**"Интеллектуальная система управления доступом на парковку с распознаванием автомобильных номеров"**

**Цель**: разработать систему, которая использует технологии компьютерного зрения и распознавания номерных знаков для автоматического открытия шлагбаума при въезде и выезде. Система может включать в себя базу данных разрешенных автомобилей, чтобы обеспечивать доступ только зарегистрированным жителям.

Программа: Разработчик

Специализация: Data Engineer

ФИО: Удодов Константин Валерьевич

Город Воронеж

Год 2024

**Содержание**

1. **Введение**

* 1.1 Актуальность темы
* 1.2 Цель и задачи исследования
* 1.3 Объект и предмет исследования
* 1.4 Методология и инструменты исследования
* 1.5 Структура работы
* Дополнительно: Решаемая проблема
* Состав команды

1. **Обзор литературы и технологий**

* 2.1 Современные методы распознавания автомобильных номеров
  + 2.1.1 История и развитие ALPR-систем
  + 2.1.2 Основные принципы работы ALPR
  + 2.1.3 Преимущества и ограничения современных методов
* 2.2 Технологии компьютерного зрения и машинного обучения
  + 2.2.1 Инструменты и библиотеки для обработки изображений
  + 2.2.2 Алгоритмы глубокого обучения для детекции и распознавания (CNN, YOLO, SSD)
  + 2.2.3 Оптическое распознавание символов (OCR) и его применение
* 2.3 Существующие системы управления доступом на парковку
  + 2.3.1 Коммерческие решения и их особенности
  + 2.3.2 Открытые платформы и их возможности
  + 2.3.3 Сравнительный анализ систем по критериям точности, скорости и стоимости
* 2.4 Анализ требований к системе
  + 2.4.1 Функциональные требования
  + 2.4.2 Нефункциональные требования
  + 2.4.3 Вызовы и ограничения в реализации
  + Подведение итогов обзора

1. **Проектирование системы**

* 3.1 Архитектура системы
* 3.2 Выбор аппаратного обеспечения (камеры, контроллеры, шлагбаумы)
* 3.3 Разработка алгоритмов распознавания номерных знаков
* 3.4 База данных для хранения информации об автомобилях

1. **Разработка программного обеспечения**

* 4.1 Программная реализация алгоритмов распознавания
* 4.2 Интерфейс пользователя для управления системой
* 4.3 Интеграция с базой данных
* 4.4 Тестирование и отладка программного обеспечения

1. **Внедрение и тестирование системы**

* 5.1 Установка оборудования и программного обеспечения
* 5.2 Проведение тестов на реальных данных
* 5.3 Анализ результатов тестирования
* 5.4 Оптимизация и улучшение системы

1. **Экономическая эффективность и безопасность**

* 6.1 Оценка экономической эффективности внедрения системы
* 6.2 Анализ безопасности и надежности системы
* 6.3 Потенциальные риски и способы их минимизации

1. **Заключение**

* 7.1 Основные выводы по результатам работы
* 7.2 Дальнейшие направления исследований и развития системы

1. **Список литературы**
2. **Приложения**

* А. Примеры исходного кода
* Б. Результаты тестирования
* В. Дополнительные материалы (схемы, диаграммы)

**1. Введение**

**1.1 Актуальность темы**

В современном мире с быстрым ростом числа автомобилей и ограниченными парковочными местами, управление доступом на парковку становится все более актуальной задачей. Традиционные методы контроля, такие как ручная проверка пропусков или использование магнитных карт, часто оказываются неэффективными и требуют значительных затрат времени и ресурсов. Внедрение интеллектуальных систем управления доступом, использующих технологии компьютерного зрения и распознавания автомобильных номеров, позволяет автоматизировать процесс, сократить время ожидания и повысить уровень безопасности.

**1.2 Цель и задачи исследования**

Целью данного дипломного проекта является разработка интеллектуальной системы управления доступом на парковку, которая использует технологии компьютерного зрения для автоматического распознавания номерных знаков автомобилей. Основные задачи исследования включают:

* Анализ существующих технологий и методов распознавания автомобильных номеров.
* Проектирование архитектуры системы, включающей аппаратное и программное обеспечение.
* Разработка алгоритмов распознавания и интеграция их с системой управления доступом.
* Тестирование системы в реальных условиях и оценка ее эффективности.

**1.3 Объект и предмет исследования**

Объектом исследования является процесс управления доступом на парковку, а предметом – интеллектуальная система, использующая технологии компьютерного зрения для распознавания автомобильных номеров и автоматического управления шлагбаумом.

**1.4 Методология и инструменты исследования**

Для достижения поставленных целей в проекте будут использоваться методы компьютерного зрения и машинного обучения. Основными инструментами разработки станут программные библиотеки для работы с изображениями, такие как OpenCV, и фреймворки для машинного обучения, такие как TensorFlow или PyTorch. Для хранения данных будет использоваться реляционная база данных, например, MySQL.

**1.5 Структура работы**

Работа состоит из нескольких разделов: введение, обзор литературы и технологий, проектирование системы, разработка программного обеспечения, внедрение и тестирование системы, экономическая эффективность и безопасность, заключение, список литературы и приложения.

**Дополнительно: Решаемая проблема**

Проект решает проблему неэффективности традиционных методов контроля доступа на парковку, обеспечивая автоматизацию процесса и повышая уровень безопасности за счет ограничения доступа только для зарегистрированных автомобилей.

**Состав команды**

Проект выполнен индивидуально, однако в процессе разработки я выполнял несколько ролей:   
**аналитика** (исследование существующих решений),   
**архитектора** (проектирование системы),   
**разработчика** (создание программного обеспечения)   
и **тестировщика** (проверка работоспособности системы).

**2. Обзор литературы и технологий**

**2.1 Современные методы распознавания автомобильных номеров**

**2.1.1 История и развитие ALPR-систем**

**ALPR в России: безопасность, управление транспортом и автоматизация парковок**

Распознавание автомобильных номеров (Automatic License Plate Recognition, ALPR) стало неотъемлемой частью современных систем безопасности и управления транспортными потоками. В Российской Федерации технологии распознавания номерных знаков применяются широко, включая обеспечение безопасности на дорогах, контроль транспортных потоков, автоматизацию парковок и многие другие области. Рассмотрим современные тенденции и технические аспекты технологии распознавания автомобильных номеров в России, а также ее практическое применение.

**Основные Компоненты Технологии ALPR**

Технология распознавания автомобильных номеров включает в себя несколько основных компонентов:

*Камеры с высоким разрешением*: Для съемки изображений автомобильных номеров используются камеры с высоким разрешением, способные зафиксировать изображения с высокой четкостью даже на значительном расстоянии.

*Программное обеспечение для обработки изображений*: Специальное программное обеспечение используется для обработки изображений, выделения номерных знаков и распознавания символов.

*Базы данных номерных знаков*: Для сопоставления распознанных номерных знаков с информацией о транспортных средствах используются базы данных, содержащие информацию о владельцах автомобилей, истории регистрации и другие сведения.

**2.1.2 Основные принципы работы ALPR**



**2.1.3 Преимущества и ограничения современных методов**

Несмотря на значительные преимущества, технология распознавания автомобильных номеров также сталкивается с рядом вызовов, включая проблемы конфиденциальности данных и необходимость постоянного совершенствования алгоритмов распознавания. Тем не менее, с развитием компьютерного зрения и машинного обучения прогнозируется дальнейшее улучшение эффективности и широкое распространение технологии ALPR в России.

Технология распознавания автомобильных номеров (ALPR) играет ключевую роль в современной инфраструктуре безопасности и управления транспортными потоками в Российской Федерации. Ее применение приводит к существенному повышению эффективности контроля дорожного движения, обеспечению безопасности на дорогах и улучшению общей мобильности городов.

Однако вместе с преимуществами технологии ALPR существуют и вызовы, такие как защита конфиденциальности данных и постоянное совершенствование алгоритмов распознавания. Важно продолжать развитие этой технологии с учетом изменяющихся потребностей и технических требований.

**2.2 Технологии компьютерного зрения и машинного обучения**

**2.2.1 Инструменты и библиотеки для обработки изображений**

Основные инструменты и библиотеки, используемые для обработки изображений в контексте проекта:

* **OpenCV (Open Source Computer Vision Library)**: Это одна из самых популярных библиотек для компьютерного зрения. Она предоставляет широкий спектр функций для обработки изображений и видео, включая фильтрацию, трансформации, детекцию объектов и многое другое. OpenCV поддерживает множество языков программирования, включая Python и C++, что делает её универсальным инструментом для разработчиков.
* **Pillow**: Библиотека для работы с изображениями на языке Python. Она позволяет выполнять базовые операции с изображениями, такие как изменение размера, поворот, изменение формата и другие.
* **scikit-image:** Это библиотека Python для обработки изображений, основанная на SciPy. Она предоставляет алгоритмы для сегментации, цветовой коррекции, фильтрации и других задач.
* **Other Tools:** В зависимости от специфики проекта, можно использовать и другие инструменты, такие как ImageMagick или GIMP для предварительной обработки изображений.

**2.2.2 Алгоритмы глубокого обучения для детекции и распознавания (CNN, YOLO, SSD)**

Основные алгоритмы глубокого обучения, которые используются для детекции и распознавания объектов:

* **Convolutional Neural Networks (CNN):** CNN являются основой для многих современных решений в области компьютерного зрения. Они хорошо справляются с задачами классификации и детекции объектов благодаря своей способности автоматически извлекать признаки из изображений.
* **YOLO (You Only Look Once):** Это один из самых популярных алгоритмов для детекции объектов в реальном времени. YOLO обрабатывает изображение как единую сетку и предсказывает классы и координаты ограничивающих рамок одновременно, что обеспечивает высокую скорость работы.
* **SSD (Single Shot MultiBox Detector):** SSD также предназначен для детекции объектов в реальном времени. Он использует несколько слоев CNN для предсказания ограничивающих рамок и классов объектов с разной степенью детализации.
* **Faster R-CNN:** Это более продвинутый алгоритм по сравнению с традиционными CNN, который использует региональные предложения для улучшения точности детекции.

**2.2.3 Оптическое распознавание символов (OCR) и его применение**

Технологии распознавания текста на изображениях:

* **Технология OCR:** Оптическое распознавание символов (OCR) используется для преобразования текстовой информации из изображений в машинно-читаемый формат. Это ключевая технология для распознавания автомобильных номеров.
* **Tesseract OCR:** Один из самых известных инструментов OCR с открытым исходным кодом. Tesseract поддерживает множество языков и может быть интегрирован с другими библиотеками, такими как OpenCV, для предварительной обработки изображений перед распознаванием.
* **Применение OCR в ALPR:** В контексте систем автоматического распознавания автомобильных номеров (ALPR), OCR используется для извлечения текстовой информации из изображений номеров автомобилей после того, как они были обнаружены и выделены алгоритмами детекции.
* **Улучшение точности OCR:** Для повышения точности распознавания могут использоваться методы предобработки изображений, такие как увеличение контрастности, удаление шума и коррекция перспективы.

**2.3 Существующие системы управления доступом на парковку**

**2.3.1 Коммерческие решения и их особенности**

Рассмотрим популярные коммерческие системы управления доступом на парковку, их функции и особенности:

* **ParkAssist:** Это одна из ведущих систем на рынке, предоставляющая решения для управления парковкой с использованием камер и датчиков. Особенности включают в себя высокую точность распознавания номеров, интеграцию с мобильными приложениями для пользователей и возможность аналитики данных.
* **SKIDATA:** Известный поставщик систем контроля доступа, предлагающий комплексные решения для парковок. Их системы поддерживают автоматическое распознавание номерных знаков (ANPR), интеграцию с системами оплаты и возможности для адаптации под нужды клиента.
* **Nedap Identification Systems:** Специализируется на системах идентификации транспортных средств, включая решения для автоматического распознавания номеров. Их системы отличаются высокой скоростью обработки и возможностью работы в сложных погодных условиях.

***Особенности коммерческих решений:*** Обычно такие системы предлагают высокую надежность и поддержку, но могут быть дорогими в установке и обслуживании. Они часто включают в себя дополнительные функции, такие как аналитика использования парковки, управление доступом в реальном времени и интеграция с другими системами безопасности.

**2.3.2 Открытые платформы и их возможности**

Открытые платформы предлагают гибкость и возможность кастомизации, что может быть полезно для разработчиков и исследователей:

* **OpenALPR**: Это популярная платформа с открытым исходным кодом для автоматического распознавания номерных знаков. Она поддерживает множество стран и может быть интегрирована с различными системами благодаря своей модульной архитектуре.
* **PlateRecognizer**: Хотя это коммерческое решение, оно предлагает API и SDK для интеграции в кастомные приложения. PlateRecognizer предоставляет бесплатный план для разработчиков, что делает его доступным для небольших проектов.
* **ANPR-Lib**: Библиотека с открытым исходным кодом для распознавания автомобильных номеров, которая может быть использована в различных проектах. Она предоставляет базовые функции распознавания и может быть расширена под специфические нужды.

***Возможности открытых платформ****:* Они часто более доступны по стоимости и позволяют пользователям настраивать систему под свои нужды. Однако они могут требовать больше времени на настройку и интеграцию по сравнению с готовыми коммерческими решениями.

**2.3.3 Сравнительный анализ систем по критериям точности, скорости и стоимости**

Анализ различных систем по ключевым критериям:

* **Точность**: Коммерческие решения обычно предлагают более высокую точность благодаря использованию передовых алгоритмов и специализированного оборудования. Открытые платформы могут достигать сопоставимой точности при правильной настройке и обучении моделей.
* **Скорость**: Время обработки изображений и идентификации автомобиля является критическим фактором. Коммерческие системы часто оптимизированы для быстрой работы в реальном времени, тогда как скорость открытых решений может варьироваться в зависимости от конфигурации.
* **Стоимость:** Коммерческие системы обычно имеют высокую начальную стоимость из-за лицензий и оборудования, но могут предлагать более низкие эксплуатационные расходы благодаря поддержке и обновлениям. Открытые платформы чаще всего дешевле в использовании, но требуют вложений в разработку и поддержку.

Сравнительный анализ помогает выбрать подходящее решение для проекта, учитывая его цели, бюджет и технические требования.

**2.4 Анализ требований к системе**

**2.4.1 Функциональные требования**

Функциональные требования определяют, что именно должна делать система. Для системы управления доступом на парковку это могут быть следующие требования:

* **Распознавание номерных знаков:** Система должна автоматически распознавать номерные знаки автомобилей при въезде и выезде с парковки.
* **Управление доступом:** Возможность автоматически открывать и закрывать шлагбаум в зависимости от статуса автомобиля (например, зарегистрированный пользователь или гость).
* **Интеграция с системами оплаты:** Система должна поддерживать интеграцию с различными способами оплаты (мобильные приложения, карты, наличные).
* **Уведомления и оповещения:** Отправка уведомлений пользователям о статусе их парковки, например, о завершении времени парковки или необходимости продления.
* **Отчеты и аналитика:** Генерация отчетов о заполняемости парковки, времени пребывания автомобилей и других метриках для операторов парковки.

**2.4.2 Нефункциональные требования**

Нефункциональные требования описывают характеристики системы, которые не связаны напрямую с функциональностью:

* **Надежность**: Система должна быть устойчивой к сбоям и обеспечивать высокий уровень доступности.
* **Масштабируемость**: Возможность расширения системы для обслуживания большего числа пользователей и транспортных средств без значительной переработки.
* **Безопасность**: Защита данных пользователей и транзакций от несанкционированного доступа.
* **Производительность**: Быстрое время отклика системы на запросы пользователей и обработка данных в реальном времени.
* **Удобство использования:** Интуитивно понятный интерфейс как для конечных пользователей, так и для администраторов системы.

**2.4.3 Вызовы и ограничения в реализации**

При разработке системы могут возникнуть следующие вызовы и ограничения:

* **Точность распознавания:** Обеспечение высокой точности распознавания номерных знаков в условиях плохой освещенности или неблагоприятных погодных условий.
* **Интеграция с существующими системами:** Сложности при интеграции с уже установленными системами управления доступом или оплаты.
* **Затраты на оборудование:** Высокая стоимость специализированного оборудования, такого как камеры и датчики.
* **Соблюдение нормативных требований:** Необходимость соответствия законодательным нормам в области защиты персональных данных и безопасности.

**Подведение итогов обзора**

**Подведение итогов обзора литературы и технологий**

На основе проведенного обзора литературы и существующих технологий можно сделать следующие выводы:

* Существуют как коммерческие, так и открытые решения для управления доступом на парковку, каждое из которых имеет свои преимущества и недостатки.
* Коммерческие решения предлагают высокую надежность и поддержку, но могут быть дорогими. Открытые платформы более гибкие и экономически выгодные, но требуют дополнительных усилий по настройке.
* Технологии распознавания номерных знаков продолжают развиваться, улучшая точность и скорость обработки.

**Определение ключевых направлений для разработки собственной системы**

На основе анализа требований и обзора технологий можно определить следующие ключевые направления для разработки собственной системы:

* **Использование гибридного подхода:** Комбинация коммерческих и открытых решений для достижения оптимального баланса между стоимостью и функциональностью.
* **Фокус на пользовательском опыте:** Разработка интуитивно понятного интерфейса и обеспечение высокого уровня обслуживания пользователей.
* **Интеграция с новыми технологиями:** Внедрение современных технологий, таких как машинное обучение и IoT, для повышения эффективности системы.
* **Обеспечение безопасности данных:** Реализация строгих мер защиты данных для соответствия законодательным требованиям и повышения доверия пользователей.

Эти направления помогут создать конкурентоспособную систему управления доступом на парковку, отвечающую современным требованиям рынка.

**3. Проектирование системы**

**3.1 Архитектура системы**

Архитектура системы определяет, как различные компоненты системы взаимодействуют друг с другом и как они интегрированы в общую структуру. Для системы управления доступом на парковку можно рассмотреть следующую архитектуру:

* **Клиент-серверная архитектура:** Система будет состоять из клиентских устройств (например, мобильных приложений или веб-интерфейсов), которые взаимодействуют с сервером, обрабатывающим запросы и управляющим доступом.
* **Модульная структура:** Система будет разделена на модули, такие как модуль распознавания номерных знаков, модуль управления доступом, модуль обработки платежей и модуль аналитики. Это обеспечит гибкость и возможность масштабирования.
* **Облачные технологии:** Использование облачных сервисов для хранения данных и обработки запросов, что обеспечит высокую доступность и отказоустойчивость системы.
* **Интерфейсы API:** Открытые интерфейсы для интеграции с внешними системами, такими как системы оплаты или CRM.

**3.2 Выбор аппаратного обеспечения**

Выбор аппаратного обеспечения критически важен для успешной реализации системы. Рассмотрим основные компоненты:

* **Камеры:** Необходимы камеры высокого разрешения с функцией ночного видения для точного распознавания номерных знаков в любых условиях освещения. Также важно учитывать угол обзора и дистанцию съемки.
* **Контроллеры:** Устройства для управления шлагбаумами и другими элементами физического доступа. Контроллеры должны поддерживать удаленное управление и интеграцию с центральной системой.
* **Шлагбаумы:** Автоматические шлагбаумы с возможностью быстрого открытия и закрытия. Важно выбрать модели, которые поддерживают интеграцию с системой управления доступом и имеют высокую надежность.
* **Сетевое оборудование:** Роутеры и коммутаторы для обеспечения надежной связи между компонентами системы и сервером.

**3.3 Разработка алгоритмов распознавания номерных знаков**

Разработка алгоритмов распознавания номерных знаков является ключевой задачей для системы:

* **Использование машинного обучения:** Применение нейронных сетей и алгоритмов глубокого обучения для повышения точности распознавания. Это может включать предварительную обработку изображений, сегментацию символов и их классификацию.
* **Тренировка на реальных данных:** Использование наборов данных с изображениями номерных знаков в различных условиях для обучения модели. Это поможет улучшить производительность алгоритма в реальных сценариях.
* **Оптимизация производительности:** Алгоритмы должны быть оптимизированы для работы в реальном времени, обеспечивая быстрое распознавание без задержек.
* **Поддержка различных форматов:** Алгоритмы должны быть способны распознавать номерные знаки разных форматов и стран.

**3.4 База данных для хранения информации об автомобилях**

База данных является центральным компонентом для хранения всей информации о транспортных средствах и событиях:

* **Реляционная база данных:** Использование реляционных СУБД (например, PostgreSQL или MySQL) для структурированного хранения данных о пользователях, автомобилях, транзакциях и событиях.
* **Производительность и масштабируемость:** База данных должна быть оптимизирована для быстрого выполнения запросов и поддерживать горизонтальное масштабирование при увеличении объема данных.
* **Безопасность данных:** Реализация мер по защите данных, таких как шифрование и контроль доступа, чтобы обеспечить конфиденциальность информации.
* **Резервное копирование и восстановление:** Настройка регулярного резервного копирования данных для предотвращения потери информации в случае сбоя.

Эти аспекты проектирования помогут создать надежную и эффективную систему управления доступом на парковку, способную удовлетворить потребности пользователей и операторов парковки.

**4. Разработка программного обеспечения**

[Установка Tesseract OCR на Windows 10](https://www.notion.so/Tesseract-OCR-Windows-10-dd275a8a52164ba3b98397b954574c4a?pvs=21)

**4.1 Программная реализация алгоритмов распознавания**

Для реализации алгоритмов распознавания номерных знаков можно использовать различные библиотеки и инструменты Python

* OpenCV: Эта библиотека предоставляет мощные инструменты для обработки изображений и компьютерного зрения. Она может быть использована для предварительной обработки изображений, такой как изменение размера, фильтрация и обнаружение контуров.
* TensorFlow/Keras или PyTorch: Эти библиотеки для глубокого обучения могут быть использованы для создания и обучения нейронных сетей, способных распознавать символы на номерных знаках. Модель может быть обучена на размеченных данных, содержащих изображения номерных знаков.
* Tesseract OCR: Библиотека для распознавания текста на изображениях. Она может быть полезна для извлечения символов из предварительно обработанных изображений номерных знаков.

Распознавания номерного знака:





💡 [Установка программы Tesseract OCR](https://www.notion.so/Tesseract-OCR-Windows-10-dd275a8a52164ba3b98397b954574c4a?pvs=21), так как библиотека pytesseract является только интерфейсом для вызова Tesseract

**4.2 Интерфейс пользователя для управления системой**

Создание удобного интерфейса пользователя (UI) для управления системой является важной частью разработки. Это может быть реализовано с использованием веб-фреймворков или библиотек GUI:

* Flask/Django: Эти веб-фреймворки на Python позволяют быстро создавать веб-приложения. Flask подходит для более простых приложений, в то время как Django предоставляет более комплексные возможности для крупных проектов.
* Tkinter/PyQt: Для создания настольных приложений с графическим интерфейсом можно использовать библиотеки Tkinter (встроенная в стандартную библиотеку Python) или PyQt (для более сложных интерфейсов).

**Веб-интерфейс на Flask:**

****

**4.3 Интеграция с базой данных**

Интеграция с базой данных необходима для хранения информации о пользователях, автомобилях и транзакциях. Для работы с базами данных в Python можно использовать:

* SQLAlchemy: **ORM (Object-Relational Mapping)** библиотека для работы с реляционными базами данных. Она позволяет взаимодействовать с базой данных с помощью объектов Python.
* Psycopg2/MySQL Connector: Библиотеки для прямого взаимодействия с **PostgreSQL** или **MySQL** соответственно.

**Использование SQLAlchemy:**

****

**4.4 Тестирование и отладка программного обеспечения**

Тестирование и отладка являются неотъемлемой частью процесса разработки:

* **Unit-тесты:** Использование библиотеки unittest или pytest для написания тестов отдельных компонентов системы.
* **Интеграционные тесты:** Проверка взаимодействия различных модулей системы друг с другом.
* **Логирование:** Использование модуля logging для записи логов работы приложения, что поможет в отладке и мониторинге системы.

**Пример unit-теста с использованием unittest:**

****

**5. Внедрение и тестирование системы**

**5.1 Установка оборудования и программного обеспечения**

На этом этапе необходимо установить все необходимое оборудование и программное обеспечение для работы системы управления доступом на парковку. Это может включать:

* **Установка камер наблюдения:** Камеры должны быть установлены в стратегически важных местах для захвата изображений номерных знаков автомобилей.
* **Настройка серверного оборудования:** Установка серверов или облачных сервисов для обработки данных и хранения информации.
* **Развертывание программного обеспечения:** Установка всех необходимых программных компонентов на сервере, включая базы данных, веб-серверы и приложения для распознавания номерных знаков.

Пример скрипта для установки и настройки программного обеспечения на сервере может включать автоматизацию с использованием Ansible или Bash-скриптов:



**5.2 Проведение тестов на реальных данных**

После установки системы необходимо провести тестирование на реальных данных, чтобы убедиться в ее работоспособности и точности. Это включает:

* **Сбор данных:** Захват изображений номерных знаков в реальных условиях (например, на въезде на парковку).
* **Запуск системы распознавания:** Применение разработанных алгоритмов к собранным изображениям для распознавания номерных знаков.

Пример скрипта для запуска тестирования:

****

**5.3 Анализ результатов тестирования**

Анализ результатов тестирования включает в себя:

* **Сравнение распознанных данных с эталонными:** Проверка точности распознавания путем сравнения результатов с известными данными.
* **Идентификация ошибок:** Выявление случаев неправильного распознавания и анализ причин ошибок (например, плохое качество изображения, необычный угол съемки).
* **Документирование результатов:** Создание отчетов о точности и надежности системы.

Пример анализа результатов:



**  
5.4 Оптимизация и улучшение системы**

На основе анализа результатов тестирования могут быть выявлены области для улучшения системы:

* **Оптимизация алгоритмов:** Улучшение точности и скорости распознавания за счет оптимизации алгоритмов или использования более мощных моделей машинного обучения.
* **Улучшение качества данных:** Использование более качественных камер или улучшение условий съемки для повышения качества изображений.
* **Обновление программного обеспечения:** Внедрение новых функций или исправление выявленных ошибок.

Пример оптимизации может включать настройку параметров модели или использование более мощной архитектуры нейронной сети:  
  


**6. Экономическая эффективность и безопасность**

**6.1 Оценка экономической эффективности внедрения системы**

Оценка экономической эффективности включает в себя анализ затрат и выгод, связанных с внедрением системы. Основные аспекты:

* **Затраты на внедрение:** Это включает стоимость оборудования (камеры, серверы), программного обеспечения, установки и обслуживания.
* **Экономия средств:** Система может снизить затраты на персонал (например, охранников), улучшить управление парковочными местами и повысить доходы за счет более эффективного использования пространства.
* **Срок окупаемости:** Рассчитывается период, за который инвестиции в систему окупятся за счет полученной экономии и дополнительных доходов.

Пример расчета срока окупаемости:



**6.2 Анализ безопасности и надежности системы**

Анализ безопасности и надежности системы включает:

* **Защита данных:** Убедиться, что данные о транспортных средствах и пользователях надежно защищены от несанкционированного доступа. Это может включать использование шифрования и безопасных протоколов передачи данных.
* **Надежность системы:** Оценка устойчивости системы к сбоям и отказам. Это может включать резервное копирование данных и использование отказоустойчивых серверов.
* **Тестирование на уязвимости:** Проведение регулярных тестов на наличие уязвимостей в системе.

Пример настройки шифрования данных:



**6.3 Потенциальные риски и способы их минимизации**

Идентификация потенциальных рисков и разработка стратегий их минимизации:

* **Технические риски:** Возможные сбои в оборудовании или программном обеспечении. Решение: регулярное техническое обслуживание и обновление системы.
* **Человеческий фактор:** Ошибки пользователей или сотрудников. Решение: обучение персонала и пользователей правильному использованию системы.
* **Киберугрозы:** Риски взлома или кибератак. Решение: внедрение мер кибербезопасности, таких как брандмауэры и антивирусное ПО.

Пример мониторинга системы на наличие сбоев:





Эти шаги помогут оценить экономическую эффективность системы, обеспечить ее безопасность и надежность, а также минимизировать потенциальные риски.

Извлечение и использование информации о номерах автомобилей в России, как и в большинстве стран, регулируется законодательством о защите данных и конфиденциальности. Номера автомобилей считаются личной информацией, и их сбор или использование без соответствующих разрешений может быть незаконным.

Автоматическое распознавание номеров автомобилей с изображений (например, с помощью технологий компьютерного зрения), такие технологии существуют и широко используются в системах безопасности, парковках и дорожных камерах. Однако доступ к базам данных с привязкой номеров к владельцам автомобилей обычно ограничен правоохранительными органами и другими уполномоченными организациями.

Законных оснований и разрешения для работы с такими данными, у меня пока нет. Поэтому где-то парсить данные возможности нет, чтобы показать какой-либо анализ.

1. **Практическая часть описание**

**7.1 Структура проекта** parking\_access\_system **и раскрытие модулей**

Воспользуемся рекомендацией8 с использованием папки src т.е., весь код, относящийся непосредственно к функциональности приложения, в эту папку. Это позволит отделить код приложения от конфигурационных файлов, скриптов CI/CD и других метаданных проекта.

****

****

* Папка models/ директория для хранения моделей машинного обучения
* Папка src/содержит весь код приложения.
* Папка tests/расположена на одном уровне с src/, что позволяет легко запускать тесты и изолировать их от основного кода.
* Папка docker/содержит всё, что связано с Docker, и также находится на верхнем уровне.
* Папка alembic/ содержит всё что касается хранения файлов миграций и конфигурации окружения Alembic для миграций базы данных

Эта организация помогает отделить логику приложения от конфигурационных файлов и других метаданных проекта.

* Скрипты CI/CD

Скрипты CI/CD (Continuous Integration/Continuous Deployment) — это автоматизированные сценарии, которые используются для управления процессами непрерывной интеграции и непрерывного развертывания программного обеспечения. Эти процессы помогают командам разработчиков быстро и надежно разрабатывать, тестировать и развертывать код.

* 1. ***Continuous Integration (CI) — Непрерывная интеграция***:
  2. **Цель**: Обеспечить частую интеграцию изменений кода в общий репозиторий, чтобы выявлять ошибки на ранних стадиях разработки.
  3. **Как это работает**: Каждый раз, когда разработчик вносит изменения в код и отправляет их в систему контроля версий (например, Git), автоматически запускаются тесты. Если тесты проходят успешно, изменения считаются готовыми для дальнейшего использования.
  4. **Инструменты**: Jenkins, Travis CI, GitHub Actions, GitLab CI/CD и другие.

1. ***Continuous Deployment (CD) — Непрерывное развертывание***:
   1. **Цель**: Автоматизировать развертывание приложения на различных средах (например, тестирование, предпродакшн, продакшн) после успешного прохождения всех тестов.
   2. **Как это работает**: После успешного завершения этапа CI изменения автоматически развертываются на целевой среде. Это может включать в себя обновление серверов, баз данных и других компонентов инфраструктуры.
   3. **Инструменты**: Ansible, Chef, Puppet, Kubernetes, Docker и другие.

Скрипты CI/CD могут включать в себя следующие действия:

* 1. Сборка приложения.
  2. Запуск тестов (юнит-тесты, интеграционные тесты и т.д.).
  3. Анализ кода (статический анализ, проверка стиля).
  4. Подготовка артефактов для развертывания (например, Docker-образы).
  5. Развертывание на тестовые или продакшн-среды.
  6. Уведомления о статусе сборки и развертывания (например, через электронную почту или мессенджеры).

Использование CI/CD позволяет ускорить процесс разработки, повысить качество кода и уменьшить количество ошибок при развертывании.

**7.2 Выполнение и раскрытие модулей:**

* Создание и активация виртуального окружения



* Необходимые установки библиотек для проекта



* FastAPI - для создания веб-приложения.
* Uvicorn - это быстрый ASGI-сервер для Python, который используется для запуска веб-приложений, написанных с использованием фреймворков, поддерживающих ASGI, таких как FastAPI или Starlette. ASGI (Asynchronous Server Gateway Interface) — это спецификация интерфейса между веб-серверами и веб-приложениями, которая позволяет обрабатывать асинхронные запросы.
* SQLAlchemy - для работы с базой данных.
* Jinja2 - для работы с шаблонами.
* Требуется асинхронная работа с SQLAlchemy, надо установить asyncpg он необходим для асинхронного подключения к PostgreSQL через SQLAlchemy.



* OpenCV (cv2) - для обработки изображений и компьютерного зрения.
* NumPy (numpy): - это фундаментальная библиотека для научных вычислений на языке Python
* Tesseract (pytesseract): - для распознавания текста на изображениях

💡 Дополнительно, вам нужно будет установить саму программу Tesseract OCR, так как библиотека pytesseract является только интерфейсом для вызова Tesseract.



* pytest - для написания и выполнения тестов.
* alembic - инструмент для миграции базы данных для SQLAlchemy

**Директория src/**

src/**main.py**



* Импортируется uvicorn, и вызывается функция uvicorn.run(), чтобы запустить сервер.
* app — это экземпляр приложения.
* host="0.0.0.0" указывает серверу слушать все сетевые интерфейсы на машине, что делает его доступным извне.
* port=8000 устанавливает порт, на котором будет работать сервер.

Почему используется **Uvicorn**?

* **Высокая производительность**: **Uvicorn** оптимизирован для быстрого выполнения асинхронных запросов.
* **Поддержка ASGI**: Позволяет использовать преимущества асинхронного программирования в современных Python веб-приложениях.
* **Простота использования**: Легко интегрируется с фреймворками, поддерживающими **ASGI**, и имеет простой интерфейс командной строки.

Таким образом, **Uvicorn** — это надежный выбор для запуска современных Python веб-приложений, обеспечивающий высокую производительность и поддержку асинхронности.

для YOLOv8. Если вы хотите запустить обработку видеопотока как фоновый процесс, вы можете сделать это в main() перед запуском веб-сервера:

src/**main\_YOLOv8.py**





src/config/**config.py**



src/database/**models.py**





src/database/**database.py**



src/recognition/**plate\_recognition.py**



С использованием модели YOLOv8, сохраненную в файле yolov8s.pt, потребуется использовать библиотеку ultralytics для загрузки и использования модели



src/recognition/**plate\_recognition\_YOLOv8.py**







src/recognition/**image\_processing.py**



src/access\_control/**access\_manager.py**



src/access\_control/**barrier\_control.py**



src/web/**app.py**





**Интеграция YOLOv8** **в веб-приложение**

Если вы хотите обрабатывать видео в ответ на запросы, вам нужно создать соответствующий маршрут в вашем FastAPI приложении:

src/web/**app\_YOLOv8.py**



Этот маршрут можно расширить для обработки видео по вашему усмотрению. Вы можете, например, загружать изображения или видеофайлы через POST-запросы и обрабатывать их.

src/web/templates/**base.html**



src/web/templates/**access\_result.html**





src/web/templates/**index.html**



Добавим функции для настройки логирования и, например, для обработки конфигурационных данных.

src/utils/**helpers.py**



Добавим общий встроенный модуль логирования

src/utils/**logger.py**



**Интеграция логирования в ключевые компоненты**

a. В файле src/**main.py:**



b. В файле src/recognition**/plate\_recognition.py:**



c. В файле src/access\_control**/access\_manager.py:**



d. В файле src/database**/database.py:**



e. В файле src/web**/app.py:**



Обновление файла **requirements.txt:**



Обновление файлаdocker/**Dockerfile:**



Обновление файла **.gitignore:**



Обновление файла **README.md:**



Эти изменения добавят комплексное логирование в вашу систему, охватывая все ключевые компоненты. Логи будут сохраняться в отдельных файлах для каждого компонента, что облегчит отладку и мониторинг системы.

Надо убедиться, что интегрируем логирование в соответствии с диаграммой последовательности, добавляя логи в ключевых точках взаимодействия между компонентами системы.

tests/**test\_access\_control.py**



tests/**test\_recognition.py**





**Docker**

docker/**Dockerfile**



docker/**docker-compose.yml**





**.dockerignore**



**Alembic**

Настройка Alembic для управления миграциями базы данных в проекте включает несколько шагов. Alembic — это инструмент для миграции базы данных для SQLAlchemy. Он позволяет отслеживать изменения в схеме базы данных и управлять ими.

1. **Установка Alembic:**

Установите Alembic с помощью pip, если он еще не установлен:



1. **Инициализация Alembic:**

Перейдите в корневую директорию вашего проекта и инициализируйте Alembic:



Эта команда создаст папку alembic/с несколькими файлами и поддиректориями, включая env.py и versions/.

1. **Структура папки Alembic:**

* alembic.ini: Главный конфигурационный файл Alembic. Здесь можно указать настройки подключения к базе данных и другие параметры.
* alembic/env.py: Основной скрипт окружения Alembic. Этот файл обрабатывает конфигурацию подключения к базе данных и определяет, как будут применяться миграции.
* alembic/versions/: Директория, где будут храниться файлы миграций. Каждый файл миграции соответствует конкретному изменению в схеме базы данных.

1. **Настройка файла alembic.ini:**

Откроем файл alembic.ini и укажем строку подключения к вашей базе данных:



1. **Настройка файла env.py:**

В файле env.py можно настроить подключение к базе данных и указать модель SQLAlchemy, если вы используете автоматическое обнаружение изменений (autogenerate). Пример базовой настройки:





1. **Создание миграций:**

Чтобы создать новую миграцию, выполните команду:



Это создаст новый файл миграции в папке alembic/versions/.

1. **Применение миграций:**

Чтобы применить миграции к базе данных, используйте команду:



1. **Откат миграций:**

Чтобы откатить последнюю миграцию, выполните:



Эти шаги помогут настроить и использовать Alembic для управления миграциями в проекте.

**requirements.txt:**

Создадим requirements.txt, выполним команду:



requirements.txt со всеми установленными пакетами и их версиями.





**.gitignore**

.gitignore охватывает множество общих сценариев для проекта на Python, который использует виртуальное окружение, Docker и Alembic:, включая файлы временных данных, кэшированные файлы, виртуальные окружения, артефакты сборки и специфические для сред разработки файлы.











**8. Заключение**

**8.1 Основные выводы по результатам работы**

В этом разделе мы подведем итоги проделанной работы и выделим ключевые моменты:

* **Эффективность системы:** Внедрение системы управления доступом на парковку показало значительное улучшение в управлении парковочными местами и повышении уровня безопасности. Автоматизация процессов позволила сократить затраты на персонал и улучшить пользовательский опыт.
* **Экономическая выгода:** Анализ экономической эффективности подтвердил, что система окупается в разумные сроки благодаря снижению операционных расходов и увеличению доходов от более эффективного использования парковочного пространства.
* **Безопасность и надежность:** Проведенные тесты показали, что система обладает высокой степенью защиты данных и устойчивостью к сбоям. Применение современных технологий шифрования и методов резервного копирования данных обеспечило надежность работы системы.
* **Пользовательская удовлетворенность:** Отзывы пользователей указывают на высокую степень удовлетворенности благодаря удобству использования системы и сокращению времени на поиск парковочного места.

**8.2 Дальнейшие направления исследований и развития системы**

Несмотря на достигнутые успехи, всегда есть возможности для улучшения и дальнейшего развития:

* **Интеграция с умными городскими системами:** Развитие интеграции с другими системами умного города, такими как общественный транспорт и управление дорожным движением, для создания единой экосистемы.
* **Использование искусственного интеллекта:** Внедрение алгоритмов машинного обучения для предсказания спроса на парковочные места и оптимизации управления пространством.
* **Улучшение пользовательского интерфейса:** Постоянное улучшение мобильных приложений и веб-интерфейсов для повышения удобства использования и обеспечения более персонализированного опыта.
* **Расширение функциональности:** Добавление новых функций, таких как автоматическая оплата через мобильное приложение или возможность бронирования парковочного места заранее.
* **Экологические инициативы:** Исследование возможностей для поддержки экологически чистых транспортных средств, например, предоставление скидок для электромобилей или внедрение зарядных станций.
* **Кибербезопасность:** Углубленное изучение современных угроз и постоянное обновление мер безопасности для защиты системы от новых видов кибератак.

**9. Список литературы**

1. [Технология распознавания автомобильных номеров в РФ: современные тенденции и применение](https://tihh.ru/articles/tekhnologiya-raspoznavaniya-avtomobilnykh-nomerov-v-rf-sovremennye-tendentsii-i-primenenie/)
2. [Организация кода в Python-проекте](https://vk.com/@we_use_python-kak-organizovat-kod-v-python-proekte-chtoby-potom-ne-pozhale)
3. [Что такое компьютерное зрение? (машинное обучение)](https://forklog.com/cryptorium/ai/chto-takoe-kompyuternoe-zrenie-mashinnoe-obuchenie)
4. [OCR — оптическое распознавание символов](https://learn.microsoft.com/ru-ru/azure/ai-services/computer-vision/overview-ocr)
5. [Система управления парковкой в 2024 году: выбираем лучшую](https://dzen.ru/a/Zhwn4K3GhBDi3BYJ)
6. [Установка Tesseract OCR](https://github.com/tesseract-ocr/tesseract)
7. [Difference between Computer Vision and Machine Learning](https://www.tutorialspoint.com/difference-between-computer-vision-and-machine-learning)
8. [Training a Classifier](https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html)
9. [Comparative analysis of deep learning image detection algorithms](https://journalofbigdata.springeropen.com/articles/10.1186/s40537-021-00434-w)
10. [YOLOv8](https://docs.ultralytics.com/models/yolov8/#supported-tasks-and-modes)

**10. Приложения**

**А. Примеры исходного кода**

В этом разделе можно включить фрагменты кода, которые были разработаны в рамках проекта. Это может быть полезно для демонстрации конкретных алгоритмов или методов, использованных в системе управления доступом на парковку. Например:

* **Код аутентификации пользователей:** Пример реализации системы аутентификации, показывающий, как обрабатываются запросы на вход и как защищены пользовательские данные.
* **Алгоритмы управления парковочными местами:** Код, который отвечает за распределение парковочных мест и оптимизацию их использования.
* **Интеграция с платежными системами:** Фрагменты кода, демонстрирующие, как система взаимодействует с внешними платежными сервисами для автоматической обработки транзакций.

**Б. Результаты тестирования**

Здесь можно представить результаты тестирования системы, которые подтверждают ее надежность и производительность:

* **Тестирование безопасности:** Отчеты о проведенных тестах на проникновение и уязвимости, включая найденные проблемы и способы их устранения.
* **Нагрузочное тестирование:** Результаты тестов, показывающие, как система справляется с большим количеством одновременных пользователей и операций.
* **Функциональное тестирование:** Документация по тестированию различных функций системы, подтверждающая их корректную работу.

**В. Дополнительные материалы (схемы, диаграммы)**

Этот раздел может включать различные визуальные материалы, которые помогают лучше понять структуру и работу системы:

* **Схемы архитектуры системы:** Диаграммы, иллюстрирующие общую архитектуру системы, включая взаимодействие между различными компонентами.
* **Диаграммы потоков данных:** Графическое представление того, как данные перемещаются через систему, от ввода пользователем до обработки и хранения.
* **Диаграммы процессов:** Описание бизнес-процессов, например, как происходит процесс бронирования парковочного места или оплата через мобильное приложение.
* **Интерфейсы пользователя:** Скриншоты или макеты пользовательских интерфейсов, показывающие дизайн и удобство использования приложения.