СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время на предприятиях существует потребность в учете рабочего времени персонала в рабочих помещениях на предприятии для контроля рабочего времени и передвижения сотрудников и не идентифицированных лиц, а также в сборе статистики о времени, проведённом в различных помещениях предприятия.

Системы учета рабочего времени позволяют проанализировать занятость персонала при выполнении рабочих функций в течение рабочего времени, результаты анализа помогают руководству предприятия оптимизировать некоторые рабочие процессы, что повышает производительность труда сотрудников. Контроль передвижений персонала позволяет проанализировать внутреннюю логистика, что приводит к сокращению времени на перемещение как продукции в процессе изготовления на территории предприятия, так и самого персонала, дополнительно система позволит контролировать доступ персонала в различные помещения объекты предприятия.

Целью дипломного проекта является разработка автоматической системы учёта персонала в помещении с использованием технологии распознавания лиц.

Для достижения поставленной цели дипломного проекта необходимо решить следующие задачи:

– изучить методы распознавания лиц, в особенности свёрточные нейронные сети;

– выбрать средства разработки системы;

– выбрать оборудование и программное обеспечение для развёртывания системы;

разработать программный код для обработки лиц и веб-интерфейса системы;

– сконфигурировать параметры безопасности системы;

– спроектировать варианты масштабируемости;

– реализовать сбор статистики;

– разработать механизм сбора отчётов и интеграции системы.

– рассчитать экономические затраты на разработку, а также отпускную цену продажи системы.

Дипломный проект выполнен самостоятельно, проверен в системе «Антиплагиат». Процент оригинальности соответствует норме, установленной кафедрой.

1. ОПИСАНИЕ ОБЪЕКТА УПРАВЛЕНИЯ

1.1 Системы и методы распознавания лиц

Распознавание лиц – практическое применение теории распознавания образов, задача которого состоит в автоматической локализации лица на изображении, а также в идентификации персоны по лицу [1]. Поэтому для начала необходимо понимание теории распознавания образов.

Сама теория распознавания образов – это раздел информатики и смежных дисциплин, развивающий основы и методы классификации и идентификации предметов, явлений, процессов, сигналов, ситуаций и т. п. объектов, которые характеризуются конечным набором некоторых свойств и признаков. Необходимость в таком распознавании возникает в самых разных областях - от военного дела и систем безопасности до оцифровки аналоговых сигналов. Проблема распознавания образов приобрела выдающееся значение в условиях информационных перегрузок, когда человек не справляется с линейно-последовательным пониманием поступающих к нему сообщений, в результате чего его мозг переключается на режим одновременности восприятия и мышления, которому свойственно такое распознавание[2].

Алгоритмы распознавания образов зависят от типа вывода метки, от того, является ли обучение контролируемым или неконтролируемым, а также от того, является ли алгоритм статистическим или нестатистическим по своей природе. Статистические алгоритмы можно далее разделить на генеративные и дискриминационные.

Алгоритмы можно разделить на несколько множеств, характеризующиеся различными методами:

– задача классификации, который в свою очередь можно разделить на параметрический(пример: линейный дискриминантный анализ) и не параметрический(пример: свёрточные нейронные сети);

– кластерный анализ(пример: корреляционная кластеризация);

– ансамблевое обучение(пример: усреднение по ансамблю);

– регрессионный анализ(пример: анализ главных компонентов).

Примеры систем:

– «Amazon Rekognition»: это сервис обработки изображений и видео от «Amazon Web Services». Он предоставляет API для распознавания лиц, а также для анализа эмоций, определения пола и возраста, идентификации знаменитостей и других функций.

– «Face++»: это платформа и API для распознавания лиц, разработанная китайской компанией «Megvii». Она обладает высокой точностью и может

использоваться для идентификации лиц, анализа эмоций, определения возраста и пола и других задач.

– «Microsoft Azure Face API»: это облачный сервис от «Microsoft», предоставляющий API для распознавания лиц. Он позволяет идентифицировать лица на фотографиях и в видео, а также проводить анализ эмоций, определять возраст и пол и выполнять другие задачи.

– «Google Cloud Vision API»: это сервис от «Google», который включает в себя функции распознавания лиц. Он может определять лица на изображениях и проводить анализ эмоций, определять возраст и пол и выполнять другие задачи с использованием машинного обучения.

– «OpenFace»: это открытое программное обеспечение для распознавания лиц, разработанное компанией «Carnegie Mellon University». Оно предоставляет набор инструментов и библиотек для обнаружения и идентификации лиц на изображениях и в видео.

– «Kairos»: это платформа для распознавания лиц, которая предоставляет API для идентификации лиц, анализа эмоций, определения возраста и пола и других функций. Она может использоваться в различных отраслях, включая безопасность, маркетинг и развлечения.

– «IBM Watson Visual Recognition»: это сервис от «IBM», который включает в себя функции распознавания лиц. Он может определять лица на изображениях и в видео, а также проводить анализ эмоций, определять возраст и пол и выполнять другие задачи с использованием искусственного интеллекта.

– «FaceID»: это система распознавания лиц, разработанная компанией «Apple». Она используется для разблокировки устройств, авторизации платежей и других задач, связанных с идентификацией лиц.

Исходя из требований технического задания был выбран способ распознавания лиц с помощью нейронных сетей, а именно свёрточных нейронных сетей.

1.2 Нейронные сети и их виды

Нейронные сети – это ветвь моделей машинного обучения, которые построены с использованием принципов нейронной организации, открытых соединениях в биологических нейронных сетях, составляющих мозг животных [3].

Нейронный сети основаны на наборе связанных единиц или узлов, называемых искусственными нейронами, которые в общих чертах моделируют нейроны биологического мозга. Каждое соединение, подобно

синапсам в биологическом мозге, может передавать сигнал другим нейронам. Искусственный нейрон получает сигналы, затем обрабатывает их и может передавать сигналы подключенным к нему нейронам. «Сигналом» соединения является действительное число, а выходной сигнал каждого нейрона вычисляется с помощью некоторой нелинейной функции суммы его входных сигналов. Нейроны и ребра обычно имеют вес, который корректируется по мере обучения. Вес увеличивает или уменьшает силу сигнала при соединении. Нейроны могут иметь порог, при котором сигнал отправляется только в том случае, если совокупный сигнал пересекает этот порог.

Математически нейрон представляет собой взвешенный сумматор, единственный выход которого определяется через его входы и матрицу весов следующим образом:

$$y = f(u), \text{где } u = \sum_{i=1}^{n} \omega_i x_i + \omega_0 x_0$$

Здесь x_i и ω_i – соответственно сигналы на входах нейрона и веса входов, функция называется индуцированным локальным полем, а $f(u)$ является передаточной функцией, определяющая зависимость сигнала на выходе нейрона от взвешенной суммы сигналов на его входах. Возможные значения сигналов на входах нейрона считают заданными в интервале [0,1]. Они могут быть либо дискретными (0 или 1), либо аналоговыми. Дополнительный вход x_0 и соответствующий ему вес ω_0 используются для инициализации нейрона.

На рисунке 1 представлена схема искусственного нейрона, состоящая из следующих частей: 1 – нейроны, выходные сигналы которых поступают на вход; 2 – сумматор входных сигналов; 3 – вычислитель передаточной функции; 4 – нейроны, на входы которых подаётся выходной сигнал; ω_i – веса входных сигналов.

Обычно нейроны объединяются в слои. Разные слои могут выполнять разные преобразования на своих входах. Сигналы передаются от первого слоя (входной слой) к последнему слою (выходной слой), возможно, после многократного прохождения слоев.

Нейронные сети обучаются (или обучаются) путем обработки примеров, каждый из которых содержит известные «входные данные» и «результат», образуя между ними взвешенные по вероятности ассоциации, которые хранятся в структуре данных самой сети. Обучение нейронной сети по заданному примеру обычно проводится путем определения разницы между

обработанным выходным сигналом сети (часто прогнозом) и целевым выходным сигналом.

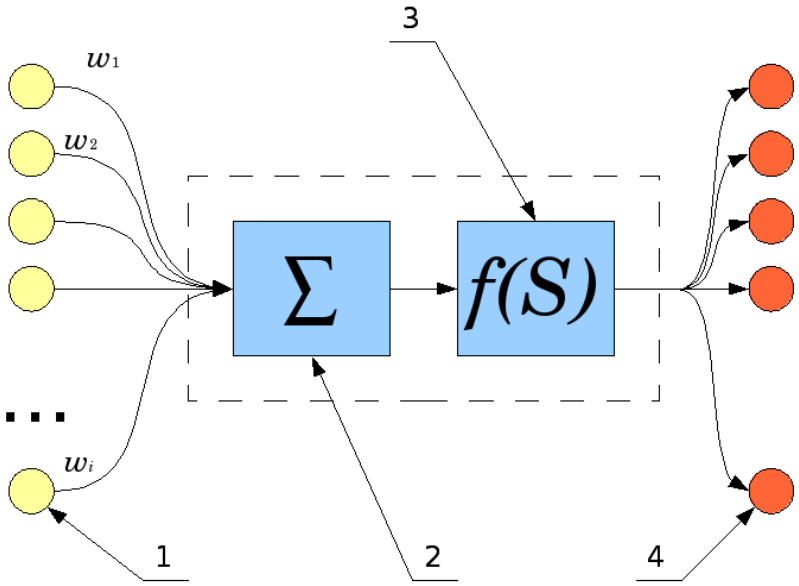


Рисунок 1 – Схема искусственного нейрона

Эта разница и есть ошибка. Затем сеть корректирует свои взвешенные ассоциации в соответствии с правилом обучения и использованием этого значения ошибки. Последовательные корректировки заставят нейронную сеть выдавать выходные данные, которые все больше похожи на целевые выходные данные. После достаточного количества таких корректировок обучение может быть прекращено по определенным критериям. Это форма контролируемого обучения .

Классификация нейронных сетей по характеру связей:

– прямого распространения – характеризуются направлением потока информации между ее слоями. Его поток является однонаправленным, что означает, что информация в модели течет только в одном направлении – вперед – от входных узлов через скрытые узлы (если таковые имеются) и к выходным узлам без каких-либо циклов или петель. В отличие от рекуррентных нейронных сетей, которые имеют двунаправленный поток. Примером сети прямого распространения являются свёрточные нейронные сети.

– радиально-базисных функций – использует радиальные базисные функции в качестве функций активации. Выход сети представляет собой линейную комбинацию радиальных базисных функций входов и параметров нейрона.

– глубокая сеть доверия - один из типов глубинных нейронных сетей, состоящая из нескольких скрытых слоев, в которых нейроны внутри одного слоя не связаны друг с другом, но связаны с нейронами соседнего слоя.

– рекуррентные – характеризующиеся направлением потока информации между ее слоями. В отличие от однонаправленной нейронной сети прямого распространения, это двунаправленная искусственная нейронная сеть, что означает, что она позволяет выходным данным некоторых узлов влиять на последующий ввод в те же узлы.

1.3 Свёрточные нейронные сети в распознавании лиц

Свёрточная нейронная сеть специальная архитектура искусственных нейронных сетей, предложенная Яном Лекуном в 1988 году. Структура сети это однонаправленная (без обратных связей), принципиально многослойная [4].

Работа свёрточной нейронной сети интерпретируется как переход от конкретных особенностей изображения к более абстрактным деталям, и далее к ещё более абстрактным деталям вплоть до выделения понятий высокого уровня. При этом сеть самонастраивается и вырабатывает сама необходимую иерархию абстрактных признаков (последовательности карт признаков), фильтруя маловажные детали и выделяя существенное.

Сверточные сети состоят слоёв, которые разделены на 2 типа:

– слой свёртка;

– слой пулинга.

Слой свёртки – это основной блок свёрточной нейронной сети. Слой свёртки включает в себя для каждого канала свой фильтр, ядро свёртки которого обрабатывает предыдущий слой по фрагментам (суммируя результаты поэлементного произведения для каждого фрагмента). Весовые коэффициенты ядра свёртки (небольшой матрицы) неизвестны и устанавливаются в процессе обучения. Скалярный результат свёртки попадает, а функцию активации, которая представляет собой некую нелинейную функцию.

Слой пулинга (иначе подвыборки, субдискретизации) - представляет собой нелинейное уплотнение карты признаков, при этом группа пикселей (обычно размера 2×2) уплотняется до одного пикселя, проходя нелинейное преобразование. Используются функции максимума, минимума и среднего значения. Преобразования затрагивают непересекающиеся прямоугольники или квадраты, каждый из которых ужимается в один пиксель, при этом выбирается пиксель, имеющий максимальное значение. Операция пулинга

позволяет существенно уменьшить пространственный объём изображения. Пулинг интерпретируется так: если на предыдущей операции свёртки уже были выявлены некоторые признаки, то для дальнейшей обработки настолько подробное изображение уже не нужно, и оно уплотняется до менее подробного. Слой пулинга всегда следует за слоем свёртки.

В свёрточной нейронной сети в операции свёртки используется лишь ограниченная матрица весов небольшого размера, которую «двигают» по всему обрабатываемому слою (в самом начале – непосредственно по входному изображению), формируя после каждого сдвига сигнал активации для нейрона следующего слоя с аналогичной позицией. То есть для различных нейронов выходного слоя используется одна и та же матрица весов, которую также называют ядром свёртки. Её интерпретируют как графическое кодирование какого-либо признака, например, наличие наклонной линии под определённым углом. Тогда следующий слой, получившийся в результате операции свёртки такой матрицей весов, показывает наличие данного признака в обрабатываемом слое и её координаты, формируя так называемую карту признаков. Естественно, в свёрточной нейронной сети набор весов не один, а набор, кодирующая элементы изображения (например линии и дуги под разными углами). При этом такие ядра свёртки не закладываются исследователем заранее, а формируются самостоятельно путём обучения сети классическим методом обратного распространения ошибки. Проход каждым набором весов формирует свой собственный экземпляр карты признаков, делая нейронную сеть многоканальной (много независимых карт признаков на одном слое). При переборе слоя матрицей весов её передвигают обычно не на полный шаг (размер этой матрицы), а на небольшое расстояние. Так, например, при размерности матрицы весов 5×5 её сдвигают на один или два нейрона (пикселя) вместо пяти, чтобы не «перешагнуть» искомый признак.

Типовая сеть состоит из большого количества слоёв. После начального слоя (входного изображения) сигнал проходит серию свёрточных слоёв, в которых чередуется свёртка и субдискретизация (пулинг). Чередование слоёв позволяет составлять «карты признаков», на каждом следующем слое карта уменьшается в размере, но увеличивается количество каналов. На практике это означает способность распознавания сложных иерархий признаков. Обычно после прохождения нескольких слоёв карта признаков вырождается в вектор или даже скаляр, но таких карт признаков возникают сотни. На выходе свёрточных слоёв сети дополнительно устанавливают несколько слоёв полносвязной нейронной сети (перцептрон), на вход которых подаются оконечные карты признаков.

Вот несколько конкретных технологий, которые используют сверточные нейронные сети для распознавания лиц:

– «FaceNet» - это технология, разработанная «Google», которая использует сверточные нейронные сети для создания уникальных векторных представлений лиц. Она позволяет сравнивать и идентифицировать лица на основе этих векторных представлений.

– «DeepFace» - это технология, разработанная «Facebook», которая также использует сверточные нейронные сети для распознавания лиц. Она способна определять и идентифицировать лица на фотографиях с высокой точностью.

– «OpenFace» - это открытая библиотека, которая использует сверточные нейронные сети для распознавания лиц. Она предоставляет возможность извлекать признаки лиц и сравнивать их для идентификации.

– «Dlib» - это библиотека машинного обучения, которая включает в себя реализацию сверточных нейронных сетей для распознавания лиц. Она предоставляет инструменты для обнаружения лиц, извлечения признаков и идентификации.

Вывод: распознавание лиц является практическим приложением теории распознавания образов; существует большой спектр алгоритмов для распознавания образов, которые можно систематизировать по основным методам, лежащим в основе; нейронные сети математическая интерпретация организации и функционирования сетей нервных клеток живого организма; основной составной часть. нейронных сетей является персептрон, принимающий числовые значения умножение на весы каждого из входов, сумма результатов которых передаётся в передаточную функцию; свёрточные нейронные сети состоят из 2 чередующихся слоёв: свёртки и пулинга; свёрточные нейронные сети широко используются для идентификации различных объектов на изображениях.

2. РАЗРАБОТКА СТРУКТУРНЫХ СХЕМ

2.1 Разработка структурной схемы системы

Структурная схема представлена на рисунке 2 и листе графического материала ГУИР.466152.001 ПД.

Структурную схему по функциональности можно разделить на 2 части: кабинетная, серверная.

Кабинетная часть представляет совокупность всех камер системы, а также коммутаторов, объединяющих их в отдельные локальные сети на каждом этаже. Кабинетную часть можно разделить 3 составные части, привязанные к номеру этажа, где расположены камеры. К каждому проходу привязано 2 камеры: на вход и выход из помещения, а также один контроллер двери, который управляет одним замком.

Далее будет представлен список оборудования, т.к. количество замков равно количеству контроллеров двери, то будет указываться только количество контроллеров на кабинет/кабинеты, подразумевая тоже самое количество замков.

Первый этаж состоит из: 18 камер, объединённых одним коммутатором; 9 сетевых контроллеров дверей, объединённых одним коммутатором.

Распределение камер и сетевых контроллеров дверей следующее:

– кабинеты №101–103 содержат по одному проходу, суммарное количество камер – 6, суммарное количество сетевых контроллеров дверей – 3;

– кабинет №104 содержит 2 прохода, суммарное количество камер – 4, суммарное количество сетевых контроллеров дверей – 2;

– кабинет №105 содержит 4 прохода, суммарное количество камер – 8, суммарное количество сетевых контроллеров дверей – 4.

Второй этаж состоит: 16 камер, объединённых одним коммутатором; 8 контроллеров дверей, объединённых одним коммутатором.

Распределение камер и контроллеров следующее:

– кабинет №201 содержит три прохода, суммарное количество камер – 6, суммарное количество сетевых контроллеров дверей – 3;

– кабинеты №202–203 содержит 2 прохода, суммарное количество камер – 8, суммарное количество сетевых контроллеров дверей – 4;

– кабинет №204 содержит 1 проход, суммарное количество камер –2, суммарное количество сетевых контроллеров дверей – 1.

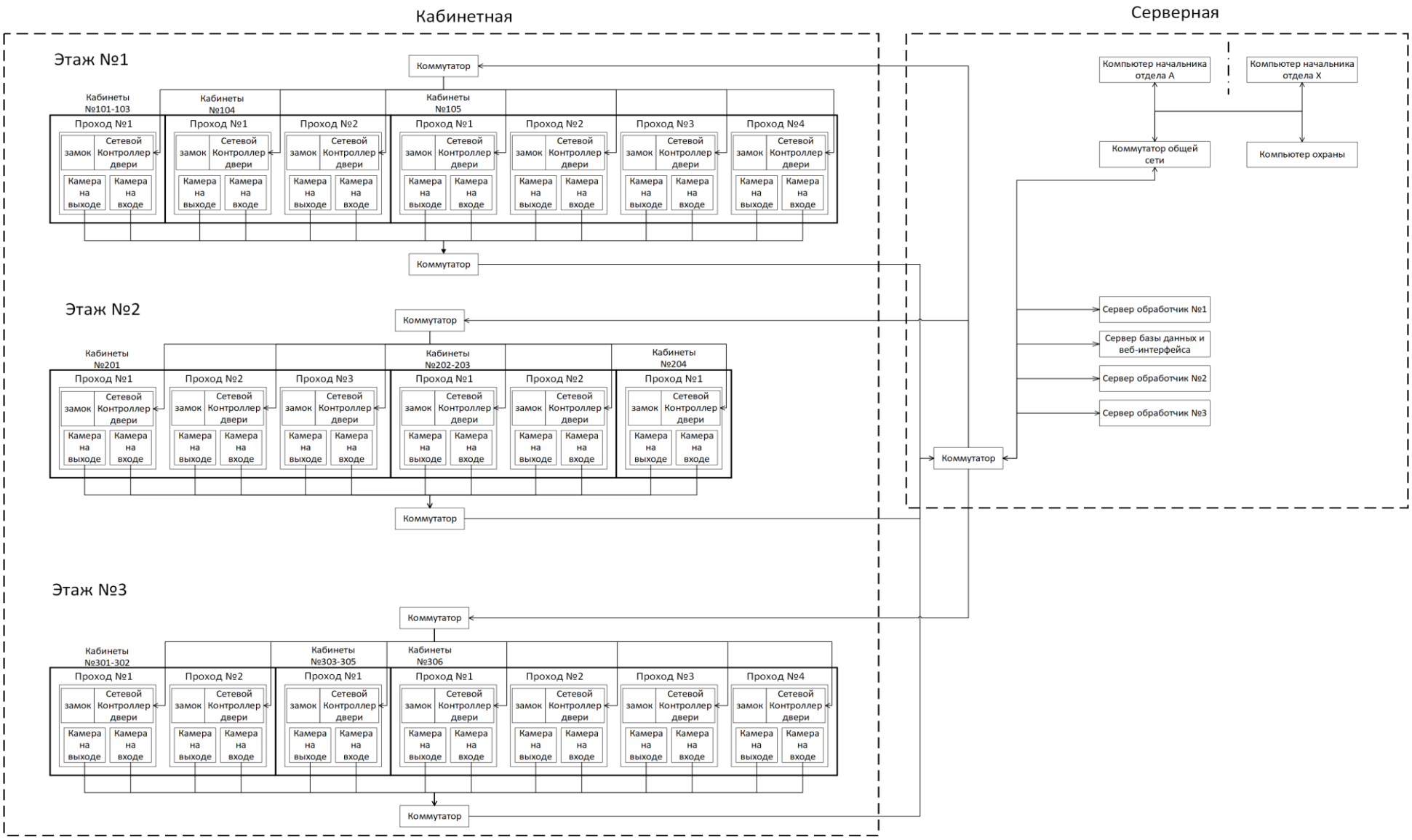


Рисунок 2 – Структурная схема системы

Третья составная часть состоит: 2 камер, объединённых одним коммутатором; 1 контроллеров дверей, объединённых одним коммутатором.

Распределение камер и сетевых контроллеров дверей следующее:

– кабинеты №301–302 содержат два прохода, суммарное количество камер – 8, суммарное количество сетевых контроллеров дверей – 4;

–кабинеты №303–305 содержит 1 проход, суммарное количество камер – 6, суммарное количество сетевых контроллеров дверей – 3;

– кабинет №306 содержит 4 прохода, суммарное количество камер –8, суммарное количество сетевых контроллеров дверей – 4;

Серверная часть представляет совокупность серверов системы, объединённых одним коммутатором. Этот коммутатор предназначен для 3 задач:

– объединения серверов в единую сеть;

– связь серверов с общей сетью предприятия;

–связь с камерами, путём подключения к коммутаторам составных частей камерной части.

Сервера представляют собой следующих 4 сервера:

– сервер базы данных и веб–интерфейса – предназначен для размещения базы данных сервера, а также веб–интерфейса системы;

– сервера обработчики №1–3 – предназначены для размещения программ обработчиков данных с камер.

2.2 Разработка структурной схемы модулей взаимодействия проходов с серверами

Структурная схема модулей взаимодействия проходов с серверами иллюстрирует потоки передачи данных между модулями системы. Она представлена на рисунке Функциональная схема представлена на рисунке 3 и листе графического материала ГУИР.466152.002 ПД.

Схему функционально можно разделить на 3 части:

– сбор и обработка данных;

– хранение и чтение данных;

– управление.

Часть сбора данных и управления представлена совокупностью всех камер системы, отвечающих за сбор визуальных данных, а также сетевых контроллеров дверей, отвечающих за открытие электромеханических замков дверей.

Часть обработки данных представлена тремя серверами, отвечающими за обработку данных с камер. Эти серверы программно представляют собой

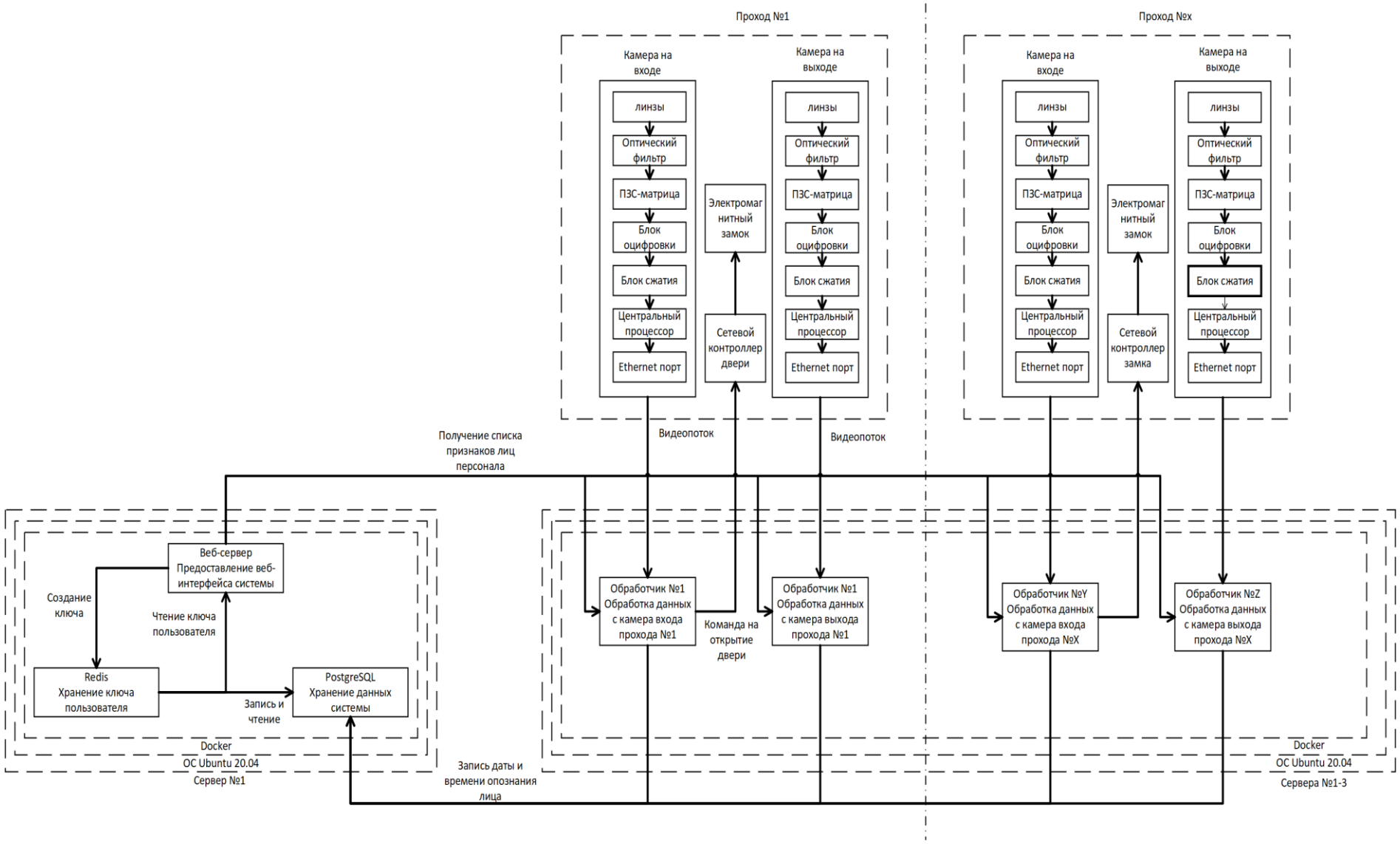


Рисунок 3 – Структурная схема модулей взаимодействия проходов с серверами

операционную систему «Ubuntu 20.04», на которой установлена программа «Docker», осуществляющая поддержку работы с контейнеризацией, а именно разворачиваются контейнеры с программой обработчиком. Каждый такой контейнер отвечает за обработку данных с одной конкретизированной камеры, к которой привязана, а также отправку команды на открытие сетевому контроллеру двери. Каждый контейнер имеет доступ к базе данных системы, в которую и отправляются данные по обнаружению лиц, а также получают вектора признаков лиц персонала.

Часть хранения и чтения данных представлена сервером базы данных и веб–интерфейса. Задача сервера – размещение веб–сервера и баз данных системы. На данном сервере также установлена операционная система «Ubuntu 20.04», как и установлена программа «Docker», где развёрнуты следующие контейнеры:

– контейнер с «PostgreSQL», который является основной базой данных для системы и хранит в себе список персонала вместе с их биометрическими данными, а также список камер, привязанный к списку помещений;

– контейнер с «Redis», который предназначен для хранения временных ключей пользователей, предназначенных для входа в веб–интерфейс;

– контейнер с веб–интерфейсом, который предоставляет доступ данным системы с помощью браузера.

Между 3 вышеперечисленными контейнерами существуют следующие взаимодействия: веб–сервер производит запись и чтение из таблиц базы данных «PostgreSQL»; веб–сервер создаёт ключ пользователя в «Redis», а также производит го чтение оттуда.

Вывод: структурная схема системы представляет перечень всего оборудования задействованного в систем, состоит из: 56 камер, 6 коммутаторов, 28 сетевых контроллеров дверей, 28 электромагнитных замков, 4 сервера; функциональная схема содержит описать ключевых программно–аппаратных элементов системы: обработчики получают данные с камер и из базы данных, куда так же вносят данные по результатам работы; сервер базы данных и веб–интерфейса предоставляет доступ к базе данных и веб–интерфейсу соответственно.

3. ВЫБОР ОБОРУДОВАНИЯ И РАЗРАБОТКА ДИАГРАММЫ РАЗВЁРТЫВАНИЯ

3.1 Выбор оборудования системы

Исходя из структурной и функциональной схем система будет состоять из следующего оборудования:

– 56 камер;

– 6 однотипных коммутаторов для объединения камер на каждом этаже;

– 28 сетевых контроллеров дверей;

– 28 электромагнитных замков;

– 1 сервер базы данных и веб–интерфейса;

– 3 сервер обработчика.

Камеры должны соответствовать нескольким условиям:

– поддержка «RTSP» протокола;

– минимальное разрешение передаваемого видеопотока – 640 на 480 пикселей;

– минимальная частота передаваемого видеопотока – 10 кадров в секунду, с возможностью её регулировки;

– наличие Web–интерфейса, по которому можно производить настройку камеры;

– применение внутри помещения – т.к. офисное помещение расположено внутри здания;

– проводной сетевой интерфейс – для более стабильной передачи данных и высоко уровня безопасности, в сравнении с беспроводным интерфейсом;

– стандартная конструкция – для монтажа на уровне средней высоты лица сотрудников;

– наличие ИК–подсветки – для работы в условиях отключения освещения;

– формат сжатия видео «H.264» – для обеспечения приемлемого качества получаемого изображения.

Исходя из вышеописанных требований выбрана модель камеры – «Hikvision DS–2CD2046G2–IU/SL(C)».

Коммутаторы, необходимые для объединения камер на каждом этаже и объединения сетевых контроллеров дверей должны быть однотипными, должен быть настраиваемым либо управляемы, для настройки работающих

портов подключения, иметь количество портов более 22. Исходя из требования выбрана модель – «Utepo UTP–7224E–POE–L2».

Сетевые контроллеры дверей должны иметь Ethernet интерфейс и SDK, для управления извне, а также возможность подключения и контроля одного электромеханического замка и датчика положения двери. Исходя из требований выбрана модель – «ZKTeco C3–100».

Электромагнитные замки должны обладать силой удержания более 250 кг, работать в диапазоне температур от +5°C до +45°C. Напряжение питания – 12 В. Исходя из требований выбрана модель – «ZKTeco CM–280».

Коммутатор, необходимый для объединения серверов системы, должен обладать схожие требования, как и вышеописанные, кроме количества портов. Так же необходимо чтобы не было возможности поддержки портами подключения одновременно подключений «WAN» и «LAN» типов, чтобы исключить человеческий фактор при монтаже системы. Исходя из требования был выбран маршрутизатор – «D–Link DES–3200–28/C1A».

Сервера–обработчики должны быть однотипными, далее будут изложены требования к одному:

– минимум 24 потоков процессра;

– наличие оперативной памяти типа «DDR4» и объёмом 32 Гб – для обеспечения быстроты работы контейнеров;

– наличие твердотельного накопителя объёмом 500 Гб – для быстроты работы операционной системы.

Исходя из вышеописанных требований была выбрана модель – «MulitOffice 7C137D32S512IMG7».

Для сервера базы данных и веб–интерфейса необходимо соблюдение следующих условий:

– 8 ядер центрального процессора – для работы двух контейнеров, обрабатывающих данные с камер;

– наличие оперативной памяти типа «DDR4» и объёмом 8 Гб – для обеспечения быстроты работы контейнеров;

– наличие твердотельного накопителя объёмом 125 Гб – для быстроты работы операционной системы.

Исходя из вышеописанных требований была выбрана модель – «Jet Office 7i10700D8HD05SD12VGALW50».

3.2 Выбор средств разработки

Выбор языка программирования и выбор библиотек для разработки приложений тесно взаимосвязан.

Существует 3 основных варианта выбора:

– вначале выбираем язык программирования, исходя из требования по разработке (например, высокая производительность или высокая читаемость программного кода языка), а уже потом выбираются библиотеки, необходимые для разработки. В таком случае, мы можем выбрать язык программирования, удовлетворяющий непобедимым техническим требованиям, но на стадии выбора библиотек может оказаться, что под конкретные задачи библиотеки отсутствуют на выбранном языке, поэтому надо разрабатывать нужные механизмы самим, что сильно влияет на время разработки;

– по заданному списку решаемых задач подбираем необходимые библиотеки, написанные под один язык программирования. Однако данный вариант может привести к тому, что разработка проекта на этом языке будет более затратной по времени разработки, чем выбрать другой язык, с менее обширным объемом библиотек. Также проект, реализованный на этом языке, может оказаться более затратным по техническим ресурсам.

– комплексный анализ библиотек и языков; позволяет подобрать оптимальное сочетания языка программирования и библиотек, для приемлемого решения задач, требующих решения в разработке проекта.

Выбор языка программирования, а также библиотек будет исходить из задач, требующих решения для разработки:

– для захвата видеопотока (желательно захват видеопотока по протоколу «RTSP»);

– поиск объектов(лиц) на изображении;

– анализ принадлежности человеку;

– коннектор к базе данных, хранящей список допущенного персонала с их изображения лиц и время ухода и прихода персонала.

Также стоящие перед нами задачи будут требовать высокой производительности от приложения.

Для осуществления захвата экрана и поиска объектов существуют следующие библиотеки:

– библиотека «OpenCV» – разработанный под языки программирования: Python, C++, Java;

– библиотека «SimpleCV» – разработанный для языка Python;

– библиотека «Accord.NET Framework» – разработанный для языка C#.

Для анализа лиц существуют:

– библиотека «dlib» – разработанная для языка программирования C++;

– библиотека «face_recogniton» – разработанная для языка программирования Python;

– библиотека «libfacedetection» – разработанный для языка Python, C++.

Выбор коннекторов к базам данных является несложной задачей, т.к. почти все популярные базы данных имеют коннекторы к множеству языков программирования.

Для достижения максимальной производительности приложения, исходя из представленных выше библиотека и поддерживаемых ими языков, а также, учитывая представленный график (см. рисунок 4) подойдёт C++. Однако, имея проблемы с кроссплатформенностью, а именно, что под каждый тип операционной системы нужно отдельно компилировать программу, то данный язык программирования не подходит. Т.к. у перечисленных библиотек есть другой общий язык программирования – Python, то на нём и будет разрабатываться проект.

Python - высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ[5].

Преимущества языка Python:

– динамическая типизация;

– автоматическое управление памятью;

– обширная база данных библиотек;

– минималистичный синтаксис ядра;

– высокая читаемость кода;

– мультиплатформенность программного кода;

– полностью объектно–ориентированный;

– наличие удобного загрузчика библиотек – pip.

Недостатки языка Python:

– более низка скорость работы, в сравнении с компилируемыми языками;

– более высокое потребление оперативной памяти, в сравнении с компилируемыми языками.

Библиотеки будут выбраны следующие: «OpenCV», «face_recognition» и фреймворк «FastAPI».

Библиотека «OpenCV» – это библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом [6]. Может свободно использоваться в академических и коммерческих целях – распространяется в условиях лицензии «BSD».

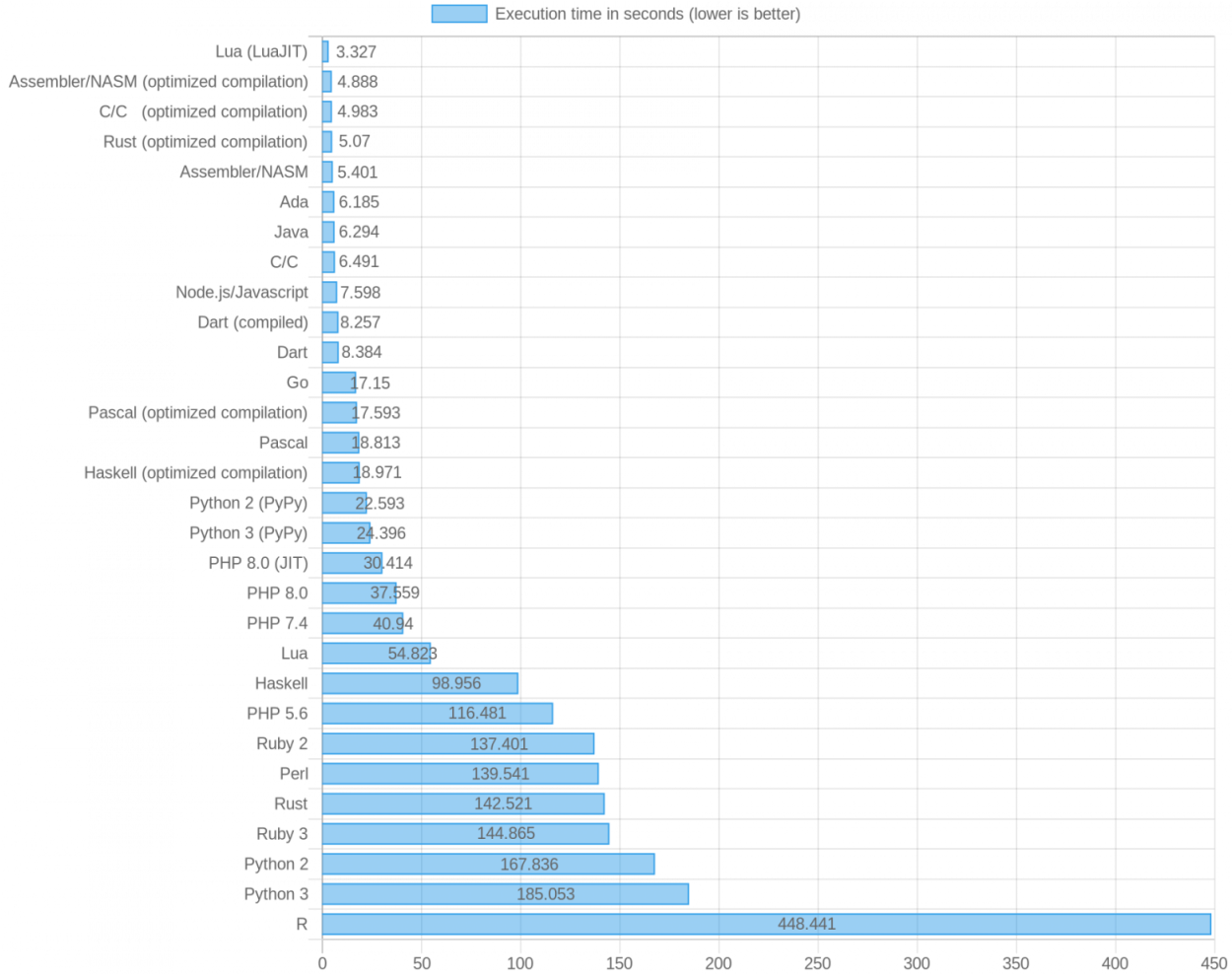


Рисунок 4 – График скорости выполнения теста на основе перебора простых чисел для различных языков программирования

Области применения библиотеки «OpenCV» включает:

– наборы инструментов для 2D– и 3D–функций
– оценка одометрии;
– система распознавания лиц;
– распознавание жестов;
– взаимодействие человека и компьютера (HCI);
– мобильная робототехника;
– понимание движения;
– обнаружение объекта;
– сегментация и распознавание;

– стереозрение «Stereopsis»: восприятие глубины с 2–х камер;

– структура от движения (SFM);

– отслеживание движения;

– дополненная реальность.

Библиотека «Face_recognition» – библиотека, предназначенная для распознавания лиц на изображениях [7]. Основана на базе библиотеки «dlib», созданная с помощью глубокого обучения. Модель обладает точность 99.38% в тесте «Labeled Faces in the Wild benchmark». И обучена на базе наборов: «Labeled Faces in the Wild», «AFAD». Библиотека обладает следующими возможностями:

– поиск лиц на изображениях;

– поиск и манипуляции черт лиц на изображения;

– идентификация лиц на изображениях.

Фреймворк «FastAPI» был выбран для разработки backend части веб–интерфейса, предназначен для создания RESTful API. Особенности:

– генерация документации «OpenAPI»;

– полная поддержка асинхронного программирования;

– валидация данных;

– высокая производительность;

– простота использования.

За отображение веб–интерфейса будет отвечать сочетание языка гипертекстовой разметки «HTML» и каскадных таблиц стилей «CSS».

Преимущества сочетания:

– структуризация элементов веб–интерфейса [8];

– разграничение представления информации от её расположения [9];

– поддержка любого веб–браузера;

– высокий уровень безопасности данных;

– высокая стабильность;

– кэширование стиля страницы, для повышения скорости загрузки страницы;

– централизованное хранение данных – в одном «.html» файле описывается расположение всех элементов страницы, а в одном «.css» файле хранится описание вида всех элементов страницы.

Фреймворк «FastAPI» для работы с базой данных использует программу «Redis». Это резидентная система управления базами данных (размещает базу данных в оперативной памяти) класса «NoSQL» с открытым исходным кодом, работающая со структурами данных типа «ключ – значение». Используется как для баз данных, так и для реализации кэшей, брокеров сообщений [10].

Базой данных, для хранения списка пользователей, их лицевой метрики, будет использоваться «PostgreSQL». Это бесплатная система управления реляционными базами данных с открытым исходным кодом (СУБД) с упором на расширяемость и соответствие «SQL». Программа библиотека «PostgreSQL» библиотека поддерживает транзакции со свойствами «Atomity», «Consistency», «Isolation», «Durability» («ACID»), автоматически обновляемыми представлениями, материализованными представлениями, триггерами, внешними ключами и хранимыми процедурами [11]. Он предназначен для обработки различных рабочих нагрузок, от отдельных машин до хранилищ данных или веб–служб с множеством одновременных пользователей. Программа «PostgreSQL» является обладает более быстрой скоростью обработки таблиц нежели программа «MySQL», исходя из тестов. Далее приведены графики сравнения времени выполнения различных операций в СУБД «MySQL» и «PostgreSQL».

В случае добавления записей «PostgreSQL» оказался быстрее в среднем в 3,46 раза (рисунок 5). При внутреннем объединении на небольшом количестве строк СУБД «PostgreSQL» отрывается от СУБД «MySQL» незначительно (в районе 25 процентов), но с увеличением размера второй таблицы преимущество СУБД «PostgreSQL» становится более очевидным. При достижении второй таблицей максимального размера скорость объединения в СУБД «PostgreSQL» уже в 2.8 раз выше. В среднем СУБД «PostgreSQL» оказалась быстрее в 1,66 раза (рисунок 6) [12].

Для получения информации(кадров) с камер будет использоваться протокол «RTSP». Являющийся прикладным протоколом, предназначенным для мультиплексирования и пакетирования мультимедийных транспортных потоков (таких как интерактивные медиа, видео и аудио) по подходящему транспортному протоколу и использования в системах, работающих с мультимедийными данными (мультимедийным содержимым, медиасодержимым), и позволяющий удалённо управлять потоком данных с сервера, предоставляя возможность выполнения команд, таких как запуск (старт), приостановку (пауза) и остановку (стоп) вещания (проигрывания) мультимедийного содержимого, а также доступа по времени к файлам, расположенным на сервере [13].

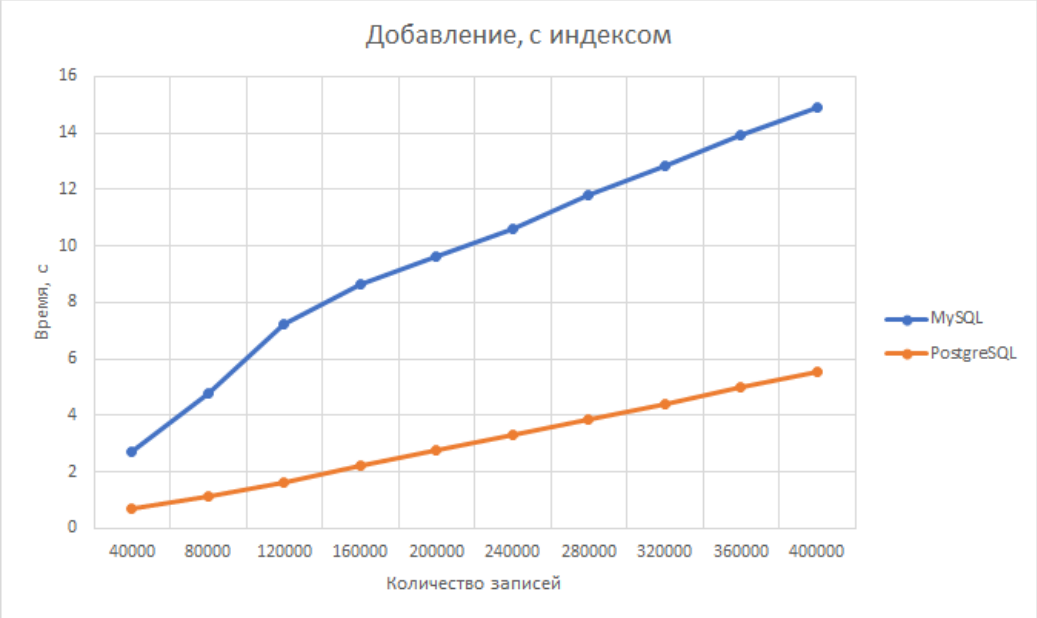


Рисунок 5 – График производительности при выполнении добавлении записей в таблицу.

Рисунок 6 – График производительности, при внутреннем объединении записей

3.3 Разработка диаграммы развёртывания системы

Для размещения системы будет использоваться технология контейнеризации. Контейнеризация – метод виртуализации, при котором ядро операционной системы поддерживает несколько изолированных экземпляров

пространства пользователя вместо одного. Эти экземпляры (обычно называемые контейнерами или зонами) с точки зрения выполняемых в них процессов идентичны отдельному экземпляру операционной системы.

Преимущества контейнеризации:

– обеспечении ядром системы полной изолированности контейнеров, поэтому программы из разных контейнеров не могут воздействовать друг на друга [14], на рисунке 7 схематично показана структура реализации приложения «Docker» на «Linux»;

– отсутствие дополнительных ресурсных накладных расходов на эмуляцию виртуального оборудования и запуск полноценного экземпляра операционной системы, характерных при аппаратной виртуализации [14].

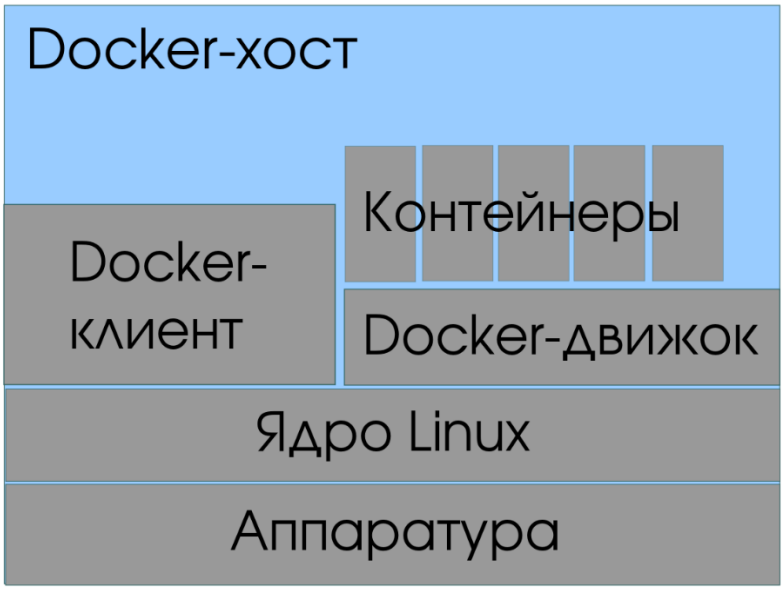


Рисунок 7 – Docker на физическом Linux–сервере

Главным недостатком контейнеризации в отличие от аппаратной виртуализации, при которой эмулируется аппаратное окружение и может быть запущен широкий спектр гостевых операционных систем, выражен в том, что в контейнере может быть запущен экземпляр операционной системы только с тем же ядром, что и у операционной системы на хостинговой машине (все контейнеры узла используют общее ядро).

В связи с опытом и простотой использования выбрано приложение «Docker». Это программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации – контейнеризатор, который позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер, который может быть развёрнут на

любой Linux–системе с поддержкой контрольных групп в ядре, а также предоставляет набор команд для управления этими контейнерами [15].

Так же для развёртывания системы будет использоваться такой инструмент как «Docker–Compose». Это инструмент для определения и запуска много контейнерных приложений в «Docker». Он использует файлы .yaml для настройки служб приложения и выполняет процесс создания и запуска всех контейнеров с помощью одной команды. Утилита «CLI» позволяет пользователям запускать команды для нескольких контейнеров одновременно, например, создавать образы, масштабировать контейнеры, запускать остановленные контейнеры и т. д. Команды, связанные с манипулированием изображениями или параметрами взаимодействия с пользователем, не имеют значения в приложении «Docker–Compose», поскольку они относятся к одному контейнеру. Файл docker–compose.yml используется для определения служб приложения и включает в себя различные параметры конфигурации. Например, «build» параметр определяет параметры конфигурации, такие как путь к файлу Dockerfile, «command» параметр позволяет переопределять команды программы «Docker» по умолчанию и многое другое.

Операционной системой для развёртывания контейнеров будет использоваться операционная система «Ubuntu» версии 20.04. Она основана на системе «Debian». Ориентирована на удобство и простоту использования. Она включает широко распространённое использование утилиты «sudo», которая позволяет пользователям выполнять администраторские задачи, не запуская потенциально опасную сессию суперпользователя [16]. Преимущества операционной системы «Ubuntu»:

– безопасность;

– наличие обширной базы решений различных проблем, возникающих при эксплуатации системы;

– низкие системные требования;

– постоянная поддержка со стороны разработчиков;

– бесплатная модель распространения.

Для обеспечения стабильности соединения и целостной работы с запросами используется веб–сервер «Nginx». Это веб–сервер и почтовый прокси–сервер, работающий на Unix–подобных операционных системах (тестировалась сборка и работа на «FreeBSD», «OpenBSD», «Linux», «Solaris», «macOS», «AIX» и «HP–UX») [17].

Основные функции веб–сервера «Nginx», при использовании как HTTP сервера:

– обслуживание неизменяемых запросов, индексных файлов, автоматическое создание списка файлов, кэш дескрипторов открытых файлов;

– акселерированное проксирование без кэширования, простое распределение нагрузки и отказоустойчивость;

– поддержка кеширования при акселерированном проксировании и «FastCGI»;

– акселерированная поддержка «FastCGI» и memcached–серверов, простое распределение нагрузки и отказоустойчивость;

– модульность, фильтры, в том числе сжатие («gzip»), «byte–ranges» (докачка), chunked–ответы, HTTP–аутентификация, SSI–фильтр;

– несколько подзапросов на одной странице, обрабатываемых в SSI–фильтре через прокси или «FastCGI», выполняются параллельно;

– поддержка «SSL»;

– поддержка «PSGI», «WSGI»;

– экспериментальная поддержка встроенного «Perl».

В веб–сервере «Nginx» рабочие процессы обслуживают одновременно множество соединений, мультиплексируя их вызовами операционной системы «select», «epoll» («Linux») и «kqueue» («FreeBSD»). Рабочие процессы выполняют цикл обработки событий от дескрипторов. Полученные от клиента данные разбираются с помощью конечного автомата. Разобранный запрос последовательно обрабатывается цепочкой модулей, задаваемой конфигурацией. Ответ клиенту формируется в буферах, которые хранят данные либо в памяти, либо указывают на отрезок файла. Буфера объединяются в цепочки, определяющие последовательность, в которой данные будут переданы клиенту.

Развёртывание системы подразделяется на 3 этапа: подготовка, монтаж и установка с настройкой.

Первый этап включает себя действия, начинающиеся от планирования использования и до этапа получения закупленного оборудования.

В первую очередь создаётся план, в котором должно указываться следующая информация:

– список кабинетов, требующих использования данной системы;

– список персонала, а также их фотографии, для дальнейшей их обработки;

– схема сетевой топологии системы, содержащий все элементы данной системы, в схеме должны быть отображены связи этих элементов, а также их минимальные настройки;

– составление списка оборудования для системы;

– составление списка денежных затрат на монтаж системы.

В список оборудования для закупки обязательно должны включаться:

– камеры;

– кабель LAN;

– внешние кабельные каналы;

– маршрутизаторы;

– сервер обработчик.

Далее происходит согласование плана, после чего происходит закупка оборудования, а также поиск подрядных организаций для монтажа системы, если отсутствуют возможности монтажа собственными силами организации.

Этап монтажа включает в себя:

– создание кабельных каналов для прокладки цепей связи между оборудованием системы;

– монтаж устройств ограничения физического доступа для камер и помещений, содержащих в себе элементы системы;

– прокладка дополнительных питающих электрических цепей, при отсутствии или нехватки их;

– прокладка кабельных линий связи;

– монтаж камер.

Этап установки и настройки подразумевает настройка программного обеспечения и сетевого окружения системы. А именно:

– настройка видео потока с камер;

– настройка сетевых настроек операционной системы;

– запуск контейнеров, в которых буду производится работа частей системы.

Вначале необходимо соединить камеры, а также сервер разработчик в одну локальную сеть, либо напрямую, либо используя маршрутизаторы.

Первоначально на сервере обработчике необходимо установить операционную систему «Ubuntu 20.04», при необходимости или нежелании пользоваться графическим интерфейсом операционной системы можно установить серверную вариацию.

На установленной операционной системе необходимо установить следующий набор программ:

– «Docker»;

– «Docker–compose»;

– «OpenSSH–server».

Далее нужно на сервер обработчик перенести весь программный проект. Затем в терминале операционной системе нужно запустить «Docker compose»,

указав путь до docker–compose.yml. После этого нужно следить за docker контейнерами, которые запускаются в система. После того как все контейнеры запустятся, можно отключить серверу–обработчику доступ к интернету, т.к. все нужные пакеты данных были скачаны.

На рисунке 8 и листе графического материала ГУИР.466152.003 ПД рассмотрена диаграмма развёртывания, дающая схематичное представление о ходе развёртывания программного обеспечения на сервере–обработчике и сервере базы данных и веб–интерфейса.

Порядок развёртывания сервера обработчика:

– запуск программы «Docker» с помощью файла Dockerfile;

– создание контейнера и копирование в него файлов, содержащих программный код сервера обработчика.

Порядок развёртывания веб–сервера:

–запуска файла runs.sh;

–запуск docker–composer с использованием файла docker–compose.yml;

– создание контейнера с базой данных «PostgreSQL»;

– создание контейнера с программный кодом фреймоврка «Django» и копирование в него программного кода веб–интерфейса части «backend»;

– создание контейнера с «Nginx»;

– создание контейнера с «Redis».

После развёртывания системы необходимо наполнение базы данных, с использованием административной части веб–интерфейса.

Вывод: разработана диаграмма развёртывания, иллюстрирующая порядок развёртывания программной части системы. Описаны экранные формы веб–интерфейса системы. Подобрано оборудования для системы, согласно требованиям. Для разработки системы используется язык программирования Python, использование которого позволяет упростить разработку системы. Для хранения данных учёта система использует базу данных «PostgreSQL», обеспечивающей быстроту обработки данных, в сравнении с другими базами данных SQL–типа. Для разработки веб–интерфейса системы используется фреймворк «FastAPI», обеспечивающий удобство разработки интерфейса, веб–страница представлена в виде html–страницы с использованием таблицу стилей css. Для развёртывания системы на операционной системе серверов используется программа «Docker», позволяющая быстро и безопасно развёртывать программное обеспечение. Разработаны диаграмма сетевой топологии и диаграмма компонентов.

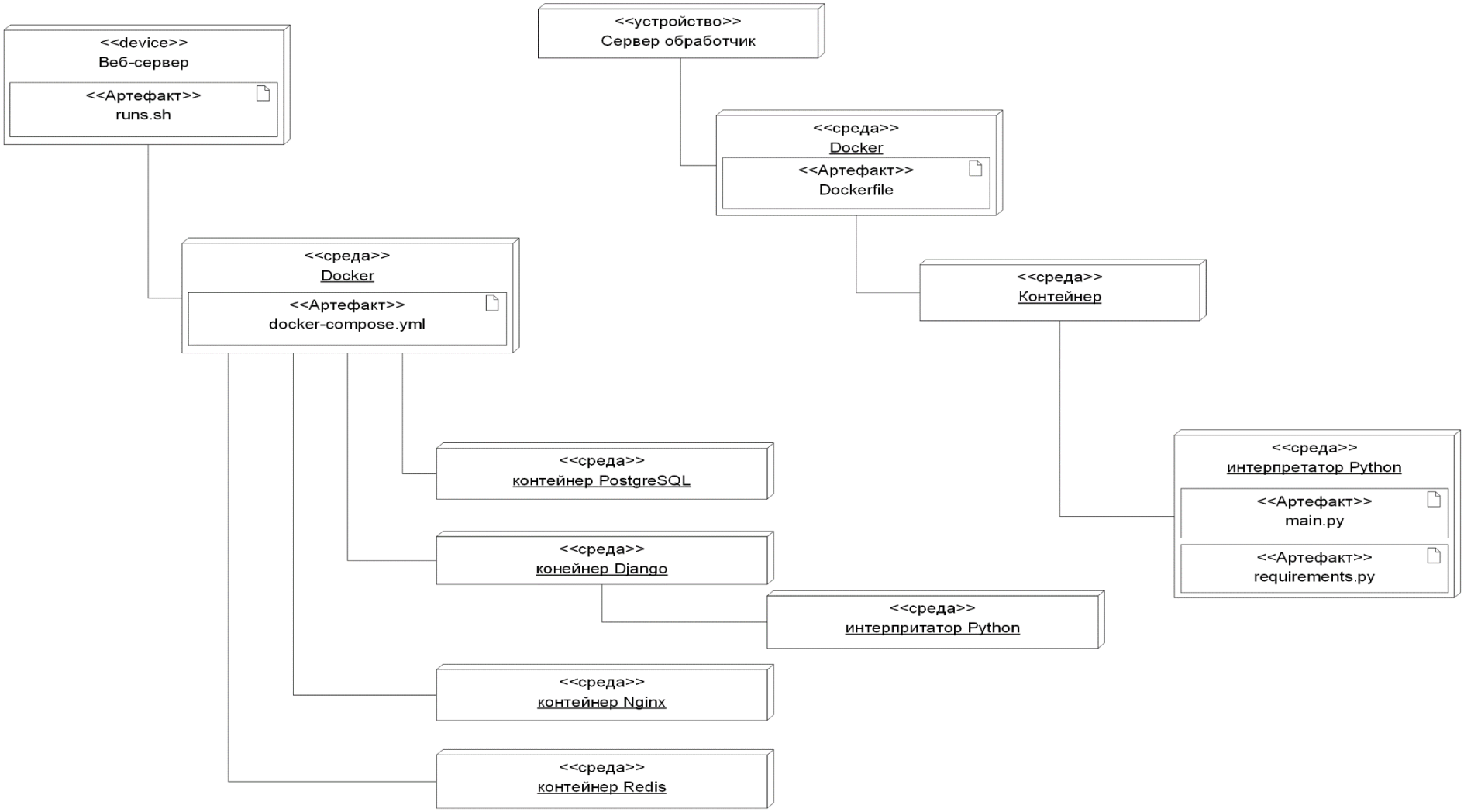


Рисунок 8 – Диаграмма развёртывания

4. РАЗРАБОТКА АЛГОРИТМОВ И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ АВТОМАТИЧЕСКОЙ СИСТЕМЫ УЧЁТА ПЕРСОНАЛА

4.1 Структуризация компонентов программного кода системы

Вся программная часть системы разделяется на 2 части, которые работают независимо друг от друга, но взаимодействующих с одной базой данных: веб–интерфейс и обработка данных с камеры.

В свою очередь веб–интерфейс делится на 2 части: frontend и backend.

Backend часть представляет собой api–запросы, возвращающие необходимую информацию в зависимости от названия запроса и содержания входных данных запроса. Все запросы требуют при запросе содержания в шапке запроса api–ключа пользователя, получаемый при положительной авторизации – отправке корректных данных об имени пользователя и его пароле в веб–интерфейсе. Построение backend части построено на связи api–запросов с сервисами (наследованы от внутреннего класса «service» фреймворка), что позволяет разграничить описание запроса с реализацией его выполнения.

На рисунке 9 представлена файловая схема backend части веб интерфейса. На нем представлены следующие объекты:

– файл main.py – запускает всю программу и создаёт пользователя по умолчанию, если такового не имеется;

– файл setting.py – предназначен для структуризации переменных, передаваемых через .env файл;

– файл core.py – содержит базовые настройки для фреймворка «FastAPI»;

– файл .env – предназначен для передачи в программу переменных окружения, без использования флагов;

– папка «api» – содержит описание api–запросов программы;

– папка «service» – содержит реализацию api–запросов программы;

– папка «schemas» – содержит описание классов для работы с БД;

– папка «models» – содержит описание моделей запросов;

– папка «databese» – содержит описание подключения к БД;

– папка logs – содержит файл «script.log», содержащий логи работы программы.

Далее будут подробнее рассмотрено содержание папок.

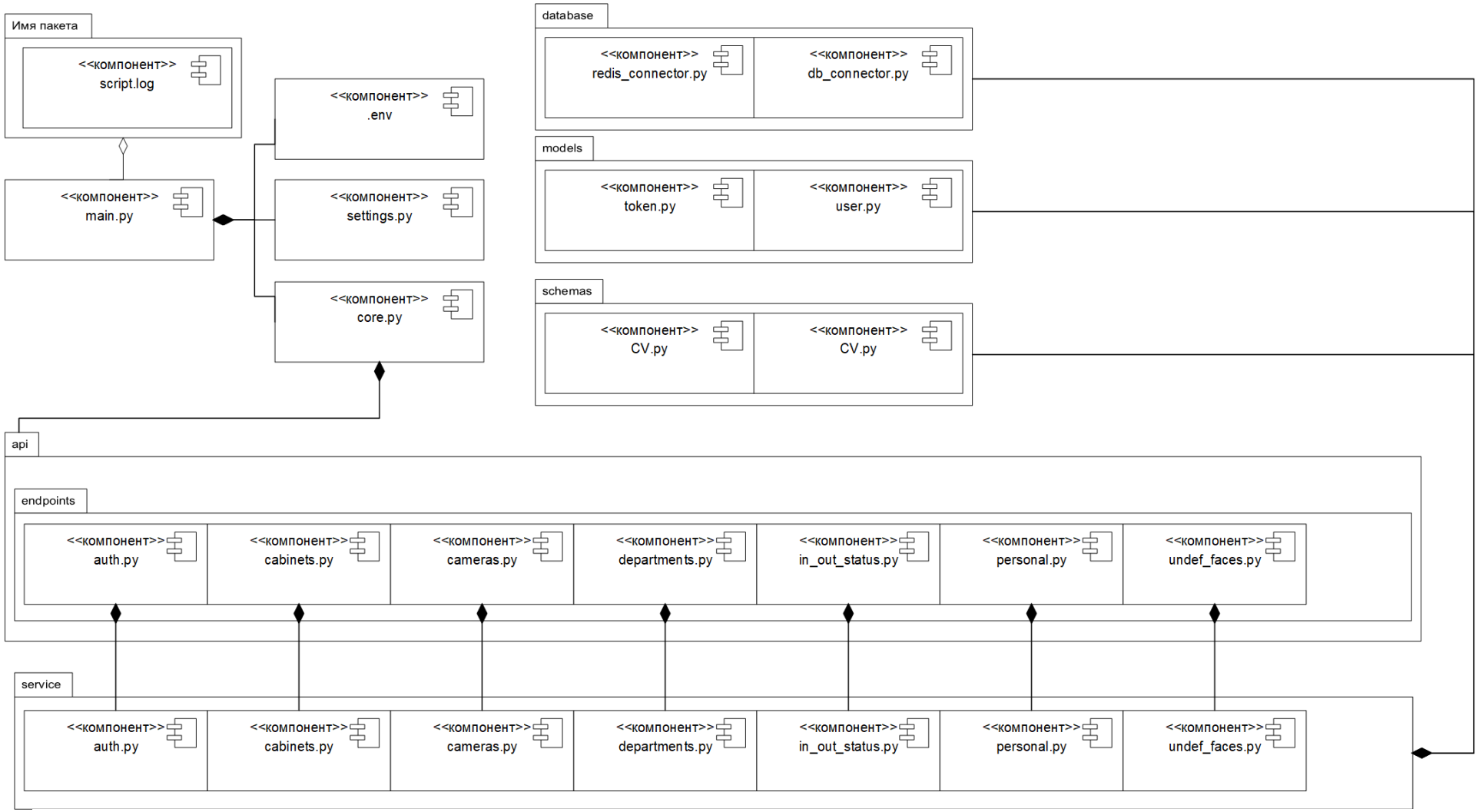


Рисунок 9 – Диаграмма компонентов backend

Папка «api» содержит файл routers.py, в котором идёт включение api–запросов из папки endpoints к основному api–пути программы. Также содержит подпапку «endpoints», которая предназначена для хранения файлов api–запросов, разделённых на категории. Эти файлы в api запроса обращаются к методам классов из одноимённым файлов из папки проекта «service», которые предоставляют выполнение запросов в базу данных системы. Файлы:

– auth.py – содержит запросы на авторизацию пользователя и его «выход» из веб–интерфейса, а также запрос на предоставлении информации о пользователе(требующий для этого заранее пройденной авторизации);

– cabinets.py – содержит запросы, отвечающие за предоставление информации о помещениях системы;

– cameras.py – содержит запросы, отвечающие за предоставление информации о камерах системы;

– departments.py – содержит запросы, отвечающие за предоставление информации об отделах системы;

– in_out_status.py – содержит запросы, отвечающие за предоставление информации о посещениях;

– personal.py – содержит запросы, отвечающие за предоставление информации о работниках предприятия в системе;

– undef_faces.py – содержит запросы, отвечающие за предоставление информации о неопознанных лицах.

Далее будут рассмотрены запросы, содержащиеся в файлах.

Файл auth.py содержит следующие запросы:

– «sign_in» – возвращает ключ пользователя;

– «logout» – удаляют ключ из системы;

– «userinfo» – возвращает полную информацию о пользователе.

Файл cabinets.py содержит следующие запросы:

– «get_cab» – возвращает подробную информацию о помещении, по его «id»;

– «get_all» – возвращает список всех помещений системы;

– «get_cabinet_per_ids» – возвращает список лиц, допущенных в конкретное помещение;

– «cab_visits» – возвращает список посещений работниками конкретного помещения за определённую дату;

–«cab_visits_pos» – возвращает список посещений работниками конкретного помещения за определённую дату, в зависимости от направления движения;

–«pass_visits» – возвращает список прохождения через конкретный проход за определённую дату;

– «pass_visits_pos» – возвращает список прохождения через конкретный проход за определённую дату, в зависимости от направления движения.

Файл cameras.py содержит следующие запросы:

– «get_all» – возвращает список всех камер;

– «get_one» – возвращает подробную информацию о камере, по его «id»;

– «get_cabinet_cameras» – возвращает список камер, стоящих в конкретном помещении.

Файл departments.py содержит следующие запросы:

– «get_all» – возвращает список всех отделов.

Файл in_out_status.py содержит следующие запросы:

– «get_limit» – возвращает лимитированный список посещений кабинет.

Файл personal.py содержит следующие запросы:

– «get_user» – возвращает полную информацию о работнике;

– «get_all» – возвращает список всех работников;

– «worker_day_visits» – возвращает список помещений, которые посетил работник за определённую дату;

– «worker_day_visits_pos» – возвращает список помещений, которые посетил работник за определённую дату, в зависимости от направления движения;

– «get_personal_faces_ids» – возвращает список идентификаторов лиц, которые относятся к конкретному работнику;

– «get_single_faces» – возвращает фотографию конкретного лица работника;

– «get_day_work_time» – возвращает количество часов, проведённых работником в рабочих кабинетах, а также количество часов, проведённых в нерабочих кабинетах за определённую дату;

– «get_day_work_time_moth» – возвращает количество часов, проведённых работником в рабочих кабинетах, а также количество часов, проведённых в нерабочих кабинетах за месяц.

Файл undef_faces.py содержит следующие запросы:

– «get_limit» – возвращает лимитированный список неопознанных лиц, без их фото;

– «get_by_timestamp» – возвращает фотографию неопознанного лица.

Папка «service» содержит описание классов, унаследованных от класса service фреймворка, которые предназначены для реализации запросов. Вначале каждого метода класса идёт проверка на валидность присылаемого

токена пользователя, далее идёт запрос в базу данных, затем метод возвращает ответ в «json» формате, для чего могут потребоваться дополнительные преобразования ответов из базы данных в словари.

Папка «schemas» содержит 2 файла: «CV.py» и «userdb.py». Файлы представляют совокупность классов, предназначенных для представления таблиц базы данных в программе, для удобства взаимодействия с этими таблицами. Поля представляют собой классы «Column» и типов данных, хранящихся в одноимённых с полем колонках в БД, которые являются членами библиотеки «sqlalchemy». Привязка к определённой таблице осуществляется через переменную «__tablename__» где указывается название таблицы. Файл CV.py содержит интерпретацию таблиц, задействованных в системе, но не содержит таблиц, отвечающих за авторизацию в веб–интерфейсе, но файл userdb.py как раз предназначена для этого.

Папка «models» содержит файлы «token.py» и «user.py» описывающие модель данных для работы с токенами и пользователями.

Папка «database» содержит файлы: «db_connector.py» и «redis_connector.py». Эти файлы содержат описание классов–коннекторов, предназначенных для соединения с соответствующими им базами данных.

Frontend представлен в виде 3 файлов:

– файл index.html – представляет собой файл разметки, который отвечает за расположение элементов на странице и их связь со скриптами в файле app.js;

– файл style.css – файл стилей, в котором описан внешний вид элементов страницы;

– файл app.js – файл, хранящий скрипты, с помощью который страница получает информацию с backend.

Обработка данных с камер является ресурсно–затратным со стороны использования центрального процессора компьютера. Для повышения производительности программы используется метод многопоточной разработки

Первая часть проекта представляет собой алгоритм, состоящий из следующих шагов:

– получение кадра из видеопотока с камеры;

– поиск лиц на кадре;

– анализ найденных лиц на совпадение с персоналом предприятия;

– отправка в базу данных информации о найденных лицах;

– обновление лицевых данных из базы данных.

Для получения кадра используется библиотека «OpenCV», а именно функция «VideoCapture», в аргументы который мы передаём адрес камеры.

Для поиска лица используется библиотека «face_recognition», а именно функция «face location», в которую в качестве аргументов мы отправляем кадр, полученный из «OpenCV» и который перед этим будет сжат, для увеличения работоспособности.

Далее идёт сравнение полученного лица, с лицами, находящимися в буфере. При совпадении, отправляется строка в базу данных, о прохождении персонала у камеры.

Так же в параллельном потоке работает таймер, который раз в 24 часа проверяет внутренний буфер лиц программы на актуальность, а при наличии новых загружает их из базы данных.

4.2 Разработка базы данных

В СУБД «PostgreSQL» находится база данных «cvdb», которая хранит в себе всю информацию, необходимую на работы система – список сотрудников, их биометрия лиц и т.д. На рисунке 10 и листе графического материала ГУИР.466152.004 ПД представлена диаграмма базы данных. База данных состоит из следующих таблиц:

– таблица «department»;
– таблица «personal»;
– таблица «faces»;
– таблица «cabinets»;
– таблица «cameras»;
– таблица «in_out_date»;
– таблица «in_undendified_faces»;
– таблица «hours_stats».

Таблица «department» хранит список отделов предприятия, состоит из 2 колонок:

– поле id – типа serial, авто заполняемый уникальный идентификатор отдела, является первичным ключом таблицы;

– поле name – типа text, хранит название отдела.

Таблица «personal» хранит список персонала, который работает в каждом отделе, состоит из 7 колонок:

– поле id – типа serial, авто заполняемый уникальный идентификатором персонала, является первичным ключом таблицы;

– поле first_name – типа text, хранит имя работника;

– поле last_name – типа text, хранит фамилию работника;

– поле father_name – типа text, хранит отчество работника;

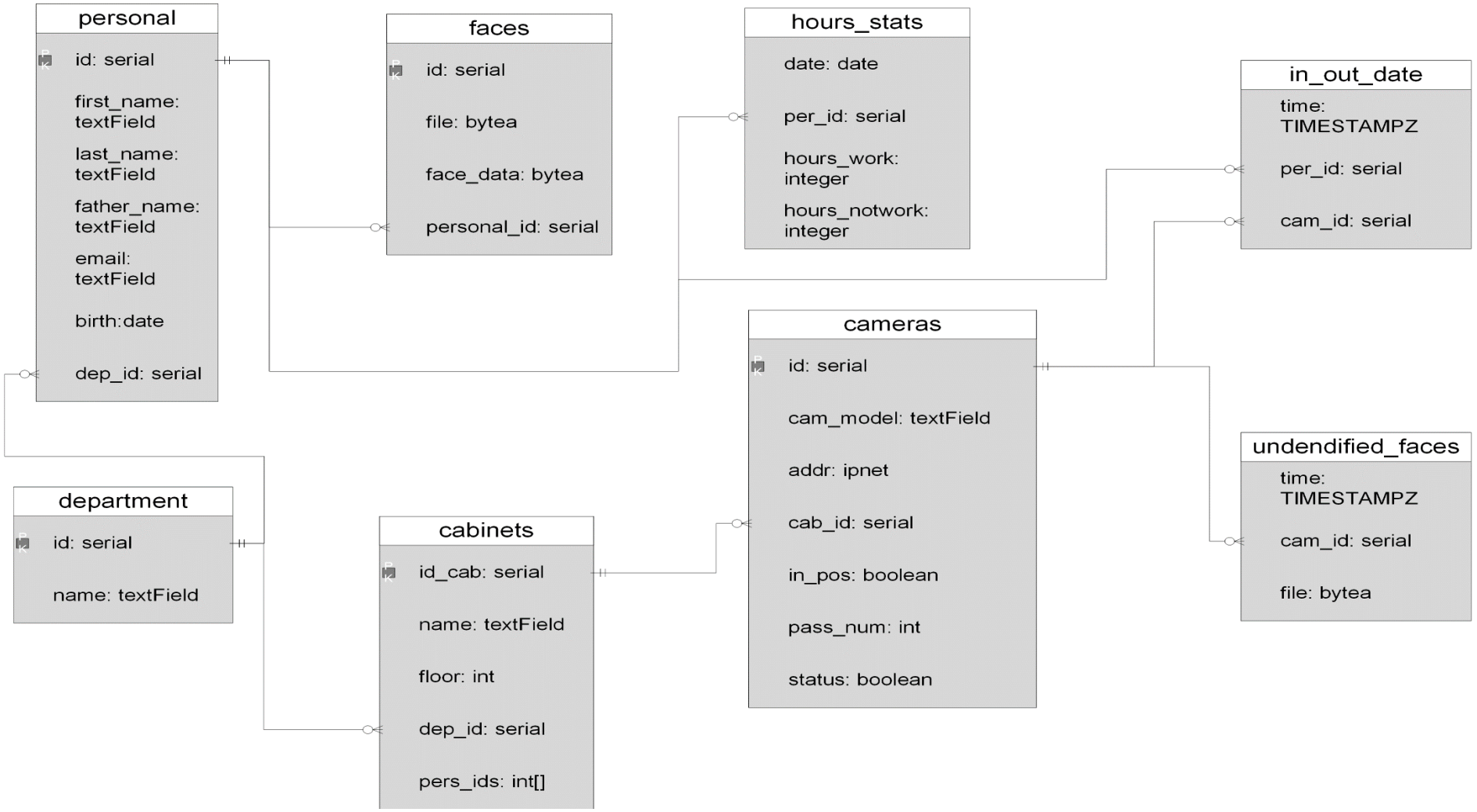


Рисунок 10 – Схема базы данных

– поле email – типа text, хранит email адрес работника;

– поле birth – типа date, хранит дату рождения работника;

– поле dep_id – типа serial, хранит ссылку на отдел, к которому относится сотрудник.

Таблица «faces» хранит лицевые параметры, состоит из 4 колонок:

– поле id_face – типа serial, авто заполняемый уникальный идентификатором лицевого параметра, является первичным ключом таблицы;

– поле file – типа bytea, хранит файл фотографии из которого сформировались лицевые параметры;

– поле face_data – типа bytea, хранит лицевые параметры;

– поле personal_id – типа serial, хранит ссылку на работника персонала, которому принадлежат данные лицевые параметры.

Таблица «cabinets» предназначена для хранения списка кабинетов, состоит из 4 колонок:

– поле id_cab – типа serial, авто заполняемый уникальный идентификатором кабинета, является первичным ключом таблицы;

– поле name – типа text, предназначена для хранения названия кабинета;

– поле floor – типа int, хранит номер этажа, на котором расположен кабинет;

– поле dep_id – типа serial, хранит ссылку на отдел, которому принадлежит кабинет;

– поле pers_ids – типа массив integer, хранит id работников, допущенных в кабинет.

Таблица «cameras» предназначена для хранения списка камер, состоит из 7 колонок:

– поле id – типа serial, авто заполняемый уникальный идентификатором камеры, является первичным ключом таблицы;

– поле cam_model – типа text, предназначен для хранения названия камеры;

– поле addr – типа ipnet, хранит ip address, по которому доступна камера;

– поле cab_id – типа serial, хранит ссылку на кабинет, к которому приписана камера;

– поле in_pos – типа bool, в значении True, если камера расположена на вход в кабинет;

– поле pass_num – типа int, хранит номер прохода в помещение, к которому привязана камера;

– поле status – типа bool, хранит значение параметра работоспособности камеры.

Таблица «in_out_date» предназначена для фиксирования информации, о входящих и выходящих сотрудников, состоит из 3 колонок:

– поле time – типа TIMESTAMPZ, авто заполняемая дата и время, когда опознали лицо;

– поле per_id – типа serial, хранит ссылку на работника персонала, которому принадлежат данные определённого лица;

– поле cam_id – типа serial, хранит ссылку на камеру, которая опознала лицо.

Таблица «undendified» хранит дату и время появления неопознанного лица, а также его изображение, состоит из 3 колонок:

– поле time – типа TIMESTAMPZ, авто заполняемая дата и время, когда зафиксировала система лицо;

– поле cam_id – типа serial, хранит ссылку на камеру, которая зафиксировала лицо;

– поле file – типа bytea, хранит файл изображения, на котором зафиксировано неопознанное лицо.

Так же в PostgreSQL созданы несколько ролей, обладающими различными права:

– поле admin – обладает всеми возможностями для работы с базой данных;

– поле reader_all – обладает только возможностями чтения данных из всех таблиц;

– поле reader_basic – обладает только возможностями чтения всех таблиц, кроме таблиц faces, department, personal.

Таблица «hours_stats» предназначена для хранения статистика рабочего времени в закреплённых и незакреплённых за работником кабинетов, состоит из 4 колонок:

– поле date – типа date, хранит дату, за которую собраны данные;

– поле per_id – типа serial, предназначена для хранения ссылки на работника, по которому собраны данные;

– поле hours_work – типа int, хранит минуты, которые провёл работник в закреплённых кабинетах;

– поле hours_notwork – типа int, хранит минуты, которые провёл работник в незакреплённых кабинетах.

4.3 Разработка механизмов отчёта для интеграции в различные системы

Механизм системы отчётов для интеграции в различные системы представляет собой разработанные SQL–функции для базы данных системы. Данный способ реализации позволяет использовать готовые инструментарий систем, для которых данных механизм и разработан, также он позволяет миновать промежуточные звенья в передаче сигналов, которые бы имели место при реализации API, для работы системы.

Отчеты сгруппированы по следующим критериям:

– отчеты по каждому работнику;

– отчеты по кабинету;

– отчеты по проходу;

Отчёты по каждому работнику представляют совокупность таблиц, которые отражают информацию по конкретному работнику предприятия и представлены следующими SQL–функциями:

– worker_day_visits возвращает список посещённых работником помещений за определённую дату, входные переменные: date_f – день, за который нужно отфильтровать значения, per_in – id работника;

– worker_day_visits_pos возвращает список посещённых работником помещений за определённую дату в зависимости от направления движения, переменные аналогичные предыдущей функции, кроме одной дополнительной pos – булева переменная (TRUE – заходил, FALSE – выходил).

Отчеты по кабинету предоставляют таблицы, представляющие совокупность информации о пользовании кабинетами, представлены следующими функциями:

– cab_visits – возвращает список посещений работниками помещения за определённую дату, входные переменные: date_f – дата посещения, cab_in – id кабинета;

– cab_visits_pos – аналогично предыдущей функции, но показывает, в зависимости от флага, кто заходил или выходил, один дополнительный входный параметр – pos, является булевой переменной (TRUE – заходил, FALSE – выходил).

Отчеты по проходу, предоставляют таблицы, о использовании конкретных проходов в помещения. Представлены следующими функциями:

– pass_visits – показывает кто и во–сколько прошёл через проход конкретного помещения за определённую дату, содержит следующие входные

параметры: date_f – дата посещения, pass – номер прохода, cab_in – номер кабинета;

– pass_visits_pos – аналогично предыдущей функции, так же конкретизирует направление движения персонала.

Реализация вышеперечисленных отчётов представлена в виде функций языка «PSQL» представлена в приложении Б.

Также есть отчёты по статистике рабочего и не рабочего времени, которые содержат сколько часов работник провёл в помещении, закреплённом за ним и других, незакреплённых помещения незакреплённых как рабочие. Отчёты хранятся в базе данных в таблице «hours_stats», в которой каждая строка хранит данные по определённому работнику за определённую дату.

Все отчёты продублированы в backend части веб-интерфейса, в качестве одноименных запросов. В следующие группы на разделах находятся отчёты:

– в разделе «personal» находятся отчёты по конкретному работнику и статистике рабочего и нерабочего времени за день и месяц;

– в разделе «cabinets» находятся отчёты по конкретному работнику и по конкретному проходу.

4.4 Разработка алгоритма работы и программного обеспечения системы

Алгоритм представлен на рисунке 11 и листе графического материала ГУИР.466152.005 ПД изображена диаграмма классов, отражающая взаимодействие программных классов веб–интерфейса между собой.

Алгоритм работы программы обработчика можно разделить на 3 части:

– подготовительная;

– основной цикл;

– цикл сравнения.

Подготовительная часть включает в себя считывание данных .env файла, содержащего данные, для подключения к базе данных система, а также параметры для камеры. При отсутствии таковых данных программа логирует сообщение об ошибке и выключается, но т.к. она находится в docker–контейнере, то контейнер выключается. После положительного прохождения проверки данных, программа проверяет возможность подключения к базе данных системы, при отсутствии такового программа логирует сообщение об ошибке, затем приостанавливается на 5 секунд и пробует заново проверить возможность подключения и так продолжается, то получения возможности подключения. Далее происходит запрос в БД на получение списка векторов

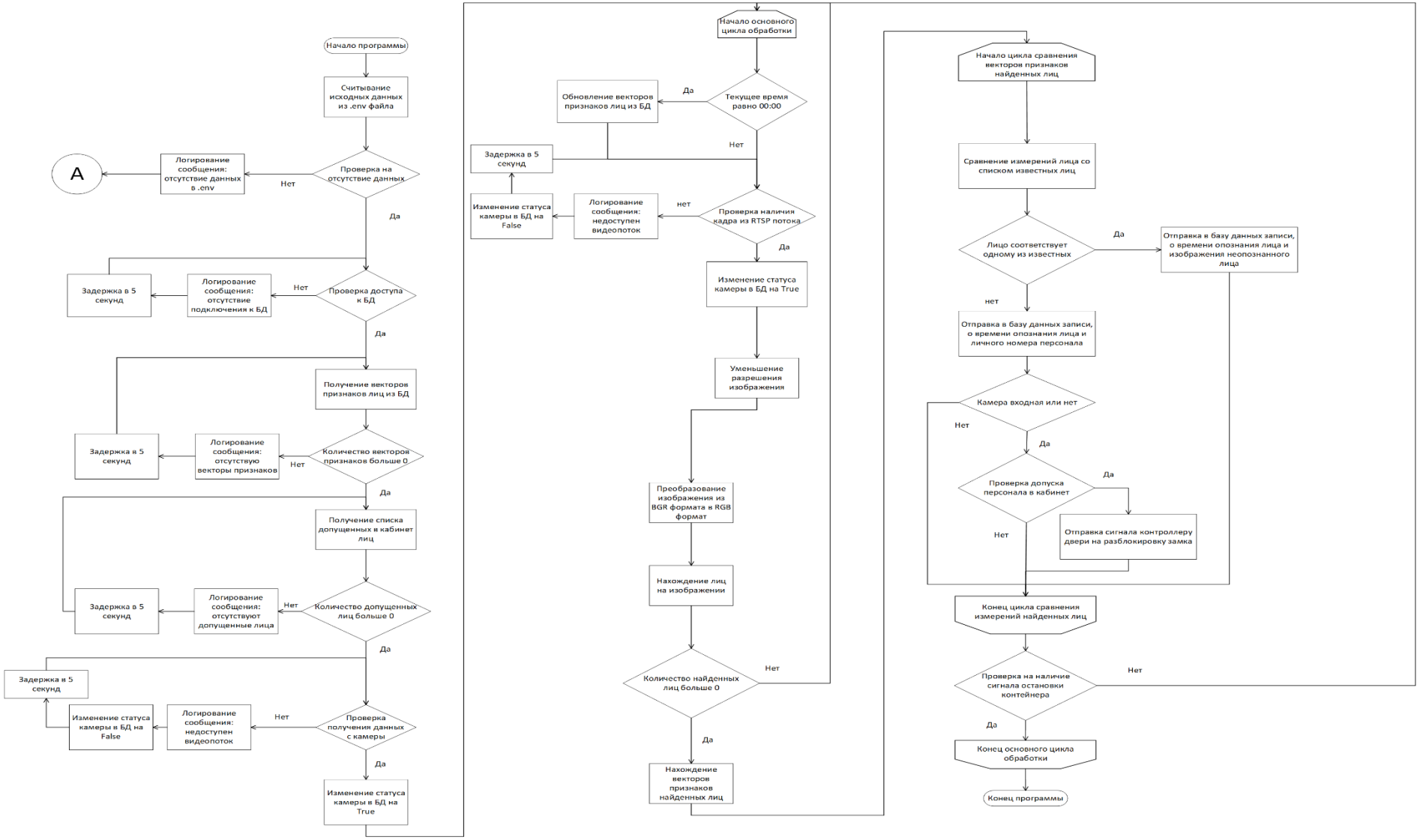


Рисунок 11 – Схема лгоритма программы обработчика

лиц. Если на запрос пришло 0 векторов, то программа логирует сообщение об ошибке, затем останавливается на 5 секунд и повторяет запрос и так, пока список векторов не станет содержать хотя бы один вектор. После получения наполненного списка векторов, происходит запрос в БД для получения списка идентификаторов персонала, допущенных в кабинет. Потом происходит проверка, на количество полученных идентификаторов, если количество равно 0, то происходит логирование сообщения об ошибке, затем задержка на 5 секунд и выполнение повторного запроса списка. При наличии в списке хотя бы одного идентификатора персонала, происходит проверка на получения данных с камеры. При отсутствии данных система вначале логирует сообщение об ошибке, затем изменяет статус камеры (за обработку данных с который данная программа отвечает) на значение False, потом останавливается на 5 секунд и снова производит проверку на получения данных с камеры, при положительном результате проверки программа меняет статус камеры на True. Далее идёт основной цикл.

Основной цикл является бесконечным. Начинается с проверки текущего времени, если оно равно 00:00, то обновляется список векторов признаков. Далее идёт получение кадра с видеопотока камеры, если не получается получить кадр, то изменяется статус камеры в базе данных системы на False, далее программа логирует сообщение об ошибке, затем приостанавливается на 5 секунд и снова пытается получить кадр и так, пока не удастся получить кадр. После получения кадра статус камеры в базе данных меняется на True, если до этого он в программе менялся. Далее происходит сжатие изображения на 0.25 по каждой из осей. Далее изображение преобразуется из BGR формата в RGB. Далее на изображение происходит лиц, используя «hog» алгоритм. Далее происходит сравнение количества найденных лиц, если лица не найдены то программа переходит к новой итерации основного цикла обработки. Если лица найдены(может быть и одно) то далее происходит процесс получения векторов признаков найденных лиц с помощью обученной сверточной нейронной сети. Далее начинается цикл сравнения.

Цикл сравнения заключается в том, что он проходит по списку всех векторов признаков – каждую итерацию берётся последующий вектров признаков из списка. Вектор сравнивается с векторами из базы данных, путём вычитания из вектора базы данных сравниваемого вектора (а точнее сравнивается со всем списком векторов базы данных). Если разница меньше допустимого порога, то найдено лица из базы данных и в базу данных отправляется запись о том, что опознано лицо; если же разница больше порога, то лицо считается не опознанным и в базу данных отправляется запись в базу

данных о том, что найдено неопознанное лицо, а также изображение этого лица и идёт сравнение следующих лиц. Далее при положительном опознании лица происходит проверка на направление камеры. Если направление камеры True (камера фиксирует входящих), то затем происходит проверка на наличие идентификатора персонала, которому принадлежит опознанное лицо, на наличие в списке допущенных лиц в кабинет, если идентификатор в списке, то происходит отправка сигнала сетевому контроллеру двери на открытие, если же вне списка, то цикл заканчивается. Если направление камеры False (камера фиксирует выходящих) – то идёт переход к следующей итерации цикла сравнения.

После того как весь список векторов найденных лиц пройдёт сравнение цикл сравнения заканчивается. Также заканчивается итерация основного цикла обработки и начинается заново.

Программный код веб–интерфейса приведён в приложении А.

4.5 Экранные формы веб–интерфейса

Далее подробнее рассмотрены элементы графического листа ГУИР.466152.006 ПД.

На рисунке 12 представлена таблица, предоставляющая данные о посещении конкретного работника помещений за определённую дату. Состоит из 3 столбцов:

– «Datetime» – временная метка, в которой было зафиксирован работник;

– «Cabinet number» – идентификатор кабинета, в который зашёл или вышел работник;

– «Direction» – показывает направление движения работника: зелёный цвет – внутрь помещения, красный – из помещения.

На рисунке 13 представлена таблица, предоставляющая данные о посещении конкретного работника помещений за определённую дату и в зависимости от определённого направления движения. Состоит из 2 столбцов:

– «Datetime» – временная метка, в которой было зафиксирован работник;

– «Cabinet number» – идентификатор кабинета, в который зашёл или вышел работник;

На рисунке 14 представлена таблица, предоставляющая данные о посещении работниками определённого помещений за определённую дату. Состоит из 3 столбцов:

– «Datetime» – временная метка, в которой было зафиксирован работник;

– «Personal id» – идентификатор работника, который зашёл или вышел из кабинета;

– «Direction» – показывает направление движения работника: зелёный цвет – внутрь помещения, красный – из помещения.

Datetime	Cabinet number	Direction
2024.01.09 13:32:39.917642	5	
2024.01.09 13:32:39.919133	5	
2024.01.09 13:32:39.920129	3	
2024.01.09 13:32:39.921119	3	
2024.01.09 13:32:39.922112	2	
2024.01.09 13:32:39.923105	2	
2024.01.09 13:32:39.923602	4	
2024.01.09 13:32:39.924595	4	
2024.01.09 13:32:39.925092	3	
2024.01.09 13:32:39.926086	3	
2024.01.09 13:32:39.927079	5	
2024.01.09 13:32:39.928074	5	
2024.01.09 13:32:39.929066	4	
2024.01.09 13:32:39.931053	4	
2024.01.09 13:32:39.931549	2	
2024.01.09 13:32:39.935025	2	
2024.01.09 13:32:39.943965	3	
2024.01.09 13:32:39.952409	3	

Рисунок 12 – Таблица посещений работником помещений

Datetime	Cabinet id
2024.01.09 13:40:50.515229	3
2024.01.09 13:40:50.51773	4
2024.01.09 13:40:50.519722	5
2024.01.09 13:40:50.521216	3
2024.01.09 13:40:50.523208	2
2024.01.09 13:40:50.524702	4
2024.01.09 13:40:50.526195	3
2024.01.09 13:40:50.527689	4
2024.01.09 13:40:50.529183	3
2024.01.09 13:40:50.53117	5
2024.01.09 13:40:50.533157	3

Рисунок 13 – Таблица посещений работником помещений в зависимости от направления движения

Datetime	Personal id	direction
2024.01.09 13:03:09.377871	1	
2024.01.09 13:03:09.378364	1	
2024.01.09 13:03:09.37932	1	
2024.01.09 13:03:09.379817	1	
2024.01.09 13:03:09.380314	1	
2024.01.09 13:03:09.38081	1	
2024.01.09 13:03:09.381307	1	
2024.01.09 13:03:09.381803	1	
2024.01.09 13:32:39.920129	2	
2024.01.09 13:32:39.921119	2	
2024.01.09 13:32:39.925092	2	
2024.01.09 13:32:39.926086	2	
2024.01.09 13:32:39.943965	2	
2024.01.09 13:32:39.952409	2	
2024.01.09 13:40:50.515229	3	
2024.01.09 13:40:50.516222	3	
2024.01.09 13:40:50.521216	3	
2024.01.09 13:40:50.522212	3	
2024.01.09 13:40:50.526195	3	
2024.01.09 13:40:50.527191	3	
2024.01.09 13:40:50.529183	3	
2024.01.09 13:40:50.530177	3	
2024.01.09 13:40:50.533157	3	
2024.01.09 13:40:50.533654	3	

Рисунок 14 – Таблица посещений работниками определённого помещения

На рисунке 15 представлена таблица, предоставляющая данные о посещении работниками определённого помещений за определённую дату и в зависимости от определённого направления движения. Состоит из 2 столбцов:

– «Datetime» – временная метка, в которой было зафиксирован работник;

– «Personal id» – идентификатор работника, который зашёл или вышел из кабинета;

Datetime	Personal id
2024.01.09 13:03:09.377871	1
2024.01.09 13:03:09.37932	1
2024.01.09 13:03:09.380314	1
2024.01.09 13:03:09.381307	1
2024.01.09 13:32:39.920129	2
2024.01.09 13:32:39.925092	2
2024.01.09 13:32:39.943965	2
2024.01.09 13:40:50.515229	3
2024.01.09 13:40:50.521216	3
2024.01.09 13:40:50.526195	3
2024.01.09 13:40:50.529183	3
2024.01.09 13:40:50.533157	3

Рисунок 15 – Таблица посещений работниками определённого помещения в зависимости от направления движения

На рисунке 16 представлена таблица, предоставляющая данные о посещении конкретного работника помещений за определённую дату и определённый проход помещения. Состоит из 3 столбцов:

– «Datetime» – временная метка, в которой было зафиксирован работник;

– «Cabinet number» – идентификатор кабинета, в который зашёл или вышел работник;

– «Direction» – показывает направление движения работника: зелёный цвет – внутрь помещения, красный – из помещения.

Datetime	Personal id	direction
2024.01.09 13:03:09.37932	1	
2024.01.09 13:03:09.379817	1	
2024.01.09 13:03:09.381307	1	
2024.01.09 13:03:09.381803	1	
2024.01.09 13:32:39.920129	2	
2024.01.09 13:32:39.926086	2	
2024.01.09 13:40:50.515229	3	
2024.01.09 13:40:50.527191	3	
2024.01.09 13:40:50.530177	3	
2024.01.09 13:40:50.533157	3	

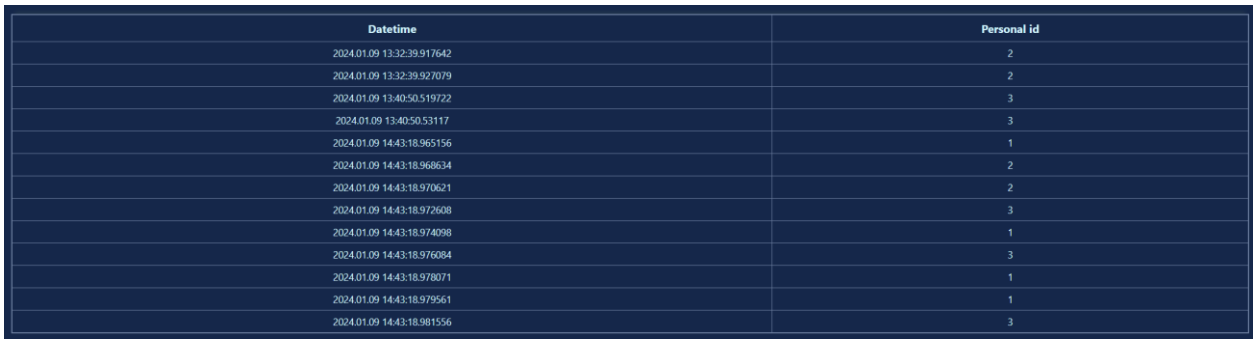
Рисунок 16 – Таблица прохождений работниками определённого прохода в помещение

На рисунке 17 представлена таблица, предоставляющая данные о посещении конкретного работника помещений за определённую дату и

определённый проход помещения, а также в зависимости от определённого направления движения. Состоит из 2 столбцов:

– «Datetime» – временная метка, в которой было зафиксирован работник;

– «Cabinet number» – идентификатор кабинета, в который зашёл или вышел работник;



Datetime	Personal id
2024.01.09 13:32:39.917642	2
2024.01.09 13:32:39.927079	2
2024.01.09 13:40:50.519722	3
2024.01.09 13:40:50.53117	3
2024.01.09 14:43:18.965156	1
2024.01.09 14:43:18.968634	2
2024.01.09 14:43:18.970621	2
2024.01.09 14:43:18.972608	3
2024.01.09 14:43:18.974098	1
2024.01.09 14:43:18.976084	3
2024.01.09 14:43:18.978071	1
2024.01.09 14:43:18.979561	1
2024.01.09 14:43:18.981556	3

Рисунок 17 – Таблица прохождений работниками определённого прохода в помещение в зависимости от направления движения

Вывод: Система состоит из двух частей – веб–интерфейс и обработчик–камеры; веб–интерфейс реализован с помощью фреймворка «FastAPI», реализующего frontend и backend части веб–интерфейса. Разработана диаграмма базы данных, описывающая структуру базы данных разрабатываемой системы. Обработчик камер реализован с помощью языка программирования Python и библиотек «OpenCV», «face_recognition». Алгоритм работы обработки камер состоит нескольких этапов: получение кадра из видеопотока камеры, нахождение лица на кадре, сравнение лица с лицами из базы данных.

5. НАСТРОЙКА МАСШТАБИРУЕМОСТИ И БЕЗОПАСТНОСТИ СИСТЕМЫ

5.1 Конфигурирование параметров безопасности системы

Безопасность системы разделяется на 3 физический, сетевой, программный.

Реализация безопасности системы на физическом уровне заключается в осуществлении следующих пунктов:

– ограничен физический доступа в помещение, где расположен сервер–обработчик, путём запирающих устройств, ключи к которым находятся только у системного администратора, начальника отдела, курирующего данную системы;

– ограничен физический доступ к промежуточному сетевому оборудованию системы (коммутаторы, маршрутизаторы и т.п.), путём размещения их в помещениях/распределительных коробах, имеющих системы запирания и ключевой доступ;

– ограничен физический доступа к узлам подключения и настройки камер, путём монтажа, не допускающего доступ, без использования специализированных инструментов – внутренняя проводка;

– ограничен физический доступа к узлам питания всех элементов системы – прокладка линий питания напрямую в щиты электропитания, которые в свою очередь имеют запирающие устройства.

Сетевой уровень безопасности подразумевает отключения возможностей доступа к серверу–обработчику не через http порт, а также доступа к ip адресам камер и промежуточного оборудования, для этого выполняются следующие действия:

– у коммутаторов отключено использование не задействованных физических портов;

– включен фаервол на серверах–обработчиках, а также на сервере веб–интерфейса и базы данных;

– добавления в реестр разрешённых ip адресов – на сервере обработчике в реестр добавляются только ip адреса камер, с которых получает данные сервер, а также свеянного с ним сервера веб–интерфейса, также добавлен один ip адресс компьютера системного администратора;

– при развёртывании контейнеров в docker в файлах Dockerfile контейнеров в поле «ports» указывается ip адрес – 127.0.0.1, чтобы к

контейнерам был доступ только с хостинговой машины, кроме контейнера отвечающего за web часть системы;

– у промежуточного сетевого оборудования включено использование «white–list» ip адресов, в этот же список добавить только ip адреса устройств, используемых в системе;

– на сервере–обработчике для контейнеров, обрабатывающих данные с камер создана отдельная внутренняя сеть в «Docker», на сервере веб–интерфейса для контейнеров тоже создана отдельная подсеть в «Docker».

Программный уровень безопасности веб-интерфейса заключается в использовании механизма авторизации. При необходимости задействовать напрямую backend часть, то запрос на авторизацию вернёт ключ, который нужно будет отсылать в заглавии при каждом следующем запросе, для подтверждения полномочий на обработку запроса.

Так же в контейнерах, которые обрабатывают изображения с камер, будет использоваться специально созданный пользователь в базе данных, чтобы контейнер имел доступ только к необходимой информации. Данный пользователь будет обладать только следующими разрешениями: чтение списка биометрии лиц, добавление информации, об опознанном лице.

Так же благодаря использованию фреймфорка, исключена возможность атак с использование SQL–инъекций.

5.2 Проектирование вариантов масштабируемости и интеграции системы

Масштабируемость, в электронике и информатике, означает способность системы, сети или процесса справляться с увеличением рабочей нагрузки (увеличивать свою производительность) при добавлении ресурсов (обычно аппаратных). Масштабируемость – важный аспект электронных систем, программных комплексов, систем баз данных, маршрутизаторов, сетей и т. п., если для них требуется возможность работать под большой нагрузкой. Система называется масштабируемой, если она способна увеличивать производительность пропорционально дополнительным ресурсам. Масштабируемость можно оценить через отношение прироста производительности системы к приросту используемых ресурсов. Чем ближе это отношение к единице, тем лучше. Также под масштабируемостью понимается возможность наращивания дополнительных ресурсов без структурных изменений центрального узла системы [18].

В системе с плохой масштабируемостью добавление ресурсов приводит лишь к незначительному повышению производительности, а с некоторого «порогового» момента добавление ресурсов не даёт никакого полезного эффекта.

Она может быть:

– горизонтальной – разбиение системы на более мелкие структурные компоненты и разнесение их по отдельным физическим машинам (или их группам), и (или) увеличение количества серверов, параллельно выполняющих одну и ту же функцию. Масштабируемость в этом контексте означает возможность добавлять к системе новые узлы, серверы для увеличения общей производительности. Этот способ масштабирования может требовать внесения изменений в программы, чтобы программы могли в полной мере пользоваться возросшим количеством ресурсов;

– вертикальной – увеличение производительности каждого компонента системы с целью повышения общей производительности. Масштабируемость в этом контексте означает возможность заменять в существующей вычислительной системе компоненты более мощными и быстрыми по мере роста требований и развития технологий. Это самый простой способ масштабирования, так как не требует никаких изменений в прикладных программах, работающих на таких системах.

В разрабатываемой системе в основном используется горизонтальное масштабирование. Оно заключено в увеличении числа обрабатываемых. Данное действие может повлечь замену части оборудования либо увеличении его количество далее будут рассмотрены несколько наиболее вероятных вариантов масштабирования.

Добавление новых камер, на этажи, где уже развёрнута сеть – в данном случае затрагиваются следующие части системы:

– коммутатор, размещенный на этаже – у него могут закончится свободные порты подключения, тогда возможны 2 варианта:

– Замена коммутатора, на аналогичный по параметрам, но с большим числом подключаемых портов;

– Установка дополнительно коммутатора на этаже, который придётся либо подключать к уже установленному на этаже коммутатору. Либо подключить к серверному коммутатору, однако если у серверного не будет доступных портов, его тоже придётся заменить на схожий, но с большим числов портов.

– сервера–обработчики – для новой камеры необходимо будет разворачивать контейнер с программой обработчиком, но у серверов может

уже быть использован весь запас функциональности, в данном возможно 2 варианта действий:

– добавление ещё одного сервера–обработчика, что также может повлечь замены серверного коммутатора, при отсутствии свободных портов в оном.

– Увеличение производительности отдельного сервера обработчика, путём замены процессора и увеличения ОЗУ, однако материнская плата сервера может не поддерживать более производительные процессоры, что повлечёт за собой замену материнской платы из–за чего придётся перенастраивать весь сервер. Данный вариант действия является менее приоритетным.

Если все вышеизложенное действия по масштабированию связаны также с добавлением в систему ещё одной двери для контроля, до это также вызывает проблемы с добавлением контроллера этой двери в локальную сеть, т.к. у коммутатора контроллеров могу закончится порты, что потребует его замены на другую модель, с большим числом портов.

В качестве вертикального масштабирования используется замена вычислительных мощностей, на более производительные аналоги. А именно, замена материнской платы сервера, замена центрального процессора, замена оперативной памяти. Замена одного их перечисленных компонентов сервера может повлечь замену других компонентов.

Например, замена оперативной памяти может потребовать замену материнской платы, т.к. новая оперативная память может обладать более высокой частой памяти, нежели та частота, которую может поддерживать материнская плата. Так же схожая ситуация возникает, при решении, о замене оперативной памяти на более новое поколение, т.к. с каждым новым поколением частота памяти сильно возрастает.

Аналогичная ситуация складывается и при решении, о замене центрального процессора на более производительный. В разрабатываемом проекте ключевыми параметрами для процессора являются:

– максимальная частота – от данного параметра зависит скорость обработки данных, получаемых от камеры;

– максимальное количество потоков – параметр, который в проекте накладывает ограничения на количество обрабатываемых камер.

Т.к. максимальное число потоков процессора напрямую зависит от количества ядер процессора, то соответственно при росте числа ядер – возрастает число поддерживаемых потоков, что напрямую сказывается на

производительности процессора. На рисунке 21 мы можем видеть график роста производительности процессора при возрастании числа его ядер

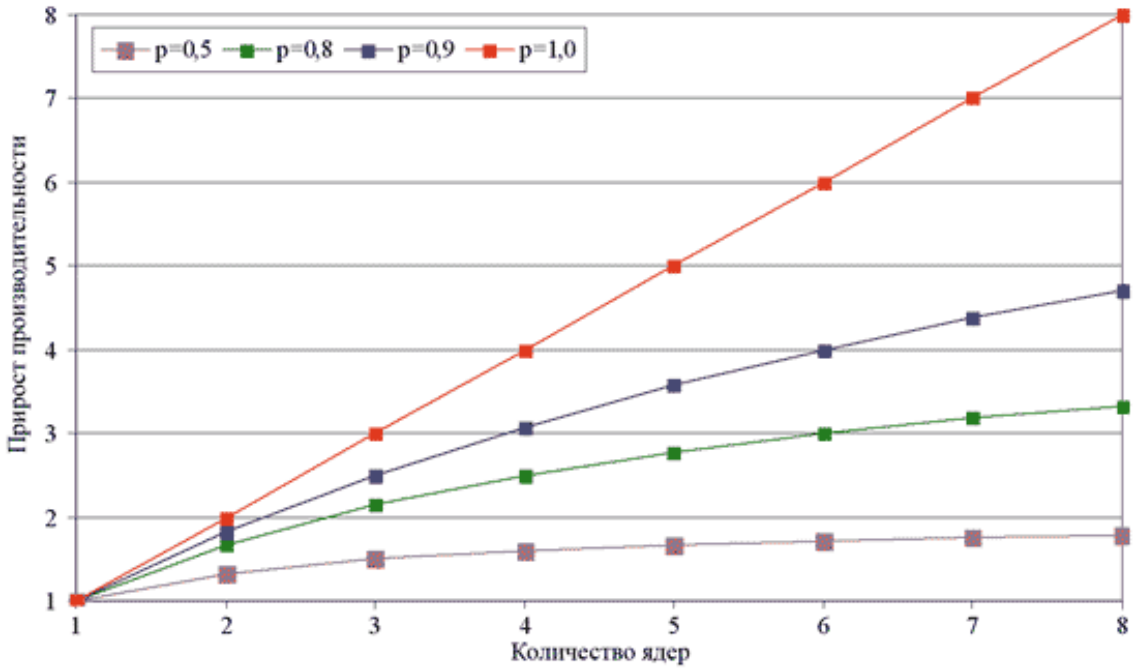


Рисунок 21 – График зависимости прироста производительности от числа ядер процессора.

Оба параметра взаимосвязаны, т.к. даже если заменять процессор на более производительный, но имеющий то же число максимальных потоков, то общая производительность системы не особо возрастёт. Однако, при замене, новый процессор выбирать исходя из увеличения количестве поддерживаемых потоков [19].

Интеграция данной системы затрагивает множество спектров работы предприятия, а т.к. система помогает решать задачи в двух сферах (безопасности и учёте рабочего времени сотрудников), то она затрагивает все спектры функционирования предприятия, но также это зависит от целей, ради которых внедряется данная система. Далее в данном пункте об интеграции системы будет рассмотрена исходя из этих целей.

Независимо от целей введения системы, потребуется интеграция системы в компьютерную сеть предприятия, это необходимо для осуществления доступа к базе данных системы с не хостинговой машины. Так же, системе нужен будет доступ в интернет, для скачивания необходимого программного обеспечения, но этот доступ необходим только на этапе установки и наладки система, а также при необходимости обновления программного обеспечения системы.

Необходимым является создание пользовательских учетных записей в базе данных для осуществления системы контролируемого доступа. Самым важным является добавление в базу данных системы изображений допущенных сотрудников.

При введении системы ради целей безопасности, то важным является интеграция системы в систему безопасности предприятия, это осуществляется разработкой программы, сканирующей базу данных системы на наличии людей в помещении, а также создание ролей для доступа в базу данных, обладающей необходимыми полномочиями (например, добавление записи, что помещение очищено, когда не сработала фиксация лица, вышедшего из помещения).

Так же, если необходимо использовать систему для использования её базы данных, для осуществления доступа в помещения, но интеграция заключается лишь в добавлении специализированной роли в базу данных системы, а также осуществить связь по сети предприятия между системами.

Если же система необходима для учета рабочего времени сотрудников, то для этого необходима интеграция базы данных системы в систему учета рабочего времени сотрудников, для чего необходимо создать в базе данных специализированную роль\и, а в системе учета рабочего времени лишь использовать «коннектор» подходящий под базу данных вводимой системы. Так же для этого можно использовать api–запросы, позволяющие получить туже информацию, что присутствует в веб–интерфейсе.

Вывод: Безопасность системы строится на использование встроенного функционала используемого оборудования и программного обеспечения. Для безопасности системы ограничивается доступ по каналам связи, также создание отдельных подсетей внутри общей сети системы позволяет для разграничения доступа к различным компонентам системы. Система поддерживает вертикальное и горизонтальное масштабирование. Горизонтальное выражено в виде увеличение количества оборудования, задействованного при расширении сети наблюдаемых проходов в помещения, что может вызвать замену уже использующегося оборудования. Вертикальное масштабирование заключается в повышении вычислительных мощностей оборудования, а именно серверов–обработчиков и сервера базы данных и веб–интерфейса. Вертикальное масштабирование выраженно в виде замены комплектующих серверов на более производительные варианты. Разрабатываемая система может быть интегрирована в другие систем предприятия, путём создания отдельных профиле доступа в базе данных системы и обеспечения физического доступа к базе данных.

6. ТЕХНИКО–ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ СИСТЕМЫ УЧЁТА ПЕРСОНАЛА В ПОМЕЩЕНИИ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИИ РАСПОЗНАВАНИЯ ЛИЦ

6.1 Краткая характеристика автоматической системы учёта персонала в помещении с использованием технологии распознавания лиц

Разрабатываемая система используется учёта входа и выхода персонала из помещений. Данный учёт может использоваться для различных задач в компаниях и на предприятиях:

– учёт рабочего времени сотрудника – подтверждение информации сотрудника, о его местонахождении в рабочее время;

– проверка лиц, входящих и выходящих из помещений – для проверки лиц, не имеющих допуска в помещения;

– на основе учёта составление графика перемещения кабинетов персонала, для повышения работоспособности персонала;

– иное (предприятие может использовать).

Исходя из вышеописанного, данная система не может использоваться только как система безопасности, также по причине того, что система не заточена под использование конкретных марок и типов оборудование (конкретная модель видеокамер, сервера–обработчика). Поэтому экономическая часть будет представлена в виде технико–экономических обоснований разработки и использования (продажи) программных средств.

Использование данной системы не привязано к конкретному предприятию.

Преимущества в разработке данный системы в сравнении с аналогами заключается:

– в использовании библиотек с открытым исходным программным кодом, что позволяет проверять библиотеки на скрытые угрозы;

– предоставление программного кода системы, что позволяет предприятию, использующему систему, модифицировать программный код системы под специфику своих задач;

– отсутствие жесткой привязки к конкретным типам оборудования, что позволяет предприятию закупать оборудование, выгодное им, а также использовать уже имеющееся оборудование.

– простота масштабируемости системы.

5.2 Расчёт на разработку ПО

Упрощённый расчёт затрат на разработку ПО следует делать в разрезе следующих статей:

– затраты на основную заработную плату разработчиков;

– затраты на дополнительную заработную плату разработчиков;

– отчисления на социальные нужды;

– прочие затраты (амортизационные отчисления, расходы на электроэнергию, командировочные расходы, арендная плата за офисные помещения и оборудование, расходы на управление и реализацию и т.п.).

Затраты на основную заработную плату команды разработчиков.

Расчёт основной заработной платы участников команды осуществляется по формуле

$$З_0 = К_{пр} \cdot \sum_{i=1}^{n} З_{д.i} \cdot t_i \ ,$$

где n – количество исполнителей, занятых разработкой конкретного ПО;

$К_{пр}$ – коэффициент, учитывающий процент премий $К_{пр} = 1.4$;

$З_{ч.i}$ – часовая заработная плата i–го исполнителя, р.;

t_i – трудоёмкость работ, выполняемых i–м исполнителем, ч.

Так как система состоит из 2 частей – вэб–интерфейс и программа анализа, но для обоих частей были задействованы разные должности. Для вэб–интерфейса – 2513–001 – инженер программист из группы «Разработчики веб– и мультимедийных приложений». Для программы анализа – 2512–001 – инженер–программист из группы «Разработчики программного обеспечения».

$З_{ч.i}$ – для 2512–001 – 15,47 р, для 2513–001 – 14,28 р.

t_i – для 2512–001 – 140 часов, для 2513–001 – 20 часов.

Затраты на дополнительную заработную плату разработчика. Включают выплаты, предусмотренные законодательством о труде (оплата трудовых отпусков, льготных часов, времени выполнения государственных обязанностей и других выплат, не связанных с основной деятельностью исполнителей), и определяется по формуле:

$$З_Д = \frac{З_о \cdot Н_д}{100\%} \ ,$$

где $З_о$ – затраты на основную заработную плату, равняются 3431,96 р.;

$Н_д$ – норматив дополнительной заработной платы $Н_Д = 20\%$.

$$З_Д = \frac{3431{,}96 \cdot 20\%}{100\%} = 686{,}39 \text{ р}.$$

Таблица 1 – Расчёт затрат на основную заработную плату разработчиков

Наименование должности разработчика	Вид выполняемой работы	Месячная заработная плата, р.	Часовая заработная плата, р.	Трудоёмкость работ, ч	Сумма, р.
1. Инженер–программист 2512–001	Разработка программного кода	2600	15,47	140	2165,8
2. Инженер–программист 2513–001	Разработка вэб–интерфейса	2400	14,28	20	285,6
Итого					2451,6
Премия (40%)					980,36
Всего основная заработная плата					3431,96

Отчисления на социальные нужды (в фонд социальной защиты населения и на обязательное страхование).

Определяются в соответствии с действующими законодательными актами по формуле:

$$Р_{соц} = \frac{(З_о + З_Д) \cdot Н_{соц}}{100\%},$$

где $Н_{соц}$ – норматив отчислений от фонда оплаты труда равный 35%;

$З_О$ – затраты на основную заработную плату, равняются 3431, 96 р.;

$З_Д$ – затраты на дополнительну заработную плату, равняются 686, 39р.

$$Р_{соц} = \frac{(3431{,}96 + 686{,}39) \cdot 35\%}{100\%} = 1441{,}423 \text{ р}.$$

Прочие затраты:

$$Р_{пз} = \frac{З_о \cdot Н_{пз}}{100\%},$$

где $Н_{пз}$ – норматив прочих затрат ($Н_{ПЗ} = 100\%$);

$З_О$ – затраты на основную заработную плату, равняются 3431, 96 р.

$$Р_{пз} = \frac{3431{,}96 \cdot 100\%}{100\%} = 3431{,}96 \text{ р}.$$

Таблица 2 – Расчёт общей суммы инвестиций

Наименование статьи затрат	Значение, р.
1. Основная заработная плата разработчиков	3431,96
2. Дополнительная заработная плата разработчиков	686,39
3. Отчисления на социальные нужды	1441,423
4. Прочие затраты	3431,96
Общая сумма инвестиций в разработку	8991,733

5.3 Оценка результата (эффекта) от использования (или продажи) ПО

Система является масштабируемой, то есть используя её в заранее определенном количестве помещений, она может быть расширена для использования в дополнительных помещениях.

Основываясь на приведённых выше особенностях системы – невозможно точно просчитать экономический эффект от использования разрабатываемой системы. Однако, если вместо системы, использовать персонал, который бы выполнял роли данной системы, то предприятию бы пришлось на каждый кабинет нанимать работника, который бы следил за входящим, либо выходящим персоналом. Данный аналог потребовал бы следующих действий:

– увеличение ежемесячного фонда оплаты труда работников предприятия;

– затраты на наём подменного персонала, при выходе сотрудников в отпуск, либо по больничному;

– что так же ведёт увеличение объёма работы бухгалтерам, начисляющим заработную плату;

– затраты на выполнение условий работы персонала (например, помещения могут быть неотапливаемыми, поэтому контролирующему персоналу потребуется отопление).

Поэтому для каждого предприятия расчёт экономической эффективности должен производится отдельно. Для этого вначале составляется список помещений, требующих наблюдения, оценить технические требования для сервера обработчика, который сможет обрабатывать требуемое количество камер и т.п.

Затраты на сырьё и материалы отсутствуют, т.к. они необходимы для изготовления единицы продукции, что не подходит для системы.

Расчёт затрат на комплектующие изделия и полуфабрикаты так же отсутствует, т.к. у нас только программный продукт.

Расчёт отпускной цены будет идти из расчёта планов продаж для компенсации затрат на разработку в течение года, с учётом получения прибыли с каждой продажи (40%).

Т.к. система может применятся на любых предприятиях, независимо от сферы деятельности – планируемое количество проданных копий в год рассчитывается из следующих условий: что в каждом районом центре Минска (118 районов в Республике) систему закупят минимум 11 организация (сумма из: 3 субъекта министерства здравоохранения, 3 производственных предприятия, 5 владельцев офисных зданий) – 118 районом умноженных на 11 организаций и округлённо до сотен в большую сторону и равняется 1300 копиям.

Таблица 3 – Расчёт себестоимости и отпускной цены единицы продукции

Наименование статьи затрат	Условное обозначение	Значение, р.	Примечание
1. Сырьё и материалы	$Р_м$	0	Т.к. у нас не производится материальная продукция
2. Покупные комплектующие изделия	$Р_к$	0	У нас продаётся только программный продукт

Продолжение таблицы 5

Наименование статьи затрат	Условное обозначение	Значение, р.	Примечание
3. Основная заработная плата производственных рабочих	$З_о$	3431,96	См. табл «Расчёт затрат на основную заработную плату разработчиков»
4. Дополнительная заработная плата производственных рабочих	$З_д$	686,39	См. подраздел 4.2
5. Отчисления на социальные нужды (отчисления в ФСЗН и обязательное страхование)	$Р_{соц}$	1441,423	См. подраздел 4.2
6. Накладные расходы	$Р_н$	3431,96	$Р_н = \frac{З_о \cdot Н_н}{100}$ $Н_н = 100\%$
7. Производственная себестоимость	$С_{пр}$	8 991,73	$С_{пр} = Р_м + Р_к + З_о + З_д + + Р_{соц} + Р_н$
8. Расходы на реализацию	$Р_р$	449,587	$Р_р = \frac{С_{пр} \cdot Н_р}{100}$, $Н_р = 5\%$
9. Полная себестоимость	$С_п$	9441,317	$С_п = С_{пр} + Р_р$
10. Плановая себестоимость продаж	$С_{пп}$	7,26	$С_{ПП} = \frac{Сп}{n}$, n=1300 – план продаж, единицы продукции
11. Плановая прибыль на единицу продукции	Пед	2,904	$П_{ед} = \frac{Спп \cdot Н_{ре}}{100}$, $Н_{ре} = 40\%$
12. Отпускная цена	Цотп	10,164	$Ц_{отп} = С_п + П_{ед}$

5.4 Экономический эффект при разработке ПО для свободной реализации на рынке информационных технологий

Экономический эффект организации–разработчика программного обеспечения в данном случае представляет собой прибыль (чистую прибыль) от его продажи множеству потребителей. Прибыль от реализации в данном случае напрямую зависит от объёмов продаж, цены реализации и затрат на разработку данного ПО.

Далее расчёт прибыли будет рассчитываться исходя из выполнения годового плана продаж, который можно посмотреть в пункте 4.3, где описано обоснование. Годовой план равен – 1300 копий.

Прибыль, полученная разработчиком от реализации ПО на рынке, поскольку организация освобождена от уплаты налога на прибыль(организация использует упрощённую систему налогообложения с уплатой НДС), рассчитывается по формуле:

$$П = Ц \cdot N - НДС - З_р .$$

Налог на добавленную стоимость определяется по формуле:

$$НДС = \frac{Ц \cdot N \cdot Н_{ДС}}{100\% + Н_{ДС}},$$

где Ц – отпускная 10,164 р (исходя из таблицы «Расчёт себестоимости и отпускной цены единицы продукции»).

N – 1300 копий, см. п. 4.3;

$Н_{дс}$ – ставка налога на добавленную стоимость согласно действующему законодательству, ($Н_{дс}$ = 20%).

$$НДС = \frac{10{,}164 \cdot 1300 \cdot 20}{100 + 20} = 2202{,}2 \text{ р},$$

где $З_Р$ – 8991,733 р (см. п. 2).

$$П = 10{,}164 * 1300 - 2202{,}2 - 8991{,}733 = 2019{,}267 \text{ р}.$$

Уровень рентабельности рассчитывается по формуле:

$$У_р = \frac{П(П_ч)}{З_р} \cdot 100\% \ ,$$

где П = 2019,267 р.

$З_Р$ – 8991,733 р (см. п. 2).

$$У_р = \frac{2019,267}{8991,733} \cdot 100\% = 22,46\ \% .$$

Средняя процентная ставка в Республике Беларусь на октябрь 2023 года для юридических лиц на срок до 1 года является 2,93% [20], что значительно меньше уровня рентабельности, что означает что проект экономически эффективен

5.5 Вывод

Планируемая прибыль, с учётом выполнения годового плана продаж, составит – 2019,267 р. Однако проект выйдет на окупаемость только при продаже минимум 929 копий (полная себестоимость делится на отпускную цену). Уровень рентабельности разработки – 22,46 %. Общая сумма затрат на разработку – 8991,733 р.

ЗАКЛЮЧЕНИЕ

В процессе разработки дипломного проекта была изучена теория распознавания лиц, а также различия и преимущества применяемых методик в данной теории. Были рассмотрены действующие системы, использующие технологию распознавания лиц, их функциональные возможности и сферы применения. Согласно техническому заданию, был выбран способ распознавания с помощью свёрточных нейронных сетей.

В ходе работы были разработаны функциональная и структурная схемы, иллюстрирующие основные составные части для реализации система и их способы взаимодействия между собой соответственно.

Были сформированы требования, согласно которым было выбрано оборудование для развёртывания системы. Далее были выбраны средства разработки программной части системы. Было выбрано программное обеспечения для развёртывания системы и разработана диаграмма развёртывания, на основе которой было сконфигурирована последовательность развёртывания системы

Был разработан программный код веб-интерфейса системы, состоящий из 2 частей: frontend и backend. Для каждой из частей был разработан программный код. Была разработана диаграмма базы данных, на основе которой была реализована структура базы данных разрабатываемой системы. Далее был разработан и реализован алгоритм работы программы обработчика. Также были реализованы механизмы интеграции и сбора статистики. Далее была представлена визуализация экранных форм веб-интерфейса системы.

Сконфигурированы параметры безопасности системы, включающие обеспечение безопасности системы на трёх уровнях: физическом, сетевом и программном. Были спроектированы варианты вертикального и горизонтального масштабирования системы.

На заключительной стадии дипломного проектирования было приведено технико-экономическое обоснование разработки системы. Которые заключались в расчёте отпускной цены программного обеспечения, свободно реализуемого на рынке, а также рентабельности разработки системы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Facial recognition system [Электронный ресурс]. - Режим доступа: https://en.wikipedia.org/wiki/Facial_recognition_system. Дата доступа: 30.10.2023.

[2] Pattern recognition [Электронный ресурс]. - Режим доступа: https://en.wikipedia.org/wiki/Pattern_recognition. Дата доступа: 30.10.2023.

[3] Artificial neural networks [Электронный ресурс]. - Режим доступа: https://en.wikipedia.org/wiki/Artificial_neural_network. Дата доступа: 30.10.2023.

[4] Свёрточная нейронная сеть [Электронный ресурс]. - Режим доступа: https://ru.wikipedia.org/wiki/Свёрточная_нейронная_сеть. Дата доступа: 30.10.2023.

[5] Python [Электронный ресурс]. - Режим доступа: https://ru.wikipedia.org/wiki/Python. Дата доступа: 1.11.2023.

[6] OpenCV Wiki [Электронный ресурс]. - Режим доступа: https://github.com/opencv/opencv/wiki. Дата доступа: 5.11.2023.

[7] face recognition [Электронный ресурс]. - Режим доступа: https://github.com/ageitgey/face_recognition. Дата доступа: 6.11.2023.

[8] HTML [Электронный ресурс]. - Режим доступа: https://en.wikipedia.org/wiki/HTML. Дата доступа: 19.11.2023.

[9] CSS [Электронный ресурс]. - Режим доступа: https://en.wikipedia.org/wiki/CSS. Дата доступа: 19.11.2023.

[10] Redis [Электронный ресурс]. - Режим доступа: https://ru.wikipedia.org/wiki/Redis. Дата доступа: 20.11.2023.

[11] PostgreSQL [Электронный ресурс]. - Режим доступа: https://en.wikipedia.org/wiki/PostgreSQL. Дата доступа: 12.11.2023.

[12] Сравнение производительности MySQL vs PostgreSQL [Электронный ресурс]. - Режим доступа: https://ru.bmstu.wiki/Сравнение_производительности_MySQL_vs_PostgreSQL. Дата доступа: 12.11.2023.

[13] RTSP [Электронный ресурс]. - Режим доступа: https://ru.wikipedia.org/wiki/RTSP. Дата доступа: 13.11.2023

[14] Контейнеризация [Электронный ресурс]. - Режим доступа: https://ru.wikipedia.org/wiki/Контейнеризация. Дата доступа: 8.11.2023.

[15] Docker [Электронный ресурс]. - Режим доступа: https://en.wikipedia.org/wiki/Docker_(software). Дата доступа: 8.11.2023.

[16] Ubuntu [Электронный ресурс]. - Режим доступа: https://ru.wikipedia.org/wiki/Ubuntu. Дата доступа: 10.11.2023.

[17] Nginx [Электронный ресурс]. - Режим доступа: https://ru.wikipedia.org/wiki/Nginx. Дата доступа: 22.11.2023.

[18] Масштабируемость [Электронный ресурс]. - Режим доступа: https://ru.wikipedia.org/Масштабируемость. Дата доступа: 22.11.2023.

[19] Эра многоядерных энергоэффективных процессоров [Электронный ресурс]. - Режим доступа: https://compress.ru/article.aspx?id=16962. Дата доступа: 25.11.2023.

[20] Динамика ставок кредитно-депозитного рынка [Электронный ресурс]. - Режим доступа: https://www.nbrb.by/statistics/creditdepositmarketrates. Дата доступа: 22.12.2023.

ПРИЛОЖЕНИЕ А
(обязательное)
Программный код проекта

```
import sys
import cv2
import face_recognition as fa_re
import datetime
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
from sqlalchemy import Boolean, Column, Integer, String, DateTime
from sqlalchemy.dialects.postgresql import BYTEA
from sqlalchemy.orm import declarative_base
import base64
from PIL import Image
import io
import numpy
import base64
from http.server import BaseHTTPRequestHandler, HTTPServer
import socketserver
#from flask import Flask
import time
import numpy as np
import setuptools


# DB settings----------------------------------------------------
db_url = 'postgresql://userP:mypass@192.168.126.130:5432/facedb'
engine = create_engine(
   db_url, pool_size=10, max_overflow=20
)
Session = sessionmaker(
   engine,
   autocommit=False,
   autoflush=False,
)
session = Session()

Base = declarative_base()

# classes for DB-------------------------------------------------
class facesDB(Base):
   __tablename__ = "faces"

   id = Column(Integer, primary_key=True)
   file = Column(BYTEA)
   face_data = Column(BYTEA)
   personal_id = Column(Integer)
```

```
class face_check(Base):
    __tablename__ = "in_out_date"

    time = Column(DateTime, default=datetime.datetime.now, primary_key=True)
    per_id = Column(Integer)
    cam_id = Column(Integer)


class undef_face(Base):
    __tablename__ = "undendified_faces"
    time = Column(DateTime, default=datetime.datetime.now, primary_key=True)
    cam_id = Column(Integer)
    file = Column(BYTEA)


def get_faces():
    #ans = session.query(facesDB).filter(facesDB.id == 0).all()
    ans = session.query(facesDB.personal_id, facesDB.face_data).all()

    answer_id = []
    answer_face = []
    for i in ans:
        bytes_converted = np.frombuffer( i[1], np.float64)
        answer_id.append(i[0])
        answer_face.append(bytes_converted)

    return answer_id, answer_face


def find_faces(known_faces, new_faces):
    for face_new in face_encodings:
        t1 = time.time()
        matches = fa_re.compare_faces(faces, face_new)

        if True in matches:
            print(f'id={ matches.index(True) }')
        else:
            print('unknown')


if __name__ == '__main__':

    ids, faces = get_faces()
    print(ids)

    video_capture = cv2.VideoCapture(0)
```

```
#video_capture = cv2.VideoCapture("rtsp://192.168.100.8:8554/webcam.h264")

process_this_frame = True

while True:

   ret, frame = video_capture.read()

   if process_this_frame:
      small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)

      rgb_small_frame = np.ascontiguousarray(small_frame[:, :, ::-1])

      face_locations = fa_re.face_locations(rgb_small_frame)

      if len(face_locations)==0:
         continue

      face_encodings = fa_re.face_encodings(rgb_small_frame, face_locations, model="small")

      t1 = time.time()

      for face_new in face_encodings:
         #time.sleep(0.5)
         matches = fa_re.compare_faces(faces, face_new)

         if True in matches:
            p_id=ids[ matches.index(True) ]
            print(f'id={ p_id }')
            face_add = face_check(
               per_id=p_id,
               cam_id =1
            )

            session.add(face_add)
            session.commit()

         else:

            face_unden = undef_face(
               cam_id =1,
               file= cv2.imencode('.png', frame)[1].tobytes()
            )
            session.add(face_unden)
            session.commit()
            print('unknown')
      print(f"{time.time()-t1}")
```

ПРИЛОЖЕНИЕ Б
(обязательное)
Функции отчётов

```
CREATE OR REPLACE FUNCTION worker_day_visits
        (IN date_f date,IN per_in integer)
    returns TABLE (
        p_date timestamp,
        cab integer,
        pos boolean
    )
    AS
$$
DECLARE
BEGIN
    return query
        select
        public.in_out_date.time as p_date,
        public.cameras.cab_id as cab,
        public.cameras.in_pos as pos
        from public.in_out_date

        INNER JOIN public.cameras
        ON public.in_out_date.cam_id=public.cameras.id
        and DATE(public.in_out_date.time)=date_f
        and public.in_out_date.per_id=per_in
        ;
end;
$$
LANGUAGE 'plpgsql'
-----------------------------------------------------------------------
CREATE OR REPLACE FUNCTION worker_day_visits_pos
```

```
        (IN date_f date,IN per_in integer, IN pos_in boolean)
    returns TABLE (
        p_date timestamp,
        cab integer
    )
    AS
$$
DECLARE
BEGIN
    return query
        select
        public.in_out_date.time as p_date,
        public.cameras.cab_id as cab

        from public.in_out_date

        INNER JOIN public.cameras
        ON public.in_out_date.cam_id=public.cameras.id
        and DATE(public.in_out_date.time)=date_f
        and public.in_out_date.per_id=per_in
        and public.cameras.in_pos=pos_in
        ;
end;
$$
LANGUAGE 'plpgsql'
-----------------------------------------------------------------------
CREATE OR REPLACE FUNCTION cab_visits
        (IN date_f date,IN cab_in integer)
    returns TABLE (
        date_o timestamp,
        per integer,
```

```
            pos boolean
        )
        AS
$$
DECLARE
BEGIN
        return query
            select

            public.in_out_date.time as date_o,
            public.in_out_date.per_id as per,
            public.cameras.in_pos as pos

            from public.in_out_date

            INNER JOIN public.cameras
            ON public.in_out_date.cam_id=public.cameras.id
            and DATE(public.in_out_date.time)=date_f
            and public.cameras.cab_id=cab_in
            ;
end;
$$
LANGUAGE 'plpgsql'
------------------------------------------------------------------------
CREATE OR REPLACE FUNCTION cab_visits_pos
            (IN date_f date,IN cab_in integer, IN pos boolean)
        returns TABLE (
            date_o timestamp,
            per integer
        )
        AS
```

```
$$
DECLARE
BEGIN
	return query
		select

		public.in_out_date.time as date_o,
		public.in_out_date.per_id as per

		from public.in_out_date

		INNER JOIN public.cameras
		ON public.in_out_date.cam_id=public.cameras.id
		and DATE(public.in_out_date.time)=date_f
		and public.cameras.cab_id=cab_in
		and public.cameras.in_pos=pos
		;
end;
$$
LANGUAGE 'plpgsql'
-----------------------------------------------------------------------
CREATE OR REPLACE FUNCTION pass_visits
		(IN date_f date, IN pass integer,IN cab_in integer)
	returns TABLE (
		date_o timestamp,
		per integer,
		pas_bool boolean
	)
	AS
$$
DECLARE
```

```
BEGIN
	return query
		select

		public.in_out_date.time as date_o,
		public.in_out_date.per_id as per,
		public.cameras.in_pos as pas_bool

		from public.in_out_date

		INNER JOIN public.cameras
		ON public.in_out_date.cam_id=public.cameras.id
		and DATE(public.in_out_date.time)=date_f
		and public.cameras.pass_num=pass
		and public.cameras.cab_id=cab_in
		;
end;
$$
LANGUAGE 'plpgsql'
-----------------------------------------------------------------
CREATE OR REPLACE FUNCTION pass_visits_pos
		(IN date_f date, IN pass integer,
		 IN cab_in integer, IN pass_in boolean)
	returns TABLE (
		date_o timestamp,
		per integer,
		pas_bool boolean
	)
	AS
$$
DECLARE
```

```
BEGIN
	return query
		select

		public.in_out_date.time as date_o,
		public.in_out_date.per_id as per,
		public.cameras.in_pos as pas_bool

		from public.in_out_date

		INNER JOIN public.cameras
		ON public.in_out_date.cam_id=public.cameras.id
		and DATE(public.in_out_date.time)=date_f
		and public.cameras.pass_num=pass
		and public.cameras.cab_id=cab_in
		and public.cameras.in_pos=pass_in
		;
end;
$$
LANGUAGE 'plpgsql'
-----------------------------------------------------------------
```

Наименование	*Кол.*	*Примечание*
Сетевое оборудование		
Utepo UTP-7224E-POE-L2	6	
D-Link DES-3200-28/C1A	1	
Сервер-обработчик		
MulitOffice 7C137D32S512IMG7	3	
Сервер веб-интерфейса		
Jet Office 7i10700D8HD05SD12VGALW50	1	
Камеры		
IP-камера Hikvision DS-2CD2046G2-IU/SL(C)	56	
Программное обеспечение		
Контейнеризатор приложений Docker	4	
Оркестратор контейнеров Docker-Compose	4	
СУБД PostgreSQL	1	
Веб-сервер Nginx	1	
Резидентная СУБД Redis	1	
Сетевые контроллеры дверей		
ZKTeco C3-100	28	
Электромагнитные замки		
ZKTeco CM-280	28	

					ГУИР.466452.007 ПО
Изм.	Л.	№ докум.	Подп.	Дата	
Разраб.		Вертинский			Автоматическая система учёта персонала в помещении с использованием технологии распознавания лиц Перечень оборудования
Провер.		Шведова			
Т.контр.		Шведова			
Н.контр.		Крупская			
Утв.		Марков			

Лит	Лист	Листов
Т	76	77

Кафедра СУ

Группа 082471

Обозначение	*Наименование*	*Доп. сведения*
	Текстовые документы	
БГУИР ДП 1-53 01 07 004 ПЗ	Пояснительная записка	76 с.
	Отзыв руководителя	
	Рецензия	
	Графические документы	
ГУИР.466152.001 ПД	Автоматическая система учёта персонала в помещении с использованием технологии распознавания лиц Структурная схема	Формат А1
ГУИР.466152.002 ПД	Модули взаимодействия проходов с серверами Схема структурная	Формат А1
ГУИР.466152.003 ПД	Автоматическая система учёта персонала в помещении с использованием технологии распознавания лиц Диаграмма развёртывания	Формат А1
ГУИР.466152.004 ПД	Автоматическая система учёта персонала в помещении с использованием технологии распознавания лиц Диаграмма базы данных	Формат А1
ГУИР.466152.005 ПД	Программа обработчик Схема алгоритма	Формат А1
ГУИР.466152.006 ПД	Автоматическая система учёта персонала в помещении с использованием технологии распознавания лиц Экранные формы	Формат А1

					БГУИР ДП 1-53 01 07 004 Д1
Изм.	Л.	№ докум.	Подп.	Дата	
Разраб.		Вертинский			Автоматическая система учёта персонала в помещении с использованием технологии распознавания лиц Ведомость дипломного проекта
Пров.		Шведова			
Т.контр.		Шведова			
Н.контр.		Крупская			
Утв.		Марков			

Лит	Лист	Листов
Т	77	77

Кафедра СУ

Группа 082471