Создание системы непрерывного анализа качества трансляции с камер видеонаблюдения на строительных площадках через алгоритм детектирования Кэнни

Д.И. Сигалов, М.Г. Жабицкий

Аннотация **- На строительных площадках установлены ip-камеры, работающие через протокол RTSP (real time streaming protocol). Целью исследовательской работы является погружение в алгоритм детектирования Кэнни, его реализация в качестве одного из микросервисов нашей системы, а также разработка остальных модулей системы для принятия решений о некачественном видеопотоке. Результатом исследовательской работы является система, через которую заинтересованные лица могут регистрировать RTSP камеры, просматривать работу детектирования границ с каждым кадром видеопотока, а также получать уведомления при обнаружении некачественной трансляции.**

Ключевые слова **– RTSP, REST, Токен доступа, JSON, оператор Кэнни, БД, CV2, OAUTH 2.0, KEYCLOAK, S3 MINIO.**

I. ВВЕДЕНИЕ

В современном мире жилищное строительство занимает одно из ведущих мест в экономике любой промышленно развитой страны. Каждый из процессов на строительной площадке требует особого контроля для успешного завершения. Одним из инструментов такого контроля является видеонаблюдение. Камеры видеонаблюдения, расположенные на строительных площадках, играют важную роль для осуществления таких функций как: безопасность, удаленный мониторинг, документирование событий. Основная проблема такого способа контроля является отсутствие систем автоматического обнаружения дефектов при трансляции. Ведь огромное количество факторов может повлиять на изображения в видеопотоке, например неблагоприятные погодные условия, которые способны искажать транслируемое видео или поломка камеры, вызванное разного рода внешними условиями. Обнаружение подобного дефекта в настоящее время контролируется человеческим ресурсом, что в свою очередь имеет ряд недостатков: ограничение в просмотре, так как если на площадке установлено множество камер видеонаблюдения, человек не способен контролировать их все, ограничение в постоянном мониторинге, так как человеческий ресурс не способен непрерывно осуществлять контроль за трансляцией и также высокие затраты, так как увеличение процессов видео мониторинга требует увеличения сотрудников.

Статья получена 29 мая 2024.
Сигалов Давид Игоревич, Национальный Исследовательский Ядерный Университет МИФИ, магистрант, duff1k@inbox.ru
Жабицкий Михаил Георгиевич, Национальный Исследовательский Ядерный Университет МИФИ, Заместитель директора ВИШ, jabitsky@mail.ru

II. ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К СИСТЕМЕ

Разрабатываемая система включает в себя несколько ключевых микросервисов, каждый из которых выполняет специфические функции и задачи. Для эффективного функционирования системы необходимо чётко определить требования к каждому микросервису системы. В данном разделе подробно опишем требования к каждому из перечисленных сервисов, что позволит обеспечить их корректное и эффективное взаимодействие в рамках единой системы.

Сервис анализа видеопотока:

- Сервис должен обеспечивать безопасное подключение, при котором запросы могут отправлять только аутентифицированные и авторизованные пользователи.
- Сервис должен в асинхронном режиме отправлять RTSP ссылку на сервис расчета количества контуров.
- Сервис должен в асинхронном режиме для каждой заведенной камеры сравнивать количество контуров на каждых кадрах друг с другом.
- Сервис должен отправлять почтовые уведомления пользователям, которые были указаны при добавлении камеры в систему.

Сервис расчета количества контуров:

- Сервис должен обеспечивать безопасное подключение, при котором запросы могут отправлять только аутентифицированные и авторизованные пользователи.

- Сервис должен на вход принимать RTSP ссылку на камеру.
- Сервис должен считать количество контуров на кадре.
- Сервис должен сохранять кадр, а также наглядно демонстрировать работу алгоритма на кадре, тем самым сохраняя в хранилище несколько изображений одного кадра.

Исходя из требований к микросервисам нашей системы нам также необходимы:

- Сервер аутентификации и авторизации для обеспечения безопасной работы сервисов.
- S3 хранилище для хранения файлов (кадры и его преобразования)
- База данных.

III. РАЗРАБОТКА СИСТЕМЫ

После того как мы сформировали требования к системе, перед началом разработки необходимо описать общую архитектуру системы и рассмотреть стек технологий, который будет использоваться. Это позволит четко определить структуру системы, взаимодействие между ее компонентами и выбрать наиболее подходящие инструменты и платформы для реализации поставленных задач. Архитектура сервиса представлена на рисунке 1.

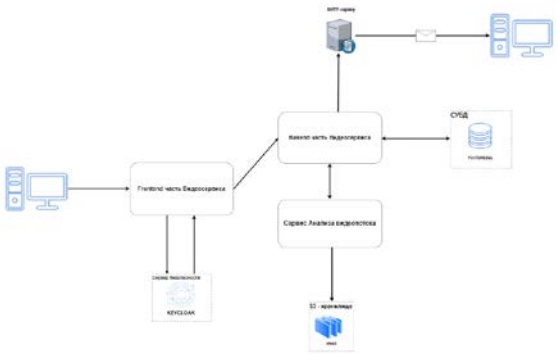


Рисунок 1 – Архитектура системы

В качестве СУБД будет использоваться PostgreSQL, так как в нем высокая скорость выполнения операций ввода-вывода и эффективная обработка большого количества транзакций в секунду, а также предоставляет поддержку ACID (атомарность, согласованность, изоляция, долговечность) для обеспечения надежности и целостности данных.

Структура БД:

1. Таблица "users_kc" (пользователи):

id (первичный ключ)

user_name (имя пользователя)

user_email (почта пользователя)

created_at (дата создания записи)

Таблица хранит данные о пользователе, запись появляется при первом входе в систему.

2. Таблица "camera" (видео):

id (первичный ключ)

user_kc_id (внешний ключ, ссылается на таблицу "users")

rtsp_name (название видео)

rtsp_link (RTSP ссылка на видео)

metter (счетчик камеры)

last_counter_image (количество нормальных контуров)

problem_link (ссылка на проблемные изображения)

email_notification (почта для уведомлений по данному видео)

created_at (дата создания записи)

Таблица Camera хранит подробную информацию о добавленных видео пользователя, поэтому есть внешний ключ, который ссылается на таблицу User_KC, а именно на его идентификатор (user_kc_id).

3. Таблица "cadr" (анализ кадров):

id (первичный ключ)

camera_id (внешний ключ, ссылается на таблицу "camera")

counters_count (количество контуров на кадре)

cadr_link (ссылка на изображение с кадром)

Таблица кадров хранит информацию о количестве контуров на каждом кадре видеопотока, соответственно идет привязка к определенной камере из таблицы camera, где в свою очередь привязка к пользователю.

Связи между таблицами:

Таблица "camera" связана с таблицей "users_kc" через внешний ключ "user_kc_id".

Таблица "cadr" связана с таблицей "camera" через внешний ключ "camera_id".создания одного или нескольких сетевых соединений поверх другой сети). Визуализируем структуру на ERD диаграмме (рисунок 2).

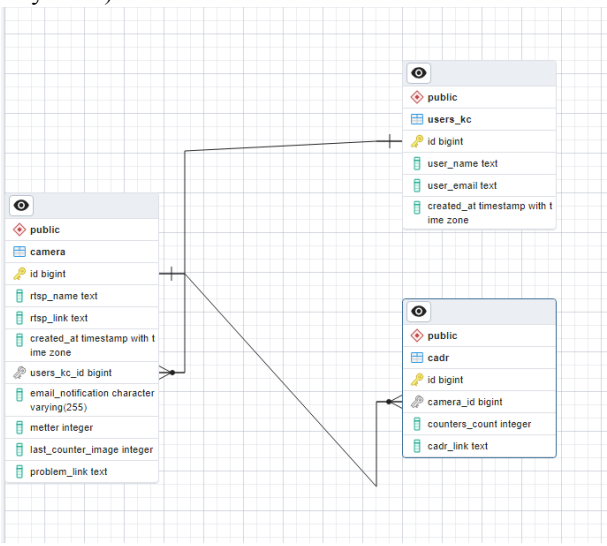


Рисунок 2 – ERD диаграмма

Сервером безопасности является open-source решение KEYCLOAK с возможностью технологией единого входа single sign-on. В основе использования данной технологии лежит протокол

авторизации OAUTH 2.0 [1], который позволяет сервисам получать ограниченный доступ аккаунта пользователя на ресурсе без необходимости раскрывать учетные данные пользователя этому приложению.

Ключевым элементом всего протокола OAUTH 2.0 является Access Token, который содержит служебные данные (медаданные, которые как бы заменяют логин/пароль), чтобы запросы пользователя допускались к сервисам. Форматом хранения и передачи токена доступа является JSON WEB TOKEN (JWT токен).

Токен доступа – секретные данные пользователя

Токен мы получаем от сервера безопасности, которым в нашем случае выступает KEYCLOAK. Если рассматривать наше приложение, сервер безопасности и сервис, к которому мы можем обратиться, только если получили токен доступа, то можно представить взаимодействие следующим образом (рисунок 3).

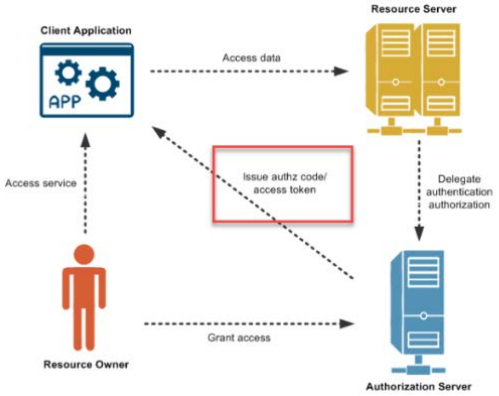


Рисунок 3 – Схема получения токена доступа для взаимодействия

Структура токена доступа состоит из следующих частей:

1) Заголовок (header). Заголовок представлен, как JSON объект, который закодирован в BASE64 формат.

2) Полезные данные (playload). Payload внутри себя представляет утверждения (claims), которые имеют за собой данные о пользователе и другую метаинформацию.

Поля:

iss (Issuer): Система, которая выдала токен.

sub (Subject): Идентификатор пользователя, для которого был выдан токен.

aud (Audience): Предполагаемая аудитория токена, обычно идентификатор клиента в сервере безопасности.

exp (Expiration Time): Время истечения срока действия токена

iat (Issued At): Время выдачи токена, также в формате времени Unix.

auth_time: Время аутентификации пользователя.

jti: Уникальный идентификатор JWT.

typ: Тип токена, например, Bearer.

azp (Authorized Party): Клиент, для которого был выдан токен.

session_state: Состояние сессии.

acr (Authentication Context Class Reference): Ссылка на контекст аутентификации.

3) Подпись (signature). Подпись является верификацией того, что отправитель является тем, за кого себя выдает, и что данные токена не были изменены после его создания. Подпись создается на основе заголовка (Header), полезной нагрузки (Payload) и секретного ключа, известному только серверу безопасности, который и является издателем токена. То есть в нашем случае это как раз клиент KEYCLOAK, которого мы создадим.

Нужно понимать, что токен доступа имеет свой срок жизни (обычно это 5-10 минут). То есть по истечении срока действия пользователю заново будет необходимо авторизоваться заново, что является не очень удобным решением. Поэтому пользователю при первой успешной аутентфикации кроме токена доступа, мы также будем выдавать токен обновления (рисунок 4). У него также будет срок жизни, но более долгий, чем у токена доступа (мы поставим 30 минут). При запросе пользователя, но с учетом истечения срока действия токена доступа, но не истечения срока действия токена обновления, наше приложение само обновит токен доступа и токен обновления, то есть для пользователя все происходит незаметно и ему не надо заново выполнять все запросы с начала.

Рисунок 4 – Схема работы REFRESH TOKEN

Для запуска сервиса безопасности KEYCLOAK необходимо его развернуть. Для этого скачаем его с официального сайта https://www.keycloak.org и выполним все инструкции для установки (рисунок 5).

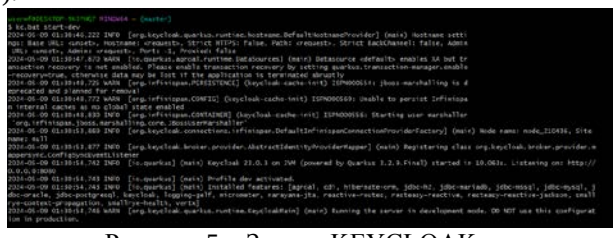


Рисунок 5 – Запуск KEYCLOAK

Далее необходимо запустить KEYCLOAK, оставим порт сервисе дефолтный, в данном случае 8080. Команда для запуска – kc.bat start-dev.

Прежде чем переходить к настройке для нашей системы, необходимо выделить ряд абстракций и процессов аутентификации и авторизации в сервисах, представленных на архитектурной схеме. Frontend часть Видеосервиса – пользователь при переходе в систему должен представиться. Frontend Видеосервиса должен получить токен доступа, если пользователь прошел аутентификацию.

Сначала необходимо создать так называемый Рилм (REALM) – некая абстракция/область где мы управляем пользователями, их учетными данными, группами, ролями, параметрами токенов и т.д.

Далее необходимо создать Клиента (CLIENT) для нашего Frontend Видеосервиса с типом получения PKCE. В контексте KEYCLOAK клиентом у нас выступают приложения, которые используют KEYCLOAK в качестве сервера безопасности и авторизации.

Создадим тестового пользователя данного клиента, то есть, когда мы будем переходить на фронтенд часть видеосервиса, нам будет всплывать окно аутентификации и авторизации KEYCLOAK с содержанием в URL нашего Рилма и Клиента, чтобы ее пройти необходимо зайти под тем пользователем, которого мы в нем и заведем (рисунок 6). Сейчас мы это сделаем руками, но в перспективе при передаче решения, администраторы KEYCLOAK смогут воспользоваться функцией маппинга пользователя из Active Directory или других источников. Таким образом KEYCLOAK будет автоматически заводить у себя пользователей с такими же параметрами, как в хранилищах. Но сейчас мы сделаем это вручную.

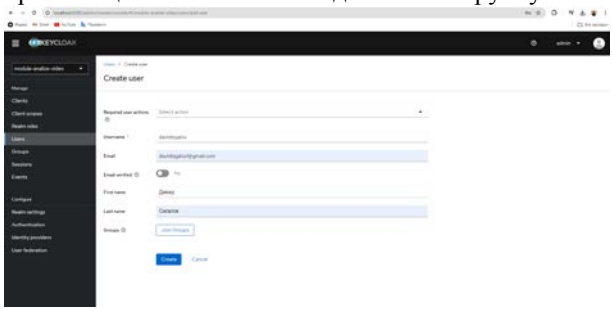


Рисунок 6 – Создание пользователя

После того как мы создали пользователя необходимо добавить ему роль.

Роли в нашем сервисе безопасности KEYCLOAK позвонят нам определить уровень доступа для конкретных пользователей. Например, для добавления видео, мы поставим ограничение только для пользователей имеющих роль “user”. Далее в токене доступа мы увидим информацию о пользователе и роли, которая ему присвоена.

Далее необходимо присвоить эту роль нашему созданному пользователю (рисунок 7).

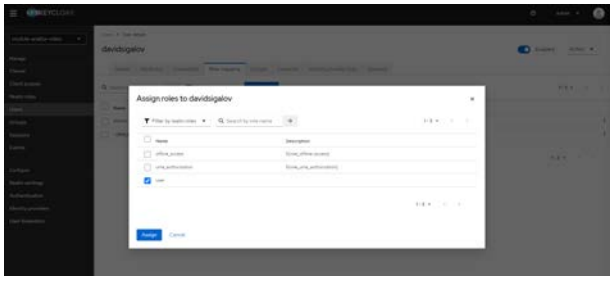


Рисунок 7 – Присвоение роли

Для реализации микросервиса анализа видеопотока будем использовать следующие технологии:

1) FAST API – веб-фреймворк на Python для создания REST API.

2) Библиотека BaseModel – определение модели запроса.

3) Библиотека CV2 – работа с изображением.

4) Библиотека Minio – работа с S3 хранилищем для сохранения изображений.

Как мы описывали ранее для обращения к сервису анализа качества видеопотока у нас используется REST API, то есть сервису будут приходить HTTP запросы, который в дальнейшем ему будет необходимо обработать.

В тело запроса положим необходимые аттрибуты:

RTSP_LINK – RTSP ссылка на видеопоток

VIDEO_NAME – то есть название созданного пользователем видео

Далее наш сервис должен открыть видеопоток по указанной RTSP ссылке, считать текущий кадр с видеопотока, преобразовать кадр в оттенки серого, на преобразованном кадре серого оттенка выделить границы.

После данных преобразований, каждый из шагов нам необходимо сохранить, для этого программно подключаемся к MINIO. Из байтов преобразованных изображений создаем объекты и сохраняем. Также создаем копию исходного изображения и рисуем на нем выделенные контуры, копируем в JPEG формат и получаем байты, также из байтов создаем объект и сохраняем в MINIO.

Алгоритм сохранения в MINIO происходит следующим образом, мы берем значение из атрибута VIDEO_NAME и при сохранении проверяем нет ли в бакете данной папки, если данной папки нет мы создаем ее, а внутрь данной папки мы кладем еще одну папку с наименованием “VIDEO_NAME” + Время анализа. Далее в созданную папку кладем четыре изображения, которые мы получили преобразованиями (Текущий кадр, Кадр в оттенках серого, Выделенные Границы, Наложенные границы на серый кадр). Если папка VIDEO_NAME уже была, то мы сразу в нее кладем подпапку с текущем времени и в нее изображения. В ответ на запрос необходимо вернуть сообщение о успешном анализе, количество контуров на

изображении, а также ссылку на папку с изображениями.

Начнем с реализации подключения веб-фреймворка и определением данных запроса. Далее, так как S3 хранилище преобразовывает пути в base64 формат, нам необходимо возвращать функцию, которая в ответ отдает путь нормальной формы. То есть ссылка, по которой пользователь может перейти на страницу с изображениями.

Теперь необходимо реализовать POST запрос, в теле которого передается модель данных, которую мы задали (рисунок 8).

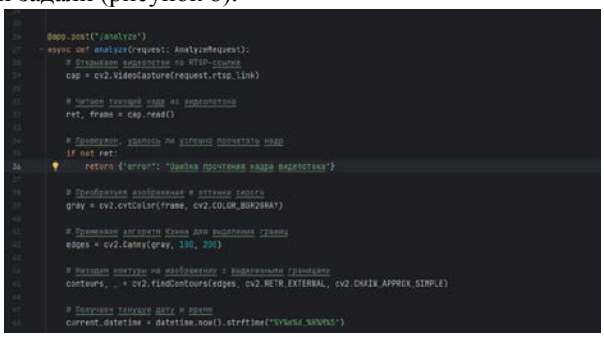


Рисунок 8 – Реализация REST API

В данном блоке кода мы написали обработку данных отправленных в теле запроса.

В первую очередь с помощью функции VideoCapture библиотеки cv2 открываем видеопоток. Далее через read() читаем текущий кадр. Переменная ret указывает удалось ли нам успешно прочитать кадр, frame содержит сам кадр. Если значение ret – false, тогда вернем строку, что во время запроса произошла “Ошибка прочтения кадра видеопотока”. Если true, тогда продолжаем обработку. Далее текущий кадр мы преобразуем в оттенки серого с помощью функции cvtColor. Далее на получившееся серое изображения задаем функцию Canny для выделения границ. Второй и третий параметр функции Алгоритм Кэнни использует два пороговых значения (threshold1 и threshold2) для классификации градиентов интенсивности пикселей как границы или не границы. Данные значения работают следующим образом:

1)Если градиент интенсивности пикселя выше threshold2, то пиксель считается границей.

2)Если градиент интенсивности пикселя ниже threshold1, то пиксель не считается границей.

3)Если градиент интенсивности пикселя находится между threshold1 и threshold2, то пиксель считается границей только в том случае, если он соединен с пикселем, который уже был отмечен как граница.

Значения threshold1 и threshold2 влияют на чувствительность алгоритма к границам:

1)Более низкие значения threshold1 и threshold2 приведут к обнаружению большего количества границ, включая слабые и шумные границы. Это может привести к большему количеству деталей, но также и к большему количеству ложных срабатываний.

2)Более высокие значения threshold1 и threshold2 приведут к обнаружению меньшего количества границ, включая только сильные и четкие границы. Это может привести к меньшему количеству деталей, но также и к меньшему количеству ложных срабатываний.

Обычно значение threshold2 выбирается в 2-3 раза больше, чем threshold1. Выбор оптимальных значений порогов зависит от конкретного изображения и желаемого результата.

Мы будем использовать значения для threshold1 – 100 и для threshold2 – 200, так как они обеспечивают хороший баланс между сохранением важных границ и подавлением шума на многих изображениях.

Далее с помощью функции findContours находим контуры на изображении с выделенными границами. Можно заменить, что в параметры функцию мы передаем RETR_EXTERNAL и CHAIN_APPROX_SIMPLE.

RETR_EXTERNAL извлекает только внешние контуры, игнорируя любые дочерние контуры, то есть контуры внутри других контуров. RETR_EXTERNAL упрощает результат и уменьшает количество контуров, которые нужно обрабатывать, так как нас интересуют только внешние границы объектов.

HAIN_APPROX_SIMPLE метод аппроксимации контуров, который сжимает горизонтальные, вертикальные и диагональные сегменты контура и оставляет только их конечные точки. Данная функция возвращает только основные точки контура, удаляя избыточные точки на прямых линиях. Тем самым мы значительно уменьшаем количество точек в представлении контура без потери его основной формы.

В сочетании RETR_EXTERNAL и CHAIN_APPROX_SIMPLE обеспечивает эффективное извлечение внешних контуров объектов с упрощенным представлением контуров. Таким образом мы получаем внешние границы объектов, а не их внутренняя структура, и когда желательно уменьшить сложность представления контура для дальнейшей обработки.

Далее считаем текущую дату и время, которую мы будем использовать для названия папки с изображениями.

Далее настраиваем подключение к MINIO (рисунок 9). Для этого указываем параметры подключения. Если бакета, указанного в параметрах не существует, то необходимо его создать. Также задаем наименования папки куда будут сохраняться преобразованные изображения.

Рисунок 9 – Подключение в MINIO

Далее создаем функцию отправки преобразованных изображений в MINIO. То есть в созданную папку мы кладем 4 изображения: Текущее, Оттенки серого, Изображение с выделенными границами и Серое изображение с выделенными границами, полученную с помощью функции drawCounters, в параметры которого передаем текущее изображение, контуры, параметр “-1”, указывающий, что будут нарисованы все контуры из списка contours, цвет контуров, которые мы накладываем, в нашем случае это зеленый цвет, а также ширину контуров в пикселях.

После успешного сохранения необходимо вернуть ответ на запрос, в котором как мы описывали ранее содержится сообщение о успешном анализе, количество контуров на изображении, а также ссылку на папку с изображениями.

Далее необходимо проверить разработанный модуль. Для этого воспользуемся инструментом POSTMAN для отправки API. Сымитируем отправку API с телом запроса и проверим ответ.

Как мы можем убедиться, в ответ к нам пришли ожидаемые параметры, которые говорят нам о том, что запрос прошел успешно. Убедиться в этом нам поможет MINIO куда должны были добавиться изображения (рисунок 10).

Рисунок 10 – Проверка сохранения изображений в MINIO

Как мы можем увидеть (рисунок 10) в MINIO появилась папка test, которая соответствует значению из аттрибута “video_name”. В папке test мы можем увидеть папку с названием “test_20240512_011517 ”, которая соответствует дате и времени выполнения запроса и в ней 4 изображения. Посмотрим эти изображения для того, чтобы убедиться в корректности отработки программы. Для удобного просмотра соединим полученные изображения в ряд (рисунок 11).

Рисунок 11 – Получившиеся изображения

Далее необходимо посчитать показатели отклонения.

В контексте анализа видеопотока каждый кадр будет выдавать количество контуров. Увеличение и снижение количества может нам давать вывод о том, что на кадре что-то изменилось. Например, появился новый живой или неживой объект или наоборот какой-то из объектов, который давал n количество контуров был убран с того места, где проходит трансляция видеопотока. Независимо от места на котором мы проводим трансляцию каждый кадр будет иметь неуникальное значение контуров, то есть если взять два разных строительных объекта где мы проводим трансляцию разница в контурах кадра видеопотоков может быть очень большая, поэтому отклонение от количества кадров будет в процентом значении, а не числовом. Нам необходимо расчет значения в меньшую сторону, то есть, когда количество контуров на кадре видеопотока стало меньше, чем на предыдущем кадре.

Для поиска значения в сторону уменьшения нам первоначально необходимо выявить сценарии при которых может произойти уменьшение контуров на кадрах видеопотока (рисунок 12).

Сценарии: отключение камеры и искажение видеопотока внешними условиями.

1)Отключение или неработоспособность камеры. Обработка ситуации, когда RTSP камера не работает или не подключена, может зависеть как от реализации веб-плеера, так и от возможностей и настроек самой камеры. Нам необходимо посчитать количество контуров на неработающих трансляций и вывить максимальное количество контуров при подобной ошибке.

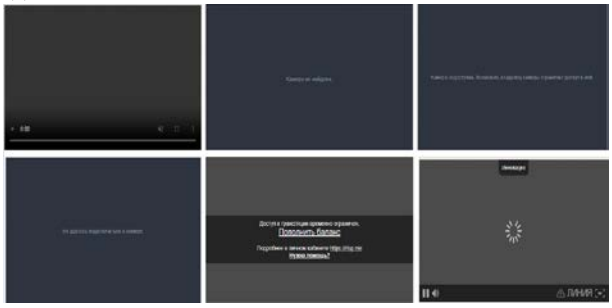


Рисунок 12 – Трансляции с неработающих камер

Выделим на данных примерах контуры и посчитаем их количество. Возьмем изображение с самый большим количеством контуров, так как именно это значение будет считать за границу по которой мы сможем сделать вывод о том, что трансляцию необходимо проверить.

Количество контуров на проблемных камерах был равен соответственно: 21, 22, 90, 54, 115, 41. Максимальное значение контуров среды них равно 115. Данное значение берем как порог ниже которого будем сообщать о проблеме с видеопотоком.

2) Искажение видеопотока внешними условиями.

Так как доступа к погодным условиям у нас нет, выделим несколько корректных кадров видеопотока и смоделируем ситуацию искажения. Возьмем качественный кадр с площадки (рисунок 13).

Рисунок 13 – Кадр для исследования

Далее напишем программу, выполнение которого будет содержать множество размытых изображений. Из них определим изображения, где степень размытости настолько велико, что детали становятся неразличимы.

Из получившийся списка изображений выберем максимально не размытую и определим количество контуров. Получившееся число будет границей, ниже которого мы точно можем сделать вывод о том, что с видеопотоком что-то не так. Считаем количество контуров. Воспользуемся алгоритмом, который мы писали ранее для обнаружения границ и сравним с показателем качественного кадра (рисунок 14).

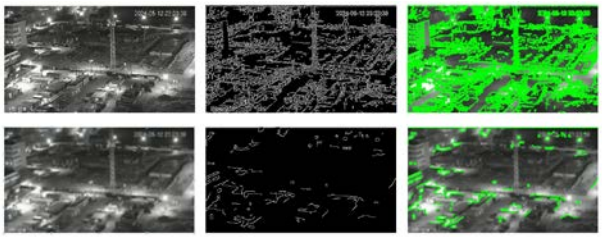


Рисунок 14 – Сравнение кадров

Как мы можем заметить, алгоритм выделения границ выявил гораздо меньше контуров на размытом изображении по сравнению с исходным.

Количество границ, которое мы получили у исходно равно 1604, а у размытого 124. При этом стоит учитывать, что мы выбрали максимально не размытое изображение, соответственно, количество контуров у более размытых изображений будет меньше.

Далее подобную процедуру проведем с 4 кадрами разных видеопотоков на разных камерах строительных площадок. Для каждой камеры построим графики, где y – количество контуров, t – время и сымитируем ситуацию, когда на кадре происходит искажение и далее ошибка в работе камеры. В начале представляем, что видеопоток показывает корректные результаты, далее на 2 секунде идет искажение видеопотока из-за каких-то внешних условий и уже на 3 отключение камеры. Построим общий график для 5 рассмотренных камер (рисунок 15).

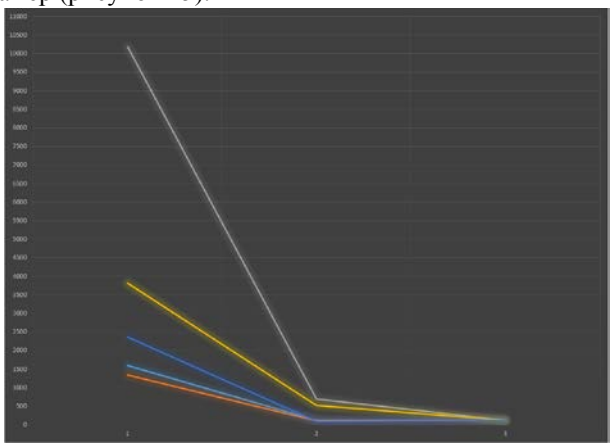


Рисунок 15 – Общий график камер

Теперь мы можем рассчитать процент отклонения. То есть процент уменьшения, при котором мы можем сделать вывод о том, что трансляцию необходимо проверить на качество. Для расчета данного значения нам необходимо:

1) Вычислить процент уменьшения количества контуров для каждой камеры при переходе от нормального значения к искаженному: (нормальное значение - искаженное значение) / нормальное значение * 100%). Если значение в количестве контуров искаженного изображение менее значения неработающей камеры, то значение получаем формуле: (нормальное значение – значение неработающей камеры) / нормальное значение * 100%.

2) Найти среднее арифметическое значение процентов уменьшения по всем камерам.

3) Округлить полученное среднее значение до целого числа.

1 камера: (1604 - 124) / 1604 * 100% = 92.27%
2 камера: (1344 - 115) / 1344 * 100% = 91.44%
3 камера: (10189 - 697) / 10189 * 100% = 93.16%
4 камера: (3812 - 521) / 3812 * 100% = 86.33%
5 камера: (2361 - 115) / 2361 * 100% = 95.13%

4) Найдем среднее значение процентов уменьшения по всем камерам: (92.27% + 91.44% + 93.16% + 86.33% + 95.13%) / 5 = 91.67%

5) Округлим полученное среднее значение до целого числа: 91.67% ≈ 92%

На основании проведенного опыта, процент отклонения количества контуров для основания принятия решение о проблеме с видеопотоком равен 92 процентам. Далее нам необходимо реализовать Backend видеосервиса, который будет контролировать количество контуров каждого кадра и реагировать на изменение в количество контуров на выявленный процент.

Реализация Backend Видеосервиса состоит из основных частей:

1) Сохранение в БД информации о пользователе, который прошел авторизацию.

2) Сохранение в БД добавленных пользователем камер видеонаблюдения.

3) Отправка сервису анализа кадров RTSP ссылки.

4) Сохранение информации о количестве кадров камеры видеонаблюдения.

5) Сравнение количества контуров на каждом кадре добавленной камеры и выявление проблемы при отклонение в количестве контуров более 92%.

6) Отправка уведомления пользователю по почте при обнаружении проблемы с прикреплением ссылки на изображения на основании которых мы сделали вывод об этой проблеме.

Конфигурация Backend Видеосервиса

Spring Web – для обработки HTTP запросов

PostgreSQL – база данных для хранения данных

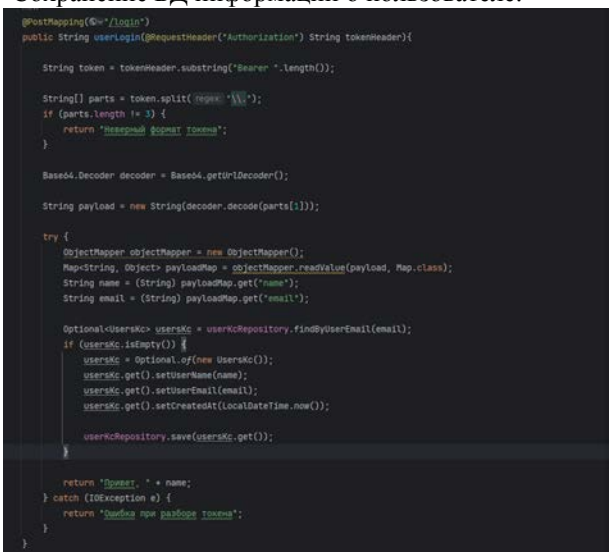
Spring Data JPA – взаимодействие с БД PostgreSQL

PostgreSQL Driver — JDBC драйве, необходимый для подключения приложения к БД PostgreSQL

Lombok – для минимизации шаблонного кода, так как геттеры, сеттеры, хеш-коды и т.д.

Spring Mail – для отправки писем по почте.

Сохранение БД информации о пользователе.

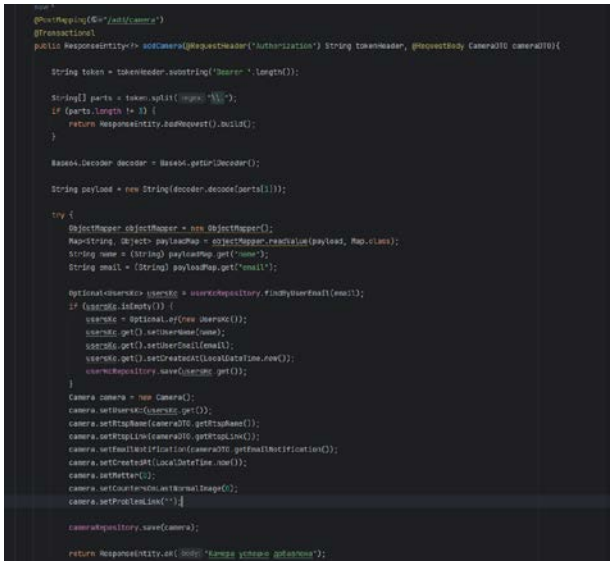


```
@PostMapping("/login")
public String userLogin(@RequestHeader("Authorization") String tokenHeader){

    String token = tokenHeader.substring("Bearer ".length());

    String[] parts = token.split( regex: "\\.");
    if (parts.length != 3) {
        return "Неверный формат токена";
    }

    Base64.Decoder decoder = Base64.getUrlDecoder();

    String payload = new String(decoder.decode(parts[1]));

    try {
        ObjectMapper objectMapper = new ObjectMapper();
        Map<String, Object> payloadMap = objectMapper.readValue(payload, Map.class);
        String name = (String) payloadMap.get("name");
        String email = (String) payloadMap.get("email");

        Optional<UsersKc> usersKc = userKcRepository.findByUserEmail(email);
        if (usersKc.isEmpty()) {
            usersKc = Optional.of(new UsersKc());
            usersKc.get().setUserName(name);
            usersKc.get().setUserEmail(email);
            usersKc.get().setCreatedAt(LocalDateTime.now());

            userKcRepository.save(usersKc.get());
        }

        return "Привет, " + name;
    } catch (IOException e) {
        return "Ошибка при разборе токена";
    }
}
```

Рисунок 16 – Добавление пользователя в БД

В данном блоке (рисунок 16) мы получаем запрос от пользователя при переходе на стартовую страницу. Так как в заголовке мы получаем Токен Доступа, мы можем взять о пользователе информацию. Нам необходимо декодировать токен доступа и получить данные пользователя. Берем пользовательское имя и почту. Далее заходим в базу данных и ищем есть ли такая запись в БД. Если такой записи нет, то сохраняем пользователя. Если такая запись есть, то ничего не сохраняем. В ответ пользователю присылаем сообщение: Привет + name” из декодированного токена доступа.

Когда пользователь добавляет камеру он заполняет следующие поля: rtsp_name – название камеры, rtsp_link – RTSP ссылка, email_notification – почту куда будут отправляться уведомление о некачественном видеопотоке.



```
@PostMapping("/add/camera")
@Transactional
public ResponseEntity<?> addCamera(@RequestHeader("Authorization") String tokenHeader, @RequestBody CameraDTO cameraDTO){

    String token = tokenHeader.substring("Bearer ".length());

    String[] parts = token.split( regex: "\\.");
    if (parts.length != 3) {
        return ResponseEntity.badRequest().build();
    }

    Base64.Decoder decoder = Base64.getUrlDecoder();

    String payload = new String(decoder.decode(parts[1]));

    try {
        ObjectMapper objectMapper = new ObjectMapper();
        Map<String, Object> payloadMap = objectMapper.readValue(payload, Map.class);
        String name = (String) payloadMap.get("name");
        String email = (String) payloadMap.get("email");

        Optional<UsersKc> usersKc = userKcRepository.findByUserEmail(email);
        if (usersKc.isEmpty()) {
            usersKc = Optional.of(new UsersKc());
            usersKc.get().setUserName(name);
            usersKc.get().setUserEmail(email);
            usersKc.get().setCreatedAt(LocalDateTime.now());
            userKcRepository.save(usersKc.get());
        }
        Camera camera = new Camera();
        camera.setUsersKc(usersKc.get());
        camera.setRtspName(cameraDTO.getRtspName());
        camera.setRtspLink(cameraDTO.getRtspLink());
        camera.setEmailNotification(cameraDTO.getEmailNotification());
        camera.setCreatedAt(LocalDateTime.now());
        camera.setMetter(0);
        camera.setCounterOnLastNormalImage(0);
        camera.setProblemLink("");

        cameraRepository.save(camera);

        return ResponseEntity.ok( body: "Камера успешно добавлена");
    }
```

Рисунок 17 – Сохранение информации о добавленной камере

В теле запроса получаем сущность CameraDTO, которая содержит те же поля, которые заполняет пользователь на Frontend части Видеосервиса. Далее мы также проверяем токен доступа. После проверки находим пользователя в таблице хранения данных о пользователе и добавляем камеру. Так как таблица содержит внешний ключ на user_id таблицы о пользователя мы привязываем камеру к тому пользователю, который ее добавил.

После того, как мы сохранили камеру. Мы можем отправлять RTSP ссылку на сервис анализа кадра. Так как нам необходимо отправлять каждый кадр видепотока необходимо реализовать функцию, которая каждую секунду отправляет RTSP ссылку на сервис анализа кадров. Далее количество полученных контуров кадра нам необходимо сравнивать с его предыдущими показателями, таким образом мы будем проверять количество контуров текущего с кадра с количеством контуров предыдущего. Но необходимо определить с каким именно из предыдущих кадров мы должны проводить сравнение. Так как на кадре видеопотоке может на очень короткое время появиться объект, который загородит транслируемую область, но на следующем кадре его уже не будет, имеет смысл сравнивать показатели с определенным интервалом.

В таблице нашей камеры есть поля у каждой камеры:

1) Счетчик камеры (metter) - значение по умолчанию равно 0.

2) Количество контуров последнего нормального кадра камеры (last_counter_image) - значение по умолчанию равно 0.

3) Ссылка на директорию с изображением испорченного кадра и его преобразований (problem_link) – по умолчанию пустая строка.

Описание Алгоритма определения некачественного видеопотока для камеры:

1) Отправляем ссылку на сервис видеопотока, получаем значение количества контуров.
2) Сравниваем количество контуров текущего кадра с количеством контуров предыдущего
3) Если предыдущего кадра нет, то объявляем, что переменная количества контуров последнего нормального значения равна количеству контуров текущего, далее возвращаемся к пункту 1.
4) Если предыдущий кадр существовал, то сравниваем количество контуров последнего нормального кадра с текущим значением контуров.
5) Если видим отклонение на 92%, тогда увеличиваем счетчик камеры на единицу и объявляем директорию с изображением испорченного кадра. Далее возвращаемся к пункту 1.
6) Если отклонения нет или оно не более 92%, тогда меняем переменную количества контуров последнего нормального значения текущего кадра и обнуляем счетчик.
7) Отправка уведомления – сторонний независимый процесс отправки уведомления на почту, когда счетчик достиг значения, у которого процент деления на 5.

Проверка работы алгоритма. Для корректной проверки работы алгоритма необходимо изменить с 92% отклонения до 1%, так как доступа к видеопотоку у нас нет, а уведомление получить надо.

При включении программы пишется в консоль то, что мы определили в выводе алгоритма. Как мы можем заметить на первой итерации не было предыдущего изображения поэтому вывели текст “не хватает кадров для анализа”. На следующих итерациях видим в выводе консоли, что последующие изображения имеют отклонения, что логично так как мы поставили отклонение в 1% (рисунок 18).

Рисунок 18 – Вывод информации по обработке алгоритма в консоли

В конце мы можем заметить в консоли текст “Отправляем уведомление”. Это означает, что счетчик камеры дошел до 5 и нам должно было прийти уведомление на почту, которую мы указали при создании камеры (рисунок 19).

Рисунок 19 – Получение уведомления пользователю

IV. ЗАКЛЮЧЕНИЕ

Разработаны архитектура, функционал и интерфейсы программного обеспечения для системы непрерывного анализа качества трансляции с камер видеонаблюдения на строительных площадках через алгоритм детектирования Кэнни.

В данной работе был произведен расчет критического процента отклонения и алгоритма его определения, также реализован модуль анализа качества видеопотока на строительной площадке, с возможностью добавления видеопотоков пользователем через визуальную часть с защитой на основе токена доступа и непосредственного анализа сразу после добавления через созданный алгоритм сравнения количества контуров на каждом кадре видеопотока методом выделения границ оператором Кэнни с сохранением кадров и его преобразований в хранилище данных и отправки уведомления пользователю с наличием ссылки на начальный проблемный кадр.

БЛАГОДАРНОСТИ

Авторы выражают благодарность Высшей инжиниринговой школе НИЯУ МИФИ за помощь в возможности опубликовать результаты выполненной работы.

БИБЛИОГРАФИЯ

[1] Дакетт Д. HTML и CSS. Разработка и дизайн веб-сайтов / Д. Дакетт. - Москва: Эксмо, 2013. - 480 с.
[2] Дакетт Д. Javascript и jQuery. Интерактивная веб-разработка / Д. Дакетт. - Москва: Эксмо, 2020. - 640 с.
[3] Лутц М. Изучаем Python. 3-е издание / М. Лутц. - Санкт-Петербург: Символ-Плюс, 2009. – 548 с.
[4] Сурарес Д.О. Обработка изображений с помощью OpenCV / Д.О. Суарес, Гарсия Г.Б. - Москва: ДМК Пресс, 2016. - 210 с.
[5] Эккель Б. Философия Java / Б. Эккель. - Санкт-Петербург: Питер, 2022. – 1168 с.
[6] Adeshina A.A. Building Python Web APIs with FastAPI: A fast-paced guide to building high-performance, robust web APIs with very little boilerplate code / A.A. Adeshina. - Mumbai: Packt Publishing Ltd, 2022. - 187 p.
[7] Debalauwe W. Taming Thymeleaf. Practical Guide to building a web application with Spring Boot and Thymeleaf / W. Debalauwe. - Mumbai: Packt Publishing Ltd, 2022. - 410 p.
[8] Drake J.D. Practical PostgreSQL / J.D. Drake, J.C. Worsley. - New York: O'Reilly Media, Inc., 2022. - 622 p.
[9] Richer J. OAuth 2 in Action First Edition / J. Richer, A. Sanso. - New York: PublisherManning Publications Co. LLC, 2017, 360 p.
[10] Thorgersen S. Keycloak - Identity and Access Management for Modern Applications: Harness the power of Keycloak, OpenID Connect, and OAuth 2.0 protocols to secure applications / I. Silva, S. Thorgersen. - Mumbai: Packt Publishing Ltd, 2021. - 362 p.
[11] Вы кто такие, я вас не знаю, или Как мы делаем JWT-аутентификацию [Электронный ресурс]. – URL: https://habr.com/ru/companies/doubletapp/articles/764424 (дата обращения: 13.02.2024).

[12] K Canny Edge Detection (PKCE) [Электронный ресурс]. – URL: https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html (дата обращения: 13.02.2024).
[13] Ключ подтверждения обмена кодами (PKCE) [Электронный ресурс]. – URL: https://cloudentity.com/developers/basics/oauth-extensions/authorization-code-with-pkce/ (дата обращения: 13.02.2024).
[14] Отправка электронных писем с помощью Spring [Электронный ресурс]. – URL: https://habr.com/ru/companies/otus/articles/557798/ (дата обращения: 13.02.2024).
[15] Приложение Spring Boot CRUD с Thymeleaf [Электронный ресурс]. – URL: https://www.baeldung.com/spring-boot-crud-thymeleaf (дата обращения: 13.02.2024).
[16] Что такое RTSP и зачем он нужен? [Электронный ресурс]. – URL: https://flussonic.ru/blog/news/about-rtsp/ (дата обращения: 13.02.2024).
[17] Find and Draw Contours using OpenCV | Python [Электронный ресурс]. – URL: https://www.geeksforgeeks.org/find-and-draw-contours-using-opencv-python (дата обращения: 13.02.2024).
[18] Image Smoothing using Gaussian Blur in OpenCV Python [Электронный ресурс]. – URL: https://www.tutorialkart.com/opencv/python/opencv-python-gaussian-image-smoothing/#gsc.tab=0 (дата обращения: 13.02.2024).
[19] Lombok. Полное руководство [Электронный ресурс]. – URL: https://habr.com/ru/companies/piter/articles/676394 (дата обращения: 13.02.2024).
[20] MinIO: Connect to MinIO from Python [Электронный ресурс]. – URL: https://www.stackhero.io/en/services/MinIO/documentations/Getting-started/Connect-to-MinIO-from-Python (дата обращения: 13.02.2024).

Creation of a system for continuous analysis of video surveillance camera broadcast quality at construction sites via Canny detection algorithm

D.I. Sigalov, M.G. Zhabitsky

Abstract. Construction sites are equipped with ip cameras operating via RTSP (real time streaming protocol). The aim of the research work is to dive into the Canny detection algorithm, to implement it as one of the microservices of our system, and to develop the rest of the system modules to make decisions about poor quality video streaming. The result of the research work is a system through which interested parties can register RTSP cameras, view the boundary detection performance with each frame of the video stream, and receive notifications when a low-quality broadcast is detected.

Keywords – RTSP, REST, Access Token, JSON, Canny operator, DB, CV2, OAUTH 2.0, KEYCLOAK, S3 MINIO.

REFERENCES

[1] Duckett D. HTML and CSS. Development and design of websites / D. Duckett. - Moscow: Eksmo, 2013. - 480 c.
[2] Duckett D. Javascript and jQuery. Interactive web development / D. Duckett. - Moscow: Eksmo, 2020. - 640 c.
[3] Lutz M. Studying Python. 3rd edition / M. Lutz. - St. Petersburg: Symbol-Plus, 2009. - 548 c.
[4] Surares D.O. Image processing with OpenCV / D.O. Suarez, Garcia G.B. - Moscow: DMK Press, 2016. - 210 c.
[5] Eckel B. Philosophy of Java / B. Eckel. - St. Petersburg: Peter, 2022. - 1168 c.
[6] Adeshina A.A. Building Python Web APIs with FastAPI: A fast-paced guide to building high-performance, robust web APIs with very little boilerplate code / A.A. Adeshina. - Mumbai: Packt Publishing Ltd, 2022. - 187 p.
[7] Debalauwe W. Taming Thymeleaf. Practical Guide to building a web application with Spring Boot and Thymeleaf / W. Debalauwe. - Mumbai: Packt Publishing Ltd, 2022. - 410 p.
[8] Drake J.D. Practical PostgreSQL / J.D. Drake, J.C. Worsley. - New York: O'Reilly Media, Inc., 2022. - 622 p.
[9] Richer J. OAuth 2 in Action First Edition / J. Richer, A. Sanso. - New York: PublisherManning Publications Co. LLC, 2017, 360 p.
[10] Thorgersen S. Keycloak - Identity and Access Management for Modern Applications : Harness the power of Keycloak, OpenID Connect, and OAuth 2.0 protocols to secure applications / I. Silva, S. Thorgersen. Silva, S. Thorgersen. - Mumbai: Packt Publishing Ltd, 2021. - 362 p.
[11] Who Are You, I Don't Know You, or How We Do JWT Authentication [Electronic resource]. - URL: https://habr.com/ru/companies/doubletapp/articles/764424 (accessed 13.02.2024).
[12] Canny Edge Detection (PKCE) [Electronic resource]. - URL: https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html (accessed 13.02.2024).
[13] Code Exchange Confirmation Key (PKCE) [Electronic resource]. - URL: https://cloudentity.com/developers/basics/oauth-extensions/authorization-code-with-pkce/ (accessed 13.02.2024).
[14] Sending emails with Spring [Electronic resource]. - URL: https://habr.com/ru/companies/otus/articles/557798/ (accessed 13.02.2024).
[15] Spring Boot CRUD application with Thymeleaf [Electronic resource]. - URL: https://www.baeldung.com/spring-boot-crud-thymeleaf (accessed 13.02.2024).
[16] What is RTSP and why is it needed? [Electronic resource]. - URL: https://flussonic.ru/blog/news/about-rtsp/ (accessed 13.02.2024).
[17] Find and Draw Contours using OpenCV | Python [Electronic resource]. - URL: https://www.geeksforgeeks.org/find-and-draw-contours-using-opencv-python (accessed 13.02.2024).
[18] Image Smoothing using Gaussian Blur in OpenCV Python [Electronic resource]. - URL: https://www.tutorialkart.com/opencv/python/opencv-python-gaussian-image-smoothing/#gsc.tab=0 (date of address: 13.02.2024).
[19] Lombok. Complete manual [Electronic resource]. - URL: https://habr.com/ru/companies/piter/articles/676394 (accessed 13.02.2024).
[20] MinIO: Connect to MinIO from Python [Electronic resource]. - URL: https://www.stackhero.io/en/services/MinIO/documentations/Getting-started/Connect-to-MinIO-from-Python (accessed 13.02.2024).