МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
вычислительной техники	МОиПЭВМ

Направление подготовки 09.03.02 «Информационные системы и технологии»

Направленность Информационные системы и технологии

БАКАЛАВРСКАЯ РАБОТА

на тему

Разработка приложения для бронирования парковочных мест автостоянки с использованием нейросетевых технологий распознавания автомобильных номеров

Студент _____ Ворнаков Илья Васильевич
(подпись, дата) (ФИО полностью)

Руководитель _____ Попова Н.А_____
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ _____Такташкин Д.В._____
(подпись, дата) (фамилия, инициалы)

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

Заведующий кафедрой _____ ___Козлов А.Ю.____
(подпись) (фамилия, инициалы)

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

Секретарь ГЭК _____ ____Попова Н.А._____
(подпись) (фамилия, инициалы)

Пенза, 2024

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

Факультет	*Кафедра*
Вычислительной техники	МО и ПЭВМ

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

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

БАКАЛАВРА

1. Студент Ворнаков Илья Васильевич гр. 20ВИ1 факультета ВТ
<Фамилия, имя, отчество полностью>

2.Тема работы: Разработка приложения для бронирования парковочных мест автостоянки с использование нейросетевых технологий распознавания автомобильных номеров

Тема утверждена приказом ПГУ №_____от "_____"_____20__ ___

3. Руководитель работы: доцент кафедры «МО и ПЭВМ» Попова Н.А.

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

Назначение: разработка приложения для бронирования парковочных мест автостоянки с использованием нейросетевых технологий распознавания автомобильных номеров.

Основные функции системы:

1. Хранение данных о пользователе, парковке, бронировании;
2. Распознавание автомобильных номеров;
3. Бронирование парковочных мест;
4. Выбор местоположения парковки;
5. Добавление новых пользователей.

Технологии разработки: ООП, UML.
Язык программирования: C#, Python.
Среда программирования: Microsoft Visual Studio 2022

5. Перечень подлежащих разработке вопросов

1.	Анализ предметной области и требований к системе;
2.	Проектирование программных средств;
3.	Реализация программных средств;
4.	Планирование и оценка бюджета проекта;
5.	Тестирование программных средств;
6.	Оформление пояснительной записки

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	15%	14.04.2024 – 21.04.2024	
3	Реализация программных средств	30%	23.04.2024 – 05.05.2024	
4	Планирование и оценка бюджета проекта	15%	08.05.2024 – 12.05.2024	
5	Тестирование программных средств	10%	13.05.2024 – 22.05.2024	
6	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 114 листов, 32 рисунка, 8 таблиц, 5 листингов, 24 использованных источников и 6 приложений.

ПРИЛОЖЕНИЯ ДЛЯ БРОНИВАРОВАНИЯ ПАРКОВОЧНЫХ МЕСТ АВТОСТОЯНКИ С ИСПОЛЬЗОВАНИЕМ НЕЙРОСЕТЕВЫХ ТЕХНОЛОГИЙ, БАЗА ДАННЫХ, ООП, SQL, C#, PYTHON, КЛИЕНТ, СЕРВЕР, UML, MICROSOFT SQL SERVER, MICROSOFT VISUAL STUDIO 2022,

Объект разработки: приложение для бронирования парковочных мест автостоянки с использованием нейросетевых технологий распознавания автомобильных номеров.

Цель работы: разработать приложение для бронирования парковочных мест автостоянки с использованием нейросетевых технологий распознавания автомобильных номеров, состоящее из клиентской, серверной части и нейронной сети.

Технология разработки: C#, Python.

Результаты работы: спроектировано и разработано приложение для бронирования парковочных мест автостоянки с использованием нейросетевых технологий распознавания автомобильных номеров. Все требования к разработке приложения соблюдены.

Содержание

Введение 7
1 Анализ предметной области 8
1.1 Введение в предметную область 8
1.2 Основные проблемы стоянки транспорта в городах 10
1.3 Автоматизации стоянок автомобилей 11
1.4 Распознавание расположения автомобильных номеров на изображении 12
1.5 Распознавание автомобильных номеров на изображении 13
1.6 Анализ программных аналогов 14
2 Постановка задачи и анализ требований 17
2.1 Общие сведения о процессе распознавания автомобильных номеров 17
2.2 Сегментация изображения 18
2.2.1 Математические методы сегментации изображения 18
2.2.2 Нейросетевые методы сегментации изображения 20
2.3 Распознавание текста на изображении 22
2.3.1 Математические методы распознавания текста на изображении 22
2.3.2 Нейросетевые методы распознавания текста на изображении 24
2.4 Обоснование использования нейронных сетей в сегментации изображения и распознаванию текста 26
3 Проектирование приложения 28
3.1 Описание архитектуры нейронной сети YoloV5 28
3.2 Описание нейронной сети Tesseract 30
3.3 Анализ технологий используемых для реализации приложения для бронирования парковочных мест стоянки с использованием нейросетевых технологий распознавания автомобильных номеров 31
3.3.1 Обоснование выбора языка программирования 31
3.3.2 Обоснование выбора программных средств 33
3.3.3 Обоснование выбора инструментальных средств 35
3.4 Описание микро-сервисной архитектуры приложения 36
3.5 Описание структуры проектируемого приложения 38
3.6 Проектирование пользовательского интерфейса 43

3.7 Бюджетирование и планирование проекта 48
4 Реализация приложения 55
4.1 Реализация преобработки изображения 55
4.2 Реализация сегментации изображения 56
4.3 Реализация распознавания текста на изображении 58
4.4 Реализация хранения данных 59
4.5 Реализация пользовательского интерфейса 61
5 Тестирование приложения 65
5.1 Тестирование черным ящиком 65
5.1.1 Общие сведения 65
5.1.3 Метрика качества кода 69
5.1.4 Метрика SLOC 71
Заключение 73
Список использованных источников 74
Приложение А (справочное) Глоссарий 77
Приложение Б (рекомендуемое) Листинг клиентского приложения C# 79
Приложение В (рекомендуемое) Листинг распознавания номера Python 98
Приложение Г (рекомендуемое) Спецификация сущностей 102
Приложение Д (рекомендуемое) Описание методов клиентского приложения 105
Приложение Е (рекомендуемое) Результаты тестирования 109

Введение

В сегодняшних реалиях идентификация регистрационных автомобильных номеров имеет множество недостатков, что позволяет выявить множество способов их решения с целью организации бесперебойной работы идентификации автомобильных номеров.

Распознавание автомобильных номеров (РАН) – ключевой процесс, с помощью которого можно реализовывать множество функций, позволяющих улучшить жизнь людей. К ним можно отнести контроль транспортных потоков, регулирование плотности движения автомобилей, мероприятия по контролю проезда транспортных средств. Эти функции можно реализовать с помощью автостоянки.

Данная выпускная квалификационная работа имеет целью разработку приложения для бронирования парковочных мест автостоянки с использованием нейросетевых технологий распознавания автомобильных номеров.

В одно время с разработкой нейросети для РАН проектируется приложение, которое послужит средством автоматизации пользователя и подобного класса технологий. Программные продукты, предоставляющие подобный программный функционал, кроют в себе дороговизну, но вместе с тем и стабильную, качественную, точную и безотказную работу.

Таким образом, автоматизация процессов бронирования парковочных мест послужит средством упрощения жизни людей, из чего можно сделать вывод о том, что тема данной выпускной квалификационной работы является актуальной в реализации и требует дальнейшего, детального рассмотрения.

1 Анализ предметной области

1.1 Введение в предметную область

Контроль за транспортом в городах становится все более проблематичен, ведь с каждым годом количество автомобилей только растет. В связи с этим, каждый автотранспортное средство имеет свой уникальный набор символов, регламинтрующий его право на передвижение по дорогам общего пользования.

Российская Федерация, имеет в своем обиходе единый стандарт автомобильных номеров. Нормативно правовым документом в России по регламентации номеров является ГОСТ Р 50577-93, введенным в 1994 году. Основные требования по этому стандарту определяют количество символов три буквы и три цифры. При этом по размеру цифры должны превосходить буквы. На номера наносится флаг, сокращенное название страны в английском варианте (RUS) и числовой индекс (код) региона Российской Федерации [1]. Базовый вариант структуры номера представлен на рисунке 1.

Рисунок 1 – Структура автомобильного номера

Ранее применялся формат, имеющий иную структуру, которая вкорне отличалась от ныне используемого. Он был независим и имел буквы кириллического алфавита. Формат имел 4 цифры, 3 буквы и не имел региона, так как за место него выступала первая буква номера.

Из-за того, что все номера имеют единую структуру, можно использовать нейросетевые технологии для их распознавания. Данные технологии предназначены как для автоматизации различных процессов, так и соблюдения определенных правил, позволяющих контролировать и

обеспечивать безопасность движения автотраспортных средств. Ключевым источником для получения результатов в данной технологии является нейронная сеть.

Нейросеть – математическая модель, построенная по принципу нейронный связей человека [2]. Нейронная сеть является способом реализации РАН. Данная модель необходима для определения и распознавания номера и имеет множество ключевых методов, с помощью которых будет достижим конечный результат идентификации номера.

Разработка приложения с внедеренной нейросетевой технологией РАН, как многие считают, является задачей, с легким уровнем сложности. Это не так. Для реализации подобного приложения необходимо имение предельноразвитых навыков разработки, а также умение анализировать возможные критические факторы, ведущие к некорректной работе. К ним можно отнести: визуальный шум, окружающее освещение (естественное и неестественное), плосткостное размещение распознанного номера, искажение цветового фона и прочее.

Для исключения воздействя подобных факторов на работу приложение для бронирования парковочных мест автостоянки с использованием нейросетевых технологий распознавания автомобильных номеров используются различные методы, способные преобразовать и выводить картинку на уровень, необходимый для корректного прочтения и использования: фильтрация и обработка (медианный фильтр, монохром). После избавления от «помех» на изображении, его необходимо сегментировать для дальнейшего процесса распознавания.

Базовая логика приложения представлен на рисунке 2.

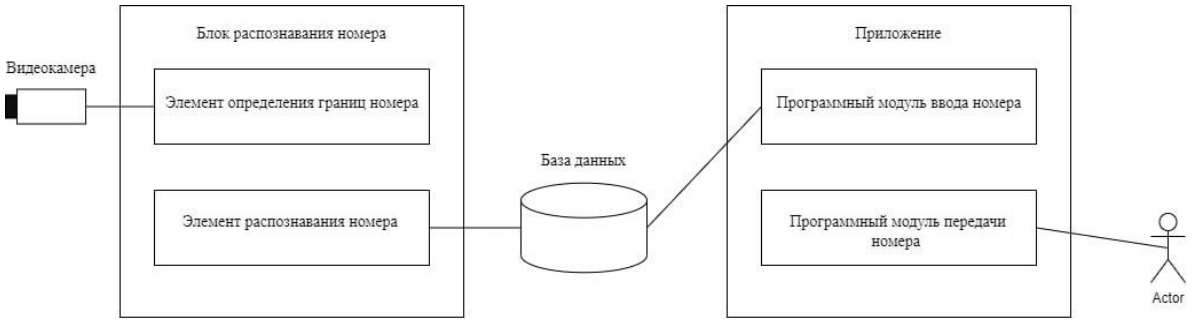


Рисунок 2 – Базовая логика приложения

Все процессы должны реализовываются последовательно, для необходимости выстраивания логики и обеспечения структурированности будущих процессов.

1.2 Основные проблемы стоянки транспорта в городах

Критическим моментом использования автомобилей в современных реалиях является необходимость парковки данного транспортного средства. Во многих городах, с населением свыше уже 300 тысяч человек парковка является головной болью для автовладельца. Данную проблему можно ссылать на множество факторов, администрирование транспортных потоков, перегруженность автомобилями и много другое. Для выявление проблематики данного вопроса необходим тщательный анализ.

В первую очередь проблема стоянки является ограниченность дорожного полотна. На узких участках дороги парковка организована хаотично, что затрудняет как саму парковку, так и транспортный поток, который движется без целей поиска парковочного места.

Во-вторых нехватка места на специализированных парковках. На данных участках наличие парковочного места является наиболее актуальной, ведь концентрация автомобилей превышает предложения.

Автоматизация данного рынка, а именно предоставление услуг по беспрепятственному паркинку неимоверно важна. Многие люди используют парковочные места не по назначениею, а именно место прибытия иногда не совпадает с местом парковки транспорта.

Решений данного вопроса большое множество, и они могут приобретать различные формы, вплоть до пересаживания большей части населения на велосипеды, они не занимают много места. Но с целью организации парковки, приложение для бронирование парковочных мест стоянки, будет считаться наиболее актуальным, ведь бронирование парковочных мест заранее, является эффективным решением по противодействую хаотичной парковки автомобилей.

1.3 Автоматизации стоянок автомобилей

Паркинг автомобилей является главным элементом в жизни почти каждого человека. Без них использование автотранспорта будет практически невозможным, так как без правильной организации такого ключегого момента наступит хаос.

В следствии чего появляется крайняя необходимость в правильной организации данного процесса, который будет удовлетворять всем требованиям.

Автоматизация процесса паркинга задача не из простых. Необходимо правильность действий, ведущих в достижению цели. Данная цель – распознавание автомобильных номеров.

Технологии распознавания автомобильных номеров в данный момент времени набирает неимоверную скорость. Развитие подобных технологий является ключевым в организации парковки.

Идея заключается в том, чтобы человек никоем образом не контактировал с современными аппаратами оплаты и имел возможность бронировать парковочное место заранее, через специальное приложение, не находясь вблизи самой парковки. В том числе имел возможность добавлять номером, просто сфотографировав его.

Удобность данной автоматизации – это понимание и простота. В любую погоду, в любое время, в любом месте, любой человек имеет доступ к неограниченным ресурсам процесса паркинга. Человек, забронировавший место, автоматически числится в базе данных, и при подъезде к самой парковке имеет возможность без предпятсвенно, без лишних физических усилий, не стоя в очереди на заезд, заехать на выбранную им территорию.

В идее автоматизации парковки учавствуют такие объекты как:

– нейронная сеть;
– база данных;
– пользователь (администратор).

Нейронная сеть – это ключевой элемент цепочки, без которого автоматизация будет невозможным, так как через нее происходят основные процессы взаимодействия.

База данных является серединным звеном. Она представляет собой переговорщика, принимая информацию от нейронной сети, храня ее и предоставляя конечном пользователю (администратору).

Самымы главным преимуществом автоматизации процесса стоянки является ее интуитивность, быстрота, эффективность и практичность в использовании в повседневной жизни.

1.4 Распознавание расположения автомобильных номеров на изображении

Существуют определенные этапы, последовательно реализующиеся в процессе распознавания номеров. Эти этапы четко структурированы и взаимосвязаны. К ним относятся: обучение, фильтрация и преобразование, сегментация и прочее. Самым главным этапом является обучение нейронной сети.

Обучение нейронной сети – это процесс, в ходе которого модель искусственного интеллекта определяет процесс, для выполнения определенных задач по исходным данным [3]. Обучение происходит на больших массивах данный, они нужны для получения более качественного и точного результата. В нашем случае обучение проходит на автомобильных номерах, целью чего является выявление области номера авто и распознавание соответствующих символов. В общем случае такой процесс называется детекцией объектов.

Детекция (Object detection) – распознавание объектов. Задача детекции формируется, если требуется определить наличие или отсутствие на изображении объектов определенного класса. Найденный объект в пиксельной оболочке исходного изображения обычно выделяется ограничивающей рамкой (bounding box) [4]. Данный процесс наиболее

распространен в сфере медицины, контроля и актуален в данной предметной области.

Процесс детекции реализуется определенными программными продуктами. Одним из таких продуктов является YOLOv5. YOLO (You Only Look Once) – архитектура нейронных сетей, предназначенная для обнаружения объектов на изображении. Отличительной особенностью YOLO является подход к решению задачи детекции [5].

Подобный процесс крайне необходим в разработке данного приложения и актуальность данного механизма приводит к усовершенствованию автоматизированной работы нужных механизмов.

1.5 Распознавание автомобильных номеров на изображении

Огромное множество различных продуктов используют в своей реализации технологии распознавание текстовых символов. Данные технологии, на сегодняшний день, крайне востребованы и применимы с нейросетевыми технологиями. Одной из таких технологий является OCR. Распознавание текста из изображения (OCR) – это технология для автоматического преобразования текста с различных файлов с целью последующего преобразования текста в машиночитаемую форму [6]. Ее эффективное применение будет являться качественным и работоспособным приложением.

Технология OCR также имеет свою последовательность работы. Данный процесс похож на конвеер, который выполняет все действия поочередно для достижения конечного результата.

Основой данной технологии является движок Tesseract, предназначенный для идентификации символов на основе оптической составляющей, с применение технологии OCR. Данный движок представляет из себя OCR-библиотеку, реализующей поиск символьных совпадений на изображении.

Так же, как и распознавание расположения номера, Tesseract, в качестве сопровождающего элемента использует нейросетевые технологии, которые

реализуют поиск текстовых символов в ряде изображений. Как и многие подобные технологии, этапность данного процесса является ключевой составляющей.

Этапы определения символов на конкретном изображении можно подразделить на:

– начальное форматирование исходного изображения;

– определение разрывов и отступов между символов сеточным принципом;

– сегментация символов (строк, строки, конкретное слово);

– идентификация объектов.

За счет качественного обучения YOLOv5, идентифицируется нужный bounding box, на которым отображен номер. Данный снимок предается обработке, а именно монохромированию, повышению контраста изображения до уровня черно-белый, получая четкое изображение с возможностью однозначного определения символов. Далее происходит определение положения символов на фото, для корректности процесса определения. Этот процесс реализует нейронная сеть Tesseract.

1.6 Анализ программных аналогов

Приложение для бронирования парковочных мест автостоянки с использованием нейросетевых технологий распознавания автомобильных номеров – это автоматизированный программный продукт, хранящий в себе автомобильные номера, данные о пользователе и имеющий внедренный нейросетевой механизм распознавания самих номеров.

В настойщее время распознавание автомобильных номеров является важным аспектом организации и контроля движения автотранспорта. Множество компаний предоставляют услуги, связанные с контролем транспортных потоков. Одной из таких компаний является «Автомаршал». «Автомаршал» решает задачи автоматизации въезда/выезда и учета автотранспорта в транспортном потоке и в пунктах контроля [7]. Данная система имеет свое собственное программное обеспечение, которые

функционирует на базовой операционной системе Windows. Также в составе станции имеется другие объекты инфраструктуры системы, которые оперативно коммуницируют с помощью локальной сети Ethernet.

Программный продукт «Автомаршал» имеет такой принцип работы:

– анализ видео;
– распознавание номера;
– сохранение информации в базу данных;
– сопоставление номера со списками;
– отдача сигналов периферийный устройствам сопряжения.

Также необходимо рассмотреть преимущества и недостатки данной системы. К преимуществам системы можно отнести:

– простота внедрения;
– расширяемость;
– надежность;
– предоставление анализа данных;
– расширенный функционал.

К недостаткам системы «Автомаршал» относится:

– невозможность администрирования пользователя системы;
– недостаточность использования множества иностранных языков.

Поскольку в системе «Автомаршал» используется БД MS SQL Server, использование ПО данной компании может быть затратным. Так как система настолько обширна и засорена ненужными данными, поддержка SQL Server будет затратна еще и отчасти на поддержание и приобритение дополнительных доработок.

Еще одной системой подобного класса является система компьютерного зрения «Angel Vision». Данная система предназначена для формирования фото и видеоаналитики на основе нейросетевых технологий распознавания автомобильных номеров. Имеет в своем составе различные архитектуры внедрения, серверные мощности и поддерживается государственными фондами. Система имеет, как и богатый набор базовых модулей для автоматизации типовых процессов набора дополнительных

модулей для автоматизации отраслевых задач (учёт автотранспорта), так и возможность улучшения функционала для решения задач конкретного учреждения.

Возможности данной системы:

1) определение марок авто;
2) контроль нагрузок транспортных потоков;
3) распознавание вида авто;
4) производство аналитики системы.

К тому же в данной системе недостатки, такие как:

1) невозможность улучшения устройств слежение после покупки;
2) стандартные требования к системе (невозможность замена на аналоги);
3) нет распознавания номеров на больших скоростях.

Анализируя данные системы, можно сделать вывод а том, что все эти системы не достигают необходимого уровня развития в данной сфере, и их применение не является оптимальным вложением денежных средств.

Таким образом, актуальным остается задача разработки системы, главной задачей которой будет автоматизация процесса бронирования парковочных мест автостоянки с использование нейросетевых технологий распознавания автомобильных номеров.

2 Постановка задачи и анализ требований

2.1 Общие сведения о процессе распознавания автомобильных номеров

Выпускная квалификационная работа напрямую посвящена разработке приложения для бронирования парковочных мест автостоянки с использованием нейросетевых технологий распознавания автомобильных номеров.

Насущной задачей разрабатываемого приложения является распознавание автомобильных номеров с последующей автоматизацией процесса бронирования парковочных мест с помощью приложение. Исходя из вышесказанного, следует выделить основные этапы реализации, а именно:

– обучение нейронной сети на достаточном количестве тестовых данных;

– сегментация, определяющая границы номера автомобиля;

– монохромирование выделенной области;

– распознование номера автомобиля;

– передача номера базе данных;

– использование номера в приложении.

Все процессы системы необходимы для реализации автоматизации, исключения хотя бы одного этапа из него приведет к неработоспособности приложения. Для исключения этого фактора из возможных, необходимо обеспечить стабильное функционирование каждого из них.

Обучение нейронной сети – самый важный момент, занимающий коллосальное количество времени для получение стабильного и верного результата. Обученная нейросеть будет предоставлять результат обработки изображения, определяя из общей размерности изображения только нужную область. Данная область должна быть выверена и не иметь больше никаких элементов, которые смогли бы препятствовать качественной обработке. В последующеем, данная область переходит на следующие этапы для преобразования.

Монохромирование (перевод в черно-белый формат) и сегментация объекта, также являются неотъемлемой частью достижение конечного результата. С помощью этих преобразований, изображение получает конечный вид, который будет использован как итоговый.

Нейросеть, имея выделенную область, обрезает остатки исходного изображения, формируя конечный вид требуемой фотографии. Полученное изображение, преобразовавываясь с повышением контраста. Так как номер является, в цветовом диапазоне, двухцветным, черным и белым, белый цвет в этом этапе придает черному цвету четкости, исключение внешних факторов, шумов, размытий, что позволяет нейросети с легкостью, в последующем, считать номер, имя четкие границы видимости содержащихся на нем символов.

Есть множество методов распознавания, но ключевой – это разметка. С помощью оптического распознавания символов (OCR) и размещенных разметок на изображении, распознавание происходит моментально, считывая символы и переводя их в текст, необходимый для продолжения работы процессов.

Далее происходит передача полученного результата базе данных, которая будет хранить в себе номер, с последующем сравнением его в самом приложении при работе. Получается автоматизированное приложение, которое будет в определенной момент хранить в себе результат распознавания и хранящийся в базе данных номер авто.

Само приложение заключается в упрощении процесса осуществления бронирование парковочных мест, в котором пользователь имеет возможность использовать его по назначению, не вдаваясь в глубинную составлющую. Для реализации необходим простой и понятный интерфейс приложения.

2.2 Сегментация изображения

2.2.1 Математические методы сегментации изображения

В данной разработке, методы сегментации изображения имеют большую значимость, ведь именно в сфере распознавания номеров автомобиля,

определение местоположения самого номера будет главной задачей, которая должна приводить к получению объективного результата. Сегментация работает на простых принципах компьютерного зрения, которые помогают в работе по осуществлению сегментации.

Существует огромное количество методов, осуществляющих процесс сегментации изображения. Одним из таких методов является «Алгоритм сегментации по водоразделам»

Данный алгоритм работает с изображением как с функцией от двух переменных:

$$f = I(x, y),$$

где x,y – координаты пикселя: значением функции может быть интенсивность или модуль градиента.

Для наибольшего контраста можно взять градиент от изображения. Если по оси OZ откладывать абсолютное значение градиента, то в местах перепада интенсивности образуются хребты, а в однородных регионах – равнины [8]. Этот алгоритм осуществляет поиск единственный минимум функции, от которого запускается процесс, так называемый «заполнение». Его можно сопоставить с реализацией поднятия кораблей на вершины рек, с помощью поднятия уровня воды в последовательных отсеках, то есть, достигая наступившего минимума, так называемое «заполнение» начинает вводится в процесс, до достижения идущего следом минимума. Процесс реализации достижения минимумов представлен на рисунке 3.

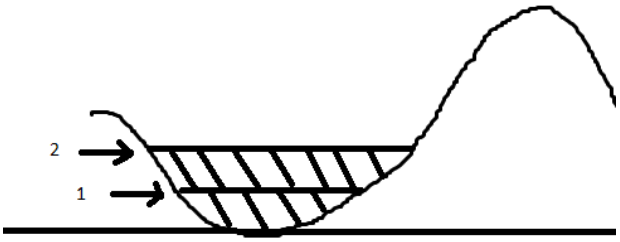


Рисунок 3 – Достижение минимумов

Проблемным фактором данного математического метода, является желательное ограничения подобных минимумов. При увеличении локальных

минимумов, процесс сегментации может быть невыполним и алгоритм будет считаться неэффективным.

В конечном итоге, математический метод, реализующий алгоритм сегментирования получает четко сегментированные объекта, представляющие собой пиксели различных цветов. Сегментированные объекты формируются в одноцветные объекты (сегменты), образуя уровни видимости четко-определенных областей.

Схожим методом сегментации изображения является алгоритм «MeanShift», формирующий поля однотипных объектов, создавая конечную, одноуровневую область.

Алгоритм имеет в своей реализации базу признаков, координатную плоскость, а также составляющие основу, пиксели основных цветов, воспринимающихся глазом человека: красный, зеленый, голубой (RGB).

Основой этого метода являются признаки, которые могут влиять на сегментирование пикселей и формирование конечного результата. При изменении составляющей признака, картинка может быть сегментирована не качественно и слиться с другим признаком, считая их обоих за один объект, набор пикселей. Также схожесть пикселей, аналогично предыдущему методы, играет немаловажную роль, ведь именно цветовая гамма отделяет и делает объекты объектами, формируя их в единую схему.

2.2.2 Нейросетевые методы сегментации изображения

Нейросетевые методы сегментации изображения имеют немаловажное место в определении границ отрабатываемых объектов. Задачей таких методов является поиск родительских меток исследуемоего объекта. Их применение улучшает и автоматизируют различные жизненные процессы, связанные с поиском, определением, фильтрацией и т. д.

Методы нейросетевого сегментирования имеют различия в процессе определения обнаружения конечного результата. Существуют методы как одноэтапные, так и двухэтапные. Оба они реализовывают конечный результат, но процесс отличен друг от друга. Одноэтапный метод не имеет в своей

корневой реализации алгоритма генерации регионов, которые с огромной точностью определяют содержание в себе нужного объекта и его границы. Алгоритм определяется уровень уверенности, с помощью которого формирует рамки с метками, редактируя рамочное положение. Рассмотрим антивариант однотипного метода – двухэтапный метод.

Двухэтапный метод (R-CNN) – алгоритм, опирающийся на сверточных нейронных сетях [9]. Реализация этого алгоритм строитна на отметение базовых поисках однотипных элементов, одинаковых и схожих по размеру.

Его особенность заключается в поисках самих «регионов», рамок, которые имеют различные размеры. Эти рамки, и их поиск, основаны лишь на предсказаниях и доводах о том, что элемент, находящийся в рамке, является итоговым объектом. Процесс поиска представлен на рисунке 4.

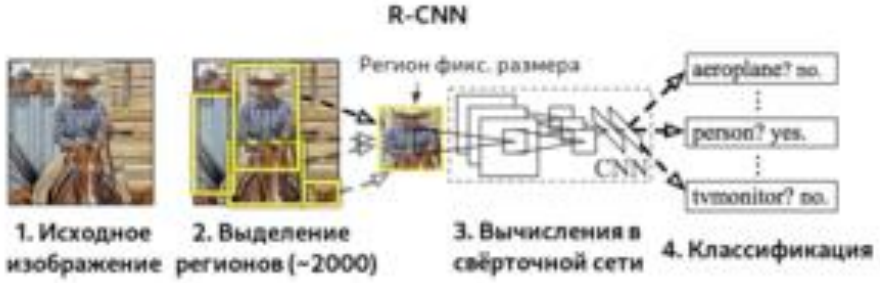


Рисунок 4 – Процесс поиска регионов

Суммарное количество регионов для каждого изображения, сгенерированных на первом шаге, примерно равно двум тысячам. Найденные регионы при помощи аффинных преобразований приобретают размер, который нужно подать на вход CNN [9]. Также существуют различные преобразования, которые реализовывают форматирование размероности рамок.

Одноэтапный метод может представить алгоритм «YOLO» основанный на сегментации объектов в реальном времени. Данный алгоритм формирует сеточное поле, необходимое для определение искомого объекта.

Алгоритм «YOLO» образует сеточное поле их перемножение одинакового количества ячеек самого поля (N*N). Сам процесс опознания нужного объекта очень прост: объект выступает независимой единицей, от

которой происходят различные преобразования, определяющие его точное границы. При прогоне алгоритма, единица объекта может попасть в поле зрения ячейки сеточного поля. В следствии чего, ячейка может реализовывать различные преобразования над искомым объектом, а именно расширение и определение точных границ объекта.

Поддержка степени уверенности – фактор предсказания точности определения, помимо определения его размеромности и прочего. Уверенность в правильности определения объекта происходит на анализе последнего слоя, который является заключительным перед предоставлением информации о результате.

Одноэтапный метод, в сравнении с двухэтапным, является простым, более понятным и простым в реализации. Алгоритмы одноэтапных методов не имеют в своем составе дробление ячеек, которое тормозит процесс сегментации. Дробление, в двухэтапных алгоритмах, реализуется для каждого определенного региона, что ведет в увеличению времени распознавания, а вместе с этим и к замедлению работоспособности механизмов.

2.3 Распознавание текста на изображении

2.3.1 Математические методы распознавания текста на изображении

Распознавание номера автомобиля – одна из важнейших задач разработки приложения для распознавания. Основой реализации данного метода является математическая составляющая. В математической логике заложен весь принцип работы отдельных частей метода.

Метод обратного распространения ошибки основана на стандартном преобразовании и опознании символов текста. Он основан на свёрточной нейронной сети, которая ищет объекты одинаковой размерности и сопоставляет их между собой.

Подобная архитектура включается в себя этапы распознавания символов:

– преобразование изображения;

– определение сетки между символами;

– распознавание символов.

Первый этап строится на детекции и сегментации изображения, именно нахождение символьных районов, преобразовывая их в конечный продукт, убирая шумы, корректируя цветовую гамму. Каждый фрагмент изображения текста поэлементно умножается на матрицу весов (ядро), после чего результат суммируется. Эта сумма является пикселем исходного изображения, называемого картой признаков. После этого взвешенная сумма входов пропускается через функцию активации:

$$\text{OUT}= \frac{1}{1+\exp-\alpha Y},$$

где – параметр наклона сигмоидальной функции активации.

Изменяя этот параметр, можно построить функции с разной крутизной [10].

Второй этап строится на преобразовании сетки, которая формирует отступы, сначала между словами, потом между символами слова. Этот этап крайне необходим для сверки конкретного символа с множеством набора исходного словаря. Сама сверка происходит на третьем этапе.

«Метод ближайшего соседа» относится к методам, определяющих близость объектов по их признакам, в котором признаки являются совокупностью характеристик, описывающих целевой объект.

Идея метода k-NN заключается в том, что если два объекта расположены близ друг друга в пространстве признаков, то зачастую они схожи между собой. При классификации нового объекта метод k-NN находит k ближайших соседей этого объекта и присоединяет его к классу, который является наиболее распространенным среди этих соседей [11].

В зависимости от количества находящихся в близи определенного класса, объект может по-разному интерпретировать свое название, и выбор класса, к которому примкнуть, изменяется. Если объект будет более 1, определение его присоединение к определенному району, будет решаться соседями, находящимися по близости.

Распознавание символов – ключевая функция метода. Производится это для точного определения символа, каким он является, в независимости от того, какой шрифт используется. Распознавание элементов слова, их близости нахождения друг с другом позволяет увеличить процент правильности определения символа, а математическая интерпретация реализует все процессы выполнения.

2.3.2 Нейросетевые методы распознавания текста на изображении

Распознавание текста на изображении – основополагающий фактор, реализующий перевод цифрового изображения текста в печатный вид. Такой процесс реализовывает технология OCR – оптическое распознавание символов.

Проблемы оптического зрения очень велики, и для их решения необходимо соблюдать определенные правила, которые приведут к достижению положительного результата. Для этого необходимо иметь качественное, исходное изображение, такое как: правильное расположение, стандартность формата образов, отсутствие шума на изображении и много другое. Распознавание номеров автомобиля может вполне закрыть большую часть проблем, ведь имея стандартный формат, четкость изображение, однотипный шрифт. Исходя из этого, распознавание номера с изображение будет являться решаемым вопросом.

Решение задачи распознавания текста заключается в использование датасета. Dataset – это обработанная и структурированная информация в табличном виде. Строки такой таблицы называются объектами, а столбцы – признаками. Строки и столбы являются размеченными данными, которые реализуются в процессе машинного обучения [12]. Датасет имеет свою базу фотографий, либо можно загрузить свои, с целью проведения машинного обучения для выявления всех видов текста, находящихся на фото.

Одна из архитектур нейронных сетей, которая используется для распознавания текста – VGG-16. Она основана на сверточной нейросети, которая выделяет из общей массы регионы, в которых расположены

различный надписи, текст. Данная архитектуры основана на 2 принципах работы: поиск регионов и выделение текста. Качество распознавания такого метода остаются на достаточном уровне, удовлетворяющим качеством конечного пользователя.

Реализацией данной архитектуры является модель «CRAFT».

В своей полноценной реализации данная модель реализует 2 оценочных критерия – это определение области символа, с отделением его от общей массы и объединение в конечном итоге этих символов, формируя полное слово.

Ключевой особенностью данного модели является обнаружение центра символа с помощью изотропической гауссовской карты. На первом этапе из изображения выделяются слова (word box). Далее формируются регионы символов выделенной строки. Строка делится на символы, формируя bounding box и на основе них слова формируются с исходное слово, без фонового изображения (cropping) [13]. Процесс распознавания текста на изображении приведен на рисунке 5.

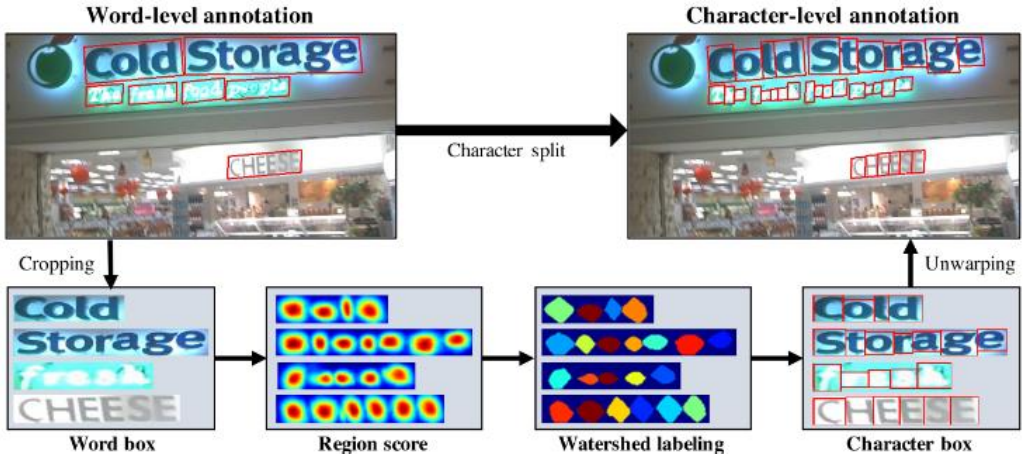


Рисунок 5 – Процесс распознавания текста

Исходя из всего вышеперечисленного, можно сделать вывод что методы распознавания могут иметь как общую, так и отличающуюся архитектуру. Результатом каждого метода является сформированное слово, которое способно распознаться объекты (символы) с дальнейшим переводом их в сформированные картинки, а также распознавание символов с

использованием датасетом, которые упрощают преобразование символов в текст, путем их фильтрации

2.4 Обоснование использования нейронных сетей в сегментации изображения и распознаванию текста

Для разработки приложения существует крайняя необходимость в использовании нейронных сетей. Нейронные сети играют немаловажную роль в автоматизации процесса обработки исходных изображений номера.

Сегментация является основным этапом реализации формирования необходимого результата для достижения целей распознавания номеров. Сегментируя исходное изображение, исключается попадание ненужных элементов исходной фотографии в итоговый результат, получив лишь фото, с параметрами номера авто. Зрение человека не способно в полной мере справится с подобной задачей, необходимо выполнить множество действий, прежде чем номер будет отобран из общего количества подобных номеров.

Распознавание номера, с точки зрения человека, может быть выполнено на все 100%, но время, затрачиваемое на распознавание, запоминание и запись, будет в-корне отличаться. Человека не способен за секунду воспринимать информацию, считывать ее и выводить результат. Процесс распознавания представлен на рисунке 6.

Рисунок 6 – Диаграмма состояний

Изображение поступает на вход. Нейросеть фиксирует рамки номера автомобиля, преобразовывает и сегментирует изображение. На следующем этапе изображение отправляется на распознавание символов, которые считываются и выводятся на экран. Для этих задач нам крайне необходима в использовании нейросеть распознавания номеров и обработки изображения,

которая будет выполнять базовые функции человека за минимальное время, предоставляя точный результат работы.

3 Проектирование приложения

3.1 Описание архитектуры нейронной сети YoloV5

YOLOv5 в настоящее время считается самым производительным алгоритмом поиска и обнаружения объектов в реальном времени. Нейросеть имеет одноступенчатую архитектуру поиска объектов, делая ее производительней двухступенчатых архитектуры.

Архитектура нейросети состоит их нескольких этапов:

– вхождение исходного изображения;

– разбиение изображение на слои, карты признаки;

– преобразование карт признаков;

– предсказывание bounding box`ов.

Этапы обнаружения представлены на рисунке 7.

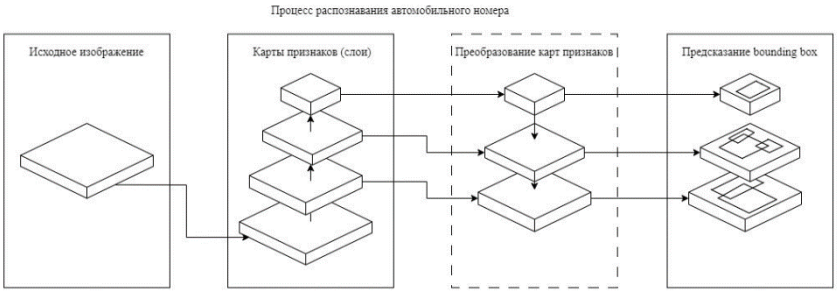


Рисунок 7 – Этапы обнаружения расположения

Данные этапы четко регламентируются, и представляют из себя строение человека, которое исследуют от общего к частному. Вначале у нас есть цельное изображение, которое необходимо обработать и отобрать из него нужный элемент. Далее изображение можно представить в виде позвоночника, содержащего в себе различные отделы, слои, образуя карты признаков для улучшения результатов и полноценного исследования. Далее каждый слой конкретизируется, формируя себя как независимый элемент, на каждом из которых формируются bounding box`ы, предназначенные для поиска и предоставления нужного результата.

Нейронная сеть YOLOv5 обладает функцией дополнения, добавления новых элементы, которая необходима для быстрой обработки множества комбинаций объектов изображения.

К ним относятся методы:

– мозаичное наращивание: техника обработки изображений, при которой четыре обучающих изображения объединяются в одно таким образом, что модели обнаружения объектов лучше справляются с различными масштабами и переводами объектов;

– copy-Paste augmentation: инновационный метод увеличения данных, который копирует случайные участки из изображения и накладывает их на другое случайно выбранное изображение, эффективно генерируя новую обучающую выборку;

– случайные аффинные преобразования: сюда входят случайное вращение, масштабирование, перевод и сдвиг изображений;

– mixUp augmentation: метод, который создает составные изображения путем линейной комбинации двух изображений и связанных с ними меток;

– albumentations: мощная библиотека для дополнения изображений, поддерживающая самые разные техники дополнения;

– hsv augmentation: случайные изменения оттенка, насыщенности и ценности изображений;

– случайное горизонтальное переворачивание: метод увеличения, который случайным образом переворачивает изображения по горизонтали [14].

Нейросеть имеет возможность подстраиваться под тип обучения. Данная функция позволяет увеличить или уменьшить производительность исследуемой модели и карты признаков. Она в силах: увеличивать размерность изображения, использовать определенные методы, способствующие увеличению продолжительности обучения или уменьшению, а также, для повышения производительности самого исследования, использование функций половинной точности, что позволять рационально

использовать память устройства. Все эти возможности влияют положительно на формирование результата.

Нейросеть YOLOv5 является лучшей в сфере прогнозирования и определения объектов, предоставляющая широкий спектр возможностей для улучшения процессов. Имея последовательную архитектуру и различные методы повышения эффективности, она является наиболее подходящей в разработки подобных видов приложений.

3.2 Описание нейронной сети Tesseract

Нейронная сеть Tesseract является высококлассным инструментом, предназначенным для идентификации символьных наборов на различных видах изображений.

Выполнение функций нейронный сети делится на множество этапов, которые следуют последовательно. Первым этапом является определение границ изображения, необходимые для идентификации в выделенной области самих текстовых областей. Вторым этапом следует определение самых текстовых областей. Области формируются так, чтобы были определены не только сами области, но и строки, с последующем разбиением этих строк на отдельные слова, используя шаг сетки.

Нейронная сеть имеет возможность подстраиваться под определенный макет шрифта выделенного текста, слова. В зависимости от макета шрифта, а именно моноширинного или пропорционального, нейросеть может определить расстояние между символами. Понятие «моноширинный» подразумевает, что ширина символов распознанного слова одинакова, отступы между символами стабильны и неизменяемы, а понятие «пропорциональный» имеет вид, что различия в отступах между символами может быть разным и определение рядом стоящих символов может быть затруднено. Решение об определении принимается на заключительном этапе процесса.

Работа нейросети по распознаванию текста на изображении проста. Алгоритм распознавания проходит по всему изображению, определяя области

слов, как результирующую выборку. Проходит по второму кругу, обрабатывая лишь области выделения слов и заключая на каждом этапе их в выборку для обучения на следующих этапах. При ситуации, если на первом этапе не удалось распознать области слов, работа с ними происходит на следующих этапах. Tesseract обладает возможностью принятия решения, которое включает или отказывает в добавлении нераспознанных областей слов.

Принципиальным отличием Tesseract 4 от предыдущих версий является использование LSTM-сетей для определения макета страницы (layout analysis), а также для выделения границ строк и отдельных слов параллельно со стандартным алгоритмом.

Рекуррентные нейронные сети (РНС), подвидом которых является LSTM (long short-term memory), – тип нейронных сетей, задействующих обратную связь. В отличие от нейросетей прямого распространения, в которых информация передается последовательно от слоя к слою, в рекуррентных нейронных сетях имеет возможность запоминать предыдущий этап состояния сети, вместе с текущим состоянием, что позволяет отслеживать последовательность данных [15].

Обучение нейросети может происходить полностью с нуля. Для этого необходимо предоставить большое количество тестовых данных для анализа и получения необходимых информации, которая будет результирующей.

Данная нейронная сеть является эффективно обучающимся и качественно распознающим программным продуктом, который подходит для выполнения данной выпускной квалификационной работы.

3.3 Анализ технологий используемых для реализации приложения для бронирования парковочных мест стоянки с использованием нейросетевых технологий распознавания автомобильных номеров

3.3.1 Обоснование выбора языка программирования

Для реализации приложения необходимо выбрать правильные языки программирования, удовлетворяющие всеми необходимыми требованиями.

Для разработки приложения для бронирования парковочных мест автостоянки с использованием нейросетевых технологий распознавания автомобильных номеров были определены ключевые языки, которые будут использоваться в разработке. Одним из таких языков является Python.

Python – это высокоуровневый язык программирования, имеющий простую и понятную логику работы. Применяется для разработки различных типов приложений, а также применяется для эффективного обучения нейросетей и большого количества данных [16].

Данный язык программирования обладает множеством преимущество относительно других языков. Одним из его преимуществ является простота использования в различных программных реализациях. Также Python обладает обширным спектром встроенных библиотек, необходимых для обработки изображения с номером авто и распознавание символов этого номера.

Еще одним ключевым фактором выбора именно Python является быстрота выполнения распознавания номеров. Код, написанный для этих целей, с легкостью поддается улучшениям, в целях оптимизации процессов, в частности распознавания номеров в реальном времени, сводя его к минимуму, тем самым улучшая быстродействие процесса.

Вторым языком программирования, выбранным для достижения целей выпускной квалификационной работы, был выбран язык программирования C#.

C# – один из наиболее быстро растущих, востребованных и при этом «удобных» языков программирования. Этот язык был создан, как наиболее универсальное средство для разработки программного обеспечения для большого количества устройств и операционных систем [17].

Язык программирования C# также обладает множеством полезных возможностей для реализации приложения подобного рода. Одно из них это – кроссплатформенность. Запуск разработанного приложения можно будет запускать с различных операционные систем, помимо Windows. Также C# обладает возможностью работы с различными библиотеками, необходимые для разработки. К тому же данный язык обеспечивает безопасную разработку,

оповещая и предоставляя ошибки кода, а также изъяны уже написанного кода, которые могут повлиять на безопасность приложения.

Таким образом, можно сделать вывод о том, оба этих языка программирования полностью удовлетворяют всеми необходимыми требованиями, предоставляемые для разработки приложение и выполнения выпускной квалификационной работы.

3.3.2 Обоснование выбора программных средств

Объективный выбор программных средств для реализации полноценного приложения есть первостепенный фактор, определяющий последовательность разработки и выполнение необходимых функций.

К одним из таких программный средств, относится Visual Studio 2022, которая является отличной средой разработки приложения, в частности организации клиент-серверного проектирования.

Для создание графического интерфейса была выбрана программная реализация Windows Forms. Windows Forms – это платформа пользовательского интерфейса для создания классических приложений Windows. Она обеспечивает один из самых эффективных способов создания классических приложений с помощью визуального конструктора в Visual Studio [18].

Для полноценной реализации приложения, в клиентской части необходимо интеграция многих объектов функционала. В частности, для выбора необходимых точек парковки, в область формы приложения был интегрирован компонент, отображающий карту с местоположением парковок – «GMap».

Обучение нейронной сети – главная составляющая распознавания автомобильных номеров. С целью обучить нейросеть и вывести получение результатов от нее на достаточный уровень, было принято решение использовать различные библиотеки машинного обучения. Одной из таких библиотек является «Pytorch», предоставляющая возможности качественного

обучения сверточных нейросетей, а также позволяет улучшать этот процесс путем совершенствования и визуального отображения результатов.

Обучение нейросети на стандартных пользовательских компьютерах является затратным процессом, а именно времени и ресурсов компьютера. К тому же необходимо соблюдение необходимых технических характеристик, для проведения этих процессов. С целью минимизации затрат и ресурсов был использован сервис облачного обучения нейросети – «Google Collab», реализованный на платформе Google Cloud, что позволяет быстро, и без потери мощности обучить нейросеть до нужного уровня.

Управление огромным количеством созданных пакетов разрабатываемого приложения, в процессе реализации, является неотъемлемой частью структурирования данных с целью минимизации потери файлов. Был выбран пакетный менеджер «Anaconda», который является основным элементом структурирования данных и их зависимостей. Данный пакетный менеджер имеет в своем составе также различные интеграционные среды. Одной из таких сред, внедренным в пакетный менеджер, является «Spyder». «Spyder» необходим в обязательной отладке программного кода нашего приложения и его тестирования, с целью нахождения проблемных участков и изъянов кода.

Разработка самого приложения подразумевает под собой определение платформы для его создания. Для создания приложения для бронирования парковочных место автостоянки с использование нейросетевых технологий распознавания автомобильных номеров была выбрана платформа «.NET». Данная платформа позволяет разрабатывать приложения, исходя из предпочтений: мобильное приложение, веб-приложение и многие другие.

Целью разработки была выбрана разработка приложения, основанной на микро-сервисной архитектуре. Фреймворк «ASP.NET Core» позволяет проектировать расширяемые веб-приложения с использованием паттерном, исходя из микро-сервисной составляющей приложения. К плюсам использования данного фреймворка относится асинхронность процесса программирования приложение, а также улучшение обрабатывания

поступающих запросов. Также еще одним плюсом является различные встроенные компоненты. К одним из таких компонентов примыкает «EF Core», фреймворк, необходимый для полноценной работы с базой данных, которая также реализована в приложении.

Все вышеперечисленные программные средства полностью удовлетворяют всем требования разработки приложение и качественно влияют на получения объективных результатов.

3.3.3 Обоснование выбора инструментальных средств

Для организации стабильной работы приложения, его составных частей, необходимо учитывать правильный подбор необходимого оборудования.

Мало того, что приложение для бронирования парковочных мест автостоянки с использование нейросетевых технологий распознавания автомобильных номеров должно быть реализовано с легкодоступным интерфейсом для потенциального пользователя, так и иметь возможность реализовывать весь функционал приложения в полном объеме.

Немаловажной частью будет являться и корректное функционирование клиент-серверной части, а именно их общение, которое не должно заторможенном. Отклик сервера на запрос пользователя должен быть моментальным.

Основной средой функционирования была выбрана операционная система Windows. Она имеет широкий спектр функционала, не требуемого большого количества затрачиваемых ресурсов, продуманный, простой интерфейс. В частности, для этих целей подходит Windows версии 7 и выше. Лицензионный пакет данного программного продукта на рынке очень востребован и ценовой диапазон покупки очень мал, что является основной причиной выбора именно этого ПО в соотношении цена/качество.

Для полноценной и стабильно работы приложения должны быть сформированы системное требования, а именно:

- операционная система: Windows 7+;
- процессор: Intel Core i5 2+ ГГц, 4 ядра;

– оперативная память 8 Гб;

– память жесткого диска 1 ТБ;

– разрешение монитора не менее 1600*900 пикселей.

Именно этот выбор операционной системы и системных требований обеспечивают качественную работу и стабильность выполнения функциональных возможностей приложения.

3.4 Описание микро-сервисной архитектуры приложения

Первостепенной задачей реализации приложения стоит правильная организация микро-сервисной архитектуры, которая будет отвечать всем необходимым требования, будет сформирована для достижения целей разработки.

Для обеспечения корректного функционирования необходимо понимать все составляющие данной архитектуры, способы ее построение, а также плюсы и минусы разработки.

Микро-сервисная архитектура представляется из себя взаимодействующие между собой отдельные сервисы, выполняющие свои, четкие функции. Главным принципом архитектуры – выделение взаимосвязанный частей приложения в отдельные, упорядоченные, согласованные и взаимосвязанные модули.

Микро-сервисная архитектура произошла из архитектуры, называемой «монолитной». Монолитная архитектура имеет традиционный подход к построению приложений, где весь функционал концентрируется в одном приложении или различных, взаимосвязанных модулях. Это единое приложение, где компоненты тесно интегрированы между собой и взаимодействуют напрямую [19]. Использование монолитной архитектуры – непрактичное решение, связанной со многими факторами, а именно:

– трудность внесение новых функциональных решений, в связи с увеличение нагрузочных составляющих, влияющих на ресурсо-оборот системы;

– невозможность внесение различных модификация приложения, из-за единой среды разработки, без разделений обязанностей взаимодействующих частей.

Из-за работы данных факторов, микро-сервисная архитектура решила часть задач, нерешенных в монолитной архитектуре. Особенностью архитектуры является децентрализованность процессов разрабатываемого приложения. В свою очередь децентрализация способствует раскрытию конкретики разработки, улучшению безопасности и тщательности проработки ключевых моментов что является плюсом над монолитной архитектурой.

Схема работы микро-сервисной архитектуры представлена на рисунке 8.

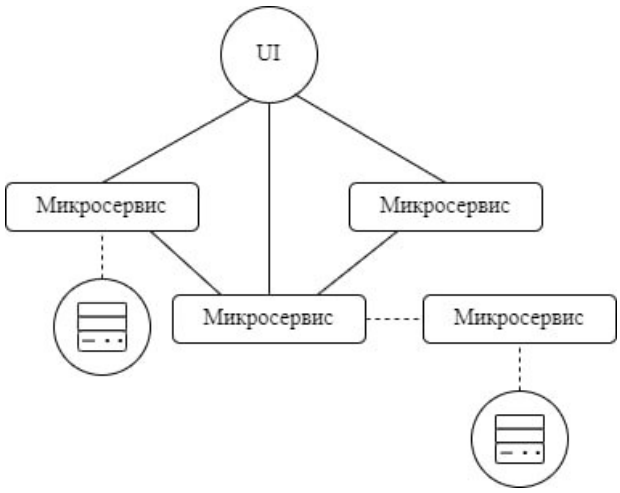


Рисунок 8 – Микро-сервисная архитектура

Каждый микро-сервис имеет собственный набор кода, базу данных и API для взаимодействия с другими сервисами. Они могут быть написаны на разных языках программирования и использовать различные технологии. Взаимодействие сервисов может осуществляться посредством сетевых запросов или сообщений напрямую [20]. Процесс работы микро-сервисной работы заключается в организации общения клиентской части с реализованными микро-сервисами, таки и общение по запросам микро-сервисов между собой для получения информации по запросу у базы данных, и предъявления информации клиенту в обратной схеме. Для разрабатываемого приложения была спроектирована диаграмма последовательности. Диаграмма последовательности представлена на рисунке 9.

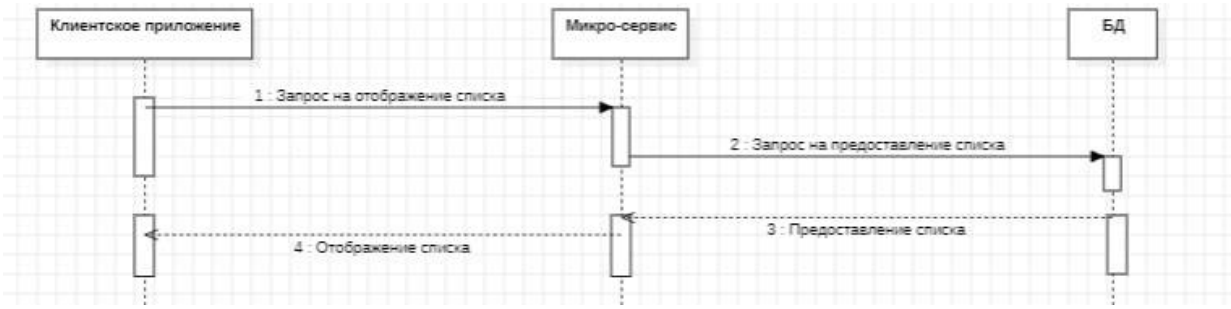


Рисунок 9 – Диаграмма последовательности

Микро-сервисная архитектура имеет ряд важных минусов, которые влияют на процесс разработки. К ним относится:

– количество сервисов (с ростом количества микро-сервисов, возникает проблема контроля за каждым из них);

– нестабильность целостности данных (уникальность данных каждого микро-сервиса увеличивает и создает проблематику сохранения данных и синхронизации самих микро-сервисов);

– тестирование (тестирование огромного количества микро-сервисов является ресурсо-затратным процессом).

Все эти минусы нивелируются с плюсами работы с микро-сервисной архитектурой. Масштабируемость, легкость проектирования, самодостаточность каждого микро-сервиса являются ключевыми, для выбора именно этой архитектуры. Масштабируемость – самый главный фактор, способствующий увеличению функциональных возможностей, а также повсеместной поддержки приложения.

3.5 Описание структуры проектируемого приложения

Целью разработки приложения для парковочных мест автостоянки с использование нейросетевых технологий распознавания автомобильных номеров является необходимость правильно организвать структуру самого приложения, для правильного функционирования составных частей. Структура приложения основана на микро-сервисной архитектуре, что

способствует взаимодействию между сервисами и логическому управлению, а также интерфейсом пользователя. Структура проектируемого приложения представлена на рисунке 10.

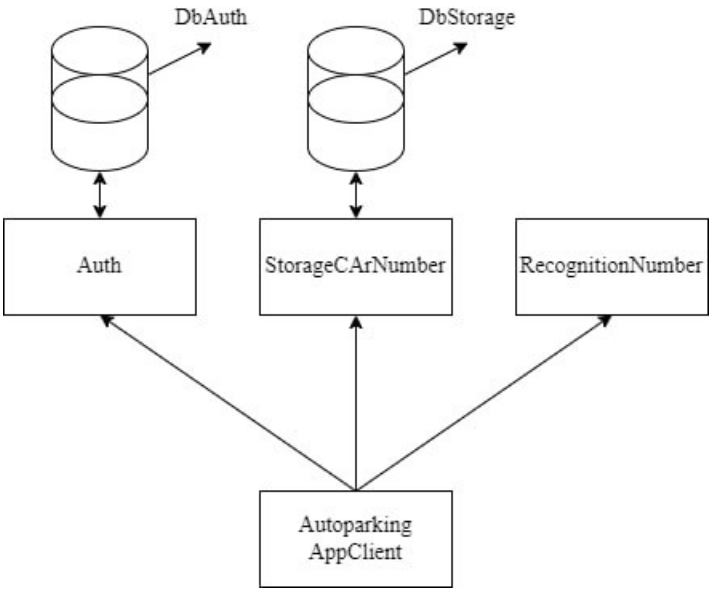


Рисунок 10 – Структура приложения

Структура приложения состоит из:

– базы данных (DbAuth DbStorage);
– микро-сервисы (Auth, StorageCArNumber, RecognitionNumber);
– клиентское приложение (Autoparking AppClient).

Основным проводником работы приложения является клиентское приложение. От него к микро-сервисам поступают запросы, которые контактируют с соответствующими базама данных, для предоставления информации как микро-сервисам, так и клиентскому приложени. Для реализации общения базы данных и клиента, существует компоненты посредники, что есть децентрализованность системы, которая способна разделяться и масштабироваться в независимые компоненты. Общение всех компонентов системы происходит по принципу запросов.

Пример реализации запросов представлены на рисунке 11.

```
public static class ServicesConstants
{
    public const string AuthService = @"http://localhost:5000/api/Auth/";

    public const string StorageCarNumbers = @"http://localhost:5001/api/Storage/";

    public const string PyNumberRecognition = @"http://localhost:5002/car-number";
}
```

Рисунок 11 – Реализации запросов

Данные запросы представлены в виде http с использованием entpoint`ов для предоставления URL-адресов. Endpoint – это реализатор, осуществляющий общение, с целью получения информации, данных.

Рассмотрим реализацию контроллера авторизации пользователя приложения представлен в листинге 1.

Листинг 1 – Код реализации контроллера авторизации пользователя

```
namespace AutonomousParkingApp.Authentication.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class AuthController : ControllerBase
    {
        private readonly IAuthService _authService;

        public AuthController(IAuthService authService) =>
            _authService = authService;

        [HttpPost("add-user")]
        public async Task<ActionResult> AddUserAsync([FromBody] RegDto
user)
        {
            try
            {
                await _authService.AddUserAsync(user);

                return NoContent();
            }
            catch (AuthException ex)
            {
                return Problem(ex.Message, statusCode: ex.StatusCode);
            }
        }

        [HttpPut("update-user")]
        public async Task<ActionResult> UpdateUserAsync([FromBody]
UserDto user)
        {
            try
            {
                await _authService.UpdateUserAsync(user);

                return NoContent();
            }
            catch (AuthException ex)
            {
                return Problem(ex.Message, statusCode: ex.StatusCode);
            }
        }

        [HttpGet("get-user/{userId}")]
        public async Task<ActionResult<UserDto>>
GetUserByUserIdAsync([FromRoute] Guid userId)
        {
            try
            {
```

```
                var result = await
_authService.GetUserByUserIdAsync(userId);

                return Ok(result);
            }
            catch (AuthException ex)
            {
                return Problem(ex.Message, statusCode: ex.StatusCode);
            }
        }

        [HttpPost("get-user-by-login-password")]
        public async Task<ActionResult<UserDto>>
GetUserAsync([FromBody] AuthDto authDto)
        {
            try
            {
                var result = await _authService.GetUserAsync(authDto);

                return Ok(result);
            }
            catch (AuthException ex)
            {
                return Problem(ex.Message, statusCode: ex.StatusCode);
            }
        }

        [HttpGet("get-userId-by-login/{login}")]
        public async Task<ActionResult<Guid>>
GetUserIdByLoginAsync([FromRoute] string login)
        {
            try
            {
                var result = await
_authService.GetUserIdByLoginAsync(login);
                return Ok(result);
}
            catch (AuthException ex)
            {
                return Problem(ex.Message, statusCode: ex.StatusCode);
            }
        }
    }
}
```

В теле запроса использует FromBody – получает значения из тела запроса. FromRoute– получает значения из данных маршрута [21].

Структура метода AddUserAsync заключается в добавлении нового пользователя, с помощью объекта RegDto. Обновление осуществляется с помощью метода UserDto и так далее. Предоставление информации о пользователе происходит через метод GetUserIdAsync и GetUserIdLoginAsync получает логин пользователя и возвращает его ID. Во всех методах

используется обработка исключений AuthException, которая возвращает сообщения об ошибках, при неправильной работе одного из методов.

Для хранения информации должна быть реализована база данных, содержащая в себе таблицы с необходимыми атрибутами. Содержимое атрибутов таблицы «User» представлено в таблице 1.

Таблица 1 – Таблица «User»

Идентификатор	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Описание
ID	ID пользователя	Целый	50	1-1000	Нет	PK	376
Login	Логин пользователя	Символьный	50	«A-z», «0-9», «А-я»	Нет	-	gog
Name	Имя пользователя	Символьный	50	«A-z», «0-9», «А-я»	Нет	-	goga
Password	Пароль пользователя	Символьный	50	«A-z», «0-9», «А-я»	Нет	-	487393bdjd
CarNumber	Номер авто	Символьный	9	«А», «В», «Е», «К», «М», «О», «Р», «С», «Т», «Y», «Х», «0-9»	Нет	-	M716BO13
Phone	Телефон пользователя	Целый	11	«0-9»	Да	-	+88005553535
CardNumber	Номер карты	Целый	16	«0-9»	Да	-	1212121212121212

По аналогии реализуются и другие таблицы базы данных. Содержимое других таблиц и их атрибутов представлены в приложении Г.

3.6 Проектирование пользовательского интерфейса

С целью предленого понимания функциональных возможностей системы, необходимо проанализировать все составляющие и процессы разрабатываемого приложения. С этой целью была реализована диаграмма вариантов использования, которая в полной мере отображает весь потенциал приложения. Диаграмма вариантов использования представлена на рисунке 12.

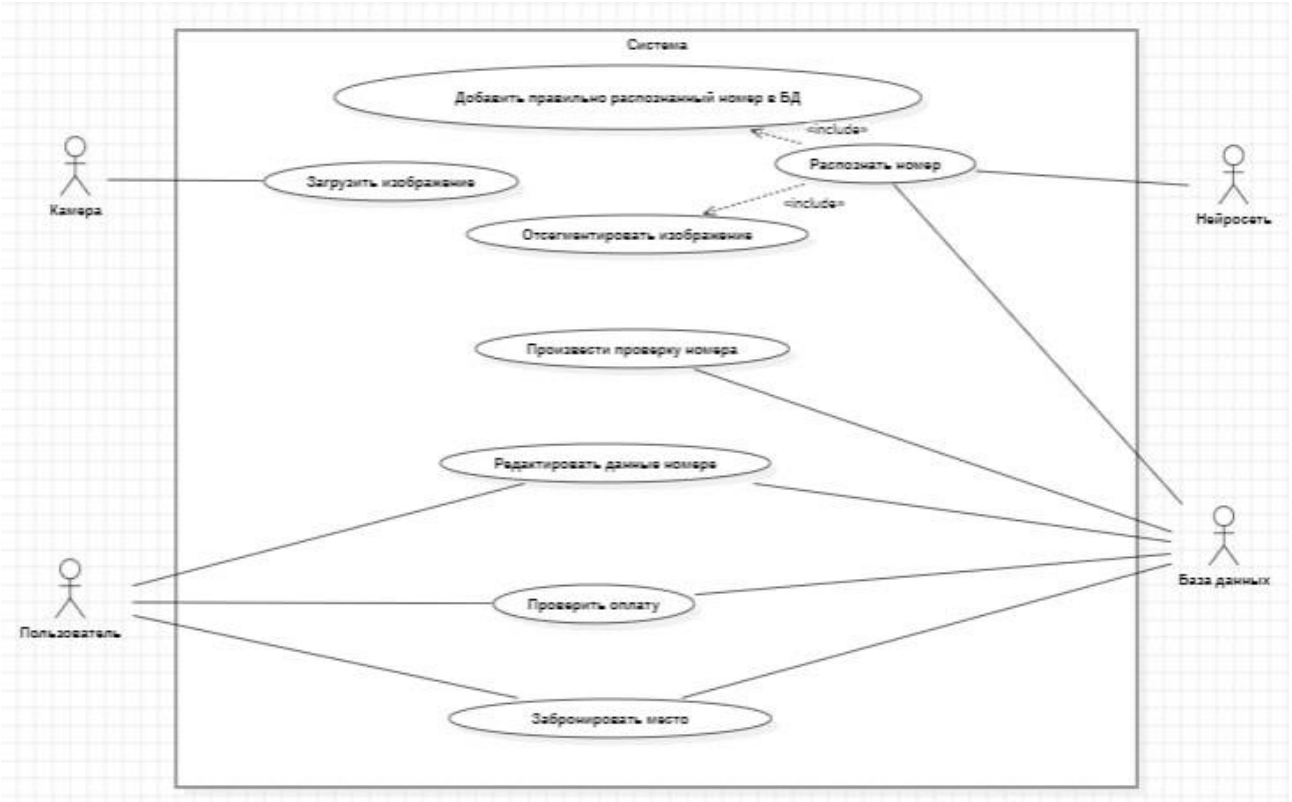


Рисунок 12 – Диаграмма вариантов использования

Проектирование пользовательского интерфейса будет вкорне зависеть от продумывания всего функционала досканально.

Данная диаграмма варинатов использования имеет в себе такие требования как:

– загрузить изображение;
– распознать номер;
– добавить в БД информацию о распознанном номере;
– отсегментировать изображение;
– произвести проверку номера;

– проверить оплату;

– редактировать (добавить, изменить, удалить) данные номера;

– забронировать место.

В данной системе ключевыми пользователями будут являться: сам пользователь системы, администратор, контролирующий функционирование системы.

Исходя из диаграммы вариантов использования будет спроектирован пользовательский интерефейс, необходимый для использования продукта конечным пользователем.

Интерфейс приложения был реализован на подходящей для этого платформе – платформе реализации графического интерфейса Windows Forms.

На рисунке 13 изображен начальный интерфейс приложения.

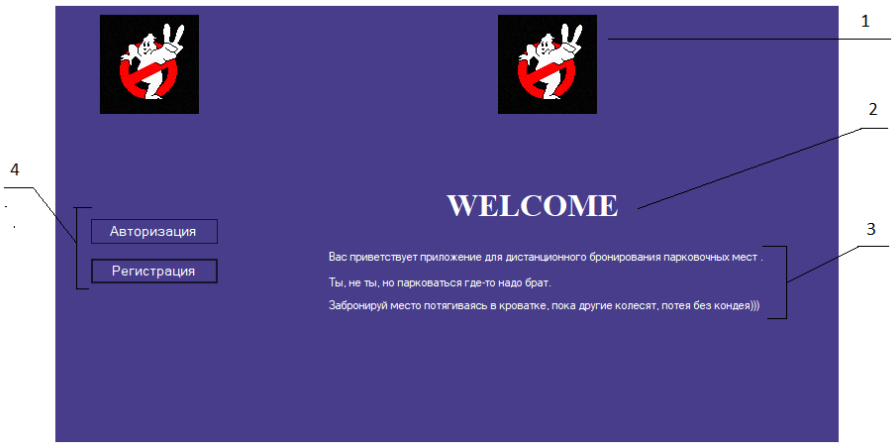


Рисунок 13 – Начальный экран

Рассмотрим основные эелементы интерфейса начального экрана разрабатываемого приложения, представленные на рисунке 13:

1 – логотив приложение, реализванный элементом PictureBox;

2 – приветственное слово, реализованное компонентов Label;

3 – слогон приложения, реализованный TextBox;

4 – кнопка, необходимая для авторизации пользователя, реализованная элементом Button.

Интерфейс вкладки авторизация предназначен для входа пользователя в приложение, если пользователь не создал аккаунт ранее, он может

зарегестрироваться. Реализации окна авторизации представлена на рисунке 14.

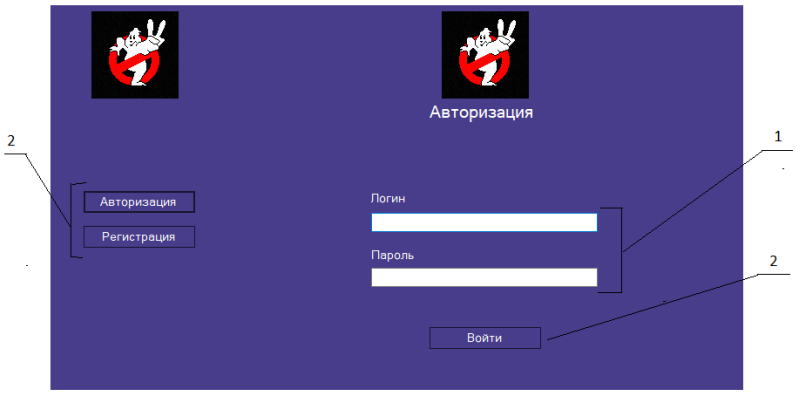


Рисунок 14 – Окно авторизации

Рассмотрим основные элементы интерфейса экрана авторизации разрабатываемого приложения, представленные на рисунке 14:

1 – поля ввода уникального пользователя, реализованные с помощью Edit;

2 – кнопки, выполнующая функцию входа в различные разделы, реализованная элементом Button.

Пользовательский интерфейс авторизации предполагает новому пользователю создать аккаунт, для использования приложения для бронирования парковочных мест. Реализация регистрации представлена на рисунке 15.

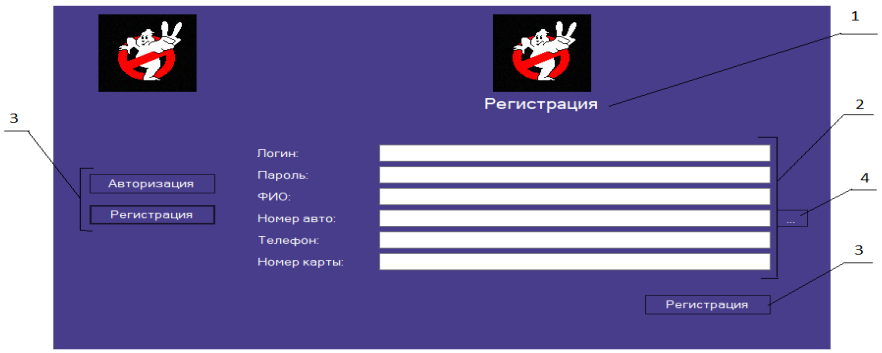


Рисунок 15 – Окно регистрации

Рассмотрим основные элементы интерфейса экрана регистрации разрабатываемого приложения, представленные на рисунке 15:

1 – надпись «Регистрация», реализованныя элементом Label;

2 – компоненты, необходимые для ввода первоначальной информации о новом пользователе, реализованные элементом Edit;

3 – кнопка «Регистрация», необходимая для активации аккаунта нового пользователя, реализована элементом Button;

4 – кнопка раздела «…», имеет возможность сфотографировать номер и система распознает номер и автоматически введет егов поле, реализованы с помощью элемента Button.

Кроме основной идентификации пользователя, ему необходимо ввести личные данные, которые будут занесены в базу данных, для их дальнейшего использования. Реализация ввода данных пользователя представлена на рисунке 16.

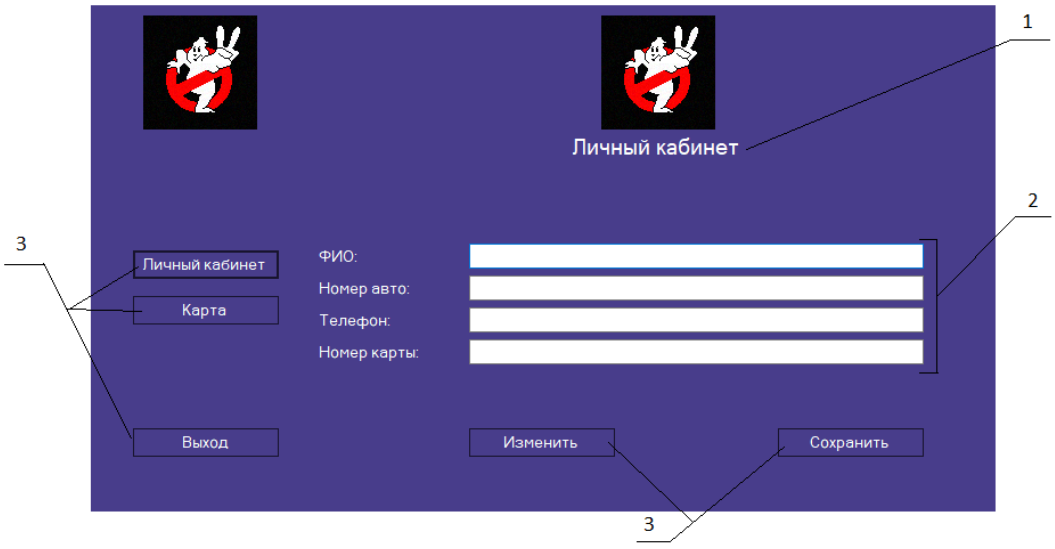


Рисунок 16 – Окно личного кабинета

Рассмотрим основные эелементы интерфейса экрана личного кабинета разрабатываемого приложения, представленные на рисунке 16:

1 – название вкладки, реализованная элементом Label;

2 – поля ввода информации, реалиованные с помощью элемента Edit;

3 – кнопки редактирования данных, позволяющая вернуться к предыдущему разделу, кнопка сохранения данный, реализованные с помощью элемента Button.

Для полноценного бронирования парковки в городе, реализована интерактивная, встроенная карта, с возможностью выбора места для парковки. Реализация данного окна представлена на рисунке 17.

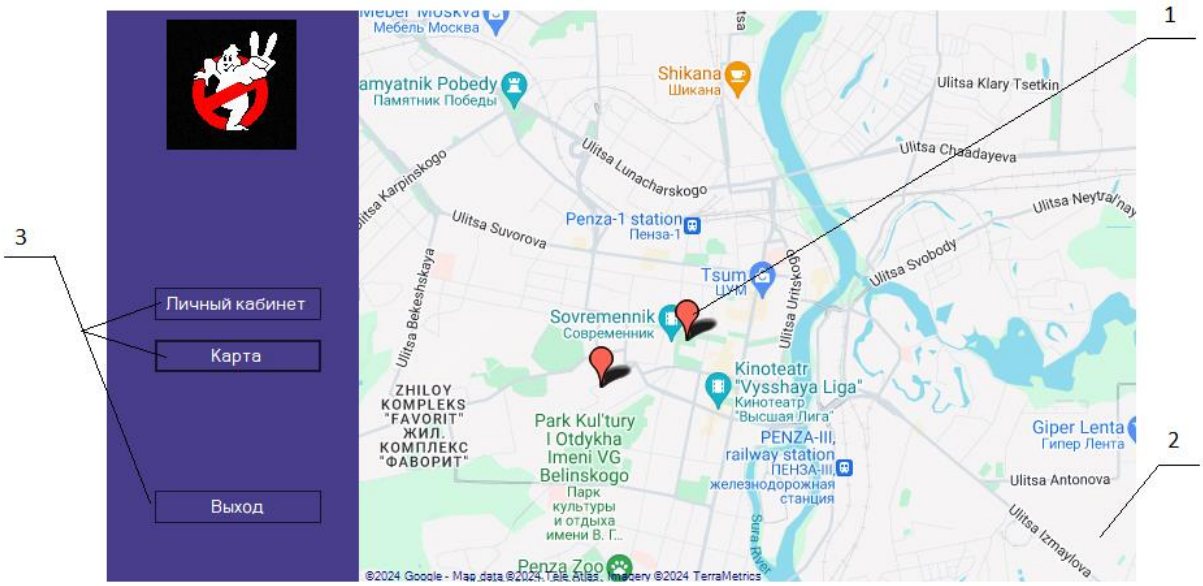


Рисунок 17 – Окно выбора парковки

Рассмотрим основные элементы интерфейса экрана выбора парковки разрабатываемого приложения, представленные на рисунке 17:

1 – метка, показывающая расположение парковки, реализованная с помощью элемента Point;

2 – встроенная карта, показывающая метки на карте так и расположение в городе с возможность. просмотра дорог подъезда в парковки, реализованная с помощью GMap;

3 – кнопки, необходимые для перехода по вкладкам, в частности вкладка «карта» обеспечивает отображение карты и меток расположений парковок на ней, реализованы с помощью элемента Button.

Представленный графический интерфейс является понятным для потенциального пользователя, обладает легкостью понимая, простым интерфейсом, что позволяет сделать вывод о качественном проектировании данного компонента приложения.

3.7 Бюджетирование и планирование проекта

Для реализации приложения необходимо грамотно спланировать все затраты на достижения целей разработки. С помощью планирования расходов, сумму бюджета, можно определить и отследить затраты на все этапы разработки приложения.

При формировании суммы бюджета проекта, а именно приложения, следует обратить внимание на его функциональные возможности, которые способствуют правильному распределению бюджета между отделами проекта.

Самый главные критерий оценки бюджета – функциональные возможности приложения. Исходя из объема реализованного функционала, происходит вычисление ключевых затрат на проектирование модулей. Для избегания критических затрат средств, необходимо привлечение аналитиков, которые сформулирует требования приложения, с выделением основных компонентов, так и компонентов для будущих обновлений.

Дизайнерские решения приложение также являются немаловажным критерием разработки. Визуальная составляющая – есть привлечение. Проработки дизайна будет являться также затратным процессом, в котором будет необходимо работа дизайнерской фокус группы.

Как и во всех приложения, ключевым критерием необходимо выделить расширяемость и поддержка функционирования приложения. Для реализации расширяемости необходимо выделять бюджет на начальном этапе, закладывая их на будущие обновления приложения. Также фактором стабильности работы приложения является удержание его на высоком уровне работы в любых ситуациях, способных нарушить принцип работы. На это решение также необходимы денежные ресурсы. Стабильность работы осуществляется необходимыми действиями, проводимыми до начала запуска приложения, а именно – тщательное тестирование программных модулей. Тестирование важнейший атрибут, позволяющий выявить все недоработки, некорректность и много другое. Тестирование, как и вышеупомянутые пункты затрат, нуждается в затратах.

Для организации планирования процесса разработки и оценки затрат бюджета лучше всего применить диаграмму Ганта.

Диаграмма Ганта – это графическая интерпретация выполнения работ, распределенная согласно утвержденному плану-графику проекта. На ней отражены задачи и последовательность их выполнения.

График работ состоит из отрезков, размещённых вдоль временной оси X. Каждый из них соответствует отдельной задаче или подзадаче. Начало и конец отрезка соответствуют моменту начала и завершения работы по задаче. Длина отрезка – продолжительность работ [22].

Для построения диаграммы необходимо выделить задачи, а также сроки их выполнения. Задачи проекта представлены в таблице 2.

Таблица 2 – Задачи проекта

Название задачи	Длительность задачи в днях	Дата начала	Дата окончания
Старт проекта	0	08.04.2024	08.04.2024
Постановка задачи и анализ аналоговразрабатываемого приложения	3	08.04.2024	11.04.2024
Анализ программ аналогов	1	08.04.2024	09.04.2024
Анализ функциональных требований	1	09.04.2024	10.04.2024
Выбор инструментальных средств проектирования и разработки	1	10.04.2024	11.04.2024
Проектирование приложения	7	12.04.2024	19.04.2024
Проектирование архитектуры приложения	1	12.04.2024	13.04.2024
Проектирование пользовательского интерфейса	1	13.04.2024	14.04.2024
Разработка и обучение нейросети	7	12.04.2024	19.04.2024
Реализация приложения	14	19.04.2024	03.05.2024
Кодирование	10	19.04.2024	29.04.2024
Реализация пользовательского интерфейса	4	29.04.2024	03.05.2024
Тестирование приложения	6	03.05.2024	09.05.2024
Тестирование методом черного ящика	3	03.05.2024	06.05.2024

Продолжение таблицы 2

Название задачи	Длительность задачи в днях	Дата начала	Дата окончания
Тестирование метрикой Холстеда	1	06.05.2024	07.05.2024
Автоматизированное тестирование	2	07.05.2024	09.05.2024
Создание пояснительной записки	13	09.05.2024	22.05.2024
Написание пояснительной записки	8	09.05.2024	17.05.2024
Оформление и печать пояснительной записки	5	17.05.2024	22.05.2024
Конец проекта	0	22.05.2024	22.05.2024

На основе выделенных задач и сроков была реализована диаграмма Ганта, представленная на рисунке 18.

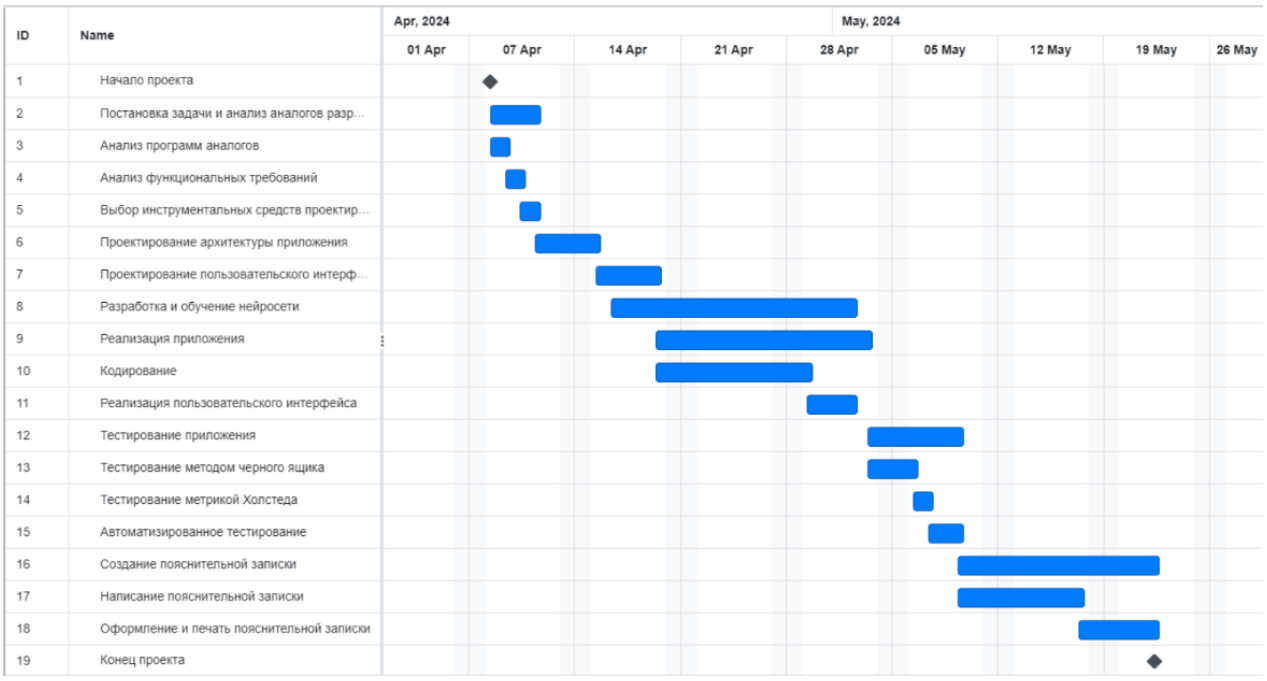


Рисунок 18 – Диаграмма Ганта

Для определения правильности расчета затрат на разработку необходимо определить ресурсы проекта. Выделенные ресурсы приведены в таблице 3.

Таблица 3 – Ресурсы проекта

Название ресурса	Тип	Стандартная ставка	Начисление
Аналитик	Трудовой	150,00 ₽/ч	Пропорциональное
Проектировщик	Трудовой	250,00 ₽/ч	Пропорциональное
Программист	Трудовой	250,00 ₽/ч	Пропорциональное
Дизайнер	Трудовой	200 ₽/ч	Пропорциональное
Тестировщик	Трудовой	150 ₽/ч	Пропорциональное
Компьютер	Материальный	60 000,00 ₽	В начале
Бумага	Материальный	550,00 ₽	В начале
Принтер	Материальный	5 000,00 ₽	В начале

Для описанных ранее задач, на основе ресурсов, была рассчитана стоимость, необходимая для выполнения каждого этапа разработки. Результат представлен в таблице 4.

Таблица 4 – Задачи и затраты проекта

Название задачи	Длительность задачи в днях	Название ресурса	Затраты
Начало проекта	0	Компьютер, Бумага, Принтер	75550,00 ₽
Постановка задачи и анализ аналоговразрабатываемого приложения	3		3600,00 ₽
Анализ программ аналогов	1	Аналитик	1200,00 ₽
Анализ функциональных требований	1	Аналитик	1200,00 ₽
Выбор инструментальных средствпроектирования и разработки	1	Аналитик	1200,00 ₽
Проектирование приложения	7		19600,00 ₽
Проектирование архитектурыприложения	1	Проектировщик	2000,00 ₽
Проектирование пользовательского интерфейса	1	Проектировщик, Дизайнер	3600,00 ₽
Разработка и обучение нейросети	7	Программист	14000,00 ₽
Реализация приложения	14		34400,00 ₽
Кодирование	10	Программист	20000,00 ₽

Продолжение таблицы 4

Название задачи	Длительность задачи в днях	Название ресурса	Затраты
Реализация пользовательского интерфейса	4	Программист, Дизайнер	14400,00 ₽
Тестирование приложения	6		7200,00 ₽
Тестирование методом черного ящика	3	Тестировщик	3600,00 ₽
Тестирование метрикой Холстеда	1	Тестировщик	1200,00 ₽
Автоматизированное тестирование	2	Тестировщик	2400,00 ₽
Создание пояснительной записки	13		15600,00 ₽
Написание пояснительной записки	8	Аналитик	9600,00 ₽
Оформление и печать пояснительнойзаписки	5	Аналитик	6000,00 ₽
Конец проекта	0		

Таким образом, по результатам оценки бюджета разработки приложения для бронирования парковочных мест автостоянки с использованием нейросетевых технологий распознавания автомобильных номеров была составлена диаграмма Ганта, рассчитано время необходимое на полный цикл разработки проекта, которое составляет 43 дня, на основании необходимых ресурсов рассчитан бюджет проекта, который равен 155950 рублей.

Таблица 5 – Задачи и затраты проекта

Название задачи	Длительность задачи в днях	Название ресурса	Затраты
Начало проекта	0	Компьютер, Бумага, Принтер	75550,00 ₽
Постановка задачи и анализ аналогов разрабатываемого приложения	3		3600,00 ₽
Анализ программ аналогов	1	Аналитик	1200,00 ₽

Продолжение таблицы 5

Анализ функциональных требований	1	Аналитик	1200,00 ₽
Выбор инструментальных средствпроектирования и разработки	1	Аналитик	1200,00 ₽
Проектирование приложения	7		19600,00 ₽
Проектирование архитектурыприложения	1	Проектировщик	2000,00 ₽
Проектирование пользовательского интерфейса	1	Проектировщик, Дизайнер	3600,00 ₽
Разработка и обучение нейросети	7	Программист	14000,00 ₽
Реализация приложения	14		34400,00 ₽
Кодирование	10	Программист	20000,00 ₽
Реализация пользовательского интерфейса	4	Программист, Дизайнер	14400,00 ₽
Тестирование приложения	6		7200,00 ₽
Тестирование методом черного ящика	3	Тестировщик	3600,00 ₽
Тестирование метрикой Холстеда	1	Тестировщик	1200,00 ₽
Автоматизированное тестирование	2	Тестировщик	2400,00 ₽
Создание пояснительной записки	13		15600,00 ₽
Написание пояснительной записки	8	Аналитик	9600,00 ₽
Оформление и печать пояснительнойзаписки	5	Аналитик	6000,00 ₽
Конец проекта	0		

Таким образом, по результатам оценки бюджета разработки приложения для бронирования парковочных мест автостоянки с использованием нейросетевых технологий распознавания автомобильных номеров была составлена диаграмма Ганта, рассчитано время необходимое на полный цикл

разработки проекта, которое составляет 43 дня, на основании необходимых ресурсов рассчитан бюджет проекта, который равен 155950 рублей.

4 Реализация приложения

4.1 Реализация преобработки изображения

Первоначальным этапом распознавания является предварительная обработка изображения, полученного с камеры. Преобработка предназначена для предотвращения помех, шума и других искажений, воздействующих на изображение. В данной реализации учета и устранения недостатков, используется простейший и действенный способ преобработки – преобразование цветной картинки в черно-белую. Обработка картинки представлена в листинке 2.

Листинг 2 – Код преобработки изображения

```
image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

В данном коде мы преобразовываем фоновое изображение в нейтральный цвет, в основном это серый, тем самым удаляя ненужные элементы, делая все однообразным. Попадание любых цветовых элементов сильно сказывается на дальнейшей обработке изображения. Результат преобработки представлен на рисунке 19.

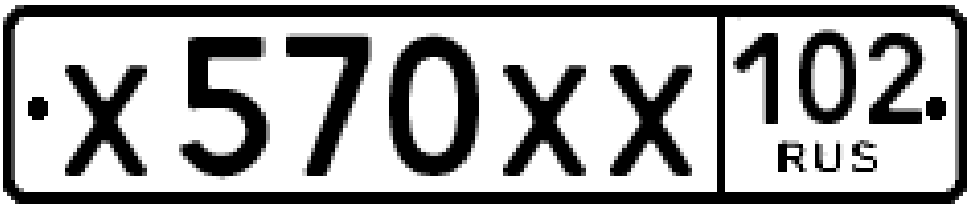


Рисунок 19 – Частичная преобработка изображения

На рисунке 19 видно, что изображение не не имеет четких границ, что может повлиять на процесс дальнего распознавание номера авто. Для устранения этого недостатка был реализован процесс повышения контрастности исходного фото, необходимый для повышения четкости с целью верного распознавания символов номеров. Контрастированное изображение представлено на рисунке 20.

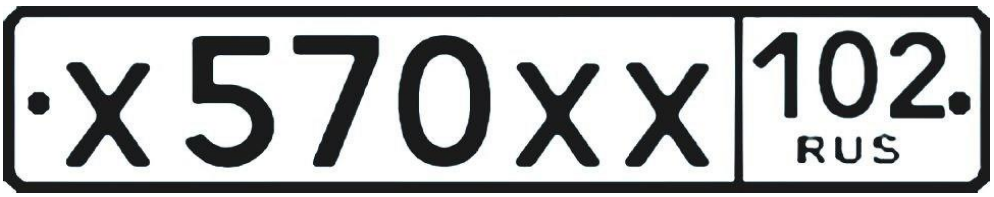


Рисунок 20 – Контрастированное изображение

Процесс контрастирования изображения имеет ключевую цель для правильного распознавания. Отсегментированное изображение увеличивает контраст белого в большую сторону, сформировав четкие границы изображения, элементов черного цвета. Метод, реализующий контрастность представлен в листинге 3.

Листинг 3 – Реализация преобработки изображения

```
cv2.threshold(cropped_image, thresh, 255, cv2.THRESH_BINARY)
```

Метод cv.threshold имеет в себе несколько аргументов. Первый аргумент отвечает за изображение, уже преобразованное в черно-белый формат. Вторым аргументом есть разновидность пиксельных значений изображения, а третьим значением является предельное пороговое значение, обозначающее что это максимальный элемент. Превышающим значение элементам присваивается максимально допустимое, то есть 255. Аргумент cv2.THRESH_BINARY – тип порогового значения [23].

Исходя из вышеперечисленных реализаций процесса преобработки изображения, можно сделать вывод, что процессе преобработки достаточно важный процесс, который реализован в полной мере и удовлетворяет всем требованиям разработки.

4.2 Реализация сегментации изображения

Процесс сегментации изображения наиболее важен, по сравнению с другими начальными преобразованиями. Сегментацией изображения мы реализуем отделение целевого сегмента от общей картинки. Сегментация происходит по множеству слоев, которые, поиском и алгоритмам, выделяют необходимый объект. Сам процесс сегментации реализован с помощью нейронной сети YOLOv5.

Сегментация происходит в несколько этапов. На первом этапе происходит разбиение изображение на множество слоев и формирование параметров,такие как: количество слоев, глубина исследуемого изображения, и много другое. На втором этапе происходит константирование словев изображения для их четкого разграничения. В третьем этапе происходит происходит определение типов слоев. К ним относятся: сверточные слои (Conv), слои сверток группировки (C3) и другие слои.

Для визуализации процесса сегментации, представлено начальное состояние определение границ номера. Определение границ номера представлено на рисунке 21.

Рисунок 21 – Определение границ номера

Данный этап является ключевм для реализации сегментации изображения. В результатеч его получается преобработанное и отсегментированное изображение, которое необходимо для дальнейшего распознавания. Отсегментированное и преобработанное изображение представлено на рисунке 22.

Рисунок 22 – Сегментированне изображение

Исходя из вышеизложенного, можно сделать вывод, что реализация процесса сегментации прошло успешно.

4.3 Реализация распознавания текста на изображении

Для реализации ключегого момента в разработки приложения для бронирования парковочных мест автостоянки с использование нейросетевых технологий распознавания автомобильных номеров является распознавание текста с преобразованного и отсегментированного изображения.

Распознавание номера авто разработано с помощью движка нейросети Tesseract. Данная нейросеть идеально подходит для разработки приложения. Она с высокой точностью считывает символы, находящие преобразованном изображении.

Для повышения эффективности выполнения РАН была применена общедоступная библиотека pytesseract. Библиотека подобного рода является готовым решением для реализации целей разработки приложения. Код использования библиотеки pytesseract представлена в листинге 4.

Листинг 4 – Код библиотеки pytesseract

```
pytesseract.image_to_string(
            cropped_image_gray,
            config='--oem 3 --psm 10 -c
tessedit_char_whitelist=ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789')
```

Разберем данный код. На вход функции подается 2 аргумента:

1. сегментированное изображение (cropped);
2. параметры распознавания (config).

Сегметирование в данной реализации является начальной точкой для формирования распознавания номера. На вход подается изображение, с которого происходит считывание символов. Параметры распознавания предназначены для определения лучшего способа распознавания.

Параметры распознавания относятся к основными элементами процесса распознавания. Параметр oem 3 определеяет режим распознавания текста с помощью нейросети сети для некоторых типов символов. Параметр psm 10 предназначен для обработки одной строки символов, реализуя режим

распознавания единой области определения, отсекая всевозможные случаи скрещивание нескольких областей.

Так как в РАН реализуется распознавание лишь одной базовой строки, в счет единого формата номеров, данные типы режимов удовлетворяют условиям реализации распознавания и разработке приложения. Результат распознавания номера представлен на рисунке 23.

Рисунок 23 – Результат распознавания номера

Распознавание номера и преобразование текста (рисунок 23) было реализованов в полной мере, что является фактором выполнения данного процесса.

4.4 Реализация хранения данных

Хранение данных для приложения является обязательным условием для исполнения определенный видов функций. Реализации их сохранности и целостности приведет работу бронирования и распознавания номеров к стабильноому функционирования приложения. Нарушение целостности и сохранности считается недопустимым, исходя из мер безопасности.

Были спроектирована база данных приложения, удовлетворяющая условиям и функциям. Схема базы данных (БД) представлена на рисунке 24.

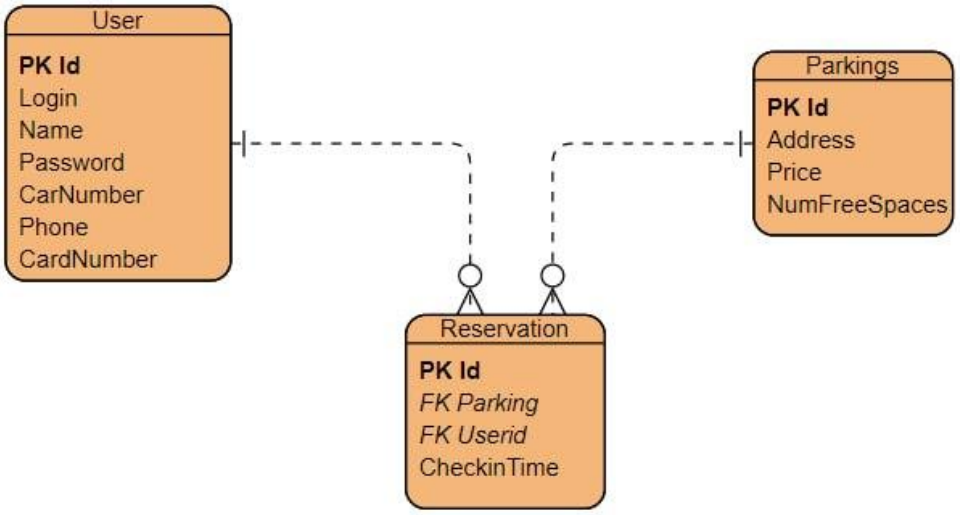


Рисунок 24 – Схема базы данных

БД состоит из трех основных таблиц:

– пользователь (User);

– парковка (Parkings);

– резервирование (Reservation).

Главной таблицей, содержащей все данные, необходимые для бронирования парковочного места является Reservation. Ее структура имеет в своем составе внешние ключи двух таблиц, хранящие в себе информацю о пользователе (User) и парковке (Parkings). Связи между таблица организованы по принципу 1:M. Это, например, говорит от том, что у одной парковки может быть много бронирований.

БД взаимодействует с фреймворком «EF Core», который упрощает работу с хранилицем данных. Пример кода метода, взаимодействующего с EF Core представлен в листинге 5.

Листинг 5 – Код метода взаимодействия с базой данных

```
public async Task<int> GetPriceByParkingIdAsync(Guid parkingId) =>
        (await _storageCarNumbersContext.Parkings
            .FirstOrDefaultAsync(x => x.Id == parkingId))!
            .Price;
}
```

Метод GetPriceByParkingIdAsync – получает цену парковки по ID. В данноме тоде используется метода фреймворка, а именно FirstOrDefaultAsync, который исполняет отправку запроса в БД. Остальные методы класса

StorageCarNumbersRepository, реализующего хранения номера авто,а также остальные методы представлены в приложении Б.

4.5 Реализация пользовательского интерфейса

Так как требования к пользовательскому интерфейсу были описаны ранее, необходимо создать работоспособную модель интерфейса, чтобы конечный пользователь смог взаимодействовать с ним.

В следствии чего был реализованы основные разделы приложения, в частности:

– приветственное окно;
– окно авторизации;
– окно регистрации;
– личный кабинет;
– карта;
– окно бронирования.

Приветственное окно приложения является начальной точкой приложения и имеет кнопки для авторизации и регистрации пользователя. Приветственное окно представлено на рисунке 25.

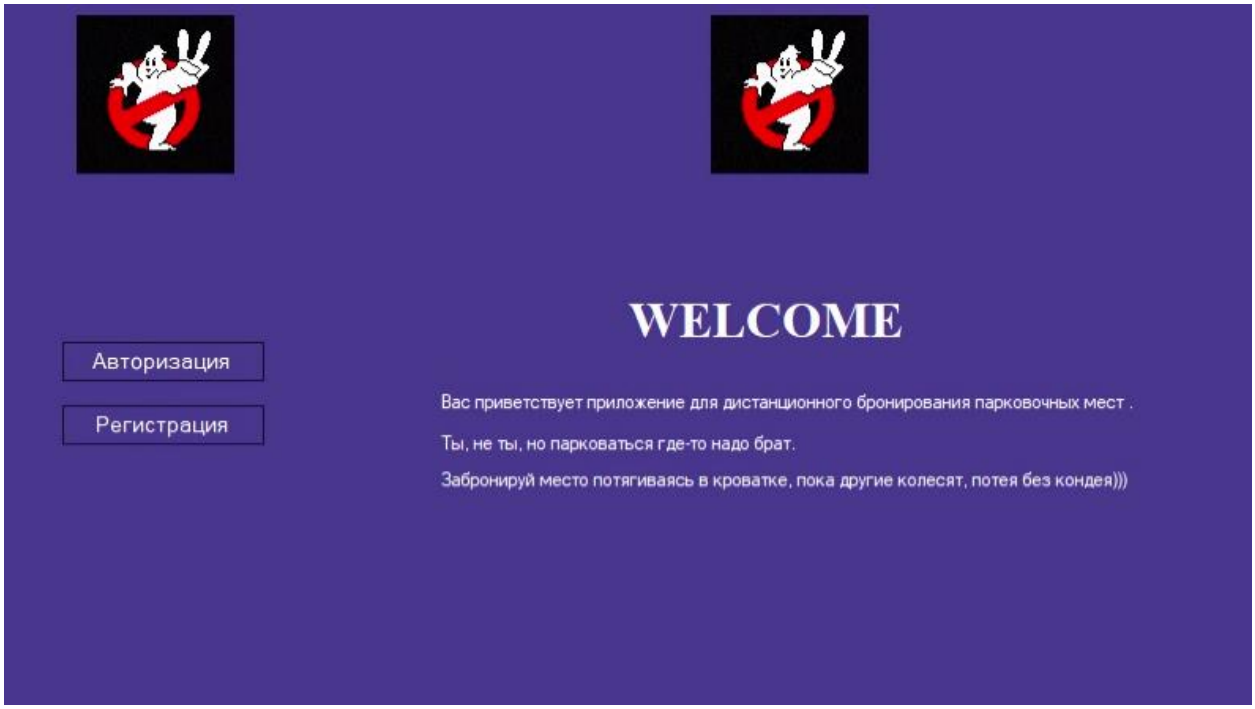


Рисунок 25 – Приветственное окно

При нажатии кнопки «Авторизация», пользователь переместится в соответсвующее окно, для прохождения индентификации. Где нужно ввести логин и пароль. Окно «Авторизация» представлено на рисунке 26.

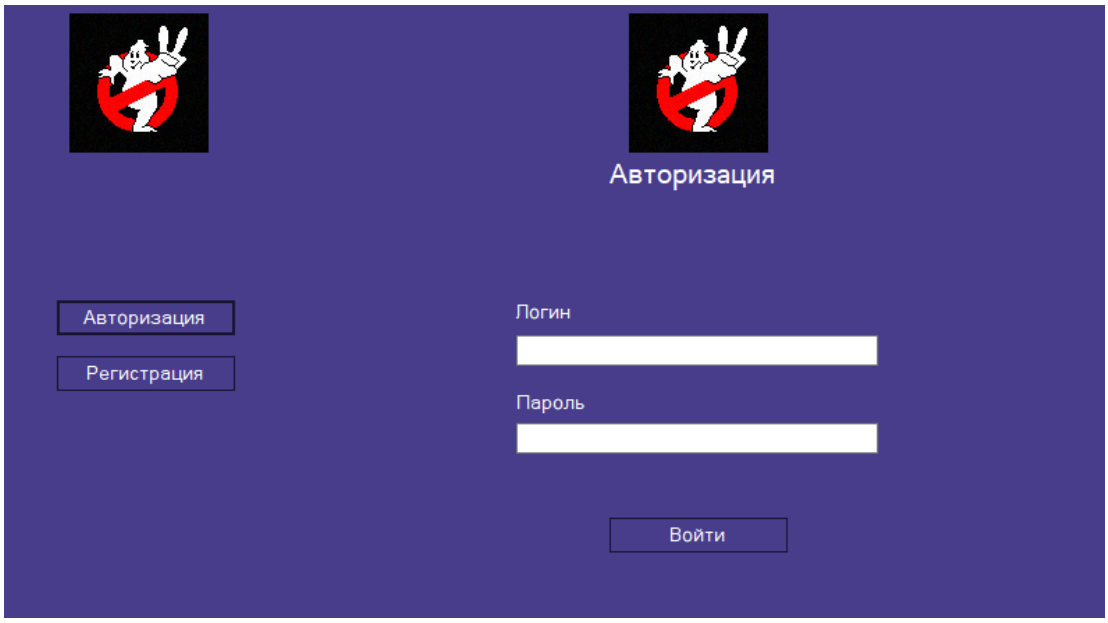


Рисунок 26 – Окно «Авторизация»

При нажатии кнопки «Регистрация», пользователь перенаправляется в окно регистрации, где происходит первичный ввод данных пользоваля. Окно «Регистрация» представлено на рисунке 27.

Рисунок 27 – Окно «Регистрация»

После прохождения авторизации или регистрации, пользователь имеет возможность зайти раздел «Личный кабинет», где может проверить свои данные. Окно «Личный кабинет» представлено на рисунке 28.

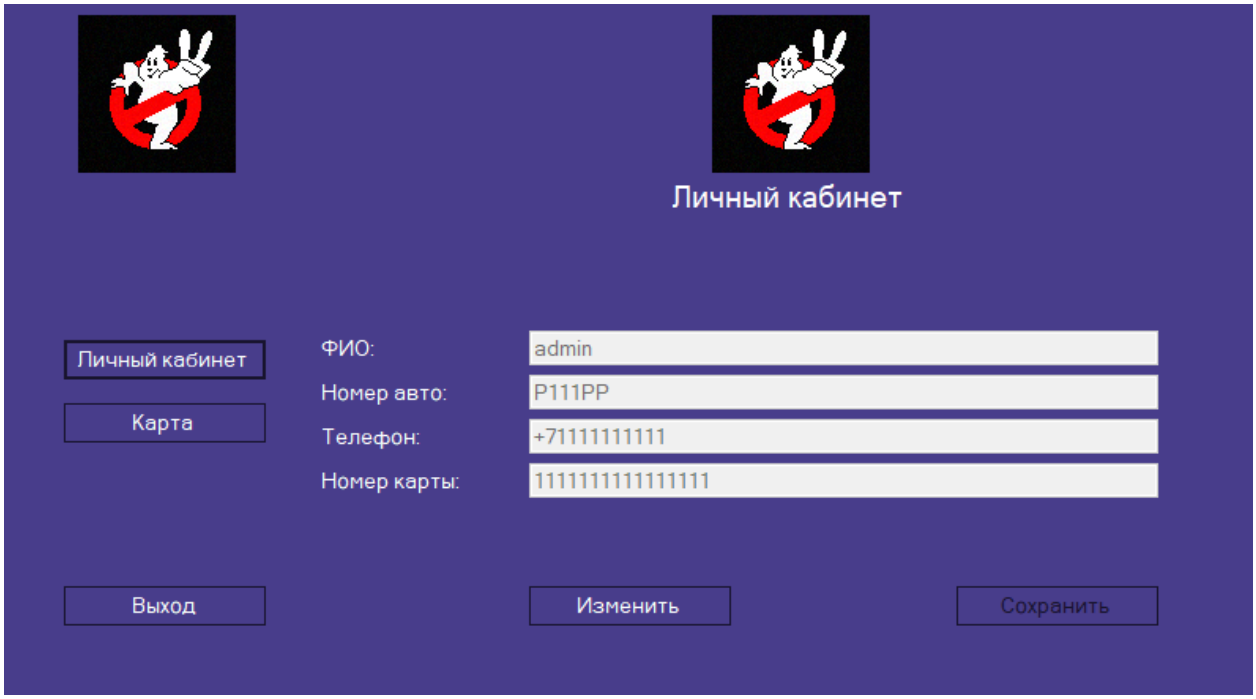


Рисунок 28 – Окно «Личный кабинет»

Как только пользователь ввел данные, он может перейти во вкладку «Карта», для выбора оптимальной парковки. Окно «Карта» представлено на рисунке 29.

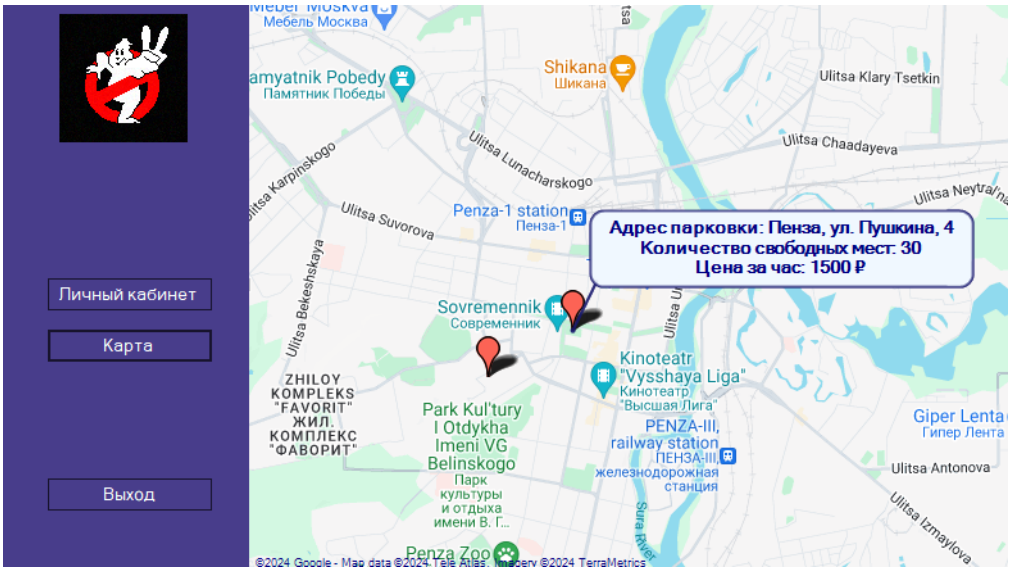


Рисунок 29 – Окно «Карта»

При выборе месторасположения нужной парковки, пользователь вправе выбрать дату и время заезда. Оно реализуется в окне «Бронирование». Окно «Карта» представлено на рисунке 30.

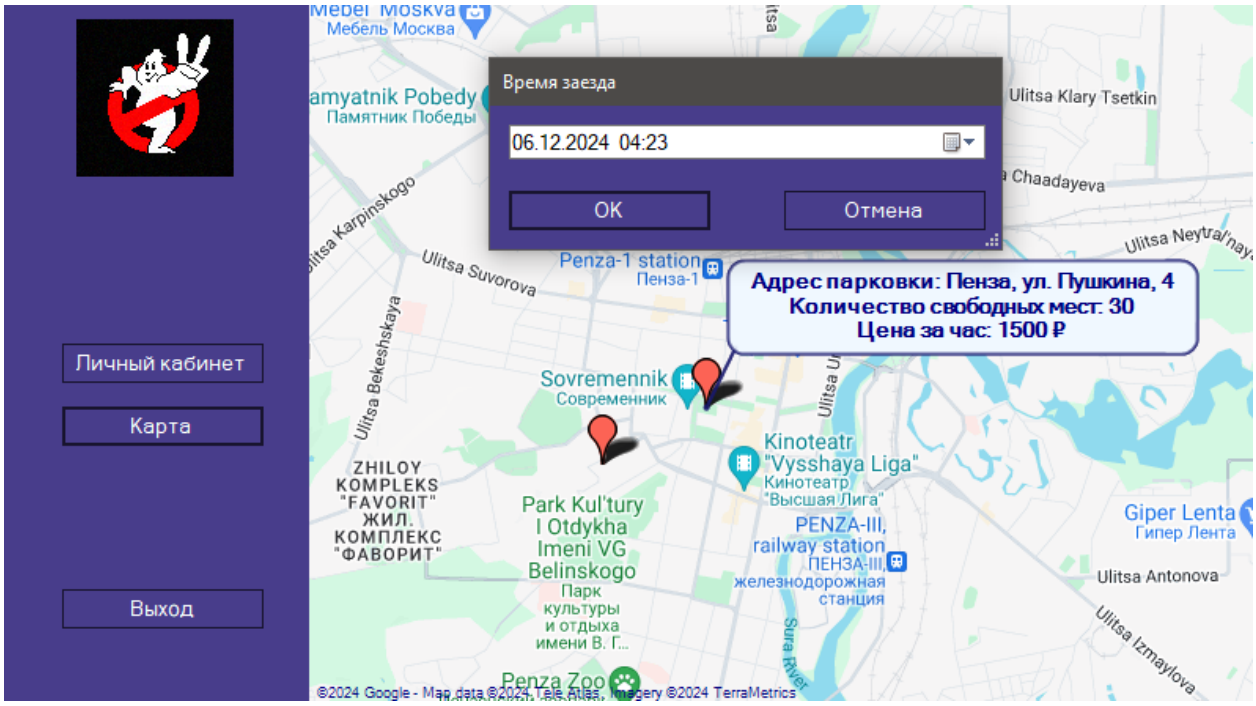


Рисунок 30 – Окно «Бронирование»

Исходя из реализованное интерфейса приложения, можно сказать о том, что были достигнуты все цели, представляемые к его разработке. Описание методов классов клиентского приложение представлена в приложении Д.

5 Тестирование приложения

5.1 Тестирование черным ящиком

5.1.1 Общие сведения

Функциональное тестирование основывается на проверки функций и особенностей ПО, а также его поведения и взаимодействия с другими системами на системном, модульном, интеграционном и приемочном уровнях тестирования.

Тестирование чёрным ящиком – стратегия (метод) тестирования функционального поведения объекта (программы, системы) с точки зрения внешнего мира, при котором не используется знание о внутреннем устройстве (коде) тестируемого объекта. Иначе говоря, тестированием чёрного ящика занимаются тестировщики, не имеющие доступ к исходному коду приложения. Под стратегией понимаются систематические методы отбора и создания тестов для тестового набора.

Тестирование черным ящиком позволяет определить реакцию системы на ожидаемые и неожиданные действия пользователя, время реакции и позволяет выявить проблемы с надежностью.

В рамках ВКР было проведено тестирование черным ящиком алгоритма распознавания автомобильных номеров по следующим методикам:

– функциональная диаграмма и таблица решений;
– модульное тестирование.

Ход и результаты тестирования представлены ниже.

5.1.2 Функциональная диаграмма и таблица решений

Функциональная диаграмма – один из методик стратегии метода тестирования черным ящиком, которая показывает, причинно-следственную связь между различными вариантами ошибок в выполнении и корректной работой системы.

В ходе тестирования этой методики необходимо было описать все причины, по которым система может функционировать некорректно и

выдавать соответствующие ошибки, а также условия корректной работы и реализовать схематичную работу системы и описать в соответствующей таблице. Условия неработоспособности недопустимы.

Было сформулировано задание:

Пользователю необходимо осуществить бронирование парковочного места. Если человек не введет какой-либо из пунктов, необходимый для регистрации, то бронирование не будет осуществлено. Главная задача корректно внести все данные и загрузить фото номера автомобиля, после чего его данные буду в базе, и процесс бронирования будет доступен.

Причины и следствия описаны в таблице 6.

Таблица 6 – Причины и следствия

П1	Зарегистрироваться
П2	Авторизоваться корректно
П3	Личные данные введены
П4	Фото загружено корректно
П5	Выбрано местоположение
С1	Введите данные для регистрации
С2	Данные введены некорректно
С3	Введите личные данные
С4	Фото загружено некорректно
С5	Место местоположение парковки не выбрано
С6	Бронирование прошло успешно

Функциональная диаграмма представлена на рисунке 31.

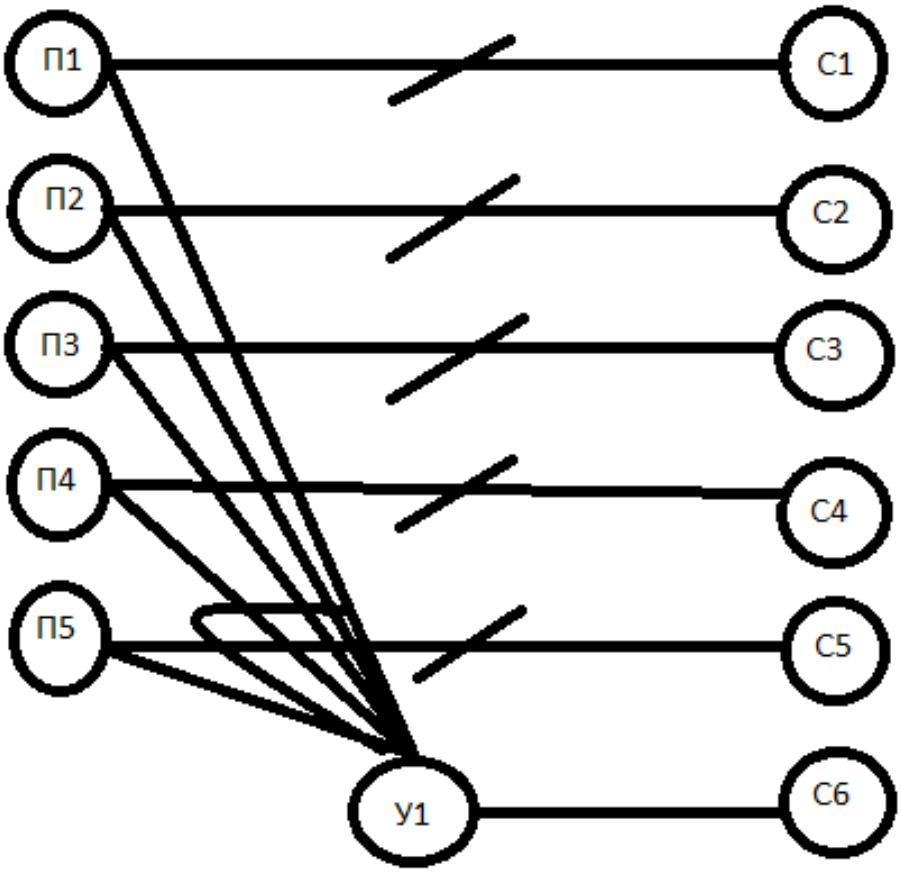


Рисунок 31 – Функциональная диаграмма

На основе диаграммы составляем таблицу решений. В верхней части – комбинация причин, в нижней – соответствующее следствие на каждую комбинацию причин.

Таблица решений представлена в таблице 7.

Таблица 7 – Таблица решений

н/н	1	2	3	4	5	6	7	8	9
П1	0	1	1	1	1	1	1	0	0
П2	0	0	1	1	1	1	1	0	0
П3	0	0	0	1	1	1	0	1	1
П4	0	0	0	0	1	1	0	1	1
П5	0	0	0	0	0	1	0	0	1

Продолжение таблицы 7

н/н	1	2	3	4	5	6	7	8	9
C1	1	0	0	0	0	0	0	1	1
C2	1	1	0	0	0	0	0	1	1
C3	1	1	1	0	0	0	1	0	0
C4	1	1	1	1	0	0	1	0	0
C5	1	1	1	1	1	0	1	1	0
C6	0	0	0	0	0	1	0	0	0

Тест – кейсы для таблицы 7 реализованы в таблице 8.

Таблица 8 – Тест -кейсы

№	Входные данные	Выходные данные
1	Регистрация: неопределено Авторизация: неопределено Личные данные: неопределено Загрузка фото: неопределено Определить время заезда: неопределено	Сообщение: «Данные введены не корректно» Сообщение: «Логин и пароль не были введены» Сообщение: «Данные введены не верно Сообщение: «Данные введены не корректно» Сообщение: «Время заезда установлено автоматически»
2	Регистрация: определено Авторизация: неопределено Личные данные: неопределено Загрузка фото: неопределено Определить время заезда: неопределено	Сообщение: «Логин и пароль не были введены» Сообщение: «Данные введены не верно Сообщение: «Данные введены не корректно» Сообщение: «Время заезда установлено автоматически»
3	Регистрация: определено Авторизация: определено Личные данные: неопределено Загрузка фото: неопределено Определить время заезда: неопределено	Сообщение: «Данные введены не верно Сообщение: «Данные введены не корректно» Сообщение: «Время заезда установлено автоматически»

Продолжение таблицы 8

№	Входные данные	Выходные данные
4	Регистрация: определено Авторизация: определено Личные данные: определено Загрузка фото: неопределено Определить время заезда: неопределено	Сообщение: «Данные введены не корректно» Сообщение: «Время заезда установлено автоматически»
5	Регистрация: определено Авторизация: определено Личные данные: определено Загрузка фото: определено Определить время заезда: неопределено	Сообщение: «Время заезда установлено автоматически»
6	Регистрация: определено Авторизация: определено Личные данные: определено Загрузка фото: определено Определить время заезда: определено	Сообщение: «Вы забронировали»

Таким образом, тестирование функциональной диаграммы и реализация таблицы решений с тест-кейсами выполнено успешно. Результаты функционального тестирование представлены в приложении Е.

5.1.3 Метрика качества кода

Метрика качества кода представляет из себя показатели оценки разрабатываемого продукта. Она реализует полноту предоставляемой информации по разработанному программному коду. С помощью метрик кода, любой член команды проекта может оценивать качество проведенныых работ, объемы выполнения и многие другие факторы, влияющие на качество разрабатываемого продукта, а также метрики помогают оценить затраченные ресурсы команды.

Комплексный показатель качества кода (Maintainability Index) является самой применимой метрикой. 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;

LoC (Lines of Code) – количество строк кода.

Принимаемый диапазон значений может достигать максимальное и минимальное значение от 0 до 100. Значения показывают инверсию легкости поддержки кода, а именно, чем выше значение метрики, тем проще поддержка реализуемого кода [24].

Для расчета вычислительной сложности программы имеет смысл произвести вычисление словоря и длины программы, а только после произвести расчет HV. Словарь программы рассчитывается по формуле:

$$\eta = \eta 1 + \eta 2,$$

где η1 – число простых операторов и операций в программе;

η2 – число простых операндов (констант и переменных) в программе.

Длина программы рассчитывается по следующей формуле:

$$N = N1 + N2,$$

где N1 – общее число простых операторов и операций в программе;

N2 – общее число простых операндов в программе.

Вычислительная сложность рассчитывается по формуле:

$$HV = N * \log 2\ \eta.$$

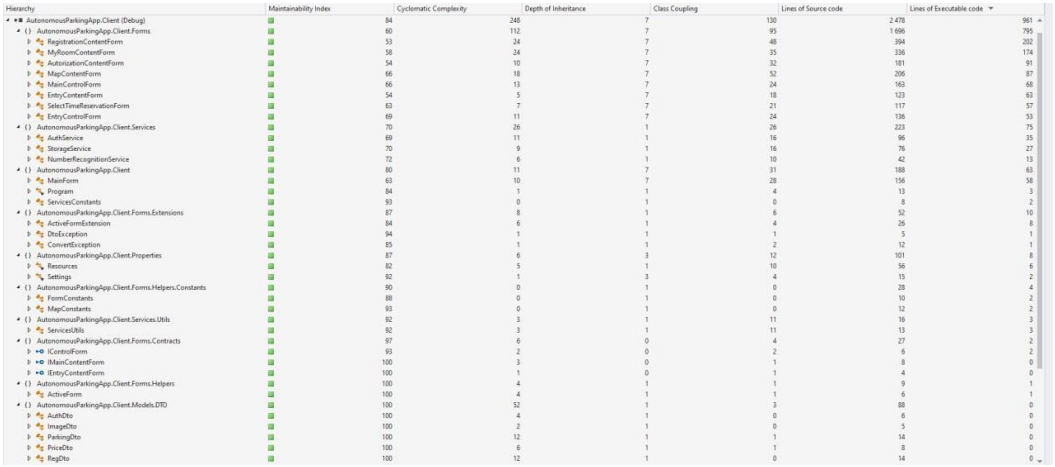
Структурную сложность программы можно рассчитать по следующей формуле:

$$CC = КОР + КОЦ + 1,$$

где $КОР$ – количество операторов разветвления;

$КОЦ$ – количество операторов цикла.

Среда разработки Microsoft Visual Studio 2022 позволяет автоматически вычислять метрики кода. Для разрабатываемого приложения были произведены вычисления метрик кода с помощью Microsoft Visual Studio 2022. Результаты метрик представлены на рисунке 32.



Hierarchy	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Source code	Lines of Executable code
AutonomousParkingApp.Client (Debug)	84	248	7	130	2 478	961
AutonomousParkingApp.Client.Forms	60	112	7	95	1 696	795
RegistrationContentForm	53	24	7	48	394	202
MyRoomContentForm	58	24	7	35	336	174
AutorizationContentForm	54	10	7	32	181	91
MapContentForm	66	18	7	52	206	87
MainControlForm	66	13	7	24	163	68
EntryContentForm	54	5	7	18	123	63
SelectTimeReservationForm	63	7	7	21	117	57
EntryControlForm	69	11	7	24	136	53
AutonomousParkingApp.Client.Services	70	26	1	26	223	75
AuthService	69	11	1	16	96	35
StorageService	70	9	1	16	76	27
NumberRecognitionService	72	6	1	10	42	13
AutonomousParkingApp.Client	80	11	7	31	188	63
MainForm	63	10	7	28	156	58
Program	84	1	1	4	13	3
ServicesConstants	93	0	1	0	8	2
AutonomousParkingApp.Client.Forms.Extensions	87	8	1	6	52	10
ActiveFormExtension	84	6	1	4	26	8
DtoException	94	1	1	1	5	1
ConvertException	85	1	1	2	12	1
AutonomousParkingApp.Client.Properties	87	6	3	12	101	8
Resources	82	5	1	10	56	6
Settings	92	1	3	4	15	2
AutonomousParkingApp.Client.Forms.Helpers.Constants	90	0	1	0	28	4
FormConstants	88	0	1	0	10	2
MapConstants	93	0	1	0	12	2
AutonomousParkingApp.Client.Services.Utils	92	3	1	11	16	3
ServicesUtils	92	3	1	11	13	3
AutonomousParkingApp.Client.Forms.Contracts	97	6	0	4	27	2
IControlForm	93	2	0	2	6	2
IMainContentForm	100	3	0	1	8	0
IEntryContentForm	100	1	0	1	4	0
AutonomousParkingApp.Client.Forms.Helpers	100	4	1	1	9	1
ActiveForm	100	4	1	1	6	1
AutonomousParkingApp.Client.Models.DTO	100	52	1	3	88	0
AuthDto	100	4	1	0	6	0
ImageDto	100	2	1	0	5	0
ParkingDto	100	12	1	1	14	0
PriceDto	100	6	1	1	8	0
RegDto	100	12	1	0	14	0

Рисунок 32 – Метрики исходного кода

Исходя из предоставленных данных, комплексный показатель кода (MI) находится на высоком уровне, так как значение приближается к 100, а именно 84, что является случаем легкой поддержки кода приложения.

Большой глубиной наследования в данном проекте обладают классы-окна (формы), описывающие графический интерфейс приложения. Остальные классы обладают глубиной наследования не более 3. Данная характеристика показывает, что интерфейс просто в понимании и использовании. Код также понятен для будущих разработчиков проекта.

Чем больше связность классов, тем сложнее поддерживать данные классы. По результатам вычисленных метрик (смотри рисунок 32) показатель взаимозависимости относительно невысокий.

Реализовав анализ метрик исходного кода, можно сделать вывод о том, что приложение для бронирования парковочных мест автостоянки с использованием нейросетевых технологий распознавания автомобильных номеров разработано качественно.

5.1.4 Метрика SLOC

Метрика SLOC представляет собой метрику количества строк кода. Подсчет данной метрики был проведен для сервиса, отвечающего за хранение автомобильных номеров. Код модуля приведен в приложении Б.

В данной программе имеем:

– физических строк: 92;

- логических строк: 68;
- количество строк комментариев: 0;
- пустые строки: 24.

Таким образом, 68 логических строк кода более полно отражают размер программы, при условии наличия пустых строк.

В процесее выполнения работы было проведено функциональное тестирование разрабатываемой системы. По результатам тестирования можно сказать о надежности, правильности работы приложения.

На этапе анализа качества кода был рассмотрен комплексный показатель качества кода, находящийся на достаточно высоком уровне и лишь в некоторых классах опускается ниже 84 %, что свидетельствует об относительной простоте поддержки исходного кода.

Заключение

В результате выпускной квалификационной работы была произведена разработка приложения для бронирования парковочных мест автостоянки с использование нейросетевых технологий распознавания автомобильных номеров, включающая в себя нейросеть распознавания номеров, клиентское приложения для бронирования и серверную часть. В корне работы заложены алгоритмы распознавания номера, его преобразования в нужную форму, обеспечивающие быстродействие и качество распознавания.

Клиентское приложение было реализовано на языке C#, реализующем объектно-ориентированный подход к программированию. База данные реализована с использованием фреймворка «EF Core», позволяющим полноценно работать с реализованной базой данных приложения. Распознавание номеров реализовано на языке Python, с использование различных готовых решений, применяемых для облегчения процесса реализации распознавания. Все эти компоненты разработки были интегрированы в одно целое в системе интеграционной разработки Microsoft Visual Studio 2022.

Кроме всего прочего было произведено функциональное тестирование, которое показало, что система работает на высоком уровне и удовлетворяет всем требования надежности и безопасности.

Было произведено планирование бюджета реализуемого проекта с помощью онлайн ресурса, выполняющий функции отрисовки диаграммы Ганта с заданными ресурсами проекта. В ходе проектирования было выявлено, что разработка проекта равна 43 дням, на основании необходимых ресурсов рассчитан бюджет проекта, который равен 155950 рублей.

Таким образом, можно сделать вывод о том, что в результате выполнения выпускной квалификационной работы было разработано приложение для бронирования парковочных мест автостоянки с использованием нейросетевых технологий распознавания автомобильных номеров, гарантирующее реализацию всех функций приложения.

Список использованных источников

1. Mafin Media // Автомобильные номера в России: типы, коды и регионы. [Электронный ресурс] URL: https://dzen.ru/a/Yk_8MoPl_XdHV6cp#:~:text=Сегодня%20большинство%20номерных%20знаков%20в,индекс%20(код)%20региона%20Российской%20Федерации (дата обращения 10.04.2024).

2. Нейронная сеть // Википедия. [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/Нейронная_сеть (дата обращения 10.04.2024)/

3. Алгоритмы обучения нейронных сетей // ETXT. [Электронный ресурс] URL: https://www.etxt.ru/subscribes/algoritmy-obucheniya-neyronnykh-setey/ (дата обращения 10.04.2024).

4. Что такое Детекция объектов? // Mathinson [Электронный ресурс] URL: https://mathison.pro/chto-takoe-detekciya-obektov/ (дата обращения 13.04.2024).

5. Использование YOLOv5 для задачи детекции // NTA. [Электронный ресурс] URL: https://vc.ru/newtechaudit/326571-ispolzovanie-yolov5-dlya-zadachi-detekcii (дата обращения 16.04.2024).

6. Анализ технологий распознавания текста на изображении // К.А. Бобров, В.Д. Шульмин, К.П. Власов [Электронный ресурс] URL: https://cyberleninka.ru/article/n/analiz-tehnologiy-raspoznavaniya-teksta-iz-izobrazheniya (дата обращения 18.04.2024).

7. «Автомаршал» – система контроля проезда, управление доступом и учета автотранспорта [Электронный ресурс]. URL: https://www.avtomarshal.ru/ (дата обращения 25.04.2024).

8. Обзор Алгоритмов сегментации // NastyaL [Электронный ресурс]. URL:https://habr.com/ru/companies/intel/articles/266347/ (дата обращения 30.04.2024).

9. Задача нахождения объектов на изображении // neerc [Электронный ресурс]. URL:

https://neerc.ifmo.ru/wiki/index.php?title=Задача_нахождения_объектов_на_изображении (дата обращения: 03.05.2024).

10. Койнова, Т. А. Алгоритмы распознавания символов / Т. А. Койнова. – Текст : непосредственный // Молодой ученый. – 2022. – № 18 (413). – С. 73-76. – URL: https://moluch.ru/archive/413/91060/ (дата обращения: 05.05.2024).

11. Мурсалиев М.Х. Сравнительный анализ математических методов распознавания образов // М.Х. Мурсалиев [Электронный ресурс]. URL: https://na-journal.ru/4-2023-informacionnye-tekhnologii/4990-sravnitelnyi-analiz-matematicheskih-metodov-raspoznavaniya-obrazov (дата обращения: 09.05.2024).

12. Что такое датасет и разметка данных: рассказываем самыми простыми словами // LabelMe [Электронный ресурс]. URL: https://dzen.ru/a/YKY53ezkPgyOc4Yr (дата обращения: 12.05.2024).

13. Как мы распознаем фото документов пользователей // ilgrad [Электронный ресурс]. URL: https://habr.com/ru/companies/inDrive/articles/598193/ (дата обращения: 12.05.2024).

14. Ultralytics YOLOv5. Архитектура [Электронный ресурс]. URL: https://docs.ultralytics.com/ru/yolov5/tutorials/architecture_description/#2-data-augmentation-techniques (дата обращения: 13.05.2024).

15. Кондратьева А.А. Почему мы пока не можем компьютерно распознать рукописный архив // А.А. Кондратьева [Электронный ресурс]. URL: https://www.nevmenandr.net/nazirov/journal/2017_2_6_recognition.pdf с.87 с. 87 (дата обращения: 14.05.2024).

16. Python // Глоссарий [Электронный ресурс]. URL: https://blog.skillfactory.ru/glossary/python/ (дата обращения: 16.05.2024).

17. Язык программирования C#: краткая история, возможности и перспективы // Space Police [Электронный ресурс]. URL: https://timeweb.com/ru/community/articles/chto-takoe-csharp (дата обращения: 16.05.2024).

18. Adegeo. Руководство по классическим приложениям (Windows Forms .NET) // Adegeo, Anderjef, Alexbuckgit [Электронный ресурс]. URL: https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/overview/?view=netdesktop-8.0 (дата обращения: 20.05.2024).

19. Валерий. Монолитная и микросервисная архитектура: что лучше выбрать? // PurpleSchool [Электронный ресурс]. URL: https://purpleschool.ru/blog/monolith-vs-microservices (дата обращения: 21.05.2024).

20. Анна Ушакова. Что такое микросервисная архитектура // Академия Selected [Электронный ресурс]. URL: https://selectel.ru/blog/what-is-microservice-architecture/ (дата обращения: 22.05.2024).

21. Привязка модели в ASP.NET Core // Learn [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/aspnet/core/mvc/models/model-binding?view=aspnetcore-8.0 (дата обращения: 24.05.2024).

22. Разбираем диаграмму Ганта – инструмент, который должен знатькаждый менеджер // Skillbox [Электронный ресурс] URL: https://skillbox.ru/media/management/razbiraem-diagrammu-ganta-instrument-kotoryy-dolzhen-znat-kazhdyy-menedzher/ (дата обращения: 27.05.2024).

23. Simple Thresholding // Open source Computer Vision [Электронный ресурс]. URL: https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html 30.05.2024) (дата обращения: 30.05.2024).

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

Приложение А
(справочное)
Глоссарий

Распознавание автомобильных номеров (РАН) – методы, реализующие распознавание автомобильных номеров, с целью преобразования цифрового изображения в текст.

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.

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

Приложение Б

(рекомендуемое)

Листинг клиентского приложения C#

Листинг Б.1 – Класс «AuthController.cs»

```
using AutonomousParkingApp.Authentication.Exceptions;
using AutonomousParkingApp.Authentication.Models.DTO;
using AutonomousParkingApp.Authentication.Services.Contracts;
using Microsoft.AspNetCore.Mvc;

namespace AutonomousParkingApp.Authentication.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class AuthController : ControllerBase
    {
        private readonly IAuthService _authService;

        public AuthController(IAuthService authService) =>
            _authService = authService;

        [HttpPost("add-user")]
        public async Task<ActionResult> AddUserAsync([FromBody] RegDto user)
        {
            try
            {
                await _authService.AddUserAsync(user);

                return NoContent();
            }
            catch (AuthException ex)
            {
                return Problem(ex.Message, statusCode: ex.StatusCode);
            }
        }

        [HttpPut("update-user")]
        public async Task<ActionResult> UpdateUserAsync([FromBody] UserDto
user)
        {
            try
            {
                await _authService.UpdateUserAsync(user);

                return NoContent();
            }
            catch (AuthException ex)
            {
                return Problem(ex.Message, statusCode: ex.StatusCode);
            }
        }

        [HttpGet("get-user/{userId}")]
        public async Task<ActionResult<UserDto>>
GetUserByUserIdAsync([FromRoute] Guid userId)
        {
            try
            {
                var result = await _authService.GetUserByUserIdAsync(userId);

                return Ok(result);
            }
            catch (AuthException ex)
            {
                return Problem(ex.Message, statusCode: ex.StatusCode);
            }
        }
```

```
        [HttpPost("get-user-by-login-password")]
        public async Task<ActionResult<UserDto>> GetUserAsync([FromBody]
AuthDto authDto)
        {
            try
            {
                var result = await _authService.GetUserAsync(authDto);

                return Ok(result);
            }
            catch (AuthException ex)
            {
                return Problem(ex.Message, statusCode: ex.StatusCode);
            }
        }

        [HttpGet("get-userId-by-login/{login}")]
        public async Task<ActionResult<Guid>>
GetUserIdByLoginAsync([FromRoute] string login)
        {
            try
            {
                var result = await _authService.GetUserIdByLoginAsync(login);

                return Ok(result);
            }
            catch (AuthException ex)
            {
                return Problem(ex.Message, statusCode: ex.StatusCode);
            }
        }
    }
}
```

Листинг Б.2 – Класс «AuthContext.cs»

```
using AutonomousParkingApp.Authentication.DB.Models;
using Microsoft.EntityFrameworkCore;

namespace AutonomousParkingApp.Authentication.DB.Contexts;

public class AuthContext : DbContext
{
    private readonly string urlDbServer;

    public DbSet<UserEntity> Users { get; set; }

    public AuthContext(IConfiguration configuration, bool isCreate = false)
    {
        urlDbServer = configuration.GetConnectionString(Constants.DbAuth)!;

        if (isCreate)
            Database.EnsureCreated();
    }
```

```
    protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder) =>
         optionsBuilder.UseSqlServer(urlDbServer);

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<UserEntity>()
            .Property(x => x.Id)
            .HasDefaultValueSql("NEWID()");
    }
}
```

Листинг Б.3 – Класс «UserEntity.cs»

```
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;
using Microsoft.EntityFrameworkCore;

namespace AutonomousParkingApp.Authentication.DB.Models;

[Index(nameof(Login)), Index(nameof(Password))]
public class UserEntity
{
    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    public Guid Id { get; set; }

    public string Login { get; set; }

    public string Name { get; set; }

    public string Password { get; set; }

    public string CarNumber { get; set; }

    public string Phone { get; set; }

    public string CardNumber { get; set; }
}
```

Листинг Б.4 – Класс «AuthRepository.cs»

```
using AutonomousParkingApp.Authentication.DB.Contexts;

using AutonomousParkingApp.Authentication.DB.Models;
using AutonomousParkingApp.Authentication.Models.DTO;
using Microsoft.EntityFrameworkCore;

namespace AutonomousParkingApp.Authentication.DB.Repositories;

public class AuthRepository
{
    private readonly AuthContext _authContext;
```

```
    public AuthRepository(AuthContext authContext)
    {
        _authContext = authContext;
    }

    public async Task AddUserAsync(UserEntity user)
    {
        _authContext.Users.Add(user);

        await _authContext.SaveChangesAsync();
    }

    public async Task UpdateUserAsync(UserEntity user)
    {
        _authContext.Users.Update(user);

        await _authContext.SaveChangesAsync();
    }

    public async Task<UserEntity?> GetUserByIdAsync(Guid userId) =>
        await _authContext.Users
            .FirstOrDefaultAsync(x => x.Id == userId);

    public async Task<UserEntity?> GetUserAsync(string login, string
password) =>
        await _authContext.Users
            .FirstOrDefaultAsync(x => x.Login.Equals(login) &&
x.Password.Equals(password));

    public async Task<UserEntity?> GetUserByLoginAsync(string login) =>
        await _authContext.Users
            .FirstOrDefaultAsync(x => x.Login.Equals(login));
}
```

Листинг Б.4 – Класс «Constants.cs»

```
namespace AutonomousParkingApp.Authentication.DB;
public static class Constants
{
    public const string DbAuth = nameof(DbAuth);
}
```

Листинг Б.5 – класс «AuthException.cs»

```
namespace AutonomousParkingApp.Authentication.Exceptions;

public abstract class AuthException : Exception
{
    public readonly int StatusCode;
    public readonly new string Message;
    public readonly new Dictionary<string, object> Data;

    public AuthException(int statusCode, string message, Dictionary<string,
object> data)
    {
        StatusCode = statusCode;
        Message = message;
        Data = data;
    }
}
```

Листинг Б.6 – Класс «UserExistException.cs»

```
namespace AutonomousParkingApp.Authentication.Exceptions
{
    public class UserExistException : AuthException
    {
        private const int _statusCode = 201;
        private const string _message = "Пользователь с таким именем уже
существует";

        public UserExistException(Dictionary<string, object> data)
            : base(_statusCode, _message, data)
        {
        }
    }
}
```

Листинг Б.7 – Класс «UserNotFoundException.cs»

```
namespace AutonomousParkingApp.Authentication.Exceptions;

public class UserNotFoundException : AuthException
{
    private const int _statusCode = 404;
    private const string _message = "Пользователь не найден";

    public UserNotFoundException(Dictionary<string, object> data)
        : base(_statusCode, _message, data)
    {
    }
}
```

Листинг Б.8 – Класс «ConvertExtension.cs»

```
using AutonomousParkingApp.Authentication.DB.Models;
using AutonomousParkingApp.Authentication.Models.DTO;

namespace AutonomousParkingApp.Authentication.Extensions;

public static class ConvertExtension
{
    public static UserDto ToUserDto(this UserEntity userEntity) =>
        new UserDto
        {
            Name = userEntity.Name,
            Login = userEntity.Login,
            Phone = userEntity.Phone,
            CardNumber = userEntity.CardNumber,
            CarNumber = userEntity.CarNumber
        };

    public static UserEntity ToUserEntity(this UserDto userDto) =>
        new UserEntity
        {
            Name = userDto.Name,
            CardNumber = userDto.CardNumber,
            Phone = userDto.Phone,
            CarNumber = userDto.CarNumber,
            Login = userDto.Login,
        };
```

```
    public static UserEntity ToUserEntity(this RegDto authDto) =>
        new UserEntity
        {
            Name = authDto.Name,
            Login = authDto.Login,
            Phone = authDto.Phone,
            CardNumber = authDto.CardNumber,
            CarNumber = authDto.CarNumber,
            Password = authDto.Password
        };
}
```

Листинг Б.9 – Класс «AuthDto.cs»

```
namespace AutonomousParkingApp.Authentication.Models.DTO
{
    public class AuthDto
    {
        public string Login { get; set; }

        public string Password { get; set; }
    }
}
```

Листинг Б.10 – Класс «RegDto.cs»

```
namespace AutonomousParkingApp.Authentication.Models.DTO;

public class RegDto
{
    public string Login { get; set; }

    public string Password { get; set; }

    public string Name { get; set; }

    public string CarNumber { get; set; }

    public string Phone { get; set; }

    public string CardNumber { get; set; }
}
```

Листинг Б.11 – Класс «UserDto.cs»

```
namespace AutonomousParkingApp.Authentication.Models.DTO;

public class UserDto
{
    public string Login { get; set; }

    public string Name { get; set; }

    public string CarNumber { get; set; }

    public string Phone { get; set; }

    public string CardNumber { get; set; }
}
```

Листинг Б.12 – Класс «IAuthService.cs»

```
using AutonomousParkingApp.Authentication.Models.DTO;

namespace AutonomousParkingApp.Authentication.Services.Contracts;

public interface IAuthService
{
    Task AddUserAsync(RegDto user);

    Task UpdateUserAsync(UserDto user);

    Task<UserDto> GetUserByUserIdAsync(Guid userId);

    Task<UserDto> GetUserAsync(AuthDto authDto);

    Task<Guid> GetUserIdByLoginAsync(string login);
}
```

Листинг Б.13 – Класс «AuthService.cs»

```
using AutonomousParkingApp.Authentication.DB.Repositories;
using AutonomousParkingApp.Authentication.Exceptions;
using AutonomousParkingApp.Authentication.Extensions;
using AutonomousParkingApp.Authentication.Models.DTO;
using AutonomousParkingApp.Authentication.Services.Contracts;

namespace AutonomousParkingApp.Authentication.Services;

public class AuthService : IAuthService
{
    public readonly AuthRepository _authRepository;

    public AuthService(AuthRepository authRepository)
    {
        _authRepository = authRepository;
    }

    public async Task AddUserAsync(RegDto user)
    {
        var findUser = await _authRepository.GetUserByLoginAsync(user.Login);

        if(findUser is not null)
            throw new UserExistException(new Dictionary<string, object> { {
nameof(user), user } });

        await _authRepository.AddUserAsync(user.ToUserEntity());
    }

    public async Task UpdateUserAsync(UserDto user)
    {
        var findUser = await _authRepository.GetUserByLoginAsync(user.Login);

        if (findUser is null)
            throw new UserNotFoundException(new Dictionary<string, object> {
{ nameof(user), user } });

        findUser.Name = user.Name;
        findUser.CarNumber = user.CarNumber;
        findUser.Phone = user.Phone;
        findUser.CardNumber = user.CardNumber;
```

```
        await _authRepository.UpdateUserAsync(findUser);
    }

    public async Task<UserDto> GetUserByUserIdAsync(Guid userId)
    {
        var user = await _authRepository.GetUserByIdAsync(userId);

        if (user is null)
            throw new UserNotFoundException(new Dictionary<string, object> {
{ nameof(userId), userId } });

        return user.ToUserDto();
    }

    public async Task<UserDto> GetUserAsync(AuthDto authDto)
    {
            var user = await _authRepository.GetUserAsync(authDto.Login,
authDto.Password);

            if (user is null)
                throw new UserNotFoundException(new Dictionary<string,
object> { { nameof(authDto), authDto } });

            return user.ToUserDto();
    }

    public async Task<Guid> GetUserIdByLoginAsync(string login)
    {
        var user = await _authRepository.GetUserByLoginAsync(login);

        if (user is null)
            throw new UserNotFoundException(new Dictionary<string, object> {
{ nameof(login), login } });

        return user.Id;
    }
}
```

Листинг Б.14 – Класс «Program.cs»

```
using AutonomousParkingApp.Authentication.DB.Contexts;
using AutonomousParkingApp.Authentication.DB.Repositories;
using AutonomousParkingApp.Authentication.Services;
using AutonomousParkingApp.Authentication.Services.Contracts;

var builder = WebApplication.CreateBuilder(args);

var services = builder.Services;

services.AddControllers();

services.AddScoped<AuthContext>();
services.AddScoped<AuthRepository>();
services.AddScoped<IAuthService, AuthService>();

var app = builder.Build();

builder.Configuration.AddJsonFile("appsettings.json");

app.MapControllers();
```

```
app.Run();
```

Листинг Б.15 – Класс «IAuthServices.cs»

```
using AutonomousParkingApp.Client.Models.DTO;
using System;
using System.Threading.Tasks;

namespace AutonomousParkingApp.Client.Services.Contracts
{
    public interface IAuthService
    {
        Task<bool> AddUserAsync(RegDto user);

        Task<bool> UpdateUserAsync(UserDto user);

        Task<UserDto> GetUserByUserIdAsync(Guid userId);

        Task<UserDto> GetUserAsync(AuthDto authDto);

        Task<Guid?> GetUserIdByLoginAsync(string login);
    }
}
```

Листинг Б.16 – Класс «INumberRecognitionServices.cs»

```
using AutonomousParkingApp.Client.Models.DTO;
using System.Threading.Tasks;

namespace AutonomousParkingApp.Client.Services.Contracts
{
    public interface INumberRecognitionService
    {
        Task<string> GetCarNumberAsync(ImageDto image);
    }
}
```

Листинг Б.17 – Класс «IStorageServices.cs»

```
using AutonomousParkingApp.Client.Models.DTO;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace AutonomousParkingApp.Client.Services.Contracts
{
    public interface IStorageService
    {
        Task<ICollection<ParkingDto>> GetAllParkingAsync();

        Task<PriceDto> GetPriceByUserIdAsync(Guid userId);

        Task<bool> AddReservationAsync(ReservationDto reservation);

        Task<bool> DeleteReservationAsync(Guid userId);
    }
}
```

Листинг Б.18 – Класс «ServiceException.cs»

```
using System.Collections.Generic;
```

```
namespace AutonomousParkingApp.Client.Services.Exceptions
{
    public class ServiceException
    {
        public string Detail { get; set; }

        public string Instance { get; set; }

        public int? StatusCode { get; set; }

        public string Title { get; set; }

        public string Type { get; set; }
    }
}
```

Листинг Б.19 – Класс «AuthService.cs»

```
using AutonomousParkingApp.Client.Models.DTO;
using AutonomousParkingApp.Client.Services.Contracts;
using AutonomousParkingApp.Client.Services.Exceptions;
using AutonomousParkingApp.Client.Services.Utils;
using System;
using System.Net;
using System.Net.Http;
using System.Net.Http.Json;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace AutonomousParkingApp.Client.Services
{
    public class AuthService : IAuthService
    {
        private readonly HttpClient _authHttpClient;

        public AuthService()
        {
            _authHttpClient = new HttpClient();
            _authHttpClient.BaseAddress = new
Uri(ServicesConstants.AuthService);
        }

        public async Task<bool> AddUserAsync(RegDto user)
        {
            var content = ServicesUtils.ToJsonContent(user);

            var result = await _authHttpClient.PostAsync("add-user",
content);

            if (result.StatusCode != HttpStatusCode.NoContent)
            {
                var exception = await
ServicesUtils.ToDtoAsync<ServiceException>(result);

                ServicesUtils.ShowMessageBoxError(exception);

                return false;
            }

            return true;
        }

        public async Task<bool> UpdateUserAsync(UserDto user)
        {
```

```
            var content = ServicesUtils.ToJsonContent(user);

            var result = await _authHttpClient.PutAsync("update-user",
content);

            if (result.StatusCode != HttpStatusCode.NoContent)
            {
                var exception = await
ServicesUtils.ToDtoAsync<ServiceException>(result);

                ServicesUtils.ShowMessageBoxError(exception);

                return false;
            }

            return true;
        }

        public async Task<UserDto> GetUserByUserIdAsync(Guid userId)
        {
            var result = await _authHttpClient.GetAsync($"get-
user/{userId}");

            if (result.StatusCode != HttpStatusCode.OK)
            {
                var exception = await
ServicesUtils.ToDtoAsync<ServiceException>(result);

                ServicesUtils.ShowMessageBoxError(exception);

                return null;
            }

            return await ServicesUtils.ToDtoAsync<UserDto>(result);
        }

        public async Task<UserDto> GetUserAsync(AuthDto authDto)
        {
            var content = ServicesUtils.ToJsonContent(authDto);

            var result = await _authHttpClient.PostAsync("get-user-by-login-
password", content);

            if (result.StatusCode != HttpStatusCode.OK)
            {
                var exception = await
ServicesUtils.ToDtoAsync<ServiceException>(result);

                ServicesUtils.ShowMessageBoxError(exception);

                return null;
            }

            return await ServicesUtils.ToDtoAsync<UserDto>(result);
        }

        public async Task<Guid?> GetUserIdByLoginAsync(string login)
        {
            var result = await _authHttpClient.GetAsync($"get-userId-by-
login/{login}");

            if (result.StatusCode != HttpStatusCode.OK)
            {
                var exception = await
ServicesUtils.ToDtoAsync<ServiceException>(result);
```

```
                ServicesUtils.ShowMessageBoxError(exception);

                return null;
            }

            return await ServicesUtils.ToDtoAsync<Guid>(result);
        }
    }
}
```

Листинг Б.20 – Класс «NumberRecognitionService.cs»

```
 using AutonomousParkingApp.Client.Services.Contracts;
using System.Net.Http;
using System;
using System.Threading.Tasks;
using AutonomousParkingApp.Client.Models.DTO;
using AutonomousParkingApp.Client.Services.Utils;
using System.Windows.Forms;

namespace AutonomousParkingApp.Client.Services
{
    public class NumberRecognitionService : INumberRecognitionService
    {
        private readonly HttpClient _numberRecognitionHttpClient;

        public NumberRecognitionService()
        {
            _numberRecognitionHttpClient = new HttpClient();
        }

        public async Task<string> GetCarNumberAsync(ImageDto image)
        {
        const int countNumber = 6;

            var result = await _numberRecognitionHttpClient.SendAsync(new
HttpRequestMessage
            {
               Method = HttpMethod.Get,
               RequestUri = new Uri(ServicesConstants.PyNumberRecognition),
               Content = new StringContent(ByteArrayToJson(image.Data),
Encoding.UTF8, "application/json")
            });

            if(result.Content == null)
            {
                var messageException = "Номер не распознан, сделайте,
пожалуйста, более чёткое изображение";

                MessageBox.Show(messageException);
            }

            return (await result.Content.ReadAsStringAsync()).Substring(0,
countNumber);

        }

    private string ByteArrayToJson(byte[,] image)
    {
        var intArray = new int[image.GetLength(0), image.GetLength(1)];
            for (int i = 0; i < image.GetLength(0); i++)
            {
              for (int j = 0; j < image.GetLength(1); j++)
              {
```

```
                intArray[i, j] = image[i, j];
              }
            }
            return JsonConvert.SerializeObject(intArray);
    }
    }
}
```

Листинг Б.21 – Класс «StorageService.cs»

```
using AutonomousParkingApp.Client.Services.Contracts;
using System.Net.Http;
using System;
using System.Threading.Tasks;
using System.Collections.Generic;
using AutonomousParkingApp.Client.Models.DTO;
using AutonomousParkingApp.Client.Services.Exceptions;
using AutonomousParkingApp.Client.Services.Utils;
using System.Net;
using System.Windows.Forms;

namespace AutonomousParkingApp.Client.Services
{
    public class StorageService : IStorageService
    {
        private readonly HttpClient _storageHttpClient;

        public StorageService()
        {
            _storageHttpClient = new HttpClient();
            _storageHttpClient.BaseAddress = new
Uri(ServicesConstants.StorageCarNumbers);
        }

        public async Task<ICollection<ParkingDto>> GetAllParkingAsync()
        {
            var result = await _storageHttpClient.GetAsync("all-parking");

            if (result.StatusCode != HttpStatusCode.OK)
            {
                var exception = await
ServicesUtils.ToDtoAsync<ServiceException>(result);

                ServicesUtils.ShowMessageBoxError(exception);

                return null;
            }

            return await
ServicesUtils.ToDtoAsync<ICollection<ParkingDto>>(result);
        }

        public async Task<PriceDto> GetPriceByUserIdAsync(Guid userId)
        {
            var result = await
_storageHttpClient.GetAsync($"price/{userId}");

            if (result.StatusCode != HttpStatusCode.OK)
            {
                var exception = await
ServicesUtils.ToDtoAsync<ServiceException>(result);

                ServicesUtils.ShowMessageBoxError(exception);
```

```
                return null;
            }

            return await ServicesUtils.ToDtoAsync<PriceDto>(result);
        }

        public async Task<bool> AddReservationAsync(ReservationDto
reservation)
        {
            var content = ServicesUtils.ToJsonContent(reservation);

            var result = await _storageHttpClient.PostAsync("add", content);

            if (result.StatusCode != HttpStatusCode.NoContent)
            {
                var exception = await
ServicesUtils.ToDtoAsync<ServiceException>(result);

                ServicesUtils.ShowMessageBoxError(exception);

                return false;
            }

            return true;
        }

        public async Task<bool> DeleteReservationAsync(Guid userId)
        {
            var result = await
_storageHttpClient.DeleteAsync($"delete/{userId}");

            if (result.StatusCode != HttpStatusCode.NoContent)
            {
                var exception = await
ServicesUtils.ToDtoAsync<ServiceException>(result);

                ServicesUtils.ShowMessageBoxError(exception);

                return false;
            }

            return true;
        }
    }
}
```

Листинг Б.22 – Класс «IStorageCarNumbersService.cs»

```
using AutonomousParkingApp.StorageCarNumbers.Models.DTO;

namespace AutonomousParkingApp.StorageCarNumbers.Services.Contracts;

public interface IStorageCarNumbersService
{
    Task<ICollection<ParkingDto>?> GetAllParkingAsync();

    Task AddReservationAsync(ReservationDto reservationDto);

    Task DeleteReservationAsync(Guid userId);

    Task<PriceDto> GetPriceByUserIdAsync(Guid userId);
}
```

Листинг Б.23 – Класс «StorageCarNumbersService.cs»

```
using AutonomousParkingApp.StorageCarNumbers.DB.Repositories;
using AutonomousParkingApp.StorageCarNumbers.Exceptions;
using AutonomousParkingApp.StorageCarNumbers.Extensions;
using AutonomousParkingApp.StorageCarNumbers.Models.DTO;
using AutonomousParkingApp.StorageCarNumbers.Services.Contracts;
using Microsoft.AspNetCore.Components.Web;

namespace AutonomousParkingApp.StorageCarNumbers.Services;

public class StorageCarNumbersService : IStorageCarNumbersService
{
    private readonly StorageCarNumbersRepository
_storageCarNumbersRepository;

    public StorageCarNumbersService(StorageCarNumbersRepository
storageCarNumbersRepository)
    {
        _storageCarNumbersRepository = storageCarNumbersRepository;
    }

    public async Task<ICollection<ParkingDto>?> GetAllParkingAsync()
    {
        var parkings = await
_storageCarNumbersRepository.GetAllParkingAsync()
            ?? throw new ParkingNotFoundException(new Dictionary<string,
object> { { nameof(ParkingDto), string.Empty } });

        var result = new List<ParkingDto>();

        foreach (var parking in parkings)
            result.Add(parking.ToParkingDto());

        return result;
    }

    public async Task AddReservationAsync(ReservationDto reservationDto)
    {
        var parking = await
_storageCarNumbersRepository.GetParkingByParkingIdAsync(reservationDto.Parkin
gId)
            ?? throw new ParkingNotFoundException(new Dictionary<string,
object> { { nameof(reservationDto), reservationDto } });

        if (parking.NumFreeSpaces <= 0)
            throw new NumFreeSpecesEquelsNullException(new Dictionary<string,
object> { { nameof(reservationDto), reservationDto } });

        parking.NumFreeSpaces -= 1;

        await _storageCarNumbersRepository.UpdateParkingAsync(parking);

        await
_storageCarNumbersRepository.AddReservationAsync(reservationDto.ToReservation
Entity());
    }

    public async Task DeleteReservationAsync(Guid userId)
    {
        var reservation = await
_storageCarNumbersRepository.GetReservationByUserIdAsync(userId)
            ?? throw new ReservationNotFoundException(new Dictionary<string,
object> { { nameof(userId), userId } });

        await
_storageCarNumbersRepository.DeleteReservationAsync(reservation);
```

```
        var parking = await
_storageCarNumbersRepository.GetParkingByParkingIdAsync(reservation.ParkingId
);

        parking!.NumFreeSpaces += 1;

        await _storageCarNumbersRepository.UpdateParkingAsync(parking);
    }

    public async Task<PriceDto> GetPriceByUserIdAsync(Guid userId)
    {
        var reservation = await
_storageCarNumbersRepository.GetReservationByUserIdAsync(userId)
            ?? throw new ReservationNotFoundException(new Dictionary<string,
object> { { nameof(userId), userId } });

        var nowDate = DateTime.Now;
        var checkInTime = reservation.CheckInTime;

        var countDay = (nowDate.Day < checkInTime.Day)
            ? (30 - checkInTime.Day) + nowDate.Day
            : nowDate.Day - checkInTime.Day;

        var countHour = ((nowDate.Hour < checkInTime.Hour)
            ? (24 - checkInTime.Hour) + nowDate.Hour
            : nowDate.Hour - checkInTime.Hour)
            + (countDay * 24 * 60);

        var countMinute = (nowDate.Minute < checkInTime.Minute)
            ? (60 - checkInTime.Minute) + nowDate.Minute
            : nowDate.Minute - checkInTime.Minute;

        var pricePerHour = await
_storageCarNumbersRepository.GetPriceByParkingIdAsync(reservation.ParkingId);
        var price = Math.Round(pricePerHour * Convert.ToDecimal(countHour +
(countMinute/60.0)), 2);

        return new PriceDto
        {
            NumHour = countHour,
            NumMinute = countMinute,
            Price = price,
        };
    }
}
```

Листинг Б.24 – Класс «Program.cs»

```
using AutonomousParkingApp.StorageCarNumbers.DB.Contexts;
using AutonomousParkingApp.StorageCarNumbers.DB.Repositories;
using AutonomousParkingApp.StorageCarNumbers.Services;
using AutonomousParkingApp.StorageCarNumbers.Services.Contracts;

var builder = WebApplication.CreateBuilder(args);

var services = builder.Services;

services.AddControllers();

services.AddScoped<StorageCarNumbersContext>();
services.AddScoped<StorageCarNumbersRepository>();
services.AddScoped<IStorageCarNumbersService, StorageCarNumbersService>();

var app = builder.Build();
```

```
builder.Configuration.AddJsonFile("appsettings.json");

app.UseAuthorization();

app.MapControllers();

app.Run();
```

Листинг Б.25 – Класс «StorageCarNumbersService.cs»

```
using AutonomousParkingApp.StorageCarNumbers.DB.Repositories;
using AutonomousParkingApp.StorageCarNumbers.Exceptions;
using AutonomousParkingApp.StorageCarNumbers.Extensions;
using AutonomousParkingApp.StorageCarNumbers.Models.DTO;
using AutonomousParkingApp.StorageCarNumbers.Services.Contracts;
using Microsoft.AspNetCore.Components.Web;

namespace AutonomousParkingApp.StorageCarNumbers.Services;

public class StorageCarNumbersService : IStorageCarNumbersService
{
    private readonly StorageCarNumbersRepository
_storageCarNumbersRepository;

    public StorageCarNumbersService(StorageCarNumbersRepository
storageCarNumbersRepository)
    {
        _storageCarNumbersRepository = storageCarNumbersRepository;
    }

    public async Task<ICollection<ParkingDto>?> GetAllParkingAsync()
    {
        var parkings = await
_storageCarNumbersRepository.GetAllParkingAsync()
            ?? throw new ParkingNotFoundException(new Dictionary<string,
object> { { nameof(ParkingDto), string.Empty } });

        var result = new List<ParkingDto>();

        foreach (var parking in parkings)
            result.Add(parking.ToParkingDto());

        return result;
    }

    public async Task AddReservationAsync(ReservationDto reservationDto)
    {
        var parking = await
_storageCarNumbersRepository.GetParkingByParkingIdAsync(reservationDto.Parkin
gId)
            ?? throw new ParkingNotFoundException(new Dictionary<string,
object> { { nameof(reservationDto), reservationDto } });

        if (parking.NumFreeSpaces <= 0)
            throw new NumFreeSpecesEquelsNullException(new Dictionary<string,
object> { { nameof(reservationDto), reservationDto } });

        parking.NumFreeSpaces -= 1;

        await _storageCarNumbersRepository.UpdateParkingAsync(parking);

        await
_storageCarNumbersRepository.AddReservationAsync(reservationDto.ToReservation
Entity());
```

```
    }

    public async Task DeleteReservationAsync(Guid userId)
    {
        var reservation = await
_storageCarNumbersRepository.GetReservationByUserIdAsync(userId)
            ?? throw new ReservationNotFoundException(new Dictionary<string,
object> { { nameof(userId), userId } });

        await
_storageCarNumbersRepository.DeleteReservationAsync(reservation);

        var parking = await
_storageCarNumbersRepository.GetParkingByParkingIdAsync(reservation.ParkingId
);

        parking!.NumFreeSpaces += 1;

        await _storageCarNumbersRepository.UpdateParkingAsync(parking);
    }

    public async Task<PriceDto> GetPriceByUserIdAsync(Guid userId)
    {
        var reservation = await
_storageCarNumbersRepository.GetReservationByUserIdAsync(userId)
            ?? throw new ReservationNotFoundException(new Dictionary<string,
object> { { nameof(userId), userId } });

        var nowDate = DateTime.Now;
        var checkInTime = reservation.CheckInTime;

        var countDay = (nowDate.Day < checkInTime.Day)
            ? (30 - checkInTime.Day) + nowDate.Day
            : nowDate.Day - checkInTime.Day;

        var countHour = ((nowDate.Hour < checkInTime.Hour)
            ? (24 - checkInTime.Hour) + nowDate.Hour
            : nowDate.Hour - checkInTime.Hour)
            + (countDay * 24 * 60);

        var countMinute = (nowDate.Minute < checkInTime.Minute)
            ? (60 - checkInTime.Minute) + nowDate.Minute
            : nowDate.Minute - checkInTime.Minute;

        var pricePerHour = await
_storageCarNumbersRepository.GetPriceByParkingIdAsync(reservation.ParkingId);
        var price = Math.Round(pricePerHour * Convert.ToDecimal(countHour +
(countMinute/60.0)), 2);

        return new PriceDto
        {
            NumHour = countHour,
            NumMinute = countMinute,
            Price = price,
        };
    }
}
```

Приложение В

(рекомендуемое)

Листинг распознавания номера Python

Листинг В.1 – Класс «PyNumberRecording.py»

```
import cv2
from PyIN import PyIN as IN
from flask import Flask, jsonify, request
import json

app = Flask(__name__)

rec = IN()

@app.route('/car-number', methods=['POST'])
def get_number_car():

    data = request.get_json()
    img = json.loads(data)
    img = np.asarray(img, dtype=np.uint8)

    image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    number = rec.get_number_car(image)

    return number

if __name__ == '__main__':
    app.run(port=5002)
```

Листинг В.2 – Класс «NNSegmentation.py»

```
import torch
import pathlib

class NNSegmentation:
    def __init__(self):
        pathlib.PosixPath = pathlib.WindowsPath
        self.model = torch.hub.load('yolov5', 'custom', path='last.pt',
source='local')

    def get_labels(self, img):
        predictes = self.model([img])

        predictes = predictes.xywhn[0]

        correct_index_predictes = self.__max_percent(predictes)
        correct_predict = predictes[correct_index_predictes]

        labels = []
        for it in range(0, len(correct_predict) - 2):
            labels.append(correct_predict[it].numpy())

        return labels

    def __max_percent(self, predictes):
        correct_index_predictes = 0
        pred_percent = -1

        it = 0
        for predict in predictes:
            current_percent = predict[len(predict) - 2]

            if current_percent > pred_percent:
                pred_percent = current_percent
                correct_index_predictes = it
```

```
            ++it
        return correct_index_predictes
```

Листинг В.3 – Класс «PyIN.py»

```
from NNSegmentation import NNSegmentation
import pytesseract
import cv2

class PyIN:
    def __init__(self):
        self.nn = NNSegmentation()
        pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-
OCR\tesseract.exe'

    def get_number_car(self, img):

        yolo_coords = self.nn.get_labels(img)
        bbox_coords = self.__yolo_to_bbox(yolo_coords, len(img[0]), len(img))

        cropped_image = self.__crop_image(img, bbox_coords)

        thresh = 125
        cropped_image_gray = cv2.threshold(cropped_image, thresh, 255,
cv2.THRESH_BINARY)[1]

        return pytesseract.image_to_string(
            cropped_image_gray,
            config='--oem 3 --psm 10 -c
tessedit_char_whitelist=ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789')

    def __yolo_to_bbox(self, yolo_coords, image_width, image_height):
        """
        Преобразует координаты ограничивающей рамки из формата YOLO в формат
ограничивающих прямоугольников (bounding box).

        Args:
            yolo_coords (tuple): Кортеж с координатами ограничивающей рамки в
формате YOLO (x_center, y_center, width, height).
            image_width (int): Ширина изображения.
            image_height (int): Высота изображения.

        Returns:
            tuple: Кортеж с координатами ограничивающей рамки в формате
(x_min, y_min, x_max, y_max).
        """
        x_center, y_center, width, height = yolo_coords
        x_min = (x_center - width / 2) * image_width
        y_min = (y_center - height / 2) * image_height
        x_max = x_min + width * image_width
        y_max = y_min + height * image_height
        return (x_min, y_min, x_max, y_max)

    def __crop_image(self, image, bbox_coords):
        """
        Вырезает область изображения согласно ограничивающему прямоугольнику.

        Args:
            image (numpy.ndarray): Исходное изображение.
            bbox_coords (tuple): Кортеж с координатами ограничивающей рамки в
формате (x_min, y_min, x_max, y_max).

        Returns:
            numpy.ndarray: Вырезанная область изображения.
```

```
    """
    x_min, y_min, x_max, y_max = bbox_coords
    cropped_image = image[int(y_min):int(y_max), int(x_min):int(x_max)]
    return cropped_image
```

Приложение Г
(рекомендуемое)
Спецификация сущностей

Таблица Г.1 – Спецификация сущности «Пользователь»

Идентификатор	Содержимое	Тип Данных	Размерность	Область значений	Значение NULL	Роль	Пример
ID	Код пользователя	Целый	5	1-10000	нет	PK	777
Login	Логин пользователя	Символьный	15	«A-z», «0-9»	нет	-	Vor55
Name	Имя пользователя	Символьный	50	«A-z», «0-9»	нет	-	Vornakov55
Password	Пароль пользователя	Символьный	50	«A-z», «0-9»	нет	-	4739dndjc
CarNumber	Номер авто	Символьный	9	«А», «В», «Е», «К», «М», «Н», «О», «Р», «С», «Т», «У», «Х», “0-9”	нет	-	М716ВО13
Phone	Телефон	Целый	11	«0-9»	да	-	22234
CardNumber	Номер карты	Целый	16	«0-9»	да	-	2234325523

Таблица Г.2 – Спецификация сущности «Парковка»

Идентификатор	Содержимое	Тип Данных	Размерность	Область значений	Значение NULL	Роль	Пример
ID	Код парковки	Целый	5	1-10000	нет	PK	777
Adress	Адрес	Символьный	100	«A-z», «А-я» «0-9»	нет	-	Маршала Крылова 2а
Price	Цена	Целый	7	1-1000000	да	-	5250
NumbFreeSpaces	Кол-во свободных мест	Целый	3	1-100	да	-	33

Таблица Г.3 – Спецификация сущности «Бронирование»

Идентификатор	Содержимое	Тип Данных	Размерность	Область значений	Значение NULL	Роль	Пример
ID	Код Бронирования	Целый	5	1-10000	нет	PK	777
ParkingID	ID парковки	Целый	5	1-10000	нет	FK	777
UserID	ID пользователя	Целый	5	1-10000	нет	FK	777
CheckingTime	Время заезда	ДатаTime	19	2024-01-01 00:00:00	да	-	2024-01-01 00:00:00

Приложение Д

(рекомендуемое)

Описание методов клиентского приложения

Таблица Д.1 – Методы класса «IAuthService»

Наименование класса	Фунциональность класса
AddUserAsync(RegDto user)	Метод принимает объект RegDto, который обозначает нового пользователя, и добавляет его в БД
UpdateUserAsync	Метод принимает объект UserDto, пользователя, и обновляет его данные в БД
GetUserByUserIdAsync	Метод принимает userId и возвращает объект UserDto, представляющий пользователя с ID
GetUserAsync	Метод принимает объект AuthDto, который содержит данные ID пользователя и возвращает объект UserDto, пользователя
GetUserIdByLoginAsync	Метод принимает логин пользователя и возвращает userId пользователя

Таблица Д.2 – Методы класса «INumberRecognitionService»

Наименование класса	Фунциональность класса
GetCarNumberAsync	Метод принимает image и возвращает объект ImageDto конкретного пользователя

Таблица Д.3 – Методы класса «IStorageService»

Наименование класса	Фунциональность класса
GetAllParkingAsync	Асинхронно отправляет GET запрос на сервер для получения информации обо всех парковочных местах
GetPriceByUserIdAsync	Асинхронно отправляет GET запрос на сервер для получения информации о цене для определенного пользователя
AddReservationAsync	Асинхронно отправляет POST запрос на сервер для добавления новой резервации

Продолжение таблицы Д.3

Наименование класса	Фунциональность класса
DeleteReservationAsync	Асинхронно отправляет DELETE запрос на сервер для удаления резервации по ID

Таблица Д.4 – Методы класса «AuthService»

Наименование метода	Фунциональность методы
AddUserAsync	Асинхронно добавляет нового пользователя
UpdateUserAsync	Асинхронно обновляет информацию о пользователе
GetUserByUserIdAsync	Асинхронно получает пользователя по идентификатору
GetUserAsync	Асинхронно получает пользователя по логину и паролю
GetUserIdByLoginAsync	Асинхронно получает идентификатор пользователя по логину

Таблица Д.5 – Методы класса «StorageService»

Наименование класса	Фунциональность класса
GetAllParkingAsync	Асинхронно отправляет GET запрос на сервер для получения информации обо всех парковочных местах
GetPriceByUserIdAsync	Асинхронно отправляет GET запрос на сервер для получения информации о цене для определенного пользователя
AddReservationAsync	Асинхронно отправляет POST запрос на сервер для добавления новой резервации
DeleteReservationAsync	Асинхронно отправляет DELETE запрос на сервер для удаления резервации по идентификатору пользователя

Таблица Д.6 – Методы класса «NumberRecognitionService»

Наименование класса	Фунциональность класса
async Task GetCarNumberAsync	Принимает объект ImageDto, отправляет запрос на сервер для распознавания номера, проверяет результат и возвращает номер автомобиля
private string ByteArrayToJson	Преобразует массив изображения в массив чисел, сериализует массив в json и вовзращает строку json

Приложение Е
(рекомендуемое)
Результаты тестирования

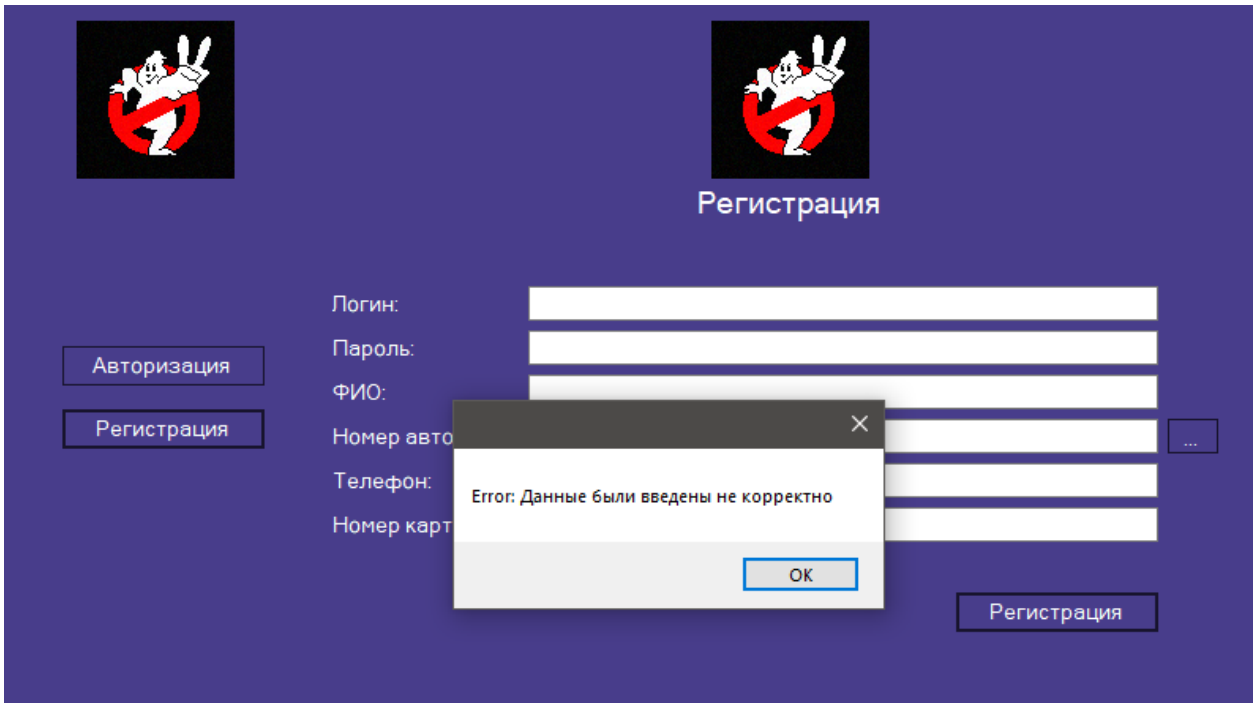


Рисунок Е.1 – Результат теста №1 таблицы решений

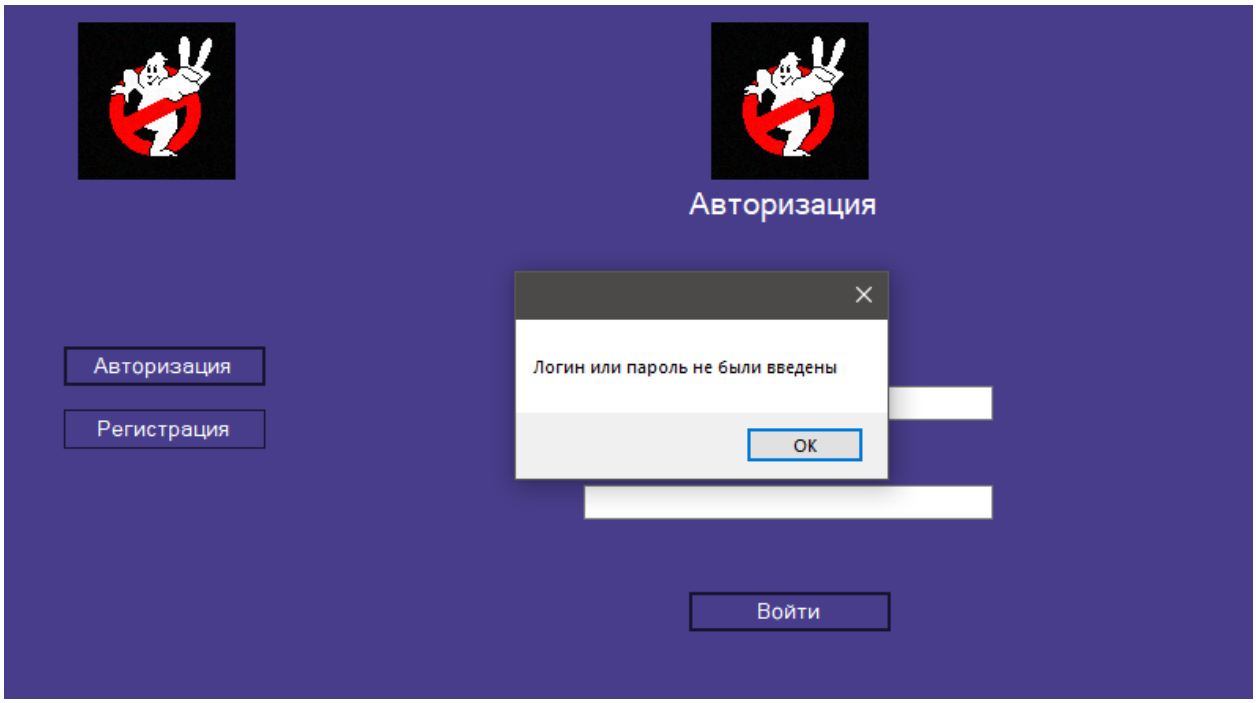


Рисунок Е.2 – Результат теста №2 таблицы решений

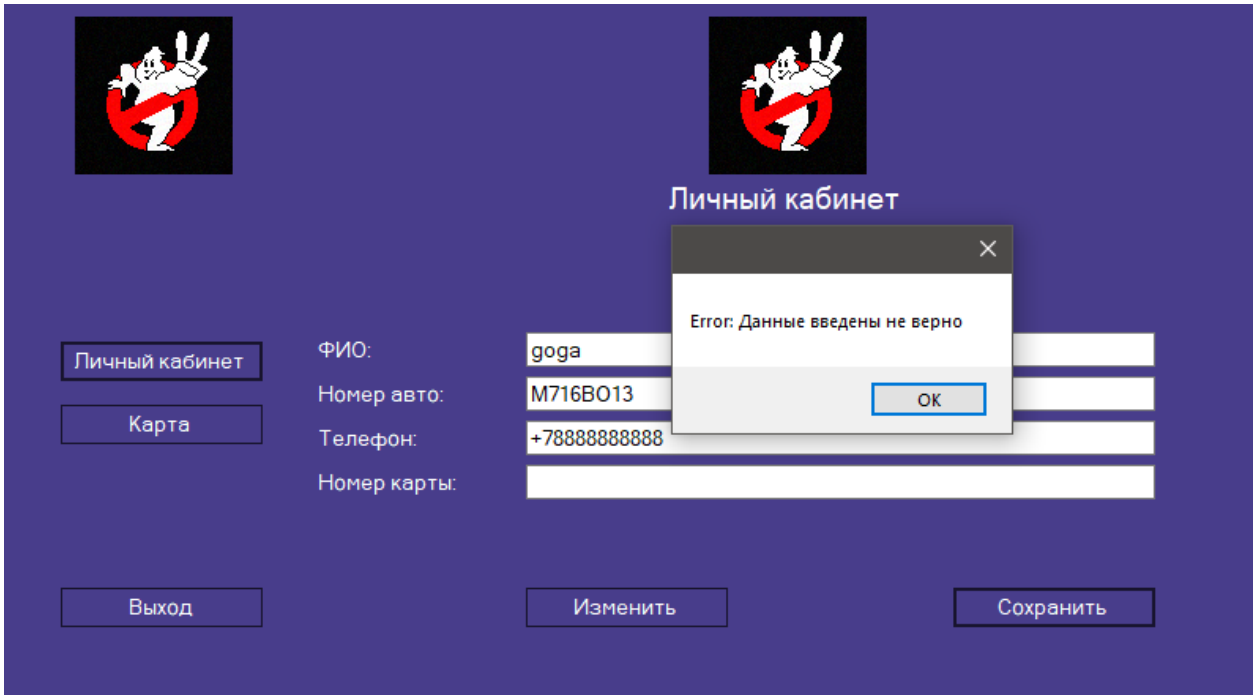


Рисунок Е.3 – Результат теста №3 таблицы решений

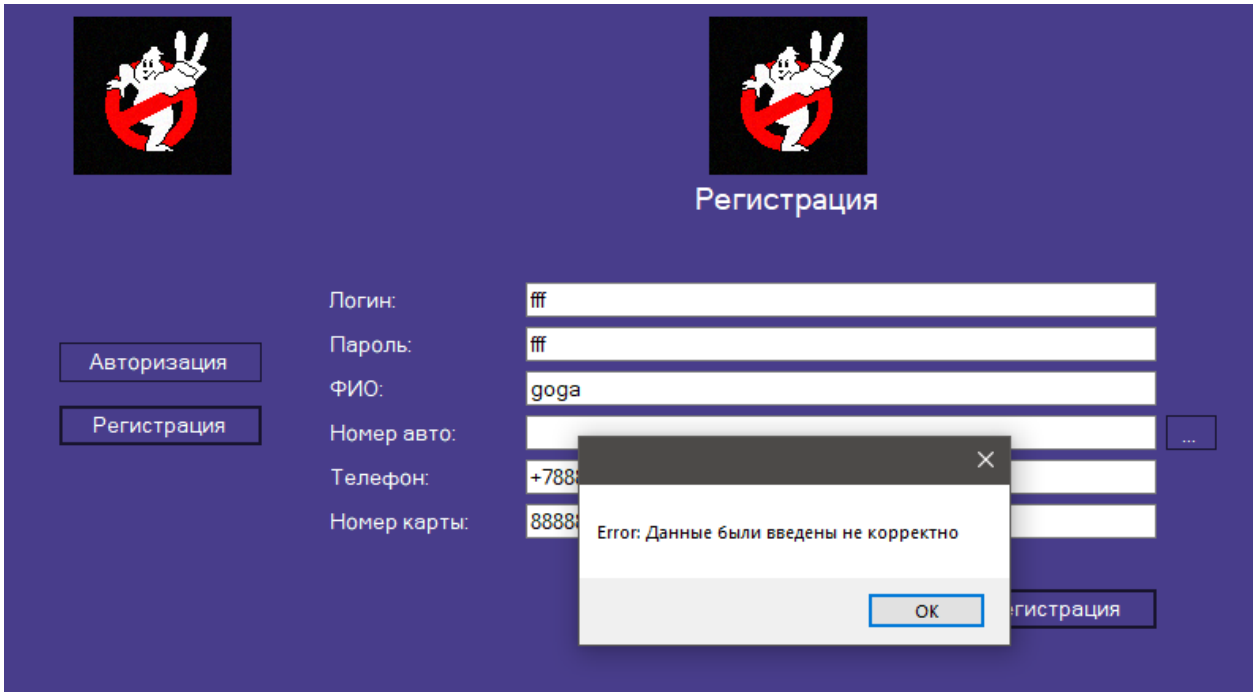


Рисунок Е.4 – Результат теста №4 таблицы решений

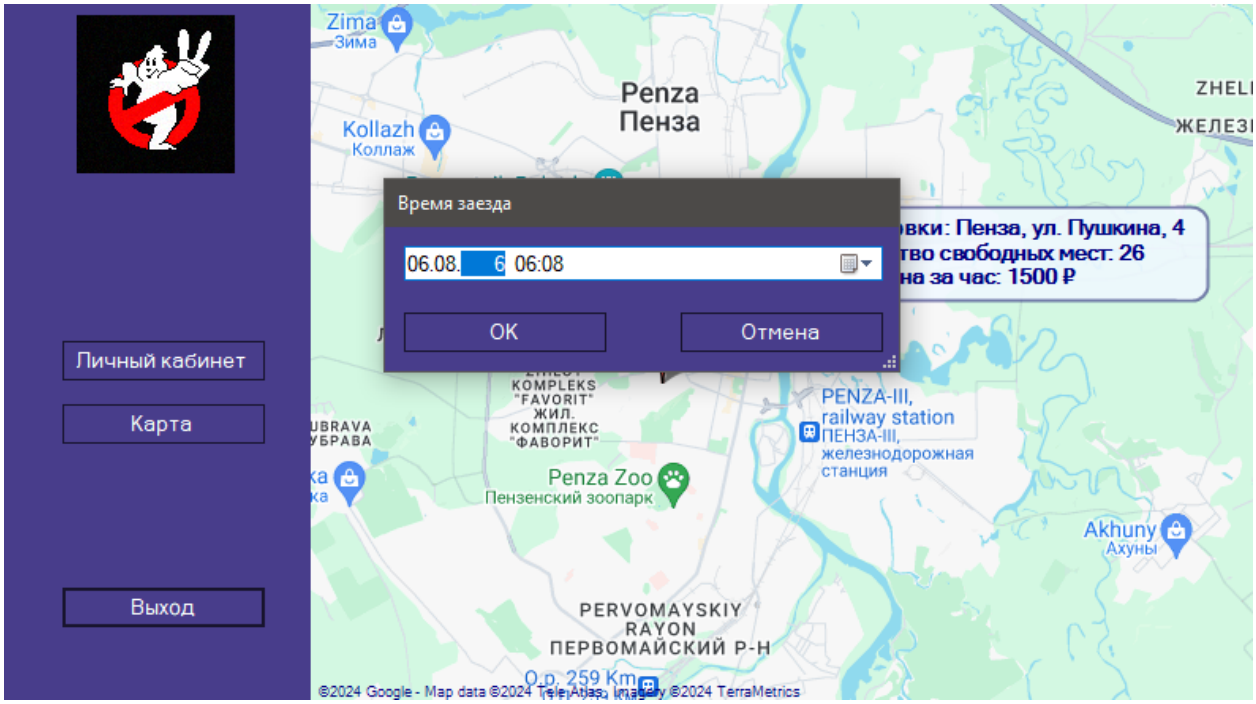


Рисунок Е.5 – Результат теста №5 таблицы решений

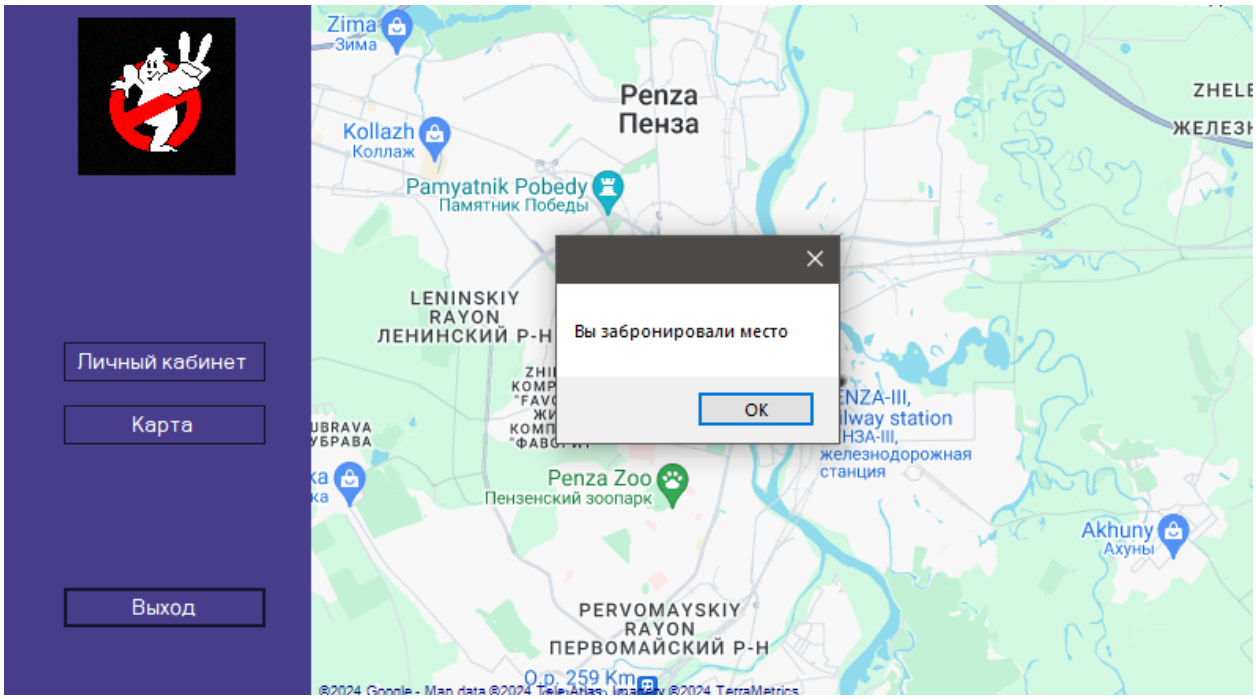


Рисунок Е.6 – Результат теста №6 таблицы решений

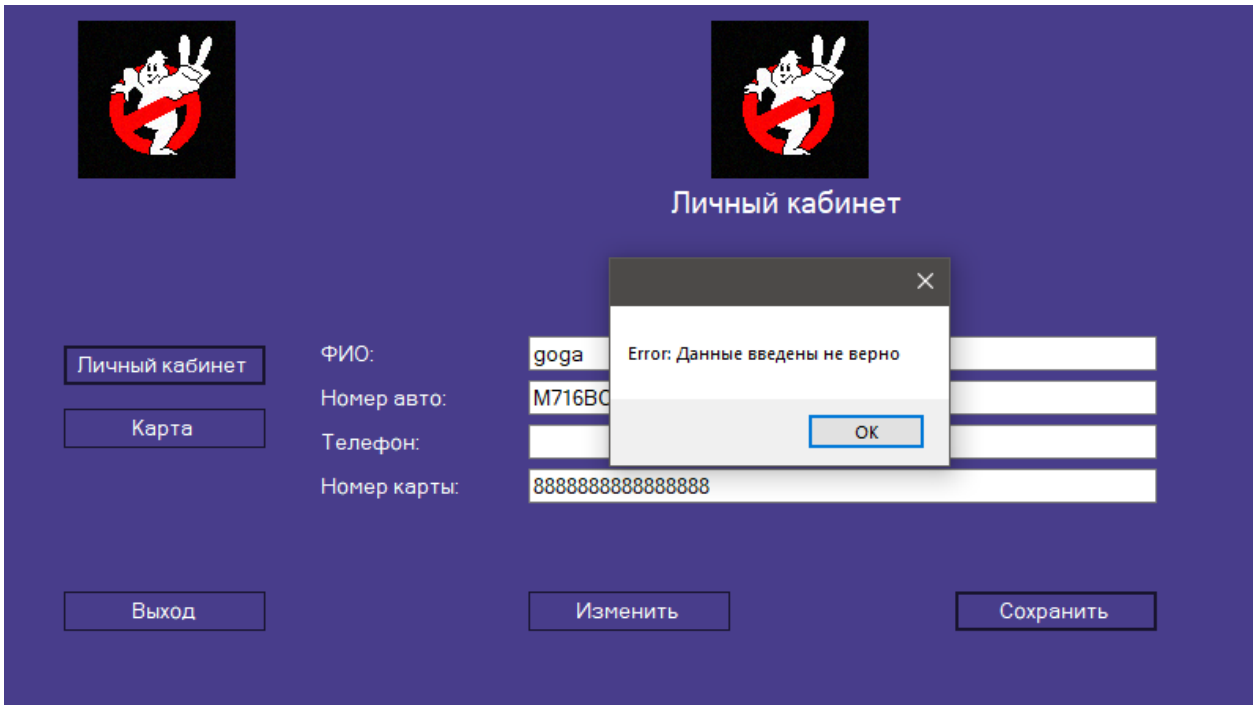


Рисунок Е.7 – Результат теста №7 таблицы решений

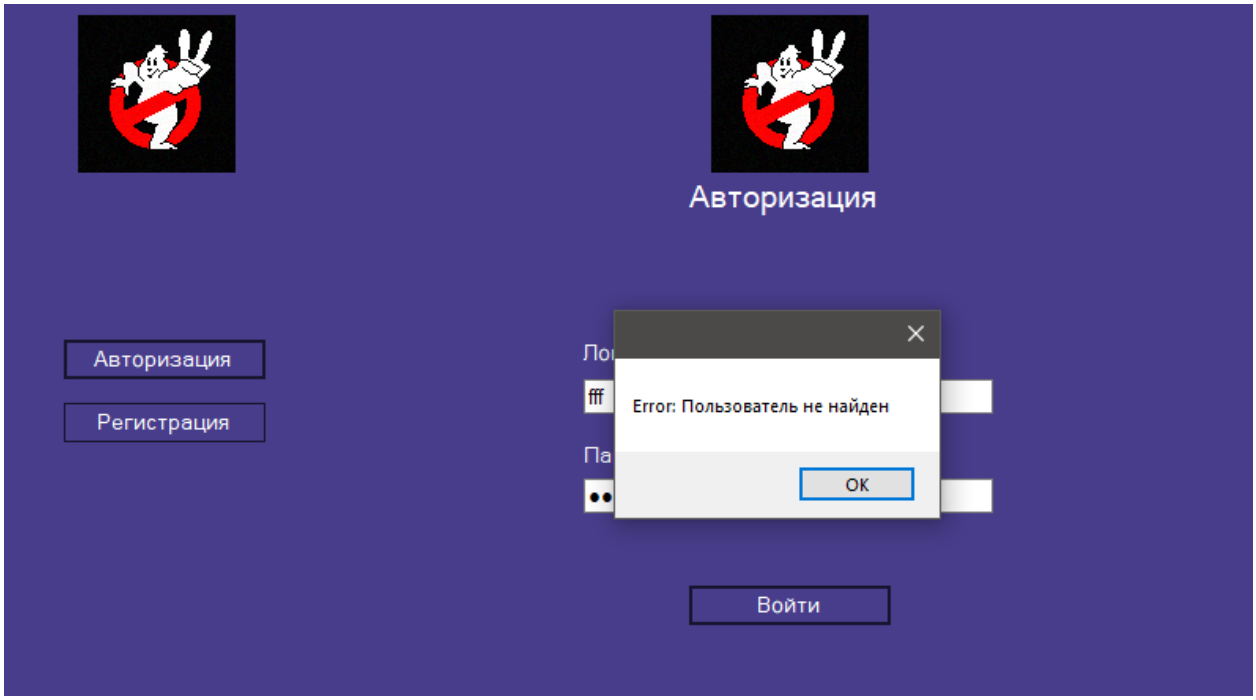


Рисунок Е.8 – Результат теста №8 таблицы решений

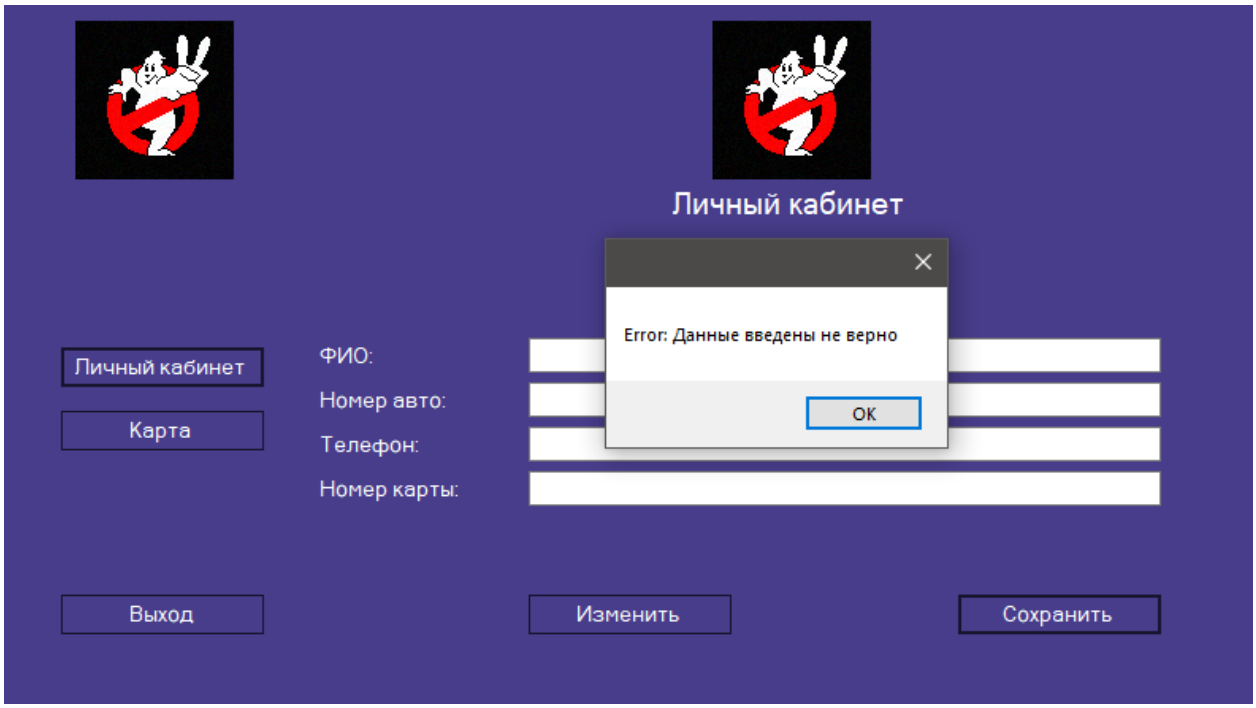


Рисунок Е.9 – Результат теста №9 таблицы решений