

## **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 представлен в виде api-сервера, реализованного с помощью фреймворка «Django». Он позволяет сформировать таблицы в базе данных, путём описания их в виде классов в Python, которые в дальнейшем программа будет использовать как интерфейс доступа к базе данных и на основе которого фреймворк создаст таблицы в базе данных. Фреймворк позволяет обрабатывать api-запросы к базе данных, которые он будет получать с frontend части, далее забирать информацию с базы данных (которая была запрошена в запросе), а потом отправляет её в виде ответа на api-запрос. На схеме диаграммы классов из списка графических документов представлены классы, создаваемые фреймворком. Так же в нём производится настройка администраторского доступа к базе 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 которых 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 которой 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 «cvddb», которая хранит в себе всю информацию, необходимую на работы система – список сотрудников, их биометрия лиц и т.д. На рисунке 13 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 «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.

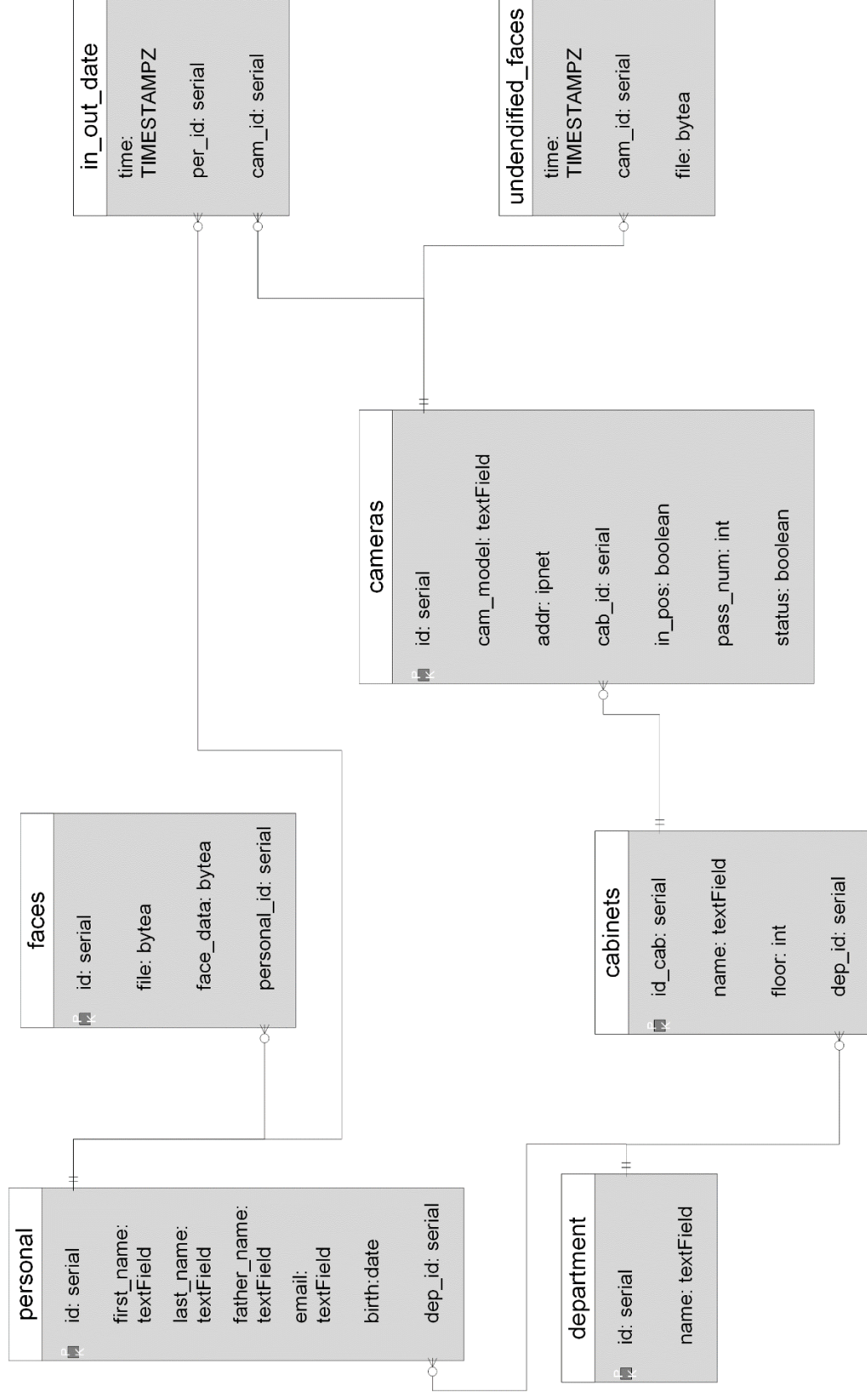


Рисунок 13 – Диаграмма базы данных

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;
- 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.

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`.

### **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` Посещения работником всех помещений за определённую дату, входные переменные: `date_f` – день, за который нужно отфильтровать значения, `per_in` – id работника;
- `worker_day_visits_pos` возвращает список помещений, в которые заходил/выходил в зависимости от переменной, переменные аналогичны 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.

#### 4.4 Разработка алгоритма работы и программного обеспечения системы

Алгоритм представлен на рисунке и листе графического материала ГУИР.466152.003 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 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 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 лиц. Если на запрос пришло 0 векторов, то программа 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 проверка на получения данных с камеры. При отсутствии данных система изменяет статус камеры (за обработку данных с которой 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 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 вектор 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.

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 Проектирование последовательности развёртывания системы

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 производиться 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 15 и листе графического материала ГУИР.466152.005 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 фреймворка «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.

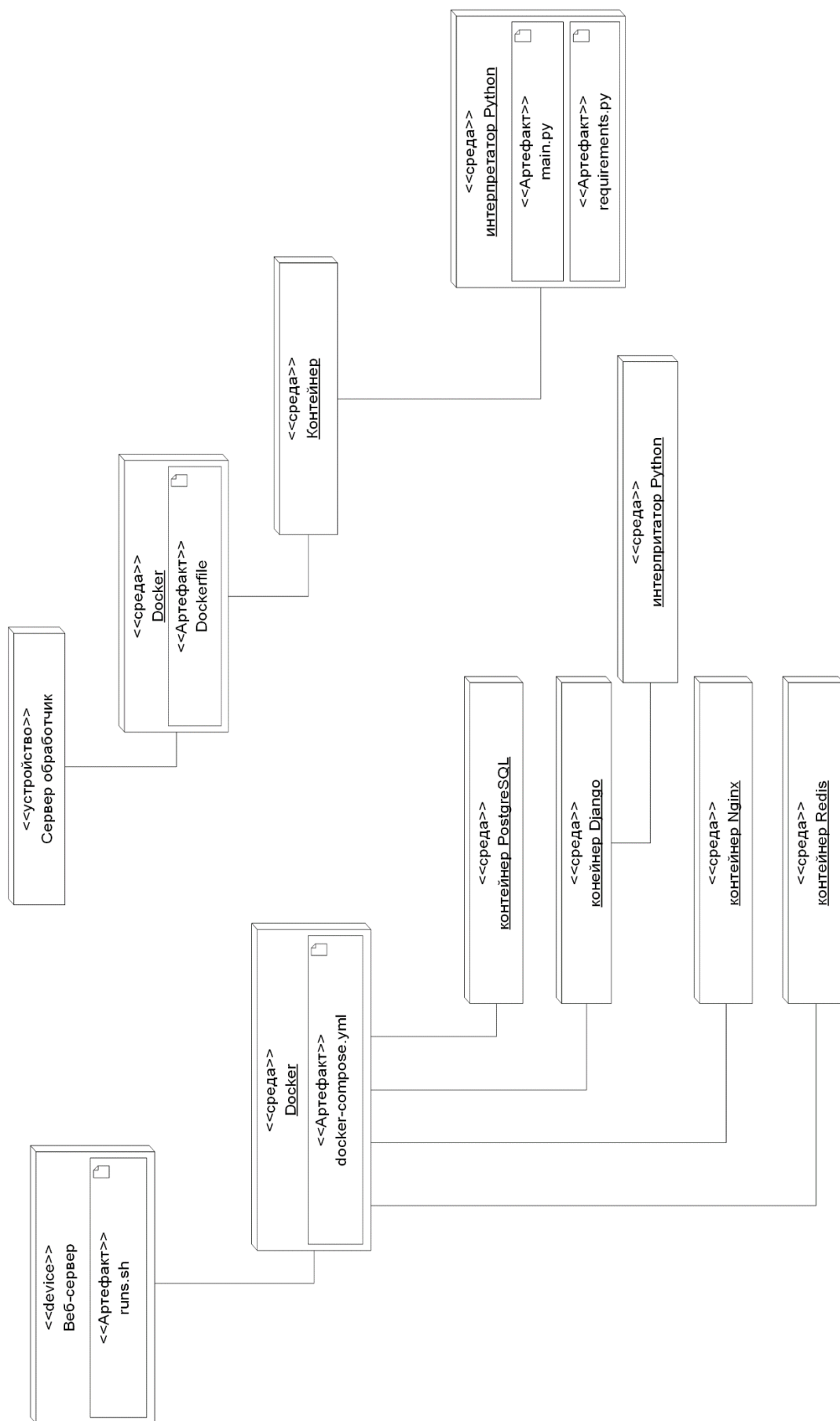


Рисунок 15 – Диаграмма развёртывания

## 4.6 Экранные формы веб-интерфейса

Веб-интерфейс по уровню допуска подразделяется на 2 части:

- пользовательская часть;
- административная часть.

Пользовательская часть предназначена для предоставления свободного доступа к части таблиц в базе данных проекта, а именно:

- список персонала - раздел «Personal Info»;
- список кабинетов - раздел «Cabinets status»;
- список камер - раздел «Camera status»;
- список посещения персонала кабинетов - раздел «In-Out status».

Более подробная информация, о возможности, пользовательской части предоставлена в схеме вариантов использования графической в списке графических документов. Далее подробнее рассмотрены элементы графического листа ГУИР.466152.007 ПД.

Страница содержит следующие элементы:

- «шапка» - верхняя часть страницы (см. рисунок 16), содержащая название открытого раздела;



Рисунок 16 – «Шапка» страницы

- навигационная панель (см. рисунок 17), предназначенная для переключения между разделами;

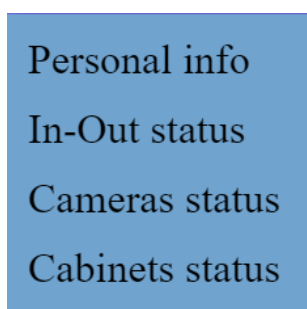


Рисунок 17 – Навигационная панель

- кнопка «Login» - находящаяся в верхнем правом углу страницы (см. рисунок 18) и открывающая окно авторизации для входа в административную часть.

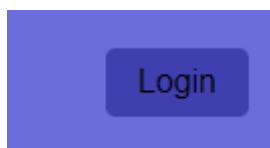


Рисунок 18 – Кнопка авторизации административного доступа

Остальная часть страницы предназначена для отображения разделов страницы. Далее описывается отображение отделов страницы.

Раздел «Personal info» - предназначен для отображения списка персонала (см. рисунок 19), список представлен в виде таблицы с 3 колонками:

- колонка «id pers» - идентификационный номер работника;
- колонка «name» - ФИО сотрудника;
- колонка «dep id» - название отдела, в котором работает сотрудник.

Так же этот раздел содержит окно поиска персонала по имени и/или идентификационному номеру в базе данных (см. рисунок 20).

id pers	name	dep id
1	kir lname fname	dep1
2	user1 lname fname	dep3
3	kir lname fname	dep3

Рисунок 19 – Список персонала

Рисунок 20 – Окно поиска

Раздел «In-Out status» - предназначен для отображения регистрации посещений помещений в виде таблицы (см. рисунок 21) со следующими столбцами:

- столбец «id» - идентификационный номер посещения;
- столбец «timedate» - дата и время посещения;
- столбец «direction» - направление, красное – из кабинета, зелёное – в кабинет;
- столбец «per id» - идентификационный номер опознанной персоны;
- столбец «cab id» - идентификационный номер кабинета, где зафиксировалось событие;
- столбец «cam id» - идентификационный номер камеры, зафиксировавшей лицо.

id	timedate	direction	per id	cab id	cam id
1	2022-12-28T11:50:08.688797Z	●	1	1	1
2	2022-12-28T11:50:19.978811Z	●	1	1	2

Рисунок 21 – Список посещения помещений

Раздел «Cameras status» предназначен для отображения списка камер, задействованных в системе, в виде таблицы (см. рисунок 22) со следующими столбцами:

- столбец «id cam» - идентификационный номер камеры;
- столбец «cam model» - название модели камеры;
- столбец «cab id» - идентификационный номер кабинета, в котором задействована камера;
- столбец «in pos» - позиция камеры, относительно направления входа в кабинет: зеленый цвет – в кабинет, красный цвет – из кабинета.

Так же этот раздел содержит окно поиска камер по модели и/или идентификационному номеру в базе данных, окно полностью повторяет окно поиска персонала (см. рисунок 20).

id cam	cam model	cab id	in pos
1	model1	cab1	●
2	cam2	cab1	●

Рисунок 22 – Список камер

Раздел «Cabinets status» предназначен для отображения списка помещений, представленного в виде таблицы (см. рисунок 23) со следующими столбцами:

- столбец «id cab» - идентификатор помещения;
- столбец «name» - название кабинета;
- столбец «floor» - номер этажа, где расположено помещение;
- столбец «dep id» - идентификатор отдела, которому принадлежит сообщение.

Так же этот раздел содержит окно поиска помещений по названию и/или идентификационному номеру в базе данных, окно полностью повторяет окно поиска персонала (см. рисунок 20).

id cab	name	floor	dep id
1	cab1	1	dep1
2	cab2	2	dep2
3	cab3	3	dep3
4	cab4	9	dep2

Рисунок 23 – Список камер

Административная часть веб-интерфейса вначале представляет собой окно авторизации (см. рисунок 24).

После прохождения авторизации, открывается главная страница администратора (см. рисунок 25), оно предназначено для организации доступа к базе данных, а также его заполнения.

Она представлена в виде нескольких разделов:

- раздел «AUTH TOKEN» - раздел создания ключей доступа, для связи базы данных с другими системами;
- раздел «AUTHENTICATION AND AUTHORIZATION» - раздел для создания пользователей базы данных и объединения их в группы;
- раздел «CORE» - предназначен для работы с таблицами базы данных, а именно: просмотр данных, редактирование, добавление новых данных в таблицы.

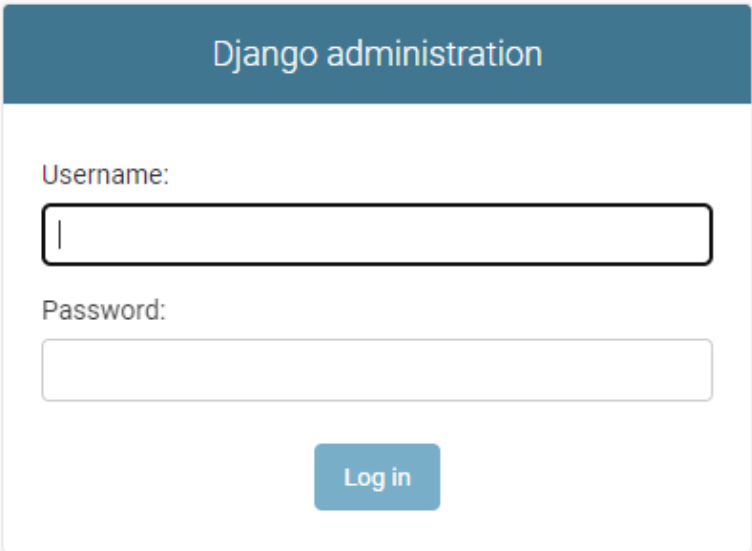
Каждый из перечисленных разделов содержит наборы таблиц, строки в которых можно редактировать либо добавлять новые.

На рисунке 26 представлен пример страницы работы с таблицей «cabinets». На которой представлен список существующих строк в таблице,



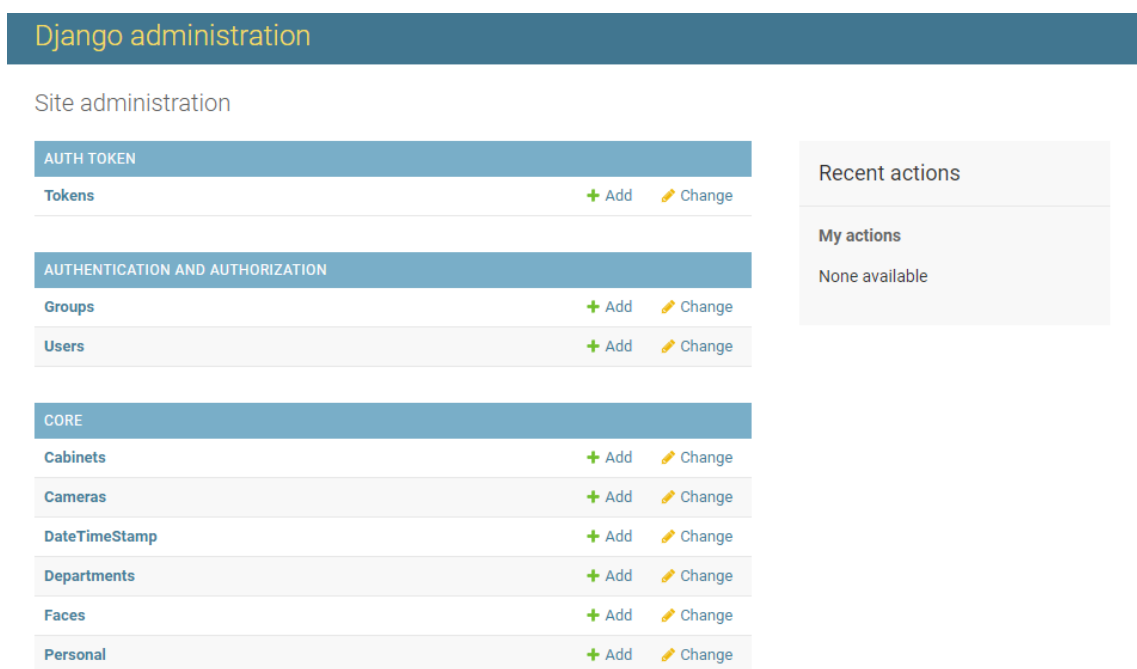
окно поиска, которое производит поиск по всем столбцам таблицы, кнопка добавления новой строки в таблицу.

На рисунке 27 представлена страница добавления новой строки в таблицу, на примере добавления новой строки в таблицу «Cabinets».



The image shows the Django administration login interface. It features a dark blue header with the text "Django administration". Below the header, there are two input fields: "Username:" and "Password:". The "Username:" field is currently empty, and the "Password:" field is also empty. Below the input fields, there is a blue button labeled "Log in".

Рисунок 24 – Окно авторизации администратора



The image shows the Django administration main page. It features a dark blue header with the text "Django administration". Below the header, there is a section titled "Site administration". This section contains three sub-sections: "AUTH TOKEN", "AUTHENTICATION AND AUTHORIZATION", and "CORE". Each sub-section contains a list of items with "Add" and "Change" links. The "AUTH TOKEN" section contains "Tokens". The "AUTHENTICATION AND AUTHORIZATION" section contains "Groups" and "Users". The "CORE" section contains "Cabinets", "Cameras", "DateTimeStamp", "Departments", "Faces", and "Personal". To the right of the "Site administration" section, there is a "Recent actions" section with a "My actions" subsection, which currently shows "None available".

AUTH TOKEN	
Tokens	<a href="#">+ Add</a> <a href="#">Change</a>

AUTHENTICATION AND AUTHORIZATION	
Groups	<a href="#">+ Add</a> <a href="#">Change</a>
Users	<a href="#">+ Add</a> <a href="#">Change</a>

CORE	
Cabinets	<a href="#">+ Add</a> <a href="#">Change</a>
Cameras	<a href="#">+ Add</a> <a href="#">Change</a>
DateTimeStamp	<a href="#">+ Add</a> <a href="#">Change</a>
Departments	<a href="#">+ Add</a> <a href="#">Change</a>
Faces	<a href="#">+ Add</a> <a href="#">Change</a>
Personal	<a href="#">+ Add</a> <a href="#">Change</a>

Recent actions

My actions

None available

Рисунок 25 – Главная страница администратора

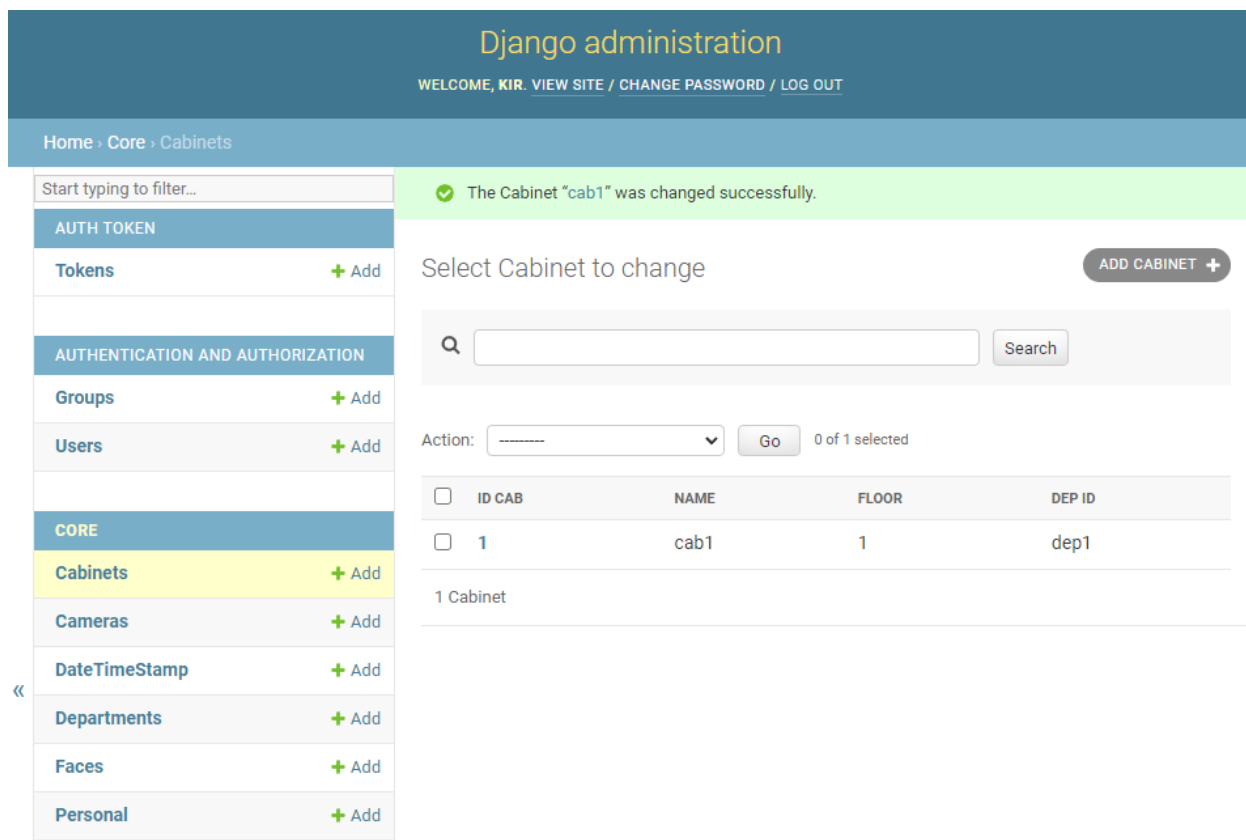


Рисунок 26 – Страница таблицы «Cabinets»

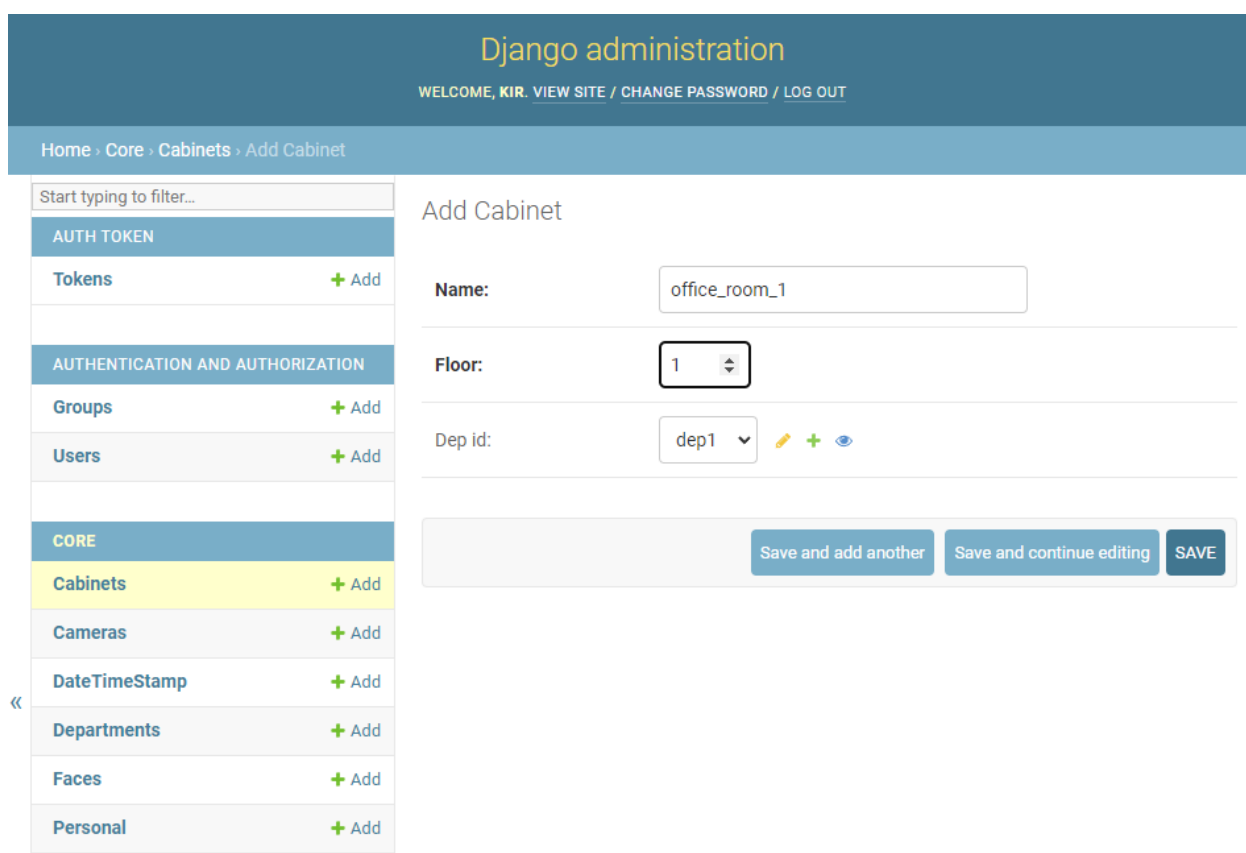


Рисунок 27 – Пример страницы добавления строки в таблицу «Cabinets»

На рисунке 28 пример того, как в базе данных сохраняется кадр с неопознанным лицом.

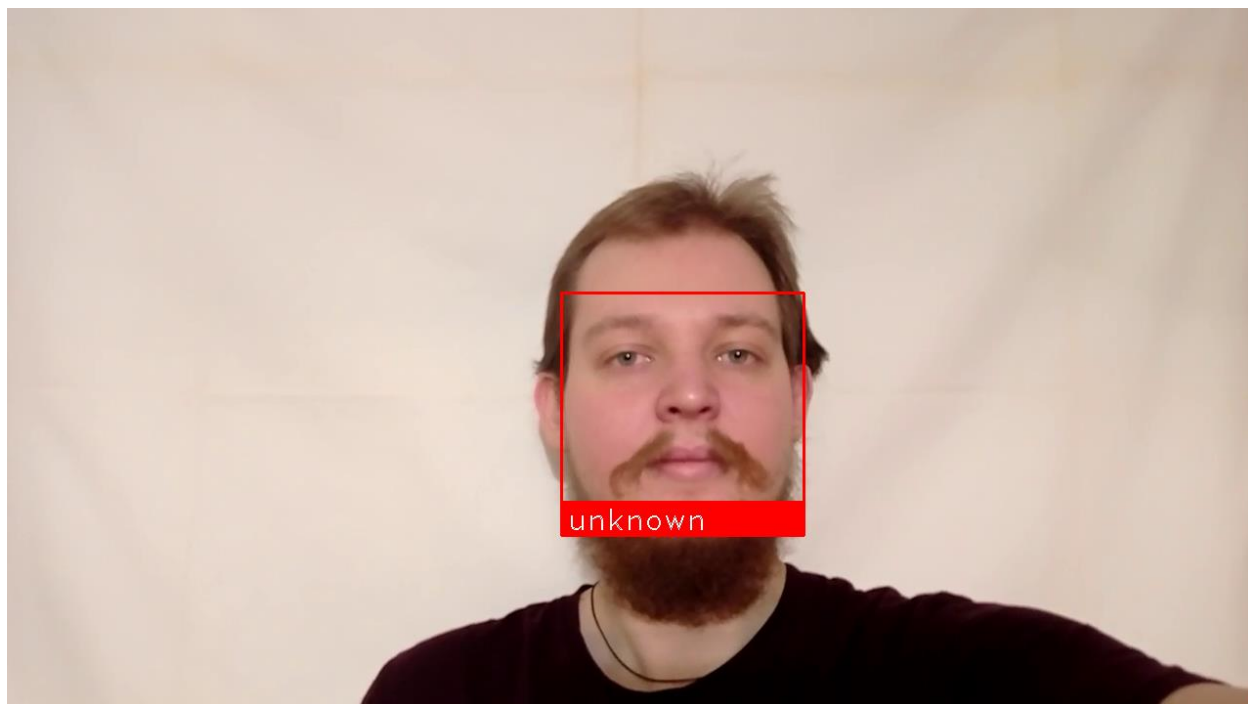


Рисунок 28 – Пример изображения с неопознанным лицом

Вывод: Система состоит из двух частей - веб-интерфейс и обработчик-камеры; веб-интерфейс реализован с помощью фреймворка «Django», 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 потоков: получение кадра из видеопотока камеры, нахождение лица на кадре, сравнение лица с лицами из базы данных. Разработана диаграмма развёртывания, иллюстрирующая порядок развёртывания программной части системы. Описаны экранные формы веб-интерфейса системы.