**ADR #1: Выбор базы данных для системы автоматизированной проверки багажа**

**1. Статус:** Принято

**2. Контекст:**

Команда из 3 разработчиков разрабатывает систему автоматизированной проверки багажа. Для хранения данных, связанных с пользователями, метриками и временными данными, необходимо выбрать подходящую базу данных. Важными критериями являются производительность, надежность, простота интеграции и возможность расширения функциональности.

**3. Решение:**

Выбираем реляционную базу данных PostgreSQL для хранения данных в нашем проекте.

* **Обоснование:**
  + **Производительность и Надежность:** PostgreSQL известна своей высокой производительностью и надежностью, что критично для обработки большого объема данных, связанных с багажом и пользователями.
  + **Распространенность и Знакомство:** Большинство членов команды имеют опыт работы с PostgreSQL, что сокращает время на адаптацию и обучение.
  + **Открытый исходный код:** Открытый исходный код обеспечивает гибкость и возможность кастомизации.
  + **Масштабируемость:** PostgreSQL хорошо масштабируется, позволяя обрабатывать возрастающие объемы данных.
  + **Интеграция с инструментами:** PostgreSQL хорошо интегрируется с различными инструментами и языками программирования.
  + **Расширяемость:** Возможность использовать расширения, например, pgvector для векторного поиска, предоставляет дополнительные преимущества.
  + **Интеграция с Azure AI:** PostgreSQL позволяет интегрировать сервисы Azure, такие как Azure OpenAI и Azure Cognitive Services, что открывает возможности для реализации задач ИИ (обработка естественного языка, анализ тональности и т.д.).

**4. Обоснование:**

* PostgreSQL представляет собой зрелую и надежную реляционную базу данных с активным сообществом разработчиков.
* Её функциональные возможности, включая расширяемость и поддержку сложных типов данных, соответствуют требованиям проекта.
* Знакомство команды с PostgreSQL позволит ускорить разработку и снизить вероятность ошибок.
* Интеграция с Azure AI позволит внедрить функции на основе ИИ.

**5. Альтернативы:**

* **MySQL:** Популярная реляционная база данных, но менее функциональна по сравнению с PostgreSQL (в контексте конкретного проекта).
* **Microsoft SQL Server:** Хорошая реляционная база данных, но менее предпочтительна из-за лицензионных ограничений и отсутствия опыта у команды.
* **NoSQL базы данных (MongoDB, Cassandra):** NoSQL базы данных могут быть полезны при определенных сценариях (например, если данные не имеют строгую схему), но для проекта, где важны реляционные связи, PostgreSQL более подходит.
* **SQLite:** Легковесная база данных, подходит для небольших проектов, но не обеспечивает необходимую масштабируемость и надежность.

**6. Последствия:**

* **Положительные:**
  + Быстрое начало разработки за счет наличия опыта у команды.
  + Возможность масштабирования в будущем.
  + Интеграция с Azure AI для расширения функциональности.
  + Возможность использования векторного поиска.
  + Надежное и производительное хранилище данных.
* **Отрицательные:**
  + Требуется больше ресурсов (по сравнению с NoSQL или SQLite) для развертывания и поддержки.
  + Потенциальные сложности с оптимизацией при больших объемах данных (требуется оптимизация запросов).

**7. Дальнейшие шаги:**

* Разработать схему базы данных с учетом требований к проекту.
* Создать миграции для управления изменениями схемы.
* Оптимизировать запросы к базе данных и настроить индексацию.
* Интегрировать PostgreSQL с другими компонентами системы.
* Начать оптимизацию и улучшения во внутренней части PostgreSQL.
* Рассмотреть необходимость внедрения pgvector для векторного поиска.

**Дата:2024-09-12**

**ADR #2: Выбор архитектуры модели ИИ для распознавания запрещенных объектов на видеопотоке**

**1. Статус:** Принято

**2. Контекст:**

Необходимо разработать систему видеонаблюдения, способную в режиме реального времени распознавать запрещенные объекты (например, оружие, наркотики, опасные предметы) на видеопотоке. Система должна быть точной, надежной и способной обрабатывать видеопоток с высокой частотой кадров. Важно минимизировать ложные срабатывания и обеспечить быструю реакцию.

**3. Решение:**

Выбираем архитектуру на основе комбинации следующих нейросетевых моделей:

* **Обнаружение объектов (Object Detection):** YOLO (You Only Look Once) v5 или v7.
  + **Обоснование:**
    - **Скорость:** YOLO обеспечивает высокую скорость обработки, что критически важно для видеопотоков в реальном времени.
    - **Точность:** YOLO имеет хорошую точность обнаружения объектов.
    - **Баланс:** YOLO обеспечивает хороший баланс между скоростью и точностью.
    - **Доступность:** Хорошо документирован и есть готовые реализации.
* **Классификация объектов (Object Classification):** Сверточная нейронная сеть (CNN), например, ResNet50, EfficientNet или аналогичная.
  + **Обоснование:**
    - **Глубокое обучение:** CNN хорошо справляются с классификацией изображений, позволяя точно определить класс обнаруженного объекта.
    - **Обучаемость:** Есть предобученные модели, которые можно дообучить для конкретных задач.
    - **Точность:** CNN обеспечивают высокую точность классификации.
* **Временная модель (Temporal Model):** (Опционально, в зависимости от конкретной задачи) Рекуррентная нейронная сеть (RNN), например, LSTM (Long Short-Term Memory) или GRU (Gated Recurrent Unit).
  + **Обоснование:**
    - **Временная зависимость:** RNN могут учитывать временную последовательность кадров, что может повысить точность распознавания, например, для движущихся объектов или для распознавания действия (например, человек достает пистолет).
    - **Контекст:** RNN могут учитывать контекст, например, что объект в руках человека - это пистолет.
    - **Опциональность:** Применяется только если требуется анализ временной зависимости в видеопотоке.

**4. Обоснование:**

* **YOLO для быстрого обнаружения:** YOLO отлично подходит для обнаружения потенциально запрещенных объектов в реальном времени, выделяя области интереса (bounding boxes) на видеокадре.
* **CNN для точной классификации:** CNN точно определяет, что именно находится внутри выделенной области, классифицируя объект по заданным классам (например, пистолет, нож, наркотики).
* **RNN для временного анализа:** RNN помогает анализировать динамику, учитывать движение объектов и их взаимодействие с окружением.

**5. Альтернативы:**

* **SSD (Single Shot MultiBox Detector):** SSD также может использоваться для обнаружения объектов, но YOLO обычно показывает лучшую скорость и точность.
* **Faster R-CNN:** Faster R-CNN обеспечивает более высокую точность, но может быть медленнее, чем YOLO.
* **Традиционные методы компьютерного зрения:** Традиционные методы (например, на основе Haar-признаков) могут быть быстрыми, но обычно менее точными, чем современные нейросети.

**6. Последствия:**

* **Положительные:** Высокая точность обнаружения запрещенных объектов, снижение ложных срабатываний, возможность работы в режиме реального времени.
* **Отрицательные:** Высокие вычислительные требования (особенно при использовании RNN), необходимость в большом объеме размеченных данных для обучения, сложность модели и ее отладки.

**7. Дальнейшие шаги:**

* Провести сравнительное тестирование YOLO v5 и v7 для определения оптимальной версии.
* Экспериментировать с разными моделями CNN для классификации (ResNet, EfficientNet) для выбора оптимальной.
* Оценить необходимость и эффективность использования RNN.
* Собрать и разметить достаточное количество данных для обучения.
* Оптимизировать модель для работы на целевой аппаратной платформе.

**Дата:2024-09-19**

**ADR #3: Выбор RabbitMQ в качестве брокера сообщений**

**1. Статус:** Принято

**2. Контекст:**

В рамках нашего проекта необходимо реализовать асинхронное взаимодействие между различными микросервисами и компонентами. Для этого требуется брокер сообщений, который обеспечит надежную доставку, маршрутизацию и обработку сообщений. Рассматривается несколько вариантов, включая RabbitMQ, Kafka и другие.

**3. Решение:**

Выбираем RabbitMQ в качестве брокера сообщений для асинхронного взаимодействия в проекте.

* **Обоснование:**
  + **Надежность:** RabbitMQ обеспечивает надежную доставку сообщений с подтверждениями и возможностью восстановления после сбоев.
  + **Гибкость:** Поддержка различных моделей обмена сообщениями (direct, topic, fanout, headers) позволяет реализовывать разнообразные сценарии взаимодействия.
  + **Широкое распространение:** RabbitMQ является популярным брокером сообщений с большим сообществом и обширной документацией.
  + **Совместимость:** Хорошая совместимость с различными языками программирования и фреймворками.
  + **Легкость в настройке и использовании:** Простая установка, настройка и управление через веб-интерфейс.
  + **Поддержка плагинов:** Возможность расширения функциональности с помощью плагинов (например, плагин для отложенных сообщений).
  + **Поддержка AMQP:** RabbitMQ реализует протокол AMQP (Advanced Message Queuing Protocol), который обеспечивает взаимодействие между разными системами.

**4. Обоснование:**

* RabbitMQ обеспечивает надежную и предсказуемую доставку сообщений, что критически важно для асинхронного взаимодействия между сервисами.
* Гибкость маршрутизации сообщений позволит реализовать сложные сценарии взаимодействия в нашем проекте.
* Большое сообщество и наличие обширной документации упростит разработку и поддержку системы.
* RabbitMQ имеет хорошую производительность при использовании не очень больших сообщений, что соответствует нашим потребностям.

**5. Альтернативы:**

* **Apache Kafka:** Kafka более ориентирован на потоковую обработку данных и подходит для больших объемов данных, но сложнее в настройке и использовании, чем RabbitMQ.
* **ActiveMQ:** Также популярный брокер сообщений, но уступает RabbitMQ в гибкости и сообществе.
* **Redis Pub/Sub:** Простой и быстрый, но не обладает всеми возможностями надежной доставки сообщений, которые предоставляет RabbitMQ.
* **ZeroMQ:** Библиотека для обмена сообщениями, но не является брокером сообщений в классическом понимании.

**6. Последствия:**

* **Положительные:**
  + Надежное асинхронное взаимодействие между сервисами.
  + Гибкость маршрутизации и обработки сообщений.
  + Упрощение архитектуры системы.
  + Возможность масштабирования путем добавления новых экземпляров RabbitMQ.
  + Уменьшение связности между компонентами.
* **Отрицательные:**
  + Требуется развертывание и настройка отдельного компонента (брокера сообщений).
  + Необходимо учитывать возможные проблемы с доставкой сообщений и их обработкой (очереди, ошибки).
  + Вводит дополнительную сложность в архитектуру системы.
  + Может потребоваться настройка мониторинга для RabbitMQ.

**7. Дальнейшие шаги:**

* Установить и настроить RabbitMQ на тестовом окружении.
* Разработать модель обмена сообщениями (очереди, обмены, маршрутизация) для нашего проекта.
* Реализовать отправку и обработку сообщений в наших микросервисах.
* Провести тестирование системы обмена сообщениями.
* Настроить мониторинг и логирование RabbitMQ.
* Уточнить конфигурацию RabbitMQ для production окружения.

**Дата:2024-10-10**

**ADR #4: Использование Windows Server для системы распознавания запрещенных объектов**

**1. Статус:** Отклонено

**2. Контекст:**

Для развертывания системы распознавания запрещенных объектов на видеопотоке рассматриваются различные операционные системы, включая Windows Server и Linux-дистрибутивы. Изначально рассматривался вариант использования Windows Server как привычной и знакомой среды для разработки и развертывания, однако, существуют факторы, которые влияют на принятие решения.

**3. Решение:**

Отклоняем использование Windows Server в качестве операционной системы для развертывания системы распознавания запрещенных объектов.

* **Обоснование отказа:**
  + **Отсутствие поддержки производителя:** Microsoft (разработчик Windows Server) прекратила поддержку и продажу своего ПО на территории Российской Федерации. Это означает отсутствие возможности получения технической поддержки, обновлений безопасности и лицензионного использования ПО, что влечет за собой риски для безопасности и стабильности системы.
  + **Зависимость от проприетарного ПО:** Использование Windows Server создает зависимость от проприетарного ПО, что ограничивает гибкость и контроль над системой.
  + **Лицензионные ограничения:** Использование Windows Server без лицензии незаконно и несет риски судебных разбирательств. Покупка лицензии в условиях ограничений со стороны Microsoft невозможна.
  + **Стоимость:** Стоимость лицензий на Windows Server и сопутствующее ПО существенно выше по сравнению с бесплатными Linux-дистрибутивами.
  + **Ограниченная кастомизация:** Возможности кастомизации Windows Server ограничены по сравнению с Linux.

**4. Обоснование решения:**

* Из-за ухода Microsoft с российского рынка и невозможности легальной покупки лицензий и получения поддержки Windows Server не является надежным и стабильным решением.
* Риски безопасности и отсутствие технической поддержки делают Windows Server непригодным для использования в критически важной системе распознавания запрещенных объектов.
* Бесплатные Linux-дистрибутивы, такие как Ubuntu или Debian, предоставляют более гибкую и надежную альтернативу.
* Linux является более подходящей платформой для развертывания и оптимизации систем на основе искусственного интеллекта, благодаря большей гибкости, доступности и управлению ресурсами.

**5. Альтернативы:**

* **Linux-дистрибутивы (Ubuntu, Debian, CentOS):** Представляют собой бесплатные операционные системы с открытым исходным кодом, обладающие высокой надежностью, гибкостью и широкой поддержкой.
* **Другие коммерческие операционные системы:** Не рассматриваются ввиду отсутствия экономической целесообразности и возможной зависимости от поставщиков.

**6. Последствия:**

* **Положительные:**
  + Избежание рисков, связанных с отсутствием поддержки и лицензирования Windows Server.
  + Снижение затрат на лицензии и сопутствующее ПО.
  + Возможность более гибкой настройки и кастомизации системы.
  + Увеличение безопасности и стабильности системы.
* **Отрицательные:**
  + Потенциальная необходимость обучения разработчиков работе с Linux.
  + Может потребоваться перенос части уже разработанного кода.

**7. Дальнейшие шаги:**

* Выбрать конкретный Linux-дистрибутив (Ubuntu, Debian, CentOS или другой) для развертывания системы.
* Подготовить окружение для разработки и тестирования на выбранной операционной системе.
* Перенести и адаптировать существующий код под Linux.
* Протестировать систему на новой платформе.

**Дата:2024-10-12**

**ADR #5: Выбор Linux Ubuntu в качестве операционной системы для системы распознавания запрещенных объектов**

**1. Статус:** Принято

**2. Контекст:**

После отклонения решения об использовании Windows Server (ADR #4) для развертывания системы распознавания запрещенных объектов на видеопотоке, необходимо выбрать подходящую операционную систему. Рассматриваются различные Linux-дистрибутивы, включая Ubuntu, Debian, CentOS и другие.

**3. Решение:**

Выбираем Linux Ubuntu в качестве операционной системы для развертывания системы распознавания запрещенных объектов.

* **Обоснование:**
  + **Широкая распространенность:** Ubuntu является одним из самых популярных дистрибутивов Linux, что обеспечивает большую пользовательскую базу, обширное сообщество и множество доступных ресурсов (документация, обучающие материалы).
  + **Активное развитие и поддержка:** Ubuntu активно развивается Canonical, компанией-разработчиком, и имеет хорошую поддержку сообщества.
  + **Простота использования:** Ubuntu ориентирована на удобство пользователя, что облегчает процесс развертывания и администрирования.
  + **Большое количество пакетов:** Ubuntu имеет огромный репозиторий программного обеспечения, что позволяет легко устанавливать необходимые зависимости и инструменты.
  + **Совместимость с AI-фреймворками:** Ubuntu хорошо совместима с популярными фреймворками для машинного обучения и нейронных сетей (TensorFlow, PyTorch), что важно для нашей системы.
  + **Поддержка контейнеризации:** Ubuntu хорошо поддерживает Docker и другие технологии контейнеризации, что упрощает процесс развертывания и управления микросервисами.
  + **Бесплатность:** Ubuntu является бесплатной операционной системой с открытым исходным кодом, что снижает затраты на лицензирование.
  + **Стабильность и надежность:** Ubuntu LTS (Long Term Support) версии обеспечивают длительную поддержку и стабильность, что важно для производственной среды.
  + **Хорошая поддержка оборудования:** Ubuntu хорошо поддерживает различное аппаратное обеспечение, включая серверное оборудование, необходимое для обработки видеопотока.

**4. Обоснование решения:**

* Ubuntu представляет собой зрелый и надежный дистрибутив Linux, который соответствует требованиям проекта.
* Большая пользовательская база и активное сообщество обеспечивают хорошую поддержку и доступность необходимых ресурсов.
* Простота использования и совместимость с AI-фреймворками облегчат разработку и развертывание системы распознавания запрещенных объектов.
* Бесплатность Ubuntu снизит затраты на лицензирование ПО.
* Ubuntu LTS версии предоставляют необходимую стабильность для длительной эксплуатации.

**5. Альтернативы:**

* **Debian:** Также надежный и стабильный дистрибутив, но Ubuntu, по мнению команды, более удобен в использовании.
* **CentOS/Rocky Linux:** Хороши для серверных задач, но Ubuntu имеет более дружественный интерфейс и более современное программное обеспечение.
* **Другие Linux-дистрибутивы:** Не рассматриваются, так как Ubuntu лучше соответствует требованиям проекта и обладает проверенной репутацией.

**6. Последствия:**

* **Положительные:**
  + Стабильная и надежная операционная система.
  + Простота развертывания и администрирования.
  + Хорошая совместимость с AI-фреймворками.
  + Большое сообщество и доступная документация.
  + Бесплатность и отсутствие лицензионных ограничений.
  + Хорошая поддержка контейнеризации.
* **Отрицательные:**
  + Может потребоваться некоторое время на освоение Ubuntu, если команда не имеет опыта работы с Linux.
  + Некоторые специфические драйвера и утилиты могут быть доступны только через сторонние репозитории.

**7. Дальнейшие шаги:**

* Выбрать конкретную версию Ubuntu LTS для развертывания системы.
* Подготовить серверную инфраструктуру с Ubuntu.
* Установить и настроить необходимые зависимости и инструменты.
* Перенести и адаптировать существующий код под Ubuntu.
* Протестировать систему на новой платформе.
* Настроить автоматизацию развертывания и обновления системы.

**Дата:2024-10-17**