольной области с точкой отсчета в верхнем левом углу. Пример вычисления приведен на рисунке 5.

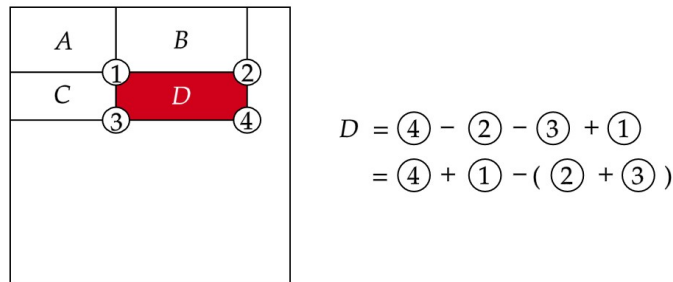


Рисунок 5 – Вычисление фрагмента изображения

Матрица, приведенная на рисунке 6, представляет собой фрагмент изображения в оттенках серого.

	x_0	x_1	x_2	x_3
y_0	1	12	45	10
y_1	6	5	11	4
y_2	3	7	10	8
y_3	5	9	4	7

Рисунок 6 – Исходная матрица

После преобразования вида:

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y'),$$

где $ii(x, y)$ – интегральный образ;

$i(x', y')$ – исходное изображение;

x, y – координаты пикселя.

Матрица будет иметь вид (рисунок 7).

	x_0	x_1	x_2	x_3
y_0	1	13	58	68
y_1	7	24	80	94
y_2	10	34	100	122
y_3	15	48	118	147

Рисунок 7 – Интегральное изображение

Алгоритм машинного обучения (Machine Learning – ML) «AdaBoost» (Adaptive Boosting) выбирает лучшее подмножество функций среди доступных. В результате работы алгоритма имеем функцию-классификатор, которая также называется функцией гипотез или прогнозирования, строгим классификатором, совокупностью линейных комбинаций слабых классификаторов.

Метод Виолы-Джонса привлекает к себе своей низкой требовательностью к вычислительной мощности эффективностью в реальном времени, низкой чувствительностью к шуму, и высокой точностью локализации.

1.6 Методы распознавания символов

Распознавание символов не целесообразно начинать без препроцессинга, описанного выше. Существует множество факторов, осложняющих работу OCR. Загрязнения, погодные условия, угол съемки, аппаратура съемки. Некоторые из них возможно нейтрализовать на предыдущих шагах. Использование инфракрасного (ИК) прожектора и системы светового обнаружения и определения дальности (Light Detection and Ranging – LIDAR) упростило бы задачу, но значительно увеличило бы стоимость [7].

После предварительной обработки и локализации начинается сегментация изображения, основанная на предположении, что текст расположен на горизонтальной линии, и символы не пересекаются друг с другом [1]. Возможны и другие случаи, которые предварительно описываемые в конфигурации. Сегментация заключается в разделении текста на отдельные символы и их сопоставление с эталоном.

1.7 Планирование и оценка бюджета проекта

«Жизненный цикл проекта» – это набор фаз, которые предстоит пройти проекту с момента начала до момента его завершения. «Фаза проекта» – набор логически связанных действий, которые завершаются достижением одной или ряда поставленных промежуточных целей [17]. Этапы реализации проекта описаны в таблице 1.

Таблица 1 – Список этапов реализации проекта

Этап	Дней	Начало	Окончание	Ресурсы
Начало реализации проекта	1	08.04	08.04	
Анализ функциональных требований	2	09.04	10.04.	Программист; Компьютер; Интернет
Анализ нефункциональных требований	3	11.04	15.04	Программист; Компьютер; Интернет
Анализ среды разработки	2	16.04	17.04	Программист; Компьютер; Интернет
Проектирование пользовательского интерфейса	4	18.04	23.04	Программист; Интернет; Компьютер
Проектирование программных средств	6	24.04	01.05	Программист; Компьютер; Интернет
Реализация пользовательского интерфейса	5	02.05	08.05	Программист; Компьютер; Интернет
Реализация программных средств	7	09.05	17.05	Программист; Компьютер; Интернет; Камера
Отладка клиентского приложения	5	20.05	24.05	Программист; Компьютер; Камера; Интернет
Функциональное тестирование	2	27.05	28.05	Программист Компьютер; Камера; Интернет
Рефакторинг	2	29.05	30.05	Программист; Компьютер; Камера; Интернет
Составление пояснительной записки	5	31.05	06.06	Программист; Компьютер; Камера; Интернет

На жизненный цикл проекта влияют особенности организации, отрасли, используемого метода разработки и технологии. У каждого проекта есть начало и окончание, но конкретные результаты и выполняемые работы могут

значительно меняться в зависимости от характера проекта. Жизненный цикл проекта обеспечивает основу для управления им и не зависит от включенных в него конкретных единиц работы. Полный набор шагов развития проекта образуют его жизненный цикл [17].

В программе Project Libre для планирования разработки программных средств управления доступом автотранспорта на закрытые территории был создан проект, начинающийся 08.04.24 и продолжающийся 44 дня. В проект были занесены выделенные выше 12 задач, для каждой задачи выделены ресурсы в лице двух разработчиков, компьютера, камеры и подключения к сети Интернет. Прочень стоимости затрат на использование ресурсов приведены в таблице 2.

Таблица 2 – Стоимость использования ресурсов

Ресурс	Тип	Стандартная ставка
Программист	Работа	300/час
Компьютер	Материал	100
Камера	Материал	100
Интернет	Материал	1

Project Libre после заполнения необходимых полей автоматически строит диаграмму Ганта для наглядного представления календарного плана. Диаграмма представлена в приложении Д.

Длительность работы над проектом составила 44 дня с учетом восьмичасового рабочего дня. Для работы привлечен один программист, компьютер с доступом в интернет и цифровая камера.

Для расчета стоимости необходимо воспользоваться встроенным генератором отчетов ProjectLibre. Сформированный отчет приведен на рисунке 9.

Dates			
Start	08.04.24 8:00	Finish	06.06.24 17:00
Baseline Start		Baseline Finish	
Actual Start		Actual Finish	

Duration			
Scheduled	44 дней	Remaining	44 дней
Baseline	0 дней	Actual	0 дней
		Percent Complete	0%

Work			
Scheduled	344 часов	Remaining	344 часов
Baseline	0 часов	Actual	0 часов

Costs			
Scheduled	102411 руб.	Remaining	102411 руб.
Baseline	0 руб.	Actual	0 руб.
		Variance	0 руб.

Рисунок 9 – Отчет стоимости проекта

Стоимость проекта составила 102411 рублей, сумма вычислена ProjectLibre автоматически исходя из указанных данных.

1.8 Анализ бизнес-процессов предметной области

С учетом задач, необходимых к реализации, и исходя из проведенного анализа предметной области, были выделены основные бизнес-процессы, которые представлены в таблице 3.

Для того чтобы наиболее полно представлять себе работу данной системы, следует провести концептуальное проектирование БД, используемое для описания предметной области, состоящей из множества сущностей и причинно-следственных связей между ними.

Таблица 3 – Бизнес-процессы

№	Бизнес-процесс	Входные данные		Выходные данные	
		Поставщик	Содержание	Потребитель	Содержание
1.	Получить список распознанных ГРН ТС	База данных	Список распознанных ГРН	Приложение	Список распознанных ГРН
2.	Получить белый список ГРН	База данных	Белый список ГРН	Приложение	Белый список ГРН
3.	Добавить нового владельца	Оператор	Данные владельца автомобиля	База данных	Данные владельца автомобиля
4.	Удалить владельца	Оператор	Телефон владельца	База данных	Сообщение об успешности
5.	Добавить автомобиль в белый список	Оператор	Данные автомобиля	База данных	Данные автомобиля
6.	Удалить автомобиль из белого списка	Оператор	ГРН владельца	База данных	Сообщение об успешности
7.	Открыть ворота	Оператор	Сигнал	Интерфейс сооружения	Звуковой сигнал
8.	Наблюдать за въездом	Клиентское приложение	Видеопоток	Оператор	Видеопоток с выделенным ГРЗ

В дальнейшем данные бизнес-процессы будут использоваться при проектировании и реализации разрабатываемого ПО системы распознавания ГРН.

Исходя из выделенных бизнес-процессов, можем обозначить основные необходимые сущности будущей базы данных (таблица 4), описать их типы и связи между ними, построить ER-диаграмму «сущность-связь» (Entity-Relationship).

Таблица 4 – Описание сущностей предметной области

№	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
1.	Автомобиль	Белый список автомобилей и их данные	Характеристическая	500	35% в год
2.	Владелец	Лицо, владеющее автомобилем	Характеристическая	400	35% в год
3.	Проезд	Список оборудованных въездов	Характеристическая	2	0% в год
4.	Распознанный ГРН	Регистрация въезда	Стержневая	1000000	1000% в год

Таким образом, сущность «Автомобиль» будет представлять из себя белый список ГРН, т. е. автомобилей, допущенных к проезду. Сущность «Владелец» будет хранить в себе данные лиц, чьи автомобили находятся в белом списке. «Проезд» будет хранить данные о проезде или КПП. «Распознанный ГРН» представляет из себя журнал (log), содержащий в себе записи о осуществленных проездах.

Описание типов связей между типами сущностей предметной области системы распознавания АРЗ представлено в таблице 5.

Таблица 5 – Описание связей между сущностями

№	Тип сущности	Тип сущности	Описание	Мощность
1.	Автомобиль	Распознанный ГРН	ГРН какого автомобиля распознан	1:M
2.	Владелец	Автомобиль	Кто владеет автомобилем	1:M
3.	Проезд	Распознанный ГРН	Где зафиксирован въезд	1:M

Концептуальная модель предметной области представлена в виде диаграммы «сущность-связь», на которой выделены сущности, их атрибуты и связи между ними (рисунке 10).

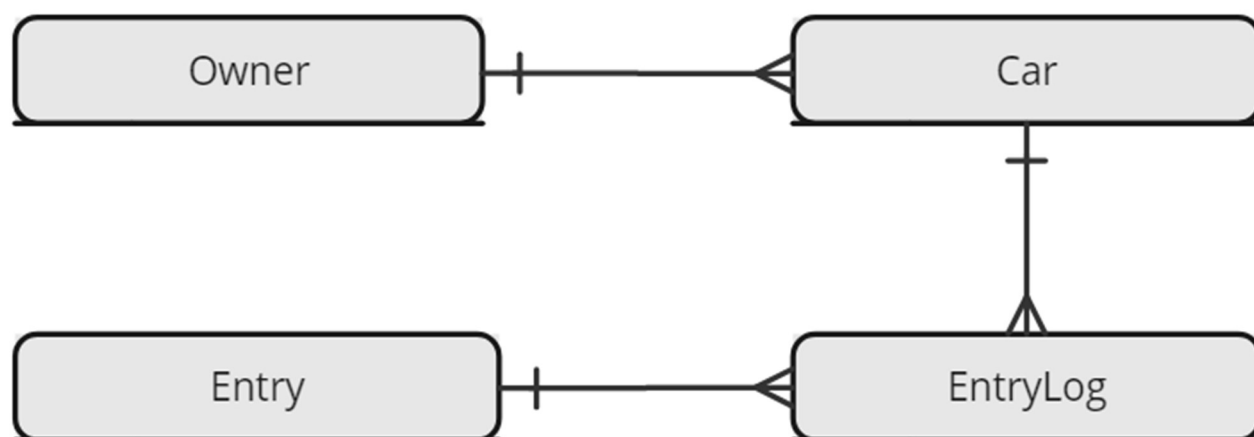


Рисунок 10 – Диаграмма «сущность-связь»

На данной диаграмме (смотри рисунок 10) представлены четыре сущности: «Owner», представляющая владельцев ТС, «Car» – белый список автомобилей, «EntryLog» – распознанные ГРН, «Entry» – проезд.

1.9 Анализ требований

1.9.1 Анализ функциональных требований

Основными функциональными требованиями системы программных средств управления доступом транспортных средств на закрытые территории являются:

- захватить кадр из видеопотока камеры;
- обнаружить ГРЗ;
- считать ГРН;
- проверить правильность распознанного ГРН;
- добавить в БД информацию о распознанном ГРН;
- сравнить правильный ГРН с белым списком;
- послать сигнал для инженерного-заградительного сооружения;
- редактировать белый список;
- редактировать данные о владельцах ТС.

Для более полного понимания функционирования системы на основе функциональных требований и типов пользователей была спроектирована диаграмма вариантов использования, представленная на рисунке 11.

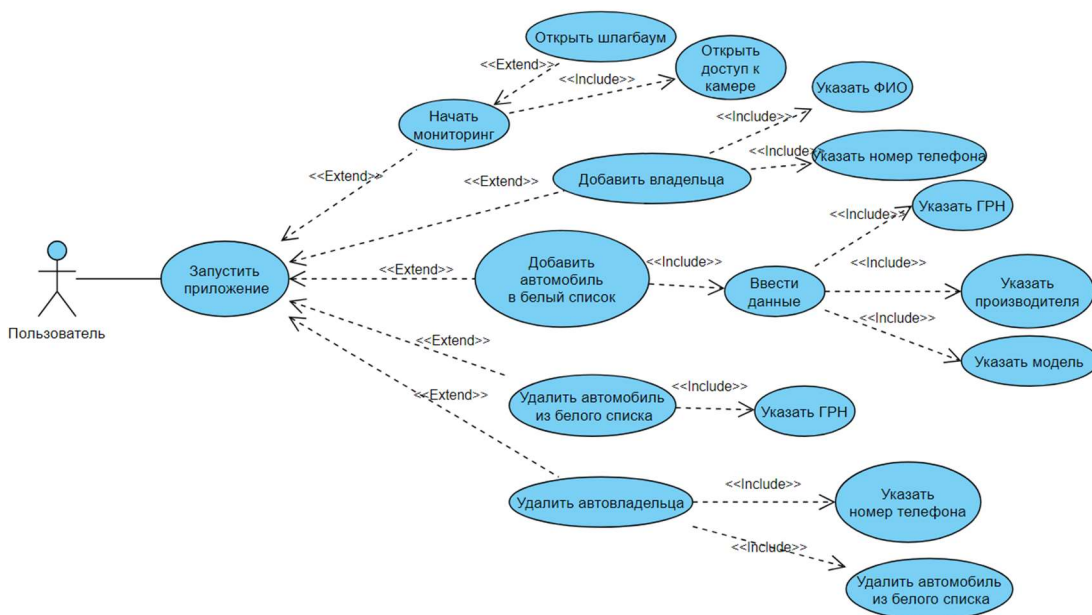


Рисунок 11 – Диаграмма вариантов использования для оператора

Исходя из требований диаграммы вариантов использования была спроектирована диаграмма деятельности, представленная на рисунке 12. Диаграмма отражает последовательность алгоритма для разработки программных средств управления доступом автотранспорта на закрытые территории.

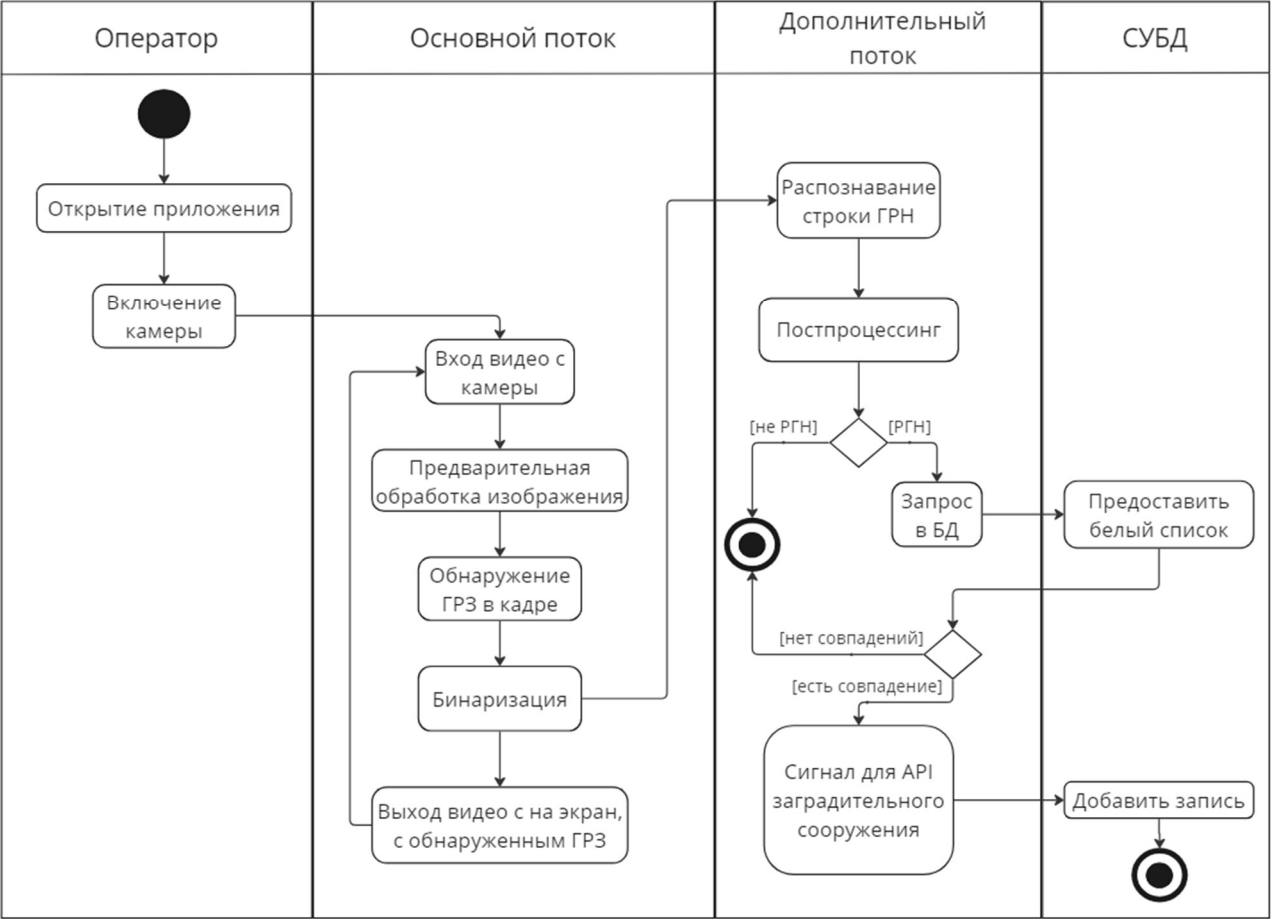


Рисунок 12 – Диаграмма деятельности

При анализе требований было решено выделить дополнительный процессорный поток для требовательного процесса распознавания символов, чтобы трансляция видеопотока с камеры не прерывалась на время действия других операций и, чтобы обеспечить оператору возможность беспрепятственно наблюдать за его зоной ответственности. Для реализации многопоточности на языке Python будет привлечена библиотека «threading». Процессы являются параллельными и не блокируют друг друга.

Таким образом определены основные функциональные требования к программным средствам управления доступом транспортных средств на закрытые территории.

1.9.2 Анализ нефункциональных требований

Нефункциональные требования включают в себя:

- пользовательский интерфейс;
- вычислительную мощность;
- аппаратно-программный комплекс средств.

Программные средства управления доступом транспортных средств должны обладать простым, удобным интерфейсом, направляющим, приветливым и исключающим возможные ошибки со стороны пользователя. Помимо удобства ПО должно позволять использовать все ключевые фрагменты системы и выполнять их корректно.

Приложение должно обеспечивать стабильную работу, высокую скорость обмена данными с сервером, при наличии качественного соединения.

Разрабатываемое приложение будет доступно на операционных системах (ОС) семейства Windows и Unix, благодаря распространенности и адаптивности языка Python. Эти ОС наиболее широко распространены на отечественном рынке, обладают привычным и удобным графическим пользовательским интерфейсом (Graphic User Interface – GUI).

Минимальные системные требования для корректной работы сервера следующие:

- операционная система Windows (7 и выше), Unix, Server (2012 и младше);
- процессор с тактовой частотой не менее 1,8 ГГц;
- оперативная память (ОП): не менее 2 Гб;
- 4,5 Гб свободного места на жестком диске;
- монитор с разрешением не менее 1024x768 пикселей.

Минимальные системные требования для корректной работы клиентского приложения следующие:

- операционная система Windows (7 и выше), Unix;
- процессор с тактовой частотой не менее 1,8 ГГц;
- оперативная память (ОП): не менее 2 Гб;
- 512Мб свободного места на жестком диске;
- монитор с разрешением не менее 1024x768 пикселей;
- цифровая видеокамера с возможностью установки автоматического

вычисления значения экспозиции и диаметра диафрагмы.

1.10 Выбор инструментальных средств

Для хранения и управления данными системы было принято решение об использовании реляционной СУБД – Microsoft SQL Server 2017. Данная СУБД отличается надежностью, стабильностью и производительностью, а также поддерживает декларированный язык программирования Transact SQL для модификации и управления данными в БД [12]. Выбор SQL связан с тем, что он наиболее популярен среди языков программирования для работы с БД и является одним из лучших средств для реализации ПО архитектуры клиент-сервер. SQL служит связующим звеном между взаимодействующей с пользователем клиентской и серверной системами, позволяя каждой из них сосредоточиться на выполнении своих функций [13].

Разработка приложения системы распознавания ГРН будет проводиться с помощью проектирования разнообразных моделей. С этой целью был выбран унифицированный язык моделирования UML, который объединил в себе наилучшие приемы разработки и моделирования ПО. Выбор UML связан с тем, что он позволяет описать систему практически со всех возможных точек зрения и затронуть разные аспекты поведения системы [14].

Для клиентского приложения был выбран язык Python 3. Данный выбор связан с тем, что он поддерживает огромным число расширений и библиотек, представляющих возможность как быстрого создания графического интерфейса

пользователя приложения, так и обеспечения взаимодействия с БД. Благодаря этому Python прекрасно подходит для разработки клиентского приложения [15].

Встроенная библиотека Tkinter позволит организовать графический пользовательский интерфейс. Он включает в себя набор различных элементов управления [16].

В качестве продукта управления проектами был выбран Project Libre – бесплатно распространяемый, мощный и удобный инструмент, использующийся для ведения отчетности, расчетов, планирования и визуализации [17].

1.11 Системные программные средства

На сегодняшний день для решения большинства практических задач используются системы компьютерного зрения (организация данных, системы видеонаблюдения и т. п.). OpenCV (Open Source Computer Vision Library) представляет собой одну из наиболее распространенных библиотек компьютерного зрения с открытыми исходными кодами, в состав которой входит большое число функций обработки видео и изображений в реальном времени [18].

Библиотека распространяется по лицензии Berkeley Software Distribution – BSD, т. е. ее можно бесплатно и свободно использовать как в открытых проектах с открытым кодом, так и в закрытых, коммерческих проектах. Библиотеку не обязательно копировать целиком в свой проект, можно использовать куски кода.

OpenCV реализует, как правило, только базовые операции, используемые в компьютерном зрении, а также поддерживает классификатор каскадов признаков Хаара для обнаружения и локализации объектов. Таким образом, ее можно рассматривать как в целом низкоуровневую библиотеку компьютерного зрения. Для решения серьезных задач необходимо на основе предоставленных библиотекой инструментов создавать свои сложные приложения.

Tesseract – модуль, предназначенный для OCR с открытым исходным кодом. Tesseract предполагает, что получает на вход изображение с указанием в конфигурации областями текста. Распознавание происходит в два этапа. На

первом этапе происходит попытка распознать каждое слово по очереди, передающиеся классификатору, являясь его обучающими данными, благодаря чему адаптивный классификатор получает возможность более точно распознать текст. Второй этап удаляет случайные пробелы, и находит текст, написанный малыми прописными [19].

Tesseract имеет ряд преимуществ, с чем и связан выбор его в качестве модуля OCR в рамках данной работы:

- находится в открытом доступе;
- эффективен при работе с бинаризированными изображениями;
- модуль поддерживает Unicode (UTF-8) и может распознавать более 100 языков (включая русский) [19].

В данной выпускной квалификационной работе для разработки программных средств распознавания ГРН используется библиотека обработки изображений OpenCV с предварительно сформированным классификатором каскадов признаков Хаара [20], библиотека для автоматического распознавания символов Tesseract OCR.

2 Проектирование системы управления доступом автотранспорта

2.1 Базовая архитектура системы

Разрабатываемое приложение имеет двухуровневую архитектуру клиент-сервер. К преимуществам данной архитектуры можно отнести:

- снижение требований к компьютерам, на которых установлен клиент, и нагрузки на информационную сеть, в связи с передачей лишь запросов и ответов на них;
- данные, хранящиеся на сервере, как правило, защищены гораздо лучше большинства клиентов;
- данные в систему вводятся только один раз, вследствие чего исчезает необходимость в дублировании информации и снижается риск ошибок в системе.

Главным недостатком данной архитектуры является зависимость информационной сети и сервера, т. к. неработоспособность сервера может сделать неработоспособной всю сеть, из-за чего будет невозможна работа с БД.

Клиент-серверная архитектура системы изображена на рисунке 13.

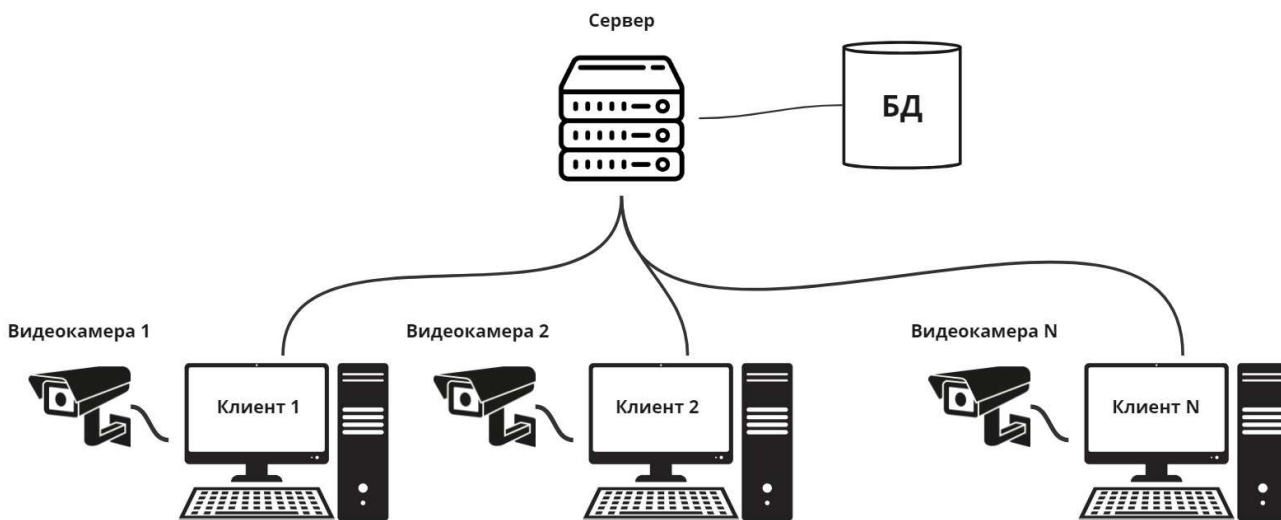


Рисунок 13 – Клиент-серверная архитектура системы

В представленной архитектуре (смотри рисунок 11) показаны четыре основных компонента системы: БД, сервер, клиент и видеокамера.

Первым компонентом является БД. В ней будут храниться постоянные данные о распознанных ГРН, данные о сотрудниках и список АРЗ, регламентирующий проезд ТС.

Второй компонент представляет собой сервер, располагающийся на отдельной станции и хранящий БД.

В роли третьего компонента выступает клиентское приложение, являющееся интерфейсом взаимодействия между пользователем и сервером.

В качестве четвертого компонента выступает видеокамера. ГРЗ ТС, попадающих в область видимости видеокамеры, распознаются при помощи каскадов признаков Хаара. При этом клиентское приложение отправляет информацию о распознанных ГРН на сервер, который, в свою очередь, сохраняет полученные от клиента данные в БД.

По результатам очередного распознавания зафиксированного ГРЗ ТС, каждое клиентское приложение для обновления своих данных отправляет запрос на сервер, посылающий обратно каждому клиенту обновленные данные.

2.2 Логическая модель базы данных

На основе спроектированной концептуальной модели, а также с учетом выбранной модели данных строится логическая модель БД. Она включает в себя описание структуры и состава предметной области, определение сущностей, их атрибутов и отношений между ними.

Ограничение на схему БД, которое вводится для исключения ненужных свойств при выполнении реляционных операций, называется нормальной формой (НФ) [14].

Отношение находится в первой НФ, если все его атрибуты являются простыми (неделимыми), а также отсутствуют повторяющиеся атрибуты [14]. Другими словами, происходит выявление неявных сущностей, представленных в виде атрибутов.

Отношение находится во второй НФ, если оно состоит в первой НФ и каждый неключевой атрибут полностью зависит от первичного ключа [14].

Отношение находится в третьей НФ, если оно состоит во второй НФ и в нем отсутствуют атрибуты, который входят в первичный ключ и находятся в транзитивной зависимости от первичного ключа [14].

Также существует четвертая НФ, пятая НФ и НФ Бойса-Кодда, но для решения практических задач используют третью НФ.

Построена логическая модель БД в нотации IDEF1X, изображенная на рисунке 14.

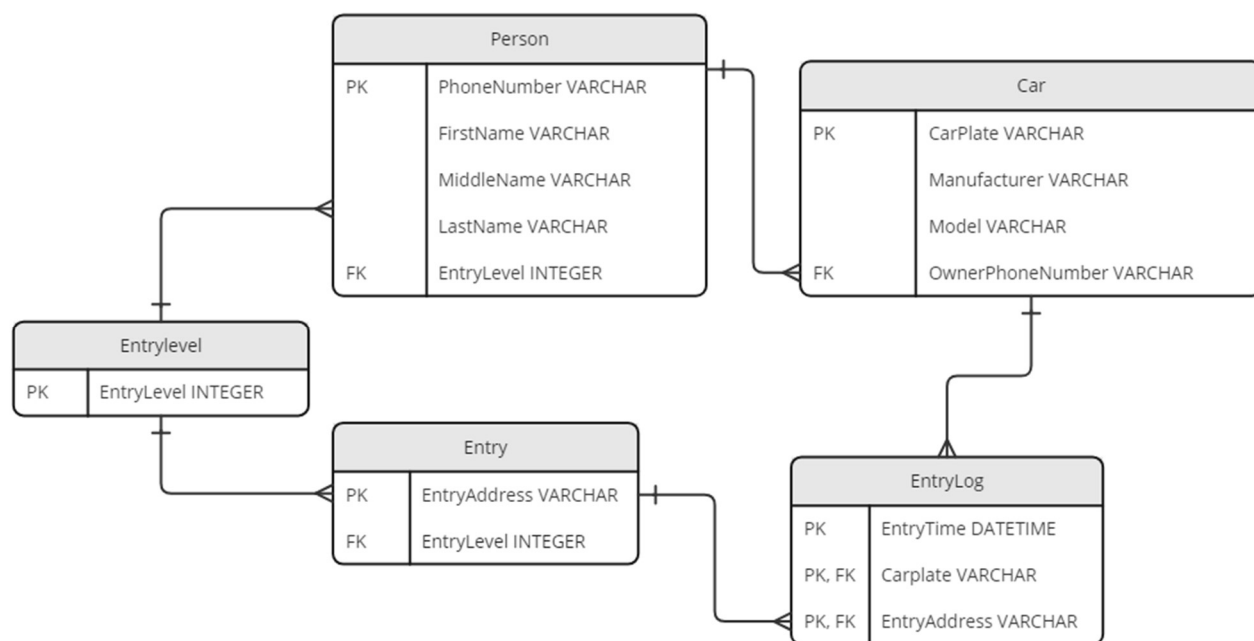


Рисунок 14 – Логическая модель БД

При проектировании логической модели БД выделена ранее не фигурирующая в концептуальной модели сущность «EntryLevel», представляющая собой уровень допуска и связывающаяся с владельцами автомобилей и проездами. В случае дифференциации уровней допуска будет актуально разделять проезды по этому атрибуту.

Т. к. спроектированная логическая модель БД не включает в себя неявные сущности, атрибуты, зависящие только от части уникального ключа, и транзитивных зависимостей, значит она соответствует третьей НФ. В дальнейшем будет производиться переход от логической модели БД к физической.

2.3 Проектирование пользовательского интерфейса приложения

При проектировании пользовательского интерфейса необходимо ориентироваться на возможности привлекаемой библиотеки и множества потребностей пользователя. Исходя из этих аспектов, следует организовать пользовательский интерфейс с возможностью ввода данных для последующего их добавления в базу данных. Результаты проектирования пользовательского интерфейса приведены на рисунках 15 – 19

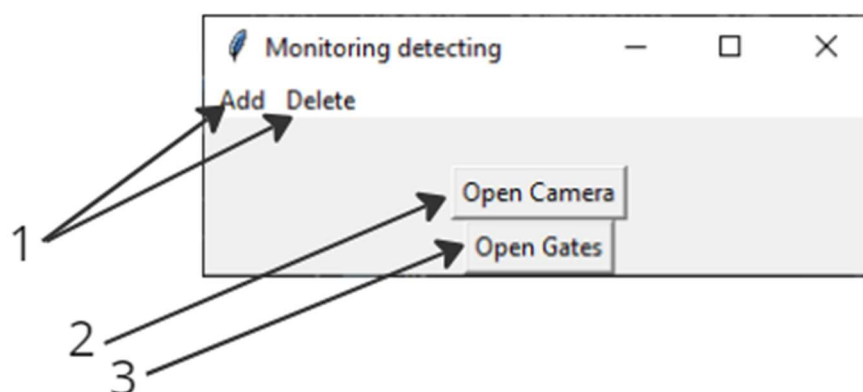


Рисунок 15 – Стартовое окно приложения

Рассмотрим основные элементы интерфейса, изображенные на рисунке 15:

- 1 – кнопка вызова меню;
- 2 – кнопка включения захвата видеопотока камеры;
- 3 – кнопка для открытия заграждения.

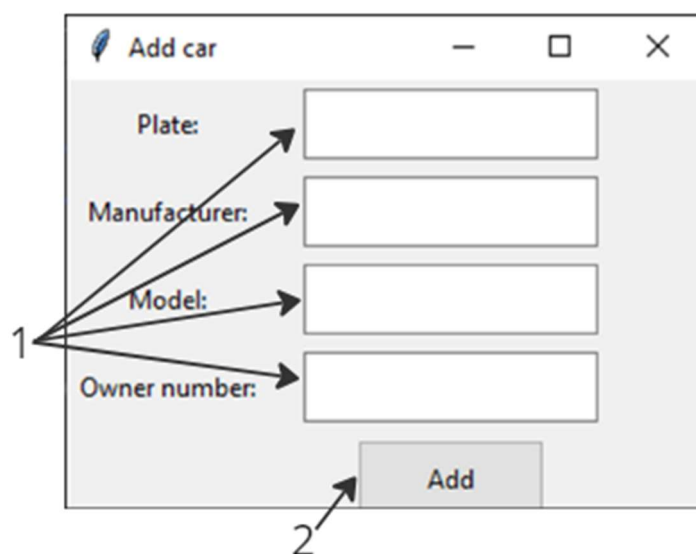


Рисунок 16 – Окно добавления автомобиля

Рассмотрим основные элементы интерфейса, изображенные на рисунке 16:

1 – поле для ввода строковых данных (ГРН, производитель, модель, номер телефона владельца);

2 – кнопка для добавления нового автомобиля в белый список базы данных.

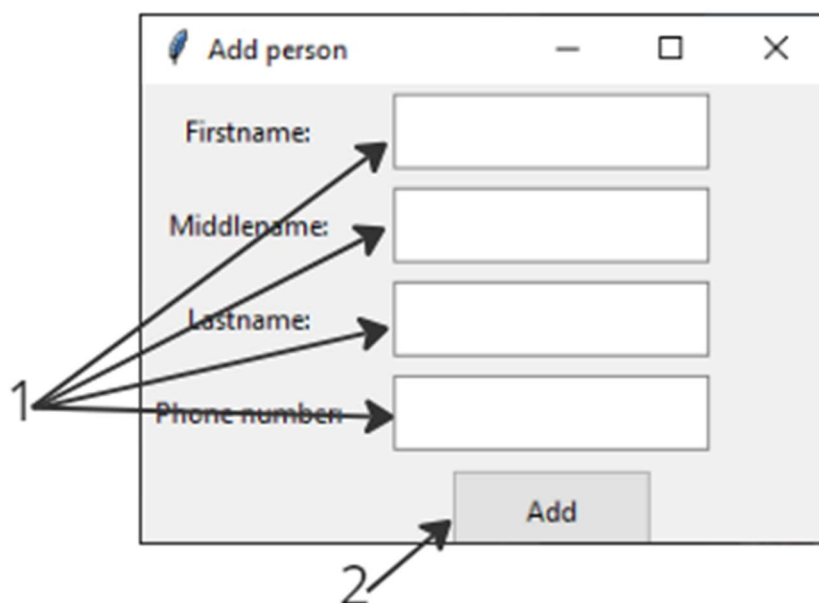


Рисунок 17 – Окно добавления владельца

Рассмотрим основные элементы интерфейса, изображенные на рисунке 17:

1 – поле для ввода строковых данных (ФИО, номер телефона);

2 – кнопка для добавления данных владельца автомобиля.

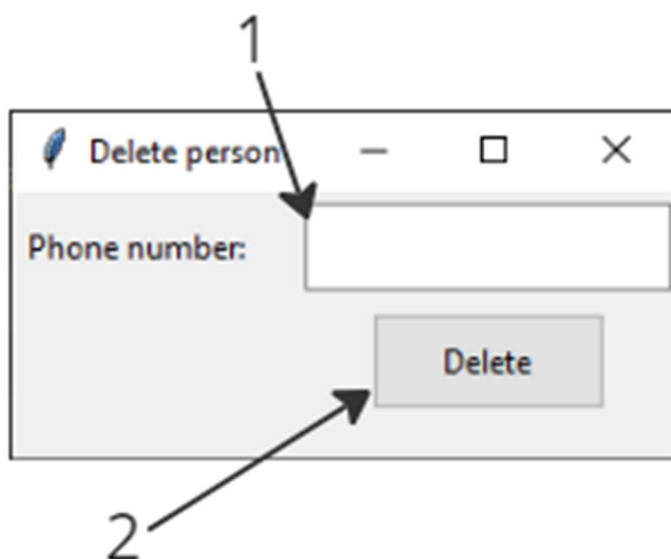


Рисунок 18 – Окно удаления владельца

Рассмотрим основные элементы интерфейса, изображенные на рисунке 18:

- 1 – поле для ввода строковых данных;
- 2 – кнопка удаления.

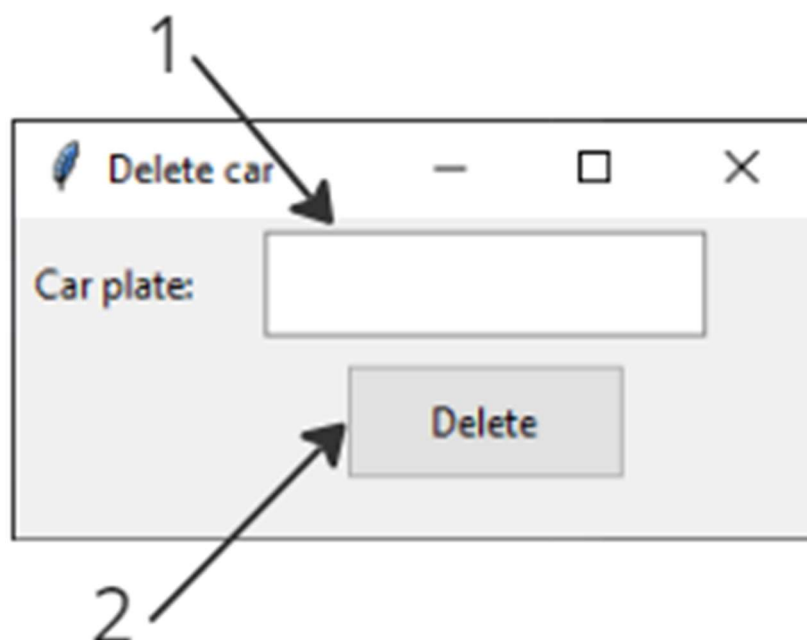


Рисунок 19 – Окно удаления автомобиля

Рассмотрим основные элементы интерфейса, изображенные на рисунке 19:

- 1 – поле для ввода строковых данных;
- 2 – кнопка удаления.

Представленный графический интерфейс пользователя является удобным и интуитивно понятным.

2.4 Проектирование программных средств для управления доступом автотранспорта на закрытые территории

Язык Python в базовом варианте не удовлетворяет все требования по разработке программных средств управления доступом автотранспорта на закрытые территории. Необходимо привлечь сторонние библиотеки, для реализации согласно модели предметной области. Исходя их требований, была спроектирована диаграмма компонентов, представленная на рисунке 20.

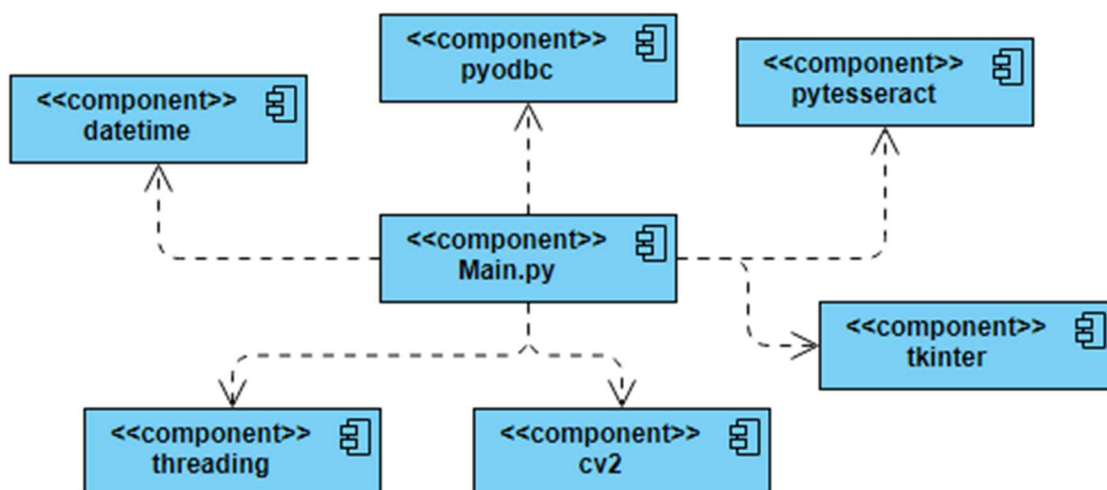


Рисунок 20 – Диаграмма компонентов

Рассмотрим каждый элемент диаграммы:

- datetime – библиотека для работы со временем;
- pyodbc – библиотека для взаимодействия приложения с БД;
- pytesseract – OCR модуль;
- tkinter – библиотека для реализации GUI;
- cv2 – библиотека для работы с изображениями;
- threading – библиотека для реализации многопоточности;
- main.py – входная точка скрипта, которая инициирует вызов

остальных компонентов.

Привлеченные сторонние библиотеки являются свободно распространяемыми мощными инструментами, часто применяемыми при разработке. Их применение позволит наиболее полно реализовать весь спроектированный функционал.

Таким образом определили структуру и модули будущего приложения.

3 Реализация и тестирование приложения для управления доступом автотранспорта на закрытые территории

3.1 Физическая модель базы данных

На основе логической модели БД была построена физическая модель, изображенная на рисунке 21 и содержащая: полную атрибутивную физическую модель БД с учетом возможностей целевой СУБД Microsoft SQL Server 2017 в нотации IDEF1X, построенную в среде Miro с указанием отношений и ограничений.

При переходе от логической модели БД к физической типы сущностей превращаются в таблицы БД, атрибуты – в столбцы физической БД, уникальные идентификаторы – в первичные ключи (Primary Keys – PK), которые не могут содержать NULL значений, а отношения между таблицами представлены с помощью внешних ключей (Foreign Keys – FK).

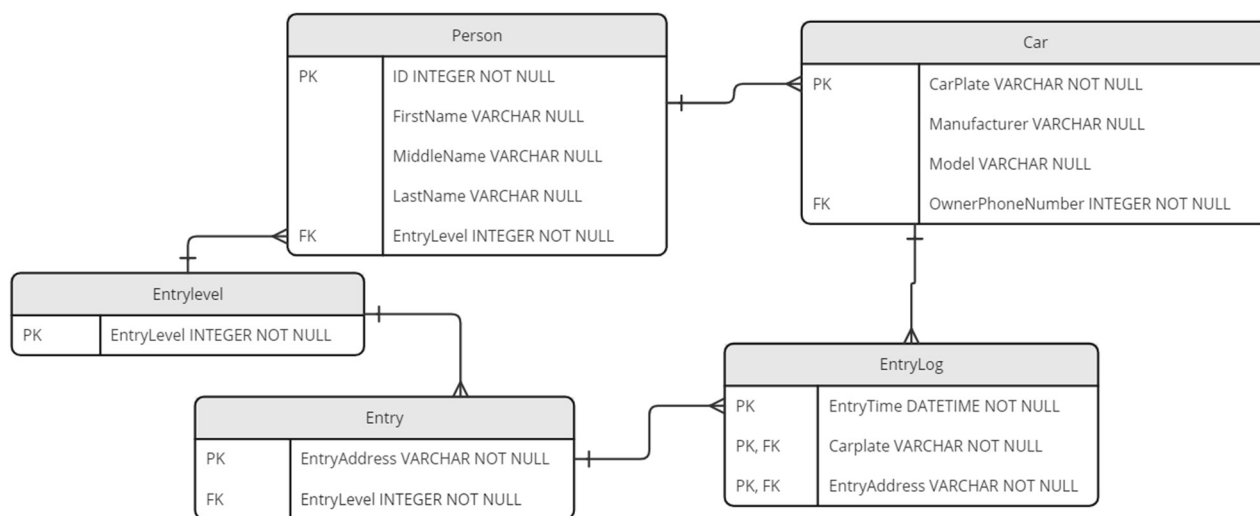


Рисунок 21 – Физическая модель базы данных

Скрипт для создания объектов БД на языке T-SQL, находится в приложении Б. В результате его выполнения в СУБД Microsoft SQL Server 2017 была получена БД, схема которой приведена на рисунке 22.

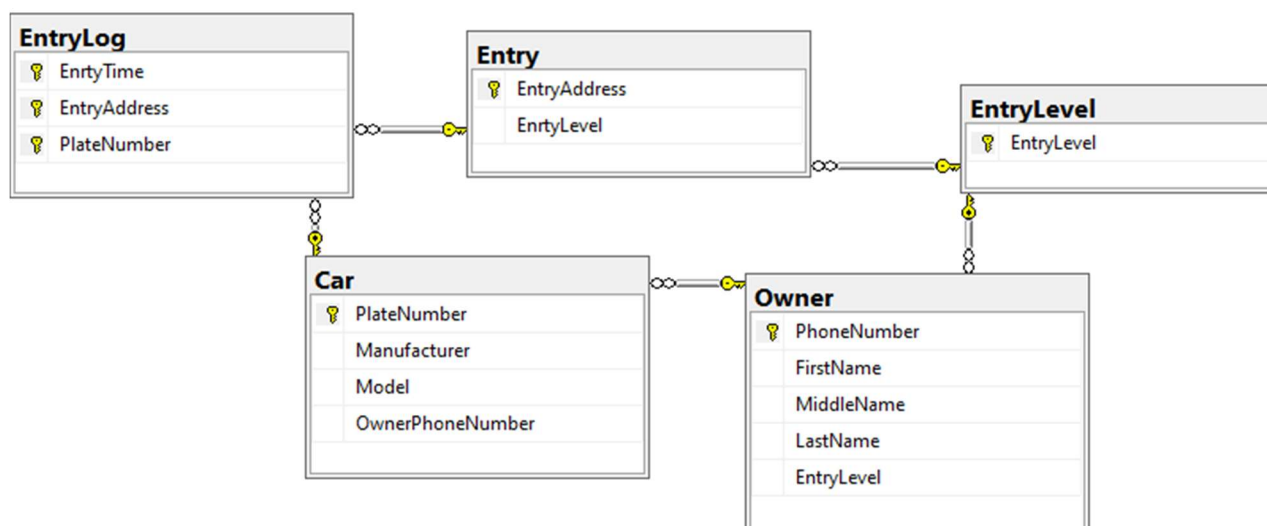


Рисунок 22 – Схема базы данных

Полученная схема БД в СУБД Microsoft SQL Server 2017 (смотри рисунок 22) идентична физической модели БД, а это означает, что БД была реализована корректно.

База данных включает таблицу EntryLevel, отражающая уровень допуска. Таблица может служить для определения соответствия уровня допуска персны и объекта, на который осуществляется попытка проникнуть. которая при потенциальном дальнейшем развитии проекта могут быть использованы для огрнализации разделения уровней допуска, если такая необходимость появится.

3.3 Реализация программных средств

Программные средства программных средств для управления доступом автотранспорта на закрытые территории состоят из функций, которые приведены в таблице 6. В таблице описано назначение функций, применяемых в скрипте main.py, без описания функций и мтодов из сторонних библиотек, и принимаемые функциями аргументы с указанием типа данных, если таковые имеют место.

Таблица 6 – Список реализованных функций

№	Компонент	Описание	Аргументы
1.	open_add_car_window()	Графический интерфейс окна с добавлением автомобиля в белый список	
2.	open_add_person_window()	Графический интерфейс и логика окна с добавлением владельца в БД.	
3.	push_to_db(table, args)	Добавление записи в БД	table – Имя таблицы в БД args – строка, содержащая строчные аргументы для операции INSERT
4.	bicolor_image(image)	Функция бинаризации изображения методом ОЦУ.	image – объект класса Image из библиотеки cv2. Возможно использование изображения только в оттенках серого.
5.	open_camera_window()	Файл, используемый для программного описания свойств и компонентов главной формы.	
6.	tesseract_read(cuted_image)	Распознавание текста с изображения и построение	image – объект класса Image из библиотеки cv2.
7.	num_check(detected_list)	Сравнение распознанного ГРН с содержащимися в БД.	detected_list – список, содержащий строки с ГРН, после распознавания и простроесинга.
8.	open_gates()	Функция, замещающая взаимодействие с API.	

Подробно рассмотрим устройство и последовательность вызова этих функций. Функция `main`, являющаяся точкой входа в приложение вызывает `open_camera_window`, в которой описывается графический интерфейс пользователя главного окна приложения, включающий в себя верхнюю панель, именуемую в библиотеке `tkinter` как `menu`, и кнопки для включения камеры вызова функции взаимодействия с API инженерно-заградительного сооружения. По нажатию кнопки включения камеры вызывается функция `open_camera` захвата видеопотока и преобразования кадра потока в цветовой формат оттенков серого.

Изображение в качестве аргумента поступает в функцию `find_plate`, где при помощи каскадов признаков Хаара [20] происходит классификация и локализация номерного знака автомобиля. Функция возвращает два аргумента – исходное изображение, но с выделенным в рамку регистрационным знаком и образованное бинаризованное, с вычислением порога методом Оцу, изображение, содержащее только информацию внутри упомянутой выше рамки.

Запускается дополнительный поток, организованный с помощью библиотеки `threading`, с целевой функцией `tesseract_read`. Основной поток продолжает непрерывно транслировать видеопоток с выделенным ГРЗ.

Функция `tesseract_read` реализует преобразование изображения в черно-белом формате (бинаризованного) в строковый тип данных. После получения строки происходит сравнение с регулярным выражением, которое определяет сходство строки государственным регистрационным номером. В случае отрицательного результата поток завершает свою работу, иначе начинается процесс ликвидации наиболее часто встречающихся артефактов. Функция возвращает список строк, подлежащих сравнению с записями из белого списка базы данных.

Сформированный список, используя в качестве аргумента в функции `num_check` посредством SQL запроса исследуется на предмет совпадений с записями в базе данных. При обнаружении совпадения вызывается функция `open_gates`, замещающая в данном случае обращение к API заградительного сооружения.

Для полной информации о физическом представлении программных средств управления доступом автотранспорта на закрытые территории была спроектирована диаграмма, проецирующая программную архитектуру на аппаратную [14]. Другими словами, данная диаграмма представляет собой физическое оборудование, на котором выполняется программная система, а также описывает, как ПО разворачивается на этом оборудовании. Диаграмма развертывания изображена на рисунке 23.

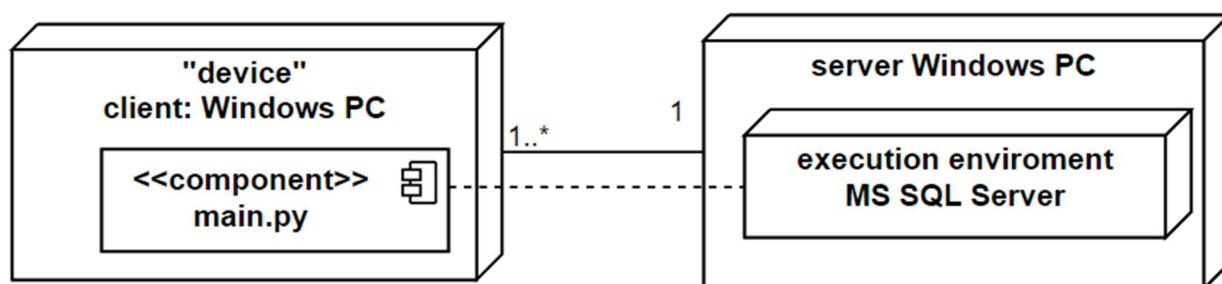


Рисунок 23 – Диаграмма развертывания

На диаграмме развертывания (смотри рисунок 23) показано, что на каждое клиентское приложение вместе с необходимыми ему модулями разворачивается на рабочих станциях пользователей, а БД располагается на отдельном сервере в среде Microsoft SQL Server.

3.4 Тестирование

3.4.1 Выбор режима тестирования

Тестирование ПО производится по двум основным причинам:

- 1) выявление ошибок и недочетов в программном коде, из-за которых могут возникнуть критические ситуации, не дающие ПО работать должным образом или прекращающие его работу;
- 2) проверка поведения и удобства интерфейса ПО с учетом внешних воздействий.

В зависимости от поставленных целей тестирование ПО делится на следующие группы:

- функциональные;
- нефункциональные;
- тесты, связанные с изменениями.

Функциональное тестирование – это вид тестирования программного обеспечения, при котором основные функциональные возможности приложения проверяются на соответствие заранее определенному набору спецификаций. Используя методы тестирования «черного ящика», функциональные тесты измеряют, возвращает ли заданный входной сигнал, желаемый выходной сигнал,

независимо от любых других деталей. Результаты являются двоичными: тесты проходят или не проходят.

Функциональные тесты проверяют, выполняются ли заданные функциональные требования, в то время как нефункциональные тесты могут использоваться для проверки нефункциональных вещей, таких как производительность, безопасность, масштабируемость или качество приложения. Другими словами, функциональное тестирование проверяет, работают ли ключевые функции, а нефункциональные тесты больше интересуются тем, как происходят эти операции [21]. Наиболее распространенными видами функциональных тестов являются:

- функциональное тестирование;
- тестирование взаимодействия;
- тестирование безопасности.

В данной работе предпочтение отдано функциональному тестированию. Обоснованием такого выбора являются проверка основных функций и поведения ПО, выявленных на этапе анализа функциональных требований.

3.4.2 Функциональное тестирование

При тестировании без обработки удалось достичь минимальной точности распознавания на различных наборах в 30%. Причиной низких результатов могут послужить изъяны алгоритма и самих изображений. Зафиксированы ложные срабатывания на нецелевые знаки, другие прямоугольные объекты, ошибки при считывании символов.

При тестировании разработанных программных средств с предварительной обработкой, которая включает в себя частотные преобразования изображения, его фильтрацию и бинаризацию, точность обнаружения ГРЗ увеличилась кратно и стремиться к 100%, помимо этого удалось минимизировать ложные срабатывания.

При привлечении постпроцессинга, исправляющего наиболее частые ошибки OCR, такие как, интерпретация левой грани ГРЗ как единицы или буквы

«Е», проверка позиционирования знака ноль («0») и буквы «О» точность распознавания ГРН повысилась до 70%.

Обработка изображения значительно повлияла на точность распознавания, что свидетельствует о ее необходимости и возможности дальнейшего поиска новых методов обработки. Результаты тестирования представлены в приложении Г. Тестирование без обработки (рисунок Г.1-Г.8), где изображение – результат локализации, подрисовочная запись – результат распознавания символов.

3.5 Метрики кода

Метрики кода являются совокупностью оценок ПО, служащей рекомендательными характеристиками и представляющую более полную информацию о программном коде. Применение метрик позволяет руководителям проектов и предприятий изучить сложность разработанного или разрабатываемого проекта, оценить объем работ, стилистику разрабатываемой программы и усилия, потраченные каждым разработчиком для реализации того или иного решения [22].

Комплексный показатель качества кода (Maintainability Index) представляет собой наиболее часто используемый пример метрики и включает в себя метрики Холстеда. Данная метрика рассчитывается по следующей формуле:

$$MI =$$

$$\max(0, (171 - 5.2 * \ln(HV) - 0.23 * CC - 16.2 * \ln(LoC)) * 100 / 171),$$

где HV (Halstead Volume) – вычислительная сложность;

CC (Cyclomatic Complexity) – структурная сложность кода, т. е. количество различных ветвей в коде;

LoC (Lines of Code) – количество строк кода.

Значения комплексного показателя качества кода может находиться между 0 и 100. Они показывают относительную сложность поддержки кода: чем больше значение данной метрики, тем легче поддерживать код.

Для определения вычислительной сложности программы сначала следует рассчитать словарь и длину программы, а затем вычислить на их основе HV. Словарь программы рассчитывается по формуле:

$$\eta = \eta_1 + \eta_2,$$

где η_1 – число простых операторов и операций в программе;

η_2 – число простых операндов (констант и переменных) в программе.

Длина программы рассчитывается по следующей формуле:

$$N = N_1 + N_2,$$

где N_1 – общее число простых операторов и операций в программе;

N_2 – общее число простых операндов в программе.

Вычислительная сложность рассчитывается по формуле:

$$HV = N * \log_2 \eta .$$

Структурную сложность программы можно рассчитать по следующей формуле:

$$CC = KOP + KOЦ + 1,$$

где KOP – количество операторов разветвления;

KOЦ – количество операторов цикла.

Значение структурной сложности больше 10 свидетельствует о затруднении ее функционального тестирования (например, методов «белого ящика») и требует упрощение программного кода приложения [22].

Среда разработки Microsoft Visual Studio позволяет автоматически вычислять метрики. Произведем вычисления для разрабатываемых программных средств для управления доступом автотранспорта. Метрики Холстеда исходного кода ПО приложения, вычисленные с помощью утилиты «Radon», приведены на рисунке 24.

```
D:\UNIVERSITY\DIPLOM\venv\main.py:
h1: 7
h2: 97
N1: 57
N2: 116
vocabulary: 104
length: 173
calculated_length: 659.8430301465546
volume: 1159.176071238409
difficulty: 4.185567010309279
```

Рисунок 24 – Метрики Холстеда

В итоге расчета метрик получены следующие результаты:

- число простых операторов и операций $\eta_1 = 7$;
- число простых операндов $\eta_2 = 97$;
- словарь программы $\eta = 104$;
- общее число простых операторов и операций $N_1 = 57$;
- общее число простых операндов $N_2 = 116$;
- длина программы $N = 173$;
- вычислительная сложность $HV = 4,2$.

Для вычисления цикломатической сложности (Cyclomatic Complexity) необходимо вызвать команду `radon` с аргументом «сс». Результаты вычисления цикломатических метрик приведены на рисунке 25.

```

PS C:\Users\Maxim\Desktop\CarPlate>radon cc
c:\Users\Maxim\Desktop\CarPlate\venv\main.py
  F 217:0 tesseract_read - C
  F 252:0 num_check - A
  F 119:0 find_plate - A
  F 101:0 push_to_db - A
  F 277:0 get_time - A
  F 19:0 open_add_car_window - A
  F 61:0 open_add_person_window - A
  F 115:0 bicolor_image - A
  F 135:0 open_camera_window - A
  F 274:0 open_gates - A
  F 284:0 main - A

```

Рисунок 25 – Метрики цикломатической сложности

Утилита вычисления метрик высоко оценила цикломатическую сложность программного кода, у каждой описанной функции наивысшая оценка, кроме функции распознавания ГРН, осложненной функционалом постпроцессинга, включающим множество условных операторов.

Комплексный показатель кода вычисляется вызовом «radon mi». Результат расчета приведен на рисунке 26.

```

PS D:\UNIVERSITY\DIPLом> RADON mi D:\UNIVERSITY\DIPLом\venv\main.py
D:\UNIVERSITY\DIPLом\venv\main.py - A

```

Рисунок 26 – Комплексный показатель кода

Разработанный скрипт удостоен высшей оценки утилиты, что свидетельствует о высоком качестве кода. Проведенное тестирование показало отличные результаты выполненной работы.

Заключение

В результате выполнения выпускной квалификационной работы разработаны программные средства управления доступом автотранспорта на закрытые территории, состоящие из клиентского приложения и базы данных. В основе работы данной системы лежат каскады признаков Хаара, сформированные методом Виолы-Джонса для обнаружения и локализации объектов и OCR библиотека Tesseract в связи с их доступностью, быстроедействие и высокой точностью.

Клиентская часть работы реализована на скриптовом языке Python 3. В серверной части ПО с помощью декларированного языка Transact SQL и СУБД Microsoft SQL Server 2017 реализована БД для хранения данных системы.

В процессе выполнения работы проведено функциональное тестирование. На основе анализа результатов тестирования можно сделать вывод о корректности и надежности реализованных программных средств.

На этапе анализа качества кода рассмотрен комплексный показатель качества кода, находящийся на достаточно высоком уровне, что свидетельствует об относительной простоте поддержки исходного кода.

В ходе работы произведено планирование и оценка бюджета разработки проекта с помощью программного продукта Project Libre, в процессе которого были выяснены длительность разработки проекта, равная 44 дням, и бюджет, равный 102411 рублям. Результат оценки бюджета и длительности разработки проекта свидетельствует о его экономической выгодности.

Таким образом, в результате выполнения выпускной квалификационной работы разработаны программные средства управления доступом автотранспорта на закрытые территории, отвечающие всем требованиям технического задания.

По теме данной выпускной квалификационной работы опубликована научная статья и получен диплом победителя международного научно-исследовательского конкурса «Мир студенческой науки».

Список использованных источников

1. Ondrej Martinsky, Algorithmic and mathematical principles of automatic number plate recognition systems // At Brno University of Technology, 2007.
2. ГОСТ 50577-93. Государственный стандарт Российской Федерации. Знаки государственные регистрационные транспортных средств типы и основные размеры. Технические требования. (утв. Постановлением Госстандарта России от 29.06.1993 №165) (ред. от 23.08.2013) – М.: Издательство стандартов. – 33 стр.
3. Приказ Министерства внутренних дел Российской Федерации от 5 октября 2017 г. № 766 «О государственных регистрационных знаках транспортных средств».
4. Приказ Министерства внутренних дел Российской Федерации от 11.03.2020 № 130 «О внесении изменений в приложения № 1 и № 2 к приказу МВД России от 5 октября 2017 г. № 766 «О государственных регистрационных знаках транспортных средств».
5. Приказ Министерства внутренних дел Российской Федерации от 26.07.2022 № 553 «О внесении изменений в приказ МВД России от 5 октября 2017 г. № 766 «О государственных регистрационных знаках транспортных средств».
6. Приказ Министерства внутренних дел Российской Федерации от 31.10.2022 № 810 «О внесении изменений в приложения № 1 и № 2 к приказу МВД России от 5 октября 2017 г. № 766 «О государственных регистрационных знаках транспортных средств».
7. Трапезников И.Н. Разработка и анализ системы распознавания автомобильных регистрационных знаков: специальность 05.12.04 «Радиотехника, в том числе системы и устройства телевидения» : диссертация на соискание ученой степени кандидата технических наук / Трапезников Илья Николаевич ; Ярославский государственный университет им. П.Г. Демидова. - Ярославль, 2014 с. 11-20.

8. Автоматическое распознавание автомобильных номеров / А. П. Кирпичников, С.А. Ляшева, А.В. Обухов, [и др.] // Вестник технологического университета. - 2015. – №4 -С. 218 – 220.
9. Тишин, М. Е. Разработка программных средств для управления доступом автотранспорта на закрытые территории / М. Е. Тишин // Мир студенческой науки. – Пенза : МЦНС «Наука и просвещение», 2024. – С. 11 – 17.
10. Satya Mallick, Histogram of Oriented Gradients explained using OpenCV [Электронный ресурс]. URL: <https://learnopencv.com/histogram-of-oriented-gradients/> (дата обращения 11.04.2024).
11. Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on : P. Viola, M. Jones Rapid Object Detection using a Boosted Cascade of Simple Features ISBN 0-7695-1272-0/0
12. Хомоненко, А.Д. Базы данных. Учебник для вузов / А.Д. Хомоненко, В.М. Цыганков, М.Г. Мальцев. – ДМК. : Пресс, 2013. – 232 с.
13. Система управления базами данных SQL Server [Электронный ресурс]. URL: https://professorweb.ru/my/sql-server/2012/level1/1_1.php/ (дата обращения 15.05.2024).
14. Арлоу, Д. UML 2 и Унифицированный процесс. Арлоу Д., Нейштадт А. – СПб.: Символ-Плюс, 2008. – С.23 – 25.
15. Python – 3 documentation [Электронный ресурс]. URL: <https://docs.python.org/3/index.html> (дата обращения 22.04.2024).
16. Tkinter Python interface to Tcl/Tk [Электронный ресурс]. URL: <https://docs.python.org/3/library/tkinter.html> (дата обращения 23.04.2024).
17. Проектный практикум. Управление проектом в ProjectLibre: учебно-методическое пособие / Сост.: А.Ю. Сапаров, Ижевск, 2020. 48 с
18. OpenCV: OpenCV modules [Электронный ресурс]. URL: <https://docs.opencv.org/4.x/> (дата обращения 23.04.2024).
19. Tesseract User Manual | tessdoc [Электронный ресурс]. URL: <https://tesseract-ocr.github.io/tessdoc/> (дата обращения 24.04.2024).

20. Car-Number-Plate-Recognition-Sysytem [Электронный ресурс]. URL: https://github.com/hasaan21/Car-Number-Plate-Recognition-Sysytem/blob/master/haarcascade_russian_plate_number.xml (дата обращения 27.04.2024).

21. Пироцкая, В. Н. Основы тестирования программного обеспечения : учеб. пособие / В. Н. Пироцкая, Д. А. Градусов ; Владим. гос. ун-т им. А. Г. и Н. Г. Столетовых. – Владимир : Изд-во ВлГУ, 2017. – 100 с. ISBN 978-5-9984-0777-2

22. Тестирование программного обеспечения. Фундаментальные концепции менеджмент бизнес-приложений: Пер. с англ./ Сэм, Джек Фолк, Енг Кек Нгуен. – К.: Издательство «ДиаСофт», 2001. – 544 с.

23. Федеральный закон от 03.08.2018 N 283-ФЗ (ред. от 14.04.2023) «О государственной регистрации транспортных средств в Российской Федерации и о внесении изменений в отдельные законодательные акты Российской Федерации»

Приложение А
(рекомендуемое)
Глоссарий

Государственный регистрационный номер (ГРН) транспортного средства – индивидуальное буквенно-цифровое обозначение, присваиваемое транспортному средству регистрационным подразделением.

Государственный регистрационный знак (ГРЗ) транспортного средства – изделие, изготовленное в соответствии с требованиями национального стандарта и содержащее государственный регистрационный номер транспортного средства.

Программное обеспечение (ПО) – совокупность программ и сопутствующей документации, позволяющая пользователю использовать ЭВМ.

База данных (БД) – организованная в соответствии с определёнными правилами и поддерживаемая в памяти компьютера совокупность данных, характеризующая актуальное состояние некоторой предметной области и используемая для удовлетворения информационных потребностей пользователей.

Система управления базами данных (СУБД) – совокупность языковых и программных средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных одним или несколькими пользователями.

Computer Vision (CV), компьютерное зрение – научная дисциплина, относящаяся к теории и технологии создания искусственных систем, способных получают информацию из изображений, производить обнаружение, отслеживание и классификацию объектов.

Optical Character Recognition (OCR), оптическое распознавание символов – преобразование изображений с текстом в цифровые текстовые данные, которые могут быть восприняты ЭВМ.

Automatic Number-Plate Recognition (ANPR), автоматическое распознавание номерных знаков – технология, использующая оптическое распознавание символов на изображениях, с целью прочесть регистрационный знак автомобиля.

Приложение Б
(рекомендуемое)
Листинг скрипта создания
базы данных на языке SQL

```

CREATE DATABASE EnrtyManagedB COLLATE Crillic_General_CI_A
GO
USE EnrtyManagedB
GO
CREATE TABLE EnrtyLevel
(
    EntryLevel INT NOT NULL PRIMARY KEY
)
GO
CREATE TABLE Owner
(
    PhoneNumber NVARCHAR(12) NOT NULL PRIMARY KEY,
    FirstName NVARCHAR NOT NULL,
    MiddleName NVARCHAR NOT NULL,
    LastName NVARCHAR NOT NULL,
    EnrtyLevel INT NOT NULL,
    FOREIGN KEY (EnrtyLevel) REFERENCES EnrtyLevel(EntryLevel)
)
GO
CREATE TABLE Car
(
    PlateNumber NVARCHAR(9) NOT NULL PRIMARY KEY,
    Manufacturer NVARCHAR(20),
    Model NVARCHAR(20),
    OwnerPhoneNumber NVARCHAR(12) NOT NULL,
    FOREIGN KEY (OwnerPhoneNumber) REFERENCES [Owner](PhoneNumber)
    ON DELETE CASCADE
)
GO
CREATE TABLE [Entry]
(
    EntryAddress NVARCHAR(30) NOT NULL PRIMARY KEY,
    EnrtyLevel INT NOT NULL,
    FOREIGN KEY (EnrtyLevel) REFERENCES EntryLevel(EntryLevel)
)
GO
CREATE TABLE EntryLog
(
    EnrtyTime DATETIME NOT NULL,

```

```
EntryAddress NVARCHAR(30) NOT NULL,  
PlateNumber NVARCHAR(9) NOT NULL,  
    FOREIGN KEY (EntryAddress) REFERENCES [Entry] (EntryAddress),  
    FOREIGN KEY (PlateNumber) REFERENCES Car (PlateNumber),  
    PRIMARY KEY (EnrtyTime,EntryAddress,PlateNumber)  
);
```

Приложение В
(рекомендуемое)

Листинг клиентского приложения на языке Python

```

import pytesseract
import cv2
from tkinter import *
import re
from tkinter import ttk
import winsound
from PIL import Image, ImageTk
import pyodbc
from datetime import datetime
import threading
from tkinter.messagebox import showerror, showinfo

haar_cascade_path =
'C:/Users/Maxim/Desktop/CarPlate/haar_cascades/haarcascade_russian_plate_
number.xml'
pytesseract.pytesseract.tesseract_cmd =
r'F:/Soft/TESSERACT/tesseract.exe'
connection_string = 'Driver={SQL Server Native Client 11.0};
Server=(LocalDB)\MSSQLLocalDB; Database=EntryManageDB ;
Trusted_Connection=yes;'
entry_address = 'Московская 72a'
tesseract_config = '--psm 6 --oem 3 -c
tessedit_char_whitelist=ABCEHKMOPTXY0123456789'

def open_add_car_window():
    add_car_window = Tk()
    add_car_window.title("Add car")
    add_car_window.geometry("300x200")

    label_plate = ttk.Label(add_car_window, text="Plate:")
    label_plate.grid(row=1, column=1, ipadx=6, ipady=6, padx=4, pady=4)

    label_manufacturer = ttk.Label(add_car_window, text="Manufacturer:")
    label_manufacturer.grid(row=2, column=1, ipadx=6, ipady=6, padx=4,
pady=4)

    label_model = ttk.Label(add_car_window, text="Model:")
    label_model.grid(row=3, column=1, ipadx=6, ipady=6, padx=4, pady=4)

```

```

label_owner = ttk.Label(add_car_window, text="Owner number:")
label_owner.grid(row=4, column=1, ipadx=6, ipady=6, padx=4, pady=4)

entry_plate = ttk.Entry(add_car_window)
entry_plate.grid(row=1, column=2, ipadx=6, ipady=6, padx=4, pady=4)

entry_manufacturer = ttk.Entry(add_car_window)
entry_manufacturer.grid(row=2, column=2, ipadx=6, ipady=6, padx=4,
pady=4)

entry_model = ttk.Entry(add_car_window)
entry_model.grid(row=3, column=2, ipadx=6, ipady=6, padx=4, pady=4)

entry_owner = ttk.Entry(add_car_window)
entry_owner.grid(row=4, column=2, ipadx=6, ipady=6, padx=4, pady=4)

btn = ttk.Button(add_car_window, text="Add", command=lambda:
push_to_db('Car', "N\'"

+entry_plate.get()+

"\' ,N\'"

+entry_manufacturer.get()+

"\' ,N\'"

+entry_model.get()+

"\' ,N\'"

+entry_owner.get()+

"\'"))
    btn.grid(row=5, column=2, ipadx=6, ipady=6, padx=4, pady=4)

    add_car_window.mainloop()

def open_add_person_window():

```

```

add_person_window = Tk()
add_person_window.title("Add person")
add_person_window.geometry("300x200")

label_firstname = ttk.Label(add_person_window, text="Firstname:")
label_firstname.grid(row=1, column=1, ipadx=6, ipady=6, padx=4,
pady=4)

label_middlename = ttk.Label(add_person_window, text="Middlename:")
label_middlename.grid(row=2, column=1, ipadx=6, ipady=6, padx=4,
pady=4)

label_lastname = ttk.Label(add_person_window, text="Lastname:")
label_lastname.grid(row=3, column=1, ipadx=6, ipady=6, padx=4, pady=4)

label_phone_number = ttk.Label(add_person_window, text="Phone
number:")
label_phone_number.grid(row=4, column=1, ipadx=6, ipady=6, padx=4,
pady=4)

entry_firstname = ttk.Entry(add_person_window)
entry_firstname.grid(row=1, column=2, ipadx=6, ipady=6, padx=4,
pady=4)

entry_middlename = ttk.Entry(add_person_window)
entry_middlename.grid(row=2, column=2, ipadx=6, ipady=6, padx=4,
pady=4)

entry_lastname = ttk.Entry(add_person_window)
entry_lastname.grid(row=3, column=2, ipadx=6, ipady=6, padx=4, pady=4)

entry_phone_number = ttk.Entry(add_person_window)
entry_phone_number.grid(row=4, column=2, ipadx=6, ipady=6, padx=4,
pady=4)

btn = ttk.Button(add_person_window, text="Add", command=lambda:
push_to_db('[Owner]','N\''

+entry_phone_number.get()+

```

```

"\',N\'"

+entry_firstname.get()+

"\',N\'"

+entry_middlename.get()+

"\',N\'"

+entry_lastname.get()+

"\',1"))
    btn.grid(row=5, column=2, padx=6, pady=6, padx=4, pady=4)

    add_person_window.mainloop()

def push_to_db(table, args):
    try:
        db_connection = pyodbc.connect(connection_string)
        executor = db_connection.cursor()
        executor.execute('INSERT '+table+' VALUES ('+args+')')
        db_connection.commit()
        db_connection.close()
        #showinfo("Success", "Данные занесены!")
        winsound.Beep(3000, 200)
        winsound.Beep(4000, 200)

    except Exception as e:
        showerror("Error :(", "Запрос не выполнен:{}".format(e))

def bicolor_image(image):
    (_, binary_color_image) = cv2.threshold(image, 128, 255,
cv2.THRESH_BINARY | cv2.THRESH_OTSU)
    return binary_color_image

def find_plate (image):
    haar_cascade_classifier = cv2.CascadeClassifier(haar_cascade_path)

```

```

while True:

    detections = haar_cascade_classifier.detectMultiScale(image,
scaleFactor=1.05, minNeighbors=7)

    cuted_image = image

    for (x, y, w, h) in detections:#рисуем рамку на картинке
        cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 255), 2)
        cv2.putText(image, "Number plate detected", (x - 20, y - 10),
cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 255, 255), 2)
        cuted_image = image[y:y + h, x:x + w]
        cuted_image = bicolor_image(cuted_image)

    return cuted_image, image

def open_camera_window():
    video_capture = cv2.VideoCapture(0)

    width, height = 800, 600
    video_capture.set(cv2.CAP_PROP_FRAME_WIDTH, width)
    video_capture.set(cv2.CAP_PROP_FRAME_HEIGHT, height)

    window = Tk()
    window.bind('<Escape>', lambda e: window.quit())
    window.title("Monitoring detecting")

    label_widget = Label(window)
    label_widget.pack()

    def open_camera():
        _, frame = video_capture.read()

        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        plate_image, opencv_image = find_plate(frame)

        captured_image = Image.fromarray(opencv_image)
        photo_image = ImageTk.PhotoImage(image=captured_image)

```

```

label_widget.photo_image = photo_image
label_widget.configure(image=photo_image)

label_widget.after(10, open_camera)

thr = threading.Thread(target=tesseract_read,
args=(plate_image,), name="tesseract_thr")
thr.start()

menu = Menu(window)
menu_item = Menu(menu)
menu_item = Menu(menu, tearoff=0)
menu_item.add_command(label='Car', command=open_add_car_window)
menu_item.add_command(label='Person', command=open_add_person_window)
menu.add_cascade(label='Add', menu=menu_item)
menu_item1 = Menu(menu)
menu_item1 = Menu(menu, tearoff=0)
menu_item1.add_command(label='Show', command=show_log)
menu.add_cascade(label='Log', menu=menu_item1)
menu_item2 = Menu(menu)
menu_item2 = Menu(menu, tearoff=0)
menu_item2.add_command(label='Owner', command=delete_person)
menu_item2.add_command(label='Car', command=delete_car)
menu.add_cascade(label='Delete', menu=menu_item2)

window.config(menu=menu)
button1 = Button(window, text="Open Camera", command=open_camera)
button1.pack()
button2 = Button(window, text="Open Gates", command=open_gates)
button2.pack()
window.mainloop()

def delete_from_db(table, column, val):
    try:
        db_connection = pyodbc.connect(connection_string)
        executor = db_connection.cursor()
        executor.execute('DELETE ' + table + ' WHERE ' + column + '=' + val)
        db_connection.commit()
        db_connection.close()

```

```

        #showinfo("Success", "Данные занесены!")
        winsound.Beep(3000, 200)
        winsound.Beep(4000, 200)

    except Exception as e:
        showerror("Error :", "Запрос не выполнен:{}".format(e))

def delete_person():
    del_person_window = Tk()
    del_person_window.title("Delete person")
    del_person_window.geometry("250x100")

    label_pn= ttk.Label(del_person_window, text="Phone number:")
    label_pn.grid(row=1, column=1, ipadx=6, ipady=6, padx=4, pady=4)

    entry_pn = ttk.Entry(del_person_window)
    entry_pn.grid(row=1, column=2, ipadx=6, ipady=6, padx=4, pady=4)
    btn = ttk.Button(del_person_window, text="Add", command=lambda:
delete_from_db('[Owner]', '[PhoneNumber]', entry_pn.get()))
    btn.grid(row=2, column=2, ipadx=6, ipady=6, padx=4, pady=4)
    del_person_window.mainloop()

def delete_car():
    del_car_window = Tk()
    del_car_window.title("Delete car")
    del_car_window.geometry("250x100")

    label_cp= ttk.Label(del_car_window, text="Car plate:")
    label_cp.grid(row=1, column=1, ipadx=6, ipady=6, padx=4, pady=4)

    entry_cp = ttk.Entry(del_car_window)
    entry_cp.grid(row=1, column=2, ipadx=6, ipady=6, padx=4, pady=4)
    btn = ttk.Button(del_car_window, text="Add",
command=lambda:delete_from_db('[Car]', '[PlateNumber]', entry_cp.get()))
    btn.grid(row=2, column=2, ipadx=6, ipady=6, padx=4, pady=4)
    del_car_window.mainloop()

def show_log():
    window = Tk()

```

```

window.title("База данных SQL Server")

# Фрейм для таблиц
frame_tables = Frame(window)
frame_tables.pack()

db_connection = pyodbc.connect(connection_string)
executor = db_connection.cursor()

executor.execute('SELECT EntryTime FROM EntryLog')
entry_time = executor.fetchall()
executor.execute('SELECT EntryAddress FROM EntryLog')
entry_address = executor.fetchall()
executor.execute('SELECT PlateNumber FROM EntryLog')
car_plate = executor.fetchall()
entry_address = executor.fetchall()
table_view = ttk.Treeview(frame_tables, columns=("EntryTime",
"Address", "CarPlate"), show='headings')
table_view.pack(fill=BOTH, expand=1)
table_view.heading("Address", text="Address")
table_view.heading("EntryTime", text="EntryTime")
table_view.heading("CarPlate", text="CarPlate")
# Обновление таблицы
table_view.delete(*table_view.get_children())
for row in entry_time:
    table_view.insert("", "end", values=row)
for row in entry_address:
    table_view.insert("", "end", values=row)
for row in car_plate:
    table_view.insert("", "end", values=row)

window.mainloop()

def tesseract_read(cuted_image):
    detected_list = []
    res_list = []
    recognized_number =
str.strip(pytesseract.image_to_string(cuted_image, config
tesseract_config))

```

```

detected_list.append(recognized_number)

if len(recognited_number) >= 8:
    res0 = []
    for i in range(len(recognited_number)):
        res0 += recognized_number[i]
        if (recognized_number[i] == '0') and (i == 0 or i == 4 or i ==
5):
            res0[i] = '0'
        if (recognized_number[i] == '0') and (i == 1 or i == 2 or i ==
3):
            res0[i] = '0'

    detected_list.append("".join(res0))

if len(recognited_number) > 8:
    ppf = []
    ppl = []
    for i in range(len(recognited_number)-1):
        ppf += recognized_number[i+1]
        ppl += recognized_number[i]
    detected_list.append("".join(ppf))
    detected_list.append("".join(ppl))

for number in detected_list:
    if re.findall(r'[ABCEHKMOPTXY]\d{3}[ABCEHKMOPTXY]{2}\d{2,3}$',
number):
        res_list.append(number)

if res_list:
    res_list = list(set(res_list))
    num_check(res_list)

def num_check(detected_list):
    print("Распознали:")
    print(detected_list)
    db_connection = pyodbc.connect(connection_string)
    executor = db_connection.cursor()
    executor.execute("SELECT PlateNumber FROM Car")

```

```

print("Б БД:")
for number in detected_list:
    for row in executor.fetchone():
        if str.strip(row) == str.strip(number):
            print(row)
            push_to_db("EntryLog", "CONVERT(DATETIME,\'"
                        +get_time()+
                        "\',21),N\'"
                        +entry_address+
                        "\',\'"+
                        str.strip(number)+
                        "\'")
            db_connection.close()
            break
    break

def open_gates():
    winsound.Beep(500, 300)

def get_time():
    res =[]
    s = str(datetime.now())
    for i in range(len(s)-3):
        res += s[i]
    return "".join(res)

def main():
    open_camera_window()

if __name__ == '__main__':
    main()

```

Приложение Г
(рекомендуемое)
Результаты функционального тестирования



Рисунок Г.1 – Результат «E708MP142»



Рисунок Г.2 – Результат «B802OP97»



Рисунок Г.3 – Результат «X001XX172»



Рисунок Г.4 – Результат «E005AM777»



Рисунок Г.5 – Результат «P030CP190»



Рисунок Г.6 – Результат «K999MA772»



Рисунок Г.7 – Результат «HT5TYC3EP»



Рисунок Г.8 – Результат «P957TO58»

Результаты		Сообщения	
	EntryTime	EntryAddress	PlateNumber
1	2024-06-04 17:48:59.937	Московская 72а	C065MK78
2	2024-06-04 17:49:00.893	Московская 72а	C065MK78
3	2024-06-04 17:49:05.860	Московская 72а	C065MK78
4	2024-06-04 17:49:05.923	Московская 72а	C065MK78
5	2024-06-04 17:49:06.490	Московская 72а	C065MK78
6	2024-06-04 17:49:09.337	Московская 72а	C065MK78
7	2024-06-04 17:49:14.703	Московская 72а	C065MK78
8	2024-06-04 18:11:53.437	Московская 72а	C065MK78
9	2024-06-04 18:11:53.670	Московская 72а	C065MK78
10	2024-06-04 18:11:53.993	Московская 72а	C065MK78

Рисунок Г.9 – Результат добавления в БД функцией push_to_db после считывания камерой ГРН

	PlateNumber	Manufacturer	Model	OwnerPhoneNumber
1	A777AA58	Мерседес-Бенц	C63	+79042665282
2	C065MK78	man	mod	1

Рисунок Г.10 – Результат добавления данных автомобиля в БД из клиентского приложения

Результаты		Сообщения			
	PhoneNumber	FirstName	MiddleName	LastName	EntryLevel
1	+79042665282	Тест	Тестович	Тестов	1
2	1				1
3	88005553535	1	2	3	1

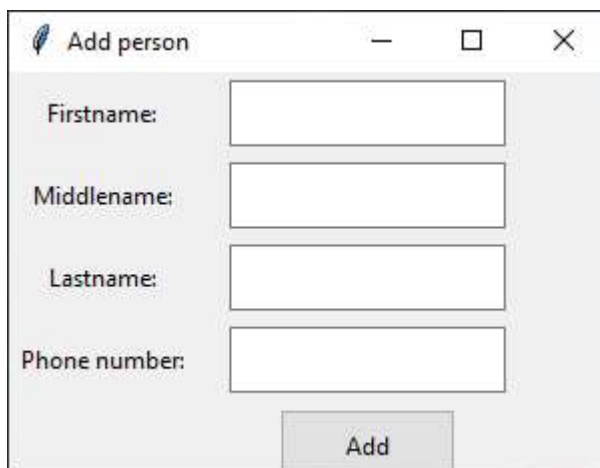
Рисунок Г.11 – Результат добавления данных владельца в БД из клиентского приложения



Рисунок Г.12 – Главное окно приложения после включения камеры

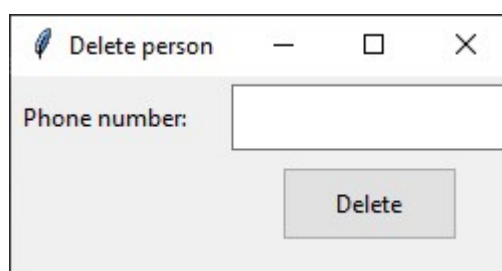
The screenshot shows a window titled "Add car". It contains four input fields with labels: "Plate:", "Manufacturer:", "Model:", and "Owner number:". Below these fields is a button labeled "Add".

Рисунок Г.13 – Окно добавления автомобиля



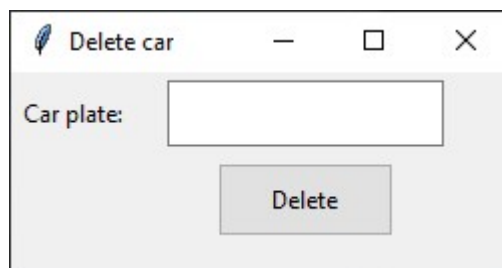
A dialog box titled "Add person" with a feather icon on the left and standard window controls (minimize, maximize, close) on the right. The dialog contains four text input fields stacked vertically, each preceded by a label: "Firstname:", "Middlename:", "Lastname:", and "Phone number:". Below the "Phone number:" field is a button labeled "Add".

Рисунок Г.14 – Окно добавления владельца



A dialog box titled "Delete person" with a feather icon on the left and standard window controls (minimize, maximize, close) on the right. The dialog contains a single text input field preceded by the label "Phone number:". Below the input field is a button labeled "Delete".

Рисунок Г.15 – Окно удаления владельца



A dialog box titled "Delete car" with a feather icon on the left and standard window controls (minimize, maximize, close) on the right. The dialog contains a single text input field preceded by the label "Car plate:". Below the input field is a button labeled "Delete".

Рисунок Г.16 – Окно удаления автомобиля

```
Распознали:  
[ 'C065MK78' ]  
Распознали:  
[ 'E065MK78' ]  
Распознали:  
[ 'EC065MK78', 'C065MK78' ]  
Распознали:  
[ 'C065MK78' ]  
Распознали:  
[ 'E065MK128', 'E065MK12' ]  
Распознали:  
[ 'CC065MK78' ]  
Распознали:  
[ 'T\nC065MK281', 'TT\nC065MK281', 'TT\nC065MK28' ]  
Распознали:  
[ 'C065MK128' ]  
Распознали:  
[ 'C065MK178' ]  
Распознали:  
[ 'EC065MK128' ]  
Распознали:  
[ 'EC065MK78', 'C065MK78' ]  
Распознали:  
[ 'EC065MK78', 'C065MK78' ]  
Распознали:  
[ 'EC065MK78' ]  
Распознали:  
[ 'C065MK28' ]
```

Рисунок Г.17 – Результат распознавания символов с применением обработки

Приложение Д
(рекомендуемое)
Диаграмма Ганта

