**4. РАЗРАБОТКА АЛГОРИТМОВ И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ АВТОМАТИЧЕСКОЙ СИСТЕМЫ УЧЁТА ПЕРСОНАЛА**

**3.1 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 данным 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 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 разделения программы на несколько поток используется библиотека «Thread», являющейся стандартной библиотекой языка.

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 13 и листе графического материала ГУИР.466152.006 ПД представлена диаграмма базы данных. База данных состоит из следующих таблиц:

- таблица «user»;

- таблица «department»;

- таблица «personal»;

- таблица «faces»;

- таблица «cabinets»;

- таблица «cameras»;

- таблица «in\_out\_date».

Таблица «user» предназначена для хранения всего списка физических лиц, состоит из 5 колонок:

- поле first\_name - типа text, хранит имя работника;

- поле last\_name - типа text, хранит фамилию работника;

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

- поле age - типа int, хранит возраст работника.

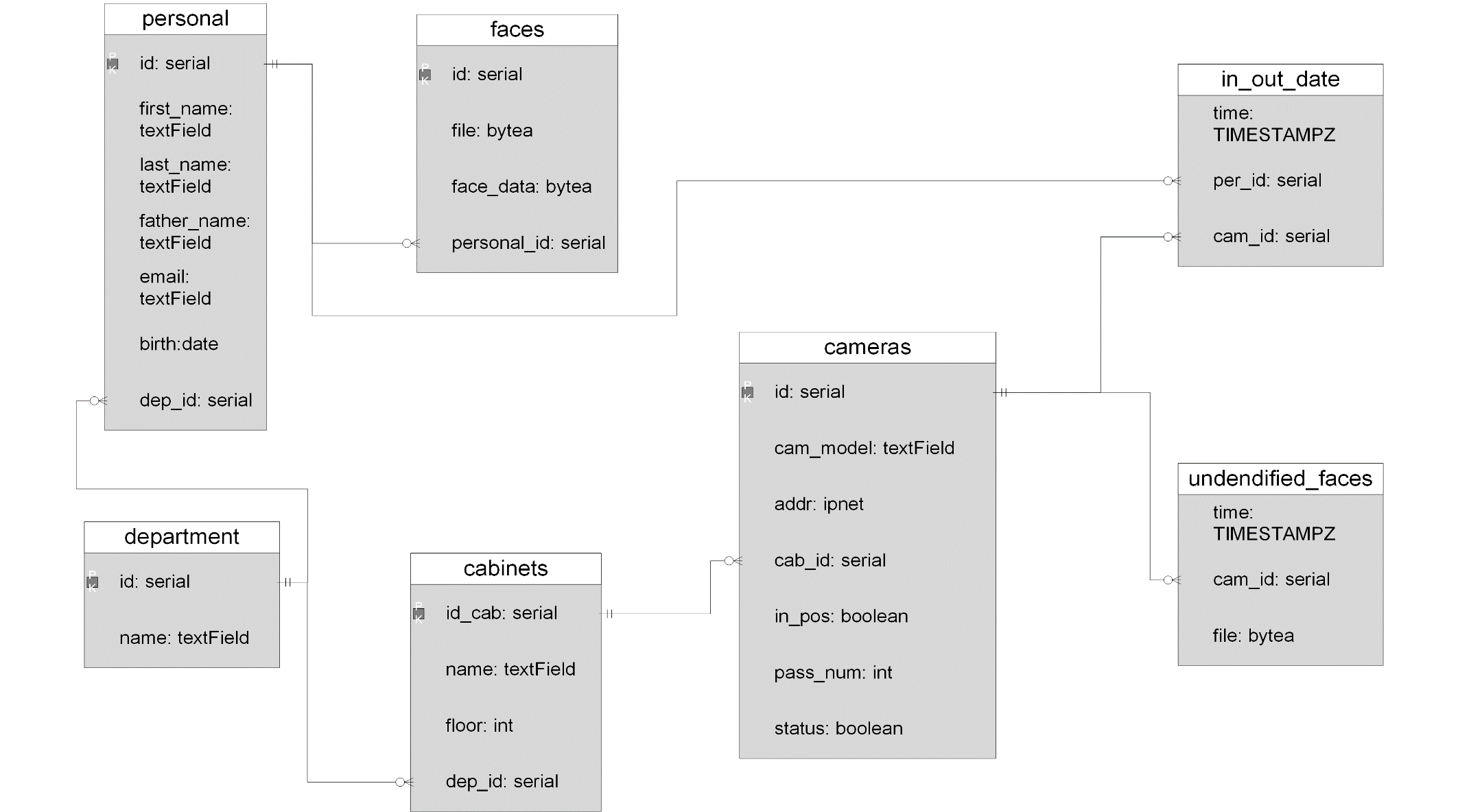


Рисунок 13 – Диаграмма базы данных

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 3 колонок:

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

- поле name – типа serial, ссылка на имя сотрудника;

- 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 5 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 status - типа bool, хранит значение параметра работоспособности камеры.

Таблица «in\_out\_date» предназначена для фиксирования информации, о входящих и выходящих сотрудников, состоит из 4 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 cab\_id – типа serial, хранит ссылку на кабинет, к которому приписана камера, определившая лицо;

- поле cam\_id – типа serial, хранит ссылку на камеру, к которому приписана камера;

- поле direction - типа bool, хранит положительное значение, если сотрудник заходио в кабинет.

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.

**3.3 Разработка программного обеспечения системы**

Вначале программного кода сервера обработчика, состоящего из одного файла, идёт включение необходимых библиотек для функционирования приложения, а именно:

- библиотека «OpenCV»;

- библиотека «face\_recognition»;

- библиотека «Thread»;

- библиотека «datetime»;

- библиотека «sys».

Далее представлено описание главного класса, который будет отвечать за обработку видеопотоков с камеры - ThreadedCamera. Класс имеет следующие внутренние переменные:

- переменная known\_faces - внутренний массив биометрических данных лиц, которые загружаюся из базы данных;

- переменная capture - объект, представляющий интерфейс для взаимодействия с захватываемым видеопотоком;

- переменная frame - хранит, считываемый из потока;

- переменная threadingFind - объект, отвечающий за запуск в отдельном потоке метода find\_face;

- переменная thread - объект, отвечающий за запуск в отдельном потоке метода update;

- переменная thread\_send - объект, отвечающий за запуск в отдельном потоке метода send\_faces;

- переменная founded\_faces - массив, хранищий список биометрических данных лиц, обнаруженных на кадре;

- переменная db\_connector - объект, предоставляющий интерфейс, для работы с базой данных (загрузка новых лиц, отправка данных в базу данных)ю

Класс состоит из следующих методов:

- метод init – конструктор класса, создаётся при объявлении переменной класса, в этом методе реализовано объявление переменных необходимых для создания видеопоток (хранящие данные видеопотока), хранения данных о параметрах лиц, так же там идёт инициализация дополнительных программных потоков, для увеличения производительности системы, а так же запуск одного из потоков (поток считывания изображения с видеопотока камеры);

- метод update - метод, в котором происходит считывания видео потока, считывания кадров после запуска видео потока, а также запуск программного потока для анализа лиц с кадров, данный метод и запускается в отдельном программном потоке;

- метод show\_frame - предназначен для показа обработанных кадров в окне;

- метод find\_face - метод запускаемый в отдельном поток, назначение данного поток состоит в считывании копии текущего кадра, а затем его обработка, для дальнейшего его анализа на поиск лиц в кадре (сжимание изображения), при обнаружении лиц в кадре, система анализирует лица со списком сохранённых лиц в памяти, при наличии данных лиц во внутреннем массиве known\_faces запуска;

- метод update\_faces - метод, также работающий в отдельном потоке и состоящий из бесконечного цикла, который реализует обновление внутреннего массива known\_faces раз в сутки, данный метод реализован без использования триггеров, т.к. триггеры не запускают функции в отдельных потоках;

- метод send\_faces - метод, использующий db\_connector, чтобы отправить в базу данных информацию об опознанных лицах или о лицах, чьи лица не находятся во внутреннем массиве.

После описание класса идёт описание главной функции программы- main. В неё в начале идёт инициализация объекта класса ThreadedCamera с предоставлением ему в конструктор аргумента, содержащего адрес видео поток камеры, далее идёт бесконечный цикл, в котором в обработчике исключений вызывается метод show\_frame, класса ThreadedCamera. При прерывании цикла вызывается функция destroyAllWindows библиотеки «OpenCV», которая закрывает все окна.

Весь программный код сервера-обработчика можно рассмотреть в приложении А.

Программный код сервера веб-интерфейса построен на использовании фреймоворка «Django», который представляет собой набор реализующих автоматическое создание таблиц в базе данных PostgreSQL, так же позволяет отправлять данные в таблицы, напрямую не используя коннектор базы данных. Однако выше описанный программный код лишь формирует «backend» часть, то есть лишь предоставляет api, которое необходимо задействовать, чтобы на веб-странице проекта отображалась информация.

Часть веб-интерфейса «frontend» пользовательского интерфейса представлена в виде 3 файлов:

- файл index.html -отвечающий за расположение элементов интерфейса страницы;

- файл style.css – отвечающий за внешнее оформление различных частей интерфейса страницы;

- файл app.js – отвечающий за связывание разделов страницы, отвечающих за отображение информации для пользователя, с данными, которые возвращают api-запросы, отправленные на «backend».

Административная часть не требует отдельной разработки схожих файлов.

На рисунке 14 и листе графического материала ГУИР.466152.002 ПД изображена диаграмма классов, отражающая взаимодействие программных классов веб-интерфейса между собой.

Программный код веб-интерфейса приведён в приложении А.

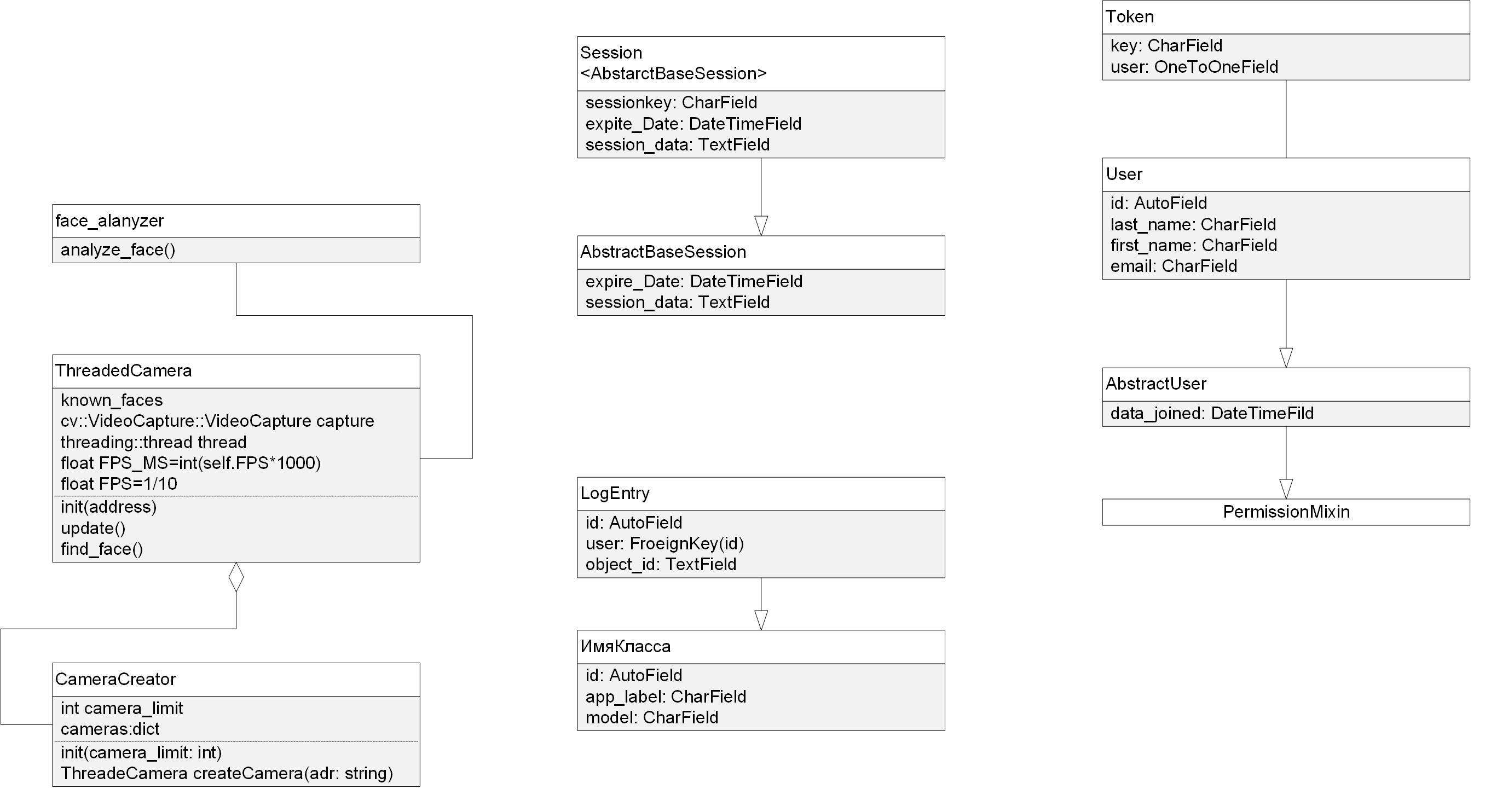


Рисунок 14 – Диаграмма классов

**3.4 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 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 15 и листе графического материала ГУИР.466152.004 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.

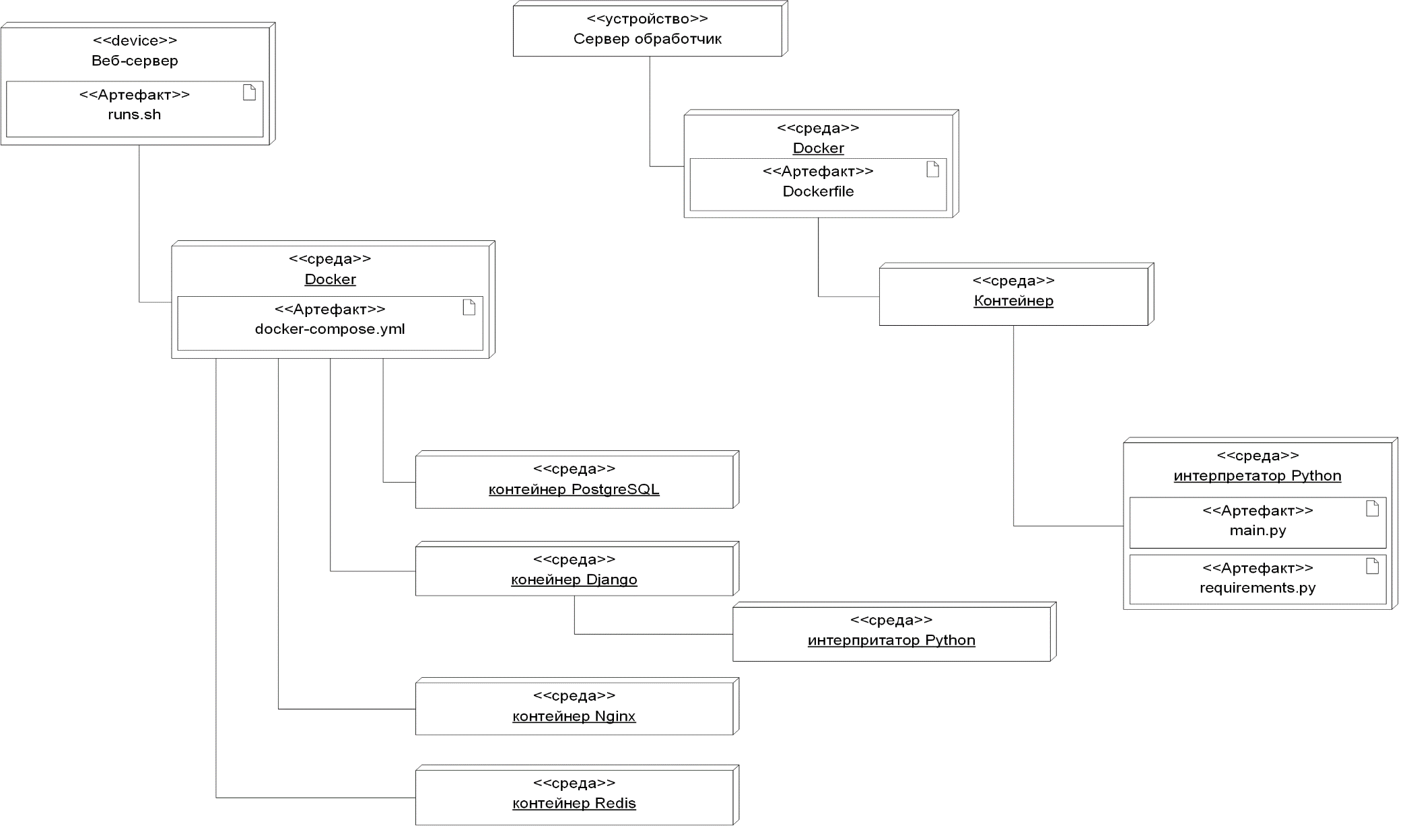


Рисунок 15 – Диаграмма развёртывания

**3.5 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 возможность, 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;

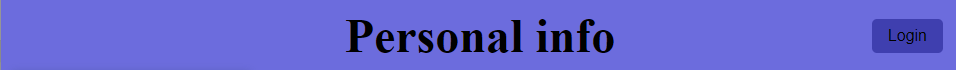


Рисунок 16 – «Шапка» страницы

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

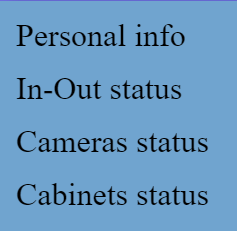


Рисунок 17 – Навигационная панель

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

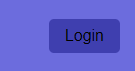


Рисунок 18 – Кнопка авторизации административного доступа

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 ip» - 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).

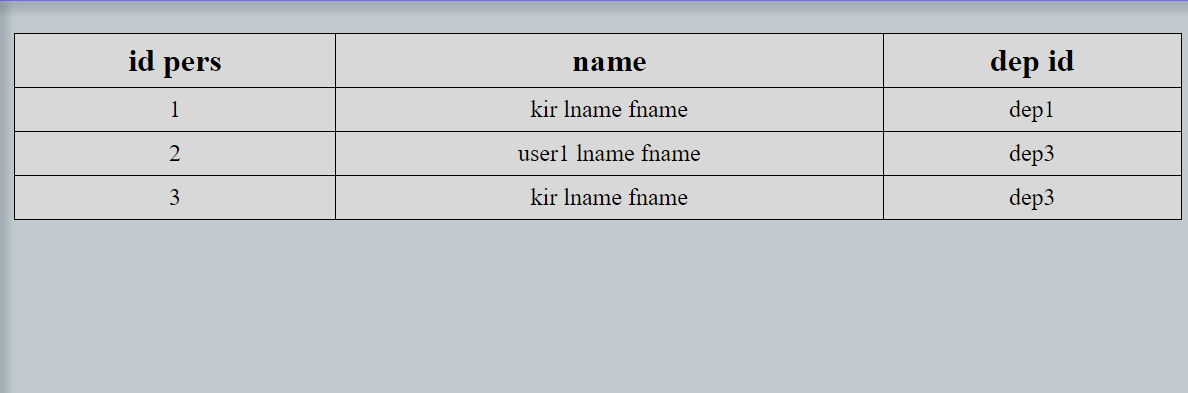


Рисунок 19 – Список персонала

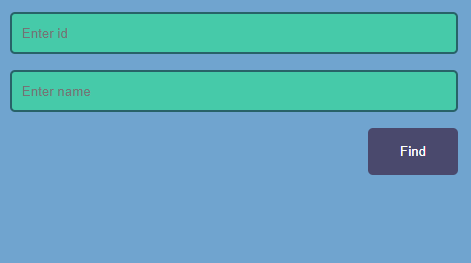


Рисунок 20 – Окно поиска

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.

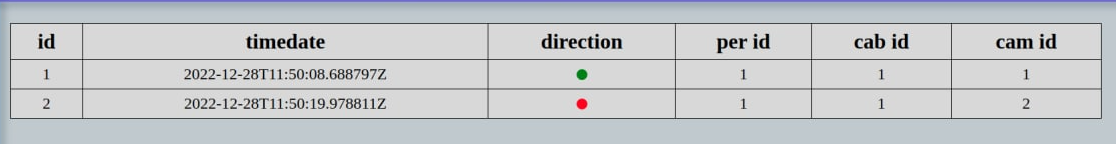


Рисунок 21 – Список посещения помещений

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).

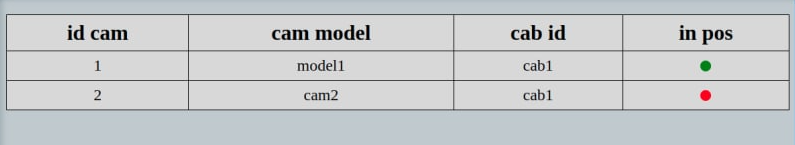


Рисунок 22 – Список камер

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 название и/или идентификационному номеру в базе данных, окно полностью повторяет окно поиска персонала (см. рисунок 20).

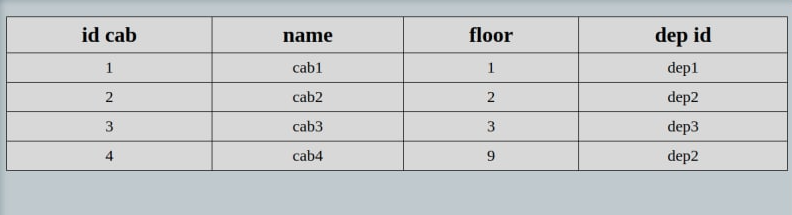


Рисунок 23 – Список камер

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 новы.

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».

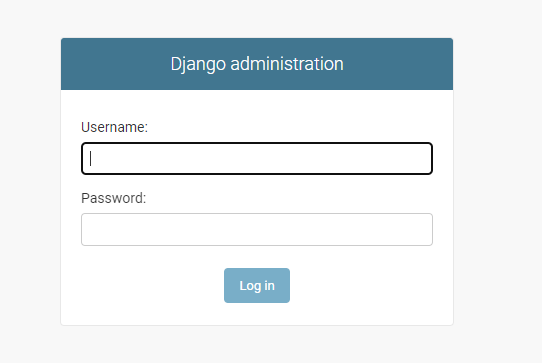


Рисунок 24 – Окно авторизации администратора

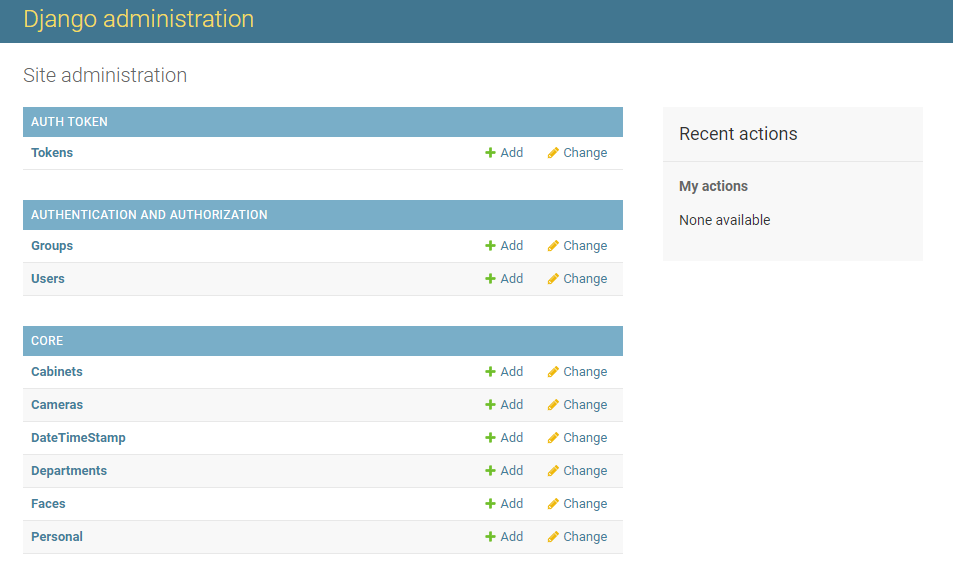


Рисунок 25 – Главная страница администратора

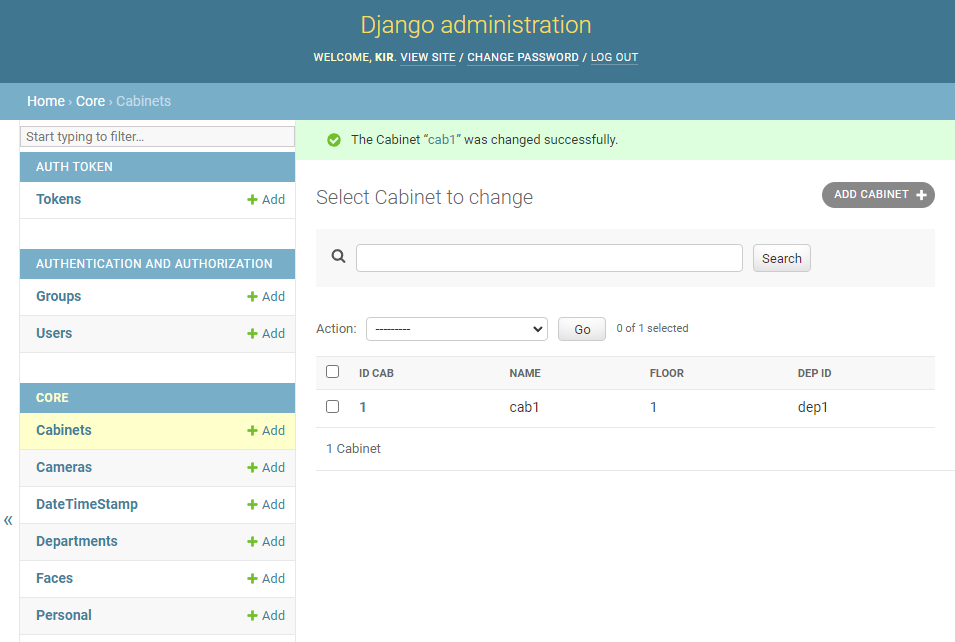


Рисунок 26 – Страница таблицы «Cabinets»

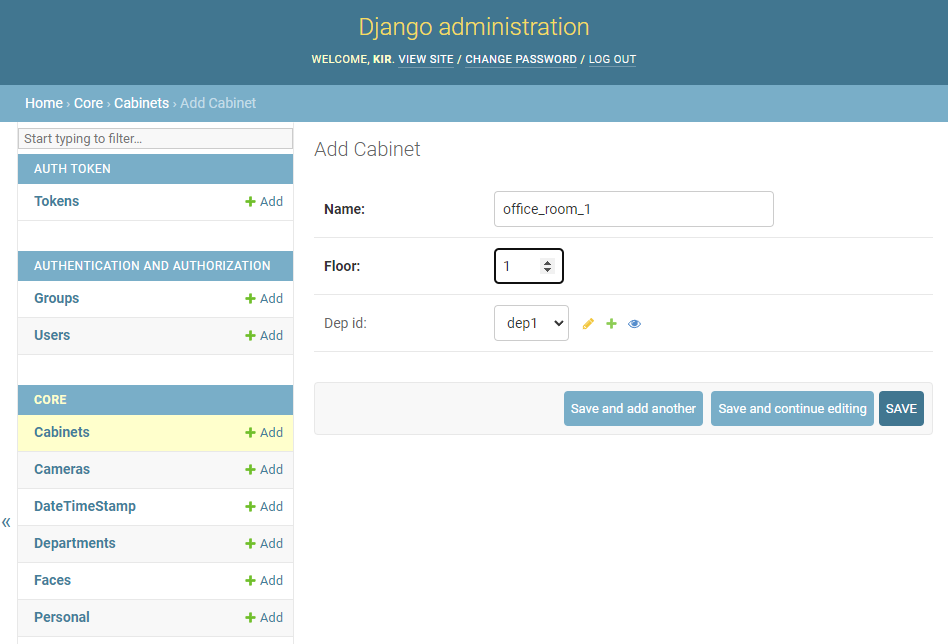


Рисунок 27 – Пример страницы добавления строки в таблицу «Cabinets»

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.