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».

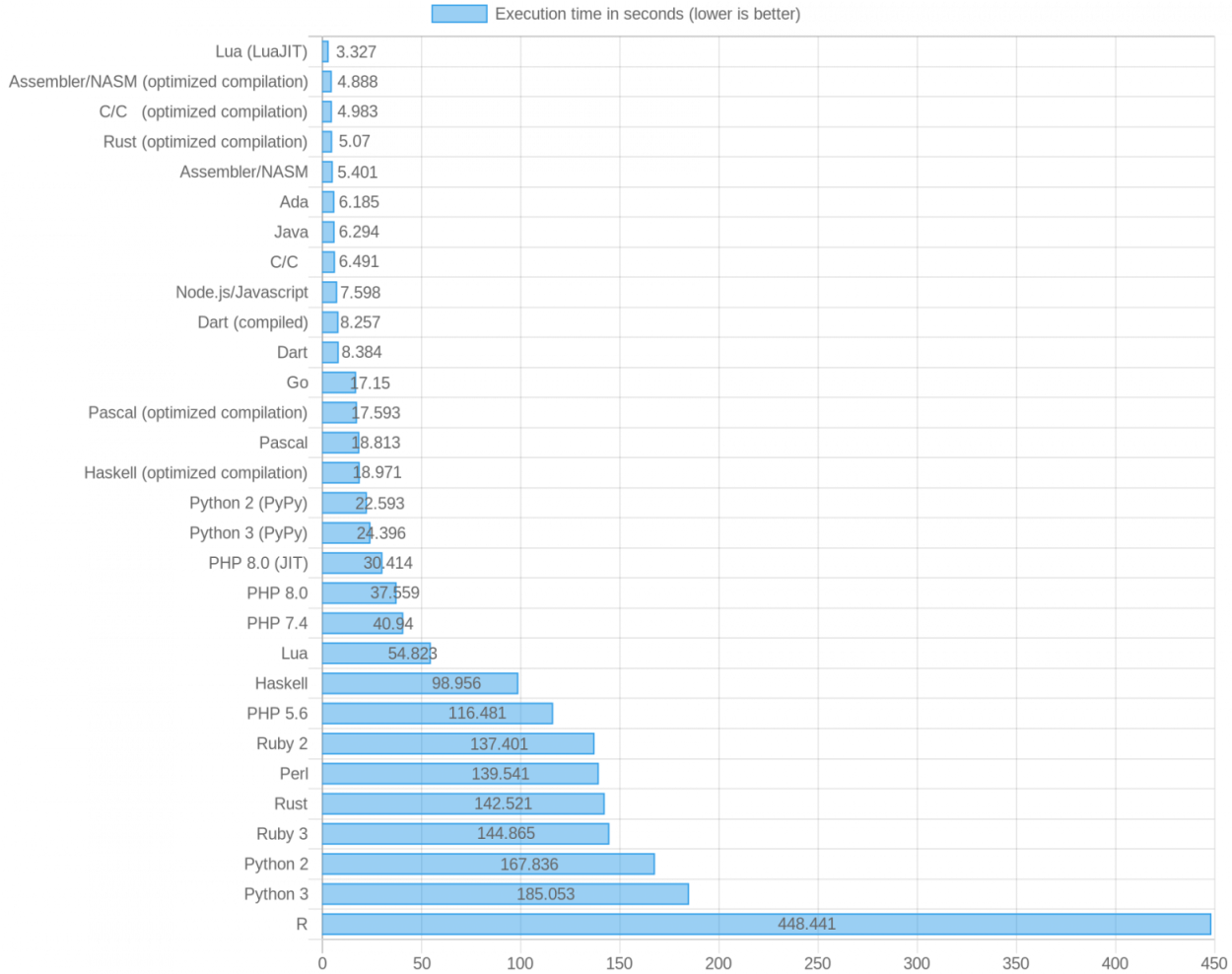

Рисунок 4 – График скорости выполнения теста на основе перебора простых чисел для различных языков программирования

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». 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].

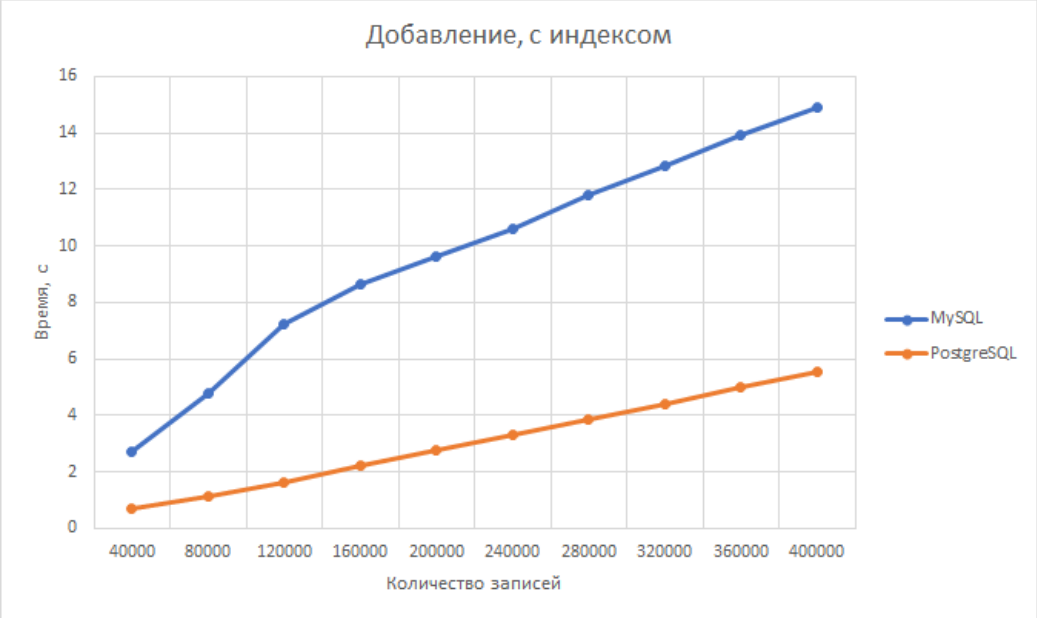

Рисунок 5 – График производительности при выполнении добавлении записей в таблицу.

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].

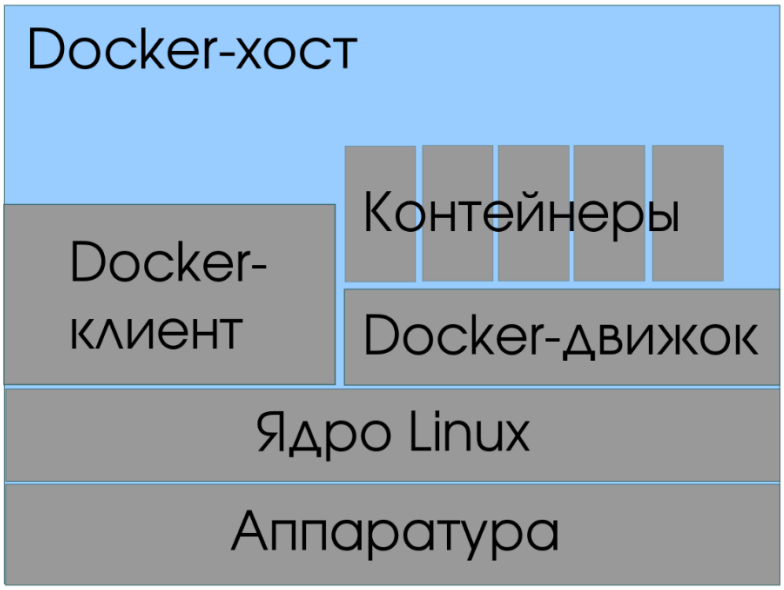

Рисунок 7 – Docker на физическом Linux–сервере

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.

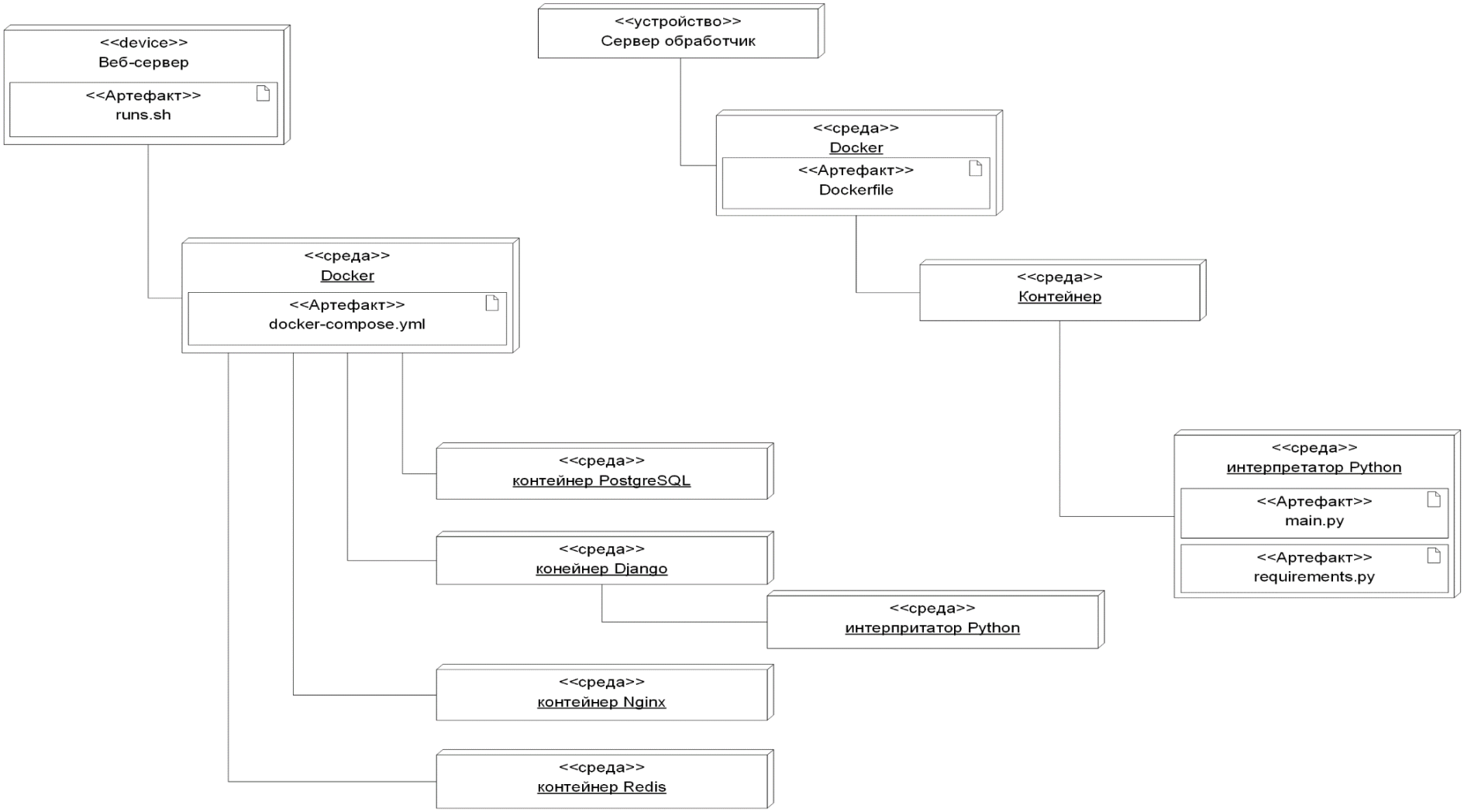

Рисунок 8 – Диаграмма развёртывания